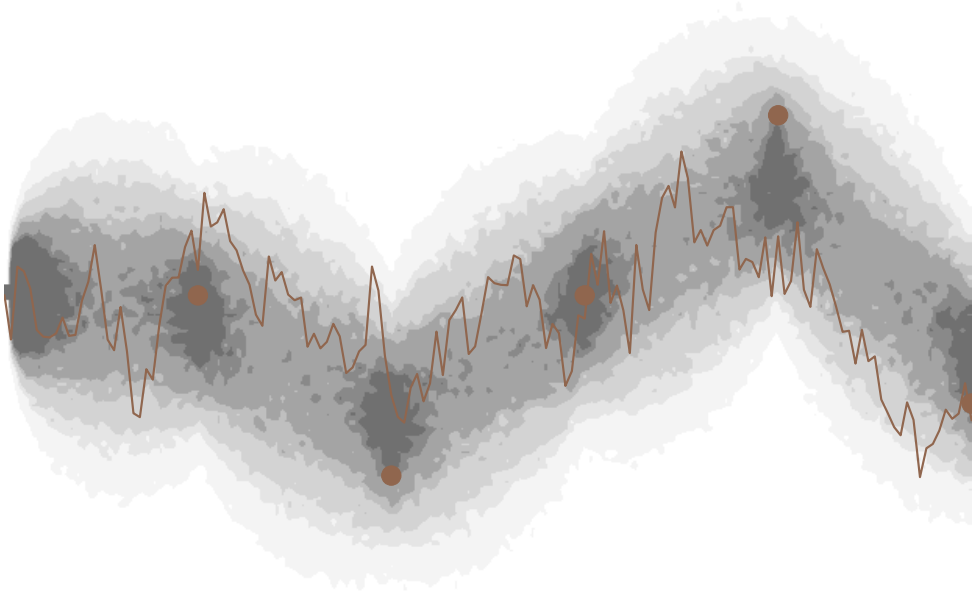


Learning Neural SDEs for Bayesian Filtering and Smoothing

A study of loss functions in adversarial training

Master's thesis in Engineering mathematics and computational science

Gustav Birath Blom and Isak Nilsson

Department of Mathematical Sciences

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Learning Neural SDEs for Bayesian Filtering and Smoothing

A study of loss functions in adversarial training

Gustav Birath Blom and Isak Nilsson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Saab AB
Chalmers University of Technology
Gothenburg, Sweden 2025

Learning Neural SDEs for Bayesian Filtering and Smoothing
A study of loss functions in adversarial training
Gustav Birath Blom and Isak Nilsson

Supervisor: Jimmy Aronsson, Karl Hammar and Benjamin Svedung Wettervik, Saab AB
Examiner: Moritz Schauer, Department of Mathematical Sciences

Master's Thesis 2025
Department of Mathematical Sciences
Engineering mathematics and computational sciences
Saab AB
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone: Gustav +46 73 840 44 88, Isak +46 76 766 88 14

Cover: Visualization of a path distribution generated from a linear Ornstein–Uhlenbeck SDE conditioned on a set of observations, with one representative sample path overlaid.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Learning Neural SDEs for Bayesian Filtering and Smoothing
A study of loss functions in adversarial training
Gustav Birath Blom and Isak Nilsson
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

This thesis investigates neural stochastic differential equations (neural SDEs) trained within a Wasserstein Generative Adversarial Network (WGAN) framework to approximate conditional probability distributions of trajectories, with applications in radar-based tracking. The study focuses on how different loss functions impact model performance for smoothing (estimating past states) and filtering (estimating current state) from noisy observations. Experiments in one- and two-dimensions show that neural SDEs effectively capture complex nonlinear dynamics and uncertainty relevant to radar tracking. Future research directions include extending the state space with additional physical quantities, incorporating Lévy jump processes, and refining loss functions for better accuracy around observations. In general, the study demonstrates the feasibility and versatility of WGAN-trained neural SDEs for Bayesian filtering and smoothing.

Keywords: Neural SDEs, Wasserstein GAN, Adversarial Training, Bayesian Inference, Path Signatures, Particle Filtering, Doob's h-Transform, Girsanov's Theorem, Filtering, Smoothing, Generative Models, Stochastic Processes.

Acknowledgements

We would like to express our sincere gratitude to our supervisors Karl Hammar, Jimmy Aronsson, and Benjamin Svedung Wettervik at Saab AB. Their deep technical knowledge and thoughtful guidance were invaluable at every stage of this project. Regardless of the complexity of the problems we faced, their support consistently helped us find the right direction. We are also grateful to our examiner at Chalmers University of Technology, Moritz Schauer, for his insightful feedback and for offering an academic perspective that enriched our work. Finally, we thank each other for countless discussions, disagreements, and collaborative breakthroughs. Our views sometimes drifted like stochastic processes, but we still found a path to convergence.

Gustav Birath Blom and Isak Nilsson, Gothenburg, June 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

BPF	Bootstrap Particle Filter
CDT	Computational Doob's h-Transform
ESS	Effective Sample Size
MC	Monte Carlo
SDE	Stochastic Differential Equation
WGAN	Wasserstein Generative Adversarial Network

Contents

List of Figures

List of Tables

1
Introduction

Military air defense systems are designed to protect people and infrastructure by constantly monitoring the airspace for potential threats. Central to these systems is radar technology, which uses electromagnetic pulses to detect and track objects such as missiles, drones, and aircraft. The signals reflected from these objects generate data that help estimate their trajectories, identify possible threats, and distinguish between friendly and hostile targets. The ability to precisely track objects in real time is vital to anticipate their behavior and effectively respond to emerging threats.

Modern air defense situations have become more challenging due to the increased complexity and unpredictable movement of newer threats. Traditional radar tracking methods often rely on simplified mathematical models and use excessive random noise to account for. Although these models are easy to compute, they may not capture the complex, nonlinear behavior seen in real scenarios and can lead to reduced accuracy.

To address this, we investigate the use of neural stochastic differential equations (neural SDEs) as a target model. Neural SDEs combine stochastic models, described using differential equations that include randomness, with the flexibility of neural networks to better capture complex behavior. This makes them especially interesting for radar-based tracking of difficult or unpredictable targets.

This thesis investigates how different ways of representing the observed trajectories, in particular through the choice of loss functions, affect how well neural SDEs can approximate the true conditional probability distribution of the path of the target. Specifically, we study how well these models can learn the distribution $\pi\Big((X_t)_{t\geq 0} \mid Z_{1:N}\Big)$, where X_t represents the latent state of the system at time t – such as position, velocity, or other relevant physical quantities – and $Z_{1:N}$ is a set of noisy observations at times $t_i, i = 1, \ldots, N$ along the trajectory.

We show that neural SDEs can effectively learn the true conditional distribution of the trajectories. In particular, we found that multiple loss functions perform well, and we were able to identify aspects of performance where one path representation outperforms another. Each method has its own strengths, making them useful tools in different modeling contexts. Although our experiments have focused on simplified test cases compared to reality, the strong performance of multiple methods suggests that neural SDEs could offer a valid and flexible approach in more complex tracking scenarios, such as those involving higher-dimensional systems or more realistic and uncertain dynamics. Successful adaptation to those scenarios would further demon-

strate their potential for use within tracking systems.

To provide the theoretical background and methodological development underlying these results, the thesis begins in Chapter 2 with the necessary mathematical foundations, introducing SDEs, conditional SDEs, and particle filtering. Chapter 3 presents the theory of path signatures, a method for capturing important structure in trajectories. In Chapter 4, we describe the Wasserstein Generative Adversarial Network (WGAN) framework used to train neural SDEs to generate trajectories conditioned on observed data. Furthermore, Chapters 5 and 6 detail the implementation of neural SDEs for smoothing (estimating past states) and filtering (estimating current state). These chapters also examine how different representations of paths influence model training, especially with respect to learning stability and convergence. Chapters 7 and 8 present the results for the smoothing and filtering problems, respectively, as applied to a set of test cases. Finally, Chapter 9 discusses these results in a broader context, highlighting both their implications and limitations. It also outlines several directions for future work, identifying opportunities to further improve model performance and extend the applicability of the WGAN-based neural SDE framework.

2

Stochastic Differential Equations

This chapter introduces the theory of stochastic differential equations (SDEs), which forms the basis for the models used throughout this thesis. We begin by showing when an SDE has a well-defined solution – specifically, when a strong unique solution exists. We then describe key mathematical tools for analyzing the evolution of such processes, including the Markov property, transition densities, and the infinitesimal generator.

Next, we show how conditioning SDEs on observations using Doob's h-transform results in a drift correction. Subsequently, we formulate an expression that the optimal drift correction satisfies. Naturally, this guides us towards the notion of path measures, which illustrates how the likelihood of different paths is altered under a particular drift adjustment, as explained by Girsanov's theorem.

These ideas enable the use of SDEs in particle filters, which are Monte Carlo (MC) methods that approximate distributions over trajectories with weighted particles. The chapter ends with a short description of the Euler–Maruyama method, which provides a simple numerical scheme to simulate solutions to SDEs.

2.1 Definition and properties of SDEs

Assume that $\big(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{P}\big)$ is a filtered probability space and let $W = (W_t)_{t\geq 0}$ be a d dimensional Brownian motion adapted to $(\mathcal{F}_t)_{t\geq 0}$. We start by defining an SDE in $\mathbb{R}^d$.

Definition 1 (Stochastic Differential Equation)**.** An SDE in $\mathbb{R}^d$ is given by

$$\mathrm{d}X_t \;=\; \mu(X_t, t)\,\mathrm{d}t \;+\; \sigma(X_t, t)\,\mathrm{d}W_t, \quad t \geq 0, \quad X_0 \sim Q, \tag{2.1}$$

where the drift $\mu : \mathbb{R}^d \times [0,\infty) \to \mathbb{R}^d$ and diffusion $\sigma : \mathbb{R}^d \times [0,\infty) \to \mathbb{R}^{d\times d}$ are measurable functions adapted to $(\mathcal{F}_t)_{t\geq 0}$, and X_0 (with $\mathcal{F}_0$-measurability) is the initial condition drawn from some distribution Q.

The standard way of writing an SDE in (2.1) is shorthand for the integral equation,

$$X_t \;=\; X_0 \;+\; \int_0^t \mu(X_s, s)\,\mathrm{d}s \;+\; \int_0^t \sigma(X_s, s)\,\mathrm{d}W_s, \quad t \geq 0.$$

where the second integral is defined in the Itô sense. Furthermore, a strong solution is a process $X = (X_t)_{t\geq 0}$ that is adapted to $(\mathcal{F}_t)_{t\geq 0}$ and satisfies the integral equation almost surely.

2.1.1 Conditions for strong solutions

To ensure that the above SDE (2.1) admits a unique strong solution, we must impose some conditions on μ and σ, as well as on the initial condition X_0. These assumptions are presented in the following.

1. **Lipschitz continuity.** There is a constant $K > 0$ such that for all $x, y \in \mathbb{R}^d$ and $t \geq 0$,
$$\|\mu(x,t) - \mu(y,t)\| \; + \; \|\sigma(x,t) - \sigma(y,t)\|_F \; \leq \; K\,\|x - y\|,$$
where $\|\cdot\|$ is the euclidean norm and $\|\cdot\|_F$ is the Frobenius norm.
2. **Linear (or affine) growth.** There is a constant $C > 0$ such that for all $x \in \mathbb{R}^d$ and $t \geq 0$,
$$\|\mu(x,t)\| \; + \; \|\sigma(x,t)\|_F \; \leq \; C\,\Big(1 + \|x\|\Big).$$
3. **Square-integrable initial condition.** The random vector X_0 is $\mathcal{F}_0$-measurable with $\mathbb{E}\Big[\|X_0\|^2\Big] < \infty$.

Under these hypotheses, the following theorem (based on a standard Picard iteration argument) asserts that there is exactly one strong solution to (2.1). We argue that these are reasonable assumptions for the target models in this project, which are physically derived SDEs.

Theorem 1 (Existence and Uniqueness of a Strong Solution)**.** *If μ and σ satisfy the Lipschitz and linear growth conditions and X_0 is square integrable, then there exists a unique strong solution X to* (2.1)*. In particular, there is a unique adapted process that satisfies*

$$X_t \; = \; X_0 \; + \; \int_0^t \mu\Big(X_s, s\Big)\,\mathrm{d}s \; + \; \int_0^t \sigma\Big(X_s, s\Big)\,\mathrm{d}W_s, \quad t \geq 0,$$

almost surely. For a detailed proof, see Theorem 5.2.1 in [1].

2.1.2 Markov property and transition densities

Having established the existence and uniqueness of solutions to SDEs, we now explore a fundamental property of these processes: the Markov property.

Definition 2 (Time-Inhomogeneous Markov Process)**.** A stochastic process X taking values in $\mathbb{R}^d$ is said to be a time-inhomogeneous Markov process if

$$\mathbb{P}\Big(X_{t+\tau} \in A \;\mid\; \mathcal{F}_t\Big) = \mathbb{P}\Big(X_{t+\tau} \in A \;\mid\; X_t\Big),$$

for all $\tau, t \geq 0$ and all Borel sets $A \subseteq \mathbb{R}^d$.

Intuitively, the Markov property states that the future evolution of X depends solely on its present state and the current time, rather than on the entire past trajectory. Furthermore, a concept that will be important in our setting is that of absolute continuity between measures.

Definition 3. Let $(\Omega, \mathcal{F})$ be a measurable space, and let $\mathbb{U}$ and $\mathbb{V}$ be two measures on $(\Omega, \mathcal{F})$. We say $\mathbb{V}$ is *absolutely continuous* with respect to $\mathbb{U}$ if for all $A \in \mathcal{F}$ the property $\mathbb{U}(A) = 0$ implies $\mathbb{V}(A) = 0$.

Next, with X being a Markov process, we define the propagator, which is an operator family $(\mathcal{T}_{t+\tau,t})_{t+\tau \geq t \geq 0}$, operating on suitable test functions f as

$$(\mathcal{T}_{t+\tau,t} f)(x) = \mathbb{E}[f(X_{t+\tau}) \mid X_t = x].$$

Choosing $f(X_t) = I_A(X_t)$ gives us the transition probability $P(t+\tau, A \mid X_t, t)$ as

$$\begin{aligned} P(t+\tau, A \mid X_t, t) &= (\mathcal{T}_{t+\tau,t} I_A)(X_t) \\ &= \mathbb{E}[I_A(X_{t+\tau}) \mid X_t] \\ &= \mathbb{P}(X_{t+\tau} \in A \mid X_t). \end{aligned}$$

When this kernel is absolutely continuous with respect to the Lebesgue measure on $\mathbb{R}^d$, the transition density $\rho(X_{t+\tau}, t+\tau \mid X_t, t)$ is defined as

$$\rho(X_{t+\tau} = y, t+\tau \mid X_t, t)\, \mathrm{d}y = \mathbb{P}(X_{t+\tau} \in \mathrm{d}y \mid X_t, t).$$

The difference between the transition probability and the transition density is that the transition probability gives the probability of moving from the state X_t to a Borel set A, while the transition density specifies the probability of transitioning from X_t to a specific point $X_{t+\tau}$. However, they are closely related in the sense that by integrating over the transition density one obtains the transition probability

$$P(t+\tau, A \mid X_t, t) \;=\; \int_A \rho(X_{t+\tau} = y, t+\tau \mid X_t, t)\, \mathrm{d}y.$$

To study functions of Markov processes, we now introduce a fundamental result: Itô's formula, which is the stochastic analogue of the chain rule.

Lemma 1 (Itô's formula)**.** *Assume that X is a Markov process and consider an arbitrary differentiable (scalar) function $\phi(X_t, t)$ of the process. Then the Itô differential of ϕ, that is, the Itô SDE for ϕ, is given as*

$$\mathrm{d}\phi(X_t, t) = \frac{\partial \phi(X_t, t)}{\partial t}\, \mathrm{d}t + (\nabla \phi(X_t, t))^\top\, \mathrm{d}X_t + \frac{1}{2} \left\langle \nabla^2 \phi(X_t, t),\, \mathrm{d}X_t\, \mathrm{d}X_t^\top \right\rangle_F,$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product.

Note that the gradient here is with respect to X_t. If nothing else is stated, this will always be the case in this report. Next, we introduce the infinitesimal generator using the propagator.

Definition 4 (Generator)**.** Let X be a Markov process taking values in $\mathbb{R}^d$. For any scalar test function $\phi \in C_K^\infty(\mathbb{R})$, the infinitesimal generator $\mathcal{A}$ acts on ϕ by

$$\mathcal{A}\phi(x) \;=\; \lim_{\tau \downarrow 0} \frac{(\mathcal{T}_{t+\tau,t})\phi(x) \;-\; \phi(x)}{\tau} \;=\; \lim_{\tau \downarrow 0} \frac{\mathbb{E}[\phi(X_{t+\tau}) \mid X_t = x] \;-\; \phi(x)}{\tau},$$

where the limit is with respect to the supremum-norm.

Furthermore, when X is a solution to an SDE, the infinitesimal generator can be written in differential operator form.

Lemma 2. *Consider a solution to the SDE*

$$\mathrm{d}X_t \;=\; \mu(X_t,t)\,\mathrm{d}t \;+\; \sigma(X_t,t)\,\mathrm{d}W_t.$$

Let ϕ be a scalar-valued function on $\mathbb{R}^d$. Then, the infinitesimal generator $\mathcal{A}$ acts on ϕ as

$$\mathcal{A}\phi(X_t) = \langle \mu, \nabla\phi(X_t)\rangle + \frac{1}{2}\left\langle \sigma\sigma^\top, \nabla^2\phi(X_t)\right\rangle_F.$$

A detailed proof is provided in Theorem 7.3.3 in [1], which is based on Itô's formula. The lemma tells us that the generator $\mathcal{A}$ completely encodes the instantaneous dynamics of the SDE by dictating how test functions $\phi(X_t)$ evolve on average over an infinitesimal increment of time. We can similarly show that for time dependent test functions $\phi(X_t,t)$ the (generalized) generator

$$\mathcal{A}_t\phi(X_t,t) \;=\; \lim_{\tau\downarrow 0}\frac{\mathbb{E}[\phi(X_{t+\tau},t+\tau)\mid X_t,]\;-\;\phi(X_t,t)}{\tau},$$

takes the form

$$\mathcal{A}_t\phi(X_t,t) = \frac{\partial\phi(X_t,t)}{\partial t} + \langle \mu, \nabla\phi(X_t,t)\rangle + \frac{1}{2}\left\langle \sigma\sigma^\top, \nabla^2\phi(X_t,t)\right\rangle_F = (\partial_t + \mathcal{A})\phi(X_t,t).$$

Note the extra partial derivative $\partial_t\phi(X_t,t) = \partial\phi(X_t,t)/\partial t$ when using a time dependent test function. With the foundational properties and operators of SDEs established, we now consider conditioning these processes on observational data.

2.2 Conditioned SDEs

Consider the SDE

$$\mathrm{d}X_t \;=\; \mu(X_t,t)\,\mathrm{d}t \;+\; \sigma(X_t,t)\;\mathrm{d}W_t, \quad t\geq 0, \quad X_0 = x_0. \tag{2.2}$$

Let

$$Z_k = X_{t_k} + \xi_k, \quad \xi_k \sim \mathcal{N}(0,\sigma_Z^2), \quad k = 1,2,...,N,$$

be a discrete-time measurement process, where t_k denotes the time of the k-th observation and σ_Z^2 is the variance of the independent Gaussian measurement noise ξ_k. We are interested in conditioning this SDE on observations $Z_{1:N}$. To do this, we will introduce the Doob's h-transform which provides a way to modify the behavior of a Markov process by reweighting its transition probabilities using a strictly positive function h. This transformation leads to a new SDE with adjusted drift, where h can be chosen to reflect the conditioning in which we are interested.

2.2.1 Doob's h-transform

We begin by establishing some properties needed for the derivation of the Doob's h-transform. Recall that $\rho(X_{t+\tau}, t+\tau \mid X_t, t)$ is the transition density of an SDE-driven Markov process X. We assume the existence of a strictly positive function $h : \mathbb{R}^d \times [0, \infty) \to (0, \infty)$ satisfying the space-time regularity condition

$$h(t, X_t) = \int \rho(X_{t+\tau} = y, t+\tau \mid X_t, t)\, h(t+\tau, X_{t+\tau} = y)\, \mathrm{d}y, \tag{2.3}$$

for all $t, \tau \geq 0$ and $x \in \mathbb{R}^d$. This property establishes a strong link between $h(t, X_t)$ and the future evolution of X_t. Using h, we define a new transition density

$$\rho^h(X_{t+\tau}, t+\tau \mid X_t, t) = \rho(X_{t+\tau}, t+\tau \mid X_t, t)\, \frac{h(t+\tau, X_{t+\tau})}{h(t, X_t)}. \tag{2.4}$$

Intuitively, this transformation modifies the probability of transitioning between states by reweighting the original kernel using the ratio $h(t+\tau, X_{t+\tau})/h(t, X_t)$. We show that this new transition density is indeed a probability density.

Proposition 1. *If h satisfies* (2.3)*, then $\rho^h(X_{t+\tau}, t+\tau \mid X_t, t)$ is a valid probability density, meaning*

$$\int \rho^h(X_{t+\tau} = y, t+\tau \mid X_t, t)\, \mathrm{d}y = 1,$$

for all $t, \tau \geq 0$ and $x \in \mathbb{R}^d$.

Proof. By substituting (2.4) we get

$$\int \rho^h(X_{t+\tau} = y, t+\tau \mid X_t, t)\, \mathrm{d}y = \int \rho(X_{t+\tau} = y, t+\tau \mid X_t, t) \frac{h(t+\tau, X_{t+\tau} = y)}{h(t, X_t)}\, \mathrm{d}y.$$

Rearranging, we obtain

$$\frac{1}{h(t, X_t)} \int \rho(X_{t+\tau} = y, t+\tau \mid X_t, t)\, h(t+\tau, X_{t+\tau} = y)\, \mathrm{d}y.$$

From (2.3), the integral simplifies to $h(t, X_t)$, leading to

$$\frac{h(t, X_t)}{h(t, X_t)} = 1.$$

Thus, $\rho^h(X_{t+\tau}, t+\tau \mid X_t, t)$ is a valid probability density. □

This result confirms that $\rho^h(\cdot \mid X_t, t)$ defines a proper Markov transition density, resulting in a new Markov process, denoted X^h. To derive Doob's h-transform, we must first establish an additional property of h.

