Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Continuous Reinforcement Learning for Delayed Dynamical Systems

A Path Signature-Based Representation Approach

JOACHIM JOBARD

Continuous Reinforcement Learning for Delayed Dynamical Systems

A Path Signature-Based Representation Approach

JOACHIM JOBARD

CDATE

Supervisors: Lionel Mathelin, Onofrio Semeraro, Rémy Hosseinkhan-Boucher, Erik Fransén

Examiner: Arvind Kumar
School of Electrical Engineering and Computer Science

Host company: INRIA (Institut National de Recherche en Informatique et Automatique)

© 2026 Joachim Jobard

Abstract

Controlling physical systems governed by Delay poses a major challenge for standard Reinforcement Learning (RL). Traditional RL algorithms rely on the Markov property and struggle with the infinite-dimensional space induced by Delayed Differential Equations (DDEs) and Functional Differential Equations (FDEs) required to represent the system continuous history.

This thesis introduces a novel, model-free control framework based on the *signature transform*, a tool from rough path theory, with Continuous Time Reinforcement Learning (CTRL). Using the universal approximation property of signatures, we encode the trajectory of the system into a fixed-size tensor, bypassing dimensionality and discretization issues.

We establish the theoretical foundations for this approach by basing our work on the Hamilton-Jacobi-Bellman (HJB) equation for functional states, and adapting CTRL algorithms to this framework. We then implement our own DDE simulation environment to test the performances of our signature-based Continuous Time Actor Critic (CTAC) and Value-Gradient (VG) algorithms.

By comparing our approach against Markovian and history-based baselines, our work highlights that the signature transform provides a mathematically principled and practically effective way to approximate classical RL functions (policy and Value Function) in non-Markovian environments.

Keywords

Reinforcement Learning, Delayed Systems, Signatures, Machine Learning

Contents

Abstract i
Keywords i
Contents ii
List of Figures v
List of Tables vii
List of Acronyms and Abbreviations 1
Acknowledgments 2
1 Introduction 3
1.1 Background 4
1.2 Problem 4
1.2.1 Resarch Questions 4
1.3 Purpose 5
1.4 Goals 5
1.5 Research Methodology 6
1.6 Delimitations 6
1.7 Structure of the Thesis 7
2 Related Work 8
2.1 Controlling Non-Markovian systems 8
2.2 CTRL 9
2.3 Signatures as a Path Representation 10
2.3.1 Signature in Machine Learning 10
2.3.2 Comparison to the Fourier Transform 10
2.3.3 Signatures in Control 11
3 Background 13
3.1 Mathematical Foundations of Signatures 13
3.1.1 Definition of the signature transform 13
3.1.2 Signature with tensor algebra 15
3.1.3 Chen’s identity 16
3.1.4 The universal nonlinearity theorem 17
3.2 DDEs and FDEs 18
3.2.1 Some properties of DDEs and FDEs 18

3.2.2 A Hamilton-Jacobi-Bellman equation for Delayed Systems . . . 20
3.3 Reinforcement Learning . . . 20
3.3.1 Optimal Control and the HJB equation . . . 21
3.3.2 Continuous Reinforcement Learning . . . 22
4 Approach . . . 24
4.1 Signatures as a function approximation . . . 24
4.1.1 Approximating the Value function . . . 25
4.1.2 Approximating the Actor . . . 25
4.2 Algorithms . . . 26
4.2.1 Continuous Time Actor Critic . . . 26
4.2.2 Value Gradient . . . 29
5 Implementation and Case Study . . . 34
5.1 Simulation Environment . . . 34
5.1.1 DDE solver . . . 34
5.1.2 Reinforcement Learning Environment . . . 36
5.2 Rolling Signatures . . . 36
5.3 Technical implementation of RL algorithms . . . 37
5.3.1 Approximation functions . . . 37
5.3.2 Tweaks and Tricks . . . 38
5.4 Test cases . . . 39
5.4.1 2D system with delay . . . 39
5.4.2 Mackey-Glass in 1D . . . 40
5.4.3 Chemical reaction . . . 41
6 Results and Analysis . . . 43
6.1 2D Damped system . . . 44
6.2 Mackey-Glass . . . 44
6.3 Chemical Process . . . 48
6.3.1 Comparison Table . . . 48
6.3.2 Best result comparison . . . 50
6.3.3 Mean cost per method comparison . . . 51
6.3.4 State evolution best per method . . . 52
6.3.5 State evolution mean per method . . . 53
6.3.6 Value Function analysis . . . 53
7 Discussion . . . 55
7.1 Interpretation of Results . . . 55

7.2 Limitations 55
7.2.1 Case study 55
7.2.2 Stability 56
7.2.3 Reward design 57
7.3 Future Work 57
7.3.1 Addressing the limitations 58
7.3.2 Policy Gradient and other algorithms 58
7.3.3 Partially Observable Markov Decision Process 58
7.4 Reflections 59
7.5 Ethics, Sustainability and Societal Impact 59
7.5.1 Sustainability 59
7.5.2 Ethics 60
7.5.3 Relations with the Surrounding Community 61
8 Conclusions 62
Bibliography 63
A Proof extensions 69
A.1 Signature simplification 69
A.2 Tensor truncation 70
A.3 Total variation of a path 70
B Supplementary experiments 72
B.1 Mackey Glass without transient 72
B.2 Window size experiments on Mackey-Glass 73
C Hyperparameter Selection 78
C.1 Linear System 78
C.2 Mackey-Glass 80
C.3 Chemical System 81
~~€€€€~~ For DIVA ~~€€€€~~ 83

List of Figures

Figure 1 Evolution of the relative error between finite difference scheme and Equation 33 w.r.t. dt the discretisation step 29
Figure 2 Comparison of `JiTCDDE` solver with our own implementation of RK4 for a 2D delayed system 36
Figure 3 Comparison of `JiTCDDE` solver with our own implementation of RK4 over the Mackey-Glass equation 36
Figure 4 2D system Equation 55 trajectory 40
Figure 5 Trajectories of Mackey-Glass systems for different values of τ, the delay 40
Figure 6 Attractor of Mackey-Glass systems for different values of τ, the delay 40
Figure 7 Evolution of the respective states of the system Equation 57 w.r.t. time 42
Figure 8 Comparison of metrics (mean and std) on a 2D delayed system for different control techniques 44
Figure 9 Comparison of trajectories and estimated V function of Mackey-Glass system with $\tau = 8$ for different control techniques over 10 seeds 48
Figure 10 Comparison of cumulated cost on Mackey-Glass system with $\tau = 8$ for different control techniques over 10 seeds 48
Figure 11 Comparison of trajectories and estimated V function of Mackey-Glass system with $\tau = 17$ for different control techniques over 10 seeds 48
Figure 12 Comparison of cumulated cost on Mackey-Glass system with $\tau = 17$ for different control techniques over 10 seeds 48
Figure 13 Comparison of trajectories and estimated V function of Mackey-Glass system with $\tau = 30$ for different control techniques over 10 seeds 48
Figure 14 Comparison of cumulated cost on Mackey-Glass system with $\tau = 30$ for different control techniques over 10 seeds 48

Figure 15 Best cumulated cost for various RL techniques (lowest is better) 50
Figure 16 Comparison of mean and quantiles (10-90 %) values for various techniques on the chemical reaction system **conditional on success** (Equation 57) 51
Figure 17 State trajectory per dimension for the system Equation 57 with different RL method, selecting the one that achieved the best cumulated cost 52
Figure 18 Mean state trajectory per dimension for the system Equation 57 with different RL method, mean and quantiles (10-90%), **conditional on success** 53
Figure 19 Value function evolution w.r.t. time for different algorithms . . . 54
Figure 20 Comparison of cumulated cost on Mackey-Glass system with $\tau = 30$ for different control techniques over 5 seeds without transient times 73
Figure 21 Comparison of cumulated cost on Mackey-Glass system with $\tau = 30$ over 5 seeds for different time window size 73
Figure 22 Evolution of mode ratio of the signature for various depths . . . 76
Figure 23 Evolution of mode ratio of the full trajectory 76

List of Tables

Table 1 Comparison of techniques on the chemical process: seed removal statistics and performance metrics. 49
Table 2 Main hyperparameters for the compared continuous-time RL methods for the linear system Equation 55. 79
Table 3 Main hyperparameters for the compared continuous-time RL methods for Mackey-Glass system Equation 56. 80
Table 4 Main hyperparameters for the compared continuous-time RL methods for the chemical system Equation 57. 81

List of Acronyms and Abbreviations

CTAC	Continuous Time Actor Critic
CTRL	Continuous Time Reinforcement Learning
DDE	Delayed Differential Equation
DP	Dynamic Programming
FDE	Functional Differential Equation
HJB	Hamilton-Jacobi-Bellman
KS	Kuramoto–Sivashinsky
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
ODE	Ordinary Differential Equation
RK	Runge-Kutta
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SAC	Soft Actor Critic
STD	standard deviation
SoTA	State-of-The-Art
TD3	twin-delayed deep deterministic policy gradient
VG	Value-Gradient

Acknowledgments

I would like to thank my supervisors: Lionel and Onofrio for their kind guidance and help through the project, for their ideas and support during these 6 months. I would also like to thank Rémy for his precious advices and his view on how to conduct research work. Thanks for having taken from your very own time to help me on this project, and for your involvement and precious discussions. I also want to thank Erik for his reviews on this manuscript and advices on the redaction.

A big thanks to you the TAU team, from which I've learned so much during these 6 months and for their friendship, for the card and board games at lunch, for the climbing sessions, for the running Thursdays… Thank you, Alena, Alex, Anaclara, Chris, Cyriaque, Dylan, Eduardo, Félix, Ferdinand, Gianluca, Giulia, Hugo, José, Lorenzo, Luca, Nicolas, Nilo, Sébastien, Solal, Stella, Stéphane, Théo, Theophanis, Thomas, Tomasso, Trang and Vae.

Finally, I want to thank Yasmine without whom these 6 months would have been much harder, thank you for being every day at my side, and thank you for allowing me to be at your side everyday.

Gif-sur-Yvette, May 2026
Joachim Jobard

1 Introduction

Reinforcement Learning (RL) has achieved remarkable success in solving complex control tasks such as robotic manipulation or strategy games. However, theoretical guarantees and practical stability rely on the Markov property: the next state of the system only depends on the current one. In many physical systems – such as chemical or aerodynamic turbulence for example – this assumption breaks down. Systems exhibit delays and memory: the evolution of a state is conditioned by the current one, but also another one at time $t - \tau$, or even the full trajectory up to a time in the past. Physically, these systems are not modelled by Ordinary Differential Equation (ODE) but by Delayed Differential Equation (DDE) or Functional Differential Equation (FDE).

A natural question then arises: how can one design control algorithms that account for such memory effects? Classical approaches to delayed optimal control, such as Dynamic Programming (DP) methods [1] or Linear Quadratic Regulator (LQR) extensions [2] exist but often suffer from the curse of dimensionality or require explicit knowledge of the system dynamics. On the other hand, RL offers a model-free framework that has proven effective in high-dimensional settings – yet its standard formulation assumes Markovian states, making it ill-suited for delayed systems out of the box.

Recently, the *signature transform*, a mathematical tool from rough path theory [3], [4] has emerged as a powerful way to encode sequential and path-dependent data into a fixed-size feature vector. Its universal approximation property [5] guarantees that any continuous function of a path can be approximated as a linear function of its signature. This makes signatures a natural candidate for representing the state of a delayed system – where it is not a vector but a function segment – in a way that is compatible with standard learning algorithms.

1.1 Background

The control of dynamical systems with delays has a long history in engineering and applied mathematics. Delayed Differential Equations appear in a wide range of fields, from chemical engineering [1] to population dynamics and network control. The theoretical framework for such systems is well established [2], [6], but optimal control methods for DDEs remain computationally challenging.

In parallel, RL has become a dominant paradigm for sequential decision-making [7], [8]. While most RL algorithms operate in discrete time, continuous-time formulations based on the Hamilton-Jacobi-Bellman (HJB) equation have been proposed by K. Doya [9] and R. Munos [10], offering more natural integration with physical systems. The signature transform, introduced by Chen [3] and recently popularized in machine learning [11], [12], [13], provides a principled way to extract features from continuous paths and has shown promising results in non-Markovian settings.

This thesis sits at the intersection of these three fields: it investigates how the signature transform can be used as a state representation within continuous-time RL algorithms to control delayed dynamical systems.

1.2 Problem

Controlling systems governed by DDEs or FDEs with RL poses a fundamental challenge: the state of such a system is not a finite-dimensional vector but a function segment $x_t : [-h, 0] \to \mathbb{R}^d$ representing the recent trajectory. Standard RL algorithms, designed for Markovian environments with vector-valued states cannot directly handle this infinite-dimensional state space. A naive approach – discretizing the trajectory into a high-dimensional vector – introduces a dependency on the discretization step and scales poorly.

1.2.1 Resarch Questions

The core problem addressed in this thesis is whether the signature transform provides an effective and principled state representation for continuous-time

RL applied to delayed dynamical systems. More specifically, we investigate whether approximating the value function and the policy as linear functions of the path signature leads to competitive control performance compared to classical state representations.More specifically, we investigate the following questions:

1. How should the state of a delayed system be encoded using signatures to ensure uniqueness and expressiveness of the representation?
2. How can the temporal derivative of the value function – required by continuous-time RL algorithms – be computed in signature space?
3. What is the impact of the signature depth on the stability and performance of the resulting control algorithms?
4. How do signature-based methods compare to Markovian baselines and augmented-state approaches on systems of varying complexity?

1.3 Purpose

The purpose of this thesis is to develop and evaluate a framework that combines the signature transform with continuous-time RL for the control of delayed dynamical systems. By leveraging the universal approximation property of signatures, we aim to provide a state representation that is both mathematically grounded and practically effective for non-Markovian control tasks.

This work contributes to advancing model-free control methods for systems with memory, which are prevalent in chemical engineering, biology, and networked systems. A successful signature-based approach would offer a representation that is robust to sampling irregularities and time reparametrization, and that does not require explicit knowledge of the system model – properties that are desirable in real-world applications.

1.4 Goals

The goal of this project is to investigate and evaluate the use of signatures as a state representation for continuous-time RL applied to delayed systems. This objective has been divided into the following sub-goals:

1. Develop a theoretical framework for using signatures in continuous-time RL, including the derivation of temporal update rules for signatures and the extension of the HJB equation to functional states.
2. Implement a DDE simulation environment and continuous-time RL algorithms (Continuous Time Actor Critic (CTAC) and Value-Gradient (VG)) with signature-based function approximation.
3. Evaluate the proposed methods on systems of increasing complexity – a linear 2D delayed system, the Mackey-Glass equation, and a nonlinear chemical process – and compare their performance against Markovian and augmented-state baselines.

1.5 Research Methodology

This work follows an experimental research methodology combining theoretical derivation with numerical evaluation. The first phase consisted of a literature review on signatures in machine learning, continuous-time RL, and the control of delayed systems. This provided the theoretical foundations and identified the gap that motivates this thesis.

The second phase focused on the theoretical contributions: deriving the temporal update rule for signatures (Equation 33), extending the HJB value gradient to functional states, and establishing how signatures can serve as function approximations within Doya's CTAC and Value Gradient algorithms.

The third phase was the implementation of the complete framework in JAX, including a custom DDE solver, a rolling signature computation pipeline, and the RL training loop. Finally, the algorithms were evaluated on the three benchmark systems across multiple random seeds, and statistical comparisons were drawn between signature-based and baseline approaches.

1.6 Delimitations

Although the methods developed on this thesis are applying to all kinds of DDEs and FDEs, this thesis focuses on deterministic delayed dynamical

systems with a single constant delay τ. The following aspects are outside the scope of this work:

- Systems with multiple or time-varying delays, or integral (distributed) delays such as Volterra-type equations.
- Stochastic systems or systems with process noise beyond the exploration noise used during training.
- Model-based RL approaches that learn the system dynamics explicitly.
- Deep neural network architectures for function approximation – we restrict ourselves to linear functions of the signature to isolate the contribution of the representation itself.
- Real-time deployment or hardware-in-the-loop experiments.

1.7 Structure of the Thesis

Section 2 presents relevant background information about the topic. Section 3 presents the methodology and method used to solve the problem. Section 6 presents the results of the experiments. Section 7 discusses the implications of the results. Section 8 concludes the thesis and suggests future work.

2 Related Work

2.1 Controlling Non-Markovian systems

Delayed Differential Equations and Functional Differential Equations are omnipresent in lots of tasks and different fields. They play an important role in biology for studying and describing population dynamics, for example with Lotka-Volterra's equations. They are also present in chemistry for Bioreactors and Process Engineering, in Computational Neurosciences, Robotics, Networks and even Traffic Control and Autonomous Vehicles. Yet, the control task is far more difficult than Markovian systems due to the infinite dimensional nature of the problem (the state of the system becomes a function in a Banach space).There has hence been longstanding interest to find solution and controls to such equations. V. Kolmanovskii [2] develops an optimal-control approach to solve the LQR problem for linear FDEs in a finite-time setup. However, the solution is highly non-trivial and requires multiple integration schemes. Following the work of V. Kolmanovskii, E. Friedmann presented Lyapunov methods to analyse such systems [14].

The main limitation of these methods are the need of full knowledge of the model. Reinforcement Learning offers an answer to this issue with approach that are model free or that learns the model to leverage better solutions.B. Chen et al. [15] propose an algorithm to delayed-actions systems that, with knowledge of the delay, offers better performances than State-of-The-Art (SoTA) model-free RL. Y. Wang et al. [16] proposes a variant of the twin-delayed deep deterministic policy gradient (TD3) algorithm by including the delay to better control the suspension of vehicles, and achieving a 30% performance gain compared to passive solutions.

2.2 Continuous Time Reinforcement Learning (CTRL)

Continuous RL was first discussed by K. Doya [9] and R. Munos [17], [10]. The field has gained more attention recently with a paper from C. Yildiz [18] presenting a solution for model based CTRL. However, learning a model can be a complex task especially one of a delayed system. Modern approach rely on Deep Neural network: S. Holt [19] proposes neural networks based on Laplace Transform to learn the dynamics, [20] implement this approach with the signatures, [21] for Neural DDEs. The advantage of K. Doya and R. Munos approach are that they are model free and don't need such fancy models. Other approach exists such as Integral Reinforcement Learning [22] that solve the problem of estimating the temporal derivate in the HJB equation by using the integral version instead. However, these papers present a major drawback (that is not present in Doya's paper): they rely on the gradient of the V function, which is a the core of the HJB equation. Since we want to approximate this with signatures, one will have to define what is the gradient w.r.t. signatures, which is not a trivial thing. Therefore, such approach have to be considered carefully.

Munos also mentions the Policy Search algorithm [10] which consists in parametrizing the policy with a given vector/network α and then moving each parameter slightly (more details in [23] for example). This method would suit us for small systems, but as the signature length scales exponentially with the system dimension (see Equation 8), it would be quite long for deep signatures or more advanced models.

Recent literature presents some ideas for improvement of Doya's algorithm: the use of an offline algorithm seems to be a consensus, and Integral RL shows some great performances. Building upon that, a recent paper from H. Han [24] compiles those ideas into one algorithm being the continuous counterpart of the Soft Actor Critic (SAC) algorithm presented by T. Haarnoja [25].

