Research Group Data Science - Optimization and Data Analysis
Department of Mathematics
Technical University of Munich

Randomization and Differential Equations in the Context of the Signature

An Introduction to an Efficient Reservoir

Vjekoslav Drvar

Thesis for the attainment of the academic degree

Master of Science (M.Sc.)

at the Department of Mathematics of the Technical University of Munich.

Examiner:
Prof. Dr. Felix Krahmer

Supervisors:
Dr. Hanna Veselovska, Marco Rauscher

Submitted:
Munich, July 12, 2023

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Munich, July 12, 2023 Vjekoslav Drvar [illegible]

Acknowledgements

Finishing university studies is rarely done without a phenomenal support system of people around you, I am no exception to this. I would therefore like to use this page to address this support and express my gratitude for having them in my life.

To my parents — thank you for your guidance and love.

To Marta — thank you for being my number one fan throughout.

To MJ — thank you for always having my back.

To $\pi\mu$ — thank you for all the holiday trips.

To Filip — thank you for the countless coffees and football matches in our dormitory.

Lastly, I would like to thank my mentors.

Prof. Krahmer — thank you for making this thesis possible, especially for additionally introducing Anna and Marco to me as supervisors.

Anna and Marco — thank you for all the brainstorming sessions and second opinions, there would not be a thesis without them.

Abstract

Modern machine learning methods have had a great impact on theories and applications that include function approximation and forecasting. New, neural architectures often offered unprecedented levels of expressive power and have proven themselves of extraordinary use in a variety of applications.

However, there was a price to pay for such improvements in expressivity. It was often the complicated, non-linear cost functions (that one needs to optimize), accompanied with long training times and a lot of only local optima that were a great disadvantage to the hyperparameterized, subsymbolic learning algorithms.

One of the setups where long training times could potentially be quite detrimental is the approximating and forecasting of time dependant data. In this case, the algorithm might be required to be able to continually receive a stream of new input data and adjust its output rules accordingly in a fast manner.

This master thesis provides and demonstrates a technique to tackle such a scenario and circumvent the problem of slow and unreliable training. The central object that we will develop is called the randomized signature.

The recurring theme is finding a low-dimensional feature set for linear regression, which would in turn be easy to optimize (and also to regularize) and could be used as an output rule for a given time-series approximation task. This is accomplished by combining modern, randomized dimensionality-reduction techniques with high-dimensional controlled differential equations that contain deep geometrical properties of the examined time-series (i.e. its continuous counterpart).

Topics include: reservoir computing, signature mapping of a path, controlled differential equations, Johnson-Lindenstrauss lemma, randomized signature of a path, state-affine spaces.

To the best of the author's knowledge, the techniques presented are state-of-the-art for handling streams of irregular, rough times-series. Among other industries, this strategy might be especially useful in finance, where inputs of that sort are ubiquitous.

We anticipate this thesis will be of interest to anyone interested in machine learning for time-series, controlled differential equations and dimensionality reduction strategies. It may be useful as a thorough, step-by-step reference for the emerging and exciting technique presented.

Contents

1 Introduction

1.1 The general setup, motivation and reservoir computing

In this work, as hinted, we will extensively engage in understanding and describing time-dependant structures. More specifically, we will investigate dependencies between trajectories that take place at the same time, but also how we can efficiently encode the geometrical behaviour of some given trajectory. A prominent example of such trajectories one may keep in mind are prices of stocks, bond or commodities in financial markets. Let us specify this more explicitly.

The situation we will assume throughout is that we observe as input a path $X : [0, T] \to \mathbb{R}^d$, i.e. a stream of $\mathbb{R}^d$-valued inputs starting from time 0 up to some predefined and arbitrary time T. We will often refer to X as the *control* and to its components X^i as the *controls* [CBO21]. We will use the subscript notation $X_t = X(t)$ to denote dependence on the parameter $t \in [0, T]$. Furthermore, we will observe another one-dimensional path $Y : [0, T] \to \mathbb{R}$, called the *target*.

As the names suggest, a general goal will be to 'explain' $Y_t,\ t \in [0, T]$ using the values $\{X_s,\ s \in [0, t]\}$, i.e. finding a function f such that

$$Y_t \approx f(X_{0:t}), \tag{1.1}$$

where $X_{0:t}$ signifies the whole stream history of X up to the fixed and arbitrary time point t.

Machine-learning technologies based on universal approximation theorems follow the following approach to this problem: on a sufficiently rich training data set, the map from initial values and control inputs to observed dynamics is learned directly, without building an intermediate model. To learn such input-output maps one often applies recurrent neural networks, LSTMs or neural ODEs, which are flexible enough to approximate all sorts of dynamics, but notoriously difficult to train [CGG+21b].

In order to remedy this, a repeatedly used ansatz will be to define an additional path $Z : [0, T] \to \mathbb{R}^N$, whose evolution depends entirely on the control X. Once we have Z, the idea is that only the current value of Z 'explains' the current value of Y, i.e. we want to find functions g, h such that

$$\begin{aligned} Z_t &= g(X_{0:t}) \\ Y_t &\approx h(Z_t). \end{aligned} \tag{1.2}$$

That way, h needs to be trained to map a vector to another vector, *not* a whole stream of data to a vector. Such a map is often called a *readout map*. This will prove to be quite advantageous later on. The object Z will be dubbed the *reservoir*.

This continuous-time setting will allow us to define and explore many concepts in an elegant analytical manner. However, due to practical and implementational considerations, we will often resort to the paths' discretized counterparts, times series. We therefore additionally consider the *control* $x : \{0 = t_1, ..., t_n = T\} \to \mathbb{R}^d$, the *target* $y : \{0 = t_1, ..., t_n = T\} \to \mathbb{R}$ and the *reservoir* $z : \{0 = t_1, ..., t_n = T\} \to \mathbb{R}^N$, where the domain of each respective time series is the discretized mesh [CLS23] of the interval $[0, T]$ of size n.

Analogously to (1.2), we may look for functions $\tilde{g}$, $\tilde{h}$ such that [CGG+21a]

$$\begin{aligned} z_t &= \tilde{g}(z_{t-1}, x_t) \\ y_t &\approx \tilde{h}(z_t). \end{aligned} \tag{1.3}$$

The ultimate goal of such a setup (generally, not just in this work) is to be able to use the same reservoir regardless of the target. This means that, once we compute a reservoir of a specific control, we just have to train the last output functions (h or $\tilde{h}$) for each target we want to consider that is potentially dependant on the chosen control. This could possibly enable the use of simple and efficient approximating output functions (e.g. linear regression), given that the reservoir is expressive enough.

This learning paradigm is called *reservoir computing* [TYH$^+$19], [LJ09], [VSdS07]. It is a new area that is attracting a lot of attention recently and the authors expect that many advancements are yet to be made in the following months and years. In the following chapter we will introduce our first reservoir: the signature of a path.

1.2 Chapter organization

The main goal of this thesis is to provide a step-by-step guide to the new and exciting technique of the randomized signature and related results. For the sake of having a broad audience and general consequentiality, we will introduce and explain a majority of the concepts needed in the development of the signature theory and dimensionality reduction.

We will start by introducing the main theoretical object of inspiration in this survey: the signature of a path. It will be shown why it is incredibly useful in the task of learning the dynamics of related paths, but also why a first naive approach for machine-learning-like applications would not work sufficiently well or fast.

Therefore, the following chapter will deal with circumventing the problematic complexity of the true signature by the means of randomized dimensionality reduction. Once we will have defined our new, easier-to-compute randomized signature, there will be further open questions about its theoretically rigorous validity, which we will investigate.

All of our theoretical considerations will be supported by numerical simulations, where we will address numerical stability and empirical quality of the technique that is previously introduced in theory. We will close this work by presenting an illustrative use-case of this method in finance: efficient learning of rough paths of stock prices.

We hope this proves insightful to the interested reader!

1.3 Prerequisites

We will assume that the reader has a solid understanding of university level mathematics, i.e. of analysis, linear algebra and probability theory. Additionally, understanding of differential equations and familiarity with general machine learning techniques will be required to fully grasp the content. Some of the prerequisites are revisited in the appendix for convenience.

Beyond these assumptions, concepts will be introduced as we need them. Many of them will be accompanied by further references for an interested reader.

1.4 Simulations and code

The entirety of numerical simulations written for this thesis was written in Python in a web-based interactive computational environment, Jupyter Notebook. The notebooks are publicly available at [Drv23]. The code chunks for simulating Brownian motion were taken from [Tir20] and the library *signatory* from [KL21] was used for efficient computing of signature values.

2 The signature of a path - a reservoir

2.1 Analytical preliminaries

In order to be able to define the signature of a path, we will introduce the concepts that perhaps not every reader is familiar with. For a more thorough guide through these topics, the reader is referred to [CK16] and [Kid21], this section relies on these works.

2.1.1 Smooth paths - elementary building blocks of our theory

As announced, we will deal with paths for the entirety of this thesis. They are the main building blocks of the theory presented. We will therefore introduce them formally at this point.

Definition 2.1 (Path [CK16])**.** A path X in $\mathbb{R}^d$ is a continuous mapping from some interval $[a, b]$ to $\mathbb{R}^d$, written as $X : [a, b] \to \mathbb{R}^d$. One generally uses the notation $X_t = X(t)$ to denote the dependency on the time parameter $t \in [a, b]$.

Remark 2.1. Note that, in contrast to the previous section, we defined the domain interval to be the general $[a, b]$ and not $[0, T]$. This was done to introduce theoretical concepts in a most general setting possible. However, for the sake of convenience regarding simulations and discretizations, we will later switch to $[0, T]$. The reader should not be confused about what interval the path is defined on, since the theoretical concepts work identically regardless of the chosen interval.

Unless otherwise stated, we always assume it for our signature discussion that the paths are smooth. By a smooth path, we mean a path that has derivatives of all orders.

Let us look at two simple examples of smooth paths in $\mathbb{R}^2$ [CK16]:

$$\begin{aligned} &\text{left image: } X_t = \{X_t^1, X_t^2\} = \{t, t^3\},\ t \in [-2, 2] \\ &\text{right image: } X_t = \{X_t^1, X_t^2\} = \{cos(t), sin(t)\},\ t \in [-2, 2] \end{aligned} \tag{2.1}$$

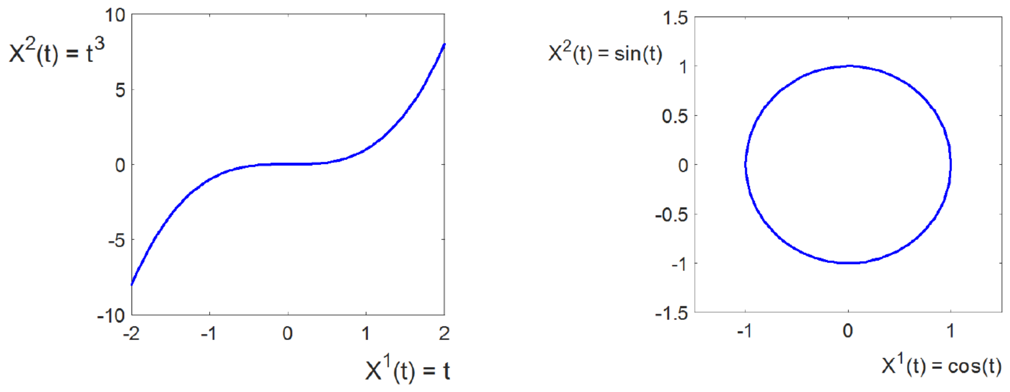


Figure 2.1 Example of two-dimensional smooth paths [CK16].

Such a parametrization can be naturally generalized in $\mathbb{R}^d$ as:

$$X : [a,b] \to \mathbb{R}^d,\ X_t = \{X_t^1, X_t^2, ..., X_t^d\}. \tag{2.2}$$

Another possibility are piecewise linear paths, that can be represented as [CK16]:

$$X_t = \{X_t^1, X_t^2\} = \{t, f(t)\},\ t \in [0,1], \tag{2.3}$$

where f is a piecewise linear function. A situation like that is very common in finance, where $f(t)$ represents a stock price at time t. Such non-smooth paths may represent a continuation of sequential data measurements made on a given mesh of a time interval.

We will investigate such scenarios in more detail in further chapters and see how we can deal with discretized data as well. At this point we are ready to introduce the next tool needed in our theory: the path integral.

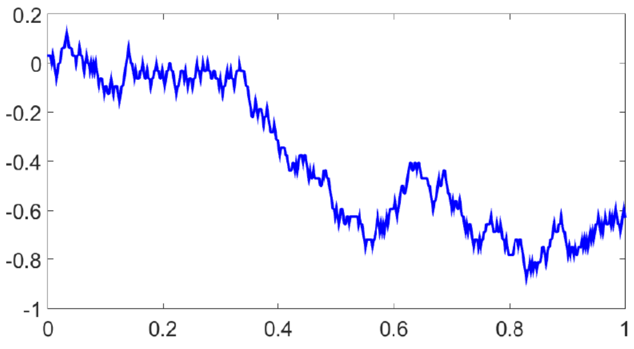


Figure 2.2 Example of non-smooth stochastic path [CK16].

2.1.2 Path integrals

Let us get straight to the definition. Note that we use the 'upper-dot' notation for differentiation with respect to a single variable: $\dot{X}_t = dX_t/dt$ [CK16].

Definition 2.2 (Riemann-Stieltjes path integral against another path [CK16])**.** For two one-dimensional paths $X : [a,b] \to \mathbb{R}$, $Y : [a,b] \to \mathbb{R}$, the path integral of Y against X is defined to be

$$\int_a^b Y_t dX_t = \int_a^b Y_t \dot{X}_t dt. \tag{2.4}$$

Intuitively, one can think of this integration in the following way: instead of multiplying the values of Y_t with the infinitesimal width of the accompanying Riemann columns, we multiply the values of Y_t with infinitesimal increments of X_t. We illustrate this notion with several examples from [CK16].

Example 2.1 ([CK16])**.** In the case of a constant path $Y_t = 1$ for all $t \in [a,b]$, the path integral of Y against any path $X : [a,b] \to \mathbb{R}$ is simply the increment of X:

$$\int_a^b dX_t = \int_a^b \dot{X}_t dt = X_b - X_a. \tag{2.5}$$

Example 2.2 ([CK16])**.** In the case of a path $X_t = t$ for all $t \in [a,b]$, it follows that $\dot{X}_t = 1$ for all $t \in [a,b]$. Therefore, the path integral for any $Y : [a,b] \to \mathbb{R}$ against X is the usual Riemann integral of Y:

$$\int_a^b Y_t dX_t = \int_a^b Y_t dt. \tag{2.6}$$

Example 2.3 ([CK16]). Finally, let us present an example with numerical calculations. Consider the two-dimensional path

$$X_t = \{X_t^1, X_t^2\} = \{t^2, t^3\},\ t \in [0,1]. \tag{2.7}$$

Then, it is possible to compute the path integral

$$\int_0^1 X_t^1 dX_t^2 = \int_0^1 t^2 3t^2 dt = \frac{3}{5}. \tag{2.8}$$

The above example is an example of an iterated integral, which, as discussed in the following section, will be central to the definition of the path signature [CK16].

2.2 The signature of a path

2.2.1 Iterated integrals

Having introduced paths and path integrals, we are now in a position to define the signature of a path. Just like in the previous section, let $X : [a,b] \to \mathbb{R}^d$ be a d-dimensional path, where $X^i : [a,b] \to \mathbb{R}$ are its coordinates, also real-valued paths. Let us first define the following quantity for any $i \in \{1, ..., d\}$ [CK16]

$$S(X)^i_{a,t} := \int_{a<s<t} dX^i_s = X^i_t - X^i_0, \tag{2.9}$$

which is nothing else but the increment of the i-th coordinate of X at the time point $t \in [a,b]$. Note that the function $S(X)^i_{a,\cdot} : [a,b] \to \mathbb{R}$ is likewise a real-valued path that indicates for every $t \in [a,b]$ how far X^i is from its original position.

Also, be aware that the a in the subscript of the function is used only to emphasize the starting time point of the path [CK16]. In some literature it might be dropped, especially if the interval starts with 0. We can go one step further now.

Let us now, for any pair $i, j \in 1, ..., d$, define the *double-iterated* integral [CK16]

$$S(X)^{i,j}_{a,t} := \int_{a<s<t} S(X)^i_{a,s} dX^j_s = \int_{a<r<s<t} dX^i_r dX^j_s, \tag{2.10}$$

whereby the path $S(X)^i_{a,t}$ is taken from (2.9) and the integration limits are as follows:

$$a < r < s < t = \begin{cases} a < r < s \\ a < s < t. \end{cases} \tag{2.11}$$

These integration limits correspond to integration over a triangle, or, generally, over a higher-dimensional simplex.

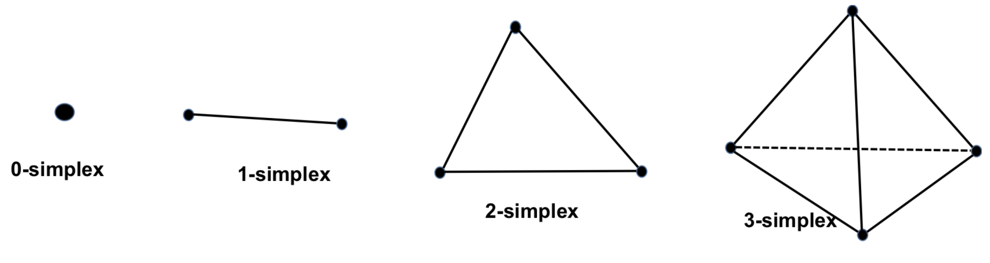


Figure 2.3 Simplices - integration bounds for multi-iterated integrals [YFXJ20].

It is once again important to emphasize that both $S(X)^i_{a,\cdot}$ and $X^j_\cdot$ are paths, so the expression in (2.10) is just another special case of a path integral. Likewise, $S(X)^{i,j}_{a,\cdot} : [a,b] \to \mathbb{R}$ is once again a real-valued path itself. These facts foreshadow how we can continue further on: we can just keep on integrating the multi-iterated integrals against the respective components of the investigated path X.

Let us therefore continue in that tone. For any triple $i, j, k \in \{1, ..., d\}$, we can define the *triple-iterated* integral [CK16]

$$S(X)^{i,j,k}_{a,t} := \int_{a<s<t} S(X)^{i,j}_{a,s} dX^k_s = \int_{a<q<r<s<t} dX^i_q dX^j_r dX^k_s. \tag{2.12}$$

Again, there is nothing conceptually new here, since $S(X)^{i,j}_{a,\cdot}$ and $X^k_\cdot$ are themselves real-valued paths. Therefore the function $S(X)^{i,j,k}_{a,\cdot} : [a, b] \to \mathbb{R}$ is itself a real-valued path.

We can continue doing this recursively, and for any integer $k \geq 1$ and a collection of indices $i_1, ..., i_k \in \{1, ..., d\}$, we define

$$S(X)^{i_1,...,i_k}_{a,t} := \int_{a<s<t} S(X)^{i_1,...,i_{k-1}}_{a,s} dX^{i_k}_s. \tag{2.13}$$

Just as before, $S(X)^{i_1,...,i_{k-1}}_{a,\cdot}$ and $X^{i_k}_\cdot$ are real-valued paths and the function $S(X)^{i_1,...,i_k}_{a,\cdot} : [a, b] \to \mathbb{R}$ is likewise a real-valued path. Equivalently, for better readibility, we can re-express (2.13) as

$$S(X)^{i_1,...,i_k}_{a,t} = \int_{a<t_k<t} ... \int_{a<t_1<t_2} dX^{i_1}_{t_1} ... dX^{i_k}_{t_k}. \tag{2.14}$$

The real number (i.e. the value at the end of the integration) is called the *k-fold iterated integral* [CK16] of X along the indices $i_1, ..., i_k$.

We are now finally in a position to introduce the central object of this chapter, the signature.

2.2.2 The definition of the signature

Definition 2.3 (Signature [CK16])**.** The signature of a path $X : [a, b] \to \mathbb{R}^d$, denoted by $S(X)_{a,b}$, is the collection (infinite series) of all the iterated integrals of X. Formally, $S(X)_{a,b}$ is the sequence of real numbers

$$S(X)_{a,b} := (1, S(X)^1_{a,b}, ..., S(X)^d_{a,b}, S(X)^{1,1}_{a,b}, S(X)^{1,2}_{a,b}, ...), \tag{2.15}$$

where the "zeroth" term, by convention, is equal to 1, and the superscripts run along the set of all *multi-indexes*

$$W = \{(i_1, ..., i_k) | k \geq 1, i_1, ..., i_k \in \{1, ..., d\}\}. \tag{2.16}$$

The set W above is often called the set of *words* on the *alphabet* $A = \{1, ..., d\}$ consisting of d letters.

Example 2.4 ([CK16])**.** Consider an alphabet consisting of three letters only: $\{1, 2, 3\}$. There is an infinite number of words which could be composed from this alphabet, namely:

$$\{1, 2, 3\} \to (1, 2, 3, 11, 12, 13, 21, 22, 23, 31, 32, 33, 111, 112, 113, 121, ...). \tag{2.17}$$

As already hinted (when discussing the arbitrary choice of the domain interval), an important property of the signature is that the signature of a path X are independent of the starting point of X. This is immediately visible in the very definition of the respective iterated integrals, since only increments of the path components themselves enter the integration.

We will often consider the k-th level of the signature [CK16], which is defined as the finite collection of all the terms $S(X)^{i_1,...,i_k}_{a,b}$ where the multi-index is of length k. For example, the first level of the signature of the path $X : [a, b] \to \mathbb{R}^d$ is the collection of d real numbers $S(X)^1_{a,b}, ..., S(X)^d_{a,b}$, and the second level is the collection of d^2 real numbers

$$S(X)^{1,1}_{a,b}, \; ... \; , S(X)^{1,d}_{a,b}, S(X)^{2,1}_{a,b}, \; ... \; , S(X)^{2,d}_{a,b}. \tag{2.18}$$

Readers might at this point notice that an l-th level ($l \in \mathbb{N}$) of the signature resides precisely in the l-th tensor space of $\mathbb{R}^d$, i.e. in $(\mathbb{R}^d)^{\otimes l}$. This in turn implies that the whole signature of a path resides in the infinitely-dimensional tensor product space $\prod_{h=0}^{\infty}(\mathbb{R}^d)^{\otimes h}$.