Proposition 2. *In addition to* (2.3)*, let $\mathcal{A}_t$ be the infinitesimal generator of the original process. Then h satisfies*

$$\mathcal{A}_t h(t, X_t) = 0.$$

Proof. Recall that for a time-inhomogeneous Markov process with transition density $\rho(\cdot \mid X_t, t)$, the generator satisfies

$$\mathcal{A}_t\phi(X_t, t) = \lim_{\tau \to 0} \frac{\mathbb{E}[\phi(X_{t+\tau}, t+\tau) \mid X_t] - \phi(X_t, t)}{\tau},$$

for a test function ϕ. Substituting $\phi = h$ and applying (2.3) we get

$$\mathbb{E}[h(X_{t+\tau}, t+\tau) \mid X_t] = \int \rho(X_{t+\tau} = y, t+\tau \mid X_t, t) h(t+\tau, X_{t+\tau} = y)\, \mathrm{d}y = h(t, X_t).$$

Thus, the numerator simplifies to $h(t, X_t) - h(t, X_t) = 0$, giving

$$\mathcal{A}_t h(t, X_t) = \lim_{\tau \to 0} \frac{0}{\tau} = 0.$$

□

To see how Doob's h-transform affects the generator and in turn the drift term, we compute the generator $\mathcal{A}^h$ of the newly constructed process with transition density $\rho^h(X_{t+\tau}, t+\tau \mid X_t, t)$. Using the definition of p^h from (2.4) and applying the generator $\mathcal{A}^h\phi$, we obtain

$$\begin{aligned}
\mathcal{A}^h\phi &\coloneqq \lim_{\tau \downarrow 0} \frac{\mathbb{E}^h\big[\phi(X_{t+\tau}) \mid X_t] - \phi(X_t)}{\tau} \\
&= \lim_{\tau \downarrow 0} \frac{\mathbb{E}\big[\phi(X_{t+\tau}) h(t+\tau, X_{t+\tau}) \mid X_t] - \phi(X_t) h(t, X_t)}{\tau h(t, X_t)} \\
&= \frac{1}{h(t, X_t)} \mathcal{A}_t\{\phi(X_t) h(t, X_t)\}.
\end{aligned}$$

The arguments will not be needed, so we suppress t and X_t for the rest of the calculation. We use the product rule and rearrange terms to get

$$\mathcal{A}_t\{\phi h\} = \phi \cdot \mathcal{A}_t h + \langle \mu, \nabla\phi \cdot h \rangle + \left\langle \sigma\sigma^\top, \nabla\phi\, \nabla h \right\rangle_F + \frac{1}{2} \left\langle \sigma\sigma^\top, \nabla^2\phi \cdot h \right\rangle_F.$$

Since $\mathcal{A}_t h = 0$, the first term vanishes. Hence,

$$\mathcal{A}^h\phi = \langle \mu, \nabla\phi \rangle + \left\langle \sigma\sigma^\top, \nabla\phi\, \frac{\nabla h}{h} \right\rangle_F + \frac{1}{2} \left\langle \sigma\sigma^\top, \nabla^2\phi \right\rangle_F.$$

It follows (by identification of terms) that the new, h-transformed SDE can be written as

$$\begin{aligned}
\mathrm{d}X_t &= \left[\mu(X_t, t) \;+\; \sigma(X_t, t)\, \sigma^\top(X_t, t)\, \frac{\nabla h(t, X_t)}{h(t, X_t)} \right] \mathrm{d}t \;+\; \sigma(X_t, t)\, \mathrm{d}W_t \\
&= \Big[\mu(X_t, t) \;+\; \sigma(X_t, t)\, \sigma^\top(X_t, t)\, \nabla \log h(t, X_t) \Big]\, \mathrm{d}t \;+\; \sigma(X_t, t)\, \mathrm{d}W_t.
\end{aligned}$$

2.2.2 Conditioning the SDE on observations

Incorporating the Doob's h-transform into our problem formulation is quite natural. Instead of considering the usual transition density $\rho(X_{t+\tau} = y, t+\tau \mid X_t, t)$, we now condition it on the sequence of observations $Z_{1:N}$, leading to

$$\rho^h(X_{t+\tau}, t+\tau \mid X_t, t) = p(X_{t+\tau} \mid X_t, Z_{1:N}).$$

To proceed, we assume that the time index satisfies $t_k < t < t+\tau \leq t_{k+1}$ and apply the following steps

$$\begin{aligned}
p(X_{t+\tau} \mid X_t, Z_{1:N}) &= \frac{p(X_{t+\tau}, X_t, Z_{1:N})}{p(X_t, Z_{1:N})} \\
&= \frac{p(Z_{k+1:N} \mid X_{t+\tau}, X_t, Z_{1:k}) p(X_{t+\tau}, X_t, Z_{1:k})}{p(Z_{k+1:N} \mid X_t, Z_{1:k}) p(X_t, Z_{1:k})} \\
&= \frac{p(Z_{k+1:N} \mid X_{t+\tau}) p(X_{t+\tau} \mid X_t, Z_{1:k}) p(X_t, Z_{1:k})}{p(Z_{k+1:N} \mid X_t, Z_{1:k}) p(X_t, Z_{1:k})} \\
&= p(X_{t+\tau} \mid X_t) \frac{p(Z_{k+1:N} \mid X_{t+\tau})}{p(Z_{k+1:N} \mid X_t)}.
\end{aligned}$$

Next, we define the function

$$h(t+\tau, X_{t+\tau}) = p(Z_{k+1:N} \mid X_{t+\tau}),$$

which allows us to express the reweighted transition density as

$$\rho^h(X_{t+\tau}, t+\tau \mid X_t, t) = \rho(X_{t+\tau}, t+\tau \mid X_t, t) \frac{h(t+\tau, X_{t+\tau})}{h(t, X_t)}.$$

Thus, we can interpret the transformed process as one that follows a modified SDE, provided that h satisfies the condition required by (2.3). To verify this, we compute

$$\begin{aligned}
&\int \rho(X_{t+\tau} = y, t+\tau \mid X_t, t) h(t+\tau, X_{t+\tau} = y) \, \mathrm{d}y \\
&\int \rho(X_{t+\tau} = y, t+\tau \mid X_t, t) p(Z_{k+1:N} \mid X_{t+\tau} = y) \, \mathrm{d}y \\
&= \int p(X_{t+\tau} = y \mid X_t) p(Z_{k+1:N} \mid X_{t+\tau} = y) \, \mathrm{d}y \\
&= \int p(Z_{k+1:N}, X_{t+\tau} = y \mid X_t) \, \mathrm{d}y \\
&= p(Z_{k+1:N} \mid X_t) = h(t, X_t).
\end{aligned}$$

The derivation above has led us to the h-transformed SDE that governs the conditioned process.

Theorem 2 (Doob h-transformed SDE conditioned on observations)**.** *Given a set of observations $Z_{1:N}$, the original SDE (2.2), when conditioned on these observations, defines a new process whose evolution is governed by the modified SDE*

$$\mathrm{d}X_t = \Big[\mu(X_t, t) + \sigma(X_t, t)\,\sigma^\top(X_t, t)\nabla \log p(Z_{k+1:N} \mid X_t)\Big]\,\mathrm{d}t + \sigma(X_t, t)\,\mathrm{d}W_t,$$

for $t \in [t_k, t_{k+1}]$ and initial condition $X_0 = x_0$.

It is important to note that the term $\nabla \log p(Z_{k+1:N} \mid X_t)$ generally does not admit an analytical solution. Although closed-form expressions can be derived for certain SDEs, such as linear ones, this is typically not possible for more complex processes. This term contains significant information that informs the modeling framework for neural SDEs. In particular, it reveals that only future observations influence the adjusted drift term. Moreover, the fact that this quantity represents the gradient of the observation log-likelihood with respect to the state X_t provides additional insight into the underlying process, even if it does not directly impact the formulation of the neural SDE. Intuitively, it means that we *steer* the process so that future observations have higher likelihood.

We have now established the necessary theory for the smoothing problem, that is, estimating the distribution of past states given past, current, and future observations. Specifically, we have shown that the conditioned SDE has a modified drift term that depends on the current state, time, and only future observations. In the next sections, we build on this theory, leading up to how SDEs can be used within particle filters to solve the filtering problem, that is, estimating the distribution of the current state given observations up to the current time.

2.2.3 Optimal control perspective

As we saw in Section 2.2.2, Doob's h-transform gives rise to an additional drift term $\sigma\sigma^\top \nabla \log h$ in the SDE. In our case $h(X_t, t, Z_{1:N}) = p(Z_{k+1:N} \mid X_t)$. In this section, we consider a controlled SDE, which is the original SDE plus an additional drift term. It turns out that the conditional SDE produced by Doob's h-transform satisfies a certain optimal control problem. This is helpful because the optimal control perspective is both instructive and gives access to other solution methods. Here, we only present a rough sketch of the approach, and refer to [2] for the inspiration for this presentation.

In this context, we only consider the case of a single observation ($N = 1$). Assume that the observation $Z = Z_1$ is made in the end at time T. Furthermore, we only consider time-homogeneous SDEs, meaning μ and σ only depends on X_t. We define the *control function* $u : \mathbb{R}^d \times [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$. A *controlled SDE* is an SDE on the form

$$\mathrm{d}X_t^{u,Z} = \Big[\mu(X_t^{u,Z}) + \sigma(X_t^{u,Z})u(X_t^{u,Z}, t, Z)\Big]\,\mathrm{d}t + \sigma(X_t^{u,Z})\,\mathrm{d}W_t,$$

where $X^{u,Z}$ is a controlled process. To complete the control problem, a cost functional needs to be specified. It is possible to choose it such that the optimal control is given

by Doob's h-trasform, see [3]. We then have the optimal control

$$u_\star(X_t, t, Z) = \sigma^\top(X_t)\nabla \log p(Z \mid X_t).$$

We recall that h satisfies the equation

$$(\partial_t + \mathcal{A})h = 0 \tag{2.5}$$

with the terminal condition $h(X_T, T, Z) = p(Z \mid X_T)$. Define the *value function* as $v(X_t, t, Z) = -\log[h(X_t, t, Z)]$. Given that h satisfies (2.5), one can verify that the value function v must satisfy

$$(\partial_t + \mathcal{A})v = \frac{1}{2}\|\sigma^\top \nabla v\|^2$$

with the terminal condition $v(X_T, T, Z) = -\log[p(Z \mid X_T)]$. Let $X^{u,Z}$ be a controlled process, driven by the Brownian motion W and with a given control u, and define the process $V_t = v(X_t^{u,Z}, t, Z)$. The generator of the controlled process $X^{u,Z}$ is then given by

$$\mathcal{A}^{u,t,Z}\phi = \mathcal{A}\phi + \langle \sigma u, \nabla\phi \rangle .$$

Thus, applying the generalized generator to the value function v yields

$$(\partial_t + \mathcal{A}^{u,t,Z})v = \frac{1}{2}\|\sigma^\top \nabla v\|^2 + \langle \sigma u, \nabla v\rangle = \frac{1}{2}\|\sigma^\top \nabla v\|^2 + \left\langle u, \sigma^\top \nabla v \right\rangle . \tag{2.6}$$

Let $Y_t = \sigma^\top(X_t^{u,Z})\nabla v(X_t^{u,Z}, t, Z)$. Applying Itô's formula on V_t and using (2.6) results in

$$V_T = V_s + \int_s^T \frac{1}{2}\|Y_t\|^2 + \langle u, Y_t\rangle \, \mathrm{d}t + \int_s^T \langle Y_t, \mathrm{d}W_t\rangle , \tag{2.7}$$

with $V_T = -\log[p(Z \mid X_T^{u,Z})]$. Note here that W in the second integral is the same Brownian motion that generated $X^{u,Z}$ and that the pair of processes (V_t, Y_t) is the unique solution to (2.7). This system of equations constitutes a so called forward backward stochastic differential equation (FBSDE) where V is a backward SDE. Crucially, the control problem is constructed such that if we solve the FBSDE, the optimal control is obtained.

2.2.4 Change of measure and path likelihoods

In this section we will show how the likelihood of paths changes when we change the SDE. In particular, we are interested in how the likelihood of a fixed path changes as we go from the original SDE to the controlled SDE. To do this, we need the Radon–Nikodym derivative. Using this, we introduce Girsanov's theorem, which will tell us how the likelihood of paths changes from a change in measure.

We start by stating the Radon–Nikodym theorem, which defines the Radon–Nikodym derivative.

Theorem 3 (Radon–Nikodym theorem)**.** *Let $(\Omega, \mathcal{F})$ be a measurable space and let $\mathbb{U}$ and $\mathbb{V}$ be two measures on $(\Omega, \mathcal{F})$. If $\mathbb{V}$ is absolutely continuous with respect to $\mathbb{U}$, then there exists an $\mathcal{F}$-measurable function $f : \Omega \to [0, \infty)$ such that for any measurable set $A \in \mathcal{F}$*

$$\mathbb{V}(A) = \int_A f \, \mathrm{d}\mathbb{U}.$$

The function f satisfying this property is unique in the sense that if another function g also satisfies the property, then $f = g$ $\mathbb{U}$-almost everywhere. We call the function f the *Radon–Nikodym derivative* and denote it

$$\frac{\mathrm{d}\mathbb{V}}{\mathrm{d}\mathbb{U}} = f.$$

We now introduce Girsanov's theorem, which describes how the dynamics of a stochastic process changes when we switch to a new probability measure.

Theorem 4 (Girsanov's theorem)**.** *Let W be a standard d-dimensional Brownian motion under the probability measure $\mathbb{U}$. Let $\theta : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ be an adapted process such that the process*

$$F(X_t, t) = \exp\left(-\frac{1}{2}\int_0^t \|\theta(X_t, t)\|^2 \, \mathrm{d}t + \int_0^t \langle \theta(X_t, t), \mathrm{d}W_t \rangle\right)$$

satisfies $\mathbb{E}[F(X_t, t)] = 1$. Then the process

$$\tilde{W}_t = W_t - \int_0^t \theta(X_s, s) \, \mathrm{d}s \tag{2.8}$$

is a Brownian motion under the probability measure $\mathbb{V}$ defined by

$$\frac{\mathrm{d}\mathbb{V}}{\mathrm{d}\mathbb{U}} = F(X_t, t).$$

In essence, Girsanov's theorem shows us how we can construct a new measure and a new process such that the new process under the new measure is a Brownian motion. To put Girsanov's theorem into our context, let $\mathbb{P}$ and $\mathbb{P}^{u,Z}$ be the probability measures from the original and controlled SDE respectively. We can write (2.8) from the theorem on differential form so that

$$\mathrm{d}W_t = \mathrm{d}\tilde{W}_t + \theta(X_t, t) \, \mathrm{d}t.$$

Substituting $\mathrm{d}W_t$ in the original SDE gives

$$\mathrm{d}X_t = (\mu(X_t) + \sigma(X_t)\theta(X_t, t)) \, \mathrm{d}t + \sigma(X_t) \, \mathrm{d}\tilde{W}_t.$$

Choosing $\theta(X_t, t) = u(X_t, Z)$, we obtain the controlled SDE under the measure $\mathbb{P}^{u,Z}$. Girsanov's theorem also tells us the Radon–Nikodym derivative

$$\frac{\mathrm{d}\mathbb{P}^{u,Z}}{\mathrm{d}\mathbb{P}} = \exp\left(-\frac{1}{2}\int_0^t \|u(X_t, t, Z)\|^2 \, \mathrm{d}t + \int_0^t \langle u(X_t, t, Z), \mathrm{d}W_t \rangle\right). \tag{2.9}$$

To gain some intuition, consider a path generated by the original SDE. According to Girsanov's theorem, the probability that this path could have been generated by a controlled SDE with a large value of $\|u\|$ is low. On the other hand, the probability that the path was generated by a controlled SDE with a small value of $\|u\|$ is high.

2.3 Particle filter

We have shown that the Radon–Nikodym derivative and Girsanov's theorem provide the basis for changing probability measures. In particular, they make precise how modifying the drift of an SDE corresponds to a specific change in measure, allowing us to quantify how the likelihood of whole paths is affected. This is especially useful in the context of particle filters.

The particle filter represents an approximate Bayesian filtering method based on MC simulation. This technique approximates the posterior distribution $p(X_{0:t_{k-1}}|Z_{1:k-1})$ through a set of weighted particle trajectories $\{X^i_{0:t_{k-1}}, \pi^i_{t_{k-1}}\}_{i=1}^I$, where I is the number of particles, as

$$p(X_{0:t_{k-1}}|Z_{1:k-1}) \approx \sum_{i=1}^{I} \pi^i_{t_{k-1}} \delta(X_{0:t_{k-1}} - X^i_{0:t_{k-1}}).$$

In this representation, $X^i_{0:t_{k-1}} = \{X_{t_0}, X_{t_1}, \dots X_{t_{k-1}}\}$ signifies the trajectory for the i-th particle from time step 0 to time step t_{k-1} and $\delta(\cdot)$ denotes the Dirac delta function. Each particle trajectory has an associated normalized weight $\pi^i_{t_{k-1}}$. Note that this formulation also implies that the PF performs smoothing, since we update past states given new observations. In simple settings, such as those with a small number of observations, low dimensionality, and high observation noise, this smoother suffices. On the contrary, in more complex scenarios, one should opt for a more precise smoother, such as the forward–backward smoother [4].

If particles $X^i_{0:t_{k-1}}$ are sampled from a chosen proposal density $q(X_{0:t_{k-1}}|Z_{1:k-1})$, then the weights $\pi^i_{t_{k-1}}$ are calculated according to

$$\pi^i_{t_{k-1}} \propto \frac{p(X^i_{0:t_{k-1}}|Z_{1:k-1})}{q(X^i_{0:t_{k-1}}|Z_{1:k-1})}. \tag{2.10}$$

Given the particle set $\{X^i_{0:t_{k-1}}, \pi^i_{t_{k-1}}\}_{i=1}^I$ at time t_{k-1}, upon arrival of a new observation Z_k, an updated particle set $\{X^i_{0:t_k}, \pi^i_{t_k}\}_{i=1}^I$ is generated to approximate the posterior $p(X_{0:t_k}|Z_{1:k})$. To achieve this posterior, the proposal density is chosen to factorize as

$$q(X_{0:t_k}|Z_{1:k}) = q(X_{t_k}|X_{0:t_{k-1}}, Z_{1:k}) q(X_{0:t_{k-1}}|Z_{1:k-1}). \tag{2.11}$$

Thus, new particles $X^i_{0:t_k}$ are obtained by augmenting previous particle trajectories $X^i_{0:t_{k-1}} \sim q(X_{0:t_{k-1}}|Z_{1:k-1})$ with newly generated particles $X^i_{t_k} \sim q(X_{t_k}|X^i_{0:t_{k-1}}, Z_{1:k})$. Using Bayes' theorem, it follows that

$$p(X_{0:t_k}|Z_{1:k}) \propto p(Z_k|X_{t_k}) p(X_{t_k}|X_{t_{k-1}}) p(X_{0:t_{k-1}}|Z_{1:k-1}). \tag{2.12}$$

Substituting (2.11) and (2.12) into (2.10) the weight updates become

$$\pi^i_{t_k} \propto \pi^i_{t_{k-1}} \frac{p(Z_k|X^i_{t_k}) p(X^i_{t_k}|X^i_{t_{k-1}})}{q(X^i_{t_k}|X^i_{0:t_{k-1}}, Z_{1:k})}. \tag{2.13}$$

A resampling step is introduced to mitigate the problem of particle degeneracy, which happens when most of the weight is increasingly concentrated in just a few particles. Resampling involves generating a new set of particles by sampling with replacement of existing particles according to their weights $\pi^i_{t_k}$. This step removes particles with low weights and duplicates those with higher weights, effectively resetting all weights to be uniform $\pi^i_{t_k} = 1/I$ for all i. As a result, we get a better representation for the next step, since we have more particles with contributing weights. Consequently, after resampling, the weights in (2.13) updates as

$$\pi^i_{t_k} \propto \frac{p(Z_k|X^i_{t_k})p(X^i_{t_k}|X^i_{t_{k-1}})}{q(X^i_{t_k}|X^i_{0:t_{k-1}}, Z_{1:k})}.$$

Until now, we assumed discrete transitions between states in the particle filter. We now extend this framework to incorporate continuous-time dynamics by modeling the state evolution between observations with SDEs. This extension becomes essential when observations are sparsely sampled in time, since the transition density becomes intractable. In addition, we are interested in the paths between detection times. We note that the ratio

$$\frac{p(X^i_{t_k} \mid X_{t_{k-1}})}{q(X^i_{t_k} \mid X^i_{0:t_{k-1}}, Z_{1:k})}$$

is the Radon-Nikodym derivative of the transition measure with respect to the proposal measure in the discrete case. Modeling transitions with SDEs means that we instead use Girsanov's theorem (2.9). This yields the weights

$$\begin{aligned}\pi^i_{t_k} &= \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{P}^{u,Z}}\, p(Z_k \mid X^i_{t_k}) \qquad (2.14)\\ &= \exp\Bigg\{ -\frac{1}{2}\int_{t_{k-1}}^{t_k} \Big\|u(X^i_t, t-t_{k-1}, Z_k)\Big\|^2 \,\mathrm{d}t \\ &\qquad - \int_{t_{k-1}}^{t_k} \Big\langle u(X^i_t, t-t_{k-1}, Z_k), \mathrm{d}W^i_t \Big\rangle \Bigg\} p(Z_k \mid X^i_{t_k})\end{aligned}$$

which can be re-written using (2.7) as

$$\exp\Big\{V^i_{t_k} + \log p(Z_k \mid X^i_{t_k}) - V_0(X_{t_{k-1}}, Z_k)\Big\}.$$

The results at the end of Section 2.2.3 imply that, if we have an optimal control, the first two terms sum to 0 and we are left with

$$\exp\Big\{-V_0(X_{t_{k-1}}, Z_k)\Big\}.$$

In other words, the weights only depend on the initial position $X_{t_{k-1}}$ and more preciously on how well it aligns with the next observation.

This result shows that any SDE with a drift correction can serve as a proposal in the particle filter. We now turn to why the choice of correction matters, specifically how it affects the efficiency and accuracy of the filtering process.

2.3.1 Locally optimal proposals

The choice of the proposal distribution q is critical to the effectiveness of the particle filter. A well-chosen proposal can significantly reduce the number of particles required to accurately approximate the posterior distribution. To illustrate this, we consider the simplest form of particle filter, known as the Bootstrap Particle Filter (BPF), which employs an unconditional proposal distribution given by the original SDE. Under this choice, the weight update (2.14) simplifies to

$$\pi_{t_k}^i \propto p(Z_k|X_{t_k}^i).$$

The BPF can lead to a poor approximation of the posterior distribution. This occurs because only a small fraction of the proposed particles are likely under the observation model, particularly in cases of low observation noise or a high-dimensional state space. This motivates the use of a proposal distribution that incorporates information from the current observation, in order to improve sampling efficiency and reduce the computational burden associated with using an unconditional proposal.

The locally optimal proposal is the proposal that minimizes the variance of the weight increments or, equivalently, minimizes the Kullback—Leibler divergence between the proposed particles $p(X_{t_{k-1}} \mid Z_{1:k-1})q(X_{t_k} \mid X_{0:t_{k-1}}, Z_{1:k})$ and the true conditional distribution $p(X_{t_k}|Z_{1:k})$ [5].

2.4 Euler–Maruyama scheme

When an analytical solution to the SDE

$$\mathrm{d}X_t = \mu(X_t, t)\,\mathrm{d}t + \sigma(X_t, t)\,\mathrm{d}W_t, \quad X_0 = x_0,$$

is not available, a common approach to simulating its trajectories is the Euler–Maruyama method. This method is the stochastic analogue of the classical Euler method for ordinary differential equations.

Assume that we discretize the time interval $[0, T]$ into H steps with the step size $\Delta t = T/H$. Denote the discrete times by $t_j = j\Delta t$ for $j = 0, 1, \ldots, H$. Then the Euler–Maruyama approximation for the process X is given by

$$X_{t_{j+1}} = X_{t_j} + \mu(X_{t_j}, t_j)\,\Delta t + \sigma(X_{t_j}, t_j)\,\Delta W_{t_j},$$

where $\Delta W_{t_j} \sim \mathcal{N}(0, \Delta t\,\mathbf{I})$ is the Brownian increment over the interval $[t_j, t_{j+1}]$. Thus, the method approximates the continuous-time process by iteratively updating the state, using the drift μ and diffusion σ evaluated in the current state and time. In the strong sense, the mean square error of this approximation converges at a rate proportional to $\sqrt{\Delta t}$.

3

Signature

Our goal is to train neural SDEs to generate sample paths that resemble those seen in real data. To evaluate how similar the generated paths are to actual ones, we need a way to represent paths that captures their essential structure in a manageable and informative form. The signature transform provides this by encoding a path as a sequence of iterated integrals, capturing its geometric and temporal features.

We start by defining what it means for a path to have finite p-variation.

Definition 5. Let $X : J \to \mathbb{R}^d$ be a d-dimensional path in a compact interval $J = [0, T]$. We say that X has *finite p-variation* for $p \geq 1$ if

$$\|X\|_{p,J} = \left(\sup_{\mathcal{P}} \sum_{l} \|X_{t_l} - X_{t_{l-1}}\|^p \right)^{1/p} < \infty,$$

where the supremum is taken over all finite partitions $\mathcal{P}$ on J.

Let $\mathcal{V}^p(J, \mathbb{R}^d)$ be the set of paths $X : J \to \mathbb{R}^d$ with finite p-variation. To introduce the reader to the signature, we will start by considering paths with a continuous derivative. Later we will extend the idea to continuous paths of finite p-variation.

Recall the concept of path (or line) integration. Given two paths $X^1, X^2 \in C^1(J, \mathbb{R})$, the path integral of X^1 against X^2 over J is defined as

$$\int_0^T X_t^1 \dot{X}_t^2 \, \mathrm{d}t,$$

where we denote $\dot{X}_t^2 = \mathrm{d}X_t^2/\mathrm{d}t$.