2.3 Signatures as a Path Representation

2.3.1 Signature in Machine Learning

The signature transform has been first introduced with the work of K.-T. Chen in the 1950s [3], [26], [27]. It has then been rediscovered by T. Lyons [4] in the late 90′s for stochastic analysis and controlled differential equation. If at first the main usage was for finance and rough path theory, it recently gained interest in the machine learning field. P. Kidger et al. [11] have successfully applied the signature transform to data generation, learning from Brownian motion and achieving good performances on Non-Markovian Reinforcement Learning (a topic in which we are particularly interested). The paper presents how signatures with the right augmentations can be used as a layer in a neural network and compares it to RNN approaches such as GRU or LSTM. Building from this idea, [20] proposes an auto-encoder architecture to learn non-markovian dynamics in continuous time, surpassing RNNs approaches again.

The work of A. Fermanian has also been a large source of inspiration for this degree project. In [12], a method to perform functional linear regression is developed. Building from these results, [13] presents an algorithm to predict time series from sparse sampling and access to observables (quantities linked to the time series of interest). The use of signatures allows transforming the problem to a linear regression that is both easier and quicker to solve than SoTA RNN-based methods. In [28], A. Fermanian presents an interpretation of the solution of an RNN as a linear function of the signature of the input sequence, and reframe RNNs as a kernel method. F. Kiraly et al. [29] also build on the kernel theory to present a signature-based sequential kernel, which yields good results and is a good way to preprocess the data.

2.3.2 Comparison to the Fourier Transform

The signature transform is a rather complex object, and one could be tempted to compare it to the Fourier transform in order to gain further intuition. P.

Kidger et al. [11] present the signature transform as being close to the Fourier transform, with the main difference being that Fourier (or wavelet transform) model a path as a linear combination in the functional basis, when signature instead provides a basis of functions. It doesn't aim to "compress" the information contained in a curve, but rather provides a basis to compute easily any function of a curve (see Equation 14).

I. Chevyrev and A. Kormilitzin [30] also emphasizes that signature transform can be an alternative to Fourier transform where instead of focusing on the frequency of the signal, the signatures focus on "moments" (the integration of the product of various coordinates, see Equation 4). However the signature transform is not easily invertible as can be the Fourier transform. First the signature is unique up to path-tree-like equivalence (see [31] for example) which can cause problem in the inversion case. One way to tackle this issue is to augment the signature by rather computing the transform of $\tilde{x} := (t, x)$ (see [11], [30]). An inversion algorithm exists and has been established in [32]. However, the main drawback is that a signature of depth N only provides $N + 1$ points. P. Kidger et al. [11] proposes an optimization algorithm to invert the signature for simple 2D paths, but it is in fact impractical for more complex signals and can be long to obtain a satisfactory result.

2.3.3 Signatures in Control

The representation power of signatures and its convenient structure has attracted attention for control-related tasks. M. Hoglund et al. [33] propose a signature-based approach for controlling non-Markovian stochastic problems via Neural Rough Differential Equations. P. Kidger et al. [11] also presented signature layers to control non-Markovian systems with RL. To be precise, they used the famous Mountain Car from the `gymnasium` RL library and masked the speed in the state representation, only leaving the position. The system must hence rely on previous observation, making it non-Markovian in a way. We could also argue that Takens theorem (see Section 3.2.1) provides the right framework to justify the link between Partially Observable and Delayed Systems.

M. Ohnishi [34] proposed an RL algorithm to perform Model Predictive Control (MPC) using signatures. MPC consists on following a predefined

path, which M. Ohnishi transformed into a signature and specifically designed cost function and Bellman equation for it, both in discrete and continuous time. His solution achieves smoother and more precise results than classical RL methods, even on complex tasks such as controlling robotic arms.

Even more recently, Z. Zhao et al. [35] provided a transformer model with Signature embeddings to perform Offline RL. They achieved better results than SoTA RL methods based on transformers on environments with delayed rewards, illustrating the power of the signature embedding .

3 Background

This chapter describes the research methodology and methods used in this thesis. We present the used algorithms and computations to answer the problems established Section 1.2.1. We will first present how signatures can be seen as an approximation function of the state of the delayed systems, and how it can be used in Reinforcement Learning algorithms.

3.1 Mathematical Foundations of Signatures

3.1.1 Definition of the signature transform

In this thesis, we will consider time-series going from an interval $[a, b] \subset \mathbb{R}$ to $\mathbb{R}^d$, and denote such functions as "paths".

Definition 1. *(path)* A path x is defined by:

$$x : \begin{cases} [a,b] \to \mathbb{R}^d \\ t \quad \mapsto x(t) \end{cases} \tag{1}$$

Here, x is considered to be a piecewise C^1 path, that is x is continuously differentiable.

Let's consider paths x defined in equation Equation 1, we suppose that x is piecewise continuously differentiable (more generally, x must be of finite variation for the following to hold). Remember that $\forall t \in [a, b], x(t) \in \mathbb{R}^d$ and can be written as $x(t) = \left(x^{(1)}(t), ..., x^{(d)}(t)\right)$. The signature being an infinite length vector, we will define it through its coefficients. In the following, we will also use the subscript notation $x(t) = x_t$ with t being any placeholder, especially in integration notations.

We define for any $i \in \{1, ..., d\}$:

$$S(x)^{(i)}_{a,t} := \int_a^t dx^{(i)}_u \tag{2}$$

Note that if x is differentiable, $dx = \dot{x}$, and $S(x)^{(i)}_{a,t} = x(t) - x(a)$. We next define for any $i, j \in \{1, ..., d\}$

$$S(x)^{(i,j)}_{a,t} := \int_a^t S(x)^{(i)}_{a,u} dx^{(j)}_u = \int_a^t \int_a^u dx^{(i)}_r dx^{(j)}_u \tag{3}$$

This recursive definition can be expanded to any word of length $N \in \mathbb{N}$ belonging to the alphabet $\{1, ..., d\}^N$. We hence define in a more general way:

Definition 2. *(signature coefficient)* The signature coefficient on the interval $[a, t] \subset \mathbb{R}$ associated to the word $I = (i_1, i_2, ..., i_N)$ is

$$S(x)^{(i_1,...,i_N)}_{a,t} := \int_a^t S(x)^{(i_1,...,i_N)}_{a,u} dx_u = \int_{a<u_1<...<u_N<t} ... \int dx^{i_1}_{u_1} ... dx^{i_N}_{u_N} \tag{4}$$

For more insights about the meaning of such iterated integral, the reader is referred to [36]

The signature of a path is then the aggregated coefficients ordered by length and lexicographical order.

Definition 3. *(signature of a path)* We note $S(x)_{a,t}$ the signature of path x on the interval $[a, t] \subset [a, b]$.

$$S(x)_{a,t} = \left(1, S^{(1)}(x), S^{(2)}(x), ..., S^{(d)}(x), S^{(1,1)}(x), ...\right) \tag{5}$$

Where here we dropped the subscript for notation simplicity, and the first coefficient is 1 by convention.

We next introduce the truncated signature by first presenting the notion of *depth* and *layers*. We say that a coefficient of the signature $S^I(x)$, $I \in \{1, ..., d\}^N$ is of depth N, and we denote the N-th layer by the set of such coefficients. In other words, the N-th layer of the signature is simply the signature coefficients which are indexed by a word of length N.

Definition 4. *(truncated signature)* The truncated signature is the ordered vector of signature coefficients of depth N or below:

$$S^N(x) = \left(S^I(x)\right)_{I\in\{1,\dots,d\}^k, 0\leq k\leq N} \tag{6}$$

With the convention that for $k = 0$, the coefficient associated is 1. From this definition, we derive the immediate following property:

Property 1. *(truncated signature limit)* Let x be a path,

$$\lim_{N\to+\infty} S^N(x) = S(x) \tag{7}$$

To finish, we insist on the number of coefficients that the truncated signature has and how it scales:

Property 2. *(truncated signature coefficients)* Let x a path as defined in Equation 1. The number of coefficients in the truncated signature $S^N(x)$ is

$$1 + d + d^2 + \dots + d^N = \frac{1 - d^{N+1}}{1 - d} \tag{8}$$

This shows how unpractical it can be to compute signatures of important depths. With $d = 5$ and $N = 5$, the truncated signature already has 3906 coefficients, while with $N = 15$ it reaches more than 10 billions coefficient (roughly 10 GB in memory).

3.1.2 Signature with tensor algebra

To comprehend and manipulate signatures, one can find it best to see them in a tensor space. This section is based on [30] as it gives a clear introduction to such notions in a way that will be useful for signatures.

Definition 5. *(Formal power series)* Let $e_1, \dots, e_d$ be d formal indeterminate. The algebra of non commuting *formal power series* in d indeterminate is the vector space of all formal series in the form

$$\sum_{k=0}^{\infty} \sum_{(i_1,\dots i_k)\in\{1,\dots,d\}^k} \lambda_{i_1,\dots,i_k} e_{i_1} \dots e_{i_k} \tag{9}$$

with $\lambda_{i_1,\dots,i_k} \in \mathbb{R}$, and $e_{i_1}, \dots, e_{i_k}$ are called *monomials*. We note the tensor algebra of $\mathbb{R}^d$ as $\mathcal{T}\left(\left(\mathbb{R}^d\right)\right) := \prod_{k=0}^{\infty} \left(\mathbb{R}^d\right)^{\otimes k}$

We then add a vector space property to the tensor algebra to add multiplication and addition with a scalar.

Property 3. *(Vector space structure)* The addition over the tensor space is defined as, for $c \in \mathbb{R}$:

$$\begin{aligned} &\sum_{k=0}^{\infty} \sum_{(i_1,\dots i_k)\in\{1,\dots,d\}^k} \lambda_{i_1,\dots,i_k} e_{i_1}\dots e_{i_k} + \\ &c\sum_{k=0}^{\infty} \sum_{(i_1,\dots i_k)\in\{1,\dots,d\}^k} \mu_{i_1,\dots,i_k} e_{i_1}\dots e_{i_k} = \\ &\sum_{k=0}^{\infty} \sum_{(i_1,\dots i_k)\in\{1,\dots,d\}^k} \left(\lambda_{i_1,\dots,i_k} + c\mu_{i_1,\dots,i_k}\right) e_{i_1}\dots e_{i_k} \end{aligned} \tag{10}$$

This allows us to define the product $\otimes$ between monomials which is analogous to a concatenation operation.

Definition 6. *($\otimes$ product)* The $\otimes$ product between monomials is defined as the concatenation between them: $e_{i_1}\dots e_{i_k} \otimes e_{j_1}\dots e_{j_l} = e_{i_1}\dots e_{i_k} e_{j_1}\dots e_{j_l}$

3.1.3 Chen's identity

One essential property of signatures is Chen's identity. It allows one to easily manipulates signatures through concatenation of paths.

Definition 7. *(concatenation of path)* Let $x : [a, b] \subset \mathbb{R} \to \mathbb{R}^d$ and $y : [b, c] \subset \mathbb{R} \to \mathbb{R}^d$. The concatenation of x and y is defined as

$$(x * y)(t) = \begin{cases} x(t) & a \le t \le b \\ x(b) + (y(t) - y(b)) & b \le t \le c \end{cases} \tag{11}$$

Theorem 1. *(Chen's identity)* Let $x : [a, b] \subset \mathbb{R} \to \mathbb{R}^d, y : [b, c] \subset \mathbb{R} \to \mathbb{R}^d$ two paths:

$$S(x * y)_{a,c} = S(x)_{a,b} \otimes S(y)_{b,c} \tag{12}$$

A demonstration of Chen's identity can be found in [30], both with and without the use of tensor algebra. This theorem allows us to derive a useful corollary:

Corollary 1. *(Signature inverse in the tensor algebra)* Let $x : [a, b] \subset \mathbb{R} \to \mathbb{R}^d$ be a path, and note $\overleftarrow{x} : [a, b] \to \mathbb{R}^d$ the reversed path such that $\forall t \in [a, b], \overleftarrow{x}(t) = x(b - t)$. Then:

$$S(x)_{a,b} \otimes S(\overleftarrow{x})_{b,a} = \mathbb{1} = (1, 0, ...) \tag{13}$$

with $\mathbb{1}$ the unit tensor being 1 on the first coordinate and 0 on all the others.

3.1.4 The universal nonlinearity theorem

This next theorem is the backbone of this thesis and its motivation. It is a strong result analogous to the Stone-Weierstrass theorem.

Theorem 2. *(Density of the signatures)* Let $\mathcal{P}\big([a, b], \mathbb{R}^d\big)$ the set of paths defined as in Equation 1. Let $x \in \mathcal{P}\big([a, b], \mathbb{R}^d\big)$, and $t \mapsto \tilde{x}(t) = \Big(x(t), t, \frac{x(a)}{b-a}t\Big)$ the augmented path. Then the linear functionals on the signature is dense in the set of function of $x \in \mathcal{P}\big([a, b], \mathbb{R}^d\big)$.

Corollary 2. *(Universal approximation)* Let $\varepsilon > 0$, f a continuous function from $\mathbb{R}^d$ to $\mathbb{R}^m, m \in \mathbb{N}$. For all $x \in \mathcal{P}\big([a, b], \mathbb{R}^d\big)$ their exist L a linear function such that

$$\|f(\tilde{x}) - L(S(\tilde{x}))\| \leq \varepsilon \tag{14}$$

A demonstration can be found in [5] for the one dimensional case. The m dimensional case can be seen as the application of the theorem to m functions, with in the end $f = (f_1, ..., f_m)$

Equation 14 basically states that we can find a matrix θ such that any function of a path over an interval is approximated by $\theta^\top S(x)$. Note however that this is true for the complete signature. Similar results are available with the truncated signature and we report the reader to [13] for more information about this. We also emphasize that it is an existence theorem and not a constructive one. Moreover, using truncated signatures instead of the whole signature can be not expressive enough, hence the possible need of augmentations to the signature or vector θ.

3.2 DDEs and FDEs

Delayed Differential Equations (DDEs) are a more general type of differential equations than ODEs. The last only relies on the current state to define the derivative of the system, where DDEs need information on past states. DDEs are a type of Functional Differential Equations (FDEs), but DDEs only need information of a number $k \in \mathbb{N}$ of past states, where the evolution of the state for FDEs can rely on a part of the current trajectory on a time interval.

Definition 8. *(DDE)* A Delayed Differential Equation is a differential equation where the derivative of the state at time t depends on the state at previous times $t - \tau_k$, where $\tau_k > 0$ are the delays. A DDE can be written as

$$\dot{x}(t) = f(t, x(t), x(t-\tau_1), ..., x(t-\tau_k)) \tag{15}$$

Definition 9. *(FDE)* A Functional Differential Equation is a differential equation where the derivative of the state at time t depends on the trajectory of the system over a time interval $[t-h, t], h > 0$. A FDE can be written as:

$$\dot{x}(t) = F(t, x_t) \quad \text{where} \quad x_t(\theta) = x(t+\theta), \quad \theta \in [-h, 0] \tag{16}$$

The intuition behind this is that DDEs have a "punctual" memory, and FDEs have a continuous one. DDEs are a special kind of FDEs, but the inverse isn't true.

In this work, we try to be as general as we can and propose algorithm that work with FDEs (and therefore DDEs), however solving FDEs is more computationally challenging than DDEs, and a big part of this work has first been done with the latter.

3.2.1 Some properties of DDEs and FDEs

Since this work focuses on Delayed Differential Equations and Functional Differential Equations, we present some results and properties of such objects. The first is a way to solve such equations, and the other is a link between DDEs and Partially Observable Systems.

Theorem 3. *(Existence and uniqueness of solutions)* Define the initial-value problem for a nonautonomous system

$$\begin{cases} x'(t) = f(t, x_t), t \geq h \\ x_h \quad = \varphi \end{cases} \tag{17}$$

with h the initial time and φ a continuous function, the state of the system at time h. Suppose f is continuous and satisfies a Lipschitz condition: for all $a, b \in \mathbb{R}$ and $M > 0$, there is a $K > 0$ such that

$$|f(t, \varphi) - f(t, \psi)| \leq K\|\varphi - \psi\|, a \leq t \leq b, \|\varphi\|, \|\psi\| \leq M \tag{18}$$

Then there exist a unique solution $x(t) = x(t, \varphi)$ for Equation 17.

A demonstration can be found in [6]. This result is essentially an extension to existence and uniqueness of ODEs, and it will prove useful for the next solving algorithm for DDEs.

Property 4. *(Method of steps)* Let a DDE be defined as in Equation 17. We wish to solve the equation on intervals $[t_0 + (k-1)\tau, t_0 + k\tau], k \in \mathbb{N}$ (take $k = 1$ for simplicity). For $t \in [t_0, t_0 + \tau]$, the term $x(t - \tau)$ corresponds to the initial history $\varphi(t - \tau)$, which is a known function.

Hence, for $t \in [t_0, t_0 + \tau]$, the DDE reduces to an Ordinary Differential Equation (ODE):

$$\dot{x}(t) = f(t, x(t), \varphi(t - \tau)) := g(t, x(t)) \tag{19}$$

Since φ is known, g is a known function, and this ODE can be solved using standard numerical or analytical methods. Once the solution $x(t)$ is computed on $[t_0, t_0 + \tau]$, it serves as the known history for the subsequent interval $[t_0 + \tau, t_0 + 2\tau]$. This process is repeated iteratively for each step k, advancing the solution by τ at each stage.

See [37] and [2] for more details. This illustrates the previous theorem about the fact that a DDE is unique up to its history function. The method of steps will also be implemented later in our own DDE solver.

Theorem 4. *(Taken's theorem)* Let M a compact manifold of dimension m, X a smooth (i.e. at least C^2) vector field and $y : M \to \mathbb{R}$ a smooth function, then the map

$$\Phi(X, y)(x) := (y(x), y(\varphi_1(x)), ..., y(\varphi_{2m}(x))) \tag{20}$$

is an embedding, with φ_t is the flow of X, meaning that the manifold thus described is diffeomorph to M.

The original proof can be found on Takens's article [38]. This theorem is a bridge between partially observable and delayed systems, stating that we can reconstruct the state of a partially observed system as delayed observations.

3.2.2 A Hamilton-Jacobi-Bellman equation for Delayed Systems

In [2], V. Kolmanovskii introduces the HJB equation for Functional Differential Equations for finite time cases, that we are going to extend to the infinite time the same way it can be done with HJB without delay.

Theorem 5. *(Functional HJB)* Suppose there are a continuous functional $V^* : [-h, 0] \to \mathbb{R}$ locally Lipschitz and a functional $u^*[-h, 0] \to \mathbb{R}$ satisfying the Carathéodory condition, such that

$$\sup_{u \in \mathcal{U}} \{L_u V^*(x_t) + r(x_t, u)\} = \frac{1}{\tau} V^*(x_t) \tag{21}$$

with

$$L_u V^*(x_t) = \limsup_{\Delta t \to 0} \frac{1}{\Delta t} (V(x_{t+\Delta t}(s; x_t, u)) - V(x_t(s))) \tag{22}$$

for all $x_t \in C([-h, 0])$. Then u^* is the optimal control

This and the operator L_u are a crucial thing to comprehend: the subscript u denotes here that the derivation does depend on the command u since it dictates how the system will evolve during the small added time Δt

3.3 Reinforcement Learning

This section covers the RL part and details how it can be used to control the studied systems. As we study Delayed Differential Equations – and continuous systems in a more general way – we wanted to propose a Continuous

Time algorithm. This suits well with the signature structure, which is best at handling continuous path (or interpolated ones).