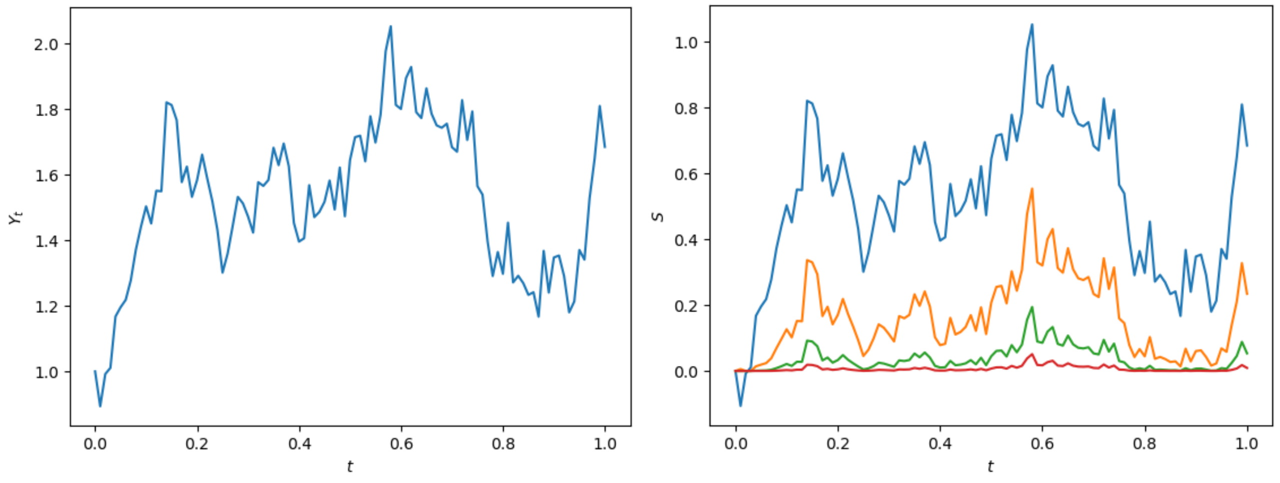
In case the reader is unfamiliar with tensors, there is a short recapitulation on this topic in the Appendix Chapter A. This is however a good place to formally introduce the set the signature is an element of.

Once we have established that the signature is an element of $\prod_{h=0}^{\infty}(\mathbb{R}^d)^{\otimes h}$, it is important to understand that the signature is actually also a path with values in this space, namely of the form

$$S_{a,\cdot}(X) : [a,b] \to \prod_{h=0}^{\infty}(\mathbb{R}^d)^{\otimes h}, \quad S_{a,t}(X) = (S(X)^{1,1}_{a,t}, \dots, S(X)^{1,d}_{a,t}, S(X)^{2,1}_{a,t}, \dots, S(X)^{2,d}_{a,t}, \dots). \tag{2.19}$$

The value we have defined previously was the signature of this path at the time point b. In the context of the general setup from Section 1.1, the path X will be the *control*, the signature will turn out to be a(n) (arguably) good *reservoir*. The final value (at the time point b) of the signature will be expressive for the explanation of some *target* at the time point b.

We are however interested in the complete evolution of the signature, since we aim to approximate the *target* at all time points, not just the final one observed. We can observe such an evolution in Figure 2.4, both a path and its signature have values in all time points $t \in [a,b]$ (in this figure the domain interval is $[0,1]$).



(a) A sample one-dimensional path. **(b)** Its first 4 signature values.

Figure 2.4 An example of a side-by-side evolution of a path and its accompanying signature (which is also a multi-dimensional path) [KL21] [Drv23].

Remark 2.2. Do note that the one-dimensional example in Figure 2.4 does not really illustrate the expressive power the signature possesses. It primarily serves the purpose of demonstrating how the signature also evolves with time, i.e. that the signature itself is a (multidimensional) path. Things get much more interesting once we assume a multidimensional control.

2.2.3 Examples of signatures

Let us look at a few examples [CK16] to amplify our imagination about this newly introduced concept.

Example 2.5 ([CK16])**.** The simplest example of a signature which one should keep in mind is that of a one-dimensional path. In this case our set of indexes (or alphabet) is of size one, $A = \{1\}$, and the set of multi-indexes (or words) is $W = \{(1, \dots, 1) | k \geq 1\}$, where 1 appears k times in $(1, \dots, 1)$.

Let us consider a concrete path now, $X : [a,b] \to \mathbb{R},\ X_t = t$. We can immediately verify that the signature of X is given by

$$
\begin{aligned}
S(X)^1_{a,b} &= X_b - X_a, \\
S(X)^{1,1}_{a,b} &= \frac{(X_b - X_a)^2}{2!}, \\
S(X)^{1,1,1}_{a,b} &= \frac{(X_b - X_a)^3}{3!}, \\
&\dots
\end{aligned}
\tag{2.20}
$$

We can in fact show that the above expression of the signature remains true for any path $X : [a,b] \to \mathbb{R}$. Hence, for one-dimensional paths, the signature depends only on the increment $X_b - X_a$, it always has the above form.

One observes precisely this case in Figure 2.4, later signature values mostly look like scaled versions of the first entry.

Example 2.6 ([CK16])**.** We present now a more involved example of the signature for a two-dimensional path. Our set of indexes is now $A = \{1, 2\}$, and the set of multi-indexes is

$$W = \{(i_1, ..., i_k) | k \geq 1,\ i_1, ..., i_k \in \{1, 2\}\}, \tag{2.21}$$

the collection of all finite sequences of 1's and 2's. Consider a parabolic path in $\mathbb{R}^2$ as shown in Figure 2.5, parameterized by

$$X_t = \{X^1_t, X^2_t\} = \{3 + t, (3 + t)^2\},\ t \in [0, 5],\ (a = 0, b = 5), \tag{2.22}$$

which implies

$$dX_t = \{dX^1_t, dX^2_t\} = \{dt, 2(3 + t)dt\}. \tag{2.23}$$

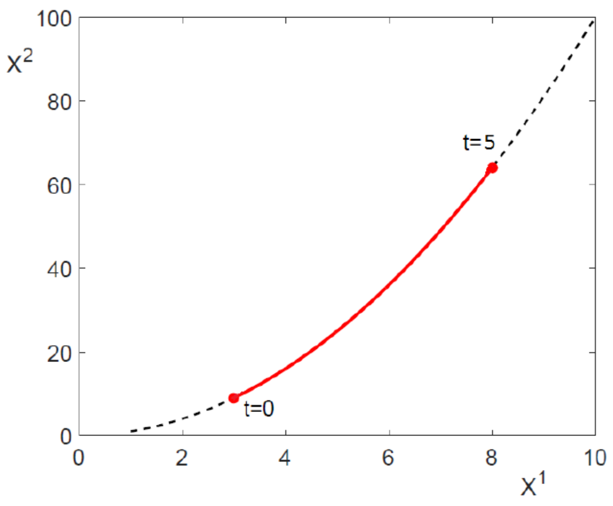


Figure 2.5 A two-dimensional path parameterized by (2.5) [CK16].

A straightforward computation gives:

$$S(X)^1_{0,5} = \int_{0<t<5} dX^1_t = \int_0^5 dt = X^1_5 - X^1_0 = 5, \tag{2.24}$$
$$S(X)^2_{0,5} = \int_{0<t<5} dX^2_t = \int_0^5 2(3+t)dt = X^2_5 - X^2_0 = 55,$$
$$S(X)^{1,1}_{0,5} = \iint_{0<t_1<t_2<5} dX^1_{t_1} dX^1_{t_2} = \int_0^5 \left[\int_0^{t_2} dt_1\right] dt_2 = \frac{25}{2},$$
$$S(X)^{1,2}_{0,5} = \iint_{0<t_1<t_2<5} dX^1_{t_1} dX^2_{t_2} = \int_0^5 \left[\int_0^{t_2} dt_1\right] 2(3+t_2)dt_2 = \frac{475}{3},$$
$$S(X)^{2,1}_{0,5} = \iint_{0<t_1<t_2<5} dX^2_{t_1} dX^1_{t_2} = \int_0^5 \left[\int_0^{t_2} 2(3+t_1)dt_1\right] dt_2 = \frac{350}{3},$$
$$S(X)^{2,2}_{0,5} = \iint_{0<t_1<t_2<5} dX^2_{t_1} dX^2_{t_2} = \int_0^5 \left[\int_0^{t_2} 2(3+t_1)dt_1\right] 2(3+t_2)dt_2 = \frac{3025}{2},$$
$$S(X)^{1,1,1}_{0,5} = \iiint_{0<t_1<t_2<t_3<5} dX^1_{t_1} dX^1_{t_2} dX^1_{t_3} = \int_0^5 \left[\int_0^{t_3} \left[\int_0^{t_2} \left[\int_0^{t_1} dt_1\right] dt_2\right] dt_3\right] = \frac{125}{6}, \tag{2.25}$$
$$...$$

Continuing this way, we can compute every term $S(X)^{i_1,...,i_k}_{0,5}$ of the signature for every multi-index $(i_1, ..., i_k),\ i_1, ..., i_k \in \{1, 2\}$. We highlight once again that this computation gives us the signature value of the path at time point b, given that we started accumulating it at time point a.

2.2.4 Geometric intuition of the signature

Let us now briefly turn our attention to the geometric intuition of the signature, more specifically, of its first two levels.

As already introduced and discussed, the first level, given by the terms $(S(X)^1_{a,b}, ..., S(X)^d_{a,b})$, is simply the increment of the path $X : [a, b] \to \mathbb{R}^d$. This is the most basic information the signature can give us, how much distance each coordinate has crossed. For the second level, note that the term $S(X)^{i,i}_{a,b}$ is always equal to $\frac{(X^i_b - X^i_a)^2}{2}$. So, expressions like these do not really give us new information about the path, since they are always squared and scaled versions of some element in the first level.

However, things get more interesting when we look at the cross elements in the second level. To give meaning to the term $S(X)^{i,j}_{a,b}$ for $i \neq j$, consider the Lévy area [CK16] (depicted in Figure 2.6), which is a signed area enclosed by the path (solid red line) and the chord (blue straight dashed line) connecting the endpoints. The Lévy area of the two dimensional path $\{X^1_t, X^2_t\}$ is given by:

$$A = \frac{1}{2}(S(X)^{1,2}_{a,b} - S(X)^{2,1}_{a,b}). \tag{2.26}$$

The signed areas denoted by A_- and A_+ are the negative and positive areas respectively, and ΔX_1 and ΔX_2 represent the increments along each coordinate [CK16].

The signature captures deep geometric and analytical properties of a path. As we observe higher levels of the signature, the meaning of its values becomes more and more abstract but nevertheless astonishingly expressive about the path itself.

Although the signature is not a bijection in general, there is strong regularity in this map. Paths with the same signature are strongly related to each other. For more information on this, consider [CK16].

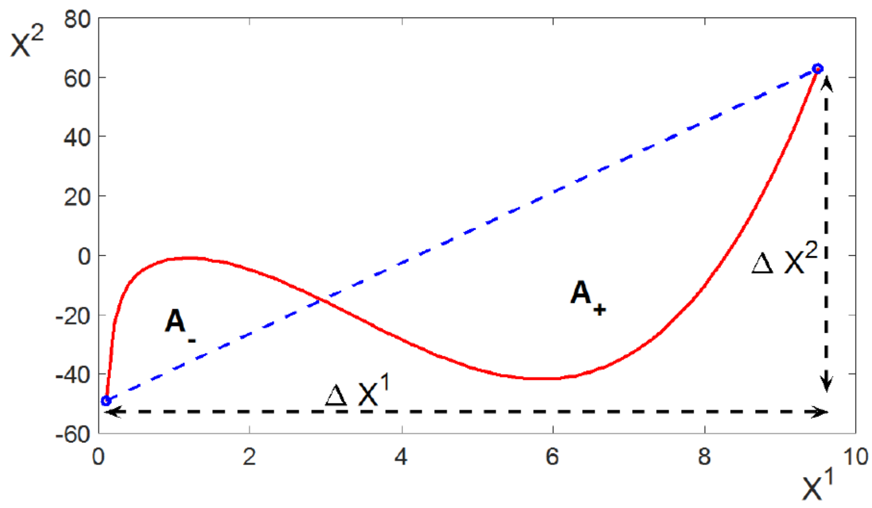


Figure 2.6 Example of signed Lévy area of a curve. Areas above and under the chord connecting two endpoints are negative and positive, respectively [CK16].

2.3 Controlled differential equations (CDEs)

We will use the following section to introduce an additional concept that will enable us to present certain reformulations and more refined results related to the signature of a path. Readers who are comfortable with controlled differential equations are welcome to skip to Section 2.4.

For a reader in need of a brush up on CDEs, however, it may be beneficial to at least glance through the following pages. These tools will be more or less ubiquitous in later concepts.

Definition and an example

Another concept not all readers might be familiar with are controlled differential equations (CDEs). Both the signature and later reservoirs can be elegantly expressed using CDEs. It is therefore important to have a working knowledge of this topic. We will base this short introduction on [Kid21] and [KMFL20], which are good references for an interested reader.

Let us briefly recover a general form of an ordinary differential equation (ODE)

$$y(0) = y_0, \quad \frac{dy}{dt}(t) = f(t, y(t)), \tag{2.27}$$

or equivalently

$$y(0) = y_0, \quad y = y_0 + \int_0^t f(s, y(s))ds, \tag{2.28}$$

where $y_0 \in \mathbb{R}^n$ is an initial condition, $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is a function that describes the dynamics of the ODE and $y : [0, T] \to \mathbb{R}^n$ is the solution.

Example 2.7. Consider the example of a simple pendulum. In this setting, y_0 may be the initial angle a pendulum is left to swing from, f describes the mechanics of the swinging and y describes the whole trajectory of the swinging, inside the observed time interval (as depicted in Figure 2.7).

If we look once more at (2.28), we notice that an extra time-like dimension is introduced and then integrated over. The presence of this extra (artificial) dimension motivates us to consider whether this model can be extended to data already exhibiting sequential structure, such as time series [Kid21].

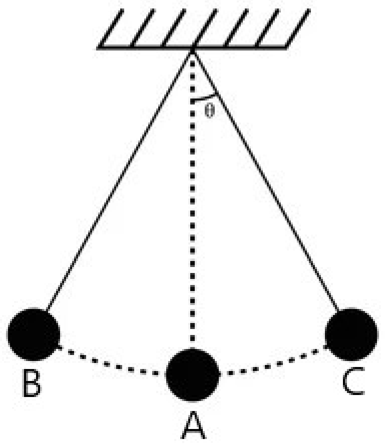


Figure 2.7 A simple pendulum - A, B, C are examples of possible choices for y_0 [Una22].

Given some ordered data $(y_0, ..., y_n)$, the goal is to extend the $y(0) = y_0$ condition to one resembling '$y(0) = y_0, ..., y(n) = y_n$', to align the introduced time-like dimension with the natural ordering of the data. The key difficulty is that the solution of an ODE is determined by the initial condition at $y(0)$, so there is no direct mechanism for incorporating data that arrives later [Kid21].

Fortunately, it turns out that the resolution of this issue how to incorporate incoming information into a differential equation is already a well-studied problem in mathematics, via *controlled differential equations*.

Definition 2.4 (Controlled differential equation [Kid21]). Let $T > 0$ and let $d_x, d_y \in \mathbb{N}$. Let $x : [0, T] \to \mathbb{R}^{d_x}$ be a continuous function of bounded variation. Let $f : \mathbb{R}^{d_y} \to \mathbb{R}^{d_y \times d_x}$ be Lipschitz continuous. Let $y_0 \in \mathbb{R}^{d_y}$. A continuous path $y : [0, T] \to \mathbb{R}^{d_y}$ is said to solve a controlled differential equation, controlled or driven by x, if

$$y(0) = y_0, \quad y(t) = y(0) + \int_0^t f(y(s))dx(s) \text{ for } t \in (0, T]. \tag{2.29}$$

Here '$dx(s)$' denotes a Riemann-Stieltjes integral, and '$f(y(s))dx(s)$' refers to a matrix-vector multiplication.

Suppose x is differentiable and has bounded derivative - a relatively weak assumption. Then x will be of bounded variation, and the Riemann-Stieltjes integral may be reduced to an ordinary integral

$$\int_0^t f(y(s))dx(s) = \int_0^t f(y(s))\frac{dx}{ds}(s)ds. \tag{2.30}$$

The reader unfamiliar with these concepts should feel free to mentally substitute the above treatment throughout [Kid21].

Remark 2.3. Equation (2.30) is essentially about reducing a CDE to an ODE. Correspondingly, the term 'vector field' may be used to refer to either $f(y(s))$ or $f(y(s))\frac{dx}{ds}(s)$ [Kid21].

CDEs are operators - controlled differential equation should be interpreted as a function from path-space to path-space. The input is a path x. The output is a path y. By choosing f carefully, we may use a CDE to compute specific functions of its control [Kid21]. This idea might ring a bell in the context of this work: we observe a *control* (input) stream and calculate the *reservoir* stream from it (sometimes called *hidden state*).

Remark 2.4. Note that the choice of function names is not fully in line with Section 1.1. This was done make this subsection more compatible with the sources, in case the reader wants to further investigate this topic. The path x plays the role of X (control) in our setup, the y plays the role of Z (reservoir).

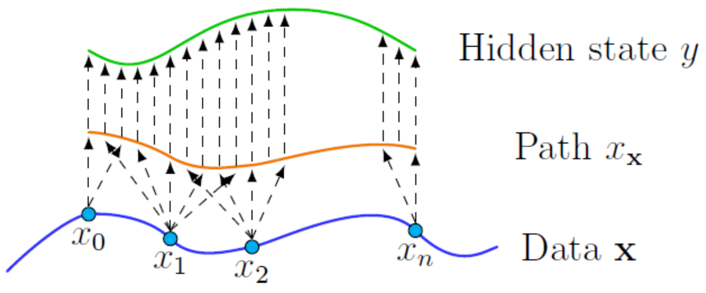


Figure 2.8 The hidden state of the CDE evolves continuously, driven by observational data [Kid21].

Remark 2.5. During implementation, there is the problem that we mostly observe sequential, discretized data, but the tools presented here work in continuous time. We overcome this issue either by interpolating the data to acquire smooth functions, or by reducing the differential equations to difference equations (finitesimal increments).

Let us further illustrate this concept with a simple example from [Kid21].

Example 2.8 (Value and integral of control [Kid21])**.** Let $f : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$ be defined by

$$f(y) = \begin{bmatrix} 0 & 1 \\ y_1 & 0 \end{bmatrix}, \tag{2.31}$$

where $y \in \mathbb{R}^2$ is decomposed into $y = [y_1, y_2]$.

Given any control $x : [0,T] \to \mathbb{R}$, let $y : [0,T] \to \mathbb{R}^2$ be the solution of the CDE driven by $t \mapsto (t, x(t))$, with vector field f, with initial condition $y(0) = [x(0), 0] \in \mathbb{R}^2$. Then $y(t)$ will compute the value, and the first integral, of x.

For example, consider the input signal $x(t) = sin(t) \in \mathbb{R}$.

Then

$$\begin{aligned} y(t) &= y(0) + \int_0^t f(y(s))d \begin{bmatrix} s \\ x(s) \end{bmatrix} \\ &= \int_0^t \begin{bmatrix} 0 & 1 \\ y_1(s) & 0 \end{bmatrix} \begin{bmatrix} 1 \\ cos(s) \end{bmatrix} ds \\ &= \int_0^t \begin{bmatrix} cos(s) \\ y_1(s) \end{bmatrix} ds \end{aligned} \tag{2.32}$$

Solving the first component, we see that

$$y_1(t) = \int_0^t cos(s) ds = sin(t) \tag{2.33}$$

and so

$$y_2(t) = \int_0^t y_1(s)ds = \int_0^t sin(s)ds. \tag{2.34}$$

As advertised: $y(t)$ computes both the value $sin(t)$ of the input signal, and its first integral $\int_0^t sin(s)ds$.

Moreover, there was nothing special about the choice of $sin(t)$, and this CDE will compute the value and first integral of any input signal. This illustrates how we can obtain useful paths dependant of the control by carefully choosing a vector field. It is precisely what we will do to express the signature as a solution of a CDE.

Remark 2.6. The differential equation for a CDE is (by convention) autonomous, in the sense that f is independent of the time t. If really desired then t may be included by adding it to the state: replace x with $[t, x]$ and f with $\begin{bmatrix} 1 & 0 \\ 0 & f \end{bmatrix}$. This implies we have replaced y with $[t, y]$ [Kid21].

Control theory [Kid21]

Despite their similar names, and treatment of similar problems, controlled differential equations and control theory are typically treated as separate fields.

The difference is to some extent philosophical. In control theory, the system f is typically specified, and the task is to find a control x producing the desired response y. Meanwhile with CDEs, this is flipped around: the control x is typically specified, and we attempt to find a system f that produces a desired response y.

The latter is the case we will find ourselves in throughout the remainder of this work.

2.4 Further sophistication and results

With some new tools and concepts at our disposal, it is now possible to express many ideas we have introduced so far (including the general setup and signature itself) in a much more elegant way. Much of this section is based on [CGG$^+$21b].

2.4.1 A reformulation of the setup using differential equations

Let us take a step back and look once again at the general goal of this work, described in Section 1.1. We are interested in the quantitative understanding of dynamics in terms of initial values, local characteristics and given controls.

To accomplish this, we will make use of the paradigm of reservoir computing: split the input-output map into a generic part (the reservoir), which is not or only in small parts trained, and a readout part, which is accurately trained and often linear. In many cases, learning the readout layer is often a simple (regularized) regression [CGG$^+$21b]. In this work we will always assume a linear regression (possibly with regularization) for the readout part.

Let us now reformulate the setup that we described with our new tools. Consider a controlled differential equation (CDE)

$$dY_t = \sum_{i=0}^{d} V_i(Y_t)dX_t^i, \; Y_0 = y_0 \in \mathbb{R}^m \tag{2.35}$$

for some smooth vector fields $V_i : \mathbb{R}^m \to \mathbb{R}^m, i = 0, ..., d$ and d smooth control curves X^i.

Remark 2.7. Note that, in this general setting, we allowed the target Y to be multidimensional. One can however quickly recover a formulation analogous to the one in Section 1.1 by considering the respective components of Y separately, as long as we assume a linear readout.

Remark 2.8. Also, note that this setup is precisely the first part of what we described in Section 1.1, now described using a controlled differential equation. We observe trajectories X and Y and assume Y is dependant on X in some way. The vector fields V_i describe this dependency, earlier this was just some function f.