Example 1. Consider the 3-dimensional path $X = (X^1, X^2, X^3) = (1, t, t^2)$ for $t \in [0, 1]$. We get the path integral of X^1 against X^2 as

$$\int_0^1 X_t^1 \dot{X}_t^2 \, \mathrm{d}t = \int_0^1 \mathrm{d}t = 1.$$

Similarly, the path integral of X^2 against X^3 is computed as

$$\int_0^1 X_t^2 \dot{X}_t^3 \, \mathrm{d}t = \int_0^1 t \cdot 2t \, \mathrm{d}t = \frac{2}{3}.$$

Now, for any d-dimensional path $X = (X^1, \dots, X^d)$ with component functions $X^i : J \to \mathbb{R}$ for all $i \in \{1, \dots, d\}$, we let $\mathrm{d}X_s^i = \dot{X}_s^i \, \mathrm{d}s$. We can then define the quantity

$$S(X)_{0,t}^i = \int_0^t \mathrm{d}X_s^i = \int_0^t \dot{X}_s^i \, \mathrm{d}s.$$

Note that $S(X)_{0,\cdot}^i : J \to \mathbb{R}$ itself is a path. We can thus integrate it against another path, and so define the following transformation for all $i, j \in \{1, \dots, d\}$:

$$S(X)_{0,t}^{i,j} = \int_0^t S(X)_{0,s}^i \, \mathrm{d}X_s^j.$$

Since $S(X)_{0,s}^i$ itself can be seen as an integral, an equivalent way to define $S(X)_{0,t}^{i,j}$ would be as two nested integrals,

$$S(X)_{0,t}^{i,j} = \int_0^t \int_0^s \mathrm{d}X_r^i \, \mathrm{d}X_s^j.$$

Once again, $S(X)_{0,\cdot}^{i,j}$ is just another path. To really generalize this iteration of integrals, for $i_1, \dots, i_k \in \{1, \dots, d\}$ we define

$$S(X)_{0,t}^{i_1,\dots,i_k} = \int_0^t S(X)_{0,s}^{i_1,\dots,i_{k-1}} \, \mathrm{d}X_s^{i_k}$$

or equivalently,

$$S(X)_{0,t}^{i_1,\dots,i_k} = \int_0^t \int_0^{t_k} \cdots \int_0^{t_2} \mathrm{d}X_{t_1}^{i_1} \dots \mathrm{d}X_{t_{k-1}}^{i_{k-1}} \, \mathrm{d}X_{t_k}^{i_k}.$$

We now extend the idea to paths of finite p-variation. Let

$$\mathcal{P} = \{0 = x_0 < x_1 < \cdots < x_n = T\}$$

be a finite partition on J. We define the norm of the partition $\mathcal{P}$ as

$$\|\mathcal{P}\| = \max_{1 \le k \le n} x_k - x_{k-1}.$$

Definition 6. Let $f, g \in \mathcal{V}^p(J, \mathbb{R})$. The *Riemann–Stieltjes integral* of f over J with respect to g is defined as

$$\int_0^T f \, \mathrm{d}g := \lim_{\|\mathcal{P}\| \to 0} \sum_{\mathcal{P}} f(\alpha_k)(g(x_k) - g(x_{k-1})),$$

where $\alpha_k \in [x_{k-1}, x_k]$, $k = 1, \dots, n$.

This definition includes a choice of the point $\alpha_k \in [x_{k-1}, x_k]$ for all k. However, this choice is proved in [6] to be irrelevant as long as f is bounded on J and both f, g are continuous on J.

Since we will typically deal with paths without continuous derivative, we extend the iterated integrals to paths $X \in \mathcal{V}^p(J, \mathbb{R}^d)$ by viewing them in the Riemann–Stieltjes sense. With all these combinations of iterated integrals, we are now ready to define the signature.

Definition 7. The *signature* of a d-dimensional path $X : J \to \mathbb{R}^d$, denoted by $S(X)_{0,T}$, is the infinite collection of all iterated integrals

$$S(X)_{0,T} = (1, S(X)^1_{0,T}, \dots, S(X)^d_{0,T}, S(X)^{1,1}_{0,T}, S(X)^{1,2}_{0,T}, \dots)$$

with indices $i_1, \dots, i_k \in \{1, \dots, d\}, k \in \mathbb{N}$ and where $S^{\varnothing}_{0,T} = 1$ by convention. In addition, we define *truncated signature* $S^m(X)_{0,T}$ of order $m \geq 1$ as the finite collection of iterated integrals with indices $i_1, \dots, i_k \in \{1, \dots, d\}, k \in \{0, \dots, m\}$.

We always use the integration interval $[0, T]$. Therefore, we drop the subscript and simply denote the signature of a path X as $S(X)$, or $S^m(X)$ for the truncated signature.

One may wonder whether it is possible to identify a path only by its signature. This turns out to be true under some conditions [7].

Theorem 5. *Let $X \in \mathcal{V}^1(J, \mathbb{R}^d)$ be a path with a fixed starting point and let at least one of its component functions be a monotone function. Then the signature $S(X)$ uniquely determines X.*

The expected signature also tells us something about the distribution of paths [8].

Theorem 6. *Let $X, Y \in \mathcal{V}^p(J, \mathbb{R}^d)$ for $p \geq 1$ be paths sampled from the distributions μ and ν, respectively. If $\mathbb{E}[S(X)] = \mathbb{E}[S(Y)] < \infty$ and $\mathbb{E}[S(X)]$ have an infinite convergence radius, then $\mu = \nu$.*

For Theorem 5 to apply, we need to assume that our paths X have finite 1-variation. This is the case in practice, since we work with discretized paths. Of course, we only ever obtain the finite dimensional truncated signature. However, in [7] the authors argue that it is the first signature dimensions that carry most of the information of the path. Thus, we will use the truncated signature and assume it suffices for our task.

4

Wasserstein Generative Adversarial Network

Our aim is to train neural stochastic differential equations (neural SDEs) to generate sample paths that resemble those drawn from the true conditional distribution of a system given noisy observations. A central challenge in this setting is how to compare two distributions over paths in a way that is both meaningful and tractable. The Wasserstein distance, which measures the minimal cost of transforming one distribution into another, offers a principled solution to this problem. Wasserstein Generative Adversarial Networks (WGANs) leverage this distance to train generative models more reliably than standard GANs, avoiding issues like vanishing gradients and mode collapse [9]. This chapter introduces the WGAN framework in its unconditional form and then extends it to the conditional setting, where the generated output is conditioned on observed data.

4.1 Unconditional WGAN

The 1-Wasserstein distance $W_1(\mu, \nu)$ between two probability measures μ and ν on a metric space (E, c) is defined as

$$W_1(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu,\nu)} \int_{E\times E} c(x, y)\, \mathrm{d}\gamma(x, y),$$

where $\Gamma(\mu, \nu)$ is the set of couplings, i.e., probability measures on $E \times E$ whose marginals are μ and ν. The Wasserstein metric represents the minimal cost of transporting mass between the two distributions, with $c(x, y)$ measuring the transportation cost between points x and y. The cost function c can be selected suitably based on the specific requirements of the problem.

Solving the primal form directly can be computationally challenging. To address this difficulty, the Kantorovich–Rubinstein duality provides an alternative formulation. Let $\mathrm{Lip}_1(E)$ denote the set of all 1-Lipschitz functions from E to $\mathbb{R}$, defined as

$$\mathrm{Lip}_1(E) = \Big\{\varphi : E \to \mathbb{R} \Big| |\varphi(x) - \varphi(y)| \leq c(x, y) \quad \text{for all } x, y \in E\Big\}.$$

The Kantorovich–Rubinstein duality shows that the 1-Wasserstein distance can be written in the form

$$W_1(\mu, \nu) = \sup_{\varphi \in \mathrm{Lip}_1(E)} \int_E \varphi(x)\, \mathrm{d}\mu(x) - \int_E \varphi(x)\, \mathrm{d}\nu(x).$$

Since μ and ν are probability measures, the dual form can be viewed as the difference of expectations

$$W_1(\mu, \nu) = \sup_{\varphi \in \mathrm{Lip}_1(E)} \mathbb{E}_{x\sim\mu}[\varphi(x)] - \mathbb{E}_{x\sim\nu}[\varphi(x)].$$

The dual form is more computationally tractable because it converts the problem into an optimization over functions rather than couplings.

The idea behind WGANs is to have two models: a generator that produces fake data and a critic φ that tries to distinguish between real and fake data. However, the critic is limited to the space of 1-Lipschitz functions on E with respect to the metric c. These models are trained iteratively *against* each other. Once the generator has generated data, the critic is trained to give large values to the real data and small values to the generated data. The generator is then trained in such a way that the data from the generator gets a maximized critic score. By this iterative training, as the generator gets better at mimicking the real data distribution, the critic gets better at distinguishing realistic fake data from real data. The ultimate goal is to obtain ν^*, a generated distribution with minimized 1-Wasserstein distance to the real data distribution μ. The objective is then formulated as

$$\nu^* = \arg\min_{\nu} W_1(\mu, \nu) = \arg\min_{\nu} \sup_{\varphi \in \mathrm{Lip}_1(E)} \mathbb{E}_{x\sim\mu}[\varphi(x)] - \mathbb{E}_{x\sim\nu}[\varphi(x)].$$

4.2 Conditional WGAN

Now, suppose that the set E consists of pairs $\mathbf{x} = (x, y)$ and that we are interested in minimizing the 1-Wasserstein distance between a real data distribution $\mu_{\mathbf{x}}$ and a generated distribution $\nu_{\mathbf{x}}$, i.e. we have the problem

$$\nu_{\mathbf{x}}^* = \arg\min_{\nu_{\mathbf{x}}} W_1(\mu_{\mathbf{x}}, \nu_{\mathbf{x}}) = \arg\min_{\nu_{\mathbf{x}}} \sup_{\varphi \in \mathrm{Lip}_1(E)} \mathbb{E}_{\mathbf{x}'\sim\mu_{\mathbf{x}}}[\varphi(\mathbf{x}')] - \mathbb{E}_{\mathbf{x}'\sim\nu_{\mathbf{x}}}[\varphi(\mathbf{x}')]. \quad (4.1)$$

The true joint distribution can be factorized as $\mu_{\mathbf{x}} = \mu_y \, \mu_{x|y}$. In order to respect the true marginal distribution of the conditioned variable y, we let $\nu_{\mathbf{x}} = \mu_y \, \nu_{x|y}$. Using the notation $\mathbf{x}' = (x', y')$ we can now rearrange the expectations in (4.1) to obtain

$$\arg\min_{\nu_{\mathbf{x}}} \sup_{\varphi \in \mathrm{Lip}_1(E)} \mathbb{E}_{y'\sim\mu_y}\Big[\mathbb{E}_{x'\sim\mu_{x|y'}}[\varphi(\mathbf{x}')] - \mathbb{E}_{x'\sim\nu_{x|y'}}[\varphi(\mathbf{x}')]\Big]. \quad (4.2)$$

The goal is now to generate a distribution $\nu_{x|y}$ such that the 1-Wasserstein distance between $\nu_{x|y}$ and the conditioned data distribution $\mu_{x|y}$ is minimized.

5

Neural SDEs for Smoothing

This chapter focuses on a practical model framework to solve the smoothing problem. We begin by describing the model setup and the data simulation process. Next, we describe the neural SDE training process using the WGAN framework. We then propose two critic architectures. A significant challenge we address is that minimization of the Wasserstein loss should lead to convergence of path distributions, which is greatly impacted by the selected critic architecture. Finally, we describe techniques for estimating the conditional expectations needed for the WGAN objective and summarize the derived algorithm configurations.

5.1 Modeling setup

To begin, we highlight the practical construction of our modeling framework, which includes parameterizing an SDE using neural networks, generating synthetic training data, and incorporating the neural SDE derived within the WGAN architecture.

5.1.1 Setup of the generator

As shown in Section 2.2, an SDE conditioned on observations $Z_{1:N}$ takes the form

$$\mathrm{d}X_t = \left[\mu(X_t, t) + \sigma(X_t, t)\,\sigma^\top(X_t, t)\nabla_{X_t} \log p(Z_{k+1:N} \mid X_t)\right]\,\mathrm{d}t + \sigma(X_t, t)\,\mathrm{d}W_t,$$

where $t \in [t_k, t_{k+1}]$ and $X_0 = x_0$. To model this, we define a neural SDE of the form

$$\mathrm{d}\tilde{X}_t = \tilde{\mu}(\tilde{X}_t, t, Z_{1:N})\,\mathrm{d}t + \tilde{\sigma}(\tilde{X}_t, t, Z_{1:N})\,\mathrm{d}W_t, \quad t \in [0, T], \quad X_0 = x_0,$$

where position, time, and observations serve as inputs. The functions $\tilde{\mu}$ and $\tilde{\sigma}$ are neural networks that represent the drift and diffusion terms, respectively, conditioned on observations $Z_{1:N}$. Note that one does not need the dependency on observations $Z_{1:N}$ for the diffusion network $\tilde{\sigma}$, as the true diffusion is independent of them. The parameters of both networks are collected in the parameter vector γ. Both $\tilde{\mu}$ and $\tilde{\sigma}$ have five fully connected hidden layers of width 128. We show a more detailed description of our networks in Appendix F. Since we use a WGAN, we sometimes refer to the neural SDE as the generator.

5.1.2 Generating data

To train our neural SDE model, we require a synthetic data set consisting of sample paths from an SDE and corresponding noisy observations. To achieve this, we

discretize the time interval $[0, T]$ into H steps of size $\Delta t = T/H$. Let $t_j = j\Delta t$, $j = 0, 1, \ldots, H$ be the discrete-time instances. The data consists of a path vector $(X_{t_j})_{j=0}^H$ of the position of the trajectory at time t_j, $j = 0, 1, \ldots, H$, sampled from an SDE

$$\mathrm{d}X_t = \mu(X_t, t)\,\mathrm{d}t + \sigma(X_t, t)\,\mathrm{d}W_t, \quad t \in [0, T], \quad X_0 = x_0.$$

using the Euler–Maruyama method. We also include the time vector $(t_j)_{j=0}^H$. Let $\mathcal{P}_t$ be a partition of $[0, t]$. We augment the data with quadratic variation, defined as

$$[X]_t := \lim_{\|\mathcal{P}_t\| \to 0} \sum_l (X_{t_l} - X_{t_{l-1}})^2.$$

Since $\mathrm{d}W_t^2 = \mathrm{d}t$ and $\mathrm{d}t^2$ gets negligible in the limit, our time discretization of the quadratic variation is simply

$$[X]_{t_j} = \sum_{l=0}^{j} \sigma^2(X_{t_l}, t_l)\,\Delta W_t^2.$$

Augmenting the data with quadratic variation has been shown to help with finding the minimum of the loss function [10]. In total, we have our augmented data $\mathbf{X} = (X_{t_j}, t_j, [X]_{t_j})_{j=0}^H$. Once a path $(X_{t_j})_{j=0}^H$ is sampled, observations

$$Z_k = X_{t_k} + \xi_k, \quad k = 1, \ldots, N,$$

with observation noise

$$\xi_k \sim \mathcal{N}(0, \sigma_Z^2),$$

are realized with some constant measurement noise σ_Z^2 at predetermined time instances t_k, $k = 1, \ldots, N$, giving us the observation vector $Z_{1:N}$.

5.1.3 Implementation of WGAN

Our goal is to adapt the 1-Wasserstein distance (4.2) to our setting involving path distributions conditioned on observations. Recall that $\mathcal{V}^1([0, T], \mathbb{R}^d)$ is the set of paths $X : [0, T] \to \mathbb{R}^d$ with finite 1-variation. Let $\mathcal{X} \subseteq \mathcal{V}^1([0, T], \mathbb{R}^d)$ and $\mathcal{Z} \subseteq \mathbb{R}^{d \times N}$ be the state and the observation space, respectively, in which case (4.2) translates to

$$\begin{aligned} &\arg\min_{\nu_{X,Z_{1:N}}} W_1(\mu_{X,Z_{1:N}}, \nu_{X,Z_{1:N}}) \qquad (5.1) \\ &= \arg\min_{\nu_{X,Z_{1:N}}} \sup_{\varphi \in \mathrm{Lip}_1(\mathcal{X} \times \mathcal{Z})} \mathbb{E}_{Z_{1:N} \sim \mu_{Z_{1:N}}} \Big[\mathbb{E}_{\hat{X} \sim \mu_{X|Z_{1:N}}} [\varphi(\hat{X}, Z_{1:N})] - \mathbb{E}_{\tilde{X} \sim \nu_{X|Z_{1:N}}} [\varphi(\tilde{X}, Z_{1:N})] \Big], \end{aligned}$$

where the conditional distribution $\nu_{X|Z_{1:N}}$ is the distribution of the paths induced by the generator G and $\mu_{Z_{1:N}}$ and $\mu_{X|Z_{1:N}}$ are the data distributions of observations and paths conditioned on observations, respectively. Figure 5.1 illustrates the scheme of the WGAN. In Section 5.3 we show how to obtain a sample $\hat{X}$ from the true conditional distribution given the unconditional SDE and a set of observations $Z_{1:N}$. Since the generator only affects the second term — the expectation over its own samples — its parameters are optimized to minimize this component of the loss,

$$\mathcal{L}_G(\gamma) = -\mathbb{E}_{Z_{1:N} \sim \mu_{Z_{1:N}}} \Big[\mathbb{E}_{\tilde{X} \sim \nu_{X|Z_{1:N}}} [\varphi(\tilde{X}, Z_{1:N})] \Big].$$

Ultimately, we wish to obtain a generator such that we minimize the 1-Wasserstein distance between the data distribution and the distribution induced by the generator.

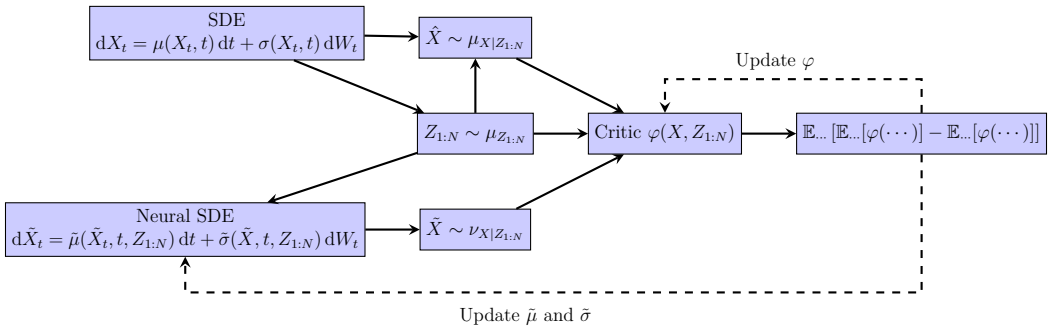


Figure 5.1: WGAN scheme

5.2 Critic approaches

Given that our data is composed of paths, we must consider how these paths should be represented to the critic. We present two different methodologies for building the critic φ: the first acts within signature space, while the second works directly within path space. The selection of the critic architecture substantially affects the loss landscape, thus impacting the difficulty level of training neural SDEs. Notably, obtaining a Wasserstein distance of zero between the generated and genuine data distributions guarantees identical path distributions, regardless of which critic design is utilized. However, simply achieving a low Wasserstein loss does not necessarily mean that the actual path distributions are closely matched. This discrepancy represents our main challenge: identifying an appropriate loss function that guarantees the convergence of distributions in path space as the Wasserstein distance nears zero.

5.2.1 Signature critic

We define the first critic as the composition

$$\varphi_S(X, Z_{1:N}) = \Phi_S(S^m(\mathbf{X}), Z_{1:N}), \tag{5.2}$$

where S^m is the truncated signature transform of order $m = 4$, as introduced in Chapter 3, and Φ_S is a neural network with problem dependent architecture, described in detail in Appendix F. The truncated signature offers a structured representation of the path by capturing essential features and their interactions. This can simplify the optimization landscape, as it is often easier to learn a function of the signature than of the original path [11].

In Figure 5.2, we visually illustrate how the signature critic works. In this illustration, we have trained the critic to separate the analytical paths conditioned on an observation $Z = 2$ from those of an untrained generator. The underlying SDE is linear, so Doob's h-transform can be derived analytically. We begin by plotting the paths from the generator and the analytical SDE. Then, we transform each path into signature space and plot the distribution over the first two dimensions. Note how well the signature transform separates the analytical paths from the generated ones. The last step is to apply the critic that has been trained to separate the distributions.

This yields two distributions of the critic score φ_S. The objective of the critic is to separate the mean of these two distributions while the generator tries to push up its mean to close the gap.

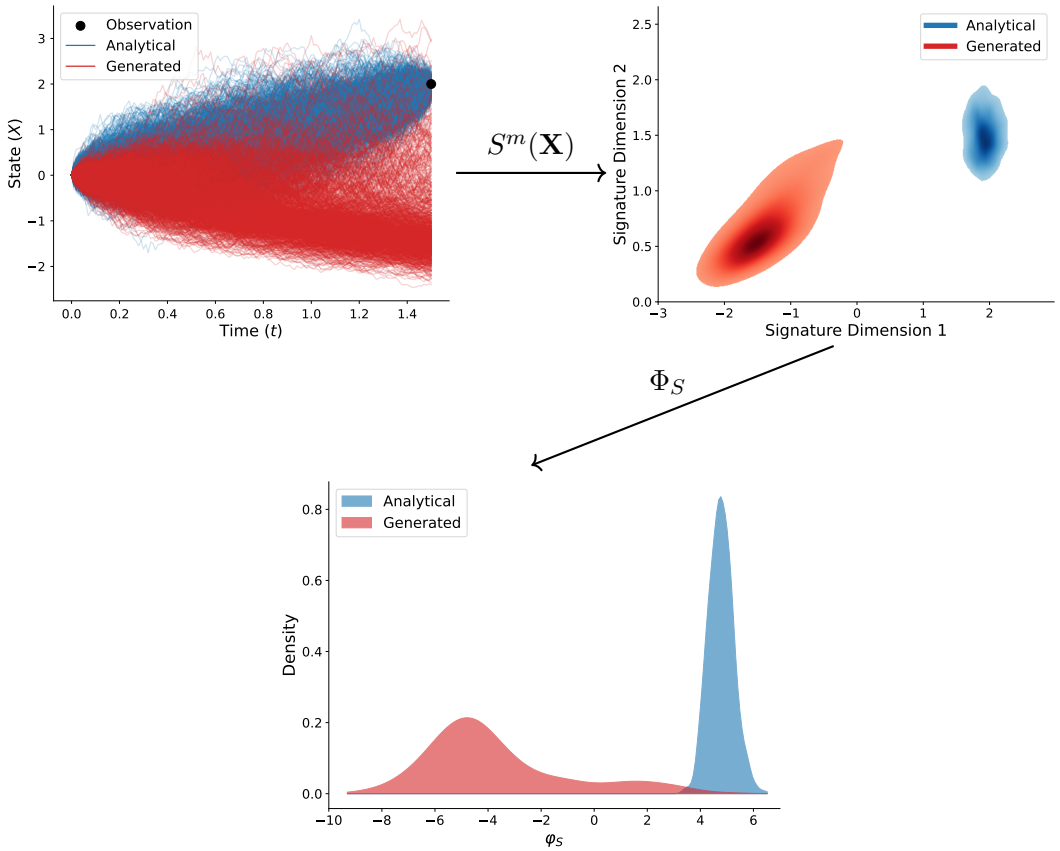


Figure 5.2: Overview of the signature transformation and critic evaluation.

5.2.2 Path critic

For the second critic, we define

$$\varphi_P(X, Z_{1:N}) = \int_0^T \Phi_P(X_t, t, [X]_t, Z_{1:N})\,\mathrm{d}t, \tag{5.3}$$

where Φ_P is also problem dependent, described in detail in Appendix F. Intuitively, Φ_P assigns a score to each point along the trajectory X_t, conditioned on observations $Z_{1:N}$, and the critic φ_P accumulates these scores throughout the path. In contrast to the signature-based critic, this direct path-based approach makes fewer structural assumptions about the data, preserving all raw information. Although this flexibility can be beneficial for capturing intricate path dynamics, it may lead to a more complex optimization landscape, potentially impacting training stability and convergence speed.

In Figure 5.3, we visualize how the path critic works. We have the same paths from the analytically derived Doob's h-transform and untrained generator as in the

signature critic example. The paths have again been generated conditioned on the observation $Z = 2$. However, this time the neural network Φ_P outputs a number for each point in the state space. We see that for any given point, the output Φ_P increases with the number of analytical paths that pass through that point. In contrast, the points through which the generated paths pass have small outputs. Then the critic value of a specific path X is determined by integrating Φ_P along the path X over t.