We first present the HJB equation, upon which is build the RL theory and more specifically the continuous RL.

3.3.1 Optimal Control and the HJB equation

Optimal control is the theory behind LQR [39]. It allows to find the command of a system such that it minimizes a given criterion. For example in LQR, one whishes to minimize a pondered sum between the squares coefficients of the current state of the system and the command. To simplify, we only consider the deterministic case as it also is the one we're working with. This section considers Control Equations of the form

$$\dot{x} = f(t, x(t), u(t)) \tag{23}$$

where f is $\mathcal{C}^1$ of $\mathbb{R}^{1+n+m}$ to $\mathbb{R}^n$. We then define the functional objective.

Definition 10. *(Functional objective)* let $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}, \tau > 0$. We define the objective function of the system of Equation 23 as J such that

$$J(x(t), u(t)) = \int_t^\infty e^{\frac{t-s}{\tau}} r(x(s), u(s)) ds \tag{24}$$

This allows us to then define the Value Function of the system.

Definition 11. *(Value function)* Let J the objective function of the system of Equation 23, the optimal Value Function is defined as:

$$V^*(x(t)) = \sup_u J(x(t), u(t)) \tag{25}$$

Given u an appropriate control, we also define the associated Value Function as

$$V^u(x(t)) = J(x(t), u(t)) \tag{26}$$

The value function is essential for control, as it enables policy improvement. In the infinite horizon setting, the exponential discount factor ensures the convergence of the integral and guarantees the well-definedness of both the Value and Objective functions.

We then express the Dynamic Programming principle that allows to derive the HJB equation.

Theorem 6. *(Dynamic programming principle)* Let $\Delta t > 0$

$$V^*(x(t)) = \sup_{u[t,t+\Delta t]} \left[\int_t^{t+\Delta t} e^{\frac{t-s}{\tau}} r(x(s), u(s)) ds + V^*(x(t+\Delta t)) \right] \tag{27}$$

This theorem basically states that the optimal path from one point at time t is the one that minimizes the criterion from t to $t + \Delta t$ and then the optimal one starting at $t + \Delta t$. We can now present the HJB equation:

Property 5. *(Hamilton-Jacobi-Bellman)* Suppose V^* is regular, then V^* is solution of the equation:

$$\cancel{\frac{\partial V^*}{\partial t}} + H(t, x, \nabla_x V^*(t, x)) = \frac{1}{\tau} V^*(x(t)) \tag{28}$$

with H the system Hamiltonian such that:

$$H(t, x, \nabla_x V^*(t, x)) := \sup_{q \in \mathbb{R}^d} \{q \cdot \nabla_x V^*(t, x) + r(x(t), q)\} \tag{29}$$

Note that since we are studying stationary problem in this thesis, V^* does not depend directly on t hence the term $\frac{\partial V^*}{\partial t}$ is cancelled. This definition of the Hamiltonian will not prove useful for this thesis, but is essential to understand the used RL algorithms as they are all based on it. More can be found on these lectures about optimal control and dynamic programming [40], [41].

3.3.2 Continuous Reinforcement Learning

One drawback of RL is its discrete nature that can lead to poor control in certain cases [18]. Moreover, Signatures are well-suited to handle continuous function, as seen in the previous part. This justifies the use of Continuous RL, built on Kenji Doya's work [9], for this work. The drawback of discrete – classical – RL is that most learning algorithm become unstable if the discretization step goes to 0 (see [10]). K. Doya [9] presents a CTAC algorithm

Algorithm 1: Continuous Time Actor Critic

Initialisation: actor function A with weights w_a and critic function V with
1 weights w_c, noise std σ, discretisation Δt, environment with step function dependent of Δt.

2 **while** not finished

3 $n \leftarrow$random noise, std $= \sigma$

4 $x_{t+\Delta t}, r_t \leftarrow \text{step}_{\Delta t}(x_t, u + n)$

5 compute $V(x_t)$ and $V(x_{t+\Delta t})$

6 $\delta_t \leftarrow \frac{V(x_t) - V(x_{t+\Delta t})}{\Delta t} + r_t - \frac{1}{\tau} V(x_t)$ *temporal difference*

7 $w_c \leftarrow w_c + \eta_c \delta_t \frac{\partial V(x_t)}{\partial w_c}$

8 $w_a \leftarrow w_a + \frac{n}{\sigma} \frac{\partial A}{\partial w_a}$

that uses the HJB equation to solve the problem, where discrete RL schemes use its discretized version also known as the Bellman equation.

The algorithm uses the temporal derivative of the V function. Note that it is still a discretized algorithm, but the paradigm here is to discretize an algorithm based on HJB, where classical RL is about discretizing the HJB equation and then establishing an algorithm.

4 Approach

This chapter presents how we derived and developped the CTRL algorithms and the different mathematical proofs associated. It is the theoretical core of our work, and what we will be implementing in Section 5.

4.1 Signatures as a function approximation

Let $\dot{x} = f(x_t, u)$, f being regular, a Delayed Differential Equation as defined in Equation 15 and let τ be the maximum delay of this equation. As we have seen in Equation 17, for a history function $\varphi : [-h, 0] \to \mathbb{R}^d$, with $h \geq \tau$, there exist a unique solution to Equation 15. This allows us to define the state of the system $\dot{x} = f(x_t, u)$ as a history function on an interval $[t - h, t]$, from which we know that the system will have unique behaviour (this is the FDE version of Lipschitz Theorem for ODEs). This means that the state can be defined as the functional $x_t : [-h, 0] \to \mathbb{R}^d, x_t(\cdot) = x(t + \cdot)$. Defining such a state guarantees that we stay on the same interval $[-h, 0]$ which is essential for signature approximation.

The Approximation Theorem over signatures (Equation 14) then states that any function of a path on the interval $[-h, 0]$ can be approximated as a linear function of its signature. Moreover, [2] defines the value function and the optimal actor (in the LQR case for FDEs) as a function of the state x_t. This justifies the idea to use the signatures as a function approximation for the control task. Any function of the path of the system $x_t : [-h, 0] \to \mathbb{R}^d$ can hence be approximated by a matrix θ times the signature of x_t. However, this is true for the real – and infinite – tensor that is the signature. Because of the limitations of computers, we will hence rely on truncated signatures. A; Fermanian and L. Bleinstein [12], [13] have argued how shallow signatures already provide good approximators, justifying our rationale.

4.1.1 Approximating the Value function

In Reinforcement Learning, the Value Function V determines the expected reward of a policy π at a state s. Here we define our state as the signature of the augmented path $\tilde{x}_t = \left(x_t, -\frac{x(0)}{h}t\right)$ with t taking values in $[-h, 0]$, the relative time with respect to our observation window. As detailed in Equation 14, this ensures that the signature is describing a unique path, as we have both a monotonically increasing or decreasing component ($-\frac{x(0)}{h}t$) and information on the origin of the path with $x(0)$. We removed the time dependency as it is redundant with $-\frac{x(0)}{h}t$ and since it doesn't get to 0 when x_t is, it can lead to unstabilities in the learning process with the trajectory not going quite to 0. Note that $-\frac{x(0)}{h}t$ is simply choosing $a = 0$ and $b = -h$ in Equation 14.

We represent the linear function as a scalar product between the signature and a vector θ of appropriated size. When working with truncated signatures, this scalar product is the canonical one since every vector is of finite dimension. It can therefore be written as

$$\langle \theta, S^N(x_t) \rangle = \theta^\top S^N(x_t) \tag{30}$$

When working with non-truncated – and hence infinite length – signatures, this scalar product is an infinite sum:

$$\langle \theta, S(x_t) \rangle = \sum_I \theta_I S^I(x_t) \tag{31}$$

with I being here the signature coefficient, $I \in \{1, ..., d\}^n, n \in \mathbb{N}$. This sum is finite for θ bounded since the signature coefficients are bounded by $\frac{L^k}{k!}$, with k the layer of the coefficients and L the total variations of the path x_t (see Section A.3 for definition), assumed to be finite here. This is not a scalar product per se, but is indeed a linear functional of the signature.

4.1.2 Approximating the Actor

The actor is roughly the same problem as the Value Function. However, the value function returns a scalar where we want a $m \in \mathbb{N}$ input as a command. We therefore define

$$\begin{pmatrix} \langle \phi_1, S(x_t) \rangle \\ \langle \phi_2, S(x_t) \rangle \\ \vdots \\ \langle \phi_m, S(x_t) \rangle \end{pmatrix} \tag{32}$$

with ϕ_i of appropriated size, finite in the case of a truncated signature (which is what is implemented in practice).

Note that the actor doesn't have the same loss function as the critic, and from Doya's CTAC algorithm (Algorithm 1), the update is essentially driven by the direction of the noise. This then becomes a key component of the code (more details in the next part).

4.2 Algorithms

In this part, we detail the algorithms used and their implementation. Although more advanced algorithms exist for Continuous Time Reinforcement Learning, many require sophisticated technical implementations or rely on model-based approaches. We therefore focus on adapting the foundational work of K. Doya [9], which provides a principled and practical framework for our purposes.

4.2.1 Continuous Time Actor Critic

Doya's algorithm – as detailed in Algorithm 1 – presents an actor-critic structure that is particularly interesting since it is model it only requires the derivative of V with respect to time. This derivative can be acquired by two different methods: finite difference or tensor operations. We are going to detail the tensor approach and how it can be used for updating the CTAC algorithm.

In next part, we will detail other methods that rely on the derivative of V w.r.t. the current state, but those are more advanced and need knowledge of the system that might not always be accessible. The advantage of this solution is that it is totally *model free* and doesn't need any a priori knowledge. However, it is detailed by K. Doya to be unstable on complex system as the inverted pole .[9].

Property 6. *(Signatures Time update rule)* Let $x_t : [-h, 0] \to \mathbb{R}^d$ the current state of the system, and $\tilde{x}_t$ its augmented counterpart (see Equation 14). The temporal derivative of the signature can be written as:

$$\frac{d}{dt} S(\tilde{x}_t) = S(\tilde{x}_t) \otimes \dot{\tilde{x}}(t) - \dot{\tilde{x}}_t(t-h) \otimes S(\tilde{x}_t) \tag{33}$$

For the proof, we will drop the $\sim$ over the x_t for clarity. Note that it doesn't change the demonstration to do it on the augmented parh or the regular path.

Proof. Let $\Delta t < h$. Since the path is only defined on $[-h, 0]$, we consider that $x_{[0,\Delta t]}$ is the extension of x_t respecting the fact that x is a solution of the dynamics. Using Chen's identity, we can develop the signature of the path over the shifted interval:

$$\begin{aligned} &S_{[-h,-h+\Delta t]}(x_t) \otimes S_{[-h+\Delta t,\Delta t]}(x_t) = S_{[-h,0]}(x_t) \otimes S_{[0,\Delta t]}(x_t) \\ \Leftrightarrow\ &S_{[-h+\Delta t,\Delta t]}(x_t) = S_{[-h,-h+\Delta t]}(\overleftarrow{x}_t) \otimes S_{[-h,0]}(x_t) \otimes S_{[0,\Delta t]}(x_t) \end{aligned} \tag{34}$$

With $S_{[a,b]}\left(\overleftarrow{f}\right)$ being the signature of the inverse path, namely the path from b to a (which corresponds to the group inverse in the tensor algebra, see Equation 13).

We perform a first-order Taylor expansion for the infinitesimal segments (more details in Section A.1):

$$\begin{aligned} S_{[-h,-h+\Delta t]}(\overleftarrow{x}_t) &= \mathbb{1} - \dot{x}(t-h)\Delta t + o(\Delta t) \\ S_{[0,\Delta t]}(x_t) &= \mathbb{1} + \dot{x}(t)\Delta t + o(\Delta t) \end{aligned} \tag{35}$$

With $\mathbb{1}$ the unit tensor, i.e. the tensor with a 1 in first position and infinitely many 0 after ($\mathbb{1} = (1, 0, 0, ...)$). Note that $\mathbb{1}$ is therefore the unit element w.r.t. the tensor algebra $\mathcal{T}\left(\left(\mathbb{R}^d\right)\right)$.

Substituting these into Equation 34 and expanding the tensor product:

$$\begin{aligned} S_{[-h+\Delta t,\Delta t]}(x_t) &= (\mathbb{1} - \dot{x}(t-h)\Delta t + o(\Delta t)) \\ &\otimes S_{[-h,0]}(x_t) \otimes (\mathbb{1} + \dot{x}(t)\Delta t + o(\Delta t)) \\ &= \Big(\mathbb{1} \otimes S_{[-h,0]}(x_t) \otimes \mathbb{1}\Big) \\ &-\Big(\dot{x}(t-h)\Delta t \otimes S_{[-h,0]}(x_t)\Big) \\ &+\Big(S_{[-h,0]}(x_t) \otimes \dot{x}(t)\Delta t\Big) + o(\Delta t) \end{aligned} \tag{36}$$

Subtracting $S_{[-h,0]}(x_t)$ and dividing by Δt:

$$\begin{aligned} \frac{S_{[-h+\Delta t,\Delta t]}(x_t) - S_{[-h,0]}(x_t)}{\Delta t} &= -\Big(\dot{x}(t-h) \otimes S_{[-h,0]}(x_t)\Big) \\ &+\Big(S_{[-h,0]}(x_t) \otimes \dot{x}(t)\Big) + o(1) \end{aligned} \tag{37}$$

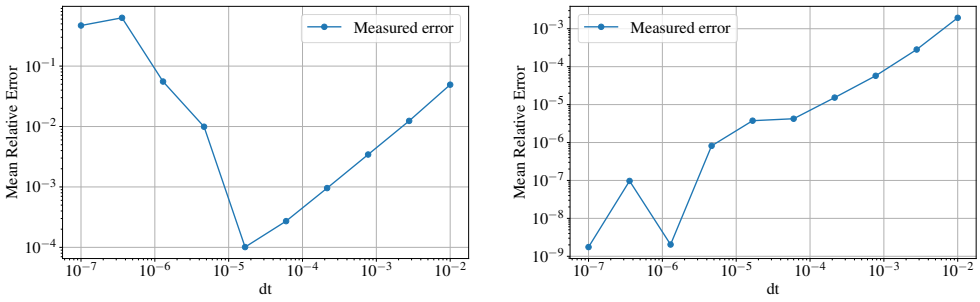
Identifying the terms with the derivative of the sliding window signature $S(x_t)$, denoted as $S(x_{t+\Delta t})$, and letting $\Delta t \to 0$, we finally obtain:

$$\frac{d}{dt}S(x_t) = -\dot{x}(t-h) \otimes S(x_t) + S(x_t) \otimes \dot{x}(t) \tag{38}$$

□

The problem with Equation 33 is that it requires the knowledge of $\dot{x}$ which can be accessed with finite difference, and tensor operations. In practice, a simple Euler scheme between $S(x_t)$ and $S(x_{t+\Delta t})$ is enough to have a good estimation of $\frac{d}{dt}S(x_t)$.

To validate this result, we compare the tensor operation of Equation 33 with the described finite difference schemes. Note that this result was derived for infinite size tensor but is still true for the truncated version of the tensor product $\otimes_N$, which is what we use here with N the depth of the signature. More detail in Section A.2. We show the obtained graphs of the relative error for two paths: $x_t = (\sin(2\pi t), \cos(2\pi t))$ and $x_t = (0.1\exp(t), \exp(-t^2))$ for $t \in [0, 1]$ Figure 1. The error tends to decrease as the time step does, with some instabilities beginning when dt reduces to small values, which could be expected for such a numerical scheme.



(a) $x_t = (\sin(2\pi t), \cos(2\pi t))$ (b) $x_t = (0.1\exp(t), \exp(-t^2))$

Figure 1: Evolution of the relative error between finite difference scheme and Equation 33 w.r.t. dt the discretisation step

4.2.2 Value Gradient

In its original paper, K. Doya also describes a Value Gradient approach to the problem: the goal is to only learn the critic $V(x)$ and compute the command $u(x)$ from its value. This technique is quite well-known in the adaptative programming community (see [42] for example).

For an LQR problem, the value function is directly given by $V(x) = x^\top Px$ with P solution of the Ricatti equation of the system. Since the optimal control is given by $u^*(x) = -Kx = -R^{-1}B^\top Px$. Knowing R and B and finding V with RL methods, it then suffices to compute $\frac{\partial V}{\partial x}\big|_x$ to get u^*. K. Doya generalizes this approach to a more generic reward function than the LQR $x^\top Qx + u^\top Ru$.

Assuming the reward function has shape

$$r(x,u) = Q(x) - \sum_{j=1}^{m} s_j(u_j) \tag{39}$$

The HJB Equation 28 then becomes in this case:

$$\max_u \left(Q(x) - \sum_{j=1}^{m} s_j(u_j) + \frac{\partial V^*}{\partial x} f(x,u) \right) = \frac{1}{\tau} V^*(x(t)) \tag{40}$$

As we want a maximum, the optimal command is the one such that the derivative of Equation 40 is 0, in other words:

$$-\sum_{j=1}^{m} {s_j}'(u_j)) + \frac{\partial V^*}{\partial x}\frac{\partial f(x,u)}{\partial u} = 0 \tag{41}$$

then assuming that the input gain is **linear** w.r.t. u and that for all j, s_j is **convex**, Equation 41 has a unique solution:

$$u_j = {s_j}'^{-1}\left(\frac{\partial V^*}{\partial x}\frac{\partial f(x,u)}{\partial u}\right) \tag{42}$$

The previous assumptions can seem strong, but are in fact consistent with most of the physical system one can encounter. Since in our case we use $s(u) = u^\top Ru$ with R symmetric definite non-negative, the convexity condition is verified and $s'(u) = 2Ru$. However, this has to be refined for delayed systems, since our state is no more a vector of $\mathbb{R}^d$, but a vector of a Banach space, the space of continuously differentiable functions over a closed interval. Such derivations exist in the literature [43] and differs from the non-delayed case by adding a term depending on the integral of the path. With the signature, this derivation can be generalized and simplified, since signatures encodes the integral. We will then try to provide a scheme idea for such systems.