We place ourselves in the situation that we observe X and Y, but do not have access to the vector fields V_i. The goal is to learn the dynamics and to simulate from it conditional on the controls X.
That it, we aim to describe the map

$$\text{(input control } X) \mapsto \text{(solution trajectory } Y),$$

which is obviously quite often a complicated, non-linear map of delicate regularity [CGG+21b].
Indeed, as anticipated, we can split the map $(X_t)_{0\leq t\leq T} \mapsto (Y_t)_{0\leq t\leq T}$ into two parts:

- An $\mathbb{R}^N$-valued system (a reservoir) with d generators (drivers)

$$dZ_t = \sum_{i=0}^{d} U_i(Z_t)dX_t^i,\ Z_0 = z_0 \in \mathbb{R}^N, \tag{2.36}$$

  where the U_i's are purposefully chosen vector fields designed to induce a good basis for regression.

- A linear map $Z \mapsto WZ$ which is actually trained (e.g. a linear regression) and which is chosen to explain $(Y_t)_{0\leq t\leq T}$ as well as possible (the readout map).

Remark 2.9. Once again, note the analogy to the construction in Section 1.1. We design another trajectory Z that depends on X and is then used to explain Y. Now, we are just able to express this evolution using CDEs. The vector fields U_i explain the dependency between Z and X, earlier this was just some function h.

As already hinted, the first approach we are going to consider will simply be to use the signature of X as a reservoir. In other words, the signature will play a role of Z in the construction above. For this to work, we need a CDE (as in (2.36)) that describes the evolution of the signature. Such an evolution is portrayed in Figure 2.4, for instance.

2.4.2 The signature-inducing CDE

So far we have established that the signature $S(X)$ of an $\mathbb{R}^d$-valued path X is likewise a path with values in $\prod_{h=0}^{\infty}(\mathbb{R}^d)^{\otimes h}$ that captures deep geometric and analytical properties of X. Also, we have introduced CDEs and explained how they are used to induce new paths dependant on their controls.

With this, we are now ready to present an advanced result, allowing us to describe the dynamics of the signature $S(X)$ in terms of a CDE controlled by X.

Theorem 2.1 (The signature-inducing CDE [CGG+21b]). *Let X be a smooth control, denote by $\{e_1, ..., e_d\}$ the canonical basis of $\mathbb{R}^d$. Then the controlled differential equation in the space $\prod_{h=0}^{\infty}(\mathbb{R}^d)^{\otimes h}$*

$$dS_{s,t}(a, X) = \sum_{i=1}^{d} S_{s,t}(a, X) \otimes e_i dX_t^i,\ S_{s,s}(a, X) = a \tag{2.37}$$

has a unique smooth evolution operator, precisely the signature of X and denoted by $S(X)$, given by

$$S_{s,t}(a, X) = a \sum_{k=0}^{\infty} \sum_{i1,...,i_k=1}^{d} \int_{s\leq t_1\leq ...\leq t_k\leq t} dX_{t_1}^{i_1} ... dX_{t_k}^{i_k} e_{i_1} \otimes ... \otimes e_{i_k}. \tag{2.38}$$

Remark 2.10. If we anchor the starting value with $\mathbf{1} := (1,0,0,...) \in \prod_{h=0}^{\infty}(\mathbb{R}^d)^{\otimes h}$, we can drop the dependency in a, leaving us with the original $S_{s,t}(X)$.

Let us briefly comment on the structure of the above equations.

In (2.37) we have a CDE driven by X with the accompanying initial value, as announced and in line with the notion of CDEs. However, the vector field is somewhat peculiar. Firstly, note that by applying $\otimes\, e_i$ to an element in $\prod_{h=0}^{\infty}(\mathbb{R}^d)^{\otimes h}$ results in an element from that same space (just without the first trivial entry), due to its infinite-dimensional structure. Secondly, note how this tensor product is used to project the 'current' values of the signature one tensor level further to define the signature's growth over the time interval.

The expression in (2.38) is merely a compact reformulation of the definition in (2.15). Notice here the use of multiple tensor products to project the right quantities to the right coordinates in the right tensor levels.

Although the formal proof requires more tools beyond the scope of this work, the CDE in (2.37) quite intuitive, once given a closer look. We address this intuition with a simple discretized example.

Example 2.9. (A discretized illustration) Let $X : [0,1] \to \mathbb{R}^2$ be a two-dimensional control and $x : \{0 = t_1, ..., t_n = T\} \to \mathbb{R}^2$ its discretized version in which the time steps are 0.2. If we were interested in the evolution of true signature $S(X)$ of X, we would numerically calculate the formula given in (2.38) for each entry and for each tensor level.

Let us, however, in order to facilitate a beginner's intuition, consider 'the discretized signature' of X, $\tilde{S}(x)$. We will only informally define this object, as the signature which does not accumulate its values continuously, but rather only at the discretization time points. This makes the integration boil down to summation. Denote with $\tilde{S}_{t_1,t_2}$ the value of $\tilde{S}$ accumulated from t_1 to t_2. Additionally, denote by $\Delta x^i_{t_1,t_2}$ the difference in the i-th component between two respective time points. We will focus on the first two tensor levels, the rest turns out to be analogous. Let us mention once more explicitly the real formulas for the signature on the first two levels, in order to be able to compare:

$$S(X)^i_{a,t} = \int_{a<s<t} dX^i_s = X^i_t - X^i_0, \quad S(X)^{i,j}_{a,t} = \int_{a<s<t} S(X)^i_{a,s} dX^j_s = \int_{a<r<s<t} dX^i_r dX^j_s. \tag{2.39}$$

At $t = 0$ we have

$$\tilde{S}_{0,0}(x) = \left(1 \left| 0 \quad 0 \right| \begin{vmatrix} 0 & 0 \\ 0 & 0 \end{vmatrix} \cdots \right), \tag{2.40}$$

which is just the initial condition of the CDE.

At $t = 0.2$ we have

$$\tilde{S}_{0,0.2}(x) = \left(1 \left| \Delta x^1_{0,0.2} \quad \Delta x^2_{0,0.2} \right| \begin{vmatrix} 0 & 0 \\ 0 & 0 \end{vmatrix} \cdots \right). \tag{2.41}$$

Note how this illustrates why the convention for the first constant entry is needed. We calculate the increments of the respective components, accordance with the first part of (2.39). Then the first tensor part of the vector field $1 \otimes e_i$ projects these increments into the first non-trivial tensor level.

At $t = 0.4$ we have

$$\tilde{S}_{0,0.4}(x) = \left(1 \left| \Delta x^1_{0,0.4} \quad \Delta x^2_{0,0.4} \right| \begin{vmatrix} \Delta x^1_{0,0.2} \cdot \Delta x^1_{0.2,0.4} & \Delta x^1_{0,0.2} \cdot \Delta x^2_{0.2,0.4} \\ \Delta x^2_{0,0.2} \cdot \Delta x^1_{0.2,0.4} & \Delta x^2_{0,0.2} \cdot \Delta x^2_{0.2,0.4} \end{vmatrix} \cdots \right). \tag{2.42}$$

The increments of the paths were further accumulated, just as in the previous time step. However, since the first non-trivial tensor level was not empty anymore, the vector field including $\otimes\, e_i$ acted on the first level and it projected the first terms one level deeper, i.e. into the second non-trivial tensor level.

At $t = 0.6$ we have

$$\tilde{S}_{0.4,0.6}(x) = \tilde{S}_{0,0.6}(x) - \tilde{S}_{0,0.4}(x) = \left(0 \left| \Delta x^1_{0.4,0.6} \quad x^2_{0.4,0.6} \right| \left| \begin{matrix} \Delta x^1_{0,0.4} \cdot \Delta x^1_{0.4,0.6} & \Delta x^1_{0,0.4} \cdot \Delta x^2_{0.4,0.6} \\ \Delta x^2_{0,0.4} \cdot \Delta x^1_{0.4,0.6} & \Delta x^2_{0,0.4} \cdot \Delta x^2_{0.4,0.6} \end{matrix} \right| \ldots \right). \tag{2.43}$$

This continues further analogously. Note how we actually expressed the increment in the signature this time to ease the notation, reflecting the path structure of the signature itself.

2.4.3 The pros and cons of the signature as a reservoir

We are now ready to lay down our first complete reservoir computing (RC) pipeline. Borrowing the structure from Section 2.4.1, we can now design the first naive attempt to learn the dynamics of some path in the following way.

Firstly, we can split map $(X_t)_{0\leq t\leq T} \mapsto (Y_t)_{0\leq t\leq T}$ into two parts:

- A $\prod_{h=0}^{\infty}(\mathbb{R}^d)^{\otimes h}$-valued system (a reservoir) with d generators (drivers)

$$dS_{s,t}(X) = \sum_{i=1}^{d} S_{s,t}(X) \otimes e_i dX_t^i,\ S_{s,s}(X) = \mathbf{1}, \tag{2.44}$$

  precisely the signature-inducing CDE we introduced.

- A linear map $S \mapsto WS$ which is actually trained (e.g. a linear regression) and which is chosen to explain $(Y_t)_{0\leq t\leq T}$ as well as possible (the readout map).

When considering actual implementation, we obviously cannot handle infinite-dimensional objects, such as the signature, in a direct way. We therefore often resort to their truncated, finite-dimensional versions. We would hence actually consider the *truncated signature*, defined in the following.

Definition 2.5 (Truncated signature [CBO21]). We define the truncated signature of X of order $M \geq 0$, as the object containing its first M tensor levels, namely

$$S^M_{0,t}(X) := (1, \mathcal{S}^1_{0,t}(X), ..., \mathcal{S}^M_{0,t}(X)) \in \prod_{h=0}^{M} (\mathbb{R}^d)^{\otimes h} := \mathcal{T}^M(\mathbb{R}^d), \tag{2.45}$$

where $\mathcal{S}^l_{0,t}(X),\ l \in \{1, ..., M\}$ stands for the l-th signature level, residing in $(\mathbb{R}^d)^{\otimes l} = \mathbb{R}^d \otimes ... \otimes \mathbb{R}^d$ (l times).

Remark 2.11. For a fairly large M, this truncation should not largely impede the expressivity of signature. This is due to the fact that, the 'deeper' the entries of the signature are, the more abstract their interpretation and additional value are. Generally, the first few levels of the signature carry the most information about their accompanying path.

With this we have theoretically described one RC system. There are two important aspects that still need to be addressed: how well the system can approximate (relatively) arbitrary targets, and how efficiently or fast the system works.

We will start with a theorem addressing the former.

Expressivity

Theorem 2.2 (Signature is a reservoir [CBO21]). *Let $V_i : R^m \to R^m, i = 1, ..., d$ be vector fields regular enough such that $dY_t = \sum_{i=1}^{d} V_i(Y_t)dX_t^i, Y_0 = y_0 \in \mathbb{R}^m$, admits a unique solution $Y_t : [0,T] \to \mathbb{R}^m$. Then, for any smooth test function $F : \mathbb{R}^m \to \mathbb{R}$ and for every $M \geq 0$ there is a time-homogeneous linear operator $W : \mathcal{T}^M(\mathbb{R}^d) \to \mathbb{R}$ which depends only on $(V_1, ..., V_d, F, M, y)$ such that $F(Y_t) = W(S^M_{0,t}(X)) + \mathcal{O}(t^{M+1})$, and $t \in [0,T]$.*

This theorem suggests the first M entries of the signature of X are sufficient to linearly explain the solution of any differential equation driven by it [CBO21]. It guarantees astonishing expressivity, particularly in the RC context described in this subsection.

This goes in line nicely with the characteristics of the signature, namely that it captures deep geometric and analytical properties of the path. It is still quite remarkable that this collection of features can be combined *only linearly* to explain another (relatively) arbitrary trajectory driven by it.

Efficiency

Unfortunately, the other RC aspect to be considered turns out to be quite a bottleneck in this first approach. Calculating $S^M_{0,t}(X)$ requires the calculation of $\frac{d^{M+1}-1}{d-1}$ iterated integrals – which in total quickly becomes computationally expensive [CBO21]. This is actually in sharp contrast to reservoir computing at work, where reservoirs are easy to evaluate.

Several techniques have been developed to circumvent this problem, in particular kernelization techniques, see e.g. [KL21]. Alternatively, one can try to compress the information of the signature through a random projection

$$\pi : \mathcal{T}^M(\mathbb{R}^d) \to \mathbb{R}^k, \tag{2.46}$$

whose image can be surprisingly approximated by a dynamical system on $\mathbb{R}^k$, which has random characteristics.

This is due to the fact that the characteristics of the signature process have a nilpotent structure, whence its random projections look by the central limit theorems almost like random matrices. The random projections will be chosen according to the Johnson-Lindenstrauss lemma [CGG$^+$21b], see Section 3.1.3.

In the next chapter we will thoroughly investigate the latter ansatz with randomization and introduce all the necessary tools to make it happen. Thereafter, we will present simulated empirical results and comment how the introduced theoretical tools hold up in practice.

3 Randomized signature and the Johnson-Lindenstrauss lemma

So far, we have seen that the signature of path is a sophisticated infinite-dimensional object that captures a lot of useful geometrical properties of the path. Once at our disposition, a simple linear combination of its entries can approximate trajectories related to the initial one.

However, directly obtaining these entries turns out to be quite resource consuming task, with different numerical complications arising during the computation. Consequently, there have been efforts to circumvent this issue, either by efficient approximation of the signature, or by some lower-dimensional representation.

In this chapter, we will investigate recent efforts to use randomized dimensionality reduction to obtain projections of the signature that still retain much of its original expressivity (in the context of reservoir computing). We will base these investigations primarily on [CBO21], [CGG$^+$21b], [CGG$^+$21a].

First, we will revise the concept of dimensionality reduction and present a modern, randomized strategy to achieve it. The primary sources used are [Gü21], [Kra22] and [BN06]. A vital result that will be employed in later developments is the so called *Johnson-Lindenstrauss lemma*.

3.1 Dimensionality reduction

3.1.1 Motivation for dimensionality reduction

In the modern data science era, terabytes of data are at our disposition while designing a model. On one hand, this is quite beneficial – models using more information, theoretically speaking, should not have worse performance than models using less information.

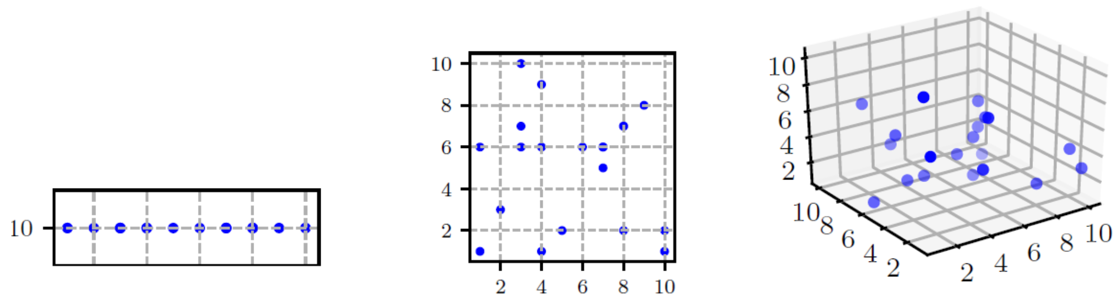
On the other hand, high-dimensional data is challenging for a variety of reasons [Gü21]:

- The computation of similarity between two data points (high-dimensional vectors) is expensive because of high complexity of distance functions.
- Highly correlated dimension could cause trouble for some algorithms.
- Curse of dimensionality: we need exponential amounts of data to characterize the density as the dimensionality goes up.
- It is hard to visualize high-dimensional data.

Example 3.1 (The curse of dimensionality [Gü21])**.** Say a discrete one-dimensional input space $x \in \{1, 2, ..., 10\}$ is given. For $N = 20$ uniformly distributed samples, the data covers 100% of the input space. If we would add a second dimension (now $x \in \{1, ..., 10\}^2$) and keep N the same, our data would only cover 18% of the input space. If we were to add a third dimension, our data would only cover 2% of the input space.

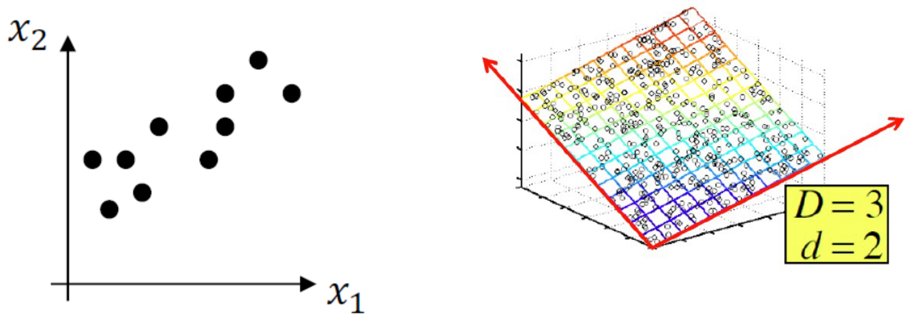
The general goal is to reduce dimensionality while avoiding information loss. The benefits thereof are, among others:

- Computational or memory savings.
- Uncovering of the intrinsic dimensionality of the data.



(a) 1 dimension – 100% covered. **(b)** 2 dimensions – 18% covered. **(c)** 3 dimensions – 2% covered.

Figure 3.1 An illustration of how the input space becomes exponentially empty when adding dimensions, given that we keep the number of data points constant [Gü21].



(a) A simple pair of correlated features.

(b) Data lying in a higher-dimensional manifold. Here D is the nominal dimension and d is the effective dimension.

Figure 3.2 Often, the data lies on a low-dimensional manifold, embedded in a high-dimensional space. A reasonable goal would be to identify this manifold and find a more efficient representation of the data set. [Gü21].

3.1.2 Strategies for dimensionality reduction

The simplest strategy

A first strategy one might deploy is to choose dimensions using a-priori knowledge or appropriate heuristics (e.g. remove low-variance dimensions). This may work well for some use cases, since there is no need for an intensive preprocessing or training phase to determine relevant dimensions.

For this to work, however, expert knowledge required and misjudgments are possible. Also, univariate feature selection ignores correlations. [Gü21]

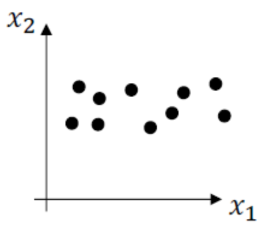


Figure 3.3 Feature selection – a scenario in which we might decide to simply ignore x_2, since x_1 contains much more variance [Gü21].

Linear techniques

Since simply discarding whole features generally is not a good idea, we often resort to more involved methods to reduce the dimensionality of a data set. One idea is to represent data in a different coordinate system via linear transformations.

From a technical perspective, this would mean the following:

- use a transformation matrix $F \in \mathbb{R}^{d \times k}$.
- $(x')^T = x^T \cdot F$ is the transformation of vector x into the new coordinate system defined by F.
- $X' = X \cdot F$ is the matrix containing all the transformed points x_i'.

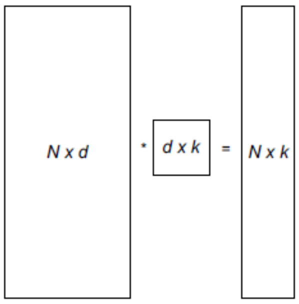


Figure 3.4 Illustration of a linear dimension reduction of some d-dimensional data set into a k-dimensional data set [Gü21].

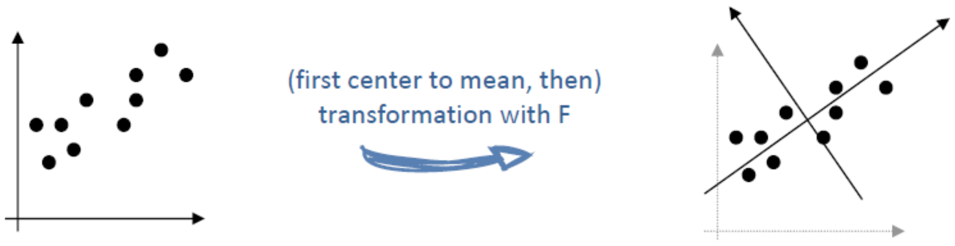


Figure 3.5 Example of a linear transformation [Gü21].

A popular and often-used method for linear dimensionality reduction is so-called Principal Component Analysis (PCA), which is in turn equivalent to Singular Value Decomposition (SVD) [Ger81]. We will not go into further detail regarding these techniques.

With this, we have somewhat illustrated linear dimensionality reduction techniques – the technique we will work with later is likewise linear, but also randomized.

Further techniques

Further (possibly non-linear) dimensionality reduction techniques are e.g. matrix factorization, neighbor graph methods (see Figure 3.6) and autoencoders [Gü21]. We will not address these topics further.

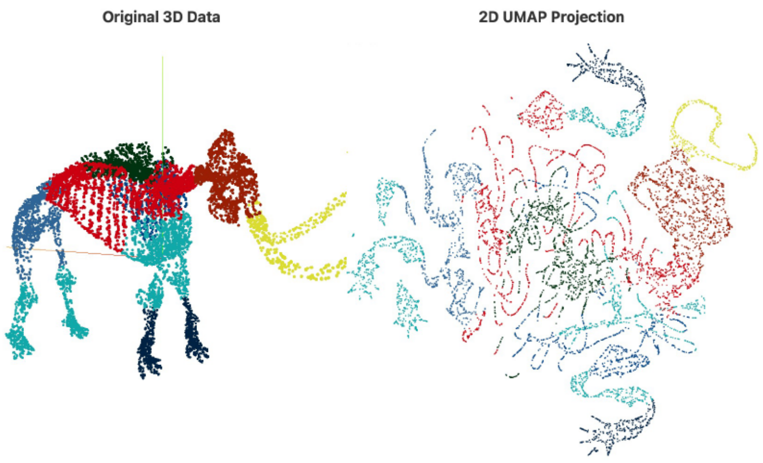


Figure 3.6 An example of a neighbor graph method - Uniform Manifold Approximation and Projection (UMAP) [Gü21] [CP19].

3.1.3 Johnson-Lindenstrauss lemma - randomized dimensionality reduction

Having recapitulated the notion of dimensionality reduction, we can now present a recently emerged linear technique that relies on randomization. That is, it is not deterministic, like PCA, for example – two reductions of the same data set into the same space might have different outcomes each time.

Theorem 3.1 (The Johnson-Lindenstrauss (JL) lemma [CGG$^+$21b])**.** *For any $0 < \epsilon < 1$ and any integer n, let k be a positive integer and such that*

$$k \geq 24\, ln(n)\ (3\epsilon^2 - 2\epsilon^3)^{-1}. \tag{3.1}$$

Then for any set P of n points in $\mathbb{R}^d$, there is a map $\phi : \mathbb{R}^d \to \mathbb{R}^k$ such that for all $v, w \in P$

$$(1-\epsilon)\|v-w\|_2^2 \leq \|\phi(v) - \phi(w)\|_2^2 \leq (1+\epsilon)\|v-w\|_2^2. \tag{3.2}$$

Furthermore, ϕ can be chosen as a linear map.