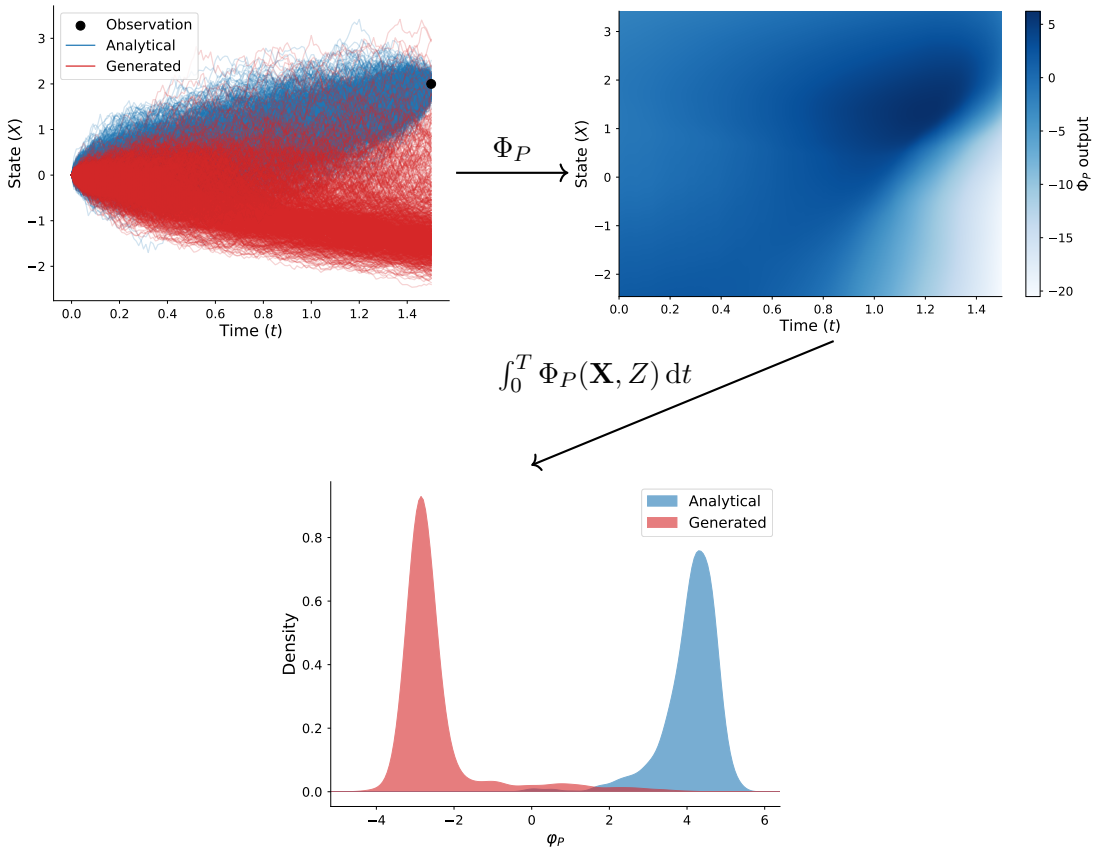


Figure 5.3: Overview of the path transformation and critic evaluation.

5.2.3 Lipschitz constraint

Remember from Section 4.1 that the critic φ must be 1-Lipschitz with respect to a chosen metric c. Crucially, the choice of metric directly constrains the class of admissible functions φ, making it an essential modeling decision. To reflect the structure of our path data, we define the metric as

$$c((X, Z_{1:N}), (Y, Z_{1:N})) = \|X - Y\|_\infty = \sup_{t\in[0,T]} \|X_t - Y_t\|_2.$$

Hence, $\varphi \in \mathrm{Lip}_1(\mathcal{X} \times \mathcal{Z})$ if

$$|\varphi(X, Z_{1:N}) - \varphi(Y, Z_{1:N})| \leq \sup_{t\in[0,T]} \|X_t - Y_t\|_2, \quad X, Y \in \mathcal{X}, \quad Z_{1:N} \in \mathcal{Z}.$$

This means that if two paths with the same observation are separated by a maximal distance M, the difference in the output of the critic between the paths is at most M. Neural networks are not guaranteed to be 1-Lipschitz so we introduce the following Lipschitz penalty term to enforce this,

$$\lambda_{\mathrm{P}} = \lambda \left[\mathrm{ReLU}\left(\frac{|\varphi(X, Z_{1:N}) - \varphi(Y, Z_{1:N})|}{\|X - Y\|_\infty} - 1 \right)^2 \right]$$

where $\lambda > 0$ is a penalty parameter. With this penalty, $\lambda_{\mathrm{P}} = 0$ iff $\varphi \in \mathrm{Lip}_1(\mathcal{X} \times \mathcal{Z})$.

5.3 Estimating the true conditional expectation

With the min-max problem at hand, another challenge emerges. How do we obtain the conditional expectation $\mathbb{E}_{\hat{X} \sim \mu_{X|Z_{1:N}}}[\varphi(\hat{X}, Z_{1:N})]$ of the true distribution in (5.1)? Sampling from this distribution is generally computationally burdensome. We will cover a method to approximate this expectation and one way to eliminate this problem entirely.

The first method is the Metropolis–Hastings (MH) algorithm which is described in Appendix A. As discussed there, MH scales poorly with increasing dimensionality and decreasing observation noise. Nevertheless, in lower-dimensional settings, it remains a valuable tool for evaluating the performance of our critic architectures.

For the second method, we simply rearrange the expectations in (5.1) to obtain

$$\begin{aligned} &\arg\min_{\nu_{X,Z_{1:N}}} W_1(\mu_{X,Z_{1:N}}, \nu_{X,Z_{1:N}}) \\ =&\arg\min_{\nu_{X,Z_{1:N}}} \sup_{\varphi \in \mathrm{Lip}_1(\mathcal{X} \times \mathcal{Z})} \mathbb{E}_{\hat{X}, Z_{1:N} \sim \mu_{X,Z_{1:N}}}[\varphi(\hat{X}, Z_{1:N})] - \mathbb{E}_{\tilde{X}, Z_{1:N} \sim \nu_{X,Z_{1:N}}}[\varphi(\tilde{X}, Z_{1:N})]. \end{aligned}$$

This reformulation shifts the focus from explicitly computing conditional expectations to working with the joint distribution of paths and observations. By doing so, we bypass the need to sample from the true conditional distribution directly. Figure 5.4 illustrates the WGAN scheme using the joint distribution.

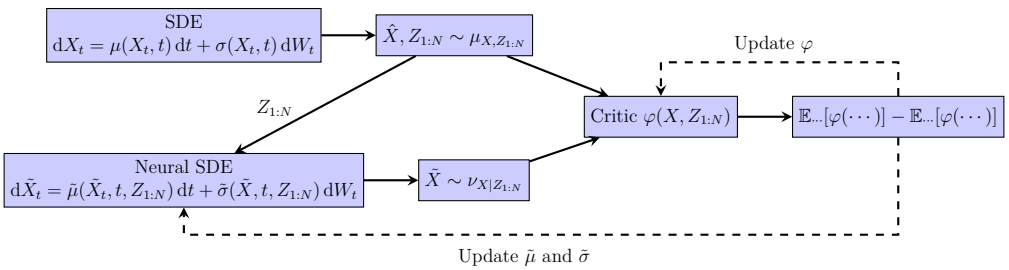


Figure 5.4: WGAN scheme with joint distribution.

5.4 Summary of algorithms

Finally, we summarize the 4 different algorithms resulting from various combinations of critic architectures and true distribution estimation methods. Table 5.1 lists the combinations and their corresponding short-hand algorithm names used throughout the rest of the thesis.

Critic Architecture	Expectation Estimation	Short Name
Signature	Metropolis–Hastings	SMH
Signature	Joint Distribution	SJ
Path	Metropolis–Hastings	PMH
Path	Joint Distribution	PJ

Table 5.1: Algorithm configurations by critic architecture and expectation estimation method.

5.4.1 Signature critic with Metropolis–Hastings

The first algorithm, which we will refer to as SMH, combines the signature critic φ_S given by (5.2) and the MH algorithm explained in Appendix A. In this setting, the parameters ϕ of the critic are updated by minimizing

$$\mathcal{L}_{\varphi_S}(\phi) = -\mathbb{E}_{Z_{1:N}\sim\mu_{Z_{1:N}}}\Big[\mathbb{E}_{\hat{X}\sim\mu_{X|Z_{1:N}}}[\varphi_S(\hat{X}, Z_{1:N})] - \mathbb{E}_{\tilde{X}\sim\nu_{X|Z_{1:N}}}[\varphi_S(\tilde{X}, Z_{1:N})]\Big] + \lambda_{\mathrm{P}},$$

where

$$\lambda_{\mathrm{P}} = \lambda\,\mathbb{E}_{Z_{1:N}\sim\mu_{Z_{1:N}},\hat{X}\sim\mu_{X|Z_{1:N}},\tilde{X}\sim\nu_{X|Z_{1:N}}}\left[\mathrm{ReLU}\left(\frac{|\varphi_S(\hat{X}, Z_{1:N}) - \varphi_S(\tilde{X}, Z_{1:N})|}{\|\hat{X} - \tilde{X}\|_\infty} - 1\right)^2\right].$$

Once samples $(Z^k_{1:N})_{k=1}^K$ with paths $(X^{k,j}_\mu)_{j=1}^J$ and $(X^{k,j}_\nu)_{j=1}^J$ have been obtained from $\mu_{X|Z^k_{1:N}}$ and $\nu_{X|Z^k_{1:N}}$ respectively, we calculate the loss using the MC estimates

$$\hat{\mathcal{L}}_{\varphi_S}(\phi) = K^{-1}J^{-1}\sum_{k=1}^K\sum_{j=1}^J -\varphi_S(X^{k,j}_\mu, Z^k_{1:N}) + \varphi_S(X^{k,j}_\nu, Z^k_{1:N}) + \hat{\lambda}_{\mathrm{P}},$$

$$\hat{\lambda}_{\mathrm{P}} = K^{-1}J^{-1}\sum_{k=1}^K\sum_{j=1}^J \mathrm{ReLU}\left(\frac{|\varphi_S(X^{k,j}_\mu, Z^k_{1:N}) - \varphi_S(X^{k,j}_\nu, Z^k_{1:N})|}{\|X^{k,j}_\mu - X^{k,j}_\nu\|_\infty} - 1\right)^2.$$

5.4.2 Signature critic with joint distribution

The second algorithm, named SJ, integrates the signature critic φ_S from (5.2) with the joint distribution introduced in Section 5.3. In this setting, the parameters ϕ of the critic are updated by minimizing

$$\mathcal{L}_{\varphi_S}(\phi) = -\mathbb{E}_{\hat{X},Z_{1:N}\sim\mu_{X,Z_{1:N}}}[\varphi_S(\hat{X}, Z_{1:N})] + \mathbb{E}_{\tilde{X},Z_{1:N}\sim\nu_{X,Z_{1:N}}}[\varphi_S(\tilde{X}, Z_{1:N})] + \lambda_{\mathrm{P}}$$

with

$$\lambda_{\mathrm{P}} = \mathbb{E}_{\hat{X}, Z_{1:N} \sim \mu_{X, Z_{1:N}}, \tilde{X} \sim \nu_{X|Z_{1:N}}} \left[\mathrm{ReLU}\left(\frac{|\varphi_S(\hat{X}, Z_{1:N}) - \varphi_S(\tilde{X}, Z_{1:N})|}{\|\hat{X} - \tilde{X}\|_\infty} - 1 \right)^2 \right],$$

which we approximate with the MC estimates

$$\hat{\mathcal{L}}_{\varphi_S}(\phi) = K^{-1} \sum_{k=1}^{K} -\varphi_S(X_\mu^k, Z_{1:N}^k) + \varphi_S(X_\nu^k, Z_{1:N}^k) + \hat{\lambda}_{\mathrm{P}},$$
$$\hat{\lambda}_{\mathrm{P}} = K^{-1} \sum_{k=1}^{K} \mathrm{ReLU}\left(\frac{|\varphi_S(X_\mu^k, Z_{1:N}^k) - \varphi_S(X_\nu^k, Z_{1:N}^k)|}{\|X_\mu^k - X_\nu^k\|_\infty} - 1 \right)^2.$$

5.4.3 Path critic with Metropolis–Hastings

The third algorithm, labeled PMH, combines the path critic φ_P from (5.3) with the MH algorithm outlined in Section 5.3. In this setting, similarly to using a signature critic, the parameters ϕ of the critic are updated by minimizing

$$\mathcal{L}_{\varphi_P}(\phi) = -\mathbb{E}_{Z_{1:N} \sim \mu_{Z_{1:N}}} \left[\mathbb{E}_{\hat{X} \sim \mu_{X|Z_{1:N}}} [\varphi_P(\hat{X}, Z_{1:N})] - \mathbb{E}_{\tilde{X} \sim \nu_{X|Z_{1:N}}} [\varphi_P(\tilde{X}, Z_{1:N})] \right] + \lambda_{\mathrm{P}},$$

where

$$\lambda_{\mathrm{P}} = \lambda\, \mathbb{E}_{Z_{1:N} \sim \mu_{Z_{1:N}}, \hat{X} \sim \mu_{X|Z_{1:N}}, \tilde{X} \sim \nu_{X|Z_{1:N}}} \left[\mathrm{ReLU}\left(\frac{|\varphi_P(\hat{X}, Z_{1:N}) - \varphi_P(\tilde{X}, Z_{1:N})|}{\|\hat{X} - \tilde{X}\|_\infty} - 1 \right)^2 \right].$$

Once samples $(Z_{1:N}^k)_{k=1}^K$ with paths $(X_\mu^{k,j})_{j=1}^J$ and $(X_\nu^{k,j})_{j=1}^J$ has been obtained from $\mu_{X|Z_{1:N}^k}$ and $\nu_{X|Z_{1:N}^k}$ respectively, we calculate the loss using the MC estimates

$$\hat{\mathcal{L}}_{\varphi_P}(\phi) = K^{-1} J^{-1} \sum_{k=1}^{K} \sum_{j=1}^{J} -\varphi_P(X_\mu^{k,j}, Z_{1:N}^k) + \varphi_P(X_\nu^{k,j}, Z_{1:N}^k) + \hat{\lambda}_{\mathrm{P}},$$
$$\hat{\lambda}_{\mathrm{P}} = K^{-1} J^{-1} \sum_{k=1}^{K} \sum_{j=1}^{J} \mathrm{ReLU}\left(\frac{|\varphi_P(X_\mu^{k,j}, Z_{1:N}^k) - \varphi_P(X_\nu^{k,j}, Z_{1:N}^k)|}{\|X_\mu^{k,j} - X_\nu^{k,j}\|_\infty} - 1 \right)^2.$$

5.4.4 Path critic with joint distribution

The fourth and last algorithm, labeled PJ, uses the path critic φ_P (5.3) with the joint distribution presented in Section 5.3. In this setting, similarly to using a signature critic, the parameters ϕ of the critic are updated by minimizing

$$\mathcal{L}_{\varphi_P}(\phi) = -\mathbb{E}_{\hat{X}, Z_{1:N} \sim \mu_{X, Z_{1:N}}} [\varphi_P(\hat{X}, Z_{1:N})] + \mathbb{E}_{\tilde{X}, Z_{1:N} \sim \nu_{X, Z_{1:N}}} [\varphi_P(\tilde{X}, Z_{1:N})] + \lambda_{\mathrm{P}}$$

with

$$\lambda_{\mathrm{P}} = \mathbb{E}_{\hat{X}, Z_{1:N} \sim \mu_{X, Z_{1:N}}, \tilde{X} \sim \nu_{X|Z_{1:N}}} \left[\mathrm{ReLU}\left(\frac{|\varphi_P(\hat{X}, Z_{1:N}) - \varphi_P(\tilde{X}, Z_{1:N})|}{\|\hat{X} - \tilde{X}\|_\infty} - 1 \right)^2 \right],$$

which we approximate with the MC estimates

$$\hat{\mathcal{L}}_{\varphi_P}(\phi) = K^{-1} \sum_{k=1}^{K} -\varphi_P(X_\mu^k, Z_{1:N}^k) + \varphi_P(X_\nu^k, Z_{1:N}^k) + \hat{\lambda}_{\mathrm{P}},$$
$$\hat{\lambda}_{\mathrm{P}} = K^{-1} \sum_{k=1}^{K} \mathrm{ReLU}\left(\frac{|\varphi_P(X_\mu^k, Z_{1:N}^k) - \varphi_P(X_\nu^k, Z_{1:N}^k)|}{\|X_\mu^k - X_\nu^k\|_\infty} - 1 \right)^2.$$

The performance of these algorithm configurations is assessed in Chapter 7 through a series of benchmark problems designed to evaluate their accuracy in learning conditional path distributions.

6
Neural SDEs for Filtering

In this chapter, we apply neural SDEs to the filtering problem by modifying the parameterization to isolate the control term. This allows for neural SDEs to be used as proposals in particle filters where we can compare the performance to other strong methods within this setting. We end the chapter by briefly explaining how the controlled SDE is integrated into the particle filter framework.

6.1 Modification of generator

Recall from Section 2.2.3 that the conditioned process can be written as

$$\mathrm{d}X_t = [\mu(X_t) + \sigma(X_t)u(X_t, t, Z)]\ \mathrm{d}t + \sigma(X_t)\,\mathrm{d}W_t, \quad X_0 = x_0, \quad t \in [0, T],$$

where we have constrained the problem to one observation Z at final time T. Instead of parameterizing the whole drift term, we now only parameterize the control term $u(X_t, Z) = \sigma^\top(X_t, t)\nabla \log h(X_t, t, Z)$ yielding

$$\mathrm{d}\tilde{X}_t = \left[\mu(\tilde{X}_t) + \sigma(\tilde{X}_t)\tilde{u}(\tilde{X}_t, t, Z)\right]\ \mathrm{d}t + \sigma(\tilde{X}_t)\,\mathrm{d}W_t, \quad X_0 = x_0, \quad t \in [0, T], \quad (6.1)$$

where $\tilde{u}$ is a neural network.

6.2 Learning the optimal control

We begin by noting that the WGAN is still trained using the procedure described in Section 5, now applied to a modified parameterization of the neural SDE where only the control term is learned. This reparameterization also enables comparison with alternative approaches, such as the computational Doob's h-transform (CDT) algorithm, which learns a control function based on the optimal control framework described in Section 2.2.3. We will first describe how the CDT algorithm is implemented, before reminding the reader how the resulting controlled SDE is incorporated into the particle filter framework.

6.2.1 CDT algorithm

For the CDT algorithm, proposed in [2], we consider the controlled process $X^{u,Z}$, driven by the standard Brownian motion W, as introduced in Section 2.2.3. Additionally, for two functions $G : \mathbb{R}^d \times [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ and $G_0 : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ consider

the value process V defined as

$$V_s = V_0 + \int_0^s \frac{1}{2}\|Y_t\|^2 + \langle u, Y_t\rangle \, \mathrm{d}t + \int_0^s \langle Y_t, \mathrm{d}W_t\rangle \,,$$

where $V_0 = G_0(X_0, Z)$ and $Y_t = G(X_t, t, Z)$. Since we are assuming that X_0 is fixed, G_0 is a function of only Z. Importantly, we remind the reader that $X^{u,Z}$ and V are driven by the same Brownian motion W. The uniqueness result in Section 2.2.3 implies that, if the condition $V_T = -\log[p(Z \mid X_T^{u,Z})]$ is satisfied for any choice of terminal observation $Z \in \mathcal{Z}$, then the following holds for all (X_t, t, Z),

$$\begin{aligned} G_0(X_0, Z) &= -\log[p(Z \mid X_0)], \\ G(X_t, t, Z) &= -\sigma^\top \nabla \log[p(Z \mid X_t, t)]. \end{aligned}$$

In particular, the optimal control is given by $u_\star(X_t, t, Z) = -G(X_t, t, Z)$. These remarks suggest parameterizing the functions G_0 and G with two neural networks, with parameters $\theta_0 \in \Theta_0$ and $\theta \in \Theta$ respectively. The goal is to minimize the following loss function, that was first proposed in [12],

$$\mathcal{L}(\theta_0, \theta; u) = \mathbb{E}\left[\left(V_T + \log\left[p(Z \mid X_T^{u,Z})\right]\right)^2\right]. \tag{6.2}$$

The expectation is taken with respect to the Brownian motion W and observation $Z \sim \mu_Z$ sampled at time T. In equation (6.2), we fix the dynamics of $X_t^{u,Z}$ and optimize over the dynamics of V. Note that any choice of control u can be used. However using an iterative scheme where we set the control as the current approximation of $-G(X_t, t, Z)$ has been shown by the authors of [2] to yield better performance.

6.2.2 Integration in particle filter

Having trained an optimal controller—using either the WGAN or the CDT algorithm—we now implement it within a particle filter. We will sample controlled trajectories from (6.1) and then weigh them with Girsanov's theorem (2.9),

$$\pi_T^i = \exp\left\{-\frac{1}{2}\int_0^T \left\|\tilde{u}(\tilde{X}_t^i, t, Z_k)\right\|^2 \mathrm{d}t - \int_0^T \left\langle \tilde{u}(\tilde{X}_t^i, t, Z_k), \mathrm{d}W_t^i \right\rangle\right\} p(Z_k | X_{t_k}^i),$$

yielding an estimation of the posterior distribution $p(X_{0:T}|Z)$. We restrict our investigation to a single filtering step of the particle filter, under the assumption of a known initial state X_0. This simplifies the setup and allows us to focus on the comparison between different proposal strategies. In a multi-step setting, the filtered state after each step becomes a distribution, not a fixed point. Consequently, to train the model over multiple steps, one would need to sample initial states from a distribution rather than assume a known value. There is nothing inherent in our algorithms that prevents this extension. However, for the purpose of this thesis, a single filtering step with a known initial condition is sufficient to evaluate and compare the different methods under consideration.

7

Results for the Smoothing Problem

In this chapter we present the results for the smoothing problem. We remind the reader that these results are associated with the framework discussed in Section 5. We start with the most basic case of a single observation, first in one spatial dimension and then moving up to two dimensions. We then tackle the more challenging smoothing problem for five observations in one dimension.

7.1 One observation in one spatial dimension

We will examine two test dynamics: one that features bimodal behavior and another that is time-dependent with multiple curves along its path. These dynamics are not derived from a real-world scenario, but they still provide good test cases to assess the performance of our algorithms. We will compare the result with that of a BPF. In both test cases, the BPF is a good enough approximation to represent the true solution to the smoothing problem.

7.1.1 Bimodal dynamics

To investigate whether our neural SDEs can effectively capture dynamics that yield a bimodal distribution, we analyze the non-linear SDE

$$\begin{cases} \mathrm{d}X_t = (-4X_t^3 + 6X_t)\,\mathrm{d}t + 0.5\,\mathrm{d}W_t, \quad t \in [0, 1.5], \\ X_0 = 0, \\ Z_1 = X_{1.5} + \mathcal{N}(0, \sigma_Z^2). \end{cases} \tag{7.1}$$

This SDE exhibits a bimodal distribution, as depicted in Figure 7.1. Note that solving the corresponding ordinary differential equation without the diffusion term results in two stable equilibria at $\pm\sqrt{\frac{3}{2}}$ and an unstable one at 0. A single measurement, Z_1, is observed at $t = 1.5$, with measurement noise set to $\sigma_Z = 2$. A higher noise level makes it more ambiguous which of the two modes the measurement comes from. Consequently, the conditional distribution still exhibits two modes, but they are weighted differently.

In Figure 7.2 we present the path distributions for our models and the BPF, conditioned on the observation $Z_1 = -0.8$, which highlights the spread and structure of the conditioned trajectories. Furthermore, we plot the marginal distribution of our models at the final time step $t = 1.5$ for the same observation in Figure 7.3. Finally,

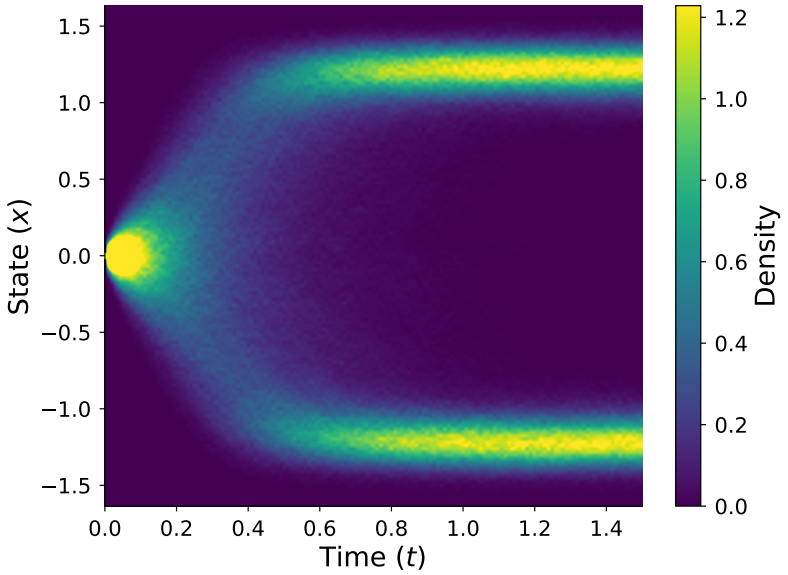
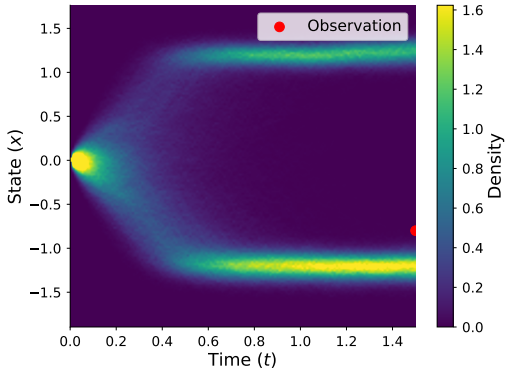


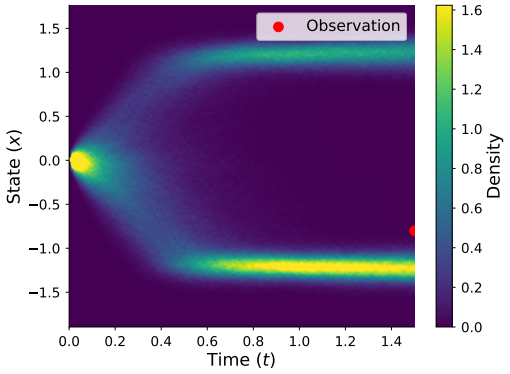
Figure 7.1: Path distribution of the SDE in (7.1).