Property 7. *(Signature Value Gradient with delayed systems)* Let $\dot{x} = f(x_t, u)$ with $x_t : s \mapsto x(t+s) \in \mathbb{R}^d$, suppose $f(x_t, u)$ has shape $F(x_t) + g(x_t)u$ i.e., the input gain is not dependent on u. Suppose also that the reward has shape $x^\top Qx - u^\top Ru$ with $Q \in \mathcal{M}_{\mathbb{R}^d \times \mathbb{R}^d}$, R symmetric definite non-negative, and the Value Function can be expressed as $V(x_t) = \langle \theta, S(x_t) \rangle$. Then the optimal command of the system is defined as

$$u^* = \frac{1}{2}R^{-1}g(x_t)^\top \nabla_{x(t)} V(x_t) \tag{43}$$

with $\nabla_{x(t)} V(x_t)$ the derivative of the Value Function at the extremity of the path $x(t)$

Proof. Since the Value Function depends on the signature, we can write the HJB equation as

$$\max_u \Big(x^\top Q x - u^\top R u + \dot{V}(x_t) \Big) = \frac{1}{\tau} V(x_t)$$

$$\Leftrightarrow \max_u \big(x^\top Q x - u^\top R u + \langle \theta, S(x_t) \otimes \dot{x}(t) - \dot{x}(t-h) \otimes S(x_t)) \rangle = \frac{1}{\tau} V(x_t) \quad (44)$$

$$\Leftrightarrow \max_u \big(x^\top Q x - u^\top R u + \langle \theta, S(x_t) \otimes (f(x_t) + g(x_t)u) - \dot{x}(t-h) \otimes S(x_t)) \rangle \big) = \frac{1}{\tau} V(x_t)$$

using the time-update rule Equation 33. We therefore want

$$\arg\max_u \big(-u^\top R u + \langle \theta, S(x_t) \otimes (g(x_t)u) \rangle \big) \tag{45}$$

removing everything that is non-dependent of u.

Note that $\dot{x}_t(t-h) \otimes S(x_t)$ does only depend on past times and is not influenced by the control at time t, explaining why it is removed from the expression. We now want to compute at which point the gradient of what's inside the parenthesis is 0.

We need to understand what the function $v \mapsto \langle \theta, S(x_t) \otimes v \rangle$, with $v \in \mathbb{R}^d$ represents. In our case, $v = g(x_t)u$ is the signal that will drive the equation and input change on the signature. We will show that it can be interpreted as the gradient of the signature at the extremity $x(t)$.

Let $\varepsilon > 0$ and $v \in \mathbb{R}^d$. Define

$$\begin{aligned} \gamma_\varepsilon : [0, \varepsilon] &\to \mathbb{R}^d \\ s &\mapsto x(t) + sv \end{aligned} \tag{46}$$

the path $x_t * \gamma_\varepsilon$ is then x_t extended by a vector in a direction v. Using Chen's Identity (Equation 12) we can write

$$S(x_t * \gamma_\varepsilon) = S(x_t) \otimes S(\gamma_\varepsilon) \tag{47}$$

$x_t * \gamma_\varepsilon$ is C^1 by part and continuous, justifying the existence of the signature. Using similar arguments as in the proof of Equation 33, we can approximate $S(\gamma_\varepsilon) = \mathbb{1} + \varepsilon v + o(\varepsilon)$. Equation 47 then becomes

$$\begin{aligned} S(x_t * \gamma_\varepsilon) &= S(x_t) \otimes (\mathbb{1} + \varepsilon v + o(\varepsilon)) \\ &= S(x_t) + \varepsilon (S(x_t) \otimes v) + o(\varepsilon) \end{aligned} \tag{48}$$

Subtracting $S(x_t)$ on both sides, dividing by epsilon and taking the limit, we get an expression of the directional derivative for vector v at the tip of the path:

$$\lim_{\varepsilon \to 0^+} \frac{S(x_t * \gamma_\varepsilon) - S(x_t)}{\varepsilon} = S(x_t) \otimes v \tag{49}$$

This clarifies the meaning of the quantity $S(x_t) \otimes (g(x_t)u)$: the map $v \mapsto \langle \theta, S(x_t) \otimes v \rangle$ is a continuous linear functional on $\mathbb{R}^d$. Since x_t is a piecewise C^1 function on a compact support (here $[-h, 0]$), it belongs to the Hilbert space of such functions. By the Riesz Representation Theorem, there exists $\nabla_{x(t)} V(x_t) \in \mathbb{R}^d$ such that for all v in $\mathbb{R}^d$,

$$\langle \theta, S(x_t) \otimes v \rangle = \langle \nabla_{x(t)} V(x_t), v \rangle_{\mathbb{R}^d} = \nabla_{x(t)} V(x_t)^\top v \tag{50}$$

Therefore, differentiating this expression w.r.t. u we get

$$\frac{\partial}{\partial u} \langle \nabla_{x(t)} V(x_t), g(x_t)u \rangle_{\mathbb{R}^d} = g(x_t)^\top \nabla_{x(t)} V(x_t) \tag{51}$$

Finally,

$$\begin{aligned} &-2Ru^* + g(x_t)^\top \nabla_{x(t)} V(x_t) = 0 \\ \Leftrightarrow\ &u^* = \frac{1}{2} R^{-1} g(x_t)^\top \nabla_{x(t)} V(x_t) \end{aligned} \tag{52}$$

□

This derivation is inspired and related to the work of B. Dupire on Functional Itô Calculus [44], where derivatives of path-dependent functionals are decomposed in vertical and horizontal components. Here, Chen's identity provide a direct way to compute the endpoint sensitivity without requiring the abstract framework of Functional Itô Calculus.

In practice, we can compute $\nabla_{x(t)} V(x_t)$ by backpropagating $S(x_t)$ relative to the whole path x_t, and then evaluate the obtained function in 0 which is equivalent of taking the last element of the returned vector. The full process is presented Algorithm 2.

Algorithm 2: Functional Value Gradient

Initialisation: Critic function V with weights w_c, noise std σ, discretisation Δt, environment with step function dependent of Δt, matrix $g(x_t)$, learning rate $\eta_c > 0$.

while not finished

- $\nabla_{x(t)} V(x_t) \leftarrow \frac{\partial V(x_t)}{x_t} \mid_{t=0}$
- $u \leftarrow \frac{1}{2} R^{-1} g(x_t)^\top \nabla_{x(t)} V(x_t)$
- $n \leftarrow$random noise, std $= \sigma$
- $x_{t+\Delta t}, r_t \leftarrow \text{step}_{\Delta t}(x_t, u + n)$
- compute $V(x_t)$ and $V(x_{t+\Delta t})$
- $\delta_t \leftarrow \frac{V(x_t) - V(x_{t+\Delta t})}{\Delta t} + r_t - \frac{1}{\tau} V(x_t)$ *temporal difference*
- $w_c \leftarrow w_c + \eta_c \delta_t \frac{\partial V(x_t)}{\partial w_c}$

5 Implementation and Case Study

This chapter describes the technical implementation of the aforementioned algorithms as long with the studied examples.

5.1 Simulation Environment

Since we are working with DDEs, we need an environment able to simulate such trajectories with the addition of a command input. This part presents the different technical details of our implementation. With this solver, we focussed on practicability and simplicity. Because of that, our solver might not be the most accurate but it is fast and precise enough to fit our goals.

5.1.1 DDE solver

There exists great implementation of DDE solvers in python such as `JiTCDDE` [45] which performs Just In Time compilation and hence provide a fast framework. Sadly, this library is incompatible with the kind of control that we want to add. `JiTCDDE` uses Hermite Splines Curves to interpolate the solution, which smoothen the noise that we might add to the control task for exploration. The RL algorithm hence receives a different signal from what the solver will use. This was preventing the convergence of the first algorithms, and required that we implemented our own solver. We used the method of steps presented in [6] with a Runge-Kutta (RK) 4 scheme, which was enough to lead to precise results. The architecture was done in a modular way such that changing the dynamics simply requires inheriting from a python class. The RK4 scheme is presented in Algorithm 3. RK schemes for delayed systems requires knowledge of the trajectory between the computed points, which is why `JiTCDDE` uses interpolation. We then decided to use a linear one for this

Algorithm 3: RK4 scheme

Initialisation current state x_t, command u, dynamics described by the function $f(x_t, u)$, solver step size δt

$k_1 \leftarrow f(x_t, u)$

$k_2 \leftarrow f\left(x_t \frac{k_1 h}{2}, u\right)$ (computed with linear interpolation)

$k_3 \leftarrow f\left(x_t \frac{k_2 h}{2}, u\right)$ (computed with linear interpolation)

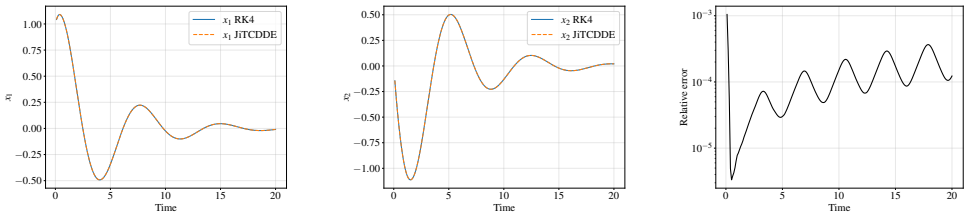
$k_4 \leftarrow f(x_t k_3 h, u)$ (computed with linear interpolation)

Return: $x_t(0) + \frac{\delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

matter: it doesn't impact the input noise as much as Hermite polynomials and worked properly with our RL algorithms.

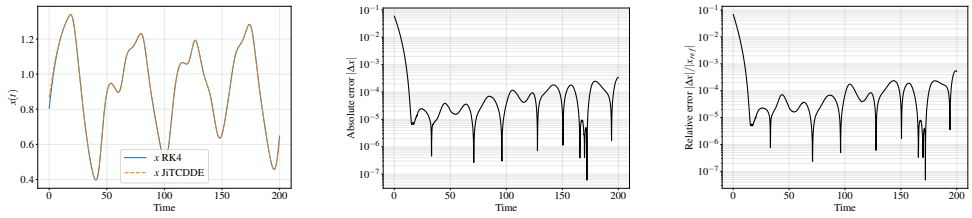
Because the solver uses linear interpolation, part of the theoretical accuracy of the RK4 scheme is lost. Since solver precision was not the primary focus of this thesis and the results remained satisfactory, we retained this design choice. Figure 2 compares our solver with `JiTCDDE` on a 2D delayed system, and Figure 3 reports the same comparison for the Mackey-Glass equation (see Equation 56). In both cases, the error remains small, indicating that our implementation is consistent for the uncontrolled dynamics.

The largest errors appear at the beginning of the simulation. This is expected: the initial history is constant, which makes the derivative at $t = 0$ ill-defined. `JiTCDDE` includes a dedicated discontinuity-handling mechanism while our implementation does not. This difference is barely visible on the 2D system but more pronounced for Mackey-Glass. Finally, the relative error increases over time, as local numerical errors accumulate during integration.



(a) x_1 (b) x_2 (c) relative error

Figure 2: Comparison of `JiTCDDE` solver with our own implementation of RK4 for a 2D delayed system



(a) comparison (b) absolute error (c) relative error

Figure 3: Comparison of `JiTCDDE` solver with our own implementation of RK4 over the Mackey-Glass equation

5.1.2 Reinforcement Learning Environment

In order to implement Reinforcement Learning algorithms, we need an environment which, being in a state s and taking a command u at a time t, returns a reward r and the next state s'. We based the coding architecture on what is proposed with the `Gymnasium` Library, with a `step` function which takes as arguments the current state, action and returns the next state and reward.

This implementation as long with the solver is pure JAX: all the RL algorithms are using JAX for optimization and learning task. Ensuring that the whole framework is JAX avoids slowing and unnecessary conversion between Numpy and JAX that would go back and forth between CPU and GPU.

5.2 Rolling Signatures

The computation of the signatures is rather efficient and multiples library like `iisignature` [46] or `signatory` [47] provide strong python options. `signatory` is the most interesting here since it is compatible with PyTorch, and hence with backpropagation. However, due to the fact that it only runs on deprecated versions of Python, we decided to focus on the JAX implementation of `signatory`, called `signax` that can be found on github: https://github.com/anh-tong/signax. This version is pure JAX and is as fast as `signatory` while still being compatible with JAX own backpropagation. This was the choice that made the code framework JAX-based [48]. We also rely on the JAX optimization library `optax` [49] and `flax` [50] for neural networks.

Our state is the signature of the path $x_t[-h, 0] \to \mathbb{R}^d$. `signax` takes the discretized version of such a path and returns its signature. In practice, since

Algorithm 4: Rolling Signatures

Initialisation $\mathfrak{B}$ buffer of size L, signature depth d
while episode not terminated:
  get the next position x
  for $i = 1...N$:
    $\mathfrak{B}[i-1] = \mathfrak{B}[i]$
  end for
  $\mathfrak{B}[-1] = x$
  Compute the signature of depth d over path stocked in $\mathfrak{B}$
end while

we have to perform this operation each time we encounter a new state, we implement a rolling strategy described in Algorithm 4. At each new position, a buffer is updated and the signature is recomputed. This strategy simplifies the workflow but can become a bottleneck for deeper signatures.

5.3 Technical implementation of RL algorithms

5.3.1 Approximation functions

One goal of this study is to use the representation capacity of the signatures in order to make control algorithms. We will not use any form of deep neural networks with multiple layers in this study. Hence, the approximated value function $\hat{V}$ is modelized as a matrix multiplication:

$$\hat{V} = \theta^\top H(x_t) \tag{53}$$

where H is a feature function:
- for signature-based algorithm, H is the signature transform:

$$H(x_t) = S^N_{[-h,0]}(x_t) \tag{54}$$

with N the chosen depth of signature, and h being the last timepoint that is taken for the signature. Except when specified the opposite, we usually take $h = \tau$ the delay in the system. We then have $\theta \in \mathcal{M}_{1\times M}(\mathbb{R})$ with M the corresponding size of the signature (with all the augmentations added).

- for Markovian algorithms (based on the current state), we choose $H(x_t) = \big(x_i(t)x_j(t)\big)_{i,j\in\{1,\dots,d\}}$ as an analogy to the LQR problem where the optimal value function is quadratic.
- for algorithms that use the whole trajectory, the derivation of [2] of Value function for delayed systems shows that it is the sum of a quadratic function of the position $x(t)$and multiple quadratic functions between current position $x(t)$ and previous positions $x(t+\theta), \theta \in [-h, 0]$. We then chose $H(x_t) = \big(x_i(t-k_1\delta t)x_j(t-k_2\delta t)\big)_{(i,j\in\{1,\dots,d\},k_1,k_2\in\{0,\dots,\lfloor\frac{h}{\delta t}\rfloor\}}$ with δt the discretization step. This hypothesis class contains the actual Value Function, but suffers from curse of dimensionality. We then have $\mathcal{M}_{1\times L}(\mathbb{R})$ with L the corresponding size of the feature function H.

5.3.2 Tweaks and Tricks

Reinforcement Learning are notorious for being unstable and ours are not escaping this rule of thumb. In his paper, [9] Kenji Doya details that the actor critic is quite unstable compared to the value gradient actor. We detail the implemented tricks to keep the algorithm converging and try to avoid these instabilities.

- **Smooth noise**: We realized that if the noise is too aggressive, it can lead to instabilities in the learning process. It was hence decided to use a Gaussian Process as a noise, whose evolution for the next step relies on the previous, which isn't the case for white/Gaussian noise. We use a squared exponential kernel of form $K(t,t') = \sigma^2 \exp\Big(-\frac{|t-t'|^2}{(2l^2)}\Big)$ with l a chosen scale, and σ the amplitude of the noise we want.
- **Decaying noise**: The amplitude of the noise σ decays linearly w.r.t. the episodes of the algorithm. This ensures exploration during the first episodes and exploitation at the end of the training.
- **Checkpoint saving**: We observed that during training, the algorithm could reach an optimum and then start to dramatically unlearn/fit noise. The

decaying noise helped a lot to prevent this, but to ensure we get the best solution out of the training, we decided to save the networks parameters each n step if it was performing better than the previous saved parameters.

- **Preheat & Burning steps**: the signature is taking a window between times $[t-h, t]$. However when $t = 0$, the window is hence taking the initial condition to compute the signature (supposing the system is indeed starting at $t = 0$). This causes troubles in the learning process as the signature can exhibit very different behaviours for $t \in [0, h]$ and after. In the 2D problem, it was quickly detected that this was preventing convergence. The algorithm was learning different things on this "transient part" rather than on the real dynamics. We simply decided to cut it off and only start the learning process at $t = h$, or even a bit more steps after to ensure proper learning. This is referred as the **preheat** phase and **burning** phase.

5.4 Test cases

This part presents the various systems used for testing the algorithms. Each of them have some specificities and limitations that we are going to discuss further on. We will test the systems with different agents:

- A signature-based agent with state being $S^{N(\hat{x}_t)}$ the truncated signature of depth N of the augmented path (see Equation 14)
- A vanilla agent with only the current state $x(t)$
- An augmented agent with state the discretized function x_t
- The Value-Gradient agent with state being $S^N(\hat{x}_t)$

The goal here is to see for each agent if it is able to stabilize the system, and compare its performances to the others.

5.4.1 2D system with delay

We first try our approach on a simple 2D system with delay. The evolution equation is

$$\dot{x}(t) = -\frac{1}{10}\begin{pmatrix}5 & 1\\ 0 & 5\end{pmatrix}x(t) + \frac{1}{10}\begin{pmatrix}1 & 1\\ 0 & 2\end{pmatrix}x(t-\tau) + \begin{pmatrix}0\\ 1\end{pmatrix}u(t) \qquad (55)$$

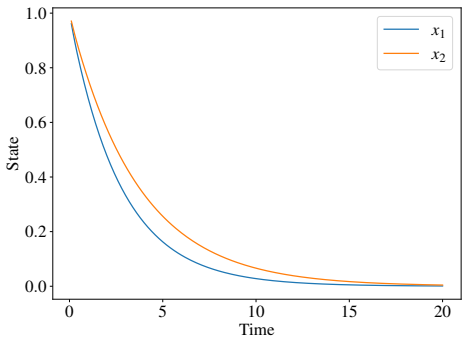


Figure 4: 2D system Equation 55 trajectory

We take $\tau = 1$. As one can see, this system is a rather simple and is asymptotically stable. Yet it provides a nice test set for our approach with reinforcement learning and signatures, and allows to see if there is any design flaw at first. Figure 4 shows the behaviour of the system without any control. As we can notice, it strongly looks like a typical 2D linear and stable system and the delay induced doesn't add much complexity.

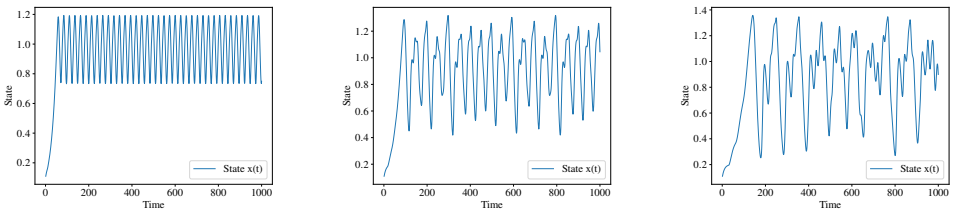
5.4.2 Mackey-Glass in 1D

In this part, we aim to control and stabilize a Mackey-Glass system defined by the equation

$$\forall t \in \mathbb{R}, \dot{x}(t) = -\mu x(t) + \frac{p x(t-\tau)}{1 + x(t-\tau)^n} \tag{56}$$

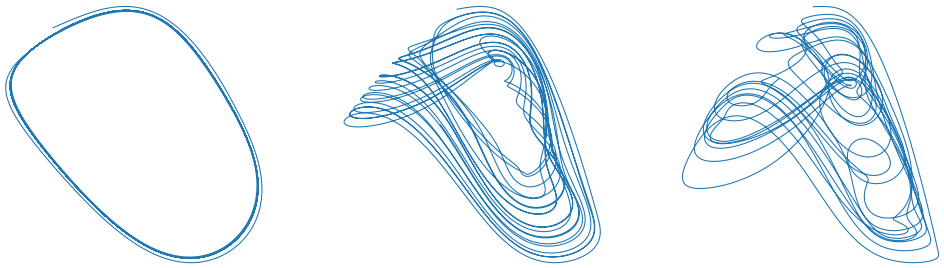
with p, μ, n, τ parameters to be chosen. We will fix $\mu = 0.1, p = 0.2$ and $n = 10$. The crucial parameter here is τ: low values will give a rather periodic trajectory, as more important values will illustrate a more chaotic one. Figure 5 shows the trajectories of such systems while Figure 6 shows the respective attractors.