This is quite an interesting result. It puts in relation the dimension we want to compress into and how 'close' two points have to stay in the smaller space (encoded with ϵ). It then guarantees the existence of a suitable linear transformation. However, the mere existence of such a map is not useful, consider the following example.

Example 3.2 (No universal map [Kra22])**.** Assume that there is a universal linear map $x \mapsto Ax$ that satisfies (3.2), given some variable values. Next, take some z from the kernel of A (i.e. $Az = 0, z \neq 0$), we can find such a non-trivial element in the kernel since A is a wide matrix.

Now, consider a data set containing x and $z + x$, where x is just some other element from $\mathbb{R}^d$. Plugging these two elements into (3.2) yields

$$(1-\epsilon)\|z+x-x\|_2^2 = (1-\epsilon)\|z\|_2^2 \overset{?}{\leq} \|A(z+x) - A(x)\|_2^2 \tag{3.3}$$

$$= \|A(z+x-x)\|_2^2 = \|A(z)\|_2^2 = 0 \overset{?}{\leq} (1+\epsilon)\|z+x-x\|_2^2 = (1+\epsilon)\|z\|_2^2$$

Since $(1-\epsilon)\|z\|_2^2$ and $(1+\epsilon)\|z\|_2^2$ are both greater than zero (note that $0 < \epsilon < 1$), we have herewith obtained a contradiction. Therefore, there is no universal JL map.

Fortunately, the proof of the Johnson-Lindenstrauss lemma is based on randomness [Kra22]. The random construction from the proof can be mimicked in applications. This provides us with a remarkably regular formula for a (randomized) linear JL map.

Lemma 3.2 (The JL lemma for Gaussian random matrices [Kra22][CGG⁺21b]). *Assume that* $0 < \epsilon < 1$*,* $n, k, d \in \mathbb{N},\ k < d$*, and a set* P *of* n *points in* $\mathbb{R}^d$*. Next, consider that the entries of* $\Phi \in \mathbb{R}^{k\times d}$ *are sampled i.i.d. according to* $\mathcal{N}(0,1)$*. Then,*

$$\mathbb{P}\Big(\Big|\|\frac{1}{\sqrt{k}}\Phi x\|_2^2 - \|x\|_2^2\Big| > \epsilon\|x\|_2^2\Big) < 2e^{-(e^2-e^3)k/d}, \tag{3.4}$$

or, equivalently,

$$\mathbb{P}\Big((1-\epsilon)\|x\|_2^2 \leq \|\frac{1}{\sqrt{k}}\Phi x\|_2^2 \leq (1+\epsilon)\|x\|_2^2\Big) \geq 1 - 2e^{-(e^2-e^3)k/d}. \tag{3.5}$$

As announced, this theorem now gives us quite a simple guide on how to construct randomized linear transformations that preserve the neighbourhood structure of our data set. It is an astonishing fact that matrices with i.i.d. Gaussian entries distributed according to $\mathcal{N}(0, \frac{1}{k})$ are good candidates to achieve this.

Example 3.3. Let us portray this mechanism with a simple simulated example. We generate three three-dimensional clusters, which we will project into a two-dimensional space. More concretely, we do the following:

- Generate 100 data points according to $\mathcal{N}\left(\begin{bmatrix}5\\0\\0\end{bmatrix}, \begin{bmatrix}\frac{1}{10} & 0 & 0\\ 0 & \frac{1}{10} & 0\\ 0 & 0 & \frac{1}{10}\end{bmatrix}\right)$, a blue cluster.
- Generate 100 data points according to $\mathcal{N}\left(\begin{bmatrix}0\\5\\0\end{bmatrix}, \begin{bmatrix}\frac{1}{2} & 0 & 0\\ 0 & \frac{1}{2} & 0\\ 0 & 0 & \frac{1}{2}\end{bmatrix}\right)$, an orange cluster.
- Generate 100 data points according to $\mathcal{N}\left(\begin{bmatrix}0\\0\\15\end{bmatrix}, \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\right)$, a green cluster.
- Generate $\Phi \in \mathbb{R}^{2\times 3}$, with entries distributed according to $\mathcal{N}(0, \frac{1}{k})$, i.e. according to $\mathcal{N}(0, \frac{1}{2})$.
- Perform a linear dimensionality reduction using Φ on all the generated points, as portrayed in Figure 3.4.

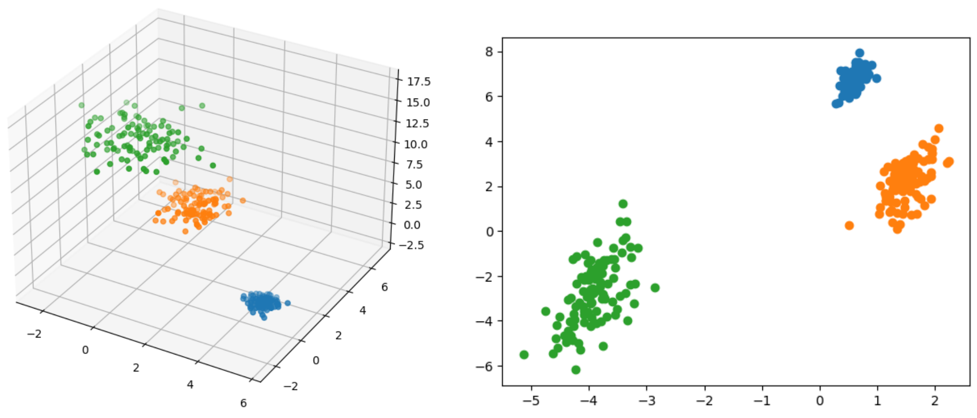
Remark 3.1. Here, we do not consider a particular value for ϵ that then provides us with a suitable k, as in Theorem 3.1. Since it is a toy example, we have predetermined $k = 2$, regardless of the error involved.

With this, we observe rather elegant results in Figure 3.7.

3.2 Randomization of the signature

Now that we have knowledge of the signature and of the Johnson-Lindenstrauss lemma, we are in a position to present a series of recent developments related to these topics. The main motif throughout these developments is the idea to use the Johnson-Lindenstrauss' reduction to obtain a randomized and projected lower-dimensional version of the signature with comparable expressivity.

If we could indeed achieve to find such an object that is more efficient to compute, we might solve the bottleneck issue arising when attempting to use the true signature as reservoir directly. This object (the randomized signature) will still be expressed as a solution of CDEs controlled by some initial path X,



(a) Original clusters in $\mathbb{R}^3$. **(b)** Projected clusters in $\mathbb{R}^2$.

Figure 3.7 Johnson-Lindenstrauss at work. We can see that the cluster structure is well preserved in the lower-dimensional space. Also, note that the variance order of the respective clusters is likewise preserved.

however its evolution will not be in a tensor product space, but rather in some lower-dimensional generic Euclidean space $\mathbb{R}^k$.

In this chapter, we will explore the main constructions as well as new developments and propositions related to this idea. We will primarily present findings based on [CBO21], [CGG$^+$21a], [CGG$^+$21b] and relate these findings between the papers.

3.2.1 Randomized signature - first definition

Let us start with a first (and relatively simple) definition from [CBO21]. Once we have this first version, we will trace back the analytical statements that relate the randomized objects back to the true signature.

Definition 3.1 (Randomized signature [CBO21])**.** For any $k \in \mathbb{N}$ big enough and appropriately chosen random matrices $A_1, ..., A_d$ in $\mathbb{R}^{k\times k}$ and random shifts $b_1, ..., b_d$ in $\mathbb{R}^{k\times 1}$, and any fixed activation function σ (notice that we explicitly allow the identity as a possible choice here), the solution of

$$dZ_t(X) = \sum_{i=1}^{d} \sigma(A_i Z_t(X) + b_i) dX_t^i, \;\; Z_0(X) = (1, 0, ..., 0) \in \mathbb{R}^k, \;\; t \in [0, T] \tag{3.6}$$

is called the *randomized signature* of X.

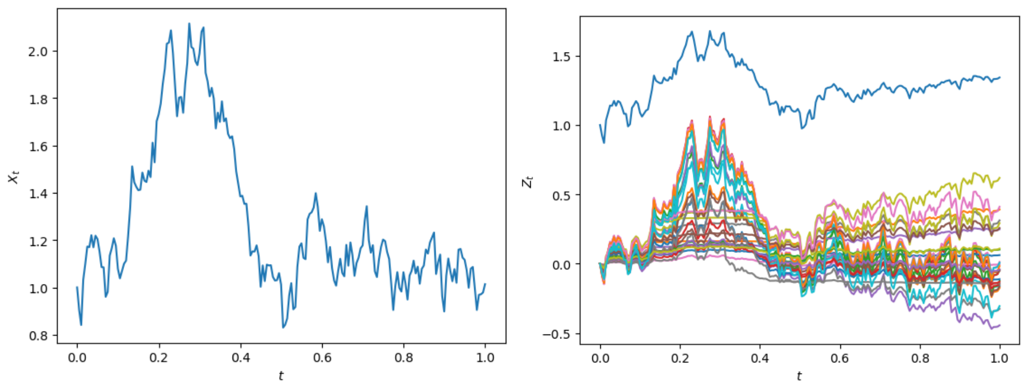
With this we have a CDE that describes how to calculate the values of an object with presumably comparable analytical properties as the true signature.

Remark 3.2. Note that we did not yet provide rigorous analytical justification for the exact form of (3.6), we will investigate this in the remainder of the chapter. It is however useful to see how elegant the end result of our efforts may be.

Firstly, notice how this object is still driven by X, but there is also randomness incorporated in its evolution due to A_i's and b_i's. This way, k, the new dimension, may be chosen significantly lower than the dimension of the corresponding truncated product tensor space of comparable expressivity.

It will just so turn out that $Z_t(X)$ is the projection of $S_t^M(X)$ from $(\mathcal{T}^M(\mathbb{R}^d), \langle\cdot,\cdot\rangle)$ to $\mathbb{R}^k$ via ϕ, where ϕ is some JL map, as per Theorem 3.1. However, this reduction is *not* performed by just applying ϕ directly (e.g. *not* like $\phi(S_t^M(X))$), but rather in a more involved and indirect way. This reduction technique, as we will see, will preserve the signature's geometric properties and its approximation power.

One of the main results in the context of the randomized signature is that the A_i's and b_i's in (3.6) turn out to be square matrices and vectors with asymptotically normally distributed, independent entries, respectively.



(a) The process trajectory. **(b)** Its randomized signature.

Figure 3.8 A sample one-dimensional trajectory X_t and its randomized signature Z_t with $k = 30$ [Drv23].

This is quite an interesting consequence related to the Gaussian version of the JL lemma. In order to get projections of the signature, one may simply sample Gaussian vectors fields and solved the given CDE.

There are multiple strategies for the choice of variance in the sampling of A_i's and b_i's. A popular method is to choose standard normal variables and deal with potential numerical issues via the choice of the activation function σ in (3.6), e.g. by choosing a flattening function such as the sigmoid.

We have admittedly made a lot new statements that leave a few things unanswered. In the next sections, we will address these findings and look at some of their proofs.

We note that the majority of the results from [CBO21] (where Definition 3.1 comes from) refer to two papers published by the same authors, namely to [CGG$^+$21a] and [CGG$^+$21b]. In following, we will analyze the contents of those two papers in our pursuit of better understanding the dynamics of the randomized signature.

3.2.2 Theoretical considerations between the true signature and its randomized version

New algebraic interpretation

Let us first take a closer look at [CGG$^+$21b]. In order to address the results about the randomized signature, we have to take a step back and put ourselves in the context of the true signature once more.

Recall that the true signature is a $\prod_{h=0}^{\infty}(\mathbb{R}^d)^{\otimes h}$-valued process, which is the solution of the following CDE

$$dS_{s,t}(X) = \sum_{i=1}^{d} S_{s,t}(X) \otimes e_i dX_t^i, \; S_{s,s}(X) = \mathbf{1}. \tag{3.7}$$

Remark 3.3. Compare the CDEs in (3.6) and (3.7). Notice that the similarity between them is that they are both driven by X. However, the latter is deterministic, while the former is contains random elements.

For advanced theoretical considerations, it turns out advantageous to observe the space $\prod_{h=0}^{\infty}(\mathbb{R}^d)^{\otimes h}$ as an abstract algebraic space that is isomorphic to it.

More concretely, in order to understand algebra, analysis and geometry of iterated integrals, one may consider the Hopf algebra $\mathbb{A}^d$, constructed by considering formal series generated by d non-commutative indeterminates $e_1, ..., e_d$ [CGG$^+$21b]. A typical element $a \in \mathbb{A}^d$ is therefore written as

$$a = \sum_{k=0}^{\infty} \sum_{i_1,...,i_k=1}^{d} a_{i_1 \dots i_k} e_{i_1} e_{i_1} \cdots e_{i_k}. \tag{3.8}$$

Remark 3.4. Note the analogy to the product tensor space. The indeterminates $e_1, ..., e_d$ have the role of a canonical basis of $\mathbb{R}^d$ and the non-commutative multiplication is analogous to the non-commutative tensor products between the elements of the canonical basis. In that sense, the collection of all the coefficients $a_{i_1 \dots i_k}$ corresponds precisely to the collection of all the entries of the true signature.

We handle this algebraic isomorphism only rudimentarily. For more details regarding the Hopf algebra structure of $\mathbb{A}^d$ we refer to Section B from [CGG$^+$21b] . For $i = 1, ..., d$ one defines on $\mathbb{A}^d$ smooth vector fields

$$a \mapsto ae_i. \tag{3.9}$$

With these concepts, it is possible to interpret (3.7) as a CDE in the Hopf algebra of $\mathbb{A}^d$.

Lastly, we will need to define a set that is analogous to the truncated signature space, as introduced in Section 2.4.3. Therefore, denote by $\mathbb{A}_d^M$ the free nilpotent algebra with d generators in which products of length more than M vanish.

In other words, instead of truncating the signature residing in $\prod_{h=0}^{\infty}(\mathbb{R}^d)^{\otimes h}$, we simply make entries after the first M ones vanish per construction. Again, this is a very informal statement and we refer to Section B from [CGG$^+$21b] for a more thorough investigation. We nevertheless make an effort to introduce this notion, such that the interested reader has it easier when investing further material (e.g. [CGG$^+$21b]).

Relation to the true signature and Gaussian-like asymptotics

We are now ready to present two theorems that relate the Johnson-Lindenstrauss lemma and the true signature to the randomized signature in Section 3.2.1. By means of a specially designed JL projection map ϕ associated to some point set (details omitted), one can project the vector field in question without losing too much information.

After that, one can project the reduced system in $\mathbb{R}^k$ back to $\mathbb{A}_d^M$ via ϕ^* ($\phi^* : \mathbb{R}^k \to \mathbb{A}_d^M$ denotes the adjoint map of ϕ with respect to the standard inner product on $\mathbb{R}^k$) and compare this reconstructed system to the original true signature. It turns out that it is ϵ-close to the signature, for some ϵ. We will write $S^M(X)$ for the finite-dimensional (truncated) version of the signature, residing in $\mathbb{A}_d^M$ [CGG$^+$21b].

The following theorem underlines the fact that the randomized signature, as defined below, is as expressive as the signature:

Theorem 3.3 (Signatures ϵ-close to each other [CGG$^+$21b]). *Let X be a smooth control and ϕ the previously mentioned JL map from $\mathbb{A}_d^M$ to $\mathbb{R}^k$. We denote by $Z_t(X)$ the randomized signature, whose evolution is defined by*

$$dZ_t(X) = \sum_{i=1}^{d} \left(\frac{1}{\sqrt{N}} \phi(\phi^*(Z_t(X))e_i) + (1 - \frac{1}{\sqrt{N}}) \phi(S_t^M(X)e_i) \right) dX_t^i, Z_0 \in \mathbb{R}^k, \tag{3.10}$$

which is a controlled differential equation on $\mathbb{R}^k$. The natural number N denotes the dimension of $\mathbb{A}_d^M$. Then

$$\langle w, S_t(X) - \phi^*(Z_t(X)) \rangle \leq \epsilon \tag{3.11}$$

for each $w \in \mathbb{A}_d^M$, where ϵ is an estimate explicitly derived in [CGG$^+$21b].

For the proof of this theorem we refer to [CGG$^+$21b]. This is an incredibly useful statement. It guarantees us that the randomized signature (as defined in (3.10), at least) is ϵ-close to the true one, i.e. that they carry comparable information. It provides us with a theoretical justification when using the randomization introduced.

Let us take a look at (3.10) in more detail. Note that $\phi^*(Z_t(X)) \in \mathbb{A}_d^M$, but by construction of $\mathbb{A}_d^M$, $\phi^*(Z_t(X))e_i \in \mathbb{A}_d^M$, also, regardless of the (tensor) product. This is rather convenient, since ϕ is a function that takes an element in $\mathbb{A}_d^M$ as input. This neatly illustrates why we needed to additionally consider the signature space algebraically.

Let us now introduce another very useful statement about the asymptotic behaviour of the vector fields. We have already hinted what the result might be.

Theorem 3.4 (Asymptotically Gaussian [CGG$^+$21b]). *For $M \to \infty$ the linear vector fields*

$$y \mapsto \frac{1}{\sqrt{N}} \phi(\phi^*(y) e_i) \tag{3.12}$$

for $i = 1, ..., d$, are asymptotically normally distributed with independent entries. The time dependent bias terms

$$(1 - \frac{1}{\sqrt{N}}) \phi(S_t^M(X) e_i) \tag{3.13}$$

are as well asymptotically normally distributed with independent entries.

Once more, we have a wonderfully elegant result. Applying it to the version of the randomized signature described in Definition 3.1, we now have a concrete strategy for sampling the A_i's and b_i's, as already suggested. Also, notice how the structures of the randomized signatures in Definition 3.1 and in Theorem 3.3 now virtually overlap, apart from the activation function σ. Asymptotically, they both have a random vector field and a random offset term.

Remark 3.5. Note that Theorem 3.4 suggests that the activation function σ from (3.6) should indeed be the identity function. We will later see, as already mentioned, how the choice of actual non-linearities helps with the numerical stability of the evolution of the signature.

For the proof of the rather interesting statement in Theorem 3.4, we are referred to another paper, namely [CGG$^+$21a]. It is not a trivial thing to show and we will have to introduce some additional concepts to tackle it. Therefore, we will dedicate to it the entirety of the next section.

3.3 One step further - signature state-affine systems

To find an answer why the random vectors fields are asymptotically Gaussian, we have to investigate [CGG$^+$21a]. Somewhat surprisingly, this paper assumes a different mathematical setup than the one we handled until now. We therefore first introduce this new model and its accompanying definitions and theorems. Afterwards, we will present the proof for the asymptotically Gaussian behaviour in the new model and address some issues with how the proof relates to claims in [CGG$^+$21b]. Finally, we present a potential strategy to bridge the models and the respective claims.

3.3.1 State-space systems - a new model

As announced, claims from [CGG$^+$21a] assume a different model structure than what we have seen so far. This time the setup is discrete, it can be described using *input/output (IO) systems*. The essential difference is we, instead of paths, observe time-series with infinite history. Let us make this more concrete. Needless to say, the following is primarily based on [CGG$^+$21a].

Some initial definitions

Along this chapter, the symbol $\mathbb{Z}$ denotes the set of all integers and $\mathbb{Z}_-$ stands for the set of negative integers with the zero element included.

Definition 3.2 (IO systems [CGG$^+$21a]). Let $D_d \subset \mathbb{R}^d$ and $D_m \subset \mathbb{R}^m$. We refer to the maps of the type $U : (D_d)^{\mathbb{Z}} \to (D_m)^{\mathbb{Z}}$ between infinite sequences with values in D_d and D_m, respectively, as discrete-time input/output systems, and to those like $H : (D_d)^{\mathbb{Z}} \to D_m$ (or $H : (D_d)^{\mathbb{Z}_-} \to D_m$) as $\mathbb{R}^m$-valued functionals.

Remark 3.6. Do note that the discretized version of our general goal is precisely to approximate the IO system between the control sequence and the target sequence.

A particularly important class of IO systems are those generated by *state-space systems* in which the output $y \in (D_m)^{\mathbb{Z}_-}$ is obtained out of the input $x \in (D_d)^{\mathbb{Z}_-}$ as the solution of the equations

$$z_t = g(z_{t-1}, x_t), \tag{3.14}$$
$$y_t = h(z_t) \tag{3.15}$$

where $g : D_N \times D_d \to D_N$ is the so-called state map, for some $D_N \subset \mathbb{R}^N, N \in \mathbb{N}$, and $h : D_N \to D_m$ is the *readout* or *observation* map.

Remark 3.7. Again, note the analogy to the general reservoir computing structure we have been pursuing throughout. The input corresponds to the control, the state space to the reservoir and the output to the target. In the discrete case, we take the new input and update the state space with some update function g, using the new input signal and the old state space value. In fact, this is how would implement the reservoir inducing CDEs in a most direct way.

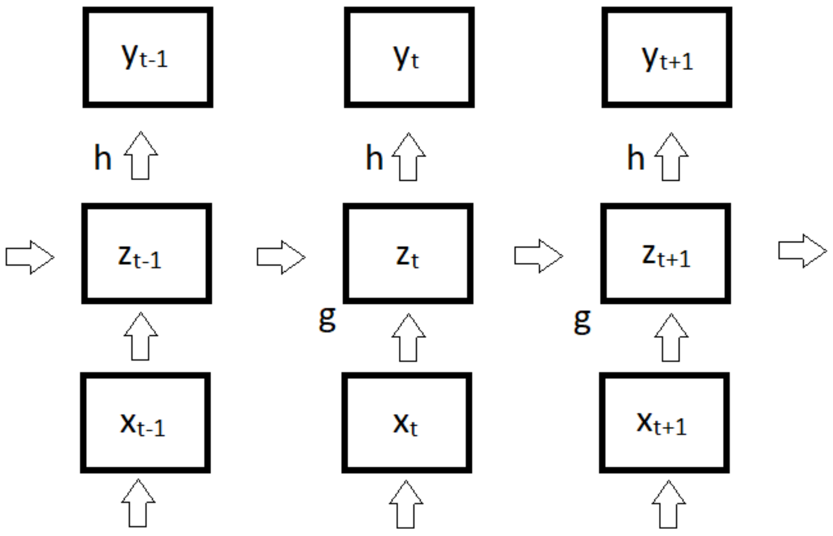


Figure 3.9 An illustration of a state space mechanism.

Remark 3.8. We have switched the roles of x and z from [CGG$^+$21a], to be more in line with the variable names in this work. We have likewise replaced the original F with g, so that the functions g, h play the analogous role as in the introduction chapter.