Figure 7.4 provides a quantitative evaluation of each algorithm by comparing its marginal distributions at each time step to those of the BPF, using the Wasserstein distance. These distances are averaged at each time step over a set of observations Z_1 sampled from the distribution μ_z.

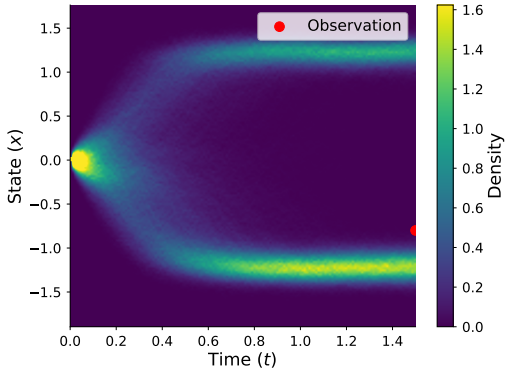
There are several noteworthy observations to make from these results. First of all, we see that we do not lose any notable performance by using a joint distribution compared to using Metropolis–Hastings. That is greatly appreciated since employing the Metropolis–Hastings algorithm becomes impractical computationally as the problem's dimensionality increases, an issue that using the joint distribution does not suffer from. Furthermore, we see that the performance between the signature and path-based algorithms is similar with slightly better performance for the path-based algorithms. In general, the four algorithms are able to closely mimic the true conditional distribution.



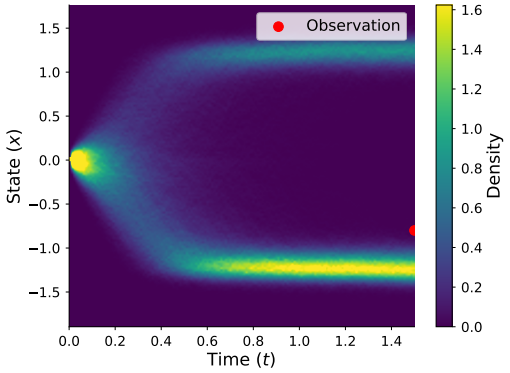
(a) SMH



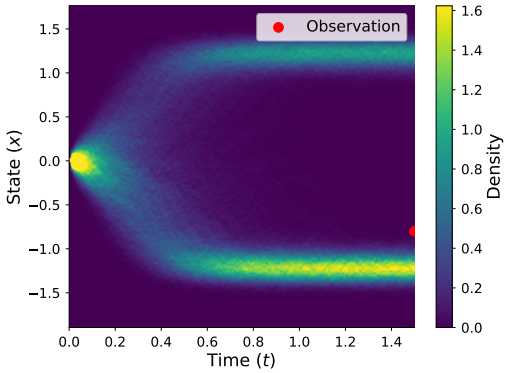
(b) SJ



(c) PMH

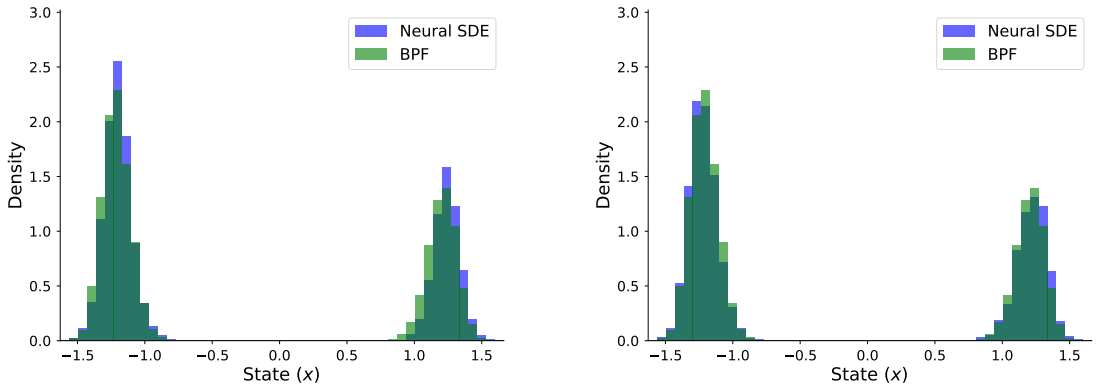


(d) PJ

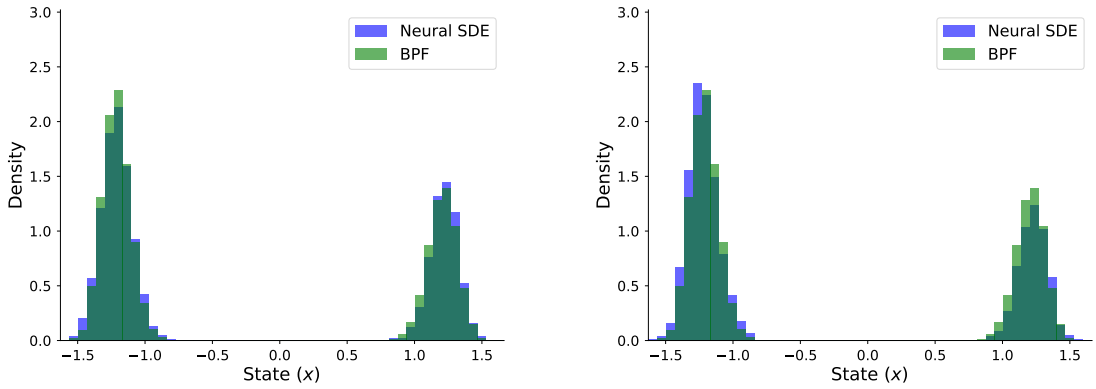


(e) BPF

Figure 7.2: Path distribution of the neural SDEs for an observation $Z_1 = -0.8$. The SDE is given by (7.1).



(a) SMH **(b)** SJ



(c) PMH **(d)** PJ

Figure 7.3: Marginal distribution of the neural SDEs and BPF at final time $T = 1.5$ for observation $Z_1 = -0.8$. The SDE is given by (7.1).

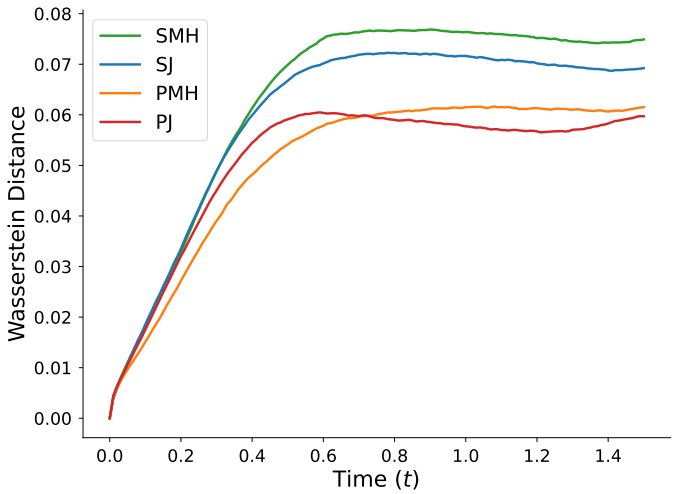


Figure 7.4: Average Wasserstein distance between the marginal distributions at each time step for the neural SDEs compared to BPF. The SDE is given by (7.1).

7.1.2 Time-dependent dynamics

In this section, we analyze the time-dependent SDE

$$\begin{cases} \mathrm{d}X_t = \left(1 + 5\sin(2\pi t) - \dfrac{X_t}{2-t}\right)\mathrm{d}t + \mathrm{d}W_t, \quad t \in [0, 1.5], \\ X_0 = 10, \\ Z_1 = X_{1.5} + \mathcal{N}(0, \sigma_Z^2), \end{cases} \tag{7.2}$$

where the measurement noise is set to $\sigma_Z = 0.2$. Due to the time-dependent drift term combined with the sinusoidal function, the system exhibits complex behavior with numerous turns, as shown in Figure 7.5. Furthermore, a lower observation noise means that the distribution will squeeze in towards the observation.

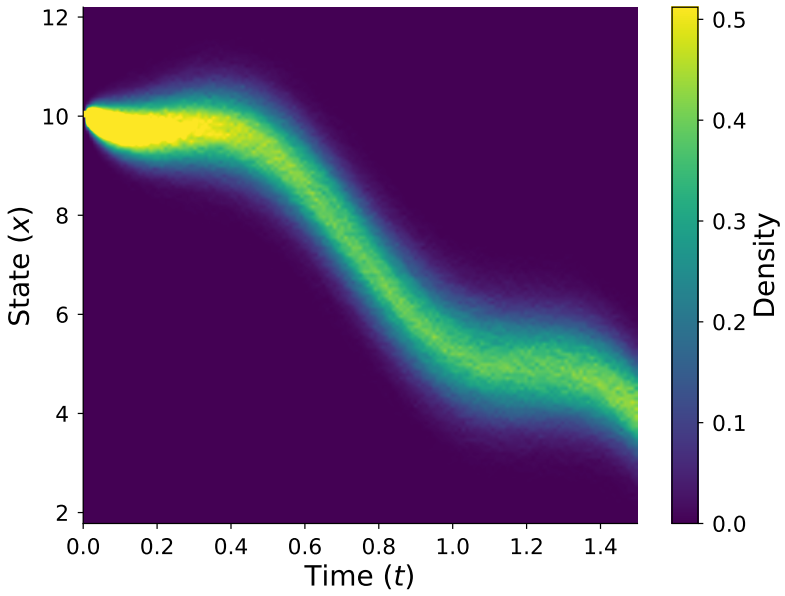
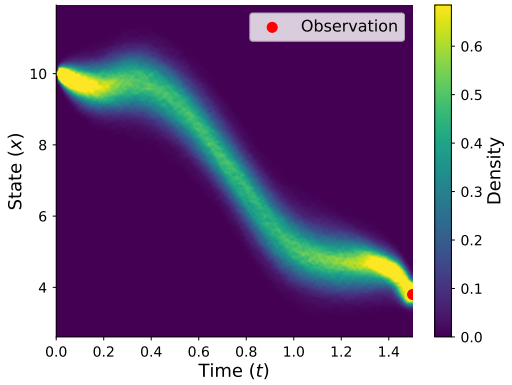


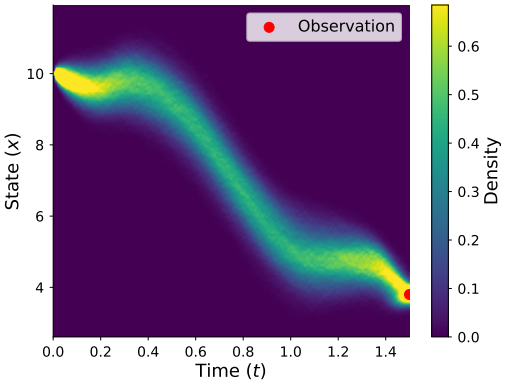
Figure 7.5: Path distribution of the SDE in (7.2).

In Figure 7.6 we present the path distribution for an observation $Z_1 = 3.8$. Furthermore, we plot the marginal distribution at the final time step $t = 1.5$ for the same observation in Figure 7.7. Finally, in Figure 7.8 we quantitatively compare the methods with the BPF by comparing the Wasserstein distance between the distributions in each time instance.

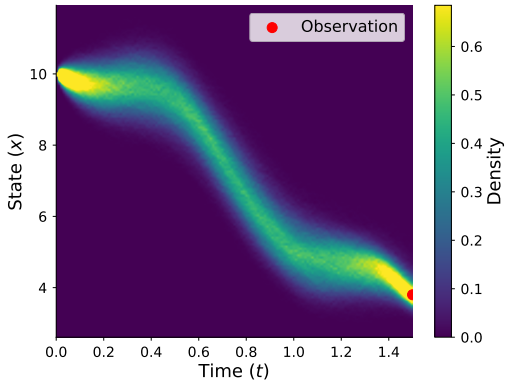
We see that the path-based algorithms perform very well, whilst the signature-based algorithms struggle a little bit with the shape in the end. Once again we do not see any notable edge in performance using Metropolis–Hastings compared to the joint expectation, so we will only consider the SJ and PJ frameworks from now on.



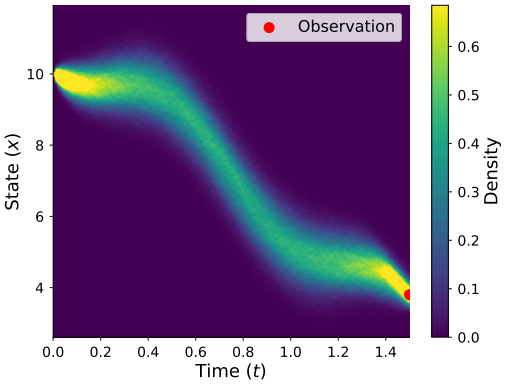
(a) SMH



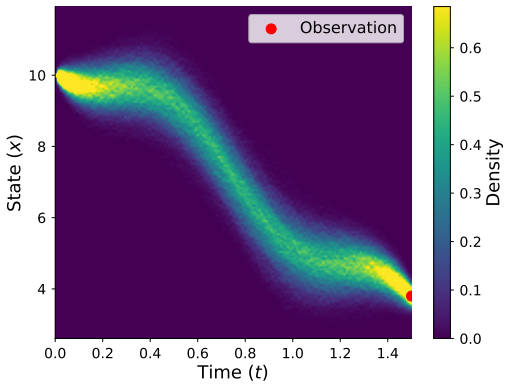
(b) SJ



(c) PMH

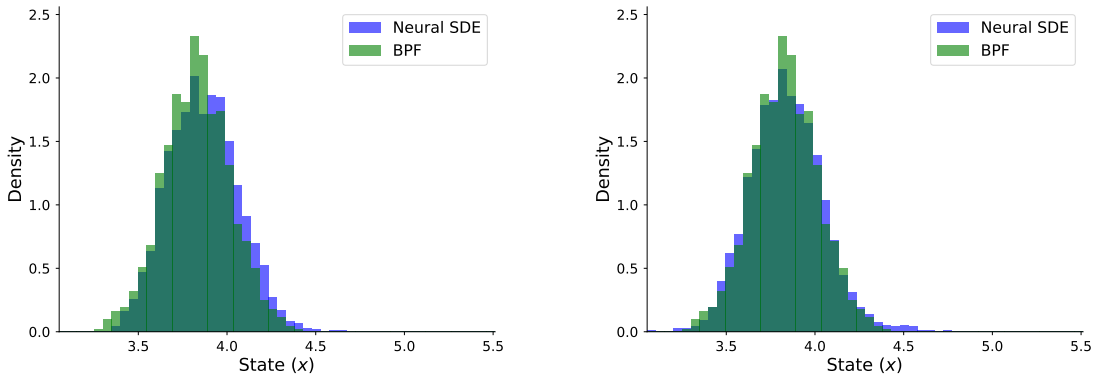


(d) PJ

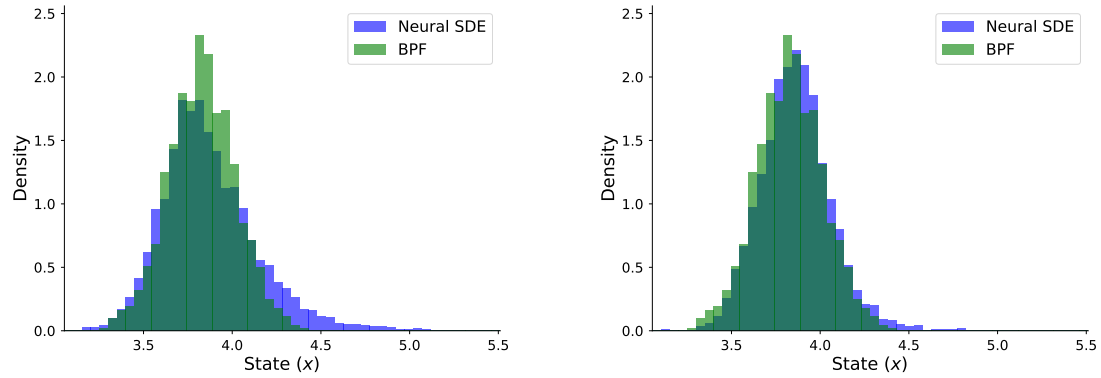


(e) BPF

Figure 7.6: Path distribution of the neural SDEs for an observation $Z_1 = 3.8$. The SDE is given by (7.2).



(a) SMH **(b)** SJ



(c) PMH **(d)** PJ

Figure 7.7: Marginal distribution of the neural SDE and BPF at final time $T = 1.5$ for observation $Z_1 = 3.8$. The SDE is given by (7.2).

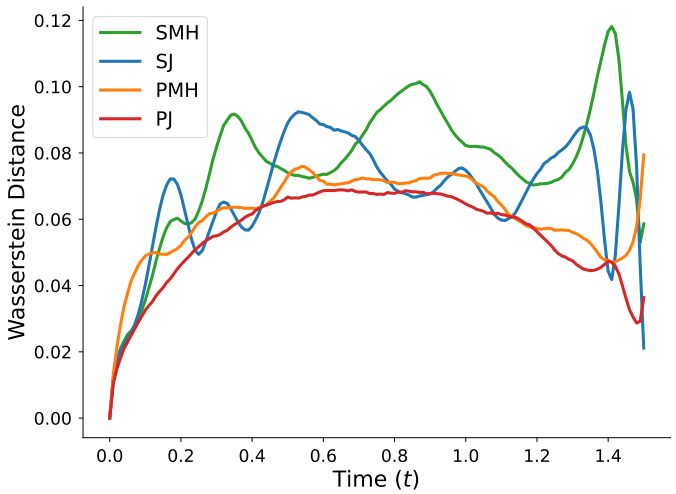


Figure 7.8: Average Wasserstein distance between the marginal distributions of the neural SDEs and the BPF across all time instances. SDE given by (7.2).

7.2 One observation in two spatial dimensions

For the test case in two spatial dimensions, we extend the bimodal distribution (7.1) by rotating it along the time axis. We seek identical radial dynamics, meaning that the associated ODE is given by

$$\mathrm{d}r = (-4r^3 + 6r)\,\mathrm{d}t,$$

with $r = \|X_t\|$. By changing the variable to X_t we get a factor $X_t/\|X_t\|$ on the right-hand side. We then add the noise back to obtain the resulting SDE

$$\begin{cases} \mathrm{d}X_t = (-4X_t\|X_t\|^2 + 6X_t)\,\mathrm{d}t + 0.5\,\mathrm{d}W_t, \quad t \in [0, 1.5], \\ X_0 = (0,0)^\top, \\ Z_1 = X_{1.5} + \mathcal{N}(0, \Sigma_z). \end{cases} \tag{7.3}$$

This SDE gives rise to a cup shape, as seen in Figure 7.9.

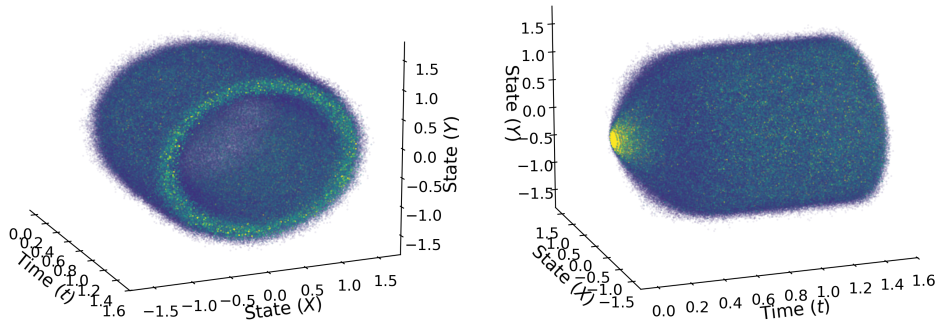
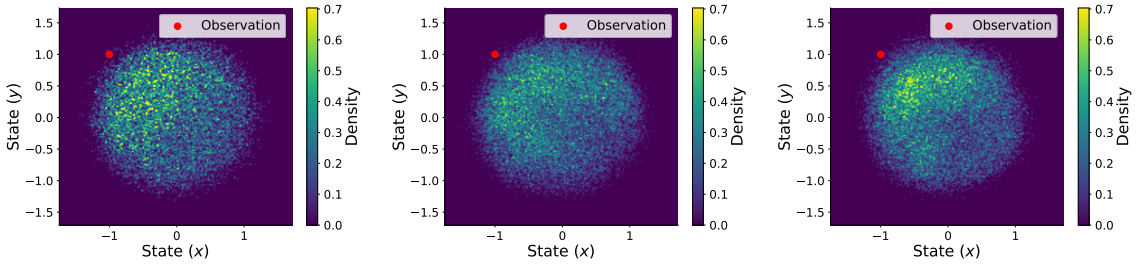
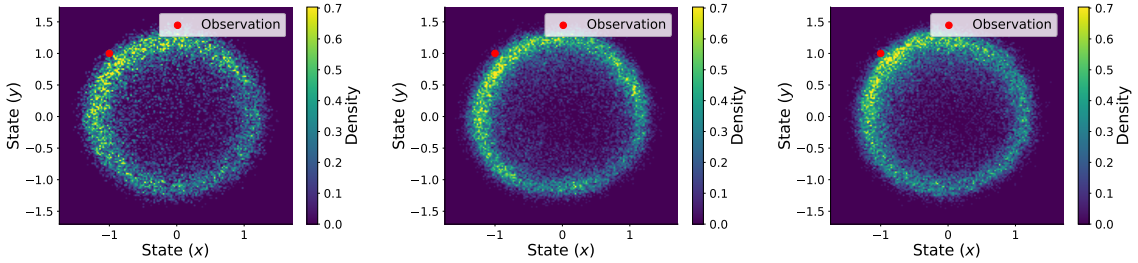


Figure 7.9: Path distribution of the SDE in (7.3).

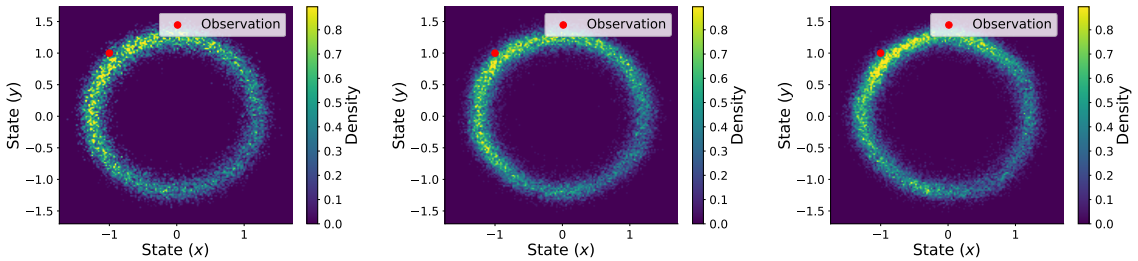
We once again have a single measurement Z_1, observed at $t = 1.5$, with measurement noise $\Sigma_z = \sigma_z \mathbf{I}$, $\sigma_z = 2$. As before, the BPF is still a good enough approximation to represent the true conditional solution. The main challenges for the neural SDEs are, first, to accurately capture the cup-shaped pattern and, second, to concentrate more density near the observation. Figure 7.10 compares the SJ and PJ frameworks with the BPF, showing the distributions at various times for observation $Z_1 = (-1, 1)^\top$.