One limitation of this approach is how easy it is to stabilize the system: [51] shows that a simple feedback loop is sufficient to stabilize the system to 0, so we do not expect a strong gain of performance with this model compared to a Markovian control (based on the current state).



(a) $\tau = 8$ (b) $\tau = 17$ (c) $\tau = 30$

Figure 5: Trajectories of Mackey-Glass systems for different values of τ, the delay



(a) $\tau = 8$ (b) $\tau = 17$ (c) $\tau = 30$

Figure 6: Attractor of Mackey-Glass systems for different values of τ, the delay

5.4.3 Chemical reaction

This example is based on S. Dadebo et al. work [1], where the following 4D system is studied:

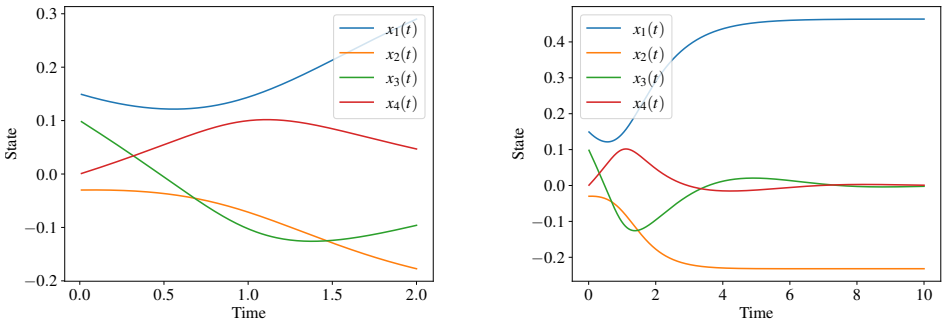
$$\begin{cases} \dot{x}_1 = 0.5 - x_1(t) - R_1 \\ \dot{x}_2 = -2(x_2(t) + 0.25) - u_1(t)(x_2(t) + 0.25) + R_1 \\ \dot{x}_3 = x_1(t-\tau) - x_3(t) - R_2 + 0.25 \\ \dot{x}_4 = x_2(t-\tau) - 2x_4(t) - u_2(t)(x_4(t) + 0.25) + R_2 - 0.25 \end{cases} \tag{57}$$

$$\begin{cases} R_1 = (x_1(t) + 0.5)\exp\left(\frac{25x_2(t)}{x_2(t)+2}\right) \\ R_2 = (x_3(t) + 0.25)\exp\left(\frac{25x_4(t)}{x_4(t)+2}\right) \end{cases} \tag{58}$$

With the goal to minimize the criterion:

$$J(x_0) = \int_0^2 x^\top Q x + u^\top R u \tag{59}$$

with $Q = I_4$, $R = \frac{1}{10} I_2$. This example is interesting for our case since the dynamics is non-trivial while verifying the hypothesis for the value-gradient



(a) Running for 2 seconds (b) Running for 10 seconds

Figure 7: Evolution of the respective states of the system Equation 57 w.r.t. time

approach. S. Dadebo et al. [1] also give the optimal values obtained with a DP algorithm for various τ and x_0, which we can then compare with our approach. We show Figure 7 the trajectory of such a system without any control.

One might note that the input matrix $\frac{\partial f}{\partial u}$ is non-trivial and depend on x. Equation 60 shows the input matrix at time t

$$\left.\frac{\partial f}{\partial u}\right|_{x,t} := B^\top(t) = \begin{pmatrix} 0 & x_2(t) + 0.25 & 0 & 0 \\ 0 & 0 & 0 & x_4(t) + 0.25 \end{pmatrix} \tag{60}$$

When needed, we will consider that we have access to this matrix. Some authors ([9], [10]) have shown that it is in fact possible to learn it (or estimate it) but we will not consider such schemes in this study.

6 Results and Analysis

This section presents the obtained results and aims to provide an analysis of them. We present the obtained results for the 3 different systems discussed Section 5. The chemical reaction system detailed in Equation 57 is the most extensively studied here, showing the most meaningful results.

Section B in the appendix also shows some experiments more related to the choice of hyperparameters of the systems, such as window size or related to the transient times of the systems. Section C provide the used training hyperparameters for each experiments.

In the following figures, we tried to highlight the most important metrics in our opinion. The most essential one might be the so-called cumulated cost, namely the function

$$t \mapsto \int_0^t \text{cost}(x(s), u(s))ds \tag{61}$$

with cost the instantaneous cost function of the system. Here, we chose an LQR cost of form $\text{cost}(x(s), u(s)) = x(s)^\top Q x(s) + u(s)^\top R u(s)$ with Q symmetric non-negative and R symmetric non-negative definite. A working and efficient algorithm should have a cumulated cost function that is asymptotically converging to a constant, as low as possible. It is of course a strictly non-decreasing function.

Since the methods studied can be quite unstable, we decided to plot the mean behaviour, as long as the best shot i.e. the run that was delivering the best performance. This choice was made to highlight the abilities of the signatures despite the concerns about stability (that we are going to discuss later on).

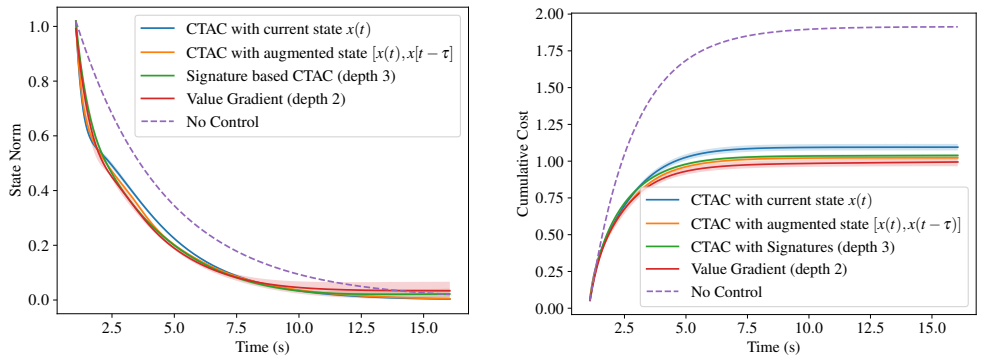
6.1 2D Damped system

Figure 8 shows how the trained algorithms behave on the system described Equation 55 for 10 different seeds. We only tested the algorithms with the depths that were providing the best results (when meaningful). As we can see, the value gradient is the leading solution here but very close to the others. Even with a simple markovian controller we are able to stabilize the system and achieve great performance.

The interest of such a system is that there exists an analytical solution for the optimal control [2]. It can be written as $u^* = \alpha(t)x(t) + \int_{-h}^{0} \beta(t,\theta)x(\theta)d\theta$ with α and β well-chosen functions. Although these functions are not trivial to compute (especially β), knowing the shape of the control allows us to better understand what might be happening. If the term β is small compared to α, then the controller based only on the current state can only learn the first term and perform as well as a controller that is more complex. There is hence a need to choose more complex systems with which the optimal control has a less trivial form.

6.2 Mackey-Glass

Since Mackey-Glass is a 1D system, we define the instantaneaous cost function as



(a) $\|x\|_2$ (b) Cumulated cost

Figure 8: Comparison of metrics (mean and std) on a 2D delayed system for different control techniques

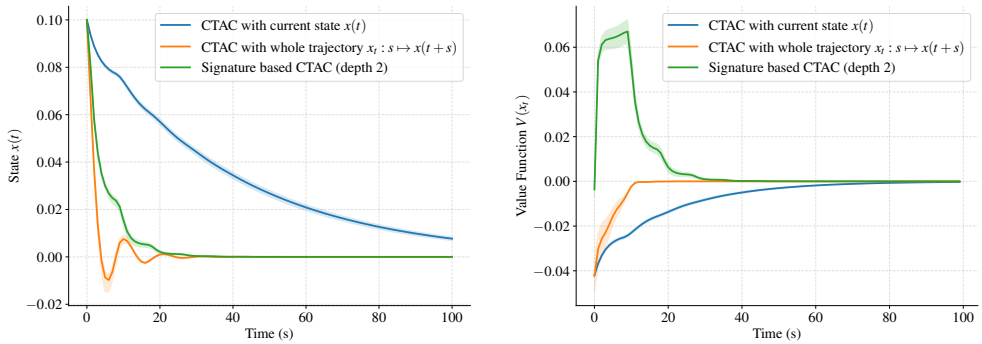
$$\text{cost}(x, u) = x^2 + 0.1u^2 \tag{62}$$

The 0.1 is purely arbitrary as we want to emphasize more on the norm of the current state of the system ($\|x\|_2$) rather than on the amplitude of the command.

Figure 10, Figure 12 and Figure 14 shows the performances of the algorithms on the Mackey-Glass equation (Equation 56) for various values of τ. We tried on 10 different seeds and show the mean and best shot cumulated cost. As it can be seen, the signature CTAC is performing better than the CTAC with the current state, but worse than the one with the full trajectory.

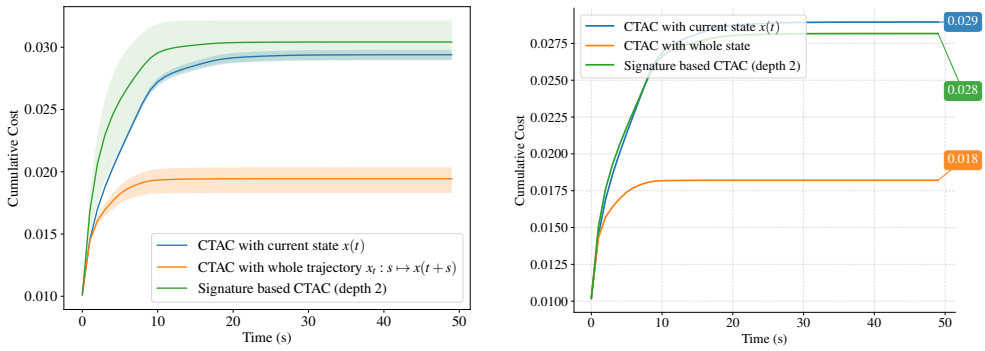
Figure 9, Figure 11 and Figure 13 show the evolution of the position $x(t)$ and the evolution of the Value function V. As we might see, the signature is always the fastest to converge, but doesn't always have the best cost since the command associated is higher for signatures than for other methods. We can also notice that the Value function is positive for the signature-based method, which is incoherent with the system (the reward is always negative and hence the cumulated cost must be too). Since the TD error used by the actor to learn depend on the derivative, it is not that much of a deal to have such positive values. Note that it converges to 0 as the system is stabilized to 0, since the signature will also converge to the null tensor.

Value-Gradient is not shown here because we couldn't manage to have a clean convergence on this particular problem. One of the causes could be the choice of the hyperparameters, or the problem itself not being adequate to such a method. As illustrated in [51], the Mackey-Glass equation can easily be stabilized with simple controllers, even with only the current state in a closed-feedback loop. However, our experiments with different delays shows that the signature-method is performing better than CTAC with the current state for problems with more delay. Overall, the signature-based methods seems more capable to seize and control the chaotic version of Mackey-Glass equation (with a $\tau = 30$) than the markovian or trajectory-based techniques. We can see that on Figure 10, Figure 12 and Figure 14, the mean cost of the Signature Actor Critic performances improves as the delay (and hence the chaotic behaviour) augments.



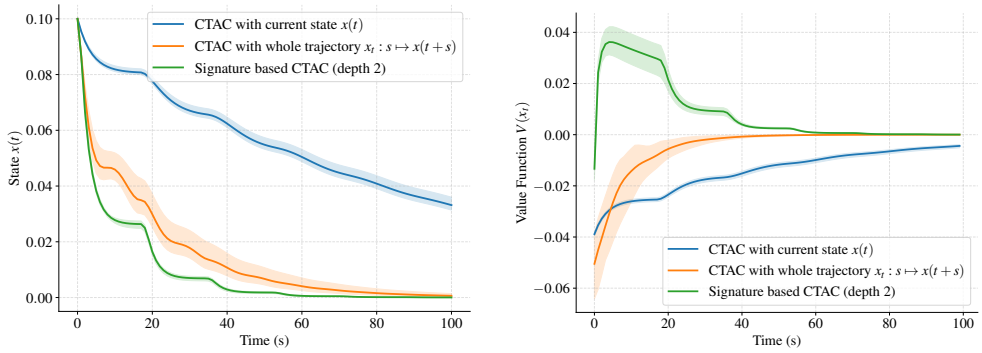
(a) State trajectory (b) Value function estimation

Figure 9: Comparison of trajectories and estimated V function of Mackey-Glass system with $\tau = 8$ for different control techniques over 10 seeds



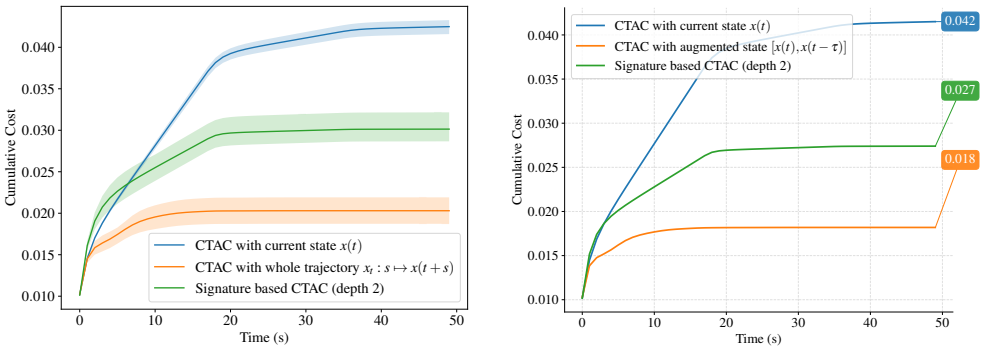
(a) Mean and 90% quantiles (b) Best shot

Figure 10: Comparison of cumulated cost on Mackey-Glass system with $\tau = 8$ for different control techniques over 10 seeds



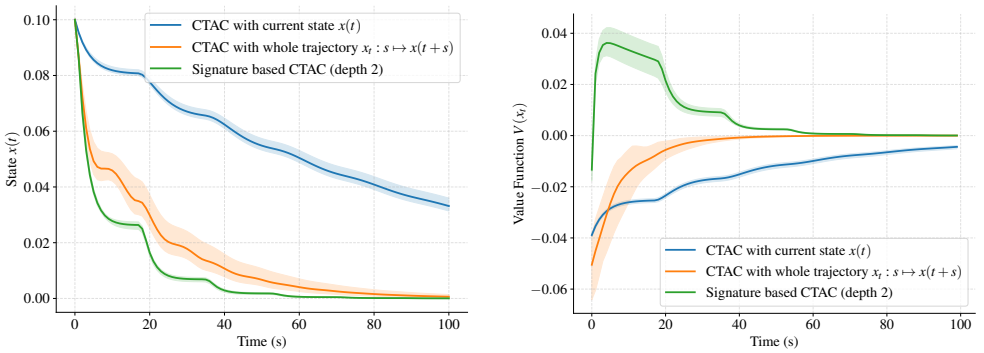
(a) State trajectory (b) Value function estimation

Figure 11: Comparison of trajectories and estimated V function of Mackey-Glass system with $\tau = 17$ for different control techniques over 10 seeds



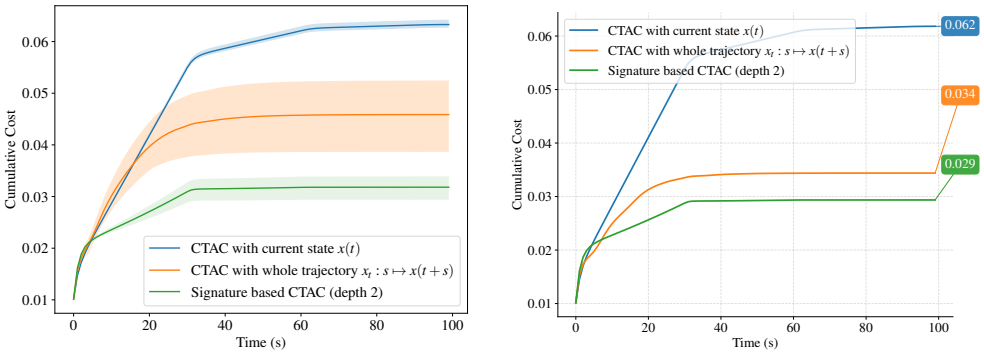
(a) Mean and 90% quantiles (b) Best shot

Figure 12: Comparison of cumulated cost on Mackey-Glass system with $\tau = 17$ for different control techniques over 10 seeds



(a) State trajectory (b) Value function estimation

Figure 13: Comparison of trajectories and estimated V function of Mackey-Glass system with $\tau = 30$ for different control techniques over 10 seeds



(a) Mean and 90% quantiles (b) Best shot

Figure 14: Comparison of cumulated cost on Mackey-Glass system with $\tau = 30$ for different control techniques over 10 seeds

6.3 Chemical Process

This subsection presents the results gathered for the system Equation 57. From the statistical viewpoint, each method was run over 50 different random seeds to have consistency in the results.

6.3.1 Comparison Table

Table 1 shows the results gathered over 50 different random initializations. Since the training is still rather unstable, we made the choice to remove any "bad performing" policies, or the ones that would go to NaNs at some point during the training process. We will denote both as "unstable" policies. A bad performing policy is a policy which final cost is below the uncontrolled cost of around 0.14. This table allows us to draw some observations:

- The most unstable scheme is the CTAC with augmented position, followed by VG methods and current position method. The CTAC method with signatures is quite stable
- The best performances are achieved by VG methods, which also achieves the best mean performances. CTAC with current position achieves a good best performance, but it seems to be a rather rare event looking to its mean performances and standard deviation (STD)
- Signature Depth and parity have an impact on the performances of the algorithm. CTAC is rather consistent with the improvement of its mean performance relative to signature depth, but VG has a sweet spot at depth 3.
- Augmenting the amplitude of the noise for VG improves stability, with rejected policies being reduced a lot without impacting that much the performances.