Let us now define a special type of state-space systems which might ring a bell.

Definition 3.3 (State-affine systems [CGG$^+$21a]). Let $\alpha = (\alpha_1, ..., \alpha_d)^T \in \mathbb{N}^d$ and $x = (x_1, ..., x_d) \in \mathbb{R}^d$ and define the monomials $x^\alpha := x_1^{\alpha_1} \cdots x_d^{\alpha_d}$. We denote by $\mathbb{M}_{N_1,N_2}$ the space of real $N_1 \times N_2$ matrices with $N_1, N_2 \in \mathbb{N}$ and use $\mathbb{M}_{N_1,N_2}[x]$ to refer to the space of polynomials in $x \in \mathbb{R}^d$ with matrix coefficients in $\mathbb{M}_{N_1,N_2}$, that is, the set of elements p of the form

$$p(x) = \sum_{\alpha \in V_p} x^\alpha A_\alpha \tag{3.16}$$

with $V_p \subset \mathbb{N}^d$ a finite subset and $A_\alpha \in \mathbb{M}_{N_1,N_2}$ the matrix coefficients. A *state-affine system (SAS)* is given by

$$z_t = p(x_t)z_{t-1} + q(x_t), \tag{3.17}$$
$$y_t = Wz_t \tag{3.18}$$

$p \in \mathbb{M}_{N,N}[x], q \in \mathbb{M}_{N,1}[x]$ are polynomials with matrix and vector coefficients, respectively, and $W \in M_{m,N}$.
This defines a class of state space systems, there is a polynomial rule for the update of the state space and a linear rule for the output function.

Remark 3.9. At this point, it is important to notice the similarities and differences to our previously studied continuous model. Indeed, there is a linear readout layer at the end that computes the output (target) using the state space (reservoir) value. This part is identical, apart from the discretization. However, the mechanism for computing the state space (reservoir) values is somewhat different. Here, there is a polynomial rule with respect to the input (control) to update the state space (reservoir), in contrast to just considering a linear-style update with all the components of the input (control). These similarities and differences will become more apparent as we introduce a special signature-inspired state-affine system in the following.

The signature state-affine system

The signature state-affine system that we will construct in this section exhibits what one calls the *strong universality property*. This means that the state equation for this state-space representation is *the same* for any IO system that is being approximated, and it is only the linear readout that changes [CGG$^+$21a]. This should sounds very familiar by now.

Interestingly enough, since the important property that we just described is reminiscent of the analogous expressivity feature of the signature object in the context of continuous paths, we will refer to this state system as the *signature* SAS (SigSAS) system [CGG$^+$21a]. Beware, this system does *not* correspond 1-to-1 to the signature map from before.

First, for any $l, d \in \mathbb{N}$, we denote by $T^l(\mathbb{R}^d)$ the space of tensors of order l on $\mathbb{R}^d$, just as before

$$T^l(\mathbb{R}^d) := \left\{ \sum_{i_1,\dots,i_l=1}^{d} a_{i_1,\dots,i_l} e_{i_1} \otimes \dots \otimes e_{i_l} | a_{i_1,\dots,i_l} \in \mathbb{R} \right\}. \tag{3.19}$$

Remark 3.10. The variable l in this context is not necessarily suggestive of the word *level*, but rather, as we will promptly see, *lag*.

Furthermore, we will be using an *order lowering map* $\pi_l : T^{l+1}(\mathbb{R}^d) \to T^l(\mathbb{R}^d)$ that, for any vector $v := \sum_{i_1,\dots,i_{l+1}=1}^{d} a_{i_1,\dots,i_{l+1}} e_{i_1} \otimes \dots \otimes e_{i_{l+1}} \in T^{l+1}(\mathbb{R}^d)$, is defined as

$$\pi_l(v) := \sum_{i_2,\dots,i_{l+1}=1}^{d} a_{1,i_2,\dots,i_{l+1}} e_{i_2} \otimes \dots \otimes e_{i_{l+1}} \in T^l(\mathbb{R}^d). \tag{3.20}$$

Intuitively, this map takes a tensor and extracts a 'subtensor' of it by taking just its first 'layer'. For example, if we had a cubic tensor, after order-lowering we would end up with its first matrix layer. The order lowering map is linear and its operator norm satisfies that $\|\pi\| = 1$ [CGG$^+$21a].

We will restrict the presentation to one-dimensional inputs, i.e. we consider input sequences $x \in K_M := \{x \in \mathbb{R}^{\mathbb{Z}_-} \mid \|x_t\| \le M, \forall t \in \mathbb{Z}_-\}$. Now, for fixed $l, p \in \mathbb{N}$, we define for any $x \in K_M$ and $t \in \mathbb{Z}_-$

$$\tilde{x}_t := \sum_{i=1}^{p+1} x_t^{i-1} e_i \in \mathbb{R}^{p+1}, \text{ and } \hat{x}_t := \tilde{x}_{t-l} \otimes \dots \otimes \tilde{x}_t. \tag{3.21}$$

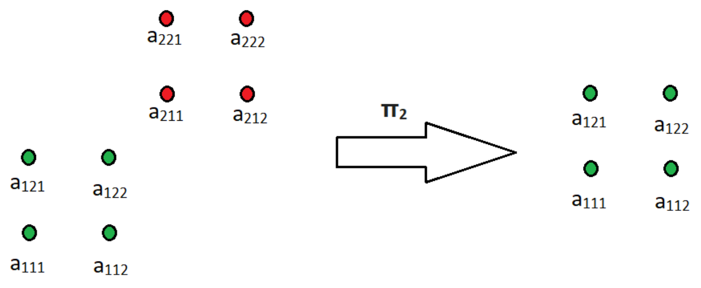


Figure 3.10 Illustration of how the operator π_l works. In this case we have a tensor in $T^3(\mathbb{R}^2)$ before and a tensor in $T^2(\mathbb{R}^2)$ after. Notice how the first layer gets extracted.

Note that $\tilde{x}_t$ is the *Vandermonde vector* [Xu16] associated to x_t and that $\hat{x}_t$ is a tensor in $T^{l+1}(\mathbb{R}^d)$ whose components in the canonical basis are all the monomials on the variables $x_t, ..., x_{t-l}$ that contain powers up to order p in each of those variables, namely [CGG+21a]

$$\hat{x}_t = \sum_{i_1,...,i_{l+1}=1}^{p+1} x_{t-l}^{i_1-1} ... x_t^{i_{l+1}-1} e_{i_1} \otimes ... \otimes e_{i_{l+1}}. \tag{3.22}$$

Note how in (3.22) one uses tensor products to elegantly express the structure of the object in one big sum. This is the same technique used in the signature inducing CDE.

Finally, given $I_0 \subset \{1, ..., p+1\}$, an arbitrarily chosen but fixed subset of cardinality higher than 1 that contains the element 1, we define [CGG+21a]

$$\hat{x}_t^0 = \sum_{i \in I_0} x_t^{i-1} \underbrace{e_1 \otimes ... \otimes e_1}_{l\text{-times}} \otimes e_i \in T^{l+1}(\mathbb{R}^d). \tag{3.23}$$

The next theorem introduces the so-called SigSAS state system of polynomial degree p and lag $-l$, where $l, p \in \mathbb{N}$. This is the system on which we will perform a similar, randomized JL dimensionality reduction and observe promising Gaussian asymptotic behaviour.

Theorem 3.5 (The SigSAS system [CGG+21a])**.** *. Let $M > 0$ and let $l, p \in \mathbb{N}$. Let $0 < \lambda < min\{1, 1/\sum_{j=0}^{p} M^j\}$. Consider the state system with uniformly bounded scalar inputs in $K_M = [-M, M]^{\mathbb{Z}_-}$ and states in $T^{l+1}(\mathbb{R}^{p+1})$ given by the recursion*

$$z_t = \lambda \pi_l(z_{t-1}) \otimes \tilde{x}_t + \hat{x}_t^0. \tag{3.24}$$

This state equation is induced by the state map $g_{\lambda,l,p}^{\text{SigSAS}} : T^{l+1}(\mathbb{R}^{p+1}) \times \mathbb{R} \to T^{l+1}(\mathbb{R}^{p+1})$ defined by

$$g_{\lambda,l,p}^{\text{SigSAS}}(z, x) := \lambda \pi_l(z) \otimes \tilde{x} + \hat{x}^0, \tag{3.25}$$

which is a contraction in the state variable with contraction constant

$$\lambda \tilde{M} < 1, \quad \text{where} \;\; \tilde{M} := \sum_{j=0}^{p} M^j. \tag{3.26}$$

Remark 3.11. Note that the object $\hat{x}_t$ does not actually enter the definition of the SigSAS system in Theorem 3.5. However, it is a good example of how the SigSAS system elements look like, i.e. what kind of information they contain. They contain interactions of different monomials of different lags of the input signal.

Once again, just as before, we have a rule how to incorporate incoming input (control) data of x into an existing and dynamically changing state space (reservoir) z. We are now ready to present some results relating to the JL compression of the newly introduced SigSAS system.

3.3.2 Random vector fields induced via the JL projection map

So far, we have presented yet another dynamical system that uses the paradigm of reservoir computing and has general expressivity properties. Similarly to our issues before, this system lies in a large tensor space and quickly gets infeasible to deal with in an efficient manner. This is why we, again, reach for randomized dimensionality reduction.

Let us briefly recover the key facts about the JL lemma and present it in a more general form using Hilbert spaces [Joh84] [DG03]. Given an N-dimensional Hilbert space $(V, \langle\cdot,\cdot\rangle)$ and Q, an n-point subset of V, the Johnson-Lindenstrauss (JL) Lemma guarantees, for any $0 < \epsilon < 1$, the existence of a linear map $\phi : V \to \mathbb{R}^k$, with $k \in \mathbb{N}$ satisfying

$$k \geq \frac{24log(n)}{3\epsilon^2 - 2\epsilon^3}, \tag{3.27}$$

that respects ϵ-approximately the distances between the points in the set Q. More specifically,

$$(1-\epsilon)\|v_1 - v_2\|^2 \leq \|\phi(v_1) - \phi(v_2)\|^2 \leq (1+\epsilon)\|v_1 - v_2\|^2 \tag{3.28}$$

for any $v_1, v_2 \in Q$. The norm $\|\cdot\|$ in $\mathbb{R}^k$ comes from an inner product that makes it into a Hilbert space [CGG$^+$21a].

In the following, we will use the original version of this result in which the JL map ϕ is realized by a matrix $\Phi \in \mathbb{M}_{k,N}$ whose entries are such that [CGG$^+$21a]

$$\Phi_{ij} \sim \mathcal{N}(0, \frac{1}{k}). \tag{3.29}$$

Let us now turn to the two main results from this [CGG$^+$21a].

Isomorphism between spaces

Theorem 3.6 (Isomorhpic state systems [CGG$^+$21a])**.** *Let $g_\rho : \mathbb{R}^N \times D_d \to \mathbb{R}^N$ be a one-parameter family of continuous state maps, where $D_d \subset \mathbb{R}^d$ is a compact subset, $0 < \rho < 1$, and g_ρ is a ρ-contraction on the first component. Let Q be a n-point spanning subset of $\mathbb{R}^N$ satisfying $-Q = Q$. Let $\phi : \mathbb{R}^N \to \mathbb{R}^k$ be a JL map that satisfies (3.28) with $0 < \epsilon < 1$, where the dimension k has been chosen so that ϕ is generically surjective. Then:*

1. *Let $g_\rho^\phi : \mathbb{R}^k \times D_d \to \mathbb{R}^k$ be the state map defined by:*

$$g_\rho^\phi(z, x) := \phi(g_\rho(\phi^*(z), x)), \tag{3.30}$$

for any $z \in \mathbb{R}^k$ and $x \in D_d$. If the parameter ρ is chosen so that

$$\rho < 1/\|\phi\|^2, \tag{3.31}$$

then g_ρ^ϕ is a contraction on the first entry. The symbol $\|\cdot\|$ in (3.31) denotes the operator norm with respect to the 2-norms in $\mathbb{R}^N$ and $\mathbb{R}^k$.

2. *Let $V_k := \phi^*(\mathbb{R}^k) \subset \mathbb{R}^N$ and let $\mathcal{G}_\rho^\phi : V_k \times D_d \to V_k$ be the state map with states on the vector space V_k, defined by:*

$$\mathcal{G}_\rho^\phi(z, x) := \phi^*(g_\rho^\phi((\phi^*)^{-1}(z), x)) = \phi^* \circ \phi(g_\rho(z, x)), \tag{3.32}$$

for any $z \in V_k$ and $x \in D_d$. If the contraction parameter satisfies (3.31), then $\mathcal{G}_\rho^\phi$ is also a contraction on the first entry. Moreover, the restricted linear map $\phi^* : \mathbb{R}^k \to V_k$ is a state-map equivariant linear isomorphism between g_ρ^ϕ and $\mathcal{G}_\rho^\phi$.

3. *Suppose, additionally, that there exist two constants $C, C_\phi > 0$ such that the state spaces of the state maps g_ρ and g_ρ^ϕ can be restricted as $g_\rho : \overline{B_{\|\cdot\|}(\mathbf{0}, C)} \times D_d \to \overline{B_{\|\cdot\|}(\mathbf{0}, C)}$ and $g_\rho^\phi : \overline{B_{\|\cdot\|}(\mathbf{0}, C_\phi)} \times D_d \to \overline{B_{\|\cdot\|}(\mathbf{0}, C_\phi)}$. Then, the state map $\mathcal{G}_\rho^\phi : \phi^*(\overline{B_{\|\cdot\|}(\mathbf{0}, C_\phi)})$ is isomorphic to the restricted version of g_ρ^ϕ and is called the JL projected version of g_ρ.*

There is admittedly a lot going on in this theorem. There are two most important takeaways for our purposes. Firstly, it gives us a formula for a compressed state space in $\mathbb{R}^k$ via the definition of $g_\rho^\phi(z, x)$ (first part of Theorem 3.6). Secondly, it establishes that the reconstruction $\mathcal{G}_\rho^\phi(z, x)$ in the bigger space $\mathbb{R}^N$ and $g_\rho^\phi(z, x)$ in $\mathbb{R}^k$ are isomorphic (third part of Theorem 3.6).

This will motivate us to apply a dimensionality reduction on the SigSAS system using (3.30) and investigate what happens mechanically in the expression for the compressed state space.

Asymptotically Gaussian behaviour

For the following theorem, additionally, the element $\hat{x}^0 \in T^{l+1}(\mathbb{R}^{p+1})$ introduced in (3.23) for the construction of the SigSAS system will be chosen in a specific randomized way. This time around, we replace (3.23) by

$$\hat{x}^0 = r \sum_{i \in I_0} x^{i-1} \underbrace{e_1 \otimes \ldots \otimes e_1}_{l\text{-times}} \otimes e_i, \tag{3.33}$$

where r is a Rademacher random variable (1 or -1 with equal probability) that is chosen independently from all the other random variables that will appear in the different constructions. If we take in $T^{l+1}(\mathbb{R}^{p+1})$ the canonical basis in lexicographic order, the element $\hat{x}^0$ can be written as the image of a linear map as [CGG+21a]

$$\hat{x}^0 = rC^{I_0}(1, x, \ldots, x^p)^T, \text{ with} \tag{3.34}$$

$$C^{I_0} := \begin{pmatrix} S^c \\ \mathbb{O}_{(p+1)((p+1)^l-1), p+1} \end{pmatrix} \in \mathbb{M}_{(p+1)^{l+1}, p+1},$$

and $S^c \in \mathbb{M}_{p+1}$ a diagonal selection matrix with the elements given by $S^c_{ii} = 1$ if $i \in I_0$ and $S^c_{ii} = 0$ otherwise. C^{I_0} is simply a matrix expanding a vector to a tensor.

Theorem 3.7 (Asymptotically Gaussian [CGG+21a])**.** *Let $M > 0$, let $\tilde{M}$ as in (3.26), $l, p, k \in \mathbb{N}$, and define $N := (p+1)^{l+1}, N_0 := (p+1)^l$. Consider a SigSAS state map $g^{\text{SigSAS}}_{\lambda,l,p} : T^{l+1}(\mathbb{R}^{p+1}) \times [-M, M] \to T^{l+1}(\mathbb{R}^{p+1})$ of the type introduced in (3.24) and defined by choosing the non-homogeneous term $\hat{x}^0$ as in (3.33). Let now $\phi : \mathbb{R}^N \to \mathbb{R}^k$ be a JL projection randomly drawn according to (3.29). Let $\delta > 0$ be small enough so that*

$$\lambda_0 := \frac{\delta}{2\tilde{M}}\sqrt{\frac{k}{N_0}} < \left\{\frac{1}{\tilde{M}}, \frac{1}{\tilde{M}\|\phi\|^2}, 1\right\}. \tag{3.35}$$

Then, the JL reduced version $\mathcal{G}^{\text{SigSAS}}_{\lambda_0,l,p,\phi}$ of $g^{\text{SigSAS}}_{\lambda_0,l,p}$ in the limit $N_0 \to \infty$ is isomorphic to the family of randomly generated SAS systems $g^{\text{SigSAS}}_{\lambda_0,l,p,\phi}$ with states in $\mathbb{R}^k$ and given by

$$g^{\text{SigSAS}}_{\lambda_0,l,p,\phi}(z, x) = \sum_{i=1}^{p+1} x^{i-1} A_i z + B(1, x, \ldots, x^p)^T, \tag{3.36}$$

where $A_1, \ldots, A_{p+1} \in \mathbb{M}_k$ and $B \in \mathbb{M}_{k,p+1}$ are random matrices whose entries are drawn according to:

$$(A_1)_{j,m}, \ldots, (A_{p+1})_{j,m} \sim \mathcal{N}(0, \frac{\delta^2}{4k\tilde{M}^2}), \tag{3.37}$$

$$B_{j,m} \sim \begin{cases} \mathcal{N}(0, \frac{1}{k}) \text{ if } m \in I_0, \\ 0 \text{ otherwise.} \end{cases} \tag{3.38}$$

All the entries in the matrices $A_1, ..., A_{p+1}$ are independent random variables. The entries in the matrix B are independent from each other and they are decorrelated and asymptotically independent (in the limit as $N_0 \to \infty$) from those in $A_1, ..., A_{p+1}$.

Proof. According to the third part of Theorem 3.6 the JL reduction $\mathcal{G}^{\text{SigSAS}}_{\lambda,l,p,\phi}$ of $g^{\text{SigSAS}}_{\lambda,l,p} = \lambda\pi_l(z) \otimes \tilde{x} + \hat{x}^0$ is isomorphic to the map $g^{\text{SigSAS}}_{\lambda,l,p,\phi}$ given by

$$g^{\text{SigSAS}}_{\lambda,l,p,\phi}(z, x) := \lambda\phi(\pi_l(\phi^*(z)) \otimes \tilde{x}) + \phi(\hat{x}^0). \tag{3.39}$$

Take now the canonical bases in $T^{l+1}(\mathbb{R}^{p+1})$ (using the lexicographic order) and $\mathbb{R}^k$ and let $\Phi \in \mathbb{M}_{k,N}$, $N = (p+1)^{l+1}$, be the random matrix associated to the JL projection ϕ in those bases. Let also $N_0 := (p+1)^l$, we separate Φ in blocks and write:

$$\Phi = (\Phi_1|...|\Phi_{p+1}), \text{ with } \Phi_1, ..., \Phi_{p+1} \in \mathbb{M}_{k,N_0}, \Phi_{ij} \sim \mathcal{N}(0, \frac{1}{k}). \tag{3.40}$$

Note how the width of the submatrices now reduces to N_0. The order-lowering map $\pi_l : T^{l+1}(\mathbb{R}^{p+1}) \to T^l(\mathbb{R}^{p+1})$ takes in these bases the matrix form

$$\Pi_l = (\mathbb{I}_{N_0}|\mathbb{O}_{N_0,pN_0}) \text{ and } \Pi_l\Phi^T = \Phi_1^T, \tag{3.41}$$

since it just takes the first tensor 'layer' and ignores the rest. Consequently, (3.39) can be rewritten in matrix form as

$$g^{\text{SigSAS}}_{\lambda,l,p,\phi}(z, x) = \lambda\sum_{i=1}^{p+1} x^{i-1}\Phi(\Pi_l(\Phi^T(z)) \otimes e_i) + \Phi\hat{x}^0. \tag{3.42}$$

We analyze this map by first spelling out the matrix associated to the linear assignment $z \to \Pi_l(\Phi^T(z)) \otimes e_i,\ i = 1, ..., p+1$. It is easy to check that:

$$\Pi_l(\Phi^T(\cdot)) \otimes e_i = \tag{3.43}$$

$$\begin{pmatrix} \mathbb{O}_{i-1,1} & \mathbb{O}_{i-1,1} & \cdots & \mathbb{O}_{i-1,1} \\ \Phi_{1_{1,1}} & \Phi_{1_{2,1}} & \cdots & \Phi_{1_{k,1}} \\ \mathbb{O}_{p-i+1,1} & \mathbb{O}_{p-i+1,1} & \cdots & \mathbb{O}_{p-i+1,1} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \mathbb{O}_{i-1,1} & \mathbb{O}_{i-1,1} & \cdots & \mathbb{O}_{i-1,1} \\ \Phi_{1_{1,(p+1)^l}} & \Phi_{1_{2,(p+1)^l}} & \cdots & \Phi_{1_{k,(p+1)^l}} \\ \mathbb{O}_{p-i+1,1} & \mathbb{O}_{p-i+1,1} & \cdots & \mathbb{O}_{p-i+1,1} \end{pmatrix}, \tag{3.44}$$

Where the first block can be referred to as Block 1 and the last one as Block $(p+1)^l$. Note that the non-trivial elements in matrix above come exclusively from the first submatrix Φ_1, due to (3.41). This fact will lead to a case distinction below. Using this matrix expression, it is easy to see that, for any $j, m \in \{1, ..., k\}$:

$$(\Phi(\Pi_l(\Phi^T(\cdot)) \otimes e_i))_{j,m} = \sum_{r=1}^{N} \Phi_{j,r}(\Pi_l(\Phi^T(\cdot)) \otimes e_i)_{r,m}$$

$$= \sum_{n=1}^{N_0} \Phi_{j,i+(n-1)(1+p)}\Phi_{m,n} \text{ with } N_0 = (p+1)^l, \tag{3.45}$$

and hence by (3.40) each of these entries is the sum of the products of two independent zero mean normal random variables with variance $1/k$, unless those two factors are identical, which can only happen whenever $j = m$ and $i + (n-1)(1+p) = n$ simultaneously, which only holds for $i = 1, n = 1$, and for the diagonal terms $j = m$. This implies that

$$(\Phi(\Pi_l(\Phi^T(\cdot)) \otimes e_1))_{j,m} \sim \begin{cases} \sum_{n=1}^{N_0} a^1_{j,m,n}, \text{ when } j \neq m, \\ \sum_{n=1}^{N_0-1} a^1_{j,m,n} + b_{j,m} \text{ otherwise,} \end{cases} \tag{3.46}$$

where $b_{j,m} = (1/k)P_{j,m},\ a^1_{j,m,n} = (1/2k)(Q_{j,m,n} - R_{j,m,n})$ (see Remark C.1), and $P_{j,m}, R_{j,m,n}, Q_{j,m,n} \sim \chi^2(1)$ are independent random variables. Analogously, for $i \in \{2, ..., p+1\}$, where we cannot have interactions between the same elements due to (3.41), it holds that

$$(\Phi(\Pi_l(\Phi^T(\cdot)) \otimes e_i))_{j,m} \sim \sum_{n=1}^{N_0} a^i_{j,m,n}, \tag{3.47}$$

with $a^i_{j,m,n}$ as above. We now study the matrix form of the summand $\Phi\hat{x}^0$ in (3.33). First of all, by (3.34),

$$\Phi\hat{x}^0 = r\Phi C^{I_0}(1, x, ..., x^p)^T, \tag{3.48}$$

and hence it can be written as $B(1, x, ..., x^p)^T = \Phi\hat{x}^0$, with $B \in \mathbb{M}_{k,p+1}$ the matrix with components

$$B_{j,m} = r\sum_{k=1}^{N} \Phi_{jk} C^{I_0}_{km} = r\Phi_{jm}\mathbf{1}_{\{m \in I_0\}}, \tag{3.49}$$

which shows (3.38), as the product of a Gaussian with a Rademacher random variable is Gaussian distributed (intuitively, think of mirroring exactly half of samples in a Gaussian-like data set).

We now prove the claim in (3.37) regarding the matrices $A_1, ..., A_{p+1}$. First of all, (3.46) and (3.47) show that for each $i \in \{1, ..., p+1\}$, the entries $(\Phi(\Pi_l(\Phi^T(\cdot)) \otimes e_i))_{j,m}$ of $\tilde{A}_i = \lambda_0\Phi(\Pi_l(\Phi^T(\cdot)) \otimes e_i) = (\delta/2M)\sqrt{k/N_0}\Phi(\Pi_l(\Phi^T(\cdot)) \otimes e_i)$ may be of two types

$$(\tilde{A}_i)_{j,m} \sim \begin{cases} \frac{\delta\sqrt{k}}{2\tilde{M}}\sqrt{N_0}\left(\frac{\sum_{n=1}^{N_0-1} a^i_{j,m,n}}{N_0} + \frac{b_{j,m}}{N_0}\right), \text{ if } i = 1, j = m, \\ \frac{\delta\sqrt{k}}{2\tilde{M}}\sqrt{N_0}\left(\frac{\sum_{n=1}^{N_0} a^i_{j,m,n}}{N_0}\right), \text{ otherwise.} \end{cases} \tag{3.50}$$

As long as we are in the cases $j \neq m$ or simultaneously $i = 1$ and $j = m$, these entries are the sum of IID mean zero random variables of variance $1/k^2$ and hence by the Lindeberg Central Limit Theorem (see Appendix Chapter C), they converge in distribution to mean zero Gaussian random variables $(A_i)_{j,m} \sim \mathcal{N}(0, \frac{\delta^2}{4kM^2})$, as required.

This straightforward argument cannot be used when $j = m$ and $i \geq 2$ as, in that situation, the random variable $\Phi_{m,i}$ appears in two different summands in the expression (3.45). The claim in that case is proved by using a martingale central limit theorem.

This is an advanced probabilistic technique and goes beyond the focus of this work. We therefore refer the interested for the rest of the proof to [CGG+21a], but also to [HH14] and [BDLR08].

□

Remark 3.12. Note that Φ is denoted with S in [CGG+21a], but we switched it to Φ to avoid confusion with the true signature and to be more explicit that the matrix is related to the JL map ϕ.

Wonderful, we have now shown why the compressed SigSAS system behaves Gaussian-like asymptotically. Although the setups are in some aspects similar, with this, we did not yet prove the analogous statement for the randomized signature, as described in Theorems 3.3 and 3.4.

Therefore, the goal in the following will be to take the definition of the randomized signature from Theorem 3.3, express it as a discretized state-space system, similar to Theorem 3.5, and then mimic the proof technique of Theorem 3.7.

3.3.3 A possibility to bridge the gap between the signature and SigSAS models

In the following, we will devise a new state-space model, combining the ideas from Theorems 3.3, 3.5 and 3.7. The main idea is to use the proof technique from the SigSAS asymptotics to show an analogous property in the randomized signature model discussed before.

Assume we observe $x' : \{0 = t_1, ..., t_n = T\} \to \mathbb{R}^d$, a discretized path. Using it, we want to define a state system motivated by the signature function. Let $S_{0,t}^{M'}(x')$ denote the true value of the M'-truncated signature of some continuous representation of x'.

An idea for the adaptation of the proof of Theorem 3.7

Let $N' = 1 + d^1 + d^2 + ... + d^{M'}$ and $N_0' = 1 + d^1 + d^2 + ... + d^{M'-1}$. The state space will be in $\mathcal{T}^{M'}(\mathbb{R}^d) = \prod_{k=1}^{M'}(\mathbb{R}^d)^{\otimes k}$, where $M' \in \mathbb{N}$ is the last signature level we consider. This is precisely the space in which the truncated signature resides. Additionally, define a new order-lowering map

$$\pi_{M'} : \mathcal{T}^{M'}(\mathbb{R}^d) \to \mathcal{T}^{M'-1}(\mathbb{R}^d),\ \pi_{M'}(\cdot) = \Pi_{M'} = \left(\mathbb{I}_{N_0'}\middle|\mathbb{O}_{N_0',\ d^{M'}}\right) \in \mathbb{M}_{N_0',\ N_0'+d^{M'}}, \tag{3.51}$$

which assumes the matrix form above once we vectorize its inputs and outputs.

Furthermore, let

$$\Delta x_t' := \sum_{i=1}^{d} \Delta(x_t')^i \otimes e_i,\ \Delta(x_t')^i := (x_t')^i - (x_{t-1}')^i \tag{3.52}$$

be the dimension-wise difference vector, defined for all the positive time points. Using it, we define an N'-dimensional state-space system z', using the following rule:

$$\Delta z_t' = \frac{1}{N'}(\pi_{M'}(z_{t-1}') \otimes \Delta x_t') + (1 - \frac{1}{N'})S_{0,t}^{M'}(x') \otimes \Delta x_t', \tag{3.53}$$

with the initial condition

$$z_0' = \mathbf{1} = \left(1 \middle| 0 \quad 0 \middle| \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \middle| \cdots \right). \tag{3.54}$$

With this, let us briefly compare the new model and the one presented in Theorem 3.5. Evidently, $x', z', \pi_{M'}$ play the role of x, z, π_l, respectively. The dimension-wise difference vector $\Delta x_t'$ plays the role of $\tilde{x}$ from before; earlier we had different monomials of the one-dimensional input in the components of the update vector, in this model we have the dimension-wise increments.

Now, our state-space is not one particular tensor space, but rather the product space of the first M' tensor spaces. Also, notice that in (3.53) we have a difference equation, expressing the increment in the state-space, unlike before. This is already reminiscent of the signature-inducing CDE.

Furthermore, $(\frac{1}{N'})$ plays the role of the scaling coefficient λ and $((1 - \frac{1}{N'})S_{0,t}^{M'}(x') \otimes \Delta x_t')$ plays the role of the offset term $\hat{x}^0$, notice the similarity to the model from Theorem 3.3 and how this makes the offset term much more involved.

With this, we can describe our new state system as follows:

$$(g^{\text{SigSAS}}_{(\frac{1}{N'}),M',d})'(\Delta z', \Delta x') = (\frac{1}{N'})\pi_{M'}(z') \otimes \Delta x' + ((1 - \frac{1}{N'})S_{0,\cdot}^{M'}(x') \otimes \Delta x'). \tag{3.55}$$

Notice how this now fully in line with the definition in Theorem 3.3, apart from the discretization.

According to the third part of Theorem 3.6 and using the technique from the proof of Theorem 3.7, the JL reduction $(\mathcal{G}^{\text{SigSAS}}_{(\frac{1}{N'}),M',d,\phi'})'$ of $(g^{\text{SigSAS}}_{(\frac{1}{N'}),M',d})'$ is isomorphic to the map $(g^{\text{SigSAS}}_{(\frac{1}{N'}),M',d,\phi'})'$ given by

$$(g^{\text{SigSAS}}_{(\frac{1}{N'}),M',d,\phi'})'(\Delta z', \Delta x') := (\frac{1}{N'})\phi'(\pi_{M'}((\phi')^*(z)) \otimes \Delta x') + \phi'(((1 - \frac{1}{N'})S_{0,\cdot}^{M'}(x') \otimes \Delta x')). \tag{3.56}$$

Let $\Phi' \in \mathbb{M}_{k,N'}$, be the random matrix associated to the JL projection ϕ' in the respective canonical bases. We separate Φ' in blocks and write:

$$\Phi' = (\Phi'_1|...|\Phi'_{M'}), \text{ with } \Phi'_1 \in \mathbb{M}_{k,1}, \Phi'_2 \in \mathbb{M}_{k,d}, ..., \Phi'_{M'} \in \mathbb{M}_{k,d^{M'}}, \Phi'_{ij} \sim \mathcal{N}(0, \frac{1}{k}). \tag{3.57}$$

Notice that these blocks are not all of the same dimensions as before. The first block Φ'_1 is just one column, Φ'_2 is a $k \times d$ matrix, and the last block $\Phi'_{M'}$ is a much bigger $k \times d^{M'}$ matrix. Also, notice how in

$$\Pi_{M'}(\Phi')^T = (\Phi'_1|...|\Phi'_{M'-1}), \tag{3.58}$$

the last, largest matrix $\Phi'_{M'}$ vanishes. Consequently, (3.56) can be rewritten in matrix form as

$$(g^{\text{SigSAS}}_{(\frac{1}{N'}),M',d,\phi'})'(\Delta z', \Delta x') = \tag{3.59}$$

$$(\frac{1}{N'})\left(\sum_{i=1}^{d} \Delta(x'_t)^i \Phi'(\Pi_{M'}((\Phi')^T(z')) \otimes e_i)\right) + (1 - \frac{1}{N'})\Phi'((S^{M'}_{0,\cdot}(x') \otimes \Delta x')). \tag{3.60}$$

We analyze this map by first spelling out the matrix associated to the linear assignment $z' \to \Pi_{M'}((\Phi')^T(z')) \otimes e_i, i = 1, ..., d$. We can check that:

$$\Pi_{M'}((\Phi')^T(\cdot)) \otimes e_i = \tag{3.61}$$

$$\left(\mathbb{I}_{N'_0}\middle|\mathbb{O}_{N'_0,\, d^{M'}}\right)\begin{pmatrix} (\Phi'_1)^T \\ --- \\ \vdots \\ --- \\ (\Phi'_{M'})^T \end{pmatrix}(\cdot) \otimes e_i = \tag{3.62}$$

$$\begin{pmatrix}
\mathbb{O}_{i-1,1} & \mathbb{O}_{i-1,1} & \cdots & \mathbb{O}_{i-1,1} \\
\Phi'_{1_{1,1}} & \Phi'_{1_{2,1}} & \cdots & \Phi'_{1_{k,1}} \\
\mathbb{O}_{d-i,1} & \mathbb{O}_{d-i,1} & \cdots & \mathbb{O}_{d-i,1} \\
\mathbb{O}_{i-1,1} & \mathbb{O}_{i-1,1} & \cdots & \mathbb{O}_{i-1,1} \\
\Phi'_{2_{1,1}} & \Phi'_{2_{2,1}} & \cdots & \Phi'_{2_{k,1}} \\
\mathbb{O}_{d-i,1} & \mathbb{O}_{d-i,1} & \cdots & \mathbb{O}_{d-i,1} \\
\mathbb{O}_{i-1,1} & \mathbb{O}_{i-1,1} & \cdots & \mathbb{O}_{i-1,1} \\
\Phi'_{2_{1,2}} & \Phi'_{2_{2,2}} & \cdots & \Phi'_{2_{k,2}} \\
\vdots & \vdots & \vdots & \vdots \\
\Phi'_{2_{1,d}} & \Phi'_{2_{2,d}} & \cdots & \Phi'_{2_{k,d}} \\
\mathbb{O}_{d-i,1} & \mathbb{O}_{d-i,1} & \cdots & \mathbb{O}_{d-i,1} \\
\mathbb{O}_{i-1,1} & \mathbb{O}_{i-1,1} & \cdots & \mathbb{O}_{i-1,1} \\
\Phi'_{3_{1,1}} & \Phi'_{3_{2,1}} & \cdots & \Phi'_{3_{k,1}} \\
\vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots \\
\mathbb{O}_{i-1,1} & \mathbb{O}_{i-1,1} & \cdots & \mathbb{O}_{i-1,1} \\
\Phi'_{(M'-1)_{1,(d^{M'-1})}} & \Phi'_{(M'-1)_{2,(d^{M'-1})}} & \cdots & \Phi'_{(M'-1)_{k,(d^{M'-1})}} \\
\mathbb{O}_{p-i+1,1} & \mathbb{O}_{p-i+1,1} & \cdots & \mathbb{O}_{p-i+1,1}
\end{pmatrix}, \tag{3.63}$$

The first three rows (the first block) in this representation correspond to the smallest submatrix Φ'_1, then there is a bigger block corresponding to the bigger submatrix Φ'_2 and then an even bigger one corresponding to the Φ'_3 and so on. Notice how, due to (3.58), the entries of $\Phi'_{M'}$ do not enter the expression above. This is analogous to entries of only Φ_1 entering the similar expression in Theorem 3.7, as shown in (3.44).

With this, we state that there is potential for this proof mechanism to continue in an analogous manner as is Theorem 3.7, showing that the update rule matrix has Gaussian-like behaviour asymptotically, as $N_0 \to \infty$.

Open questions

As for the offset term $\left((1-\frac{1}{N'})S_{0,\cdot}^{M'}(x') \otimes \Delta x'\right)$, it is defined to be a considerably more complicated object than the offset term in Theorem 3.7, defined in (3.33). Therefore, a straight-forward adaptation of the the proof of Theorem 3.7 does not suffice the investigate the asymptotic behaviour of the newly defined offset term, for instance to show that it is asymptotically Gaussian, as Theorem 3.4 suggests.

For this reason, we leave the question of the behaviour of the offset term open. One possibility to approach the investigation of the offset asymptotics might be to additionally leverage some important properties of the true signature, described e.g. in [CK16].

Also, there are open issues in the construction differences in Theorems 3.3, 3.4 and the proof idea presented above.

Firstly, the original model is in continuous time, our adaptation is discretized. It is not a straight-forward statement that the difference equations presented here imply that the differential equations in Theorem 3.3 hold. There are regularity considerations one should make before making a statement like that.

Secondly, notice the difference in how the truncation of the signature space is handled in the respective models. In the continuous one, we make use of the abstract Hopf algebra to assure that the signature space has a finite-dimensional structure. In the discretized model, however, we leverage the use of the order-lowering-map to prevent the growing of the dimensionality of the state space. It remains a subject of discussion if statements in one setup automatically imply the validity of analogous statements in the another.

With this, we conclude our theoretical considerations and turn our attention to empirical verification of the material presented.

4 Numerics and applications

After a thorough theoretical investigation, we are now ready to present a series of empirical results related to the randomized signature. We implement its computation in Python using the following difference equation algorithm, whereby we assume uniform time steps. All the simulation code is publicly available at [Drv23].

Algorithm 4.1 (Randomized signature [CBO$^+$22])**.**

```
Require: $X \in \mathbb{R}^d$ sampled at $0 = t_0 < ... < t_N = T$, dimension of randomized signature $k$, and activation function $\sigma$.
Initialize: $Z_0 = (1, 0, ..., 0) \in \mathbb{R}^k, A_i \in R^{k \times k}, b_i \in \mathbb{R}^k$ to have i.i.d. standard normal entries.
for $n$ in $N$ do
    $$Z_{t_n} = Z_{t_{n-1}} + \sum_{i=1}^{d} \sigma(A_i Z_{t_{n-1}} + b_i)(X^i_{t_n} - X^i_{t_{n-1}}) \tag{4.1}$$
end for
```

Remark 4.1. Note that we assume standard normal entries in the difference equation of the algorithm. This choice of variance is assumed for simplicity and we will deal with potential numerical instability with clever choices of the activation function σ.

In the first part, we will simulate one-dimensional and multidimensional random processes and examine the behaviour of their accompanying random signatures. We will start by investigating some initial issues regarding the number of time steps and the choice of the activation function. Afterwards, we will present systematic results of the randomized signature procedure with the optimal tuning choices.

In the second part, we will present a simple use case. Our goal will be to consider the stock price history of Amazon (regarded as a path - target), identify some correlated trajectories (controls) in the same time window and compute their randomized signature (reservoir). Once we have that, we only have to train a (regularized) linear regression between the randomized signature and the stock prices we want to recreate or predict.

4.1 Results with simulated stochastic processes

As announced, we will start by considering some simulated stochastic processes as controls. We will see if we can learn their dynamics and the dynamics of related paths expressed solely as coefficients of the linear readout. We first investigate the magnitude stability of randomized signatures of simple stochastic processes.

4.1.1 Considerations on numerical stability and activation function choice

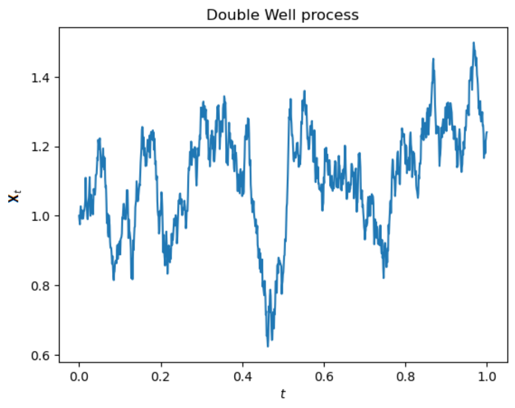
Borrowing from [CBO21], let us consider the one-dimensional stochastic Double Well process, given by

$$dX_t = \theta X_t(\mu - X_t^2)dt + \rho dB_t,\ X_0 = x_0 \in \mathbb{R},\ t \in [0, 1], \tag{4.2}$$

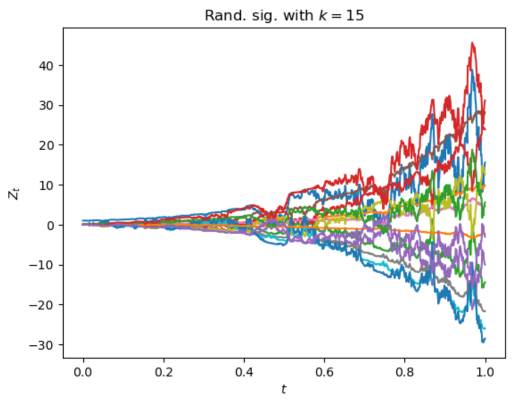
where B_t is a 1-dimensional Brownian motion, and $(\mu, \theta, \rho) \in \mathbb{R} \times \mathbb{R}^+ \times \mathbb{R}^+$.

Remark 4.2. Note that, in [CBO21], the role of parameter ρ is played by a parameter σ. We switched to ρ prevent confusion with the activation function. Likewise, Brownian motion is denoted with B_t in this work and with W_t in [CBO21], X_t plays the role of Y_t in [CBO21].

Let us fix $x_0 = 1, \mu = 2, \theta = 1, \rho = 1$ and consider the number of time steps $N_{steps} = 1001$, the dimension of the to-be-computed randomized signature $k = 15$ and the activation function to be the identity, i.e. $\sigma(x) = x$. [CBO21]. Let us simulate one realization of this process and compute its randomized signature.



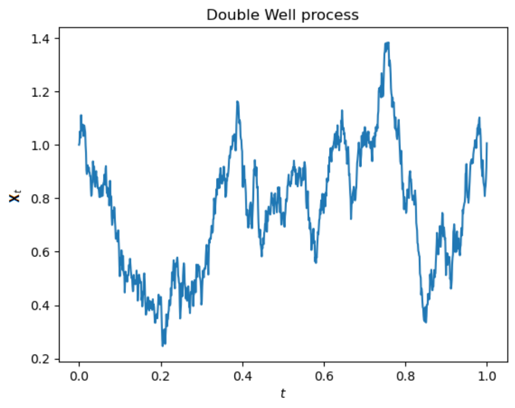
(a) The process trajectory.



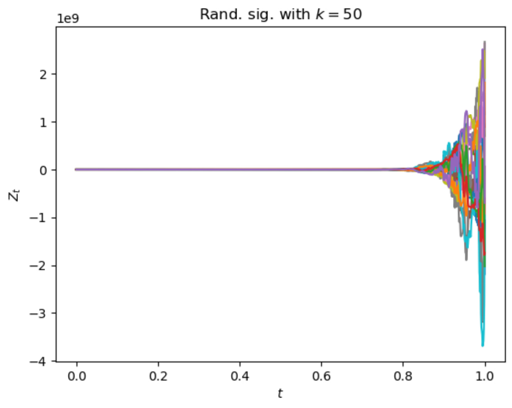
(b) Its randomized signature.

Figure 4.1 Double Well process and its randomized signature where $k = 15$ and $\sigma(x) = x$.

Note how, even with small dimensionality k, the randomized signature shows clear signs of divergence. Let us increase the parameter k and keep the rest of the parameters the same. We set $k = 50$ and simulate a new pair of results.



(a) The process trajectory.

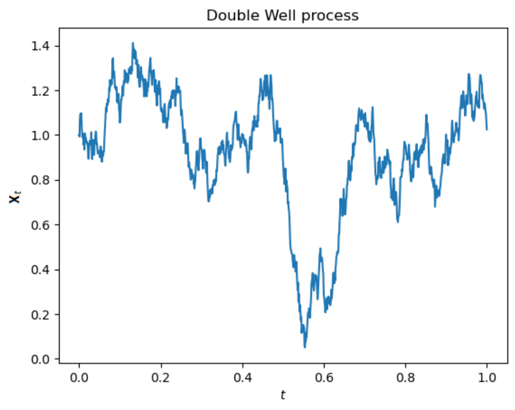


(b) Its randomized signature (first 15 entries).

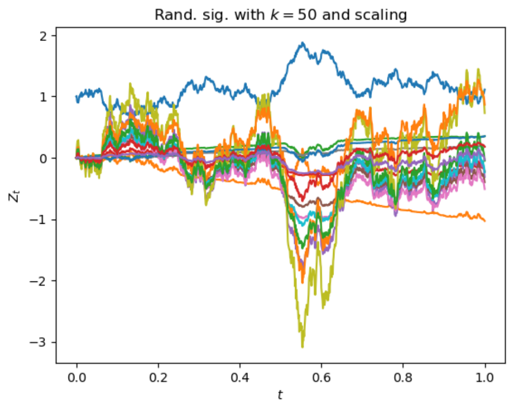
Figure 4.2 Double Well process and its randomized signature where $k = 50$ and $\sigma(x) = x$.

We can now positively identify concerning numerical instability, the randomized signature starts to diverge dramatically. As a first strategy to deal with this issue, we present a method inspired by the scaling factor λ from the SigSAS system, for instance in Theorem 3.5.

One way to adapt a scaling strategy to this context is to consider the activation functions of the form $\sigma(x) = \frac{1}{\lambda}x, \lambda > 1$. We therefore set $\sigma(x) = \frac{1}{8}x$ and keep the rest of the parameters as in the last simulation.



(a) The process trajectory.

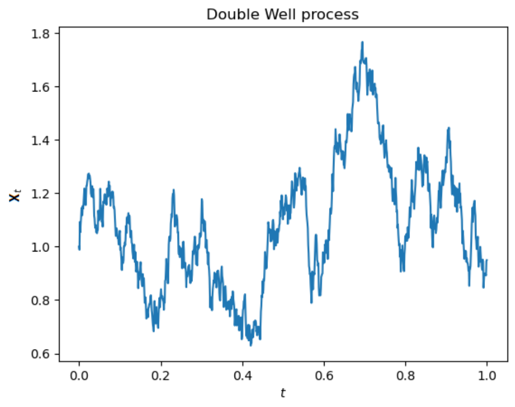


(b) Its randomized signature (first 15 entries).

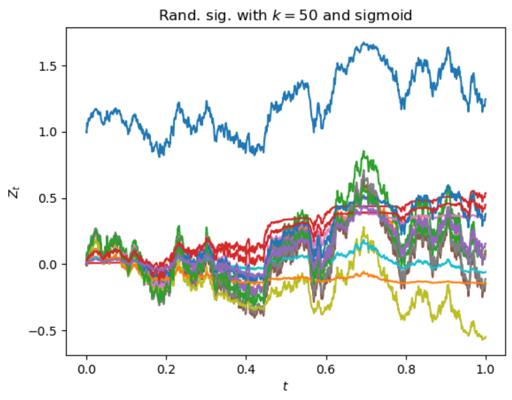
Figure 4.3 Double Well process and its randomized signature where $k = 50$ and $\sigma(x) = \frac{1}{8}x$.

This is a serious improvement, the magnitude of the values seems to be stabilized over time. However, this approach requires us to manually set the scaling coefficient λ in the expression $\sigma(x) = \frac{1}{\lambda}x$. This choice is empirically made and it depends on the specific dynamics of the investigated process.

This motivates us to define a strategy that is autonomous in this regard, i.e. that does not require any manual tuning. Luckily, literature on neural networks provides us with a well-known non-linearity that is able to accomplish this, the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. Let us keep the rest of the parameters as before and set $\sigma(x) = \frac{1}{1+e^{-x}}$.



(a) The process trajectory.



(b) Its randomized signature (first 15 entries).

Figure 4.4 Double Well process and its randomized signature where $k = 50$ and $\sigma(x) = \frac{1}{1+e^{-x}}$.

This works very well, due to the sigmoid's asymptotically flattening behaviour for larger input values. The values of the randomized signature are very much stabilized. The sigmoid will therefore be our go-to choice for the activation function σ.

With our exact algorithmic strategy, we are now ready to present more systematic results on expressivity of the randomized signature.

4.1.2 General performance results

We will now look at more structured results on the expressivity of the randomized signature. We first deal with a one-dimensional control and then with a four-dimensional one.

In both cases we do the following. We first simulate $N_{train} \in \mathbb{N}$ control trajectories and compute their accompanying randomized signatures. We then fit a ridge regression (see Appendix Chapter B) between the set of randomized signatures and some target trajectory (expressed as certain functions of the controls here). Finally, we test our trained regression on N_{test} newly simulated samples of controls and their random signatures.

Results with one-dimensional stochastic Double Well process

Let us once more briefly recover [CBO21] the definition of the one-dimensional stochastic Double Well process, given by

$$dX_t = \theta X_t(\mu - X_t^2)dt + \rho dB_t,\ X_0 = x_0 \in \mathbb{R},\ t \in [0,1], \tag{4.3}$$

where B_t is a 1-dimensional Brownian motion, and $(\mu, \theta, \rho) \in \mathbb{R} \times \mathbb{R}^+ \times \mathbb{R}^+$. We will use this process for experimentation with randomized signatures of one-dimensional controls, as in [CBO21], and build upon this work.

First, we define the following functions for a general $x \in \mathbb{R}$:

$$\begin{aligned} g_1(x) &= x \\ g_2(x) &= x^2 \\ g_3(x) &= x^3 \\ g_4(x) &= \frac{1}{1+|x|} \\ g_5(x) &= sin(x) \\ g_6(x) &= arctan(x). \end{aligned} \tag{4.4}$$

Furthermore, we fix $x_0 = 1, \mu = 2, \theta = 1, \rho = 1$ for the process [CBO21] and $N_{train} = 200, N_{test} = 100, N_{steps} = 1001, \lambda_{ridge} = 0.01$ for the training procedure. We will vary different choices of k, the dimension of the randomized signature.

As presented in the theoretical part, the idea is to find a $W \in \mathbb{R}^{1\times k}$, such that $g_i(X_t) \approx WZ_t, i \in \{1, ..., 6\}$, where Z_t is the k-dimensional randomized signature value of X_t.

Like in [CBO21], we simulate X_t according to (4.3) N_{train} times, compute the corresponding Z_t and create a training set of size $N_{train} \times N_{steps}$ (every time step of every simulation). Then, we train a ridge regression with regularization parameter λ_{ridge} between the Z_t's and $g_i(X_t)$'s from the training set.

Finally, we simulate new, unseen N_{test} samples and compare the values of $g_i(X_t)$ and $\hat{g}_i(X_t) := WZ_t$, where $W \in \mathbb{R}^{1\times k}$ is the learned linear readout via ridge regression. We compute the error in terms of an average relative L^2 error over all N_{test} samples. The dimensionality parameter k is varied from the set $\{10, 20, 50, 100, 200\}$. The best performing choice is highlighted in the results.

	$k = 10$	$k = 20$	$k = 50$	$k = 100$	$k = 200$
$g_1(x)$	0,0019128	**0,0003826**	0,0014096	0,0015360	0,0036485
$g_2(x)$	0,1916206	0,2196957	**0,1417218**	0,1607236	0,2961101
$g_3(x)$	**0,3348569**	0,4128961	0,3557608	0,3520520	0,5518399
$g_4(x)$	0,0602897	**0,0399978**	0,0431324	0,0534091	0,0589332
$g_5(x)$	0,0862645	**0,0851603**	0,0887136	0,0881958	0,1143242
$g_6(x)$	**0,0535295**	0,0549812	0,0574566	0,0647338	0,0594660

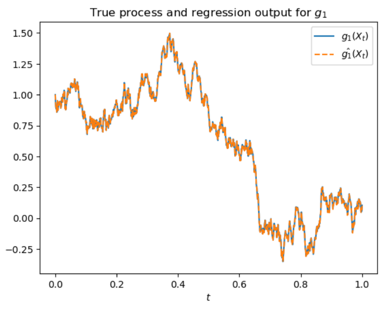
Table 4.1 Average relative L^2 error for experiments with one-dimensional controls.

An interesting observation is that, with this tuning, smaller choices for the randomized signature dimensionality k work even better than larger ones. One explanation might be that, with a relatively small N_{train}, smaller choices of k result in models of greater general performance that do not overfit the training data.

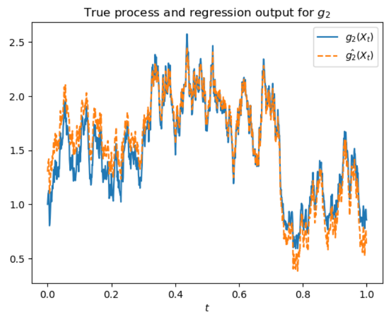
Also, we notice that the our learning model generally performs well, but has the most problems with polynomial functions of the controls. It even handles the trigonometric relations more accurately. It seems that the model has it easier with functions that have smaller value ranges and do not tend to grow rapidly, such as sin or $arctan$.

Not surprisingly, the function that was the easiest to learn the dynamics of was the identity g_1.

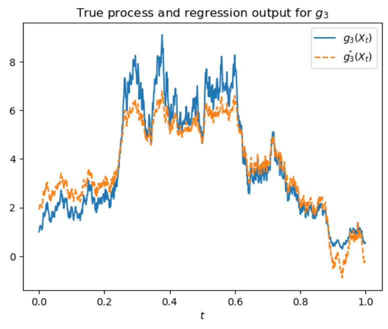
Let us additionally look at the visualizations of the respectively best performing choices of k. We visualize one randomly selected test sample: the real trajectory of $g_i(X_t)$ and its approximation $\hat{g}_i(X_t) := WZ_t \approx g_i(X_t)$.



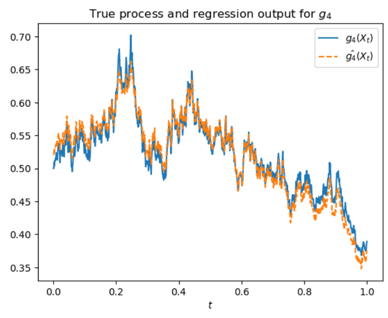
(a) $g_1(x)$



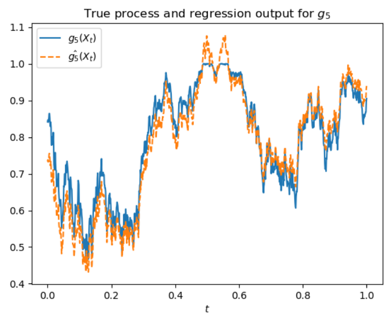
(b) $g_2(x)$



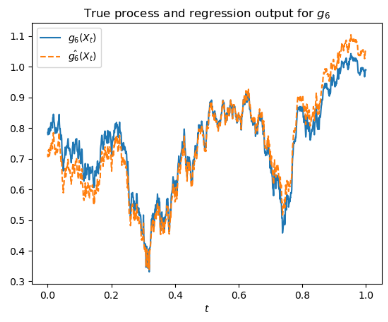
(c) $g_3(x)$



(d) $g_4(x)$



(e) $g_5(x)$



(f) $g_6(x)$

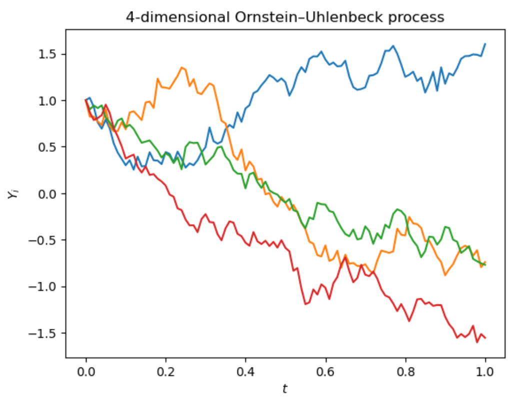
Figure 4.5 Visualizations for experiments with one-dimensional controls.

Results with four-dimensional stochastic Ornstein–Uhlenbeck process

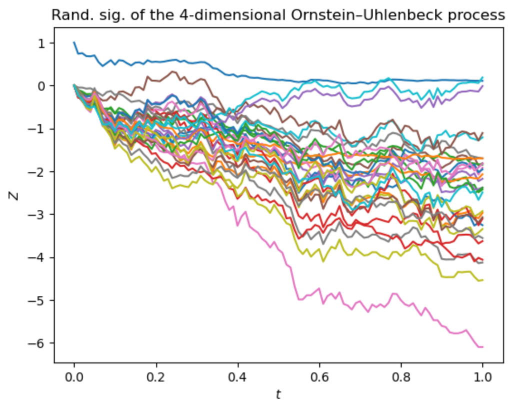
We now turn to a multidimensional control setup. Also borrowing from [CBO21], we recall that the dynamics of the four-dimensional stochastic Ornstein–Uhlenbeck process is given by

$$dX_t = (\mu - \Theta X_t)dt + \Sigma dB_t,\ X_0 = x_0 \in \mathbb{R}^4,\ t \in [0,1], \tag{4.5}$$

where B_t is a four-dimensional Brownian motion, and $(\mu, \Theta, \Sigma) \in \mathbb{R}^4 \times \mathbb{R}^{4\times4} \times \mathbb{R}^{4\times4}$.



(a) The process trajectory.



(b) Its randomized signature.

Figure 4.6 Ornstein–Uhlenbeck process and its randomized signature where $k = 30$ and $\sigma(x) = \frac{1}{1+e^{-x}}$.

Again, we define the following functions for a general $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \in \mathbb{R}^4$:

$$\begin{aligned} g_7(x) &= x_1 \\ g_8(x) &= x_2 + x_3 \\ g_9(x) &= x_1 \cdot x_4 \\ g_{10}(x) &= sin(x_3^2) \\ g_{11}(x) &= x_1^2 + x_3^3 \\ g_{12}(x) &= x_2^2 \cdot cos(x_4). \end{aligned} \tag{4.6}$$

We will do an analogous investigation using this process now. Therefore, fix $x_0 = \mathbf{1}, \mu = \mathbf{1}, \Sigma = \mathbb{I}_k, \Theta_{i,j} = i/j$ for the process [CBO21].

Due to the higher non-triviality of some test functions, we do not try out the same parameter combinations for all of them. Namely, for g_7 and g_8, we fix $N_{train} = 200, N_{test} = 100, N_{steps} = 101, \lambda_{ridge} = 0.01$ vary k from the set $\{10, 20, 50, 100, 200\}$. For the rest of the functions, we fix $N_{train} = 10000, N_{test} = 100, N_{steps} = 101, \lambda_{ridge} = 0.001$ vary k from the set $\{250, 500\}$.

Note that we reduced the number of time points in both scenarios to save some computation time, more dimensions means more computation in the CDE that induces the randomized signature. We can now do the analogous analysis as in the case of one-dimensional controls. The best performing choice is highlighted in the results.

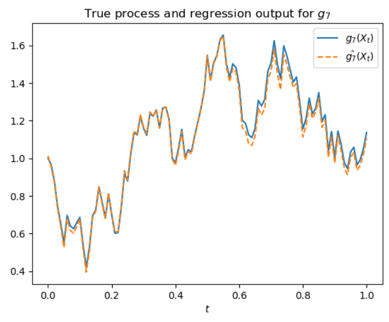
	$k = 10$	$k = 20$	$k = 50$	$k = 100$	$k = 200$
$g_7(x)$	0,2011153	0,0778641	0,0867335	0,0814606	**0,0318731**
$g_8(x)$	0,1972242	0,1150840	0,1194675	**0,0801690**	0,0848744
	$k = 250$			$k = 500$	
$g_9(x)$	0,5103904			**0,3376813**	
$g_{10}(x)$	**0,5730163**			0,7758230	
$g_{11}(x)$	**0,7366571**			0,7986202	
$g_{12}(x)$	**0,7907041**			0,8189434	

Again, the identity and functions that only depend on the respective coordinates in a linear way have quite good performance. Note that, here, bigger choices for k have better performance. Possibly, more dimensional flexibility for the model is needed once we consider multidimensional controls.

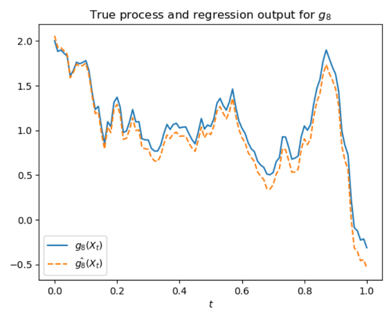
However, once we start to non-linearities and/or interactions between components the performance has a lot of room for improvement. The model can generally capture the trend of such trajectories, but it has some difficulties with strongly oscillating trajectories if they stem from some more complicated function of the control.

In this regard, there is a potential for further investigation of such models. A different hyperparameter tuning choices of some additional tinkering of the randomized signature might potentially lead to better performance results.

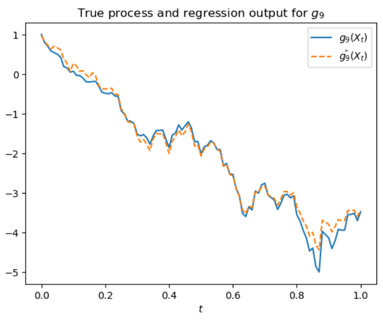
Let us again look at the visualizations of the respectively best performing choices of k. We visualize one randomly selected test sample.



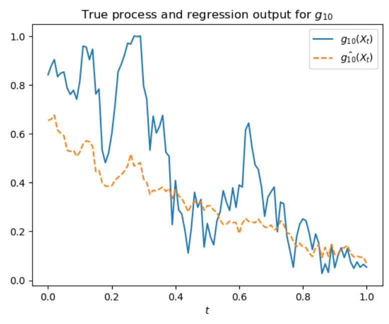
(a) $g_7(x)$



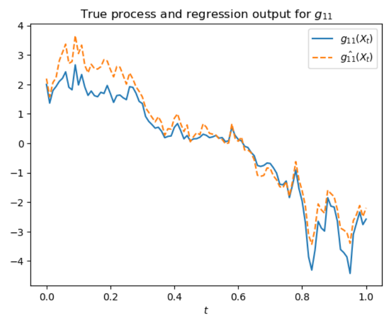
(b) $g_8(x)$



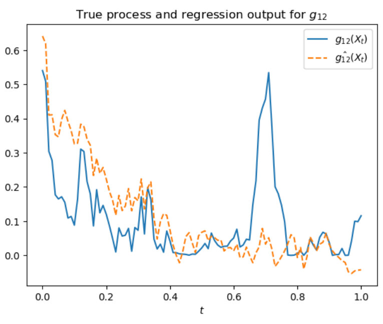
(c) $g_9(x)$



(d) $g_{10}(x)$



(e) $g_{11}(x)$



(f) $g_{12}(x)$

Figure 4.7 Visualizations for experiments with four-dimensional controls.

4.2 A simple use case - predicting Amazon stock prices

We are ready to present a simple example how the techniques discussed can be applied in industry. As an arbitrarily chosen trajectory, we will build a simple forecast algorithm for the stock price of Amazon. In other words, this will be our target.

For controls, we choose 20 other publicly available trajectories at [Yah23]. All of the data is expressed in daily frequencies (much more precision could presumably be achieved if more frequent data were available). We consider a year long interval, i.e. 365 days starting from 24/06/2022 to 23/06/2023.

Taking the notation from [Yah23], the official name of the target trajectory is

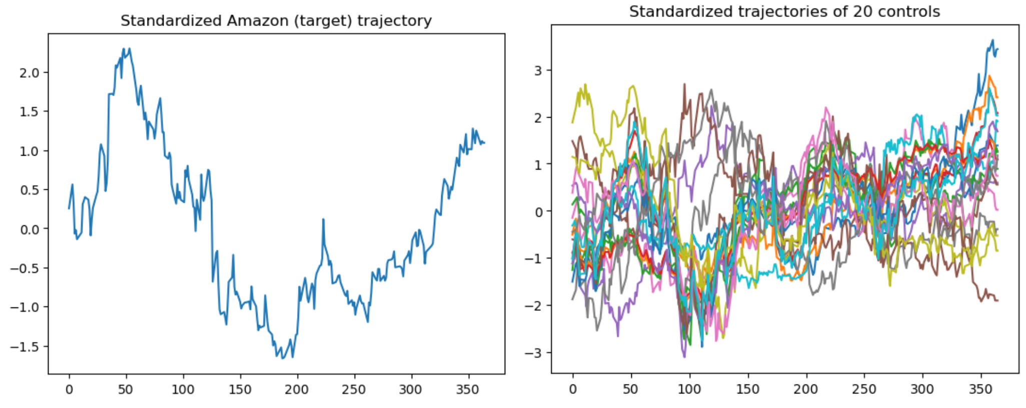
- *Amazon.com, Inc. (AMZN)*

and the other considered trajectories are as follows:

- *Nikkei 225 (N225)*
- *NASDAQ 100 (NDX)*

- *NYSE COMPOSITE (DJ) (NYA)*
- *S&P 500 INDEX (SPX)*
- *Treasury Yield 10 Years (TNX)*
- *CBOE Volatility Index (VIX)*
- *PHLX GOLD and SILVER SECTOR I (XAU)*
- *NYSE ARCA OIL and GAS INDEX (XOI)*
- *CSI 300 Index (000300.SS)*
- *Bitcoin USD (BTC-USD)*
- *Invesco S&P Global Water Index ETF (CGW)*
- *Global X DAX Germany ETF (DAX)*
- *USD/EUR (EURUSD=X)*
- *SPDR EURO STOXX 50 ETF (FEZ)*
- *USD/GBP (GBPUSD=X)*
- *Index Funds S&P 500 Equal Weight (INDEX)*
- *MSCI Emerging Markets Index Fut (MME=F)*
- *USD/JPY (JPY=X)*
- *Vanguard Real Estate Index Fund (VGSLX)*
- *Vanguard Total Stock Market Index Fund (VTI)*

Remark 4.3. Note that this a diverse set of trajectories and is not assumed to have extremely high commercial value in the context of reservoir computing for the chosen target. This example primarily has the purpose of illustrating what the implementation steps would look like, once meaningful data is at our disposal and a task is identified.



(a) The target trajectory. **(b)** The control trajectories.

Figure 4.8 Standardized data from [Yah23].

The first thing we do is standardize all data, i.e. we center it around 0 and divide it with its standard deviation. This will prevent issues due to difference in order of magnitude. We can easily recover the actual values by reversing the standardizing, so this does represent any loss of information. The standardized data is visualized in Figure 4.8

We can now construct the reservoir, i.e. the randomized signature. We have an $\mathbb{R}^{20}$-valued control, $N_{steps} = 365$ (days in a year) and we set $N_{train} = 330, N_{cv} = 15, N_{test} = 20$ to be the sizes of the training set, the cross-validation set and the test set, respectively.

After cross validation, we decide to set $k = 2000$ and $\lambda_{ridge} = 5$ to be the randomized signature dimensionality and ridge regression regularization coefficient, respectively. Due to a very large choice for k, we additionally alter the activation function to be $\sigma(x) = \frac{1}{10} \cdot \frac{1}{1+e^{-x}}$, to ensure numerical stability. Note that we combined the scaling and non-linearity strategies with that. We can then visualize our reservoir in Figure 4.9.

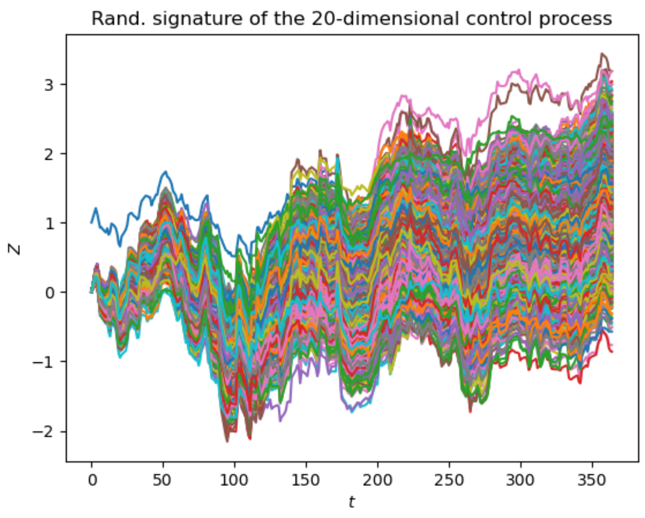


Figure 4.9 The signature entries of the controls depicted in Subfigure 4.8b.

Finally, after fitting the ridge regression between the randomized signature and target, we obtain some $W_{amzn} \in \mathbb{R}^{2000 \times 1}$ that represents the linear function rule determined by the regression and can be used as a forecasting method, once the randomized signature is at our disposal.

As we observe new values for the controls, we just have to do the linear update of the reservoir, i.e. perform further steps in the discretization of the CDE that defines the randomized signature.

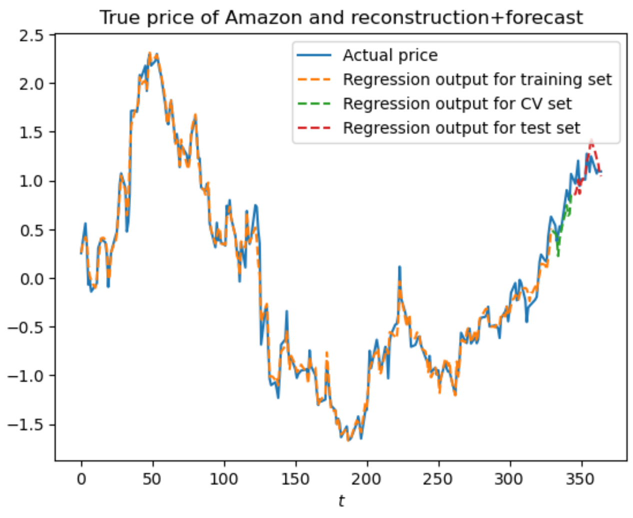


Figure 4.10 The whole learning process in one figure.

On the test set, the relative L^2 error is 0.1225176. There is, of course, a lot of room for improvement in this regard, but we emphasize once more that goal here is to just present how the techniques from this work might be applied in practice, given that enough data is at our disposition.

A Short recapitulation of tensors

This chapter will be greatly based on [KB09]. Let us, without further ado, introduce the notion of tensor.

Definition A.1 (Tensors [KB09])**.** A tensor is a multidimensional array. More formally, an N-way or N-th order tensor is an element of the tensor product of N vector spaces, each of which has its own coordinate system (basis). A third-order tensor has three indices, as shown in Figure A.1. A first-order tensor is a vector, a second-order tensor is a matrix, and tensors of order three or higher are called higher-order tensors.

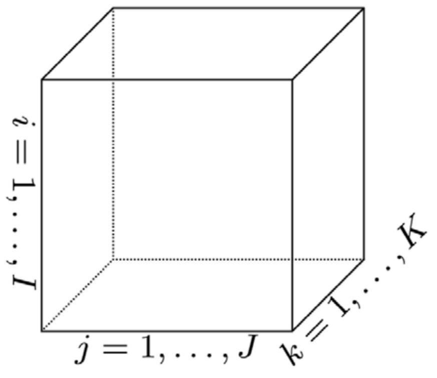


Figure A.1 A third-order tensor: $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ [KB09].

We should clarify additionally what it exactly meant by a tensor product.

Definition A.2 (Tensor product [KB09])**.** Given N vectors from N vector spaces, $a^{(1)} \in \mathbb{R}^{d_1}, ..., a^{(N)} \in \mathbb{R}^{d_N}$, the N-way tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times \, \cdots \, \times d_N}$ is the tensor product of these vectors, i.e.

$$\mathcal{X} = a^{(1)} \otimes ... \otimes a^{(N)}, \tag{A.1}$$

if the following holds:

$$\mathcal{X}_{i_1,...,i_N} = a^{(1)}_{i_1} \cdot a^{(2)}_{i_2} \cdot ... \cdot a^{(N)}_{i_N}. \tag{A.2}$$

This notion generalizes for whole vector spaces. For instance, we write $\mathbb{R}^{I \times J} := \mathbb{R}^I \otimes \mathbb{R}^J$ to denote the tensor space of all combinations of tensor products between elements of $\mathbb{R}^I$ and $\mathbb{R}^J$, respectively.

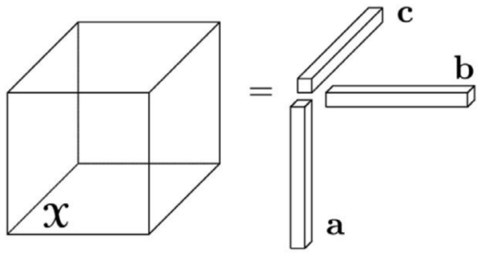


Figure A.2 A third-order tensor, $\mathcal{X} = a \otimes b \otimes c$. The (i, j, k) element of $\mathcal{X}$ is given by $\mathcal{X}_{ijk} = a_i b_j c_k$. [KB09].

A tensor is called cubical if every mode is the same size, i.e., $\mathcal{X} \in \mathbb{R}^{I \times I \times I \times \ldots \times I}$. We will deal exclusively with such tensors in this work. For instance, the l-th level of the signature resides in the standard l-th level tensor space $(\mathbb{R}^d)^{\otimes l} = \mathbb{R} \otimes \ldots \otimes \mathbb{R}$ (l times).

If we denote by $\{e_1, ..., e_d\}$ the canonical basis of $\mathbb{R}^d$, then the canonical basis of $(\mathbb{R}^d)^{\otimes l}$ is simply the set of all l-time tensor products of all combinations of l unit vectors, i.e. $\{e_{i_1} \otimes \ldots \otimes e_{i_l} \mid i_1, ..., i_l \in \{1, .., d\}\}$.

By extension, the infinite-dimensional space in which the entire signature resides is the product of all the respective level-spaces, i.e. in $\prod_{h=0}^{\infty} (\mathbb{R}^d)^{\otimes h}$ (as stated in Section 2.2.2).

B Linear regression details

This chapter serves as a refresher for the technique of linear regression, we will base it on [Gü21].

Linear regression is one of the most fundamental and simplest machine learning algorithms. With it, we try to explain one variable as a linear combination of some other variables. More formally, given are observations $X = \{x_1, x_2, ..., x_N\}, x_i \in \mathbb{R}^D$ and targets $y = \{y_1, y_2, ..., y_N\}, y_i \in \mathbb{R}$. With this, our goal is to find a mapping f such that

$$y_i \approx f(x_i), \tag{B.1}$$

where f is chosen to be a linear function, i.e.

$$f_w(x_i) = w_0 + w_1 x_{i1} + w_2 x_{i2} + ... + w_D x_{iD} \tag{B.2}$$

$$= w_0 + w^T x_i \tag{B.3}$$

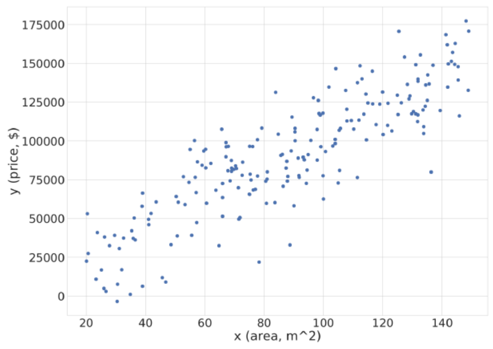


Figure B.1 Example of a use case for linear regression. We want to estimate the price of a home depending on its size using a linear rule [Gü21].

Here, w_0 is called bias or offset term. For simplicity, we can "absorb" it by prepending a 1 to the feature vector x and respectively adding w_0 to the weight vector w [Gü21]:

$$\tilde{x} = (1, x_1, ..., x_D)^T, \ \tilde{w} = (w_0, w_1, ..., w_D)^T. \tag{B.4}$$

The function f_w can compactly be written as $f_w(x) = \tilde{w}^T \tilde{x}$. To unclutter the notation, one frequently assumes the bias term to be absorbed and writes w and x instead of $\tilde{w}$ and $\tilde{x}$ [Gü21].

Standard linear regression

To find an ideal w, we have to define a loss function to needs to be optimized. A loss function measures the misfit or error between our model (parameterized by w) and observed data [Gü21]. A standard choice is the so-called least squares (LS) loss function, defined as

$$E_{LS}(w) = \frac{1}{2} \sum_{i=1}^{N} (f_w(x_i) - y_i)^2 \tag{B.5}$$

$$= \frac{1}{2} \sum_{i=1}^{N} (w^T x_i - y_i)^2. \tag{B.6}$$

We have to find the optimal weight vector w^* that minimizes the error [Gü21]:

$$w^* = \underset{w}{argmin}\ E_{LS}(w) \tag{B.7}$$

$$= \underset{w}{argmin}\ \frac{1}{2}\sum_{i=1}^{N}(w^T x_i - y_i)^2. \tag{B.8}$$

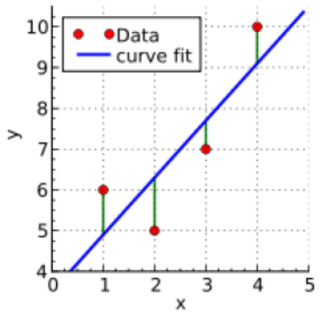


Figure B.2 An illustration of a loss function, it measures the distance between the reconstructed and actual values of the target [Gü21].

By stacking the observations x_i as rows of the matrix $X \in \mathbb{R}^{N\times D}$, one obtains

$$w^* = \underset{w}{argmin}\ \frac{1}{2}(Xw - y)^T(Xw - y). \tag{B.9}$$

To find the minimum of the loss $E(w)$, compute the gradient $\nabla_w E(w)$:

$$\nabla_w E_{LS}(w) = \nabla_w \frac{1}{2}(Xw - y)^T(Xw - y) \tag{B.10}$$

$$= \nabla_w \frac{1}{2}(w^T X^T X w - 2w^T X^t y + y^T y) \tag{B.11}$$

$$= X^T X w - X^T y. \tag{B.12}$$

Now set the gradient to zero and solve for w to obtain the minimizer (because Hessian $\nabla_w \nabla_w E(w)$ is positive (semi)definite) [Gü21]

$$X^T X w - X^T y \overset{!}{=} 0. \tag{B.13}$$

This leads to the so-called normal equation of the least squares problem

$$w^* = \underbrace{(X^T X)^{-1} X^T}_{X^\dagger} y, \tag{B.14}$$

where $X^\dagger$ is called *Moore-Penrose pseudo-inverse* of X (because for an invertible square matrix, $X^\dagger = X^{-1}$) [Gü21].

Ridge regression

In some cases, we want to add regularization to this method by constraining the magnitude of the coefficients w_i. We do this by adding the norm of w to the loss function. If one takes the 2-norm, this regularized version is called ridge regression and the loss function takes the form

$$E_{ridge}(w) = \frac{1}{2}\sum_{i=1}^{N}(w^T x_i - y_i)^2 + \frac{\lambda}{2}\|w\|_2^2, \tag{B.15}$$

where $\lambda > 0$ is the so-called regularization strength coefficient [Gü21].
Applying the same optimization details, we first have

$$E_{ridge}(w) = \frac{1}{2}(Xw - y)^T(Xw - y) + \frac{\lambda}{2}w^T w. \tag{B.16}$$

Taking the gradient yields

$$\begin{aligned} \nabla_w E_{ridge}(w) &= X^T X w - X^T y + \lambda w & \text{(B.17)} \\ &= (X^T X + \lambda \mathbf{I})w - X^T y. & \text{(B.18)} \end{aligned}$$

We now set the gradient to zero and obtain

$$(X^T X + \lambda \mathbf{I})w = X^T y, \tag{B.19}$$

implying that the optimal weight vector w^* in this case is given by

$$w^* = (X^T X + \lambda \mathbf{I})^{-1} X^T y. \tag{B.20}$$

For more details on this topic, we naturally refer to [Gü21] and [BN06].

C Probabilistic tools

Chi-squared distribution

Definition C.1 (Chi-squared distribution [Wei02])**.** If some random variables Y_i's have normal independent distributions with mean 0 and variance 1, then

$$\chi^2 = \sum_{i=1}^{r} Y_i^2 \tag{C.1}$$

is distributed as χ^2 (chi-squared) with r degrees of freedom. This makes a χ^2 distribution a gamma distribution with $\theta = 2$ and $\alpha = r/2$ [Wei04], where r is the number of degrees of freedom.

More generally, if χ_j^2 are independently distributed according to a χ^2 distribution with $r_1, r_2, ..., r_k$ degrees of freedom, then

$$\sum_{j=1}^{k} \chi_j^2 \tag{C.2}$$

is distributed according to χ^2 with $r = \sum_{j=1}^{k} r_j$ degrees of freedom.

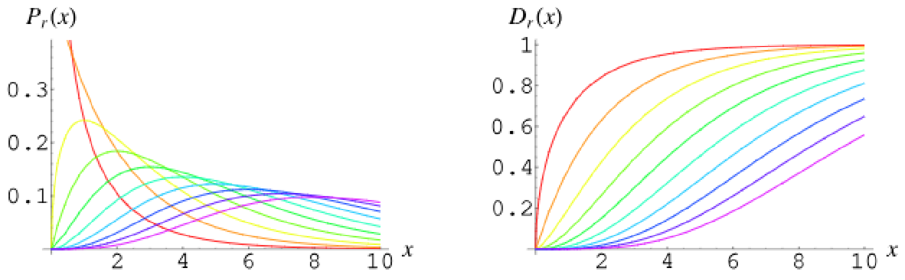


Figure C.1 The PDF and the CDF respectively of the chi-squared distribution for different choices for the number of degrees of freedom [Wei02] .

Remark C.1. Assume that we have $A, B \sim \mathcal{N}(0, 2)$ and we observe the distribution of their product AB. Then, there exist $X, Y \sim \mathcal{N}(0, 1)$, such that $X + Y = A$ and $X - Y = B$. Consequently, we can do the following:

$$AB = (X - Y)(X + Y) = X^2 - Y^2 = Q - R, \tag{C.3}$$

where $Q, R \sim \chi^2(1)$ model the distribution of X^2, Y^2, respectively.

(Lindeberg) Central Limit Theorem

Let us here state the classical central limit theorem, a fundamental law in probability theory.

Theorem C.1 (Central limit theorem [BT08]). *Let $X_1, X_2, ...$ be a sequence of independent identically distributed random variables with common mean μ and variance σ^2 , and define*

$$Z_n = \frac{X_1 + ... + X_n - n\mu}{\sigma\sqrt{n}}. \tag{C.4}$$

Then, the CDF of Z_n converges to the standard normal CDF

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-x^2/2} dx, \tag{C.5}$$

in the sense that

$$\lim_{n\to\infty} \mathbb{P}(Z_n \leq z) = \Phi(z), \ \forall z. \tag{C.6}$$

There are more general versions of this important law, the one relevant for this work is the so called Lindeberg central limit theorem. We do not go in too much detail at this point and refer the reader to additional references.

Lindeberg's condition is a sufficient condition (and under certain conditions also a necessary condition) for the central limit theorem (CLT) to hold for a sequence of independent random variables. Unlike the classical CLT, which requires that the random variables in question have finite variance and be both independent and identically distributed, Lindeberg's CLT only requires that they have finite variance, satisfy Lindeberg's condition, and be independent [Wik23].

[BDLR08] provides an in-depth guide to this probabilistic technique.

Bibliography

[BDLR08] Jean-Marc Bardet, Paul Doukhan, Gabriel Lang, and Nicolas Ragache. Dependent lindeberg central limit theorem and some applications. *ESAIM: Probability and Statistics*, 12:154–172, 2008.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[BT08] Dimitri Bertsekas and John N Tsitsiklis. *Introduction to probability*, volume 1. Athena Scientific, 2008.

[CBO21] Enea Monzio Compagnoni, Luca Biggio, and Antonio Orvieto. Empirics on the expressiveness of randomized signature. In *The Symbiosis of Deep Learning and Differential Equations*, 2021.

[CBO$^+$22] Enea Monzio Compagnoni, Luca Biggio, Antonio Orvieto, Thomas Hofmann, and Josef Teichmann. Randomized signature layers for signal extraction in time series data. *arXiv preprint arXiv:2201.00384*, 2022.

[CGG$^+$21a] Christa Cuchiero, Lukas Gonon, Lyudmila Grigoryeva, Juan-Pablo Ortega, and Josef Teichmann. Discrete-time signatures and randomness in reservoir computing. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6321–6330, 2021.

[CGG$^+$21b] Christa Cuchiero, Lukas Gonon, Lyudmila Grigoryeva, Juan-Pablo Ortega, and Josef Teichmann. Expressive power of randomized signature. In *The Symbiosis of Deep Learning and Differential Equations*, 2021.

[CK16] Ilya Chevyrev and Andrey Kormilitzin. A primer on the signature method in machine learning. Technical report, University of Oxford, 2016.

[CLS23] Nicola Muca Cirone, Maud Lemercier, and Cristopher Salvi. Neural signature kernels as infinite-width-depth-limits of controlled resnets. *arXiv preprint arXiv:2303.17671*, 2023.

[CP19] Andy Coenen and Adam Pearce. Understanding umap (google pair). *Understanding UMAP (Google PAIR). Library Catalog: pair-code. github. io*, 4, 2019.

[DG03] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

[DORK20] Jonathan Dong, Ruben Ohana, Mushegh Rafayelyan, and Florent Krzakala. Reservoir computing meets recurrent kernels and structured transforms. *Advances in Neural Information Processing Systems*, 33:16785–16796, 2020.

[Drv23] Vjekoslav Drvar. Randomization and Differential Equations in the Context of the Signature - Github repository. `https://github.com/vdrvar/rand_and_diff_eq_in_the_context_of_the_sig`, 2023.

[Ger81] Jan J Gerbrands. On the relationships between svd, klt and pca. *Pattern recognition*, 14(1-6):375–381, 1981.

[Gér22] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[Gü21] Stephan Günnemann. *Slides from the lecture 'Machine Learning'*. Technical University of Munich, 2021.

[HH14] Peter Hall and Christopher C Heyde. *Martingale limit theory and its application*. Academic press, 2014.

[Hig01] Desmond J Higham. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM review*, 43(3):525–546, 2001.

[Joh84] William B Johnson. Extensions of lipschitz mappings into a hilbert space. *Contemp. Math.*, 26:189–206, 1984.

[KB09] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[Kid21] Patrick Kidger. *On neural differential equations*. PhD thesis, University of Oxford, 2021.

[KL21] Patrick Kidger and Terry Lyons. Signatory: differentiable computations of the signature and logsignature transforms, on both CPU and GPU. In *International Conference on Learning Representations*, 2021. `https://github.com/patrick-kidger/signatory`.

[KMFL20] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33:6696–6707, 2020.

[Kra22] Felix Krahmer. *Slides from the lecture 'Foundations in Data Analysis'*. Technical University of Munich, 2022.

[LJ09] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer science review*, 3(3):127–149, 2009.

[MP10] Peter Mörters and Yuval Peres. *Brownian motion*, volume 30. Cambridge University Press, 2010.

[Per21] Ari-Pekka Perkkiö. *Slides from the lecture 'Angewandte Finanzmathematik 2021: Introduction to the Black-Scholes World'*. Ludwig Maximilian University Munich, 2021.

[Pet95] Valentin V Petrov. Limit theorems of probability theory; sequences of independent random variables. Technical report, 1995.

[PT85] Etienne Pardoux and Denis Talay. Discretization and simulation of stochastic differential equations. *Acta Applicandae Mathematica*, 3:23–47, 1985.

[Ram22] Luciano Ramalho. *Fluent python*. " O'Reilly Media, Inc.", 2022.

[RR07] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[Tir20] Tirthajyoti Sarkar. Brownian motion with python. `https://towardsdatascience.com/brownian-motion-with-python-9083ebc46ff0`, 2020.

[TYH$^+$19] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019.

[Una22] Unacademy. Pendulum. https://unacademy.com/content/nda/study-material/physics/pendulum/, 2022.

[Van16] Jake VanderPlas. *Python data science handbook: Essential tools for working with data*. " O'Reilly Media, Inc.", 2016.

[VSdS07] David Verstraeten, Benjamin Schrauwen, Michiel d'Haene, and Dirk Stroobandt. An experimental unification of reservoir computing methods. *Neural networks*, 20(3):391–403, 2007.

[Wei02] Eric W Weisstein. Chi-squared distribution. *https://mathworld. wolfram. com/*, 2002.

[Wei04] Eric W Weisstein. Gamma distribution. *https://mathworld. wolfram. com/*, 2004.

[Wik23] Wikipedia. Lindeberg's condition. https://en.wikipedia.org/wiki/Lindeberg%27s_condition/, 2023.

[Xu16] Changqing Xu. Hankel tensors, vandermonde tensors and their positivities. *Linear Algebra and its Applications*, 491:56–72, 2016.

[Yah23] Yahoo. Yahoo Finance. https://finance.yahoo.com/, 2023.

[YFXJ20] Yue Yunguan, Lei Fengchun, Liu Xingwu, and Wu Jie. Asynchronous Computability Theorem in Arbitrary Solo Models. https://www.mdpi.com/2227-7390/8/5/757, 2020.

Abbreviations

Ordered alphabetically:

CDE	controlled differential equation
CDF	cumulative distribution function
CLT	central limit theorem
IO	input-output
JL	Johnson-Lindenstrauss
LS	least squares
ODE	ordinary differential equation
PCA	principal component analysis
PDF	probability density function
RC	reservoir computing
SAS	state-affine system
SigSAS	signature state-affine system
SVD	singular value decomposition
UMAP	uniform manifold approximation and projection