(a) BPF at $t = 0.3$ **(b)** SJ at $t = 0.3$ **(c)** PJ at $t = 0.3$



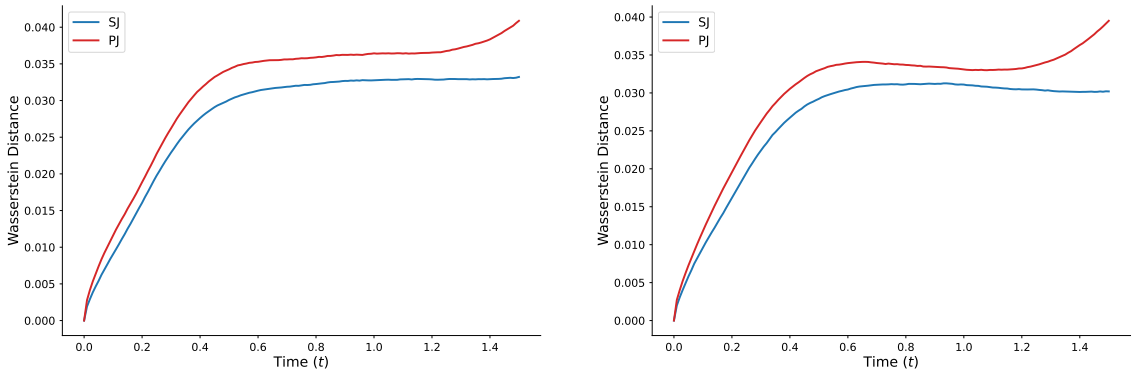
(d) BPF at $t = 0.5$ **(e)** SJ at $t = 0.5$ **(f)** PJ at $t = 0.5$



(g) BPF at $t = 1.5$ **(h)** SJ at $t = 1.5$ **(i)** PJ at $t = 1.5$

Figure 7.10: Distribution of the SDE (7.3) at various time instances.

We see both methods manage to recreate the cup shape with more density towards the observation.



(a) Marginal of the first dimension **(b)** Marginal of the second dimension

Figure 7.11: Average Wasserstein distance between the marginal distributions of the spatial dimensions of the neural SDEs and the BPF across all time instances. SDE given by (7.3).

7.3 Five observations in one spatial dimension

We extend the problem to include five observations in one spatial dimension. We consider the linear SDE.

$$\begin{cases} \mathrm{d}X_t = -X_t\,\mathrm{d}t + \mathrm{d}W_t, \quad t \in [0, 1.5], \\ X_0 = 0, \\ Z_i = X_{t_i} + \mathcal{N}(0, \sigma_Z^2), \quad i = 1, 2, 3, 4, 5, \end{cases} \tag{7.4}$$

where $t_i = (0.3, 0.6, 0.9, 1.2, 1.5)^\top$ and $\sigma_Z = 0.25$. This linear SDE, called an Ornstein–Uhlenbeck process, has an analytical solution (see Appendix D). As we will see, the BPF performs poorly in this setting with five observations, and we instead compare our models to the analytical solution. Furthermore, we illustrate the path distribution of the SDE in Figure 7.12 and note the mean-reverting behavior that comes with the drift term.

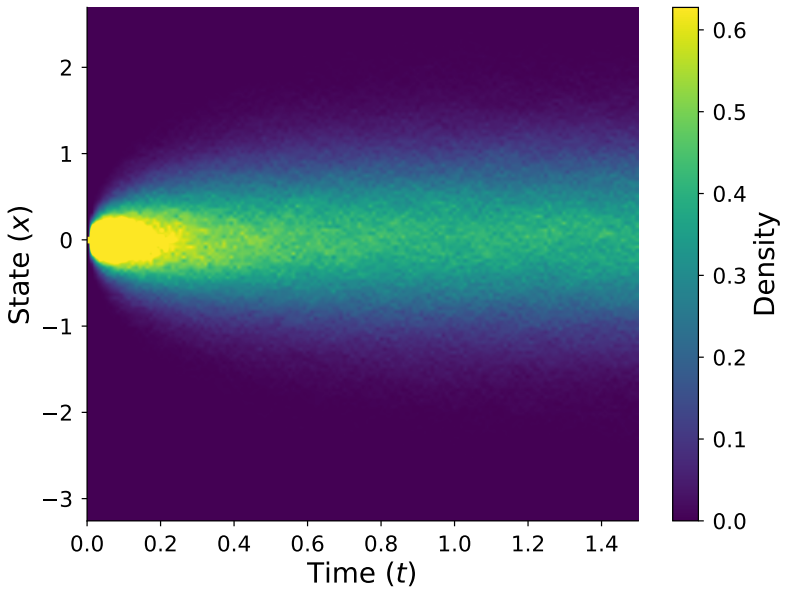
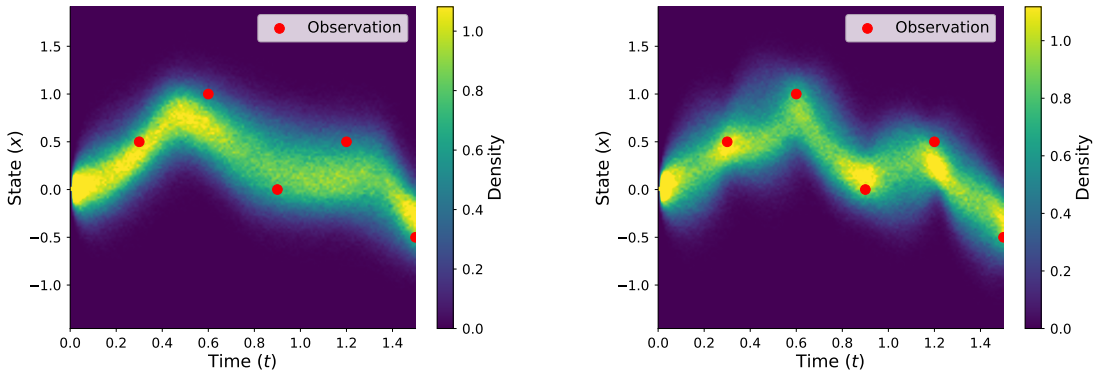


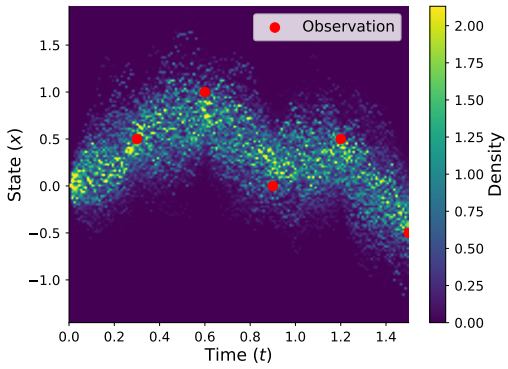
Figure 7.12: Path distribution of SDE in (7.4).

In Figure 7.13 we plot the path distributions of PJ, SJ and BPF compared to the analytical distribution. Furthermore, in Figure 7.14, we quantitatively compare the methods with the analytical solution by comparing the Wasserstein distance between the distributions and that of the analytical one in each time step. The findings underscore BPF's poor performance and also indicate that PJ appears to handle the smoothing problem with numerous observations better than SJ. However, PJ and in particular SJ struggle to tighten the distribution correctly around the observations.

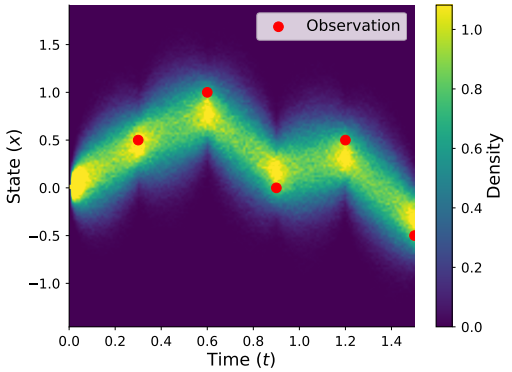


(a) SJ

(b) PJ



(c) BPF



(d) Analytical solution

Figure 7.13: Path distribution of the neural SDEs in the case of five observations $(0.5, 1, 0, 0.5, -0.5)$ at times $(0.3, 0.6, 0.9, 1.2, 1.5)$. The SDE is given by (7.4).

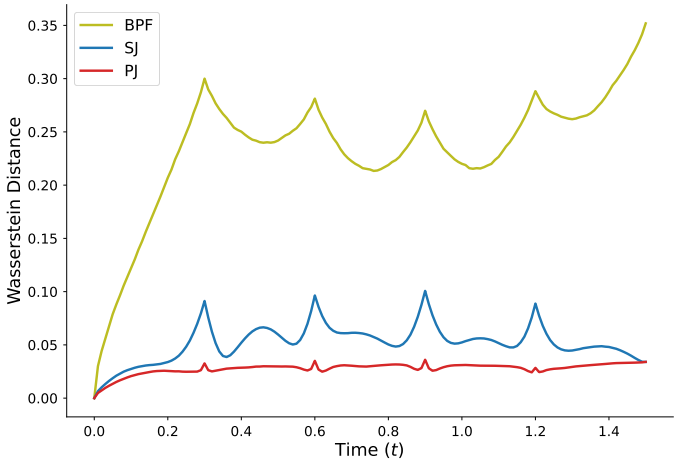


Figure 7.14: Average Wasserstein distance between the marginal distributions of the neural SDEs and the BPF across all time instances. The SDE is given by (7.4).

8

Results for the Filtering Problem

This chapter presents the results for the filtering problem, linked to the framework discussed in Section 6. We examine the same linear test case as earlier, since one can obtain an analytical solution to benchmark against. We will provide results in one and ten spatial dimensions.

8.1 One spatial dimension

Recall the SDE in Section 7.3 but now with only one observation,

$$\begin{cases} \mathrm{d}X_t = -X_t\,\mathrm{d}t + \mathrm{d}W_t, \quad t \in [0, 1.5], \\ X_0 = 0, \\ Z_1 = X_{1.5} + \mathcal{N}(0, \sigma_Z^2), \end{cases} \tag{8.1}$$

and a rather small observation noise, $\sigma_Z^2 = 0.125$. We use a small observation noise to highlight the importance of good proposal functions. We remind the reader that the unconditional dynamics is illustrated in Figure 7.12.

In Figure 8.1, we plot the proposals for each method in the first column and the weighted particles of the filtering step in the second. In the third column, we illustrate how much control is used for each method. We cap the heatmap since the norm of the analytical control becomes large when t approaches T when the state is far from the observation. Lastly, in Figure 8.1, we quantitatively evaluate the performance by analyzing the effective sample size (ESS). ESS measures how many particles contribute to the filter and is defined as

$$\mathrm{ESS} = \frac{1}{I \sum_{i=1}^{I} \pi_i^2}.$$

Note that if all particles have the same weight, π_i, ESS = 1. Analyzing the figures, we see that the CDT algorithm performs very well, with a performance similar to that of the analytical solution. In Section 2.3, we discussed that an optimal proposal would lead to equal weights if we have the same initial state, which in turn would lead to ESS = 1. The reason why this is not the case here is due to the discretization error that is introduced when integrating with Euler–Maruyama. Changing the integration method or using a smaller step size would mitigate this error. In Appendix E we discuss this further and illustrate how decreasing the step

size improves ESS. Furthermore, we see that PJ and SJ perform worse than CDT. Looking at the third column in Figure 8.1, we conclude that the reason is that these methods tend to use unnecessary control and guide the paths too early. Overall, the three methods perform much better than the BPF.

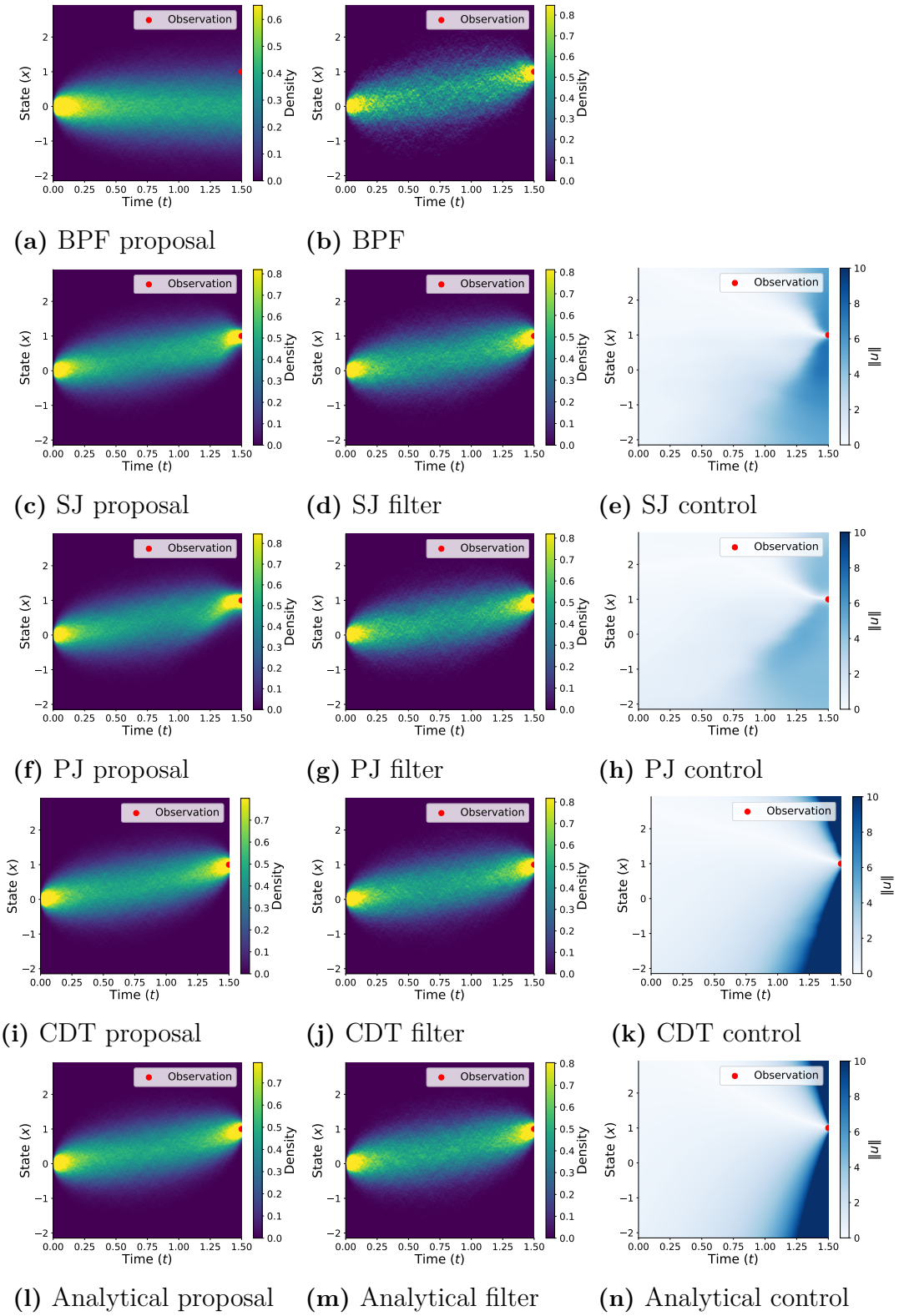
An additional subtle remark to make is that there is a white line angled down towards the observations as time approaches 1.5 in the control plot. This is where the control is zero, and it shows the optimal state to be in as we approach the observation. The reason it is angled downward is due to drift $-X_t$, which causes a mean-reversing behavior. In other words, it is expected that the paths will converge naturally. Furthermore, it is impressive that all algorithms manage to capture this behavior to some degree.

8.2 Ten spatial dimensions

We consider the same SDE as in the previous section, but this time in ten spatial dimensions and using the diagonal measurement noise matrix $\Sigma_z = \sigma_z \mathbf{I}$, $\sigma_z = 0.125$. Note that the dynamics in each dimension is uncorrelated and follows the distribution illustrated in Figure 7.12. Since the signature transform scales badly, we deviate from our standard truncation level $m = 4$ to truncation level $m = 3$.

In Figure 8.3, we analyze the first dimension for each method given the observation $(1, 0, 0, 0, 0, 0, 0, 0, 0, 0)^\top$. Once again, the first column displays the proposal distribution, the second column shows the weighted particles of the filtering step, and the third column shows the amount of control used by each method. Lastly, in Figure 8.4, we quantitatively evaluate performance by analyzing the ESS.

We see that the ESS drops drastically for all methods. CDT still performs best of our neural SDEs, but the ESS is now only half that of the analytical filter. Interestingly, SJ scales better in dimension compared to PJ. Lastly, we confirm the poor performance of the BPF that is unable to represent any distribution at all with all the density collected in just a few particles.



(a) BPF proposal **(b)** BPF

(c) SJ proposal **(d)** SJ filter **(e)** SJ control

(f) PJ proposal **(g)** PJ filter **(h)** PJ control

(i) CDT proposal **(j)** CDT filter **(k)** CDT control

(l) Analytical proposal **(m)** Analytical filter **(n)** Analytical control

Figure 8.1: Results from the particle filter for different methods. The first column lists the proposal distribution, the second shows the filtered distribution, and the third presents the control's norm. The SDE is given by (8.1) in one dimension.

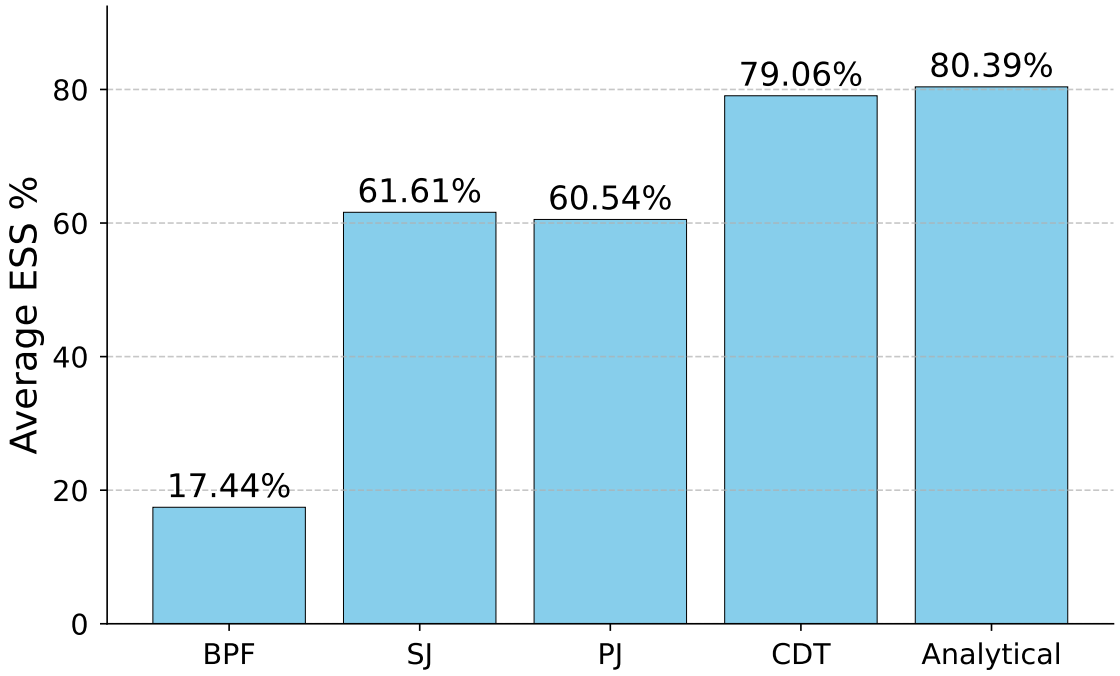
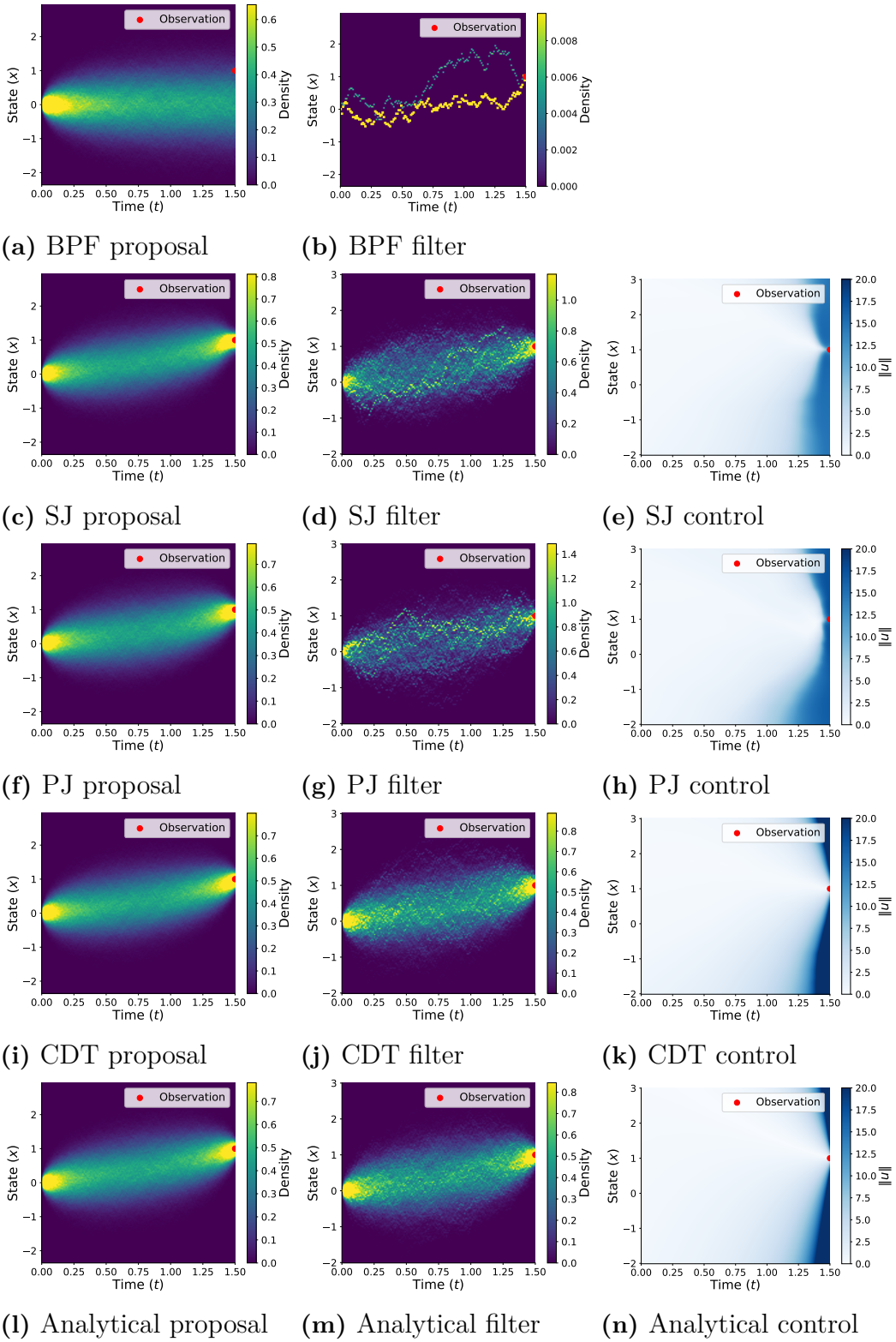


Figure 8.2: Average ESS for the different particle filters. The SDE is given by (8.1) in one dimension.



(a) BPF proposal **(b)** BPF filter

(c) SJ proposal **(d)** SJ filter **(e)** SJ control

(f) PJ proposal **(g)** PJ filter **(h)** PJ control

(i) CDT proposal **(j)** CDT filter **(k)** CDT control

(l) Analytical proposal **(m)** Analytical filter **(n)** Analytical control

Figure 8.3: Results from the particle filter for different methods. The first column lists the proposal distribution, the second shows the filtered distribution, and the third presents the control's norm. The SDE is given by (8.1) in ten dimensions.

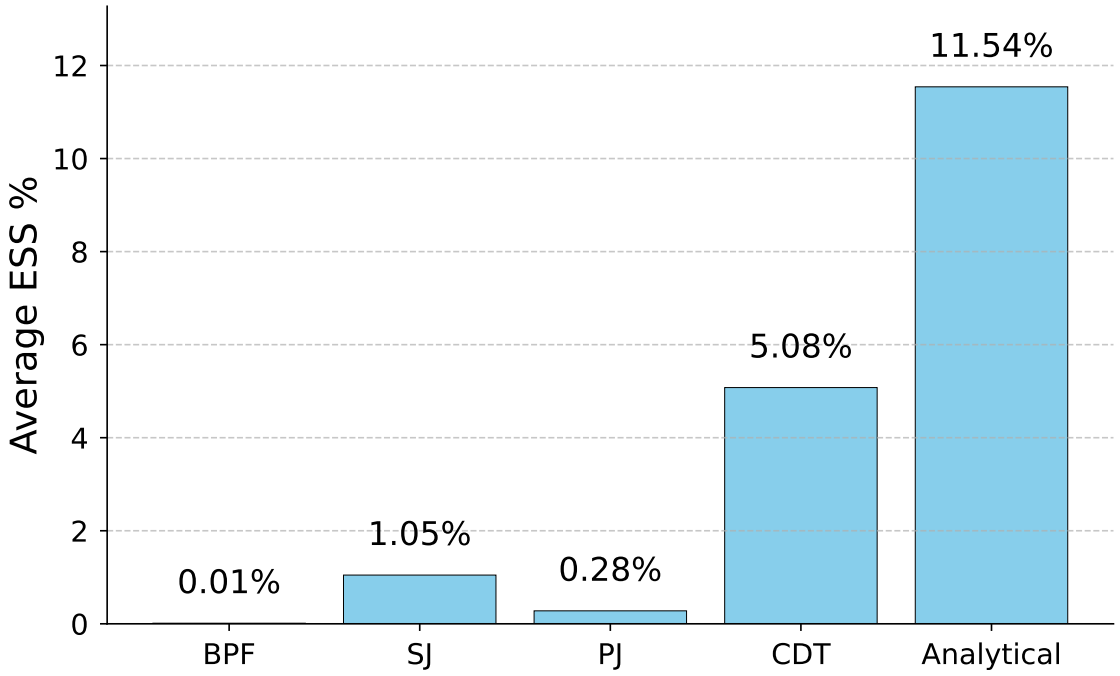


Figure 8.4: Average ESS for the different particle filters. The SDE is given by (8.1) in ten dimensions.