Technique	Bad perf. (%)	NaN (%)	Total rem. (%)	Best perf.	Mean perf. and STD
CTAC with aug. position $(x(t), x(t-\tau))$	82.35	**0.00**	82.35	0.059	0.12 ± 0.024
CTAC with current position $x(t)$	54.90	**0.00**	54.90	0.037	0.11 ± 0.024
CTAC sig. depth 2	3.92	**0.00**	3.92	0.032	0.087 ± 0.023
CTAC sig. depth 3	**0.00**	**0.00**	**0.00**	0.036	0.085 ± 0.024
CTAC sig. depth 4	1.96	**0.00**	1.96	0.038	0.082 ± 0.024
VG sig. depth 2	47.06	**0.00**	47.06	0.036	0.11 ± 0.022
VG sig. depth 3	66.67	4.00	70.59	**0.027**	**0.073 ± 0.038**
VG sig. depth 4	52.94	**0.00**	52.94	0.032	0.076 ± 0.033
VG sig. depth 2 with more noise	16.00	**0.00**	16.00	0.030	0.11 ± 0.022
VG sig. depth 3 with more noise	34.00	4.00	38.00	0.031	**0.073 ± 0.038**
VG sig. depth 4 with more noise	40.00	**0.00**	40.00	0.037	0.076 ± 0.033

Table 1: Comparison of techniques on the chemical process: seed removal statistics and performance metrics.

6.3.2 Best result comparison

Figure 15 shows the evolution of the criterion to minimize, namely the function:

$$I : t \mapsto \int_0^t \|x(s)\| + 0.1\|u(s)\| ds \tag{63}$$

We show the evolution of $I : t \mapsto I(t)$ for the best shot of each technique, independently of the depth of signature. We compare the found $I(2)$ to the values obtained by [1] at this same time (but we could have run the comparison longer to see how it behave in the long term). As we can see all achieves great performances. The CTAC with augmented state is a bit behind compared to other methods. VG is achieving the best performances, close to the value of 0.024 found by [1] using a dynamic programming approach. As detailed, the value obtained with the current state approach is rather a lucky start than a general performance here.

We also note that the I function associated with VG seems to have attained a plateau and doesn't evolve furthermore, showing that we seem to be stabilizing the system.

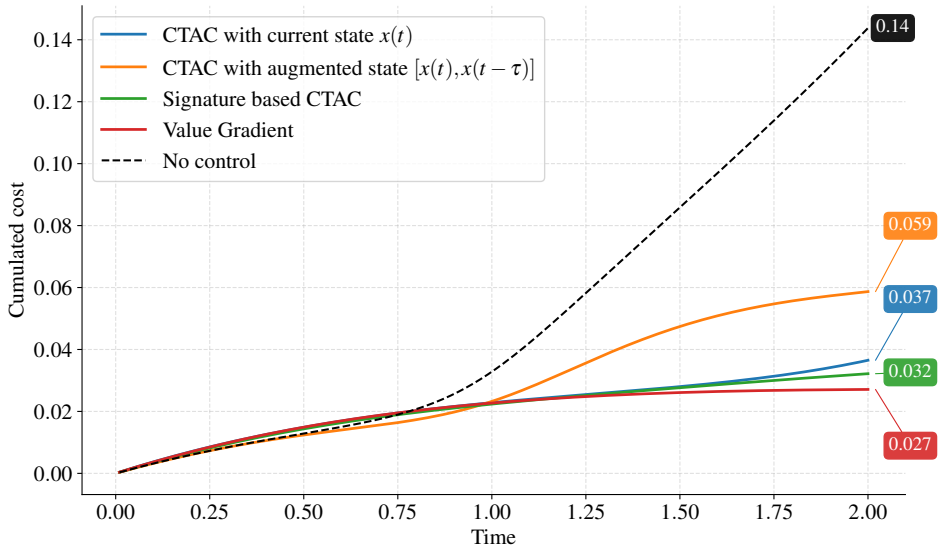
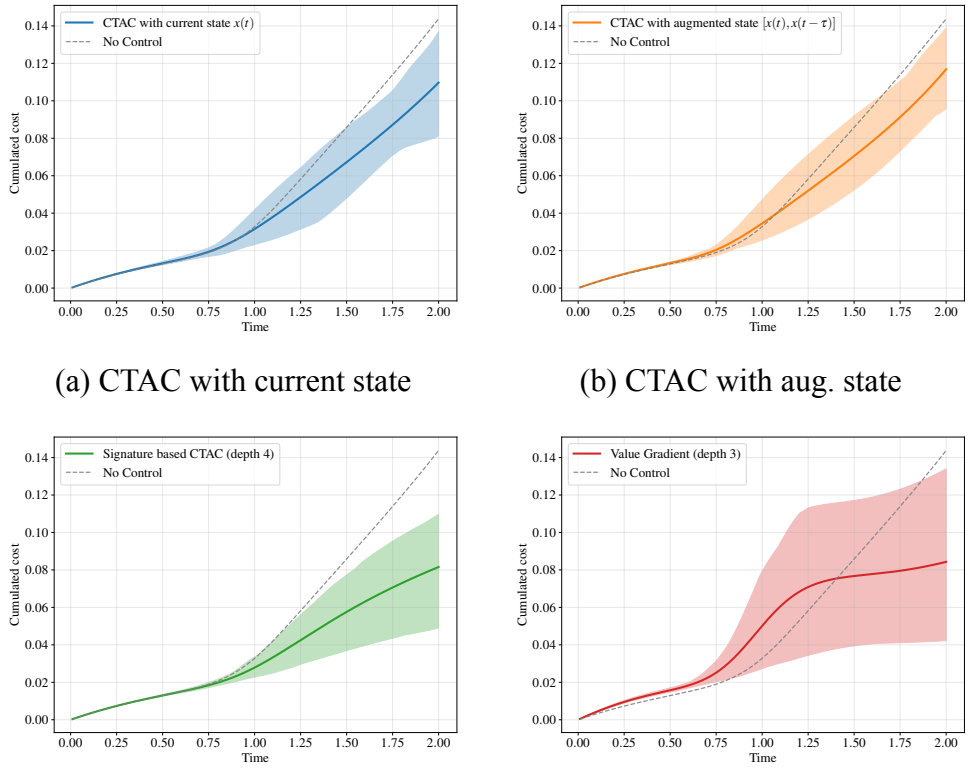


Figure 15: Best cumulated cost for various RL techniques (lowest is better)

6.3.3 Mean cost per method comparison

Figure 16 offers more insights on how the I function (Equation 63) evolves for each techniques. One can then see that VG method are quite unstable but can indeed achieve great results. CTAC with signature are more stable and achieve better results than methods that are not signatures based. This however shows that the best performance that we have is not the general one that we could expect.



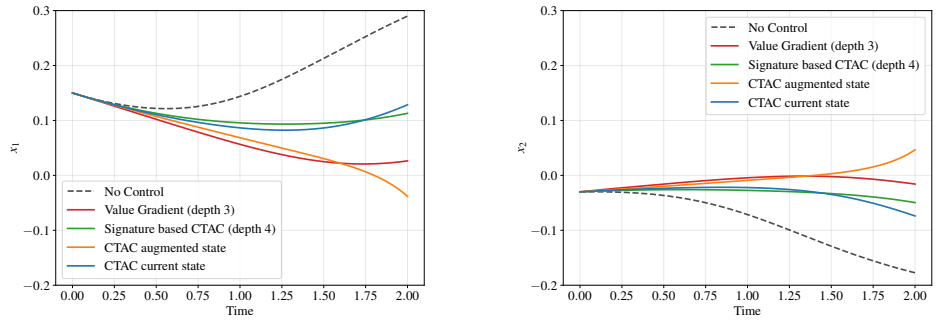
(a) CTAC with current state

(b) CTAC with aug. state

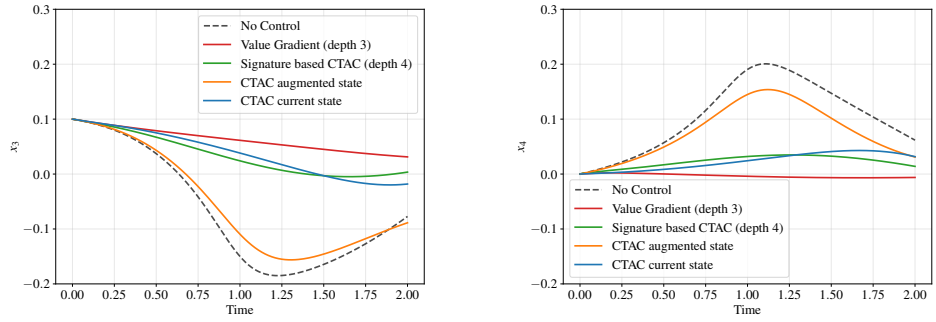
(c) CTAC with signatures (depth 4)

(d) VG with signatures (depth 3)

Figure 16: Comparison of mean and quantiles (10-90 %) values for various techniques on the chemical reaction system **conditional on success** (Equation 57)



(a) x_1 (b) x_2

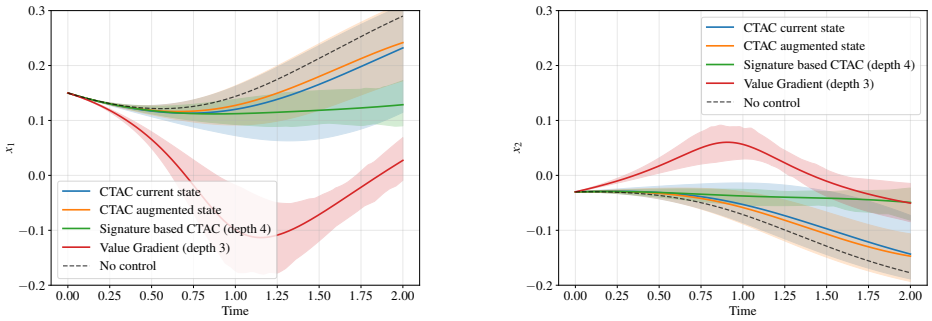


(c) x_3 (d) x_4

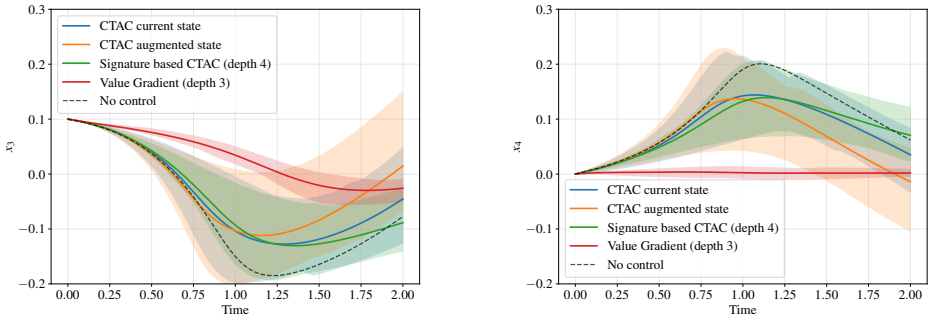
Figure 17: State trajectory per dimension for the system Equation 57 with different RL method, selecting the one that achieved the best cumulated cost

6.3.4 State evolution best per method

Figure 17 shows the trajectories per dimension of the best shot for each method. Interestingly, the trajectory of VG is the one that seems to be the more flat and distant from the non-controlled trajectory, especially on Figure 17d where the magnitude is close to 0.



(a) x_1 (b) x_2



(c) x_3 (d) x_4

Figure 18: Mean state trajectory per dimension for the system Equation 57 with different RL method, mean and quantiles (10-90%), **conditional on success**

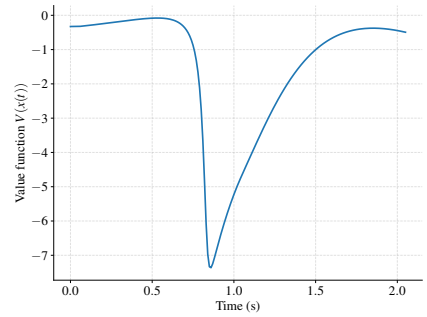
6.3.5 State evolution mean per method

Figure 18 shows the mean trajectory for each algorithm that has converged following criterions defined Table 1. The results for VG shows that the trajectory is the most stable compared to other methods, and distinguishes itself a lot compared to CTAC that are quite close from one another.

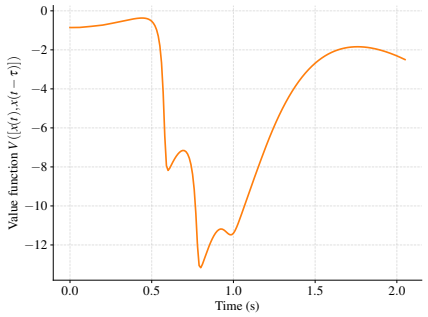
6.3.6 Value Function analysis

Figure 19 shows the value function learned by the different tested algorithms. We show one function for each method, chosen from the ones that actually put a coherent value for Signature methods. It is more discutable for methods based on current state and current + delayed state, since these methods only perform well if they have a good initialisation – that we are then keeping in

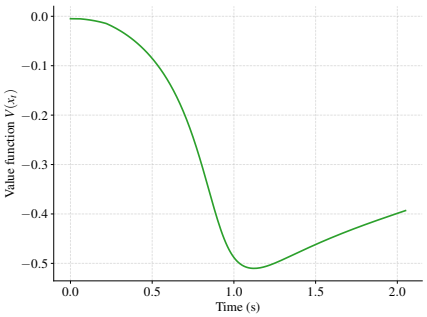
memory – but not if we keep training. Both Figure 19a and Figure 19b shows that the training conducts to a bad local minima with this spike at around 1s of high intensity. This spike illustrate the unability of the system to learn to stabilize the system with such low information on the state x_t. Figure 19c and Figure 19d on the other hand show very dinstinct shapes: the Signature Actor Critic decreases until 1s and then augments again, which appears coherent with Figure 18 learned control trajectory. The VG value function is quite different, raising above 0 – which shouldn't be possible but not a problem since the control only care about the "gradient" of this function – and then decreasing abruptly. Interestingly, the value function can take the same shape as the CTAC during training, but goes to more optimal values leading to this concave value function.



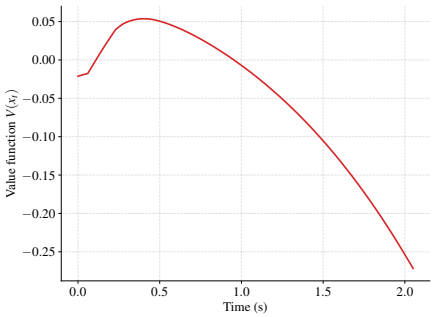
(a) CTAC with current state



(b) CTAC with aug. state



(c) CTAC with signatures (depth 4)



(d) VG with signatures (depth 3)

Figure 19: Value function evolution w.r.t. time for different algorithms

7 Discussion

This chapter discusses the implications of the results and their significance.

7.1 Interpretation of Results

Results shown Section 6 highlight a few things. First the signature-based method tend to perform well, and even better than methods based on current state or whole history.

Second, signature-based methods seems to really show a gain on non-linear and complex dynamics rather than linear ones. The example of the 2D system Equation 55 clearly shows that the advantage taken by the signature methods is slight, wherever the example from the chemical reaction Equation 57 highlight that even if a current-state bases method can yield good results, signatures have better performances on average. Mackey-Glass (Equation 56) is maybe the most successful attempt: signatures are clearly advantaged when the system is chaotic and show a real performance gain compared to other methods.

Lastly, the depth of the signature seems to have an important role both in stability and convergence. However, it is not clear how: on one hand for the chemical system, depth 3 yields the best result but on the other, it is the most unstable one for value gradient methods.

7.2 Limitations

7.2.1 Case study

A first limitation is on the studied systems. One can note that these systems all present a single punctual delay, and therefore is pretty simple to solve

for purely markovian actors even if they achieve worse performance than signature-based ones. In a next step, it could be interesting to focus on more complex systems, with integral delays. One can think of second kind Volterra equations for example $\dot{x}(t) = ax(t) + \int_{t-\tau}^{t} K(s)x(s)ds$ with K a well-chosen kernel function (exponential for example). The issue with such a scheme is that it may requires a more advanced solver, and we didn't had the time to implement it.

7.2.2 Stability

One might have notice that the presented algorithm are quite unstable, and sensitive to noise exploration, initialisation etc. It is crucial to fix these issues and have a clearer opinion about their causes. Specifically for Value-Gradient on Mackey-Glass, we do not understand why we cannot manage to have a stable solution. There is a need to investigate this problem with a sweep on hyperparameters to see if we can manage to have a working solution. Some hypotheses about these instabilities are listed here:

- **Curse of dimensionality**: the signature size grows exponentially with the depth, causing bigger tensor to learn from and more difficulty for the optimizer compared to the markovian case.
- **Finite difference estimation of $\dot{V}$**: This estimation is the keypoint of CTAC and VG. Since we divide by dt in this approximation, we can have high derivative values directly repercuting in the TD error, and hence on the learning process.
- **The Value-Gradient case**: VG is at the same time the most unstable and the methods that leads the best results on Equation 57 system. Diving a bit more in the trajectories during the learning phase, we see that the system often crosses a suboptimal valley before going to a better solution. This is quite peculiar and seems to coincide with the fact that we are disminishing the noise. It shows how rough the loss landscape is with the signature and how complicated it might be to find a suitable solution. Most of the run do not converge as illustrated in Table 1 for value gradient, and adding more noise seems to allow more effective convergence. More noise allows to explore more, thus leading in quicker convergence as the noise decreases later in training.

- **Hyperparameter optimisation**: There is quite much hyperparameters to estimate (see Table 4 for example) and finding the sweetspot is computationally expensive. Each system is different and requires a hyperparameter search of its own.

We can also view this drawback as a main limitation of the developed algorithms: this lack of stability is an important issue in the deployment of such control solutions.

7.2.3 Reward design

For the sake of simplicity, we designed the reward as the classical LQR reward, which only depends on the instantaneous state $x(t)$. While the Value Function still depends on the full functional state x_t – since the future evolution of the system is governed by the FDE – using an instantaneous reward may favour Markovian controllers, which are not penalised for ignoring trajectory history. A more appropriate formulation for delayed systems would be to design the reward as

$$r(t) = x(t)^\top Q x(t) + \int_{-h}^{0} x(t+s)^\top F(s) x(t+s) ds + u^\top(t) R u(t) \tag{64}$$

where F could be, for instance, an exponential kernel or simply the identity. Such a functional reward would penalise the entire trajectory window and could yield significantly different results, potentially strengthening the case for signature-based methods over Markovian ones. This remains an open question for future investigation.

7.3 Future Work

The question that we tried to answer by setting the detailed framework of the thesis is a current hot topic in the RL community. CTRL has gained lots of popularity on the previous year, with the work of Yildiz [18] and many others. We also think that even Delayed systems might not be the center of the attention, the link with Partially Observable systems via Takens theorem (see

Section 3.2.1) is a promising lead to use this technique to these problems. We hence present the extension that can be made from this work in order to push further the research or publish an article.

7.3.1 Addressing the limitations

The first and most important development would be to first adress the current limitations. A more extensive study on different systems would be a plus to see more comparisons and highlight that a markovian agent fails where a delayed one, and more precisely one based on signatures is more efficient. Our study has illustrated such phenomena in some extent, specifically with the study of the chemical system (see Section 6.3), but some on more complex systems would be a plus.