9
Discussion and Future Work

This thesis explored how different loss functions within the WGAN framework influence the performance of neural SDEs in learning conditional distributions over target trajectories. The results demonstrate that neural SDEs are capable of capturing complex and uncertain dynamics in both one- and two-dimensional settings.

We found a set of methods that performed equally well in most test cases, with differences emerging primarily in the smoothing setting with five observations. The signature-based method faced difficulties in squeezing the distribution around the observations. This is most likely due to the signature transform not highlighting local behavior as much as the global dynamics of the paths. On the other hand, signature-based methods had better performance when increasing the dimensionality.

Although the filtering performance did not match that of the CDT approach, this comparison must be viewed in context. CDT methods rely on strong prior assumptions, such as access to the true unconditional dynamics and the observation model. Although the same assumptions are also required for particle filtering and are reasonable in many settings, the WGAN approach investigated in this thesis does not require explicit knowledge of such models. This highlights its flexibility and potential applicability in scenarios where prior information is limited, uncertain, or difficult to specify.

One consistent pattern observed throughout the test cases is that the models sometimes fail to adequately tighten the distribution around the observation points. Although global behavior is often effectively modeled, local alignment with known data can be improved. This indicates that there is room to refine existing loss functions to place greater emphasis on accuracy at observation times, for example, by adjusting loss weightings or introducing localized penalties.

Based on all these insights, several directions for future development emerge.

- **Extend the state space.** Incorporate velocity and acceleration into the dynamics to increase the realism of the model and bring it closer to representations used in real-world tracking systems. Evaluating how the current framework scales with these richer dynamics will be important for understanding its broader applicability.
- **Incorporate Lévy jumps.** Introduce jump processes into the model to better represent discontinuous behavior such as abrupt maneuvers, thrust changes, or braking. This would allow neural SDEs to more accurately capture real-world

trajectory features that are not well modeled by purely continuous dynamics.
- **Refine the loss function.** Improve the alignment between generated trajectories and observed data by modifying the way loss is applied around observation points. This could improve local accuracy while preserving the overall structure and realism of the trajectories.

The overall purpose of this project was achieved, that is, to investigate how different loss functions affect the ability of neural SDEs to approximate conditional distributions over trajectories. The results demonstrate that the models can effectively capture uncertainty and structure in path prediction and that the choice of loss function has a meaningful impact. These findings provide a solid foundation for the further development and application of neural SDEs in more complex and realistic tracking scenarios.

Bibliography

[1] Øksendal, B. (2003). *Stochastic Differential Equations: An Introduction with Applications* (6th ed.). Springer, Berlin, Heidelberg. Available at: https://doi.org/10.1007/978-3-642-14394-6

[2] Chopin, N., Fulop, A., Heng, J., & Thiery, A. H. (2023). Computational Doob's h-transforms for online filtering of discretely observed diffusions. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*. Available at: https://doi.org/10.48550/arXiv.2206.03369

[3] Tzen, B., & Raginsky, M. (2019). Theoretical guarantees for sampling and inference in generative models with latent diffusions. *Proceedings of the 32nd Conference on Learning Theory (COLT)*. Available at: https://doi.org/10.48550/arXiv.1903.01608

[4] Zuo, J., Yan, B., & Lian, W. (2018). Forward–backward particle smoother for non-linear systems with one-step random measurement delay. *IET Signal Processing*, *12*, 836–843. Available at: https://doi.org/10.1049/iet-spr.2016.0673

[5] Naesseth, C. A., Lindsten, F., & Schön, T. B. (2024). Elements of Sequential Monte Carlo. *arXiv preprint* arXiv:1903.04797. Available at: https://arxiv.org/abs/1903.04797

[6] Yan, B. (n.d.). Chapter 6: The Riemann-Stieltjes Integral. *Course notes, Michigan State University*. Available at: https://users.math.msu.edu/users/yanb/327ch6.pdf

[7] Levin, D., Lyons, T., & Ni, H. (2013). Learning from the past, predicting the statistics for the future, learning an evolving system. *arXiv preprint* arXiv:1309.0260. Available at: https://arxiv.org/abs/1309.0260

[8] Chevyrev, I., & Lyons, T. (2013). Characteristic functions of measures on geometric rough paths. *arXiv preprint* arXiv:1307.3580. Available at: https://arxiv.org/abs/1307.3580

[9] Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. *arXiv preprint* arXiv:1701.07875. Available at: https://arxiv.org/abs/1701.07875

[10] Winqvist, S., & Wikman, I. (2024). Prospects of neural SDEs for Bayesian smoothing. Master's thesis, Chalmers University of Technology. Available at: http://hdl.handle.net/20.500.12380/308024

[11] Bonnier, P., Kidger, P., Perez Arribas, I., Salvi, C., & Lyons, T. (2019). Deep Signature Transforms. *arXiv preprint* arXiv:1905.08494. Available at: https://arxiv.org/abs/1905.08494

[12] Wang, Y., & Ni, Y.-H. (2022). Deep BSDE-ML learning and its application to model-free optimal control. *arXiv preprint* arXiv:2201.01318. Available at: https://arxiv.org/abs/2201.01318

[13] Robert, C. P., & Casella, G. (2004). *Monte Carlo Statistical Methods* (2nd ed.). Springer, New York, NY. Available at: https://doi.org/10.1007/978-1-4757-4145-2

[14] Holt, W., & Nguyen, D. (2023). Essential Aspects of Bayesian Data Imputation. *SSRN Electronic Journal*. Available at: https://ssrn.com/abstract=4494314

A

Metropolis–Hastings Algorithm

The Metropolis–Hastings (MH) algorithm makes it possible to sample paths X from the conditional distribution $\mu_{X|Z_{1:N}}$. We write the conditional density of an unconditional path as

$$p(X \mid Z_{1:N}) \;\propto\; p(X)\, p(Z_{1:N} \mid X),$$

and define a target density

$$\pi(X) \;=\; p(X)\, p(Z_{1:N} \mid X).$$

In the MH framework, we construct a Markov chain that has π as its stationary distribution. Specifically, consider an independent proposal distribution $q(X' \mid X) = p(X')$, i.e., each proposed path X' is drawn independently of the current path X. The probability of accepting X' is given by

$$\alpha(X, X') \;=\; \min\left\{1,\; \frac{\pi(X')\, q(X \mid X')}{\pi(X)\, q(X' \mid X)}\right\}.$$

If accepted, X' becomes the current path and is recorded as a sample of the distribution; if rejected, the chain remains at X, which is then recorded again. Since $q(X' \mid X) = p(X')$ and $q(X \mid X') = p(X)$, the acceptance probability simplifies to

$$\alpha(X, X') \;=\; \min\left\{1,\; \frac{p(Z_{1:N} \mid X')}{p(Z_{1:N} \mid X)}\right\}.$$

This procedure is repeated until the desired number of samples from the conditional distribution has been obtained. Additionally, we must show that π is indeed the stationary distribution and to do that we introduce the detailed balance condition. If the detailed balance condition is satisfied, then π is the stationary distribution [13]. Let $\kappa > 0$ be a constant such that $\pi(X) = \kappa\, p(X)\, p(Z_{1:N} \mid X)$ and define the balance condition as

$$\pi(X)\, q(X' \mid X)\, \alpha(X, X') \;=\; \pi(X')\, q(X \mid X')\, \alpha(X', X), \quad \text{for all } X, X'. \tag{A.1}$$

By substituting π and q we get

$$\begin{cases} \pi(X)\, q(X' \mid X)\, \alpha(X, X') = \; \kappa\, p(X)\, p(Z_{1:N} \mid X)\; p(X')\; \min\left\{1,\; \dfrac{p(Z_{1:N} \mid X')}{p(Z_{1:N} \mid X)}\right\}, \\[2ex] \pi(X')\, q(X \mid X')\, \alpha(X', X) = \; \kappa\, p(X')\, p(Z_{1:N} \mid X')\; p(X)\; \min\left\{1,\; \dfrac{p(Z_{1:N} \mid X)}{p(Z_{1:N} \mid X')}\right\}. \end{cases}$$

We have two different cases; first assume that

$$\frac{p(Z_{1:N} \mid X')}{p(Z_{1:N} \mid X)} \leq 1,$$

which yields

$$\alpha(X, X') = \frac{p(Z_{1:N} \mid X')}{p(Z_{1:N} \mid X)} \quad \text{and} \quad \alpha(X', X) = 1.$$

Hence, the balance condition is fulfilled,

$$\pi(X)\, q(X' \mid X)\, \alpha(X, X') = \pi(X')\, q(X \mid X')\, \alpha(X', X).$$

For the second case where X and X' are switched, the equality in (A.1) is maintained due to symmetry. Normalizing $\pi(X)$ over X recovers $\mu_{X|Z_{1:N}}$ as the limiting distribution. Note that the Metropolis–Hastings algorithm scales poorly, for this choice of proposal function, with increasing dimensionality, as the probability that an unconditional path aligns well with a given observation becomes exceedingly low, particularly when the observation noise is small.

B

WGAN in Signature Space with Moment-Matching

In this appendix, we present an alternative approach that extends the methodology proposed in [10]. We first outline their framework and subsequently introduce our extension, which incorporates moment matching in signature space. The corresponding results are provided, along with a discussion of the reasons for not further pursuing this approach.

We begin by expanding the notion of WGAN to signature space. We denote $S(\mathcal{X})$ as the space obtained by applying the signature transform to augmented paths $\mathbf{X}$ where the non-augmented paths $X \in \mathcal{X}$. We also define a metric that acts on signatures as $d_{\mathcal{S}} = \|S(\mathbf{X}) - S(\mathbf{Y})\|_2$, leading to a metric space $(S(\mathcal{X}), d_{\mathcal{S}})$. Lastly, we define the conditional Wasserstein distance in signature space as

$$\begin{aligned}
&\arg\min_{\nu_{X,Z_{1:N}}} W_1(\mu_{X,Z_{1:N}}, \nu_{X,Z_{1:N}}) \\
&= \arg\min_{\nu_{X,Z_{1:N}}} \sup_{\varphi \in \mathrm{Lip}_1(S(\mathcal{X})\times\mathcal{Z})} \mathbb{E}_{Z_{1:N}\sim\mu_{Z_{1:N}}}\Big[\mathbb{E}_{X'\sim\mu_{X|Z_{1:N}}}[\varphi(X', Z_{1:N})] - \mathbb{E}_{X'\sim\nu_{X|Z_{1:N}}}[\varphi(X', Z_{1:N})]\Big],
\end{aligned}$$

where $\varphi \in \mathrm{Lip}_1(S(\mathcal{X}) \times \mathcal{Z})$ if

$$|\varphi(X, Z_{1:N}) - \varphi(Y, Z_{1:N})| \leq \|S(\mathbf{X}) - S(\mathbf{Y})\|_2, \quad X, Y \in \mathcal{X}, \quad Z_{1:N} \in \mathcal{Z}.$$

In addition, the critic is defined as in Section 5.2.1 with the composition $\varphi_S(X, Z_{1:N}) = \Phi_S(S^m(\mathbf{X}), Z_{1:N})$. In [10] the authors show that this can be rewritten as a minimization problem with the truncated signature loss

$$\text{Sig-}W_1^m(\mu_{X,Z_{1:N}}, \nu_{X,Z_{1:N}}) = \mathbb{E}_{Z_{1:N}\sim\mu_{Z_{1:N}}}\Big[\Big\|\mathbb{E}_{X\sim\mu_{X|Z_{1:N}}}[S^m(X)] - \mathbb{E}_{X\sim\nu_{X|Z_{1:N}}}[S^m(X)]\Big\|_2\Big].$$

In other words, the loss function is defined as the Euclidean norm of the difference between the expected signatures of the true and generated distributions. In light of Theorem 6, this choice of loss is well justified. However, computing the true conditional expectation $\mathbb{E}_{X\sim\nu_{X|Z_{1:N}}}[S^m(X)]$ is needed. The proposed approach addresses this by using a pre-trained neural network designed to map an observation to the expected signature of the corresponding conditioned paths.

Using this loss, the authors of [10] faced convergence challenges, particularly because distributions that were close in the signature space, as measured by the Euclidean norm, did not necessarily correspond to similar trajectories in the path space. We illustrate this issue using the bimodal distribution in (7.1). Figure B.1 displays the expected signatures for three cases: paths that end in each mode individually and all paths combined. In some signature dimensions, the contributions from paths going to opposite modes have opposing signs and cancel out, causing the overall expected signature to vanish in those components. This creates ambiguity, making it difficult for a neural SDE to infer the true underlying dynamics based solely on the expected signature. Although Theorem 6 guarantees that the expected signature uniquely characterizes a distribution, this only holds when the full (infinite) signature is used. Since we rely on a truncated version in practice, key information in higher-order terms may be lost, preventing the model from capturing distributions like the bimodal one accurately.

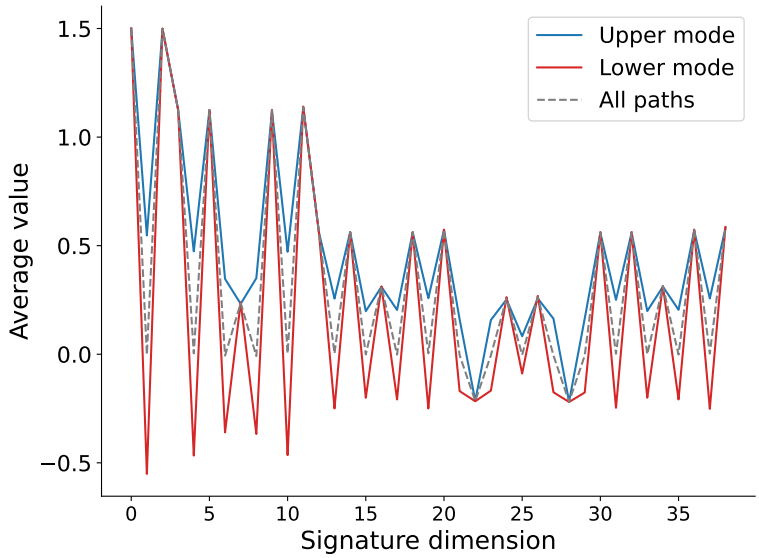


Figure B.1: Expected signature for paths of the bimodal SDE (7.1).

With this in mind, a natural approach to addressing this issue is to incorporate higher moments in signature space, under the assumption that this will compensate for the information lost in higher order terms. This leads to the following loss function to be minimized, which is formulated using the centralized moments of the true distribution,

$$\mathcal{L}(\mu_{X,Z_{1:N}}, \nu_{X,Z_{1:N}}) = \mathbb{E}_{Z_{1:N}\sim\mu_{Z_{1:N}}}\left[\sum_{k=1}^{\mathcal{K}} w_k \left\| \mathbb{E}_{X'\sim\mu_{X|Z_{1:N}}}\left[\left(S^m(X') - \mathbb{E}_{\breve{X}\sim\mu_{X|Z_{1:N}}}[S^m(\breve{X})]\right)^k\right] \right.\right.$$
$$\left.\left. - \mathbb{E}_{X'\sim\nu_{X|Z_{1:N}}}\left[\left(S^m(X') - \mathbb{E}_{\breve{X}\sim\mu_{X|Z_{1:N}}}[S^m(\breve{X})]\right)^k\right] \right\|_2\right].$$

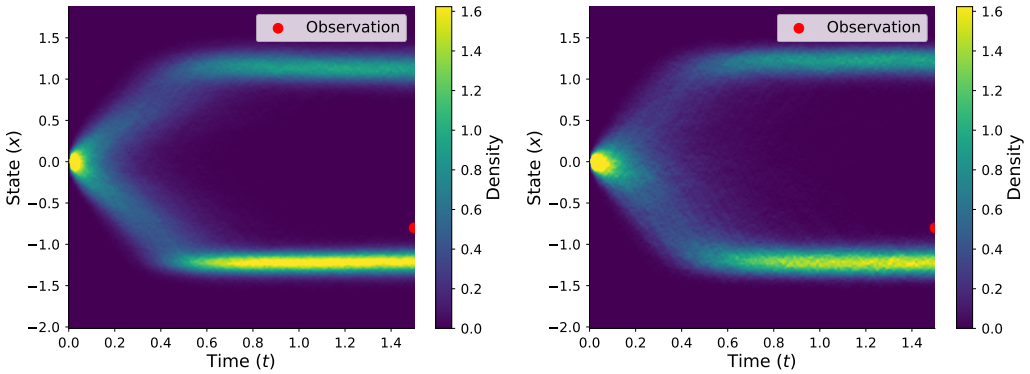
where w_k is the weight assigned to each moment. Similarly to [10], we approximate the true centralized moments

$$\mathbb{E}_{X' \sim \mu_{X|Z_{1:N}}} \left[\left(S^m(X') - \mathbb{E}_{\breve{X} \sim \mu_{X|Z_{1:N}}} [S^m(\breve{X})] \right)^k \right]$$

with pre-trained neural networks. For this approach, we use the shorthand name *MM*.

In Figure B.2, we present the path distribution for the bimodal SDE in (7.1) with observation $Z_1 = -0.8$. Furthermore, in Figure B.3, we present the marginal distribution at $t = 1.5$ for the neural SDE and the BPF given the same observation. Lastly, in Figure C.3, we compare the performance of MM with the methods presented in the main report. At each time step, we compute the Wasserstein distance between the paths generated by MM and those produced by the BPF. These distances are then averaged over observations Z_1 drawn from the distribution μ_z.

The approach demonstrates competitive performance compared to the methods presented in the main text. However, it was not included because the optimal choice of hyperparameters, $\mathcal{K}$ (the number of moments) and w_k (the corresponding weights), proved to be problem-dependent, making their selection a non-trivial task. As a result, we did not pursue this method further, although it remains an interesting finding that could be valuable for those seeking an alternative to training a notoriously hard min-max WGAN, as this approach is formulated as a minimization problem.



(a) Path distribution from MM **(b)** Path distribution from BPF

Figure B.2: Path distribution of the neural SDE for an observation $Z_1 = -0.8$. The SDE is given by (7.1).

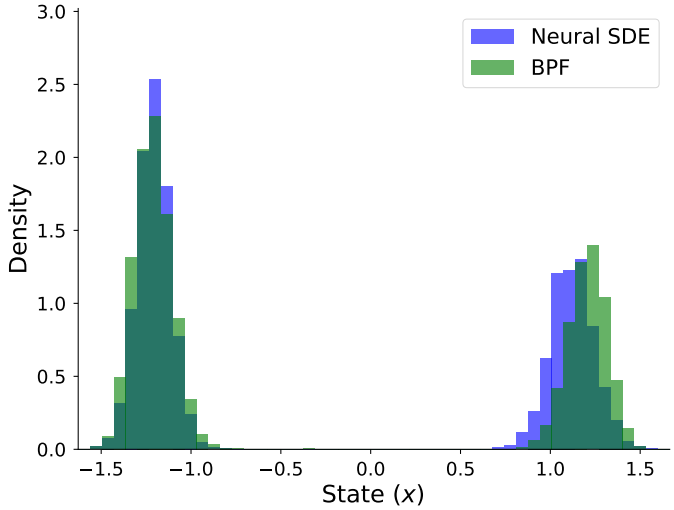


Figure B.3: Marginal distribution of the neural SDE and BPF at final time $T = 1.5$ for observation $Z_1 = -0.8$. The SDE is given by (7.1) and neural SDE trained with MM approach.

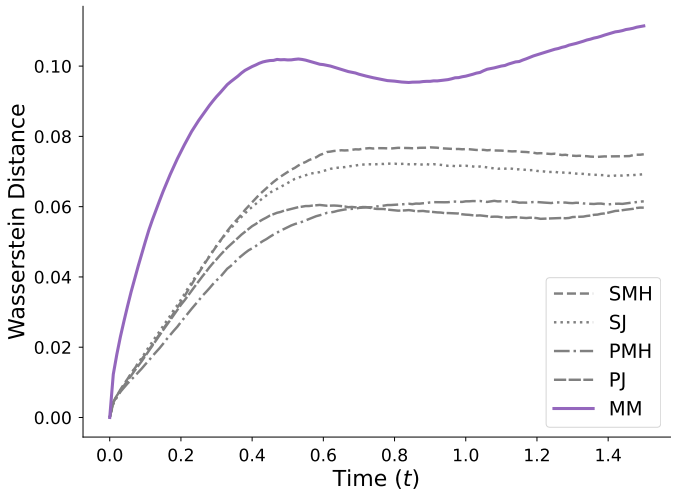


Figure B.4: Average Wasserstein distance between the marginal distributions of the neural SDEs and the BPF across all time instances. The SDE is given by (7.1).

C

Learn Expectation by a Neural Network

In this Chapter, we cover another way to address the problem of the conditioned expectation in a WGAN. Recall the Metropolis–Hastings algorithm in Appendix A, where we first got a distribution of paths conditioned on observations. The expectation over the critic was then approximated by the distribution. We will now present a method where we jump directly to the expectation over the critic, without the need of a distribution. We will do this by approximating the expectation by a neural network. With this method, we only consider the signature loss explained in Section 5.2.1. The path space critic was tested as well, but the results were deemed too poor.