7.3.2 Policy Gradient and other algorithms

One major bottleneck in this work was the derivation of the V function in continuous time relatively to the state which was a function and not a vector. Once this was done, it opens the door to lots of others algorithms that can be adapted. For example Policy Gradient in continuous time, first presented by Rémi Munos [10], but we can think of others and adapt more recent/SoTA algorithms like Integral Reinforcement Learning from H. Modares et al. [22]. This could lead to new and more stable algorithms to play with.

7.3.3 Partially Observable Markov Decision Process

As mentioned in the introduction of this part, the link between delayed and partially observable systems could be another way to use signatures and control for RL. Takens theorem allows to perceive such partially observable system as a delayed one. Therefore our method could be used for such a scheme. One good test case could be with Kuramoto–Sivashinsky (KS) equation since current numerical schemes are quite fast, and it is a quite chaotic system so a good challenge for signatures compared to other methods.

7.4 Reflections

The signature transform is a promising tool for control of delayed systems, in the sense that it replaces the state from a discretisation of the path x_t sensitive to the discretisation Δt to a vector of size $\approx d^N$ with d the dimension of the state vector and N the chosen depth of the signature. This answers the dependency of the discretisation in the sense that feeding a neural network with the raw state, having a thinner discretisation would require more features in our network. Signatures are agnostic to this and a more precise discretisation would only yield more precise coefficients. One could argue that the size of the vector is exponentially increasing with the depth, but as we have seen, even shallow signatures of depth 2 to 4 can yield satisfactory results. This claim is supported by Ohnishi [34] and Bleistein [13], where the first only use depth 2 signatures and the second provide a study of the influence of depth over the prediction task.

One of the problematic aspect however is that our algorithm seems to be very sensitive to hyperparameters and initialisation. Table 1 shows this pretty well with lots of rejected cases, especially for methods that yield good results. It could be interesting as mentioned in the stability issues to perform more test in this sense. It would validate the usefulness of such a method and solving the instability problems would reinforce the scientific credibility of this work.

7.5 Ethics, Sustainability and Societal Impact

7.5.1 Sustainability

A notable characteristic of the proposed approach is its computational frugality. Unlike modern deep RL methods that rely on large neural networks and require extensive GPU resources for training, our signature-based actors and critics are parameterised by linear models operating on fixed-size feature vectors. Training therefore runs on standard CPU hardware in minutes rather than hours, with a significantly smaller energy footprint. This is relevant in the

context of growing concerns about the environmental cost of machine learning research.

Beyond the training process itself, the long-term sustainability argument is stronger at the application level. Improved control of industrial processes — such as the chemical reactor studied in Section 6.3 — can reduce waste, lower energy consumption, and limit the production of unwanted by-products. More generally, applying principled control methods to complex dynamical systems contributes to more efficient use of physical resources, which aligns with the United Nations Sustainable Development Goals.

7.5.2 Ethics

The systems studied in this thesis are academic benchmarks and the research is of a fundamental nature, involving no personal data, no human subjects, and no deployment in production environments. Ethical concerns are therefore limited at this stage.

That said, any deployment of autonomous control systems in safety-critical settings — chemical plants, energy grids, or biological systems — raises important questions that future work must address. The training instability documented in Section 7.2 is a concrete concern: a controller that converges reliably in most runs but occasionally diverges could cause damage if deployed without sufficient validation. Responsible deployment would require formal stability guarantees or at least extensive empirical stress-testing before any real-world use.

More broadly, automated control reduces the need for human intervention and shifts decision-making to algorithms whose behaviour can be opaque. Compared to deep neural networks, the signature-based linear policy used here has the advantage of being more interpretable: the weights directly connect signature features — which have a clear mathematical meaning as iterated integrals of the trajectory — to the control action. This relative transparency is a modest but genuine ethical advantage.

7.5.3 Relations with the Surrounding Community

This work sits at the intersection of rough path theory, functional analysis, and continuous-time RL, and its primary contribution is to the academic research community. By establishing a theoretical bridge between the HJB equation for functional states and signature-based function approximation, and by providing an open implementation of a DDE simulation environment, this thesis offers a reproducible starting point for future studies in non-Markovian control.

The industrial relevance is also concrete. The chemical process benchmark studied here is representative of a broad class of processes where transport delays are inherent — reactors, pipelines, heat exchangers. Providing a principled, lightweight method for controlling such systems without requiring a model of the delay structure could lower the barrier to applying RL-based control in process engineering.

8 Conclusions

This thesis has presented a novel approach to the control of Delayed Differential Equations using Reinforcement Learning. The main objective was to address the challenge of controlling DDEs and FDEs, where the system state is a trajectory rather than a point. Using signatures, we were able to learn a value function and a control policy, successfully stabilizing and optimizing various systems.

The experimental results show that signature-based approaches are competitive and often more robust than baseline representations based only on the current state or on the full trajectory. In particular, on the chemical process benchmark, signature-based methods achieved stronger average performance, while value-gradient variants reached the best individual results when training remained stable. Moreover, the Mackey-Glass experiment showed that signatures are more effective on chaotic and less-predictable systems. These findings support the idea that signatures can effectively capture relevant delayed information for control.

At the same time, this work highlights important limitations. Training stability remains sensitive to initialization, exploration noise, and hyperparameter choices, especially for value-gradient schemes. In addition, the considered benchmarks involve mainly single-delay systems, which only partially reflect the complexity of broader non-Markovian control problems.

This thesis provides both a theoretical and empirical step toward model-free control of delayed systems with signature features. Future work should focus on improving optimization stability, extending the framework to systems with distributed or multiple delays, and adapting more advanced continuous-time RL algorithms. These directions could strengthen the reliability and scope of signature-based control in realistic applications.

Bibliography

[1] S. Dadebo and R. Luus, "Optimal Control of Time-delay Systems by Dynamic Programming," *Optimal Control Applications and Methods*, vol. 13, no. 1, pp. 29–41, Jan. 1992, doi: 10.1002/oca.4660130103.

[2] V. Kolmanovskii and A. Myshkis, *Applied Theory of Functional Differential Equations*. Dordrecht: Springer Netherlands, 1992. doi: 10.1007/978-94-015-8084-7.

[3] K.-T. Chen, "Iterated integrals and exponential homomorphisms," *Proceedings of the London Mathematical Society*, vol. 3, no. 1, pp. 502–512, 1954.

[4] T. J. Lyons, M. Caruana, and T. Lévy, *Differential equations driven by rough paths: Ecole d'Eté de Probabilités de Saint-Flour XXXIV-2004*. Springer, 2007.

[5] I. P. Arribas, "Derivatives Pricing Using Signature Payoffs," no. arXiv:1809.09466. arXiv, Sept. 2018. doi: 10.48550/arXiv.1809.09466.

[6] H. Smith, *An Introduction to Delay Differential Equations with Applications to the Life Sciences*, vol. 57. in Texts in Applied Mathematics, vol. 57. New York, NY: Springer New York, 2011. doi: 10.1007/978-1-4419-7646-8.

[7] R. S. Sutton, A. G. Barto, and others, *Reinforcement learning: An introduction*, vol. 1, no. 1. MIT press Cambridge, 1998.

[8] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[9] K. Doya, "Reinforcement Learning in Continuous Time and Space," *Neural Computation*, vol. 12, no. 1, pp. 219–245, Jan. 2000, doi: 10.1162/089976600300015961.

[10] R. Munos, "Policy gradient in continuous time," *Journal of Machine Learning Research*, vol. 7, pp. 771–791, 2006.

[11] P. Kidger, P. Bonnier, I. Perez Arribas, C. Salvi, and T. Lyons, "Deep signature transforms," *Advances in neural information processing systems*, vol. 32, 2019.

[12] A. Fermanian, "Functional Linear Regression with Truncated Signatures," *Journal of Multivariate Analysis*, vol. 192, p. 105031, Nov. 2022, doi: 10.1016/j.jmva.2022.105031.

[13] L. Bleistein, A. Fermanian, A.-S. Jannot, and A. Guilloux, "Learning the dynamics of sparsely observed interacting systems," in *International Conference on Machine Learning*, 2023, pp. 2603–2640.

[14] E. Fridman, *Introduction to Time-Delay Systems: Analysis and Control*. in Systems & Control: Foundations & Applications. Cham: Springer International Publishing, 2014. doi: 10.1007/978-3-319-09393-2.

[15] B. Chen, M. Xu, L. Li, and D. Zhao, "Delay-Aware Model-Based Reinforcement Learning for Continuous Control," *Neurocomputing*, vol. 450, pp. 119–128, Aug. 2021, doi: 10.1016/j.neucom.2021.04.015.

[16] Y. Wang, C. Wang, S. Zhao, and K. Guo, "Research on Deep Reinforcement Learning Control Algorithm for Active Suspension Considering Uncertain Time Delay," *Sensors*, vol. 23, no. 18, p. 7827, Jan. 2023, doi: 10.3390/s23187827.

[17] R. Munos and P. Bourgine, "Reinforcement Learning for Continuous Stochastic Control Problems," in *Advances in Neural Information Processing Systems*, MIT Press, 1997.

[18] C. Yildiz, M. Heinonen, and H. Lähdesmäki, "Continuous-time Model-based Reinforcement Learning," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., in Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 12009–12018. [Online]. Available: https://proceedings.mlr.press/v139/yildiz21a.html

[19] S. Holt, A. Hüyük, Z. Qian, H. Sun, and M. van der Schaar, "Neural laplace control for continuous-time delayed systems," in *International Conference on Artificial Intelligence and Statistics*, 2023, pp. 1747–1778.

[20] E. Pradeleix, R. Hosseinkhan-Boucher, A. Shilova, O. Semeraro, and L. Mathelin, "Learning Non-Markovian Dynamical Systems with Signature-based Encoders," no. arXiv:2509.12022. arXiv, Sept. 2025. doi: 10.48550/arXiv.2509.12022.

[21] T. Monsel, E. Menier, O. Semeraro, L. Mathelin, and G. Charpiat, "Neural DDEs with Learnable Delays for Partially Observed Dynamical Systems," no. arXiv:2410.02843. arXiv, Oct. 2024. doi: 10.48550/arXiv.2410.02843.

[22] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Integral Reinforcement Learning and Experience Replay for Adaptive Optimal Control of Partially-Unknown Constrained-Input Continuous-Time Systems," *Automatica*, vol. 50, no. 1, pp. 193–202, Jan. 2014, doi: 10.1016/j.automatica.2013.09.043.

[23] B. T. Polyak, *Introduction to optimization*. New York, Optimization Software, 1987.

[24] H. Han and S. Ji, "Continuous Soft Actor-Critic: An Off-Policy Learning Method Robust to Time Discretization," *Advances in Neural Information Processing Systems*, vol. 38, pp. 73771–73807, 2026.

[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, 2018, pp. 1861–1870.

[26] K.-T. Chen, "Integration of paths, geometric invariants and a generalized Baker-Hausdorff formula," *Annals of Mathematics*, vol. 65, no. 1, pp. 163–178, 1957.

[27] K.-T. Chen, "Integration of paths–A faithful representation of paths by noncommutative formal power series," *Transactions of the American Mathematical Society*, vol. 89, no. 2, pp. 395–407, 1958.

[28] A. Fermanian, P. Marion, J.-P. Vert, and G. Biau, "Framing RNN as a Kernel Method: A Neural ODE Approach," no. arXiv:2106.01202. arXiv, Oct. 2021. doi: 10.48550/arXiv.2106.01202.

[29] F. J. Király and H. Oberhauser, "Kernels for Sequentially Ordered Data," no. arXiv:1601.08169. arXiv, Jan. 2016. doi: 10.48550/arXiv.1601.08169.

[30] I. Chevyrev and A. Kormilitzin, "A Primer on the Signature Method in Machine Learning." [Online]. Available: https://arxiv.org/abs/1603.03788

[31] B. Hambly and T. Lyons, "Uniqueness for the Signature of a Path of Bounded Variation and the Reduced Path Group," *Annals of Mathematics*, vol. 171, no. 1, pp. 109–167, Mar. 2010, doi: 10.4007/annals.2010.171.109.

[32] A. Fermanian, J. Chang, T. Lyons, and G. Biau, "The Insertion Method to Invert the Signature of a Path," no. arXiv:2304.01862. arXiv, Sept. 2023. doi: 10.48550/arXiv.2304.01862.

[33] M. Hoglund *et al.*, "A Neural RDE Approach for Continuous-Time Non-Markovian Stochastic Control Problems," no. arXiv:2306.14258. arXiv, June 2023. doi: 10.48550/arXiv.2306.14258.

[34] M. Ohnishi, I. Akinola, J. Xu, A. Mandlekar, and F. Ramos, "Signatures Meet Dynamic Programming: Generalizing Bellman Equations for Trajectory Following," no. arXiv:2312.05547. arXiv, June 2024. doi: 10.48550/arXiv.2312.05547.

[35] Z. Zhao, Q. Li, and Y. Xu, "From Path Signatures to Sequential Modeling: Incremental Signature Contributions for Offline RL," no. arXiv:2602.11805. arXiv, Feb. 2026. doi: 10.48550/arXiv.2602.11805.

[36] P. K. Friz and N. B. Victoir, *Multidimensional Stochastic Processes as Rough Paths: Theory and Applications*. Cambridge University Press, 2012.

[37] A. Bellen and M. Zennaro, *Numerical methods for delay differential equations*. OUP Oxford, 2003.

[38] F. Takens, "Detecting Strange Attractors in Turbulence," *Dynamical Systems and Turbulence, Warwick 1980*, vol. 898. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 366–381, 1981. doi: 10.1007/BFb0091924.

[39] R. E. Kálmán, “Contributions to the Theory of Optimal Control,” Boletin de la Sociedad Matematica Mexicana, 1960. [Online]. Available: https://api.semanticscholar.org/CorpusID:845554

[40] G. Carlier, “PROGRAMMATION DYNAMIQUE,” 2007.

[41] O. Mula, *Optimisation et Programmation Dynamique*. Université Paris Dauphine, 2019.

[42] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, “Reinforcement Learning and Feedback Control: Using Natural Decision Methods to Design Optimal Adaptive Controllers,” *IEEE Control Systems Magazine*, vol. 32, no. 6, pp. 76–105, 2012, doi: 10.1109/MCS.2012.2214134.

[43] D. W. Ross and I. Flügge-Lotz, “An Optimal Control Problem for Systems with Differential-Difference Equation Dynamics,” *SIAM Journal on Control*, vol. 7, no. 4, pp. 609–623, Nov. 1969, doi: 10.1137/0307044.

[44] B. Dupire, “Functional Itô Calculus\dag ,” *Quantitative Finance*, vol. 19, no. 5, pp. 721–729, May 2019, doi: 10.1080/14697688.2019.1575974.

[45] G. Ansmann, “Efficiently and easily integrating differential equations with JiTCODE, JiTCDDE, and JiTCSDE,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 28, no. 4, Apr. 2018, doi: 10.1063/1.5019320.

[46] J. Reizenstein and B. Graham, “Algorithm 1004: The iisignature Library: Efficient Calculation of Iterated-Integral Signatures and Log Signatures,” *ACM Transactions on Mathematical Software (TOMS)*, 2020.

[47] P. Kidger and T. Lyons, “Signatory: differentiable computations of the signature and logsignature transforms, on both CPU and GPU,” in *International Conference on Learning Representations*, 2021.

[48] J. Bradbury *et al.*, “JAX: composable transformations of Python+NumPy programs.” [Online]. Available: http://github.com/jax-ml/jax

[49] DeepMind *et al.*, “The DeepMind JAX Ecosystem.” [Online]. Available: http://github.com/google-deepmind

[50] J. Heek *et al.*, “Flax: A neural network library and ecosystem for JAX.” [Online]. Available: http://github.com/google/flax

[51] G. Kiss and G. Röst, “Controlling Mackey–Glass Chaos,” no. arXiv:1708.05580. arXiv, Aug. 2017. doi: 10.48550/arXiv.1708.05580.

A Proof extensions

A.1 Signature simplification

Here we derive the first-order approximation of $S_{[0,\Delta t]}(x_t)$. Let S^I be the coefficient associated with the word $I = (i_1, ..., i_k) \in \{1, ..., d\}^k$. By definition:

$$S^I = \int_{0 \le s_1 < \cdots < s_k \le \Delta t} \dot{x}^{(i_1)}(t + s_1) \cdots \dot{x}^{(i_k)}(t + s_k)\, \mathrm{d}s_k \cdots \mathrm{d}s_1 \tag{A}$$

Approximating the integrands using zeroth-order Taylor expansions (constant approximation near t) yields:

$$\begin{aligned} S^I &\approx \dot{x}^{(i_1)}(t) \cdots \dot{x}^{(i_k)}(t) \int_{0 \le s_1 < \cdots < s_k \le \Delta t} \mathrm{d}s_k \cdots \mathrm{d}s_1 \\ &= \dot{x}^{(i_1)}(t) \cdots \dot{x}^{(i_k)}(t) \frac{\Delta t^k}{k!} + o\big(\Delta t^k\big) \end{aligned} \tag{B}$$

Since we are dividing the signature by Δt and letting $\Delta t \to 0$, we observe the order of convergence for different levels k:

- For $k \ge 2$, the term is of order $O(\Delta t^2)$. Divided by Δt, it becomes $O(\Delta t)$, which vanishes as $\Delta t \to 0$.
- For $k = 1$ (the first layer of the signature), we have a single integral:

$$\int_0^{\Delta t} \dot{x}^{(i_1)}(t + s)\, \mathrm{d}s = \dot{x}^{(i_1)}(t) \Delta t + o(\Delta t) \tag{C}$$

Thus, only the first level terms contribute to the derivative, while higher-order terms ($k \ge 2$) vanish in the limit.

A.2 Tensor truncation

We briefly clarify the tensor truncation operator $\otimes_N$. In continuous-time theoretical formulations, operations like $v \otimes S(x_t)$ typically assume infinite tensor series, abstracting away dimensionality constraints. However, in practice, memory is finite and we must work with the truncated signature $S^{N(x_t)}$. Computing the exact algebraic tensor product of a vector $v \in \mathbb{R}^d$ and $S^{N(x_t)}$ would map the result to a higher-degree tensor space of size $d \times \frac{d^{N+1}-1}{d-1}$. To maintain a constant state dimension for Value-Gradient or time-derivative computations, we must discard the highest-degree coefficients.

Therefore, similarly to Ohnishi et al. [34], we define the truncated tensor product $\otimes_N$ such that it preserves the original signature size:

$$v \otimes_N S^{N(x_t)} = (0, v, v \otimes S^{\mathbf{1}}(x_t), v \otimes S^{\mathbf{2}}(x_t), ..., v \otimes S^{\mathbf{N-1}}(x_t))$$

where $S^{\boldsymbol{I}}(x_t)$ denotes the tensor made of the I-th depth layer of the signature of x_t. Note that the degree-zero scalar term is 0, as the pure vector v has no scalar component. One can easily verify that the resulting tensor contains exactly $\frac{d^{N+1}-1}{d-1}$ coefficients. The right-sided product $S^{N(x_t)} \otimes_N v$ is defined analogously by computing the tensor product of each signature layer with v on the right.

A.3 Total variation of a path

Here we define the total variation of a path. This is not used in the thesis but is a common quantity when studying signatures in the stochastic setting.

Definition 12. *(Total Variation of a path)* Let $x_t : [a, b] \to \mathbb{R}^d$, the total variation of x_t is defined as:

$$\|x_t\|_{\mathrm{TV}} = \sup_{n \in \mathbb{N}} \sup_{\mathcal{P}_n} \sum_{i=1}^{n} \|x_t(s_i) - x_t(s_{i-1})\|_2 \tag{D}$$

With $\mathcal{P}_n : a = s_0 < s_1 < ... < s_n = b$ the partition of $n+1$ points in $[a, b]$

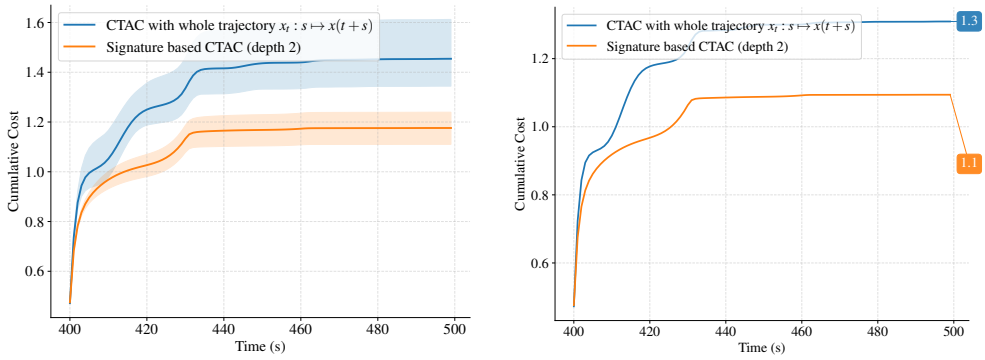
The total variation can be seen as the cumulative path length of the curve, accounting for every undulation and oscillations no matter the scale. It is

especially useful for non-$\mathcal{C}^1$ paths, as for the other it is simply the classical arc length [30]

B Supplementary experiments

B.1 Mackey Glass without transient

We retested Mackey-Glass system, but waiting for the transient time to happen before trying to stabilize it to ensure that this transient didn't had too much of an effect on the learning procedure and on performances of the algorithms. Mean and best cost are plotted Figure 20. The Markovian actor is not showed since it failed to stabilize the system and shows bad performances.



(a) Mean and 90% quantiles (b) best shot

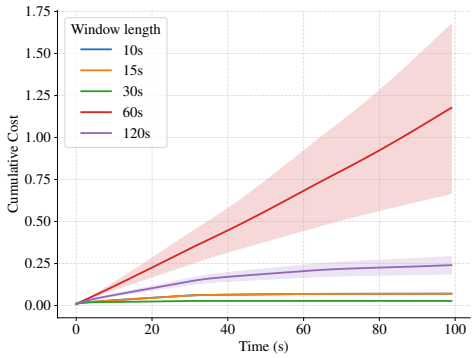
Figure 20: Comparison of cumulated cost on Mackey-Glass system with $\tau = 30$ for different control techniques over 5 seeds without transient times

B.2 Window size experiments on Mackey-Glass

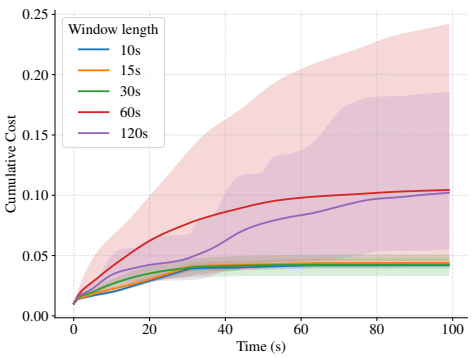
We experimented with the window size on Mackey-Glass experiment to evaluate the impact of the parameter for non-Markovian actors. Interestingly, the size seems very important for signature-based methods compared to full trajectories methods. The firsts are failing to stabilize completely the system for window size superior to the delay of the system, and show degraded performances for smaller windows, while the last always stabilize, and shows comparable performances (or even slightly better in average) for smaller windows compared to the actual delay.

To provide a clearer explanation, we analyze the evolution of the leading eigenvalues of the signature covariance matrix ($\mathrm{Cov}(X) = XX^\top$), shown in Figure 22. We compare this with the same analysis performed on the discretized trajectory (rather than on signatures), shown in Figure 23.

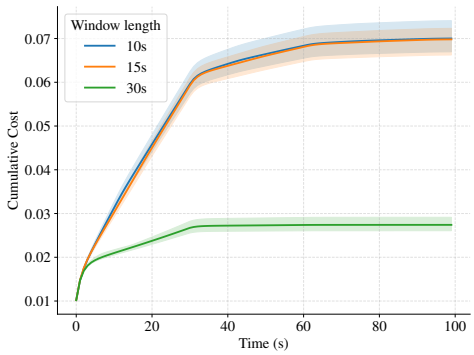
A key observation is that modes 3, 4, and 5 do not collapse to negligible values as the window size increases. In particular, for depths 3 and 4, modes 4 and 5 exhibit a noticeable peak around 60 s, which may explain why convergence at 60 s is substantially better for depth 3 than for depth 2. By contrast, in the full-trajectory case (Figure 23), the contribution of higher-order modes tends to increase monotonically with window size, highlighting a fundamentally different behavior. Overall, this analysis helps illustrate how information is encoded in signatures and why performance is sensitive to the chosen window length. It also emphasizes the importance of accurately estimating the system delay: even a rough mismatch can significantly degrade convergence.



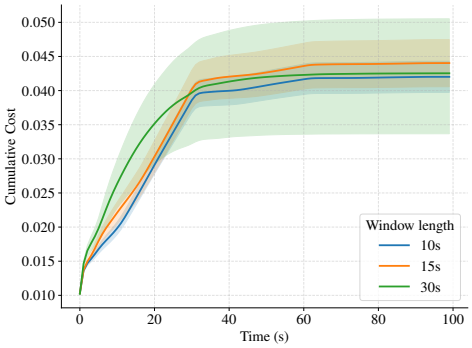
(a) Signature Based CTAC, depth 2



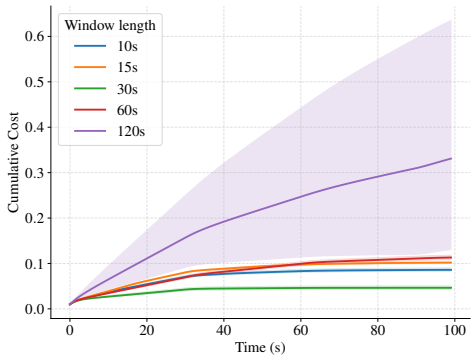
(b) CTAC with full trajectory



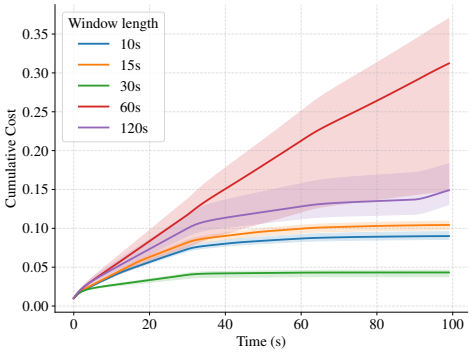
(c) Signature Based CTAC, depth 2



(d) CTAC with full trajectory



(e) Signature Based CTAC, depth 3



(f) Signature Based CTAC, depth 4

Figure 21: Comparison of cumulated cost on Mackey-Glass system with $\tau = 30$ over 5 seeds for different time window size

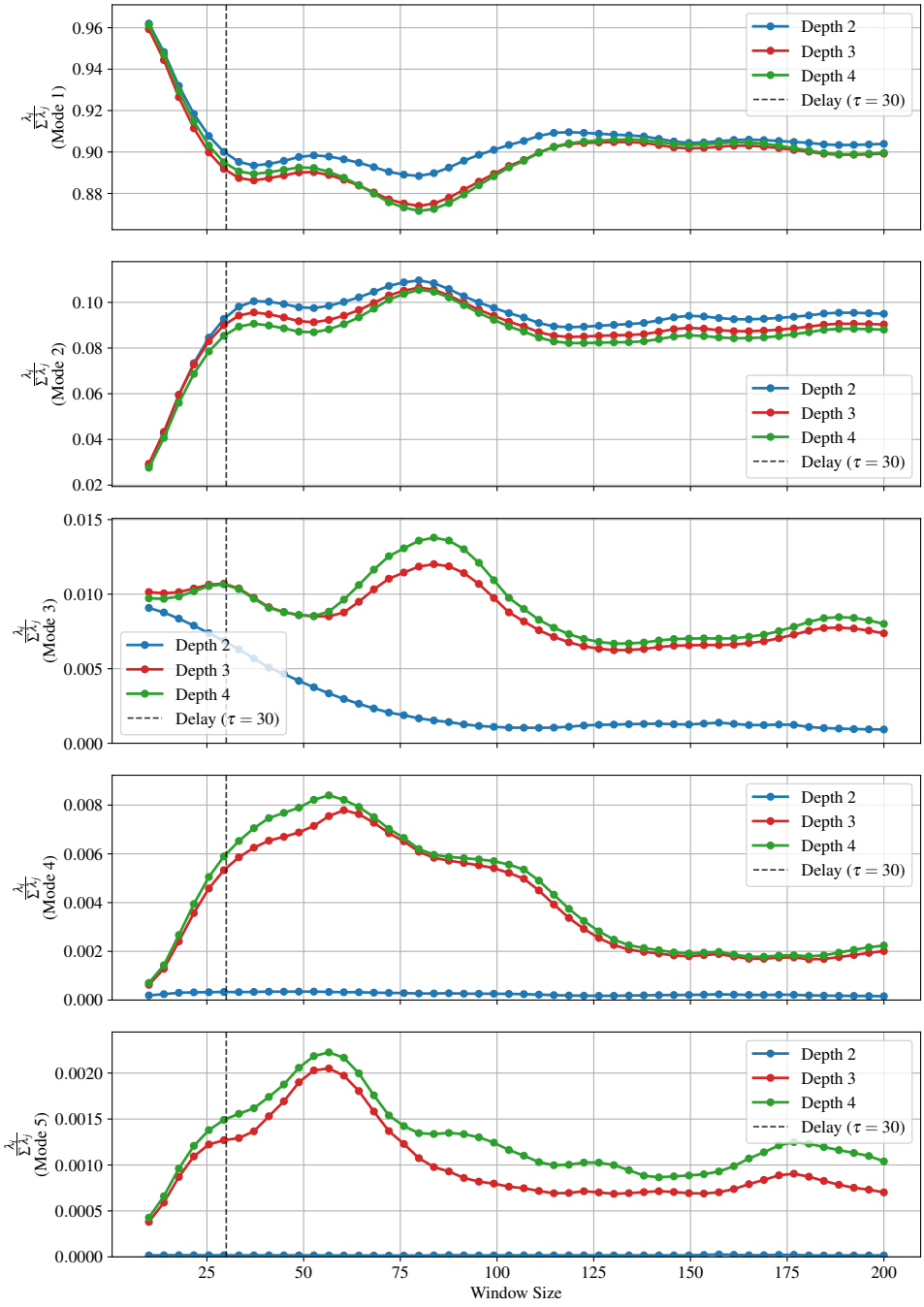


Figure 22: Evolution of mode ratio of the signature for various depths

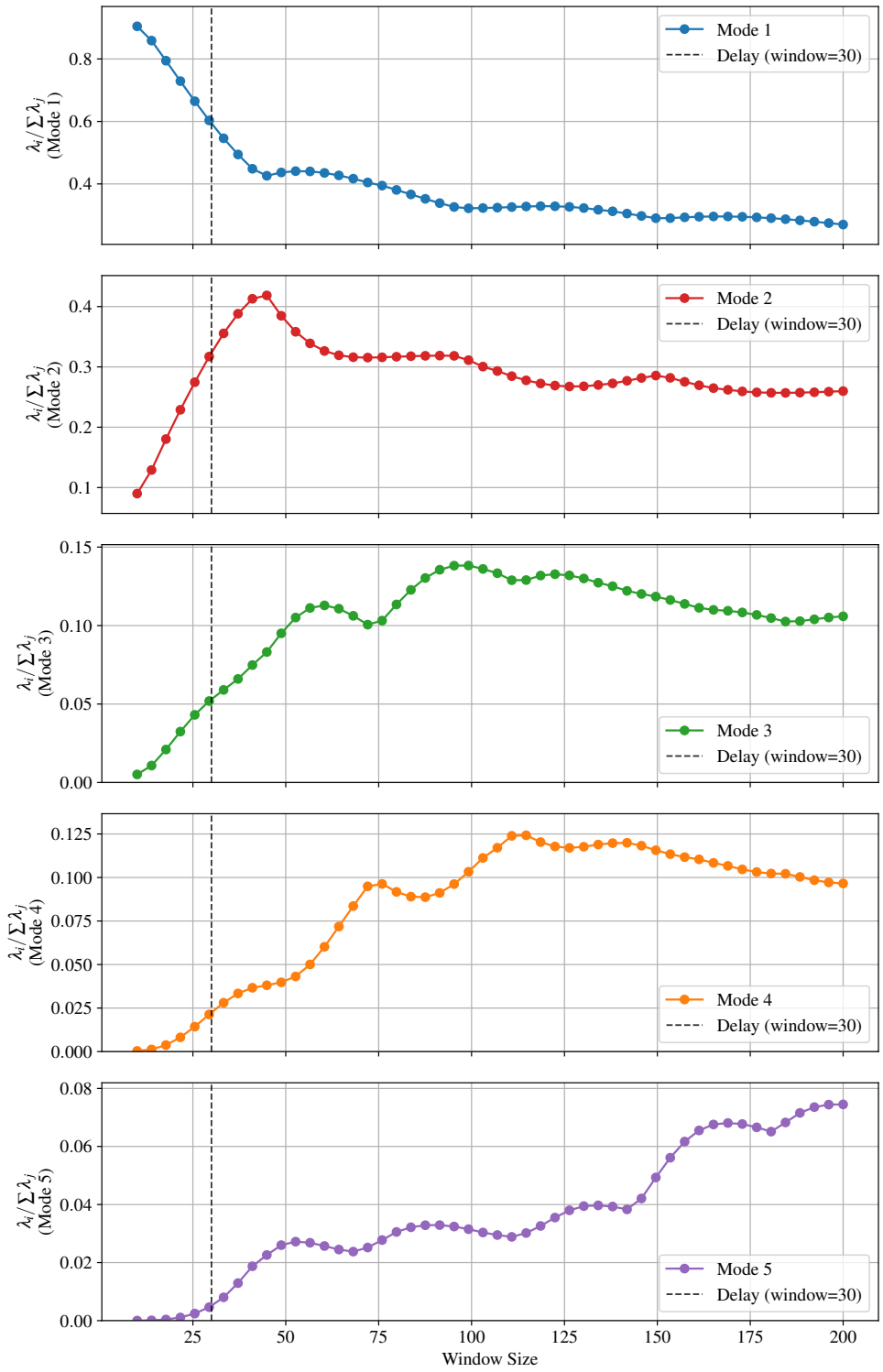


Figure 23: Evolution of mode ratio of the full trajectory

C Hyperparameter Selection

C.1 Linear System

Hyperparameter	CTAC (Markov)	CTAC (Signature)	CTAC (Full Trajectory)	Value Gradient
Critic learning rate η_c	1e-2	1e-2	1e-2	5e-4
Actor learning rate η_a	1e-3	1e-3	1e-3	not used
Weight initialization (critic / actor)	0.1/0.1	0.1/0.1	0.1/0.1	0.1/not used
Polyak averaging coefficient τ_{polyak}	not used	not used	not used	0.01
Number of training episodes N_{episodes}	1000	1000	1000	250
Exploration noise schedule $\sigma(k)$	linear decay, start value = 0.1	linear decay, start value = 0.1	linear decay, start value = 0.1	linear decay, start value = 0.1

Table 2: Main hyperparameters for the compared continuous-time RL methods for the linear system Equation 55.

C.2 Mackey-Glass

Hyperparameter	CTAC (Markov)	CTAC (Signature)	CTAC (Full Trajectory)	Value Gradient
Critic learning rate η_c	1e-3	1e-3	1e-3	1e-4
Actor learning rate η_a	1e-4	1e-4	1e-4	not used
Weight initialization (critic / actor)	0.01/0.01	0.01/0.01	0.01/0.01	0.01/not used
Polyak averaging coefficient τ_{polyak}	not used	not used	not used	0.01
Number of training episodes N_{episodes}	500	500	500	500
Exploration noise schedule $\sigma(k)$	linear decay, start value = 0.2	linear decay, start value = 0.1	linear decay, start value = 0.2	constant, start value = 0.01

Table 3: Main hyperparameters for the compared continuous-time RL methods for Mackey-Glass system Equation 56.

C.3 Chemical System

Hyperparameter	CTAC (Markov)	CTAC (Signature)	CTAC (Full Trajectory)	Value Gradient
Critic learning rate η_c	1e-3	1e-3	1e-3	1e-3
Actor learning rate η_a	1e-2	5e-2	1e-2	not used
Weight initialization (critic / actor)	0.2/0.2	0.2/0.2	0.1/0.1	0.1/not used
Polyak averaging coefficient τ_{polyak}	not used	not used	not used	0.01
Number of training episodes N_{episodes}	1500	1000	1500	1300
Exploration noise schedule $\sigma(k)$	linear decay, start value = 0.2	constant, start value = 0.1	linear decay, start value = 0.1	linear decay, start value = 0.1

Table 4: Main hyperparameters for the compared continuous-time RL methods for the chemical system Equation 57.

TRITA-EECS-EX 2025:0000
Stockholm, Sweden 2026

www.kth.se

€€€€ For DIVA €€€€

Author 1:
• Last name: Jobard
• First name: Joachim
• Local User Id: u1fxh7cs
• E-mail: jjrjo@kth.se
• Organisation: School of Electrical Engineering and Computer Science

Degree:
• Cycle: 2
• Course code: DA231X
• Credits: 30
• Program: CDATE
• Degree name: Degree of Master of Science in Engineering
• Subject area: Computer Science and Engineering

Title:
• Main title: Continuous Reinforcement Learning for Delayed Dynamical Systems
• Subtitle: A Path Signature-Based Representation Approach
• Language: eng

Supervisors:
• Supervisor 1: Lionel Mathelin (lionel.mathelin@cnrs.fr)
• Supervisor 2: Onofrio Semeraro (onofrio.semeraro@cnrs.fr)
• Supervisor 3: Rémy Hosseinkhan-Boucher (remy.hosseinkhan@universite-paris-saclay.fr)
• Supervisor 4: Erik Fransén (erikf@kth.se)

Examiner:
• Arvind Kumar (arvkumar@kth.se)
• School: School of Electrical Engineering and Computer Science

Keywords: Reinforcement Learning, Delayed Systems, Signatures, Machine Learning

Series: TRITA-EECS-EX 2025:0000