We would like to learn the function

$$(\varphi, Z_{1:N}) \mapsto \mathbb{E}_{X\sim\mu_{X|Z_{1:N}}}[\varphi(X, Z_{1:N})]$$

with a neural network. To do this in a feasible way we restrict the critic φ to the set

$$\mathcal{C} = \left\{ \varphi \in \mathrm{Lip}_1(\mathcal{X}\times\mathcal{Z}) \,\middle|\, \varphi(X, Z_{1:N}) = \sum_{l=1}^{L} c_l\, g\Big(\Big\langle a_l, (S^m(\mathbf{X}), Z_{1:N})\Big\rangle + b_l\Big)\right\},$$

where g is an activation function, the parameters of the critic $\phi = (a_l, b_l, c_l) \in \mathbb{R}^{d_s+Nd}\times\mathbb{R}\times\mathbb{R}$, $l \in \{1, \dots, L\}$ where d_s is the dimension of the truncated signature. Concretely, $\mathcal{C}$ is the set of compositions $\varphi(X, Z_{1:N}) = \Phi_S^{\mathcal{C}}(S^m(\mathbf{X}), Z_{1:N})$ where $\Phi_S^{\mathcal{C}}$ is a neural network with one hidden layer of size L. Because of linearity, the conditional expectation can now be written as

$$\mathbb{E}_{X'\sim\mu_{X|Z_{1:N}}}[\varphi(X', Z_{1:N})] = \sum_{l=1}^{L} c_l\, \mathbb{E}_{X'\sim\mu_{X|Z_{1:N}}}\Big[g(\langle a_l, (S^m(X'), Z_{1:N})\rangle + b_l)\Big].$$

We would like to train a network Ξ to map

$$(a, b, Z_{1:N}) \mapsto \mathbb{E}_{X'\sim\mu_{X|Z_{1:N}}}[g(\langle a, (S^m(X'), Z_{1:N})\rangle + b)],$$

for all parameters $(a, b) \in (a_l, b_l)_{l=1}^{L}$. We would then have

$$\mathbb{E}_{X'\sim\mu_{X|Z_{1:N}}}[\varphi(X', Z_{1:N})] = \sum_{l=1}^{L} c_l\, \Xi(a_l, b_l, Z_{1:N}).$$

To train Ξ we update its weights ξ by the loss function

$$\mathcal{L}_\Xi(\xi) = \mathbb{E}_{X',Z_{1:N}\sim\mu_{X,Z_{1:N}}}\left[L^{-1}\sum_{l=1}^{L}\left[g\Big(\Big\langle a_l,(S^m(X'),Z_{1:N})\Big\rangle + b_l\Big) - \Xi^\xi(a_l,b_l,Z_{1:N})\right]^2\right].$$

Once a well-trained Ξ is obtained, we can rewrite the loss for φ. The parameters $\phi = (a_l, b_l, c_l)_{l=1}^L$ of the critic will minimize the loss

$$\mathcal{L}_\varphi(\phi) = -\mathbb{E}_{\mathbb{Z}_{\mathbb{N}:\mathbb{N}}\sim\mu_{\mathbb{Z}_{\mathbb{N}:\mathbb{N}}}}\left[\sum_{l=1}^{L} c_l\,\Xi(a_l,b_l,Z_{1:N}) - \mathbb{E}_{X'\sim\nu_{X|Z_{1:N}}}[\varphi^\phi(X',Z_{1:N})]\right] + \lambda_{\mathrm{P}}, \quad \text{(C.1)}$$

with

$$\lambda_{\mathrm{P}} = \mathbb{E}_{\hat{X},Z_{1:N}\sim\mu_{X,Z_{1:N}},\,\tilde{X}\sim\nu_{X|Z_{1:N}}}\left[\mathrm{ReLU}\left(\frac{|\varphi(\hat{X},Z_{1:N}) - \varphi(\tilde{X},Z_{1:N})|}{\|\hat{X}-\tilde{X}\|_\infty} - 1\right)^2\right].$$

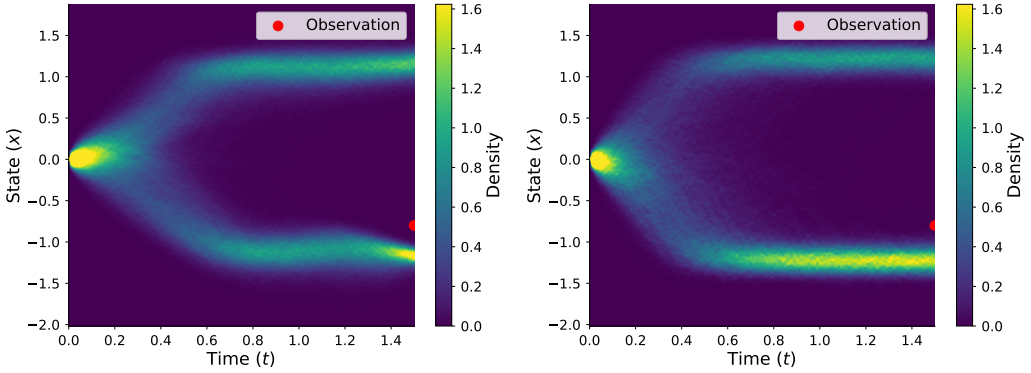
Note that instead of a standard WGAN, where we train the critic and generator iteratively, we now have yet another network in the mix. The training scheme goes as follows. Train

$$\Xi(a_l,b_l,Z_{1:N}) \approx \mathbb{E}_{X\sim\mu_{X|Z_{1:N}}}\left[g\Big(\Big\langle a_l,(S^m(\mathbf{X}),Z_{1:N})\Big\rangle + b_l\Big)\right]$$

for the current critic parameters a_l, b_l, $l = 1,\dots,L$. Update the critic by minimizing (C.1). Update the generator as normal. Repeat. For this approach, we use the shorthand name *LE*.

In Figure C.1, we present the path distribution for the bimodal SDE (7.1) with observation $Z_1 = -0.8$. Furthermore, in Figure C.2, we present the marginal distribution at $t = 1.5$ for the neural SDE and the BPF given the same observation. Lastly, in Figure C.3, we compare the performance of LE with the methods presented in the main report. At each time step, we compute the Wasserstein distance between the paths generated by LE and those produced by the BPF. These distances are then averaged over multiple observations Z_1 drawn from the distribution μ_z.

The approach performs poorly compared to the methods presented in the main text, as it is not able to generate the correct structure or weigh the modes correctly.



(a) Path distribution from LE **(b)** Path distribution from BPF

Figure C.1: Path distribution of the neural SDE for an observation $Z_1 = -0.8$. The SDE is given by (7.1).

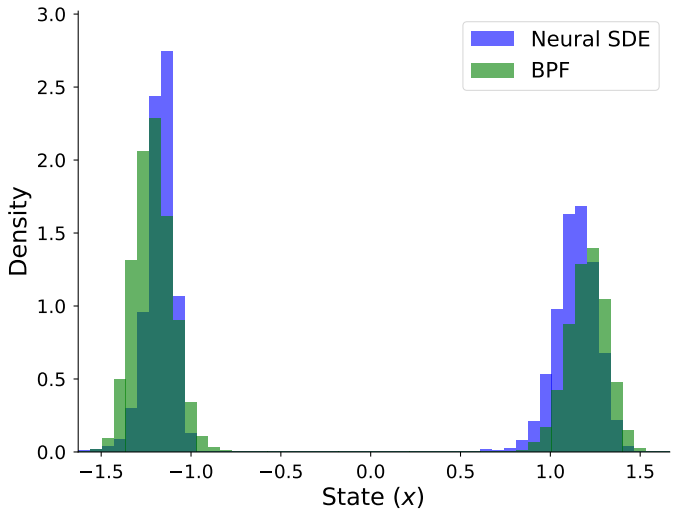


Figure C.2: Marginal distribution of the neural SDE and BPF at final time $t = 1.5$ for observation $Z_1 = -0.8$. The SDE is given by (7.1) and the neural SDE trained with LE approach.

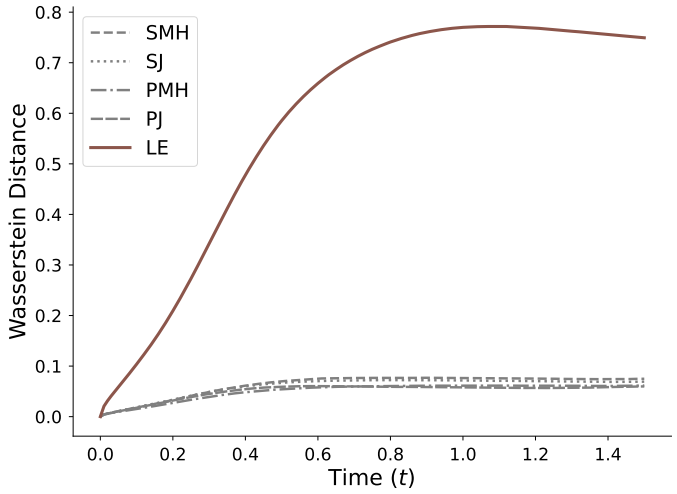


Figure C.3: Average Wasserstein distance between the distributions of the neural SDEs and the BPF across all time instances. The SDE is given by (7.1).

D

Calculating Analytical Distribution for Ornstein–Uhlenbeck Process

In this chapter we show the calculations for the analytical solution to the smoothing problem of the Ornstein–Uhlenbeck process used in Section 7.3. We also show the calculations for the optimal control of the same process used in Chapter 8.

D.1 Calculations for the smoothing problem

Consider the Ornstein–Uhlenbeck process

$$\mathrm{d}X_t = -X_t\,\mathrm{d}t + \mathrm{d}W_t, \quad X_0 = 0. \tag{D.1}$$

We have N observations

$$Z_{1:N} = X_{t_{1:N}} + \epsilon_{1:N}, \quad \epsilon_{1:N} \sim \mathcal{N}(0, \sigma_z^2\,\mathbf{I}_N)$$

occurring at times t_n, $n = 1, \ldots, N$. We want to condition the solution X on the observations $Z_{1:N} = z$. Since the SDE is linear, the solution is itself a Gaussian process. In particular, the random vector $(X_t, X_{t_1}, \ldots, X_{t_N})^\top$ has a multivariate normal distribution. Thus, the conditioned variable

$$X_t \mid Z_{1:N} = z \sim \mathcal{N}(\bar{\mu}, \bar{\Sigma}^2)$$

with

$$\begin{aligned}\bar{\mu} &= \mu_t + \Sigma_{t,Z}\,\Sigma_{Z,Z}^{-1}\,(z - \mu_Z),\\ \bar{\Sigma}^2 &= \Sigma_{t,t} - \Sigma_{t,Z}\,\Sigma_{Z,Z}^{-1}\,\Sigma_{t,Z}^\top,\end{aligned}$$

where

$$\begin{gathered}\mu_t = \mathbb{E}[X_t], \quad \mu_Z = \mathbb{E}[Z_{1:N}],\\ \Sigma_{t,t} = \mathrm{Cov}(X_s, X_t), \quad \Sigma_{t,Z} = \mathrm{Cov}(X_t, Z_{1:N}), \quad \Sigma_{Z,Z} = \mathrm{Cov}(Z_{1:N}, Z_{1:N}),\end{gathered}$$

[14]. To calculate the expectations and covariances we will start by finding an expression for X_t.

We multiply the SDE by the factor e^t to obtain

$$e^t\,\mathrm{d}X_t + e^t X_t\,\mathrm{d}t = e^t\,\mathrm{d}W_t. \tag{D.2}$$

The chain rule gives $\mathrm{d}(e^t X_t)$ on the left hand side. By integrating from 0 to t, using that $X_0 = 0$ and solving for X_t we get

$$X_t = e^{-t} \int_0^t e^s \, \mathrm{d}W_s.$$

We can now calculate

$$\mu_t = \mathbb{E}[X_t] = e^{-t} \, \mathbb{E}\left[\int_0^t e^s \, \mathrm{d}W_s\right] = 0,$$

since the integral computes to zero. Similarly,

$$\mu_Z = \mathbb{E}[Z_{1:N}] = \mathbb{E}[X_{t_{1:N}} + \epsilon_{1:N}] = \mathbb{E}[X_{t_{1:N}}] + \mathbb{E}[\epsilon_{1:N}] = 0.$$

Since $\mathbb{E}[X_t] = 0$,

$$\mathrm{Cov}(X_s, X_t) = \mathbb{E}[X_s X_t] = e^{-s-t} \, \mathbb{E}\left[\left(\int_0^s e^r \, \mathrm{d}W_r\right)\left(\int_0^t e^u \, \mathrm{d}W_u\right)\right].$$

We use Itô isometry to the expectation to get

$$\mathrm{Cov}(X_s, X_t) = e^{-s-t} \int_0^{\min(s,t)} e^{2r} \, \mathrm{d}r,$$

and after some rearrangements we obtain

$$\mathrm{Cov}(X_s, X_t) = \frac{1}{2} e^{-|t-s|}\left(1 - e^{-2\min(s,t)}\right). \tag{D.3}$$

Further we have

$$\mathrm{Cov}(X_t, Z_{1:N}) = \mathrm{Cov}(X_t, X_{t_{1:N}}) + \mathrm{Cov}(X_t, \epsilon_{1:N}),$$

and since X_t and $\epsilon_{1:N}$ are independent $\mathrm{Cov}(X_t, \epsilon_{1:N}) = 0$. The remaining covariance $\mathrm{Cov}(X_t, X_{t_{1:N}})$ we already have an expression for in (D.3). Lastly,

$$\mathrm{Cov}(Z_{1:N}, Z_{1:N}) = \mathrm{Cov}(X_{t_{1:N}}, X_{t_{1:N}}) + \mathrm{Cov}(\epsilon_{1:N}, \epsilon_{1:N})$$

where we can calculate the first covariance, again, from (D.3), and the second covariance $\mathrm{Cov}(\epsilon_{1:N}, \epsilon_{1:N}) = \sigma_z^2 \, \mathbf{I}_N$. To summarize, the distribution of the conditioned variable $X_t \mid Z_{1:N} = z$ is a normal distribution with mean

$$\bar{\mu} = \Sigma_{t,Z} \, \Sigma_{Z,Z}^{-1} \, z$$

and variance

$$\bar{\Sigma}^2 = \Sigma_{t,t} - \Sigma_{t,Z} \, \Sigma_{Z,Z}^{-1} \, \Sigma_{t,Z}^{\top},$$

where

$$\Sigma_{t,t} = \mathrm{Cov}(X_s, X_t), \quad \Sigma_{t,Z} = \mathrm{Cov}(X_t, X_{t_{1:N}}), \quad \Sigma_{Z,Z} = \mathrm{Cov}(X_{t_{1:N}}, X_{t_{1:N}}) + \sigma_z^2 \, \mathbf{I}_N.$$

D.2 Calculations for optimal control

We consider the SDE (D.1), this time with one observation Z at the terminal time T. The observation is made with some measurement noise σ_z. We start by finding the transition density from X_t to X_T. Consider (D.2) and integrate from t to T to get

$$e^T X_T - e^t X_t = \int_t^T e^r \, \mathrm{d}W_r,$$

we follow up by solving for X_T to obtain

$$X_T = e^{-(T-t)} X_t + e^{-T} \int_t^T e^r \, \mathrm{d}W_r.$$

Hence, $X_T \mid X_t \sim \mathcal{N}\left(\mu_x(X_t, t), \sigma_x^2(t)\right)$ with mean

$$\mu_x(X_t, t) = \mathbb{E}[X_T \mid X_t] = X_t e^{-(T-t)},$$

and variance

$$\sigma_x^2(t) = \mathrm{Var}\left(e^{-T} \int_t^T e^r \, \mathrm{d}W_r\right) = \frac{1}{2}\left(1 - e^{-2(T-t)}\right),$$

by Itô isometry. The probability of observing Z can now be expressed as

$$\begin{aligned} p(Z \mid X_t) &= \int_{\mathbb{R}^d} \mathcal{N}\left(Z; x_T, \sigma_z^2 \mathbf{I}_d\right) \mathcal{N}\left(x_T; \mu_x(X_t, t), \sigma_x^2(t) \mathbf{I}_d\right) \mathrm{d}x_T \\ &= (2\pi)^{-d/2} \sigma_x^{-d}(t) \sigma_x^{-d} \sigma_h^d(t) \exp\left\{\frac{1}{2}\sigma_h^2(t) \left\| \frac{\mu_x(X_t, t)}{\sigma_x(t)} + \frac{Z}{\sigma_z^2} \right\|^2 \right\} \\ &\quad \times \exp\left\{ -\frac{\|\mu_x(X_t, t)\|^2}{2\sigma_x(t)} - \frac{\|Z\|^2}{2\sigma_z^2}, \right\} \end{aligned}$$

where $\sigma_h^2(t) = \left(\sigma_x^{-2}(t) + \sigma_z^{-2}\right)^{-1}$. Further one can now compute the optimal control function $u_\star$ as

$$\begin{aligned} u_\star(X_t, t, Z) &= \sigma^\top(X_t, t) \nabla \log p(Z \mid X_t) \\ &= \frac{\sigma_h^2(t) \exp\left(-(T-t)\right)}{\sigma_x^2(t)} \left(\frac{\mu_x(X_t, t)}{\sigma_x^2(t)} + \frac{Z}{\sigma_z^2} \right) - \frac{\exp\left(-(T-t)\right)}{\sigma_x^2(t)} \mu_x(X_t, t). \end{aligned}$$

E

Impact of Step Size for Filtering

As shown in Chapter 8, the paths sampled using the analytical optimal control do not have the same weights; the reason being that we get a discretization error when we sample paths using the Euler–Maruyama scheme. Sampling paths with a smaller step size both increases the quality of the dynamics of the path, and better approximates the integral in (2.14). In Figures E.1 and E.2 we show this improvement by looking at the average ESS when filtering in one and ten spatial dimensions with step size $\Delta t = 0.005$ instead of $\Delta t = 0.01$. We also present the relative improvement in Table E.1. Note that the models have been trained with step size $\Delta t = 0.01$, yet in many cases the models increase their ESS. The effect is more significant in ten dimensions compared to one, with the CDT algorithm and analytical optimal control more than doubling their respective ESS.

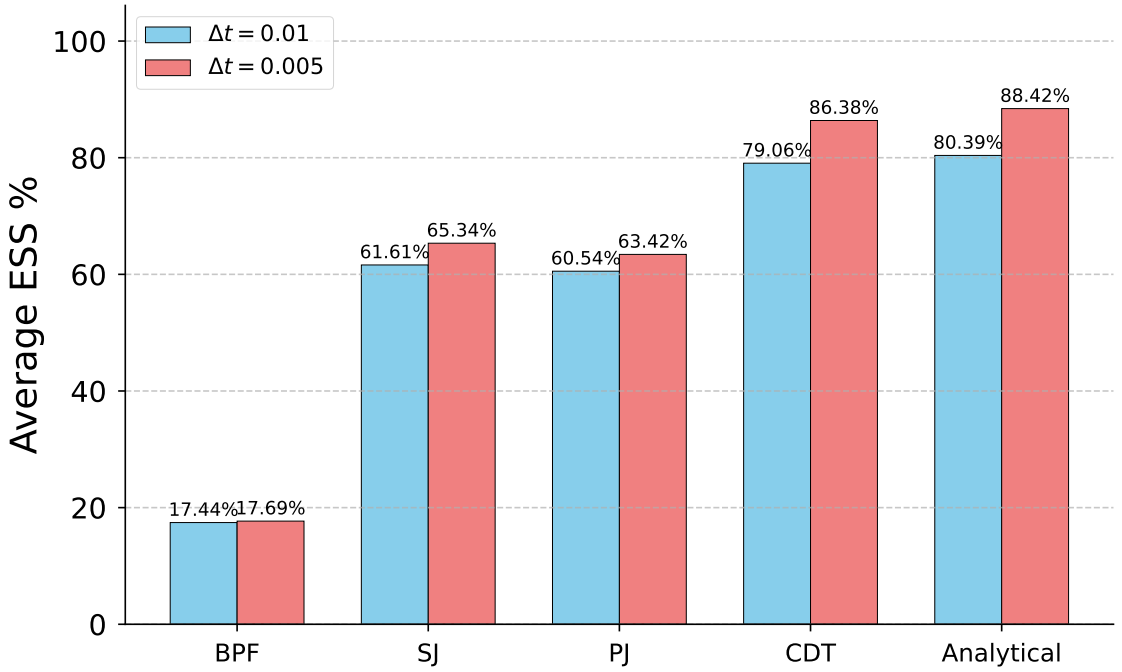


Figure E.1: Average ESS when sampling paths of different step sizes. The SDE is given by (8.1) in one dimension.

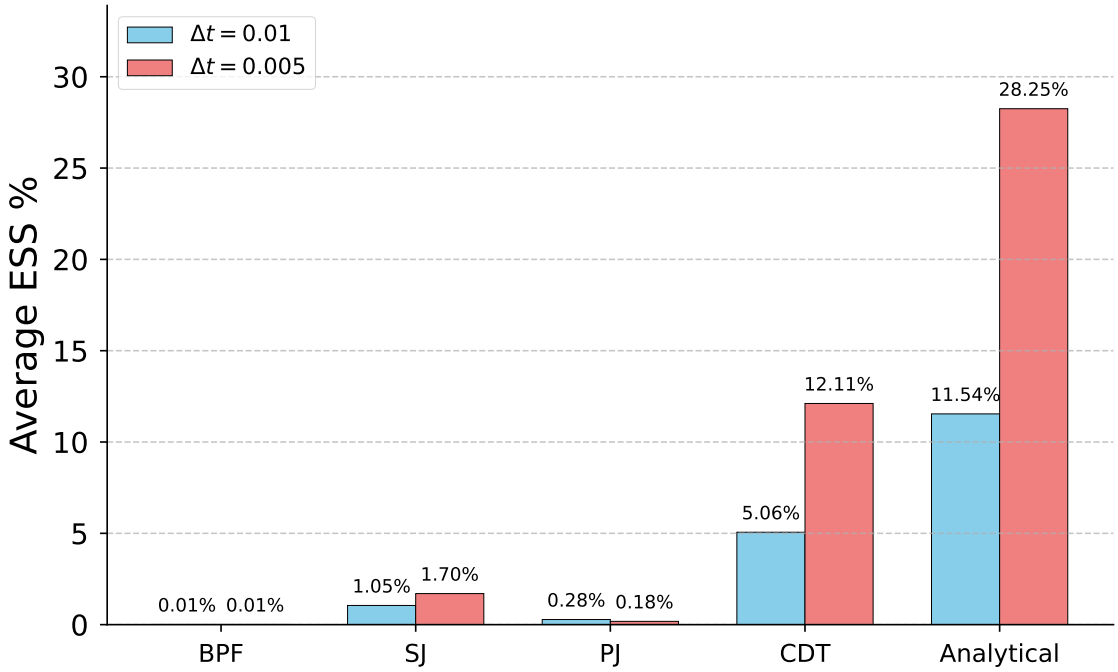


Figure E.2: Average ESS when sampling paths of different step sizes. The SDE is given by (8.1) in ten dimensions.

Table E.1: Increase in average ESS (%) for each method when using step size 0.005 compared to step size 0.01.

Method	**1 Dimension**	**10 Dimensions**
BPF	+1%	+0%
SJ	+6%	+62%
PJ	+5%	−36%
CDT	+9%	+139%
Analytical	+10%	+145%

F

Neural Network Architecture

In this appendix, we outline the neural network architectures used for the neural SDE and the critic. The purpose of this report is not to optimize the network architectures themselves; therefore, we generally employ larger models than might be strictly necessary. This approach ensures that model capacity is not a limiting factor in performance. In cases where the results are not satisfactory, we increase the architecture size either to match the complexity of the problem or to rule out insufficient capacity as a bottleneck. The neural SDE models follow a consistent architecture across problems, while we use two different architectures for the critic.

F.1 Neural SDE

The neural SDE uses the following architecture throughout all test cases.

Layer	Type	Output Size	Notes
1	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
2	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
	Dropout	–	$p = 0.2$
3	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
	Dropout	–	$p = 0.2$
4	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
	Dropout	–	$p = 0.2$
5	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
	Dropout	–	$p = 0.2$
6	Linear	d	Output layer

Table F.1: Neural network architecture used to model $\tilde{\mu}$.

Layer	Type	Output Size	Notes
1	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
2	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
	Dropout	–	$p = 0.2$
3	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
	Dropout	–	$p = 0.2$
4	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
	Dropout	–	$p = 0.2$
5	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
	Dropout	–	$p = 0.2$
6	Linear	d	Output layer
	Softplus	–	Ensures positivity

Table F.2: Neural network architecture used to model $\tilde{\sigma}$.

Layer	Type	Output Size	Notes
1	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
2	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
	Dropout	–	$p = 0.2$
3	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
	Dropout	–	$p = 0.2$
4	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
	Dropout	–	$p = 0.2$
5	Linear	128	
	LeakyReLU	–	Slope = 0.2
	LayerNorm	128	
	Dropout	–	$p = 0.2$
6	Linear	d	Output layer

Table F.3: Neural network architecture used to model $\tilde{u}$.

F.2 Critic

The two critic architectures have been chosen based on the complexity of the problem. Each has been applied across multiple test cases, and we present both here along with the cases in which they are used. The architecture applies to both path and signature critics, Φ_P and Φ_S, with the input dimensions adjusted to match the dimension of the problem. It is worth pointing out that it was found that the signature critic suffered less when the network was not deep enough, something that is not surprising given the discussion in 5.2.

F.2.1 Simple critic

Critic architecture 1 is the simplest network used in this thesis and is sufficient enough for the bimodal test case in one, its extension in two dimensions, and the linear Ornstein–Uhlenbeck process with one observation in one dimension. We also use this architecture for the Ornstein–Uhlenbeck process in ten dimensions. The SDEs are given by (7.1), (7.3), and (8.1).

Layer	Type	Output Size	Notes
1	Linear	512	
	LeakyReLU	–	Slope = 0.2
2	Linear	256	
	LeakyReLU	–	Slope = 0.2
3	Linear	128	
	LeakyReLU	–	Slope = 0.2
4	Linear	1	Output layer

Table F.4: Neural network architecture used for the simple critic.

F.2.2 Deep critic

Critic architecture 2 is a deeper network used in cases where the first architecture is not flexible enough, specifically, for the time-dependent sinusoidal dynamic and the Ornstein–Uhlenbeck process with five observations. The corresponding SDEs are given in equations (7.2) and (7.4).

Layer	Type	Output Size	Notes
1	Linear	512	
	LayerNorm	512	
	LeakyReLU	–	Slope = 0.2
2	Linear	512	
	LayerNorm	512	
	LeakyReLU	–	Slope = 0.2
3	Linear	256	
	LayerNorm	256	
	LeakyReLU	–	Slope = 0.2
4	Linear	256	
	LayerNorm	256	
	LeakyReLU	–	Slope = 0.2
5	Linear	128	
	LeakyReLU	–	Slope = 0.2
6	Linear	64	
	LeakyReLU	–	Slope = 0.2
7	Linear	1	Output layer

Table F.5: Neural network architecture used for the deep critic.

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY