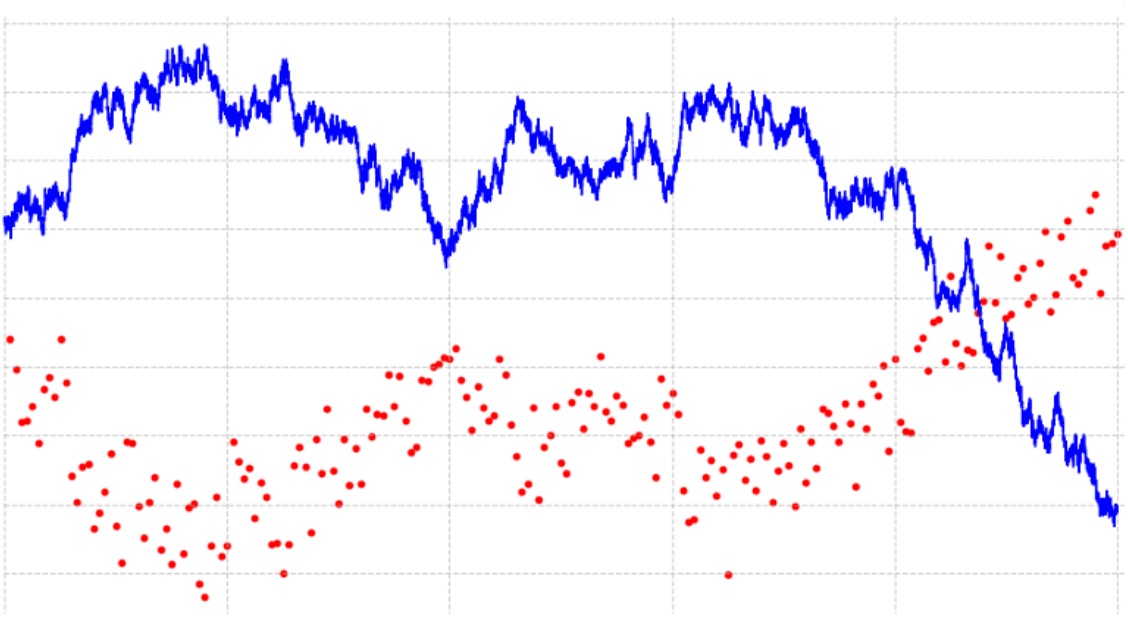

Deep Learning for the Nonlinear Filtering Problem

Adapting the Deep Backward Stochastic Differential Equation Method to Stochastic Filtering

Master's Thesis in Data Science and AI

Melker Bild

DEPARTMENT OF MATHEMATICAL SCIENCES

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

Master's thesis 2025

Deep Learning for the Nonlinear Filtering Problem

Adapting the Deep Backward Stochastic Differential Equation Method to Stochastic Filtering

Melker Bild

CHALMERS
UNIVERSITY OF TECHNOLOGY

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

Deep Learning for the Nonlinear Filtering Problem
Adapting the Deep Backward Stochastic Differential Equation Method
to Stochastic Filtering
Melker Bild

Supervisor: Moritz Schauer, Chalmers Department of Mathematical Sciences
Supervisor: Kasper Bågmark, Chalmers Department of Mathematical Sciences
Examiner: Stig Larsson, Chalmers Department of Mathematical Sciences

Master's Thesis 2025
Department of Mathematical Sciences

Chalmers University of Technology
SE-412 96 Gothenburg

Cover: An Ornstein–Uhlenbeck process (blue) and observations (red) generated from nonlinear measurement dynamics.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Deep Learning for the Nonlinear Filtering Problem
Adapting the Deep Backward Stochastic Differential Equation Method
to Stochastic Filtering
Melker Bild
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

The thesis addresses the filtering problem by estimating the probability density of a latent stochastic process given noisy observations. It does so by analysing the convergence of a novel deep learning based filter, the deep backward stochastic differential equation filter. Rooted in the connection between backward stochastic differential equations and partial differential equations, the new filter is compared against classical filters including the Kalman filter, Extended Kalman filter, and a bootstrap particle filter. While particle filters can handle nonlinear, non Gaussian systems, they suffer from the curse of dimensionality, which motivates the development of new filter approximations for nonlinear and high dimensional settings. The new method is constructed by sequentially applying a deep learning based method for backward stochastic differential equations, while incorporating observations. The thesis studies the empirical strong convergence rate of the method on both the unconditional case without observations, and the conditional case with observations. It demonstrates promising results in three examples, an Ornstein–Uhlenbeck process, a bimodal stochastic differential equation, and a 4-dimensional spring-mass system. The observed convergence rate for the first two cases is approximately $O(N^{-1/2})$, where N is number of prediction steps between observations, in accordance with the theoretical order. For the final example it is lower and this might be due to a statistical error or an approximation error.

Keywords: Nonlinear filter, Deep learning, Stochastic differential equation, Backward stochastic differential equation, Kalman filter, Particle filter, Fokker–Planck equation

Acknowledgements

First and foremost, I would like to thank my supervisors, Kasper Bågmark and Moritz Schauer, for their impeccable guidance and unwavering support. This thesis would not have been possible without Kasper developing the new filter it investigates, nor without his generosity in allowing me to analyse that method. After earning a bachelor's in Industrial Engineering and Management at Chalmers, I wrote this thesis at the Department of Mathematical Sciences, where everyone I met showed immense interest, determination, and dedication to their craft. However, it cannot be understated that my impression is mainly rooted in Kaspers's relentless pursuit of excellence, both in his own research and in the thoughtful feedback he offered me. I would like to thank Stig Larsson for his support, valuable comments, and for kindly taking on the role of examiner. I would also like to thank Chalmers e-Commons. They provided critical compute resources necessary for the analysis. Lastly, I am thankful for the insightful comments Isak Nilsson and Samuel Jakobsson provided as opponents.

Melker Bild, Gothenburg, June 2025

Contents

1
Introduction

The *filtering problem* is the task of approximating the probability density of an unknown process conditioned on all observational data obtained until the current time. The unknown process is modelled as the solution to a Stochastic Differential Equation (SDE) and the observations are obtained with a measurement function and noise. The exact solution to the filtering problem is called the *exact filtering density*, or the *Bayesian filtering density*. In some benign situations, it is possible to calculate the exact filtering density, but more often there is only an approximation available. A method used to find an approximation of the density of the state given observations is called an approximative filter. The difficulty of finding an effective approximation is mainly determined by the dimension of the state-space, type of process, the noise distribution, and the behaviour of the measurement function.

In this thesis, the empirical convergence of a new approximative Bayesian filter is studied, the Deep Backward Stochastic Differential Equation Filter (DBSDEF). This method uses deep learning and arises from the theory linking Backward Stochastic Differential Equations (BSDE) and Partial Differential Equations (PDE). The numerical convergence analysis is done by comparing the DBSDEF with established filters that are selected depending on the settting at hand. The thesis will use the Kalman filter, the extended Kalman filter, and the bootstrap particle filter. The Kalman filter solves the Bayesian filtering problem by providing the exact filtering density for linear process dynamics with additive Gaussian noise and linear measurements with additive Gaussian noise. The extended Kalman filter builds on the Kalman filter to provide an approximation of the exact filtering density for nonlinear process dynamics and non Gaussian noise. Particle filters, are a form of sequential Monte Carlo methods. They come in many variations, the simplest being the bootstrap particle filter, which is what is used in this thesis. In Figure 1.1, an example of both the Kalman filter and the bootstrap particle filter is visualised. The results from the filters represent the expected values of the underlying Ornstein–Uhlenbeck process, conditioned on the observed data. Particle filters make up for the drawback of the Kalman filter, namely they can handle nonlinear and non Gaussian dynamics, however at the cost of increased computational complexity. Additionally, as the dimension of the state increases, the computational complexity of the particle filter increases, a well-known phenomenon called the curse of dimensionality [11, 29]. There are variations of the particle filter that attempt to mitigate the curse of dimensionality by being more effective with fewer particles, *e.g.*, the Rao–Blackwell particle filter. However, there still remains a need for methods that are effective in high-dimensional nonlinear settings.

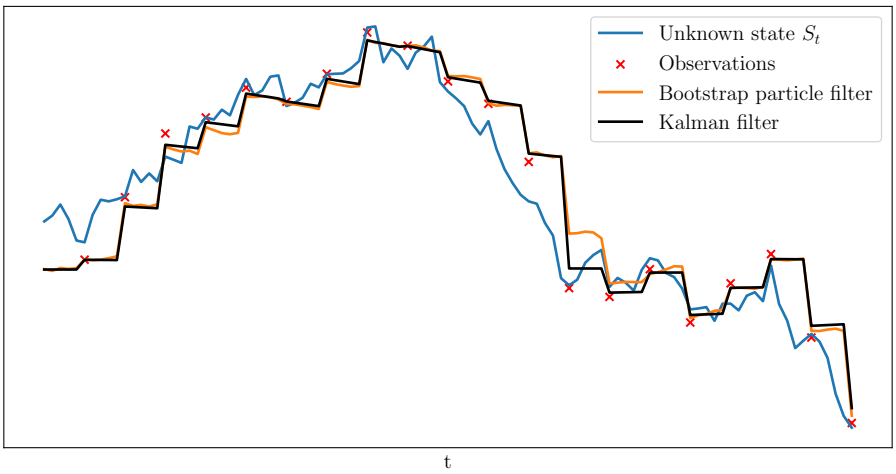

Figure 1.1: The expected values obtained from the Kalman filter and the bootstrap particle filter with 10^3 particles applied to measurements generated from an unknown state governed by an Ornstein–Uhlenbeck process, as functions of time t.

1.1 The Deep Backward Differential Stochastic Equation Filter

In the setting considered in this thesis, observations $\{\mathbb{O}_k\}_{k=1}^K$ arrive at discrete times $0 = t_0 < t_1 < t_2 \cdots < t_K = T$ from some underlying continuous process $(S_t)_{t=0}^T$, modelled as the solution to a SDE. The (marginal) probability density of the underlying state S_t, denoted p_t, is given by the solution to the partial differential equation known as the Fokker–Planck equation or the Kolmogorov forward equation. By solving a Fokker–Planck equation augmented to take the information provided by observations into account (augmented Fokker–Planck equation), it is possible to obtain the filtering distribution. To obtain the DBSDEF, the main challenge is to solve the augmented Fokker–Planck equation, a nontrivial task that becomes difficult when the underlying SDE has nonlinear dynamics and the state-dimension of S_t is high.

In the 1990s there were theoretical advancements linking the solutions of BSDEs with the solutions of PDEs [20, 37]. Under certain regularity assumptions, detailed in Chapter 3, the solution to a certain BSDE corresponds to the solution of a corresponding PDE defined by the same dynamics, and vice versa. A method from 2017 called the *deep BSDE method*, introduced in [12] uses this fact to solve certain PDEs by solving the corresponding BSDE with the help of deep learning. The deep BSDE method can therefore be used to solve the Fokker–Planck equation. By this methodology, we can obtain the probability distribution, p_T of S at the final time T. More precisely, this is done by discretizing the forward-backward pair of equations that governs the BSDE of interest. By simulating the discretized forward-backward pair of equations with the Euler–Maruyama method, one can then train a neural network that approximates p_T. The starting distribution p_0 is assumed to be known and is used as the terminal condition for the backward equation. The relevant parts

of the derivation of the Deep BSDE method is presented in Chapter 5. The Deep BSDE method has proven useful in solving many equations and converges to the true solution of the BSDE, under certain conditions [17], but to the author's knowledge, it has not yet been used in a filter.

In a concurrent work by the authors to [6], the DBSDEF is derived. It will be shown that by solving the modified Fokker–Planck equation with the deep BSDE method and normalizing its solution, an approximate Bayesian filter can be obtained. For each observation, we solve the modified Fokker–Planck equation and normalize the resulting density, applying the deep BSDE method sequentially. The approximation method is called the Deep BSDE filter since it builds on the Deep BSDE method, and it is the main object of interest in the thesis. For the nonlinear filtering problem, the DBSDEF serves as an alternative to the energy-based deep splitting method introduced by the same authors in [5].

1.2 Aims and Objectives

The aim of the thesis is to analyse the numerical convergence of the DBSDEF in linear, nonlinear and high dimensional cases. The empirical convergence of the DBSDEF is measured on three examples with varying complexity. The numerical convergence is measured against a reference filter which is selected depending on the setting.

Since the DBSDEF uses the Deep BSDE method in sequence, we first analyse the validity of this approach. We call the application of the Deep BSDE method in sequence on a setting without observations the *sub-problem approach*. Therefore, it is an intermediate objective to evaluate the convergence of the sub-problem approach. The sub-problem approach analysis is done on processes where the true density is tractable, which makes evaluation computationally inexpensive. Finally we conduct the main convergence study in the filtering setting.

1.3 Structure and Content

Besides this introductory chapter, the thesis consists of seven further chapters. Chapter 2 introduces preliminary theory on state-space models, stochastic differential equations, the Fokker–Planck equation, numerical methods and neural networks. For readers already familiar with some or all of these topics, this chapter may serve primarily to introduce and standardize the notation used throughout the rest of the thesis. Chapter 3 continues with an introduction to backward stochastic differential equations and their relationship to partial differential equations. Many details and proofs are omitted for brevity, but appropriate references are provided for the interested reader. Chapter 4 addresses the filtering problem and presents the Kalman filter, the extended Kalman filter, and the bootstrap particle filter, all of which are later used in the comparison with the DBSDEF. Chapter 5 derives the DBSDEF by first introducing the deep BSDE method and the sub-problem approach. It also

discusses key aspects of training the neural networks. Chapter 6 details the implementation and evaluation of the DBSDEF. It also explains many practical choices made in the implementation and discusses how the method could be developed to improve. Chapter 7 presents the results, beginning with a empirical convergence analysis of the sub-problem approach. This is followed by three examples to measure the convergence of the DBSDEF for filtering on an Ornstein–Uhlenbeck process, a bimodal problem, and a spring-mass system. The thesis concludes with a brief summary in Chapter 8. An additional example of the sub-problem approach with few training samples is included in the Appendix A.

1.4 Applications of Filtering

The filtering problem arises in many different fields, making the application of filters widespread. To highlight the importance of developing efficient filters, several applications are briefly mentioned below, many of which are in domains where it is critical for the filter to provide a reliable solution, such as in medicine.

- **Signal Processing**: Filtering enhances signal quality in audio, imaging, and telecommunications. Noise reduction improves speech in hearing aids, denoising sharpens medical images like MRIs, and adaptive filters reduce interference in data transmission [26].

- **Control Systems**: Filtering aids state estimation and stabilization in robotics, aerospace, and automation. Kalman filters estimate vehicle positions, stabilize aircraft controls, and ensure smooth machinery operation [19].

- **Navigation and Tracking**: Filtering supports GPS, radar, and augmented reality. Extended Kalman filters fuse sensor data for accurate positioning, radar tracks targets, and AR systems monitor user movements [16].

- **Finance**: Filtering models market dynamics. Particle filters estimate volatility for risk assessment, optimize portfolios, and process data for high-frequency trading [1].

- **Robotics**: Filtering enables sensor fusion, localization, and path planning. It combines LIDAR and camera data, supports SLAM for mapping, and predicts obstacle paths [34].

- **Biomedical Engineering**: Filtering improves physiological signal analysis. It removes noise from ECG/EEG, monitors vital signs in wearables, and reduces artifacts in medical imaging [28].

- **Environmental Monitoring**: Filtering processes sensor data for weather and pollution studies. It predicts atmospheric conditions, analyses climate trends, and tracks pollutants [9].

2
Preliminaries

This chapter presents the preliminary theory necessary to understand the forthcoming chapters on backward stochastic differential equations and Bayesian filtering, two concepts critical to understand the Deep BSDE filter. First, we introduce the general state-space model used in the thesis. Then, stochastic differential equations and the Fokker–Planck equation are discussed. Next, the numerical methods Euler–Maruyama and the Monte Carlo method are explored, which are used to simulate SDEs and to approximate expectations for metrics as well as in the learning of the Deep BSDE Filter. Lastly, a short introduction is given on neural networks and a motivation on why they are suitable function approximators for the minimization problem in the Deep BSDE Filter.

2.1 The State-Space Model

Central to the thesis is the underlying stochastic state-space model which consists of a continuous stochastic process $(X_t)_{t\in[0,T]}$ and a discrete observation process $\{\mathbb{O}_k\}_{k=1}^K$. In the general treatment that follows, we denote the continuous stochastic process by X. However, it is also referred to as S because the method requires two different continuous processes. If nothing else is said, we will throughout the thesis let $d, d', T, K, N \in \mathbb{Z}^+$ be constants, let $(\Omega, \mathcal{F}, \mathbb{P})$ be a complete probability space with a filtration $\mathbb{F} = (\mathcal{F}_t)_{t\in[0,T]}$ that is right-continuous and augmented with null sets and let $X_t \in L^2(\Omega; \mathbb{R}^d)$, for $t \in [0, T]$, be a probabilistic process adapted to the filtration $\mathbb{F}$. Given $\omega \in \Omega$, we denote the sample path of the stochastic process by $x_t = X_t(\omega)$, $t \in [0, T]$. Additionally, we assume that $(X_t)_{t\in[0,T]}$ has the Markov property.

Definition 2.1 (Markov property)**.**
A stochastic process $\{X_t\}_{t\in[0,T]}$ is said to have the Markov property if, for all $s \in [0, t]$ and for any bounded and measurable function g

$$\mathbb{E}[g(X_t) \mid \mathcal{F}_s] = \mathbb{E}[g(X_t) \mid X_s]. \tag{2.1}$$

Equivalently, in terms of conditional probabilities, for any measurable set A

$$\mathbb{P}(X_t \in A \mid \mathcal{F}_s) = \mathbb{P}(X_t \in A \mid X_s). \tag{2.2}$$

The Markov property is vital in nearly all results presented, for instance the Fokker–Planck equation [22], the Bayesian filtering equations and in the Kalman filter [32]. We assume that X and S have the Markov property throughout the thesis.

To fix notation for observations and the continuous stochastic process X, we introduce two partitions. Let $0 = t_0 < t_1 < t_2 \cdots < t_K = T$, be a grid of observation times and let $t_k = t_{k,0} < t_{k,1} < t_{k,2} \cdots < t_{k,N} = t_{k+1,0} = t_{k+1}$ be a partition between each observation, and $\Delta t = t_{k,n+1} - t_{k,n}$ be constant for all k, n. It then makes sense to define the observation process as a discrete-time stochastic process $\mathbb{O} : \{1, 2, \dots K\} \times \mathbb{R}^d \to \mathbb{R}^{d'}$ adapted to the filtration $\mathbb{F}$. Figure 2.1 illustrates the Markovian evolution of X and its corresponding observations, $\mathbb{O}$.

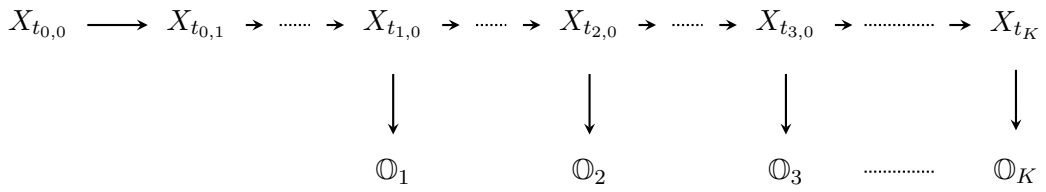

Figure 2.1: State-space model of X with observations $\mathbb{O}$

As we will see in the coming Section 2.4, the finer partition between each observation $t_{k,0} ... t_{k,N}$ will allow simulation of the stochastic process X with low discretization error. Finally, there are many applications of state-space models, many of which are in financial time series, see for instance Chapter 6 in [33].

2.2 Stochastic Differential Equations

Stochastic differential equations expands on the theory of ordinary differential equations by incorporating a stochastic term, an Itô integral driven by a Brownian motion. Define measurable functions $\mu : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ and $\sigma : [0, T] \times \mathbb{R}^d \to \mathbb{R}^{d \times d}$ such that the covariance matrix $\sigma(t, x)\sigma^\top(t, x)$ is positive semi-definite for all $t \in [0, T]$ and $x \in \mathbb{R}^d$. The equations of the form

$$\mathrm{d}X_t = \mu(t, X_t)\,\mathrm{d}t + \sigma(t, X_t)\,\mathrm{d}W_t, \quad t \in [0, T], \tag{2.3}$$

are called stochastic differential equations. Let X_0 be distributed according to some probability distribution p_0. A more meaningful representation of the SDE (2.3) is in its corresponding integral form

$$X_t = X_0 + \int_0^t \mu(s, X_s)\,\mathrm{d}s + \int_0^t \sigma(s, X_s)\,\mathrm{d}W_s, \quad t \in [0, T]. \tag{2.4}$$

There are two different types of solutions to the SDE (2.4), weak and strong solutions. A weak solution is a stochastic process $(\widehat{X}_t)_{t \in [0,T]}$ that satisfies (2.4) for *some* Brownian motion $(W_t)_{t \in [0,T]}$ and is adapted to the filtration generated by $(W_t)_{t \in [0,T]}$.

A strong solution is a stochastic process $(X_t)_{t\in[0,T]}$ adapted to the filtration generated by given Brownian motion $(W_t)_{t\in[0,T]}$ that satisfies (2.4).

Together with the Lipschitz condition, linear growth and initial bound that we state next, a strong solution is guaranteed.

Theorem 2.2.1 (Strong solution)**.** *Let $\|\cdot\|_F$ be the Frobenius matrix norm. Assume that for the functions $\mu : [0,T]\times\mathbb{R}^d \to \mathbb{R}^d$ and $\sigma : [0,T]\times\mathbb{R}^d \to \mathbb{R}^{d\times d}$ there exists constants $K, C > 0$ such that for all $x, y \in \mathbb{R}^d$ and $t \in [0,T]$ we have*

$$\|\mu(t,x) - \mu(t,y)\| + \|\sigma(t,x) - \sigma(t,y)\|_F \leq K\|x-y\|, \tag{2.5}$$

$$\|\mu(t,x)\| + \|\sigma(t,x)\|_F \leq C(1+\|x\|), \tag{2.6}$$

$$\mathbb{E}[\|X_0\|^2] < \infty. \tag{2.7}$$

Then there exists a unique strong solution to the SDE (2.4)*.*

The result can be found in Theorem 1 in [14]. A common simplification, which is relevant for this thesis, is to assume that the functions μ and σ do not explicitly depend on the time t, making the SDE (2.8) time-homogeneous

$$X_t = X_0 + \int_0^t \mu(X_s)\,\mathrm{d}s + \int_0^t \sigma(X_s)\,\mathrm{d}W_s, \quad t\in[0,T]. \tag{2.8}$$

A result crucial enough to occasionally rename the topic of stochastic calculus to Itô calculus comes from the theorem known as Itô's formula. The theorem is essential in the derivation of the Deep BSDE method and is presented next.

Theorem 2.2.2 (Itô's formula)**.** *Let $f \in C^{1,2}([0,T]\times\mathbb{R}^d;\mathbb{R})$, and let $(X_t)_{t\in[0,T]}$ be the strong solution to the time-homogeneous SDE* (2.8)*, then*

$$\begin{aligned} f(t,X_t) =& f(0,X_0) + \int_0^t \left(\frac{\partial f}{\partial s}(s,X_s) + \mu(X_s)\nabla_x f(s,X_s) + \frac{1}{2}\sigma(X_s)\sigma^\top(X_s)\nabla_x^2 f(s,X_s)\right)\mathrm{d}s \\ &+ \int_0^t \nabla_x f(s,X_s)\sigma(X_s)\,\mathrm{d}W_s, \quad t\in[0,T]. \end{aligned}$$

The proof can be found in [18]. In the context of the Deep BSDE method, it is likely more worthwhile to develop an intuition for the theorem, for instance by reading Section 3.2 in [27].

2.3 The Fokker–Planck Equation

A special partial differential equation that is important to the Deep BSDE filter method and that naturally appears as a consequence of Itô's formula is the Fokker–Planck equation, also known as the Kolmogorov forward equation. In loose terms,

the solution to the Fokker–Planck equation is the transition density of the weak solution $(\widehat{X}_t)_{t\in[0,T]}$ of (2.4). All strong solutions are by definition weak solutions which makes the Fokker–Planck equation also useful for describing strong solutions. For this thesis, the Fokker–Planck equation will mainly be used for the probability density of x_t given an initial distribution at $t = 0$, denoted $p(X_t = x_t \mid X_0 = x_0)$. Below is an outline of the derivation of the Fokker–Planck equation that starts with defining the infinitesimal generator.

Definition 2.2 (Infinitesimal generator)**.** *Let $(X_t)_{t\in[0,T]}$ be a stochastic process with the Markov property from Definition 2.1 and let $f : \mathbb{R}^d \to \mathbb{R}$ be a measurable function. The infinitesimal generator of X acting on f is defined by the following limit in function space*

$$[\mathcal{L}_t f](x) = \lim_{h\to 0^+} \frac{1}{h}\Big[\mathbb{E}[f(X_{t+h}) \mid X_t = x] - f(x)\Big], \quad t \in [0, T). \tag{2.9}$$

For the remainder of this section we let $t \in [0, T)$. Abbreviating the density $\frac{p(X_{t+h}=dy|X_t=x)}{dy} = p_{t+h,t}(y, x)$ and using the definition of the conditional expectation we rewrite the infinitesimal generator in (2.9) as

$$[\mathcal{L}_t f](x) = \lim_{h\to 0^+} \frac{1}{h}\Big[\int_{\mathbb{R}^d} f(y)p_{t+h,t}(y, x)\,\mathrm{d}y - f(x)\Big]. \tag{2.10}$$

Let $s \in [0, t)$ and by multiplying (2.10) by the transition density $p_{t,s}(x, z)$ and integrating over x we obtain

$$\begin{aligned}&\int_{\mathbb{R}^d} [\mathcal{L}_t f](x)p_{t,s}(x, z)\,\mathrm{d}x\\ &= \lim_{h\to 0^+} \frac{1}{h}\Big[\int_{\mathbb{R}^d}\int_{\mathbb{R}^d} f(y)p_{t+h,t}(y, x)p_{t,s}(x, z)\,\mathrm{d}y\,\mathrm{d}x - \int_{\mathbb{R}^d} f(x)p_{t,s}(x, z)\,\mathrm{d}x\Big].\end{aligned} \tag{2.11}$$

By the Fubini–Tonelli theorem, the order of integration can be changed. The Chapman–Kolmogorov equation can then be applied to get the simplification

$$\int_{\mathbb{R}^d} p_{t+h,t}(y, x)p_{t,s}(x, z)\,\mathrm{d}x = p_{t+h,s}(y, z). \tag{2.12}$$

Using (2.12) in (2.11) we have

$$\int_{\mathbb{R}^d} [\mathcal{L}_t f](x)p_{t,s}(x, z)\,\mathrm{d}x = \lim_{h\to 0^+} \frac{1}{h}\Big(\int_{\mathbb{R}^d} f(y)p_{t+h,s}(y, z)\,\mathrm{d}y - \int_{\mathbb{R}^d} f(x)p_{t,s}(x, z)\,\mathrm{d}x\Big). \tag{2.13}$$

Below we replace the variable y with x and obtain the partial derivative of the density $p_{t,s}(x, z)$ on the right hand side

$$\lim_{h\to 0^+} \frac{1}{h}\Big(\int_{\mathbb{R}^d} f(x)p_{t+h,s}(x, z)\,\mathrm{d}y - \int_{\mathbb{R}^d} f(x)p_{t,s}(x, z)\,\mathrm{d}x\Big) = \int_{\mathbb{R}^d} f(x)\frac{\partial}{\partial t}p_{t,s}(x, z)\,\mathrm{d}x. \tag{2.14}$$

By recalling the left hand side of (2.13) and by letting $\mathcal{L}^*$ be the adjoint operator of $\mathcal{L}$, it is then apparent that we have achieved the following

$$\int_{\mathbb{R}^d} [\mathcal{L}_t f](x) p_{t,s}(x,z)\,\mathrm{d}x = \int_{\mathbb{R}^d} f(x)\mathcal{L}_t^*[p_{t,s}](x,z)\,\mathrm{d}x = \int_{\mathbb{R}^d} f(x)\frac{\partial}{\partial t}p_{t,s}(x,z)\,\mathrm{d}x, \quad (2.15)$$

which contains the Fokker–Planck equation, classically formulated like

$$\frac{\partial}{\partial t}p_{t,s}(x,z) = \mathcal{L}_t^*[p_{t,s}](x,z). \quad (2.16)$$

If nothing else is said, we will let $s = 0$ and $x_0 \sim p_0$. To ease future notation, we state the known result that every time-homogenous SDE can be fully described by its infinitesimal generator $\mathcal{L}$, and that $\mathcal{L}$ and $\mathcal{L}^*$ can be reformulated as

$$\begin{aligned} \mathcal{L}[f](x) &= \sum_{i=1}^{d} \mu_i(x)\frac{\partial f(x)}{\partial x_i} + \frac{1}{2}\sum_{i,j=1}^{d} a_{i,j}(x)\frac{\partial^2 f(x)}{\partial x_i \partial x_j}, \\ \mathcal{L}^*[f](x) &= -\sum_{i=1}^{d}\frac{\partial}{\partial x_i}\left(\mu_i(x)f(x)\right) + \frac{1}{2}\sum_{i,j=1}^{d}\frac{\partial^2}{\partial x_i \partial x_j}\left(a_{i,j}(x)f(x)\right), \end{aligned} \quad (2.17)$$

with $a(x) = \sigma(x)\sigma^\top(x)$. The proof of (2.17) includes the use of Itô's formula and an outline of it can be found on pages 110–111 in [27].

2.4 Numerical Methods

Under the conditions of Theorem 2.2.1, we have seen that SDE in (2.4) has a unique strong solution $(X_t)_{t\in[0,T]}$, which is difficult to obtain explicitly in practice. Hence, the solution will be approximated. In this section we therefore present the Euler–Maruyama method that is used for the simulation of the SDEs followed by a concise description of the Monte Carlo method, which is useful in approximating expectations.

First, we see that (2.4) is equivalent to

$$X_{t_{k,n+1}} = X_{t_{k,n}} + \int_{t_{k,n}}^{t_{k,n+1}} \mu(s, X_s)\,\mathrm{d}s + \int_{t_{k,n}}^{t_{k,n+1}} \sigma(s, X_s)\,\mathrm{d}W_s. \quad (2.18)$$

For $t \in \{t_{0,0} \dots t_{K,N}\}$ an approximation $\{\widetilde{X}_t\}_{t=0}^T$ of X in (2.18) is constructed by an explicit time approximation, yielding the following definition.

$$\begin{aligned} &\widetilde{X}_{0,0} = X_0, \\ &\widetilde{X}_{k,n+1} = \widetilde{X}_{k,n} + \mu(t_{k,n}, \widetilde{X}_{k,n})\int_{t_{k,n}}^{t_{k,n+1}} \mathrm{d}s + \sigma(t_{k,n}, \widetilde{X}_{k,n})\int_{t_{k,n}}^{t_{k,n+1}} \mathrm{d}W_s. \end{aligned} \quad (2.19)$$

We see that the recursive step simplifies and we get

$$\begin{aligned}\widetilde{X}_{0,0} &= X_{0,0},\\ \widetilde{X}_{k,n+1} &= \widetilde{X}_{k,n} + \mu(t_{k,n}, \widetilde{X}_{k,n})\Delta t + \sigma(t_{k,n}, \widetilde{X}_{k,n})\Delta W_{k,n},\end{aligned} \tag{2.20}$$

with $\Delta W_{k,n} = W_{k,t_{n+1}} - W_{k,t_n} \sim \mathcal{N}(0, I_d \Delta t)$ which in practice is sampled by a pseudo random number generator.

The Euler–Maruyama method has a known convergence rate, which we state next. For complete proofs, see Section 9.6 and 9.7 in [23].

Theorem 2.4.1 (Strong convergence of the Euler–Maruyama method)**.** *Let μ and σ satisfy the the Lipschitz condition, the linear growth condition and let $\mathbb{E}[\|X_0\|^2] < \infty$. Then let $(X_t)_{t\in[0,T]}$ be the solution to (2.4) and let $\{\widetilde{X}_t\}_{t=t_{0,0}}^{t_{K,N}}$ be given by (2.20). Assume for a positive constant C_1 independent of Δt that we have*

$$\mathbb{E}[|\widetilde{X}_{0,0} - X_{t_{0,0}}|^2] < C_1 \Delta t. \tag{2.21}$$

Then there exists a positive constant C_2, independent of Δt, such that for all $k = 0, \dots K$, $n = 0, \dots, N$, $k + n \neq 0$

$$\mathbb{E}[|\widetilde{X}_{k,n} - X_{t_{k,n}}|] \leq C_2 (\Delta t)^{\frac{1}{2}}. \tag{2.22}$$

Clearly, by letting $\Delta t \to 0$, the error term in (2.22) converges from above to 0. The error is fittingly called *discretization error* as it depends on how granular the discretization is.

Since, the result from Equation (2.22) holds in expectation it does not guarantee convergence for any single simulation path. Therefore the strong error is calculated numerically for many samples by the Monte Carlo method. More specifically, the Monte Carlo method can be used to approximate integrals of the form

$$\mathbb{E}[g(X) \mid y_{1:k}] = \int g(x) p(x \mid y_{1:k}) \, \mathrm{d}x, \tag{2.23}$$

where $g : \mathbb{R}^d \to \mathbb{R}$ is an arbitrary function, and $p(X \mid y_{1:k})$ is the probability density of the state X given measurements $y_{1:k}$. The expectation in Equation (2.23) is then approximated by sampling M independent random samples $x^{(i)} \sim p(X \mid y_{1:k})$, $i \in \{1 \dots M\}$ and calculating

$$\mathbb{E}[g(X) \mid y_{1:k}] \approx \frac{1}{M} \sum_{i=1}^{M} g(x^{(i)}). \tag{2.24}$$

The approximation in (2.24) can be related to Theorem 2.4.1 by letting the function g be such that the expectation in (2.22) is obtained, and it can be shown that the convergence rate of the approximation in (2.24) is $O(M^{-1/2})$ [32]. Lastly, the error from the approximation (2.24) is called the *statistical error* and the sum of the discretization error and the statistical error is called the *total error*. For more theory on error rates in numerical methods, the reader is referred to [15].

2.5 Neural Networks

Traditionally, parametrized models have fewer parameters than needed to fully represent the object they aim to approximate, and are therefore less complex. This is intended to avoid the well-known phenomenon called overfitting, where the model becomes too specific to the training data and loses the ability to generalize to unseen data. By this traditional logic, one might expect that over-parametrized models, *i.e.*, models with far more parameters than necessary would underperform and overfit. However, in some cases, over-parametrized models have shown surprisingly good performance and generalizability, a phenomena called double descent. A family of over-parametrized models that is particularly important for this thesis, and has seen an explosion in popularity, is Neural Networks (NNs). The motivation for using Neural Networks is largely empirical, with limited formal theory. In what follows, we introduce the Fully Connected Neural Network (FCNN), which is used in this thesis, followed by a short discussion of the topic.

Neural networks can be viewed as a combination of a feature extractor and a linear predictor. They consist of an input layer, one or more hidden layers, and an output layer, each composed of neurons. A typical illustration of a neural network is shown in Figure 2.2. Let $N^{(l)}$, where $l \in \{1, \dots, L\}$, denote the number of neurons in layer l, then the number of hidden layers is $L-2$ for $L \geq 3$. Each neuron implements a parametrized function $f_{w_{ij}^{(l)}, b_i^{(l)}} : \mathbb{R}^\ell \to \mathbb{R}$, where $w_{ij}^{(l)}$ are the weights, $b_i^{(l)}$ are the biases, and ℓ is the number of outputs from the previous layer. In a fully connected neural network (FCNN), we have $\ell = N^{(l-1)}$, meaning that each neuron in layer l is connected to all neurons in the previous layer, hence the term *fully connected*.

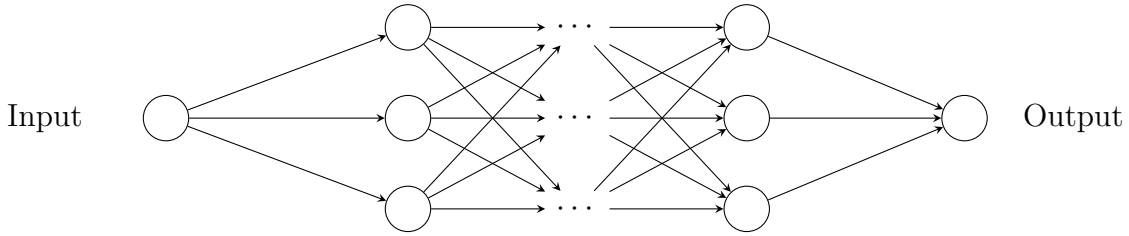

Figure 2.2: A FCNN with one input and one output neuron

If we let $x_i^{(l-1)}$ be the output of neuron i in the previous layer $l-1$ and w_{ij}^l be the weight between neuron i in layer $l-1$ and neuron j in layer l and let b_j^l be its bias. Then the output of neuron j in layer l, denoted $x_j^{(l)}$ is given by

$$x_j^{(l)} = \phi\left(\sum_{i=0}^{N^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)} + b_j^{(l)}\right), \quad i \in \{1, \dots, N^{(l-1)}\}, j \in \{1, \dots, N^{(l)}\}, \tag{2.25}$$

$$l \in \{1, \dots, L-1\}, \tag{2.26}$$

$$\tag{2.27}$$

$$x_j^{(l)} = \sum_{i=0}^{N^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)} + b^{(l)}, \quad i \in \{1, \dots, N^{(l-1)}\}, j \in \{1, \dots, N^{(l)}\},\ l = L, \quad (2.28)$$

with $\phi : \mathbb{R} \to \mathbb{R}$ being some nonlinear function. Each layer then becomes a function $F^{(l)} : \mathbb{R}^{N^{(l-1)}} \to \mathbb{R}^{N^{(l)}}$ and the entire neural network $\mathcal{N}$ is the composition of each layer $\mathcal{N} : \mathbb{R}^d \to \mathbb{R}^{N^{(L)}}$, where d is the dimension of the input $x^{(0)}$. As exemplified by (2.28), the function ϕ is typically not applied in the output layer, but this is not strictly the case.

The representational power of neural networks is guaranteed by Barron's Universal Approximation Theorem [3], which we state next.

Theorem 2.5.1 (Barron's Universal Approximation Theorem)**.** *Let $f \colon \mathbb{R}^d \to \mathbb{R}$ and define its Fourier transform as*

$$\hat{f}(\omega) = \int_{\mathbb{R}^d} f(x) e^{-i\omega^\top x}\, dx.$$

Assume that

$$\int_{\mathbb{R}^d} |\omega|\, |\hat{f}(\omega)|\, d\omega \leq C.$$

Then for all $C > 0$, $n \geq 1$ and all $r > 0$, there exists a function f_n with parameters $c_j, b_j, w_j, c_0 \in \mathbb{R}$ for $j = 1, \dots, N$

$$f_n(x) = \sum_{j=1}^{n} c_j \phi(x^\top w_j + b_j) + c_0,$$

where ϕ is a sigmoid like activation function, such that

$$\int_{|x| \leq r} \left(f(x) - f_n(x)\right)^2 dx \leq \frac{(2Cr)^2}{n}.$$

A formal proof of 2.5.1 is not provided here but can be found as Theorem 1 in [3]. However, informal visual proofs might be more useful for the intuition of neural networks, for instance the one in [25]. It is essential that the function ϕ is nonlinear to fulfil the conditions for Theorem 2.5.1. Without the nonlinearity, the neural network looses its expressive power as it would only model linear functions, regardless of neuron and layer count and could be represented by a single matrix. Common choices of nonlinear activation functions are the Rectified Linear Unit (ReLU), Gaussian error linear unit and the sigmoid function, but the architecture used in this thesis will have the ReLU as its activation function.

We denote the network parametrized by $\theta = [w^{(1)}, \dots, w^{(L)}, b^{(1)}, \dots, b^{(L)}]$ as $\mathcal{N}_\theta$. The parameters θ are learned by minimizing some objective or loss function. The loss function relevant for the Deep BSDE formulation is the Mean Squared Error (MSE) loss function, which is defined as

$$\mathcal{L}(\mathcal{N}_\theta) = \frac{1}{2N} \sum_{n=1}^{N} \|y_n - \mathcal{N}_\theta(x_n)\|^2, \quad (2.29)$$

for a labelled dataset $\{(x_n, y_n)\}_{n=1}^{N}$. The network parameters θ are then optimized using gradient based methods to minimize the loss

$$\min_{\theta} \mathcal{L}(\mathcal{N}_\theta). \tag{2.30}$$

There are many ways to solve for the objective (2.30), the most well-known being stochastic gradient descent and the ADAptive Moment estimation (ADAM) optimizer. The ADAM optimizer is used in this thesis because it offers faster and more stable convergence than many alternatives. It achieves this by not only using the gradient of the loss function but also incorporating estimates of the first- and second-order moments of the gradients. These momentum terms help smooth the optimization trajectory, resulting in more stable and efficient convergence [21]. With $\eta \in \mathbb{R}^+, \beta_1, \beta_2 \in [0, 1]$ being hyper parameters for learning rate, exponential decay rate for the first and second moment estimate, a gradient update step in ADAM is defined by

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta \mathcal{L}(\theta_t), \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2)\nabla_\theta \mathcal{L}(\theta_t)^2, \\ \hat{m}_t &= \frac{m_t}{1 - \beta_1^t}, \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t}, \\ \theta_{t+1} &= \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \end{aligned} \tag{2.31}$$

for some small $\epsilon > 0$. The gradient $\nabla_\theta \mathcal{L}(\theta_t)$ can be approximated using a single random data sample, the entire dataset, or a batch of randomly selected data points. The first option, using a single sample, is typically not preferred due to high variance and inefficiency. The second option, using the full dataset can be computationally expensive and may slow convergence. The third approach, known as *mini-batch* gradient descent, is a compromise of the two. With a carefully chosen batch size, it offers several advantages, including more stable convergence and efficient use of computational resources. By experimenting with different batch sizes, it is possible to find a good trade-off between the number of gradient updates and the computational cost per update during neural network training. This is typically an empirical exercise carried out iteratively, often during tuning other hyper parameters such as the learning rate and weight decay. Selecting batch size is also highly dependent on the hardware. A GPU greatly benefits from larger batch sizes, whereas a CPU does not see the same advantage. However increasing batch size indefinitely is impossible since it is limited by the available short-term memory, such as GPU VRAM or the system's main RAM.

In many applications of machine learning, a dataset is split into three subsets: a training set, a validation set, and a test set. The training set is used to train the model, the validation set is used to tune hyperparameters and monitor performance during training, and the test set is used for the final evaluation of the trained model. It is important that the model is not trained on the validation or test data, as doing

so would defeat their purpose and it would lead to a wrong error estimate. In this thesis, the data is synthetically generated, which is computationally inexpensive. Therefore, new data is generated before each training epoch, eliminating the need for a separate validation set.

As mentioned in the beginning of this section, the training of neural networks is highly empirical and numerous methods have been developed to have faster convergence and accuracy without overfitting, for instance dropout, skip-connections, regularization and batch normalization. In real-world applications, data augmentation is used to improve generalizability and protect against adversarial attacks, sometimes using purely synthetic data, which is not entirely different from this thesis since the data is synthetically generated. Lastly, it must be mentioned that [35] introduced a type of neural network called *Transformers*, which have performed exceptionally well by by leveraging self-attention mechanisms to model dependencies and has been ground-breaking in real-world applications such as autonomous cars.

3

Backward Stochastic Differential Equations

Backward stochastic differential equations are closely connected to the SDEs treated in Section 2.2 and here we describe their relation to partial differential equations, including the Fokker–Planck equation. We will now present parts of the BSDE formulation as can be found in [20]. We recall the SDE setting from Section 2.2

$$\begin{aligned} \mathrm{d}X_t &= \mu(t, X_t)\,\mathrm{d}t + \sigma(t, X_t)\,\mathrm{d}W_t, \quad t \in (0, T], \\ X_0 &= x, \end{aligned} \tag{3.1}$$

for $x \in \mathbb{R}^d$. Then if the conditions of Theorem 2.2.1 are fulfilled, there exists a unique strong solution of the SDE (3.1) which we denote $(X_t^{(x)}, 0 \leq t \leq T)$. The associated BSDE formulation is then

$$\begin{aligned} -\,\mathrm{d}Y_t &= f(t, X_t^{(x)}, Y_t, Z_t)\,\mathrm{d}t - Z_t^*\,\mathrm{d}W_t, \quad t \in [0, T), \\ Y_T &= \Psi(X_T^{(x)}), \end{aligned} \tag{3.2}$$

where Z^* denotes the adjoint matrix of Z. We will denote the solution to (3.2) by the pair $\{(Y_t^{(x)}, Z_t^{(x)}), 0 \leq t \leq T\}$, where f, Ψ are Borel measurable functions taking values in $\mathbb{R}^{\tilde{d}}$, for some constant $\tilde{d} \in \mathbb{Z}^+$. Furthermore, the function f is called the generator of (3.2) and $\Psi(X_T^{(x)})$ is called the terminal condition of (3.2).

Besides the already established assumptions on μ, σ in Theorem 2.2.1, further assumptions are needed to guarantee uniqueness and existence of a solution to (3.2). Let $\mathbb{L}_T^2(\mathbb{R}^d)$ be the space of all $\mathcal{F}_T$ measurable random variables $X : \Omega \to \mathbb{R}^d$ satisfying $\|X\|^2 = \mathbb{E}[\|X\|^2] < +\infty$ and $\mathbb{H}_T^2$ denote the space of all mean-square continuous, predictable processes. Assume $f(\cdot, \cdot, 0, 0) \in \mathbb{H}_T^2(\mathbb{R}^d)$ and $\Psi(X_T^{(x)}) \in \mathbb{L}_T^2(\mathbb{R}^d)$ and for all $x, y, y_1, y_2, z, z_1, z_2 \in \mathbb{R}^d$ that there exists constants $C, K > 0$ such that

$$\begin{aligned} \|f(t, x, y_1, z_1) - f(t, x, y_2, z_2)\| &\leq C\Big(\|y_1 - y_2\| + \|z_1 - z_2\|\Big), \\ \|f(t, x, y, z)\| + \|\Psi(x)\| &\leq C\Big(1 + \|x\|^p\Big), \end{aligned} \tag{3.3}$$

for real valued $p \geq \frac{1}{2}$. Then, f and $\Psi(x)$ are called standard parameters. With these assumptions, Theorem 2.1 in [20] uses the Burkholder–Davis–Gundy inequality and a fixed point argument to prove the existence and uniqueness of the solution $\{(Y_t^{(x)}, Z_t^{(x)}), 0 \leq t \leq T\}$ to (3.2). Also, the stated assumptions (3.3) on the parameters of the BSDE (3.2) implies regularity properties on the solution

$\{(Y_t^{(x)}, Z_t^{(x)}), 0 \leq t \leq T\}$. The details are left out for brevity as the proof can be found in Proposition 4.1 in [20].

3.1 The Process Y

Looking at Equation (3.2) it might not be apparent that the solution $Y_t^{(x)}$ is adapted to the filtration $\mathcal{F}_t$. To see this, we first need the Theorem 6.1 from [13].

Theorem 3.1.1 (Martingale representation theorem)**.** *Let W_t be a Brownian motion on the the complete probability space $(\Omega, \mathbb{F}, \mathbb{P})$ with the augmented natural filtration $\mathbb{F} = (\mathcal{F}_t)_{t\in[0,T]}$. Let $(M_t)_{t\in[0,T]}$ be a squared integrable $(\mathcal{F})$-martingale, then there exists a adapted and predictable process $(C_t)_{t\in[0,T]}$ such that*

$$M_t = M_0 + \int_0^t C_s \, \mathrm{d}W_s. \tag{3.4}$$

We temporarily assume that the generator f does not depend on Y_t and Z_t and define the Martingale M_t as

$$M_t := \mathbb{E}\left[\Psi(X_T^{(x)}) + \int_0^T f(s, X_s) \, \mathrm{d}s \mid \mathcal{F}_t\right], \quad t \in [0, T]. \tag{3.5}$$

We can then use the Martingale representation theorem to express M_t with

$$M_t = M_0 + \int_0^t Z_s^* \, \mathrm{d}W_s. \tag{3.6}$$

According to Theorem 2.1 in [20], the solution $Y_t^{(x)}$, $t \in [0, T]$ can be written as

$$Y_t^{(x)} = M_t - \int_0^t f(s, X_s) \, \mathrm{d}s. \tag{3.7}$$

Hence, by using equations (3.5) and (3.7), the solution $Y_t^{(x)}$ can be written like

$$Y_t^{(x)} = \mathbb{E}[\Psi(X_T^{(x)}) + \int_t^T f(s, X_s) \, \mathrm{d}s \mid \mathcal{F}_t]. \tag{3.8}$$

Let $t' < t$, and by the tower property of conditional expectation write $\mathbb{E}[Y_t^{(x)} \mid \mathcal{F}_{t'}]$ as

$$\mathbb{E}[Y_t^{(x)} \mid \mathcal{F}_{t'}] = \mathbb{E}\left[\Psi(X_T^{(x)}) + \int_t^T f(s, X_s) \, \mathrm{d}s \mid \mathcal{F}_{t'}\right] = Y_{t'}^{(x)} - \mathbb{E}\left[\int_{t'}^t f(s, X_s) \, \mathrm{d}s \mid \mathcal{F}_{t'}\right]. \tag{3.9}$$

It is then clear that by adding $\int_{t'}^t f(s, X_s) \, \mathrm{d}s$, one gets the following Martingale

$$H = Y_t^{(x)} + \int_{t'}^t f(s, X_s) \, \mathrm{d}s. \tag{3.10}$$

Let b be a $\mathcal{B}([0,T]) \oplus \mathcal{B}(\mathbb{R}^{d\times\tilde{d}})$-measurable function. Then we can apply the Martingale representation Theorem 3.1.1 to the Martingale in (3.10) to see that

$$\int_{t'}^{t} f(r, X_r^{(x)})\ \mathrm{d}r + Y_t^{(x)} = \int_{t'}^{t} b(r, X_r^{(x)})^* \sigma(r, X_r^{(x)})\ \mathrm{d}W_r, \quad 0 \le t' \le t \le T. \tag{3.11}$$

Equation (3.11) makes it apparent that the solution $Y_t^{(x)}$ is adapted to the filtration $\mathcal{F}_t$. Despite being 'backward' in some sense, $Y_t^{(x)}$ is entirely determined by the forward diffusion X_t at time t. A further generalization that can be shown is that when the generator f is not independent of Y, Z, the solution $Y_t^{(x)}$ is still adapted to $\mathcal{F}_t$. The proof builds on the result in (3.11) and is made in Theorem 4.1 in [20].

3.2 Solving the BSDE

In the previous section, we saw that the process Y is uniquely determined with respect to $X_t^{(x)}$ and t. Therefore it is natural to express Y as a $\mathcal{B}([0,T]) \oplus \mathcal{B}(\mathbb{R}^d)$-measurable function u

$$Y_t^{(x)} = u(t, X_t^{(x)}). \tag{3.12}$$

If we set

$$Z_t^{(x)} = b(t, X_t^{(x)})^* \sigma(t, X_t^{(x)}), \tag{3.13}$$

Z is also adapted to the filtration $\mathcal{F}_t$. Under regularity assumptions, it is now possible to show that $Y_t^{(x)}$ and $Z_t^{(x)}$ as defined in (3.12) and (3.13) is in fact the unique solution to (3.2).

Next, we state a theorem that is helpful in describing the solution $Z_t^{(x)}$.

Theorem 3.2.1. *Let u be as in* (3.12)*. If μ, σ, f, Ψ are continuously differentiable in (x, y, z) and assume that the derivatives are uniformly bounded. Then for $0 \le t \le T, x \in \mathbb{R}^d$ we have*

$$Z_t^{(x)} = \sigma(t, X_t^{(x)})^{\perp} \frac{\partial}{\partial x} u(t, X_t^{(x)})\, \mathrm{d}\mathbb{P} \oplus \mathrm{d}t \quad a.s. \tag{3.14}$$

The proof of Theorem 3.2.1 is concise but relies on Malliavin calculus and several results from [20]. It appears as Corollary 4.1 in [20].

3.3 The Link Between PDEs and BSDEs

As mentioned in the beginning of the chapter, solutions to BSDEs of the form (3.2) solve certain PDEs. We will now underpin this with key results, but first the quasilinear parabolic PDE is introduced.

Let $\mathcal{L}$ be the operator defined in (2.17). Additionally let v be a function such that $v \in C^{1,2}$ and

$$\|\frac{\partial}{\partial t}v(t,x)\| + \|\sigma(t,x)^* v(t,x)\|_F \leq C\left(1 + \|x\|\right), \tag{3.15}$$

for some $C > 0$. Assuming Ψ, f are standard parameters, a quasilinear parabolic PDE is an equation of the form

$$\begin{aligned} &\frac{\partial}{\partial t}v(t,x) + \mathcal{L}[v](t,x) + f(t,x,v(t,x),\sigma(t,x)^*\frac{\partial}{\partial x}v(t,x)) = 0, \\ &\quad v(T,x) = \Psi(x), \end{aligned} \tag{3.16}$$

and the function v is its solution. Proposition 4.3 in [20] shows that if v solves (3.16), then $v(t,x) = Y_t^{(x)}$ solves the BSDE (3.2). This conclusion comes from a generalization of the Feynman–Kac formula, a result we state and prove next.

Theorem 3.3.1 (Generalization of the Feynman–Kac formula)**.**
Let $v \in C^{1,2}$ be a a solution to the PDE (3.16) *and assume there exists a constant C such that for each $(t,x) \in [0,T] \times \mathbb{R}^d$,*

$$|v(t,x)| + |\sigma(t,x)^*\frac{\partial}{\partial x}v(t,x)| \leq C(1+|x|). \tag{3.17}$$

Then, $(Y_t, Z_t) = \left(v(t, X_t^{(x)}), \sigma(t, X_t^{(x)})^* \frac{\partial}{\partial x} v(t, X_t^{(x)})\right)$, $0 \leq t \leq T$) is the unique solution to the BSDE (3.2).

Proof of Theorem 3.3.1. Let $t \in [0,T]$ and $x \in \mathbb{R}^d$. Applying Itô's formula to $v(t, X_t^{(x)})$, we obtain

$$\mathrm{d}v(t, X_t^{(x)}) = \left[\frac{\partial}{\partial t}v(t, X_t^{(x)}) + \mathcal{L}[v](t, X_t^{(x)})\right]\mathrm{d}t + \frac{\partial}{\partial x}v(t, X_t^{(x)})^*\sigma(t, X_t^{(x)})\,\mathrm{d}W_t. \tag{3.18}$$

Since v solves the PDE (3.16), we have

$$\begin{aligned} -\mathrm{d}v(t, X_t^{(x)}) &= f\left(t, X_t^{(x)}, \sigma(t, X_t^{(x)})^*\frac{\partial}{\partial x}v(t, X_t^{(x)})\right)\mathrm{d}t - \frac{\partial}{\partial x}v(t, X_t^{(x)})^*\sigma(t, X_t^{(x)})\,\mathrm{d}W_t, \\ v(T, X_T^{(x)}) &= \Psi(X_T^{(x)}). \end{aligned} \tag{3.19}$$

Hence for $t \in [0,T]$, $\left(v(X_t^{(x)}), \sigma(t, X_t^{(x)})^* \frac{\partial}{\partial x} v(t, X_t^{(x)})\right)$ is the unique solution to the BSDE (3.2). □

A central aspect of this thesis is the reverse connection between backward stochastic differential equations and partial differential equations. Specifically, under certain regularity assumptions, the solution to the BSDE (3.2) also provides a solution to the PDE (3.16) in the viscosity sense. To establish this connection, one typically requires the Comparison Theorem for BSDEs and a definition of viscosity solutions for PDEs. These are well covered concepts in the literature and we do not repeat the formal definitions here, we refer to Theorem 7.9 in [8] for the Comparison Theorem and

Definition 4.1 [20] for the definition of viscosity solution. Viscosity solutions allow us to interpret solutions to PDEs even when classical solutions may not exist, and the comparison theorem helps establish the necessary monotonicity used in proving that the BSDE solution satisfies the PDE in the viscosity sub- and supersolution sense.

The key result we rely on is Theorem 4.2 from [20]. It states that if the terminal condition $\Psi(x)$ and the generator f taking values in $\mathbb{R}$, are uniformly continuous in x, and satisfy standard growth and Lipschitz conditions, then the solution $Y_t^{(x)}$ of the BSDE defines a function

$$u(t, X_t^x) := Y_t^{(x)}.$$

Furthermore, it is a viscosity solution of the associated PDE (3.16). Under stronger regularity conditions, this solution is even classical (*i.e.*, continuously differentiable in both time and space), as established in Proposition 4.4 of [20].

These results highlight a strong connection between BSDEs and PDEs. In particular, it is the result that allow many numerical methods, including the Deep BSDE method, which reformulates the task of solving high-dimensional PDEs as solving BSDEs. This alternative way of solving PDE allows us to circumvent solving the Fokker–Planck equation, to instead solve a BSDE numerically by learning neural networks through Euler–Maruyama schemes, a method called the Deep BSDE method. We will return to the derivation of the Deep BSDE method in Chapter 5. Lastly, the area is theoretically heavy, so for the interested reader, we refer to [8] for more details on PDEs and their solutions and to [20, 37] which in details proves the above statements on the connection between PDEs and BSDEs.

4

Bayesian Filtering Problem

The Bayesian filtering problem can be summarized as the problem of estimating the unknown state of a process X, given noisy observations O. More specifically assuming we have a state space model, see Figure 2.1, we are interested in finding the probability density $p(X_{t_k} = x \mid \mathbb{O}_{1:k} = O_{1:k})$ for $k = 1, \ldots, K$. We call this probability density the Bayesian filtering density, or the exact filtering density. In some cases, as we will see below, the exact filter is tractable and in other cases it is not. When it is not we will try to approximate the true filter by approximative methods, sometimes referred to as approximative filters. Before introducing theory on specific filters, the main components of Bayesian filtering are introduced, along with a formalization of the problem via the *Bayesian filtering equations*.

Definition 4.1 (Filter components)**.**

1. *The prior model p_0 which is the probability density of the $\mathbb{R}^d$-valued random variable X_0.*
2. *An unknown Markovian state-space model $(X_t)_{t=0}^T$ that evolves according to a transition probability density $p(X_{t_{k,n}} = x \mid X_{t_{k,n-1}} = z) = p(X_{t_{k,n}} \mid X_{t_{k,n-1}})$. Given process noise $Q_{k,n-1}$, we define $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ to model the next state $X_{t_{k,n}}$*

$$X_{t_{k,n}} = f_{k,n-1}(X_{t_{k,n-1}}, Q_{k,n-1}), \quad n \in \{1 \ldots N\}, k \in \{0 \ldots K-1\}. \tag{4.1}$$

3. *The measurement model $p(\mathbb{O}_k = O_k \mid X_{t_k} = x) = p(\mathbb{O}_k \mid X_{t_k})$ which is the conditional probability of the observation process $\mathbb{O}_k$ given the state X_{t_k}. Let $h_k : \mathbb{R}^d \times \mathbb{R}^{d'} \to \mathbb{R}^{d'}$, then for some measurement noise V_k, the measurements are given by*

$$\mathbb{O}_k = h_k(X_{t_k}, V_k), \quad k \in \{1 \ldots K\}. \tag{4.2}$$

4. *The predictive posterior distribution $p(X_{t_k} = x \mid \mathbb{O}_{1:k-1} = O_{1:k-1}) = p(X_{t_k} \mid \mathbb{O}_{1:k-1})$ which is the probability distribution of the next state X_{t_k} given previous observations $\mathbb{O}_{1:k-1}$.*
5. *The conditional probability distribution of a state X_{t_k} given an observation $O_{1:k}$, is called the posterior distribution $p(X_{t_k} = x \mid \mathbb{O}_{1:k} = O_{1:k}) = p(X_{t_k} \mid \mathbb{O}_{1:k})$.*

We now arrive at the Bayesian filtering equations which are solutions for the predictive posterior distribution $p(X_{t_k} \mid \mathbb{O}_{1:k-1})$ and the posterior distribution $p(X_{t_k} \mid \mathbb{O}_{1:k})$.

Theorem 4.0.1 (Bayesian filtering equations)**.** *Given the filter components from Definition 4.1 we have:*

- *(Chapman–Kolmogorov) The predictive posterior $p(X_{t_k} \mid \mathbb{O}_{1:k-1})$ is given by*

$$p(X_{t_k} = x|\mathbb{O}_{1:k-1}) = \int p(X_{t_k} = x|X_{t_{k-1}} = z)p(X_{t_{k-1}} = z|\mathbb{O}_{1:k-1})\,\mathrm{d}z. \tag{4.3}$$

- *(Bayes') The posterior $p(X_{t_k} \mid \mathbb{O}_{1:k})$ is given by*

$$p(X_{t_k} = x \mid \mathbb{O}_{1:k}) = \frac{p(\mathbb{O}_k \mid X_{t_k} = x)p(X_{t_k} = x \mid \mathbb{O}_{1:k-1})}{\int_{\mathbb{R}^d} p(\mathbb{O}_k \mid z)p(z \mid \mathbb{O}_{1:k-1})\,\mathrm{d}z}. \tag{4.4}$$

There are both closed-form solutions and approximate solutions to the Bayesian filtering equations (4.3) and (4.4). The Kalman filter provides a closed-form solution when the measurement and state-space models are linear and their noise is Gaussian, a solution that is unimodal Gaussian. When the measurement and state-space models are nonlinear, the Extented Kalman Filter (EKF) and particle filters provide approximate solutions to the Bayesian filtering equations. The approximate solution provided by the EKF is obtained by linearizing the nonlinear system around the current estimate. Particle filters comes with the additional benefit of being invariant to modality, *i.e.*, particle filters can approximate state-space distributions that are multimodal. Based on Simo Särkka's book *Bayesian Filtering and Smoothing* [32] we introduce the three filters that are relevant for this thesis, *i.e.* the Kalman filter, the Extended Kalman filter and the bootstrap particle filter.

4.1 The Kalman Filters

The main idea behind what is currently known as the Kalman filter was initially conceived by T.N. Thiele during the 1880s [24]. However, it was not until the late 1950s and early 1960s that Hungarian-American engineer and mathematician Rudolf E. Kálmán developed and formalized it to the *Kalman filter* [19]. The original Kalman filter has since then been complemented by adjacent ideas into a family of filters *e.g.* the Extended Kalman filter, the Unscented Kalman filter, the Kalman–Bucy filter [30] and the Discriminative Kalman filter [4] to name a few. Applications range from quantitative finance and stock market prediction to GPS navigation and autonomous vehicle sensor fusion [2]. This thesis will use the standard Kalman filter and the EKF for training normalization of the Deep BSDE filter, as well as for direct comparison to the Deep BSDE filter.

4.1.1 The Standard Kalman Filter

The Kalman filter can be summarized in one theorem. However, the specific conditions when the Kalman filter provides a closed-form solution to the Bayesian filtering equations need to be defined first.

Definition 4.2 (Kalman setting)**.** *Let $Q_{k,n} \sim \mathcal{N}(0, \boldsymbol{Q}_{k,n})$ be the process noise for some covariance matrix $\boldsymbol{Q}_{k,n} \in \mathbb{R}^{d\times d}$, and let $V_k \sim \mathcal{N}(0, \boldsymbol{R}_k)$ be the measurement noise for some covariance matrix $\boldsymbol{R}_k \in \mathbb{R}^{d'\times d'}$. For some $m_0 \in \mathbb{R}^d$ and some $\boldsymbol{P}_0 \in \mathbb{R}^{d\times d}$, we let the prior distribution be Gaussian $X_0 \sim \mathcal{N}(m_0, \boldsymbol{P}_0)$. We call $\boldsymbol{A}_{k,n} \in \mathbb{R}^{d\times d}$ the transition matrix and $\boldsymbol{H}_k \in \mathbb{R}^{d'\times d}$ the measurement matrix. Given a state $X_{t_{k,n}}$, the next state $X_{t_{k,n+1}}$ and observation $\mathbb{O}_k$ is given by*

$$X_{t_{k,n+1}} = \boldsymbol{A}_{k,n} X_{t_{k,n}} + Q_{k,n}, \quad k = 0, \dots, K, n = 0, \dots, N-1, \tag{4.5}$$
$$\mathbb{O}_k = \boldsymbol{H}_k X_{t_k} + V_k, \quad k = 0, \dots, K. \tag{4.6}$$

It can be shown that for the Kalman filter to be applicable, the probabilistic model X needs to be a linear model with additive Gaussian noise, as specified in Definition 4.2. Now, two sub-procedures that are used in the Kalman filter are introduced, namely the prediction and the update steps.

To start we know m_0. Then for $k \in \{1, \dots K-1\}$, the Kalman estimate of the prior mean $m_k^- \in \mathbb{R}^d$ is given by

$$\begin{aligned} m_{k,1}^- &= \mathbf{A}_{k,0} m_{k,0}, \quad n = 1, \\ m_{k,n}^- &= \mathbf{A}_{k,n-1} m_{k,n-1}^-, \quad n = 2, \dots, N. \end{aligned} \tag{4.7}$$

Likewise we initially know P_0. Then for $k \in \{1, \dots K-1\}$, the Kalman estimate of the prior covariance matrix $\mathbf{P}_k^-$ is given by

$$\begin{aligned} \mathbf{P}_{k,1}^- &= \mathbf{A}_{k,0} \mathbf{P}_{k-1} \mathbf{A}_{k,0}^\top + \mathbf{Q}_{k,0}, \quad n = 1, \\ \mathbf{P}_{k,n}^- &= \mathbf{A}_{k,n-1} \mathbf{P}_{k,n-1}^- \mathbf{A}_{k,n-1}^\top + \mathbf{Q}_{k,n-1}, \quad n = 2, \dots, N. \end{aligned} \tag{4.8}$$

The calculations in (4.7) and (4.8) together are what is typically called the Kalman prediction step.

The update step uses the calculated m_k^- and $\mathbf{P}_k^-$ from the prediction equations (4.7) and (4.8). Let O_k be the observation, then for $k \in \{1, \dots, K\}$ the posterior mean m_k and posterior covariance $\mathbf{P}_k$ are given by

$$\begin{aligned} v_k &= O_k - \mathbf{H}_k m_{k-1,N}^-, \\ \mathbf{S}_k &= \mathbf{H}_k \mathbf{P}_{k-1,N}^- \mathbf{H}_k^\top + \mathbf{R}_k, \\ \mathbf{K}_k &= \mathbf{P}_{k-1,N}^- \mathbf{H}_k^\top \mathbf{S}_k^{-1}, \\ m_k &= m_{k-1,N}^- + \mathbf{K}_k v_k, \\ \mathbf{P}_k &= \mathbf{P}_{k-1,N}^- - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\top. \end{aligned} \tag{4.9}$$

The main result can now be stated using (4.7), (4.8) and (4.9).

Theorem 4.1.1 (the Kalman filter)**.** *Given a model like in Definition 4.2, the closed-form solution to the first Bayesian filtering equation* (4.3) *is given by*

$$p(X_{t_k} \mid \mathbb{O}_{1:k-1}) = N(X_{t_k} \mid m_k^-, \boldsymbol{P}_k^-) \tag{4.10}$$

where m_k^- *and* $\boldsymbol{P}_k^-$ *are given by the Kalman prediction step in* (4.7) *and* (4.8) *respectively. The closed-form solution to the second Bayesian filtering equation* (4.4) *is given by*

$$p(X_{t_k} \mid \mathbb{O}_{1:k}) = N(X_{t_k} \mid m_k, \boldsymbol{P}_k) \tag{4.11}$$

where the posterior mean m_k *and posterior covariance* $\boldsymbol{P}_k$ *are given by the Kalman update step in* (4.9)*.*

The reader is referred to Theorem 4.2 of [32] for a complete proof of Theorem 4.1.1.

The Kalman filter can be seen as the discrete time counterpart of the Kalman–Bucy filter, which solves the filtering problem for linear systems in continuous time. When time is discretized the Kalman–Bucy filter reduces to the standard Kalman filter as was presented above. In this thesis, we will discretize continuous SDEs which we then apply the standard Kalman filter on. To illustrate this, first consider the general setting defined in Section 2.2:

$$X_{t_k} = X_{t_0} + \int_{t_0}^{t_k} \mu(s, X_s)\,\mathrm{d}s + \int_{t_0}^{t_k} \sigma(s, X_s)\,\mathrm{d}W_s \tag{4.12}$$

and its corresponding Euler–Maruyama scheme, which we introduced in Section 2.4

$$\widetilde{X}_{k,n} = \widetilde{X}_{k,n-1} + \mu(t_{k,n-1}, \widetilde{X}_{k,n-1})\Delta t + \sigma(t_{k,n-1}, \widetilde{X}_{k,n-1})(W_{t_{k,n}} - W_{t_{k,n-1}}). \tag{4.13}$$

To obtain the probability density of X_{t_k} given observations $O_{1:k}$, we use the Euler–Maruyama scheme in (4.13) and set transition matrix $\mathbf{A}_{k,n}$ in Definition 4.2 to

$$\mathbf{A}_{k,n+1} = I_d + \frac{\partial \mu(t_{k,n}, X_{k,n})}{\partial x}\Delta t, \quad n = 0, 1, \ldots, N-1, \tag{4.14}$$

and process noise $Q_{k,n} \sim \mathcal{N}(0, \sigma(t_{k,n}, X_{t_{k,n}})\sigma^\top(t_{k,n}, X_{t_{k,n}})\Delta t)$. With linear μ, σ and measurement model that has additive Gaussian noise, and a Gaussian prior distribution, we have that the Kalman filter provides a closed form solution for the Bayesian filtering equations for the discretized version of the SDE in Equation (4.12).

4.1.2 Extended Kalman Filter

The Extended Kalman filter makes up for the most significant shortfall of the base Kalman filter, namely that it is only applicable when the state-space model and measurement model are linear and have Gaussian noise. To cover this weakness, the EKF incorporates Taylor series of the nonlinear components of the state-space model. Before presenting the EKF, derivations that lay the foundation for the EKF are made below.

Given a mean vector $m \in \mathbb{R}^d$, a covariance matrix $\mathbf{P} \in \mathbb{R}^{d\times d}$ and a nonlinear function $g : \mathbb{R}^d \to \mathbb{R}^{d'}$, let

$$X \sim \mathcal{N}(m, \mathbf{P}) \quad \textit{and} \quad Y = g(X), \tag{4.15}$$

with $X \in L^2(\Omega, \mathbb{R}^d)$ and $Y \in L^2(\Omega, \mathbb{R}^{d'})$. Writing $X = m+\delta X$ where $\delta X \sim \mathcal{N}(0, \mathbf{P})$ we then have that the first order Taylor expansion of $g(X)$ is given by

$$g(X) = g(m + \delta X) \approx g(m) + \mathbf{G}(X)\delta X + R, \tag{4.16}$$

where $\mathbf{G}(X) \in L^2(\Omega, \mathbb{R}^{d'\times d})$ is the Jacobian matrix of g evaluated at X and the remainder R is of order $O(\delta X^2)$.
The expectation of $g(X)$ in 4.16 is given by

$$\mathbb{E}[g(X)] = \mathbb{E}[g(m + \delta X)] \approx \mathbb{E}[g(m) + \mathbf{G}(X)\delta X] = g(m) \tag{4.17}$$

and the covariance of $g(X)$ in 4.16 is given by

$$\begin{aligned}
&\mathbb{E}\Big[(g(X) - \mathbb{E}[g(X)])(g(X) - \mathbb{E}[g(X)])^\top\Big] \approx \\
&\mathbb{E}\Big[(g(X) - g(m))(g(X) - g(m))^\top\Big] \approx \\
&\mathbb{E}\Big[(g(m) + \mathbf{G}(X)\delta X - g(m))(g(m) + \mathbf{G}(X)\delta X - g(m))^\top\Big] \approx \\
&\mathbf{G}(X)\mathbb{E}[\delta X(\delta X)^\top]\mathbf{G}^\top(X) = \mathbf{G}(X)\mathbf{P}\mathbf{G}^\top(X).
\end{aligned} \tag{4.18}$$

The use of Taylor expansions at appropriate steps above is in the proof of the EKF the main difference compared to the proof of the Kalman filter. This enables the covariance of a nonlinear g to be expressed in terms of the covariance $\mathbf{P}$ and the Jacobian matrix $\mathbf{G}(X)$ of g at X.

Next, we define the conditions for when the EKF is applicable.

Definition 4.3 (EKF setting)**.** *Let $Q_{k,n}$, V_k, m_0, and $\boldsymbol{P}_0$ be defined as in the Kalman filter, Definition 4.2. We let $f : \mathbb{R}^d \to \mathbb{R}^d$ be the dynamic model function and $h(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d'}$ the measurement model function. Given a state $X_{t_{k,n}}$ we then have the next state $X_{t_{k,n+1}}$ and observation $\mathbb{O}_k$ given by*

$$X_{t_{k,n+1}} = f_{k,n}(X_{t_{k,n}}) + Q_{k,n}, \quad k = 0, \dots, K, n = 0, \dots, N-1, \tag{4.19}$$

$$\mathbb{O}_k = h_k(X_{t_k}) + V_k, \quad k = 0, \dots, K. \tag{4.20}$$

Similarly to the Kalman filter, the EKF has a prediction and an update procedure. To start we know m_0. Then for $k \in \{1, \dots K-1\}$, the EKF estimate of the prior mean m_k^- is given by

$$\begin{aligned}
m_{k,1}^- &= f_{k,0}(m_{k,0}), \quad n = 1, \\
m_{k,n}^- &= f_{k,n}(m_{k,n-1}^-), \quad n = 2 \dots N.
\end{aligned} \tag{4.21}$$

Let $\mathbf{F}_f(x) : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ be the Jacobian matrix of f at the point x. Then for $k \in \{1, \dots K-1\}$, the EKF of the prior covariance matrix $\mathbf{P}_k^-$ is given by

$$\begin{aligned}\mathbf{P}_{k,1}^{-} &= \mathbf{F}_{f_{k,0}}(m_{k,0})\mathbf{P}_{k,0}\mathbf{F}_{f_{k,0}}^{\top}(m_{k,0}) + \mathbf{Q}_{k,0}, \quad n = 1, \\ \mathbf{P}_{k,n}^{-} &= \mathbf{F}_{f_{k,n-1}}(m_{k,n-1})\mathbf{P}_{k,n-1}^{-}\mathbf{F}_{f_{k,n-1}}^{\top}(m_{k,n-1}) + \mathbf{Q}_{k,n-1}, \quad n \in \{2, \ldots, N\}.\end{aligned} \tag{4.22}$$

Let $\mathbf{H}_h(x)$: $\mathbb{R}^d \to \mathbb{R}^{d \times d'}$ be the Jacobian matrix of the measurement model $h_k(x)$ in Definition 4.3 at the point x, let m_k^- and $\mathbf{P}_k^-$ be the estimate of the prior mean and covariance given by (4.21) and (4.22). Then, the posterior mean m_k and posterior covariance $\mathbf{P}_k$ are given by

$$\begin{aligned} v_k &= O_k - h_k(m_k^-), \\ \mathbf{S}_k &= \mathbf{H}_{h_k}(m_k^-)\mathbf{P}_k^-\mathbf{H}_{h_k}^{\top}(m_k^-) + \mathbf{R}_{t_k}, \\ \mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}_{h_k}^{\top}(m_k^-)\mathbf{S}_{t_k}^{-1}, \\ \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k v_k, \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^{\top}. \end{aligned} \tag{4.23}$$

With m_k^-, $\mathbf{P}_k-$, m_k and $\mathbf{P}_k$ defined as above, the EKF theorem is identical to the Kalman filter in Theorem 4.1.1 except that the solution is now approximate. Therefore, if h and g are linear and the noise distribution is Gaussian, the EKF simplifies to the Kalman filter.

4.2 The Particle Filter

As eluded to earlier, particle filters are applicable to more situations than the Kalman filters. However, due to the higher computational cost of particle filters, the Kalman filter and the EKF are preferred whenever they are applicable [32]. In this thesis, the bootstrap particle filter is implemented, but a more general theoretical framework for particle filters is presented below. Particle filters are Monte Carlo methods that approximate the Bayesian filtering equations using a concept called importance sampling. They are also commonly called sequential Monte Carlo methods which follows naturally from Particle filters being Monte Carlo methods and sequentially applying importance sampling. The reader is referred to Chapter 7 in [32] for the full derivation, since only the main ideas will be presented here.

Theorem 4.2.1 (Importance sampling)**.** *Let p_0, $p(\mathbb{O}_k \mid X_{t_k})$ and $p(X_{t_k} \mid \mathbb{O}_{1:k})$ be as in Definition 4.1. Let $\pi(X_{t_k} \mid \mathbb{O}_{1:k})$ be a probability distribution such that $\operatorname{supp}(p(X \mid \mathbb{O}_{1:k})) \subseteq \operatorname{supp}(\pi(X \mid \mathbb{O}_{1:k}))$, called the importance distribution, and let $g : \mathbb{R}^d \to \mathbb{R}^{d'}$ be an arbitrary function. The approximation of the posterior expectation $E[g(X_{t_k}) \mid \mathbb{O}_{1:k}]$ and the posterior distribution $p(X_{t_k} \mid \mathbb{O}_{1:k})$ is then given by steps (1-5):*

1. *For $i = 1, \ldots, M$ sample $x^{(i)} \sim \pi(X \mid \mathbb{O}_{1:k})$ from the importance distribution.*
2. *Compute weights $w^{(i)}$*

$$w^{(i)} = \frac{p(\mathbb{O}_{1:k} \mid X_{t_k} = x^{(i)})p(X_{t_k} = x^{(i)})}{\pi(X_{t_k} = x^{(i)} \mid \mathbb{O}_{1:k})}, \quad i = \{1, \ldots, M\}. \tag{4.24}$$

3. *Replace weights* $w^{(i)}$ *with normalized weights* $\hat{w}^{(i)}$ *such that* $\sum_{i=0}^{M} \hat{w}^{(i)} = 1$.
4. *The Monte Carlo approximation of the posterior expectation of* $g(X_{t_k})$ *is*

$$E[g(X_{t_k}) \mid \mathbb{O}_{1:k}] \approx \sum_{i=1}^{M} w^{(i)} g(x^{(i)}). \tag{4.25}$$

5. *Approximation of the posterior distribution* $p(X_{t_k} \mid \mathbb{O}_{1:k})$ *is then given by*

$$p(X_{t_k} \mid \mathbb{O}_{1:k}) \approx \sum_{i=1}^{M} w^{(i)} \delta(X_{t_k} - x^{(i)}) \tag{4.26}$$

where δ *is the delta-dirac function.*

Applying Theorem 4.2.1 repeatedly we obtain the algorithm commonly called the *Particle filter*, which derives its name from the samples that can be thought of as particles propagating through time.

Algorithm 1 (Particle filter)**.** *Approximation of the posterior distribution* $p(X_{t_k} \mid \mathbb{O}_{1:k})$ *is given by steps (1-3):*

1. *For* $i = 1, \ldots, M$ *sample* $x_0^{(i)} \sim p_0$ *and initialize weights*

$$w_0^{(i)} = \frac{1}{M}, \quad i = \{1, \ldots, M\}.$$

2. *For* $k = 1, \ldots, N$ *do*
   (a) *Draw samples* $x_{t_k}^{(i)} \sim \pi(X_{t_k} \mid X_{t_{0:k-1}} = x_{t_{0:k-1}}^{(i)}, \mathbb{O}_{1:k}), \quad i = \{1, \ldots M\}$.
   (b) *Update weights*

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p(\mathbb{O}_k \mid X_{t_k} = x_{t_k}^{(i)}) p(X_{t_k} = x_{t_k}^{(i)} \mid X_{t_{k-1}} = x_{t_{k-1}}^{(i)})}{\pi(X_{t_k} = x_{t_k}^{(i)} \mid X_{t_{k-1}} = x_{t_{k-1}}^{(i)}, \mathbb{O}_{1:k})}. \tag{4.27}$$

   (c) *Replace weights with normalized weights* $\hat{w}_k^{(i)}$ *such that* $\sum_{i=0}^{M} \hat{w}_k^{(i)} = 1$.
   (d) *(Optional) Run a resampling algorithm if effective particles are too few.*

3. *For* $k = 1, \ldots, N$*, apply importance sampling from Theorem 4.2.1 to approximate the posterior distribution* $p(X_{t_k} \mid \mathbb{O}_{1:k})$.

A resampling algorithm resamples particles based on a discrete probability distribution using the particles' weights as probability densities. It combats the particle degeneracy problem, which occurs when the majority of particles have weights close to 0. What is deemed *'too few'* effective particles is an empirical exercise. Särkka [32] provides a method to approximate the number of effective particles n_{eff} as in (4.28) and recommends to resample if $n_{eff} < \frac{M}{10}$.

$$n_{eff} \approx \frac{1}{\sum_{i=1}^{M} (w_k^{(i)})^2}. \tag{4.28}$$

Lastly, this thesis will implement the bootstrap particle filter, which is obtained by setting the importance distribution to $\pi = p(X_{t_k} \mid X_{t_{k-1}})$.

5

A Deep BSDE Method for Filtering

In this chapter, the derivation of the Deep BSDE Filter (DBSDEF) is outlined. It adapts the Deep BSDE method to a filter setting by using it to solve the Fokker–Planck equation as part of the prediction step in a filter. The chapter starts by introducing the Deep BSDE method followed by how it can be used in sequence, something we call the *sub-problem approach*. The sub-problem approach is central in how the DBSDEF is constructed and after it is introduced, the chapter finishes with the derivation of the DBSDEF.

5.1 The BSDE Method

The Deep BSDE method was introduced in [12], and has been useful in solving many equations, for instance the Allen–Cahn equation and the Hamilton–Jacobi–Bellman equation. Its convergence to the true solution for solving parabolic partial differential equations is also widely explored, for instance in [17] and for both linear and nonlinear cases [36]. The already presented connection between PDEs and BSDEs seen in Section 3.3 is used to reformulate the solution of a PDE to the solution of a pair of forward and backward equations. This pair can then be discretized with an Euler–Maruyama scheme where we seek to find p_t and the gradient ∇p_t for $t \in [0, T]$, where p_t is the probability density of the underlying SDE. The unknown functions $p_t, \nabla p_t$ are approximated by neural networks, which is what makes this a 'deep' learning-based approach. In our case, the PDE of interest is the Fokker–Planck equation (5.4) since its solution is the probability density of an SDE, which is used the prediction step for the Deep BSDE filter. This section provides an outline of the derivation of the method and the reader is referred to the original works by Han *et al.* for complete rigorous proofs [12, 17].

Define two complete probability spaces $(\Omega, \mathcal{F}, \mathbb{P})$, $(\tilde{\Omega}, \tilde{\mathcal{F}}, \tilde{\mathbb{P}})$ with filtrations $\mathbb{F} = (\mathcal{F}_t)_{t\in[0,T]}$, $\tilde{\mathbb{F}} = (\tilde{\mathcal{F}}_t)_{t\in[0,T]}$ as defined in Section 2.1 and let $(B_t)_{t\in[0,T]}$, $(W_t)_{t\in[0,T]}$ be d dimensional Brownian motions adapted to the filtrations $\mathbb{F}$ and $\tilde{\mathbb{F}}$ respectively. Also assume that the drift term μ and the diffusion term σ are time-homogenous and satisfy the conditions in Theorem 2.2.1. Then there exists a stochastic process $(S_t)_{t=0}^T$ that satisfies the SDE

$$S_t = S_0 + \int_0^t \mu(S_s)\,\mathrm{d}s + \int_0^t \sigma(S_s)\,\mathrm{d}B_s, \quad t \in [0, T], \tag{5.1}$$

where $S_0 \sim p_0$ for some known initial probability distribution p_0. Additionally, let p_t, $t \in [0, T]$, denote the probability density of the process S at time t. The quantity of interest however is p_T, which can then be used together with Bayes' rule to form a filter. A sceptical reader might notice that one can sample a histogram of the density p_T. However, the method proposed below aims to compute the function p_T, which differs from the empirical cumulative distribution derived from a histogram, as the latter usually does not cover the entire domain.

To find p_T we proceed by recalling the infinitesimal generator $\mathcal{L}$ from Section 2.3 and its adjoint operator $\mathcal{L}^*$ acting on some function ψ

$$\begin{aligned} \mathcal{L}[\psi](x) &= \sum_{i=1}^{d} \mu_i(x) \frac{\partial \psi}{\partial x_i} + \frac{1}{2} \sum_{i,j=1}^{d} a_{i,j}(x) \frac{\partial^2 \psi}{\partial x_i \partial x_j}, \\ \mathcal{L}^*[\psi](x) &= -\sum_{i=1}^{d} \frac{\partial}{\partial x_i} \left(\mu_i(x)\psi\right) + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j} \left(a_{i,j}(x)\psi\right), \end{aligned} \tag{5.2}$$

where again $a(x) = \sigma(x)\sigma^\top(x)$. We also recall the time homogenous Fokker–Planck equation from the same section

$$\frac{\partial}{\partial t} p_t(x) = \mathcal{L}^*[p_t](x), \quad t \in [0, T], \tag{5.3}$$

which we rewrite in integral form

$$p_t(x) = p_0(x) + \int_0^t \mathcal{L}^*[p_s](x) \, \mathrm{d}s, \quad t \in [0, T]. \tag{5.4}$$

By defining a measurable function $f : \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$

$$f(x, u, v) = \sum_{i,j=1}^{d} \frac{\partial a_{ij}(x)}{\partial x_i} v_j + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2 a_{ij}(x)}{\partial x_i \partial x_j} u - \sum_{i=1}^{d} \frac{\partial \mu_i(x)}{\partial x_i} u - 2 \sum_{i=1}^{d} \mu_i(x) v_i, \tag{5.5}$$

we see that the following holds for a function ψ

$$\mathcal{L}^*[\psi](x) = \mathcal{L}[\psi](x) + f(x, \psi(x), \nabla_x \psi(x)). \tag{5.6}$$

The function f captures the asymmetry in the dynamics of the SDE, however, only spatial asymmetry since we have time-homogenous SDEs. If the SDE is perfectly spatially symmetric at all points in its domain, *e.g.*, for a Brownian motion, then $f = 0$ at all points. For any asymmetric SDE, we generally have a non-zero f, except at special points, for instance at precisely the long-term mean in a Ornstein–Uhlenbeck process.

We incorporate the function f in the integral from of the Fokker–Planck equation (5.4) by replacing the adjoint operator of the infinitesimal generator $\mathcal{L}^*$ with (5.6)

$$p(t, x) = p(0, x) + \int_0^t \mathcal{L}[p](s, x) \, \mathrm{d}s + \int_0^t f(x, p(s, x), \nabla_x p(s, x)) \, \mathrm{d}s, \tag{5.7}$$

which has the differentiated form

$$\begin{aligned} \frac{\partial}{\partial t}p(t,x) &= \mathcal{L}[p](t,x) + f(x, p(t,x), \nabla_x p(t,x)), \\ p(0,x) &= p_0(x). \end{aligned} \tag{5.8}$$

This reformulation of the Fokker–Planck equation bears resemblance with the more general form of quasilinear PDEs introduced in Section 3.3

$$\begin{aligned} &\frac{\partial}{\partial t}v(t,x) + \mathcal{L}[v](t,x) + f\left(t, x, v(t,x), \sigma(t,x)^*\nabla_x v(t,x)\right) = 0, \\ &v(T,x) = \Psi(x). \end{aligned} \tag{5.9}$$

The reformulated Fokker–Planck equation (5.8) corresponds to (5.9), up to a sign reversal in the functions p_t, f, and the operator $\mathcal{L}$. Hence, with a reparametrization of (5.8) by $t \to T - t$ and by defining $q(t) = p(T - t)$ we achieve exactly (5.9)

$$\begin{aligned} \frac{\partial}{\partial t}q(t,x) + &\mathcal{L}[q](t,x) + f(x, q(t,x), \nabla_x q(t,x)) = 0, \\ &q(T,x) = p_0(x), \end{aligned} \tag{5.10}$$

where $q(t,x)$ corresponds to $v(t,x)$ in (5.9). Also notice that the terminal condition follows from the definition of $q(t,x)$, which means that the old initial condition is now our terminal condition. The reparametrization of time is essential since we assume knowledge of the function p_0, but not p_T.

We assume sufficient conditions on p_0, μ and σ such that $p, q \in C^{1,2}([0,T] \times \mathbb{R}^d; \mathbb{R})$ and thus Itô's formula is applicable. Introducing the auxiliary process X which is defined by the same dynamics as the SDE (5.1), but driven by the second Brownian motion

$$X_t = X_0 + \int_0^t \mu(X_s)\ \mathrm{d}s + \int_0^t \sigma(X_s)\,\mathrm{d}W_s, \tag{5.11}$$

we apply Itô's formula on $q(t, X_t)$ to get

$$q(t,X_t) = q(0,X_0) + \int_0^t \left(\frac{\partial}{\partial s}q(s,X_s) + \mathcal{L}[q](s,X_s)\right)\mathrm{d}s + \int_0^t \nabla q(s,X_s)\sigma(X_s)\,\mathrm{d}W_s. \tag{5.12}$$

Applying the result (5.10) we get

$$q(t,X_t) = q(0,X_0) - \int_0^t f(X_s, q(s,X_s), \nabla q(s,X_s))\,\mathrm{d}s + \int_0^t \nabla q(s,X_s)\sigma(X_s)\ \mathrm{d}W_s. \tag{5.13}$$

By defining $Y_t = q(t,X_t)$ and $Z_t = \nabla_x q(t,X_t)\sigma(X_t)$ we get

$$Y_t = q(0,X_0) - \int_0^t f\left(X_s, Y_s, Z_s\sigma^\top(X_s)(\sigma(X_s)\sigma^\top(X_s))^{-1}\right)\mathrm{d}s + \int_0^t Z_s\,\mathrm{d}W_s, \tag{5.14}$$

with the terminal condition $Y_T = q(T, X_T) = p_0(X_T)$. Now, by shifting (5.14) with $t \to T$ and $0 \to t$ we get

$$Y_T = Y_t - \int_t^T f\left(X_s, Y_s, Z_s\sigma^\top(X_s)(\sigma(X_s)\sigma^\top(X_s))^{-1}\right) \mathrm{d}s + \int_t^T Z_s \,\mathrm{d}W_s. \tag{5.15}$$

Rearranging (5.15) to express Y_t in a second way

$$Y_t = Y_T + \int_t^T f\left(X_s, Y_s, Z_s\sigma^\top(X_s)(\sigma(X_s)\sigma^\top(X_s))^{-1}\right) \mathrm{d}s - \int_t^T Z_s \,\mathrm{d}W_s, \tag{5.16}$$

which corresponds exactly to the general BSDE form introduced in the beginning of Chapter 3. Also, there exists a unique solution (Y_t, Z_t) to (5.16) if f and p_0 are standard parameters as defined in the beginning of Chapter 3.

As mentioned in Section 3.3, Theorem 4.4 in [20] helps us to conclude the method with the result that the solution (Y_t, Z_t) to (5.16) solves its corresponding PDE (5.9) with $Y_t = v(t, X_t)$. In our case, this PDE is the reformulated Fokker–Planck equation (5.10) with $v(t, x) = q(t, x)$. To summarize our efforts, the equation system we end up with is

$$\begin{aligned} X_t &= X_0 + \int_0^t \mu(X_s) \,\mathrm{d}s + \int_0^t \sigma(X_s) \,\mathrm{d}W_s, \\ Y_t &= p_0(X_T) + \int_t^T f\left(X_s, Y_s, Z_s\sigma^\top(X_s)(\sigma(X_s)\sigma^\top(X_s))^{-1}\right) \mathrm{d}s - \int_t^T Z_s \,\mathrm{d}W_s. \end{aligned} \tag{5.17}$$

The above derivation provides the main idea in the Deep BSDE method, namely that instead of solving the Fokker–Planck equation, we now aim to solve for (X, Y, Z), $X = (X_t)_{t=0}^T, Y = (Y_t)_{t=0}^T, Z = (Z_t)_{t=0}^T$ in (5.17).

Although not formally proved, the argument in Section 3.1 motivated that the process Y_t was adapted to the filtration $\tilde{\mathcal{F}}_t$. Therefore, by approximating the solution X to the auxiliary SDE (5.11), the solution (Y, Z) to the BSDE in (5.17) can likewise be approximated from the approximation of X. By assuming knowledge of the unknown $q(0, X_0) = p(T, X_0) = Y_0$ and $\nabla q_x(t, x)$, we can formulate the problem as approximating the solution (X, Y, Z) as

$$X_t = X_0 + \int_0^t \mu(X_s) \,\mathrm{d}s + \int_0^t \sigma(X_s) \,\mathrm{d}W_s, \quad t \in [0, T], \tag{5.18}$$

$$Y_t = q(0, X_0) - \int_0^t f\left(X_s, Y_s, Z_s\sigma^\top(X_s)\left(\sigma(X_s)\sigma^\top(X_s)\right)^{-1}\right) \mathrm{d}s + \int_0^t Z_s \,\mathrm{d}W_s, \quad t \in [0, T], \tag{5.19}$$

with the following Euler–Maruyama scheme $\{\mathcal{X}_k, \mathcal{Y}_k, \mathcal{Z}_k\}_{k=0}^K$ on $0 = t_0 < t_1 < \cdots < t_K = T$

$$\begin{aligned} \mathcal{X}_{k+1} &= \mathcal{X}_k + \tau\mu(\mathcal{X}_k) + \sigma(\mathcal{X}_k)\Delta W_{t_k}, \\ \mathcal{Y}_{k+1} &= \mathcal{Y}_k - \tau f\left(\mathcal{X}_k, \mathcal{Y}_k, \mathcal{Z}_k\sigma^\top(X_s)\left(\sigma(X_s)\sigma^\top(X_s)\right)^{-1}\right) + \mathcal{Z}_k\Delta W_{t_k}, \end{aligned} \tag{5.20}$$

where $\mathcal{X}_0 \sim p_0$, $\mathcal{Y}_0 = q(0, X_0)$, $\tau = \frac{T}{K}$ and $\Delta W_{t_k} = W_{t_{k+1}} - W_{t_k}$. By quantifying $\|Y_T - \mathcal{Y}_K\|_{L^2(\Omega;R)} = \|p_0(X_T) - \mathcal{Y}_K\|_{L^2(\Omega;R)}$ we estimate how well approximation works. The issue with (5.20) is that the approximation $\mathcal{Z}_k$ of $Z_t = \nabla_x q(t, X_t)\sigma(X_t)$ is unknown, and therefore we need an alternative scheme that is implementable in practice. Rather than assuming knowledge of the function $q(0, X_0)$ we parametrise a continuous function $u : \mathbb{R}^d \to \mathbb{R}$ aimed to approximate $q(0, X_0)$. In addition, functions $v_k : \mathbb{R}^d \to \mathbb{R}^d$, $k = 0, \dots, K-1$ are parametrized to approximate the gradients $\nabla_x q(t, X_t)$. Then we can formulate the following minimisation problem

$$\text{minimise}_{u,v \in C(\mathbb{R}^d;\mathbb{R}) \times C(\mathbb{R}^d;\mathbb{R}^d)^{\times K}} \mathbb{E}\left[|\mathcal{Y}_K^{u,v} - p_0(\mathcal{X}_K)|^2\right], \tag{5.21}$$

$$\mathcal{Y}_0^{u,v} = u(\mathcal{X}_0), \tag{5.22}$$

$$\mathcal{Z}_k^{u,v} = v_k(\mathcal{X}_k)\sigma(\mathcal{X}_k), \qquad k = 0, \dots, K-1, \tag{5.23}$$

$$\mathcal{X}_{k+1} = \mathcal{X}_k + \tau\mu(\mathcal{X}_k) + \sigma(\mathcal{X}_k)\Delta W_{t_k} \qquad k = 0, \dots, K-1, \tag{5.24}$$

$$\mathcal{Y}_{k+1}^{u,v} = \mathcal{Y}_k^{u,v} - \tau f\left(\mathcal{X}_k, \mathcal{Y}_k^{u,v}, v_k(\mathcal{X}_k)\right) + \mathcal{Z}_k^{u,v}\Delta W_{t_k}, \qquad k = 0, \dots, K-1. \tag{5.25}$$

The above solver is what is known as the original Deep BSDE method. As typically done in the Deep BSDE method, we use neural networks to approximate the functions $u, \{v_k\}_{k=0}^{K-1}$, and we will return to specify their precise architecture in the next chapter. In Theorem 1 from [17], the authors prove that the method converges to the true solution under sufficient conditions. Theorem 2 in the same work proves that there exist neural networks $u, \{v_k\}_{k=0}^{K-1}$ that approximate the true solution, with a number of parameters that grows at most polynomially in dimension and the sought accuracy, which implies that the Deep BSDE method can represent solutions practically without suffering from the curse of dimensionality.

5.2 A Sub-Problem Approach

Before incorporating observations to define the filter, we introduce the so called sub-problem approach of the Deep BSDE method. The sub-problem approach will help us incorporate observations, *i.e.*, creating the Deep BSDE Filter, which is the main purpose of this thesis to evaluate, however the derivation of DBSDEF is first done in the next section. We will also see that the sub-problem approach is an alternative to the Deep BSDE method for when the distribution p_0 and X_T have small overlap.

Recall the partition introduced in Section 2.1, where $0 = t_0 < t_1 < t_2 \cdots < t_K = T$, $t_k = t_{k,0} < t_{k,1} < t_{k,2} \cdots < t_{k,N} = t_{k+1,0} = t_{k+1}$, and let $\tau = t_{k,n+1} - t_{k,n}$ be constant for all k, n. By solving the Deep BSDE separately over each of the K coarser partitions, the overall problem is decomposed into K sub-problems, with each sub-problem propagating N steps in the finer partition. The first sub-problem

is formulated as

$$
\begin{aligned}
&\text{minimise}_{u_0,(v_{0,n})_{n=0}^{N-1}\in C(\mathbb{R}^d;\mathbb{R})\times C(\mathbb{R}^d;\mathbb{R}^d)^{\times N}}\mathbb{E}\left[\left|\mathcal{Y}_{0,N}^{u_1,v}-p_0(\mathcal{X}_{0,N})\right|^2\right]\\
&\mathcal{Y}_{0,0}^{u_0,v}=u_0(\mathcal{X}_{0,0}),\\
&\mathcal{Z}_{0,n}^{u_0,v}=v_{0,n}(\mathcal{X}_{0,n})\sigma(\mathcal{X}_{0,n}), && n=0,\dots,N-1,\\
&\mathcal{X}_{0,n+1}=\mathcal{X}_{0,n}+\tau\mu(\mathcal{X}_{0,n})+\sigma(\mathcal{X}_{0,n})\Delta W_{t_{0,n}}, && n=0,\dots,N-1,\\
&\mathcal{Y}_{0,n+1}^{u_0,v}=\mathcal{Y}_{0,n}^{u_0,v}-\tau f\left(\mathcal{X}_{0,n},\mathcal{Y}_{0,n}^{u_0,v},v_{0,n}(\mathcal{X}_{0,n})\right)+\mathcal{Z}_{0,n}^{u_0,v}\Delta W_{t_{0,n}}, && n=0,\dots,N-1.
\end{aligned}
\tag{5.26}
$$

It is now worth noting that the parametrized function u_0 no longer approximates $q(0,X_0)=p(T,X_0)$ which is the quantity of interest, but rather $q(T-t_1)=p(t_1)$. To approximate p_T we need to solve for the next $K-1$ sub-problems, using the previous sub-problem solution as the target function in the objective. Sub-problem 2 would then have target function u_0, which was trained in sub-problem 1 and hopefully approximates p_{t_1} well enough. For $k=1,\dots,K-1$, having solved sub-problem k with the minimizing pair $(u_k^*,(v_{k,n}^*)_{n=0}^{N-1})$, we formulate solving sub-problem k as

$$
\begin{aligned}
&\text{minimise}_{u_k\in C(\mathbb{R}^d;\mathbb{R}),\, v=(v_{k,n})_{n=0}^{N-1}\subset C(\mathbb{R}^d;\mathbb{R}^d)}\mathbb{E}\left[\left|\mathcal{Y}_{k,N}^{u_k,v}-u_{k-1}^*(\mathcal{X}_{k,N})\right|^2\right],\\
&\mathcal{Y}_{k,0}^{u_k,v}=u_k(\mathcal{X}_{k,0}),\\
&\mathcal{Z}_{k,n}^{u_k,v}=v_{k,n}(\mathcal{X}_{k,n})\sigma(\mathcal{X}_{k,n}), && n=0,\dots,N-1,\\
&\mathcal{X}_{k,n+1}=\mathcal{X}_{k,n}+\tau\mu(\mathcal{X}_{k,n})+\sigma(\mathcal{X}_{k,n})\Delta W_{t_{k+1,n}}, && n=0,\dots,N-1,\\
&\mathcal{Y}_{k,n+1}^{u_k,v}=\mathcal{Y}_{k,n}^{u_k,v}-\tau f\left(\mathcal{X}_{k,n},\mathcal{Y}_{k,n}^{u_k,v},v_{k,n}(\mathcal{X}_{k,n})\right)+\mathcal{Z}_{k,n}^{u_k,v}\Delta W_{t_{k,n}}, && n=0,\dots,N-1.
\end{aligned}
\tag{5.27}
$$

After solving K sub-problems, we have found a u_{K-1}^* that approximates the quantity of interest $p(T,X_0)$. The learned functions $(v_{k,n}^*)_{k=0,n=0}^{K-1,N-1}$, are only used as instruments to approximate the function p_T. The convergence rates of the sub-problems will be analysed in the first section in the results chapter.

Lastly, something needs to be said about the backwards nature of the process X_t and how it impacts the training of the neural networks numerically. In the Deep BSDE notation we approximate X_t forwards in time ending with the approximation $\mathcal{X}_K$ of X_T. Then we evaluate $\mathcal{X}_K$ with the known function p_0, and therefore it is preferable that the distribution of X_T and p_0 has the largest possible intersection. In the first section of Chapter 7, we demonstrate numerically how an insufficient intersection makes the Deep BSDE method perform poorly. However there could be enough intersection between the distributions p_0 and X_{t_1} and therefore the sub-problem approach becomes an alternative to deal with these scenarios. The results from this idea are also presented in the first section of Chapter 7. The idea is illustrated in Figure 5.1, where there is a small intersection between p_0 and p_1 but a larger intersection between each intermediate distribution $p_{0.25},p_{0.5},p_{0.75}$. As will be shown, this is an example the Deep BSDE method struggles with.

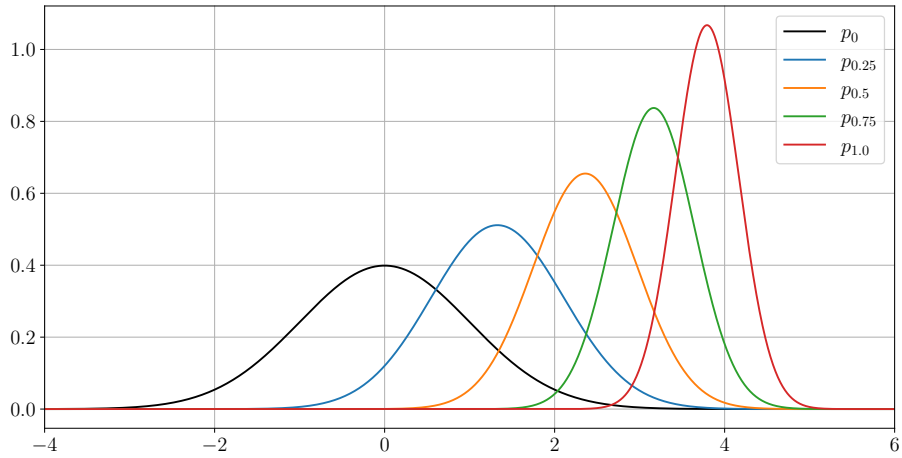

Figure 5.1: Closed form distributions of an Ornstein–Uhlenbeck process with initial distribution $\mathcal{N}(0,1)$ and long-term mean $\mu = 5$ at times $t = 0, 0.25, 0.5, 0.75, 1.0$.

5.3 The Deep BSDE Filter

In this section we extend the sub-problem approach to an approximative Bayesian filter that we call the Deep BSDE filter which is the main objective of this thesis to evaluate. Recall the discrete observation process $\{\mathbb{O}_k\}_{k=1}^K$ taking values in $\mathbb{R}^{d'}$ from a measurement model

$$\mathbb{O}_k = h(S_{t_k}) + V_k, \quad k = 1, \dots, K, \tag{5.28}$$

where $V_k \sim \mathcal{N}(0, R)$ for a covariance matrix $R \in \mathbb{R}^{d'}$. Similarly to in Section 2.1, we denote several measurements at times $n, n+1, \dots, k$ as the matrix $O_{n:k}$, which then has the shape $k - n + 1 \times d'$. We have that the likelihood for an observation is a measurable function $L : \mathbb{R}^{d'} \times \mathbb{R}^d \to \mathbb{R}$ and note that since the noise is Gaussian we have $L(O_k, x) = \mathcal{N}(O_k \mid h(x), R) = \mathcal{N}(h(x) \mid O_k, R)$.

Since there are K observations we create K sub-problems each solving the Bayesian filtering equations 4.0.1 for their respective time intervals. The solutions to the Bayesian filtering equations (4.0.1) of the k-th sub-problem are denoted

$$p_k(S_{t_k} = x \mid \mathbb{O}_{1:k-1} = O_{1:k-1}) = p_k(t_k, x \mid O_{1:k-1}),$$

and

$$p_k(S_{t_k} = x \mid \mathbb{O}_{1:k} = O_{1:k}) = p_k(t_k, x \mid O_{1:k}),$$

respectively. The first observation occurs at time t_1, hence during the interval $t \in [0, t_1)$ there are no observations available, denoted $O_{1:0}$. Therefore the Chapman–Kolmogorov equation 4.3 for the first sub-problem in the filter simplifies to

$$p_1(t_1, x, \mid O_{1:0}) = \int p_1(t_1, x, \mid O_{1:0}, S_0 = z)\, \mathrm{d}z, \tag{5.29}$$

which is just the probability distribution of the S at time t_1 and therefore the solution to the Fokker–Planck equation which we can use the Deep BSDE method to

obtain. Continuing to Bayes' equation (4.4) we obtain the first posterior distribution $p_1(t_1, x \mid O_{1:1})$ by multiplying with the likelihood of the observation $L(O_{1:1}, x)$ and dividing with the normalising constant c

$$p_1(t_1, x \mid O_{1:1}) = \frac{1}{c} p_1(t_1, x \mid O_{1:0}) L(O_1, x). \tag{5.30}$$

By using the posterior density obtained from (5.30) in the Chapman–Kolmogorov equation of the second sub-problem, we obtain

$$p_2(t_2, x \mid O_{1:1}) = \int p_2(t_2, x, \mid O_{1:1}, S_{t_1} = z) p_1(t_1, z \mid O_{1:1}) \, \mathrm{d}z, \tag{5.31}$$

and continuing directly to Bayes' equation for sub-problem 2

$$p_2(t_2, x \mid O_{1:2}) = \frac{1}{c} p_2(t_2, x \mid O_{1:1}) L(O_2, x). \tag{5.32}$$

However, here is a difference from the deterministic case. The probability distribution $p_2(t_2, x \mid O_{1:1})$ is clearly conditioned on the previous observation $O_{1:1}$, and the third distribution $p_3(t_3, x \mid O_{1:2})$ will be conditioned on both observations $O_{1:2}$. This issue is simply solved by giving the observations as inputs to the parametrized functions $(u_k, (v_{k,n})_{n=0}^{N-1})_{k=1}^{K}$, a detail we will return to.

First recall $p_0(t_0, x \mid O_{1:0}) = p_0(x)$. We have K filtering problems, one for each observation, for which we have one sub-problem each. By first solving the modified Fokker–Planck equation, we obtain the predictive distribution $p_{k+1}(t, x \mid O_{1:k})$

$$p_{k+1}(t, x \mid O_{1:k}) = p_k(t_k, x \mid O_{1:k}) + \int_{t_k}^{t} A^* p_k(t, s \mid O_{1:k}) \, \mathrm{d}s, \quad t \in (t_k, t_{k+1}]. \tag{5.33}$$

The modified Fokker–Planck equation that incorporates observations appears as a direct consequence of Bayes' theorem and it can be seen be used by many others, for instance [10, 7]. Like in (5.32), we now use the predictive distribution and the likelihood of the observation $L(O_{k+1}, x)$ to calculate the posterior distribution $p_{k+1}(t_{k+1}, x \mid O_{1:k+1})$

$$p_{k+1}(t_{k+1}, x \mid O_{1:k+1}) = \frac{1}{c} p_{k+1}(t_{k+1}, x \mid O_{1:k}) L(O_{k+1}, x), \quad k = 0, \ldots, K-1, \tag{5.34}$$

where c is the constant such that $p_{k+1}(t_{k+1}, x, O_{1:k+1})$ is a probability density. The normalisation constant c is calculated over each sample and how this is done depends on several factors, for instance state-dimension, therefore we return to normalisation later in the section. As already discussed, the measurement dynamics h, R are known, so from $p_k(t_k, x \mid O_{1:k-1})$ we can find $p_k(t_k, x \mid O_{1:k})$. It remains only to solve for $p_{k+1}(\cdot)$ in (5.33), which as already mentioned, was done in Section 5.2. However, in the filtering case, there is a new target function and previous observations will

be passed as inputs to the neural networks $(u_k, (v_{k,n})_{n=0}^{N-1})_{k=1}^{K}$ to incorporate the conditioning needed in the modified Fokker–Planck equation (5.33).

In the sub-problem approach we had that the optimal u_k^* approximated $p(t_k, X_{t_k})$. As eluded to, we now give the previous observations $O_{1:k-1}$ as input to $u_k^*(x, O_{1:k-1})$ which then approximates $p_k(t, x \mid O_{1:k-1})$. By multiplying $u_k^*(x, O_{1:k-1})$ with the likelihood $L(O_k, x)$ and dividing with the approximated normalisation constant $\hat{c}$, the approximation of $p_k(t_k, x \mid O_{1:k})$ is

$$p_k(t_k, x, O_{1:k}) \approx \frac{1}{\hat{c}} u_{k-1}^*(x, O_{1:k-1}) L(O_k, x), \quad k = 1, \dots, K. \tag{5.35}$$

Defining the target function as $g_k(x, o_{1:k}) = \frac{1}{\hat{c}} u_k^*(x, o_{1:k-1}) L(o_k, x)$ for $k = 1, \dots, K$ and $g_0(x, o_{1:0}) = p_{S_0}(x)$ for $k = 0$, we get for $k = 0, \dots, K-1$, the following problem

$$\begin{aligned}
&\text{minimise}_{u,v \in C(\mathbb{R}^d \times \mathbb{R}^{d' \times (k-1)}; \mathbb{R}) \times C(\mathbb{R}^d \times \mathbb{R}^{d' \times (k-1)}; \mathbb{R}^d)^{\times N}} \mathbb{E}\left[\left| \mathcal{Y}_{k,N}^{u,v} - g_k(\mathcal{X}_{k,N}, O_{1:k}) \right|^2 \right], \\
&\mathcal{Y}_{k,0}^{u,v} = u_k(\mathcal{X}_{k,0}, O_{1:k}), \\
&\mathcal{Z}_{k,n}^{u,v} = v_{k,n}(\mathcal{X}_{k,n}, O_{1:k}) \sigma(\mathcal{X}_{k,n}), && n = 0, \dots, N-1, \\
&\mathcal{X}_{k,n+1} = \mathcal{X}_{k,n} + \tau \mu(\mathcal{X}_{k,n}) + \sigma(\mathcal{X}_{k,n}) \Delta W_{t_n}, && n = 0, \dots, N-1, \\
&\mathcal{Y}_{k,n+1}^{u,v} = \mathcal{Y}_{k,n}^{u,v} - \tau f\left(\mathcal{X}_{k,n}, \mathcal{Y}_{k,n}^{u,v}, v_{k,n}(\mathcal{X}_{k,n})\right) + \mathcal{Z}_{k,n}^{u,v} \Delta W_{t_n}, && n = 0, \dots, N-1.
\end{aligned} \tag{5.36}$$

Now we return to how the normalisation constant c is calculated for each sample. Introducing the superscript m to denote sample m, then $c_{1:k}^{(m)}$ is the normalisation constant for the observation chain $O_{1:k}^{(m)}$

$$c_{1:k}^{(m)} = \int_{\mathbb{R}^d} L(O_{1:k}^{(m)}, x)\, u_k(x, O_{1:k-1}^{(m)})\, \mathrm{d}x, \quad m = 1, \dots, M. \tag{5.37}$$

In one dimension, the integral in (5.37) is approximated using simple grid-based techniques. However, this requires that the majority of the probability mass of the underlying process remains on the grid during the time frame of interest. This approach works well for stationary processes like the Ornstein–Uhlenbeck process, where the long-term mean lies well within the grid, but performs poorly in many other cases. Additionally, grid based normalisation methods scale exponentially in dimension and are hence unfeasible to apply in any other dimension but $d = 1$. To circumvent this issue we use importance sampling from some distribution $\mathcal{N}(x \mid \mu^{(m)}, \Sigma^{(m)})$. We know from Theorem 4.2.1 that we can obtain an exact expression of $c_{1:k}^{(m)}$ by

$$c_{1:k}^{(m)} = \mathbb{E}_{X' \sim \mathcal{N}(\mu^{(m)}, \Sigma^{(m)})} \left[\frac{L(O_{1:k}^{(m)}, X')\, p_k(t_k, X' \mid O_{1:k-1}^{(m)})}{\mathcal{N}(X' \mid \mu^{(m)}, \Sigma^{(m)})} \right]. \tag{5.38}$$

Approximating the expectation (5.38) by Monte Carlo sampling, we draw I samples $(x_i)_{i=0}^{I}$ from $\mathcal{N}(x \mid \mu^{(m)}, \Sigma^{(m)})$, and then get

$$\hat{c}_{1:k}^{(m)} = \frac{1}{I} \sum_{i=1}^{I} \left[\frac{L(O_{1:k}^{(m)}, x_i')\, p_k(t_k, x_i' \mid O_{1:k-1}^{(m)})}{\mathcal{N}(x_i' \mid \mu^{(m)}, \Sigma^{(m)})} \right]. \tag{5.39}$$

There are many methods to obtain $\mu^{(m)}$ and $\Sigma^{(m)}$. In this thesis however, we will let $\mu^{(m)}$ and $\Sigma^{(m)}$ be given by the Kalman filter or the Extended Kalman filter depending on the particular dynamics at hand.

6
Implementation and Evaluation

In this chapter we specify how the implementation, training and evaluation of the DBSDEF was made. Firstly, we present the specific neural network architectures that were used followed by a short motivation and discussion. Next, we describe how the DBSDEF's neural networks were trained to ensure the networks converge effectively. We end the chapter with a description and motivation of the metrics used for evaluating the convergence of the sub-problem approach and the Deep BSDE filter.

6.1 Neural Network Architecture

The functions that we seek to approximate are not particularly advanced, but rather simple and smooth. Therefore a simple fully connected neural network has been preferred for the task in the thesis. It has been deemed a good trade-off between network expressive power and speed. The choice of a FCNN can also be interpreted as an application of Occam's razor, the principle that the simplest solution is likely the best, making it a natural and sensible starting point for evaluating the method. Next, we examine the architecture of the neural networks in detail.

Let k be an integer such that $k \in [1, K]$. We recall that the network u is used to approximate the distribution $p_k(t_k)$, which is the filter distribution at the end of the k-the interval in the coarser partition $0 = t_0 < t_1 < t_2 \cdots < t_K = T$. We also recall that the networks $(v_n)_{n=0}^{N-1}$ are used to approximate $\nabla_x q(t_{k,n}))_{n=0}^{N-1}$. Since the networks $(v_n)_{n=0}^{N-1}$ are merely approximating the increments of the scheme we choose these networks to have fewer neurons than the network we choose for u, which acts as the initial point of the scheme. The exact architecture used for the u network and v networks can be seen in Table 6.1 and Table 6.2 respectively. To introduce nonlinearity, a ReLU activation function is applied after each hidden layer for both u and v. To ensure that layers produce non-zero activations even when the inputs are zero, we add learnable bias terms to each layer as an offset. An element wise exponential transformation is applied at the end of the u network since this ensures positive outputs, which is desired since u approximates a probability distribution. A natural development of the architecture would be to ensure Gaussian-like tails through additional transformation of the output. Since no issues with diverging tails were encountered, this was not explored in the thesis. However, the method was tested in [5].

In the deterministic case, we have that the networks in each sub-problem have the same input dimension. As observations are incorporated, the input dimension of each u increases by the observation dimension d' with each solved sub-problem. In the deterministic case, this corresponds to $k = 0$, while in the filtering case, k denotes the index of the k-th sub-problem.

Table 6.1: Architecture of the neural network u_k.

Layer	**Type**	**Input Dim.**	**Output Dim.**
Input	Linear + ReLU	$d + d' \times k$	128
Hidden 1	Linear + ReLU	128	128
Hidden 2	Linear + ReLU	128	128
Hidden 3	Linear + ReLU	128	128
Output	Linear	128	1
Output transformation	Exponentiation	1	1

Table 6.2: Architecture of the neural networks $(v_{k,n})_{n=0}^{N-1}$.

Layer	**Type**	**Input Dim.**	**Output Dim.**
Input	Linear + ReLU	$d + d' \times k$	32
Hidden 1	Linear + ReLU	32	32
Hidden 2	Linear + ReLU	32	32
Output	Linear	32	d

It is important to note that the motivation for the use of these architectures in the context of Deep BSDE filtering is highly empirical and that it is beyond the scope of this thesis to in detail investigate different neural network architectures. There are architectures with potential benefits for instance, using a Convolutional Neural Network since it can learn relationships in multidimensional inputs that have temporal causation. Newer architectures, such as Transformers could also be worthwhile testing. They also learn temporal relationship by treating time steps as tokens and using self-attention with positional encodings and multi-heads to capture dependencies across any range in parallel [35].

6.2 Training

The neural networks in Section 6.1 were implemented in Python using the standard PyTorch library. As previously stated in Section 2.5, the optimizer ADAM was used. Recalling Section 5.1, the objective is measured in the expectation of the squared error between the target function p_0 and $\mathcal{Y}_K^{u,v}$.

The data, consisting of samples of $\mathcal{X}$ and O, was synthetically generated with the Euler–Maruyama method which was outlined in Section 2.4. In each epoch, new data was generated, and consequently the models does not overfit since the data is

always unseen. Independence of samples is assumed to hold from the pseudorandom number generator sampling the Brownian motion in the Euler-Maruyama scheme and independence between epochs follows from the assumption of independence between samples. Therefore, no measures to mitigate overfitting, such as weight decay or regularization were used. To optimize convergence speed, a reduce on plateau learning rate scheme was used. The learning rate was reduced by a factor of 10 if the epoch loss did not improve between each epoch. In practice, this was done by using PyTorch's ReduceLROnPlateau scheduler with a patience of 1, the initial learning rate is always set to 0.001.

To further improve convergence time and not waste compute resources, an early stopping mechanism was used. Once the learning rate of 10^{-8} was reached, five more epochs was trained before stopping. Any learning rate at or below 10^{-8} yielded in all experiments no meaningful improvements. All models were trained for a maximum of 100 epochs, a limit most of the models never reached. Regardless if the models early stopped or not, the data was always generated for 100 epochs. This was done since the next model could continue on for longer than the previous model, which then needed unseen data for more epochs. In the deterministic cases, a data set of 10^7 samples was generated for each epoch and for the filter cases 10^6 samples. A batch size of 10^3 was used in all cases, leading to 10^4 and 10^3 gradient updates per epoch in the deterministic case and the filter case respectively. The reason for the smaller datasets in the filter case is because of the computational cost of normalizing the filtering density to obtain the target function before each epoch. How the normalization was implemented was specified at the end of Section 5.3. It should also be noted that an example of the sub-problem approach with 10^4 samples per epoch is included in Appendix A. Samples, gradient updates and batch size are highly dependent on the hardware at hand. The quantities chosen in this thesis effectively balanced GPU utilization, convergence and total runtime. For different hardware, these quantities could differ significantly, therefore we present the hardware used in this thesis next.

The computations for the deterministic sub-problem were done on an Nvidia A40 GPU with 45 GB of VRAM memory and the computations for DBSDEF were done on an Nvidia A100 GPU with 40 GB of VRAM memory. The Nvidia A100 is approximately twice as fast compared to the A40 on 16- floating point precision. The additional compute power was needed for the DBSDEF, mainly due to the computationally intensive normalization required for each individual sample. Since the data was generated during each epoch, the computations were not memory constrained. The GPU was paired with an Intel(R) Xeon(R) Gold 6338 CPU or a Platinum 8358 CPU. The particle filters were calculated on a CPU. The deterministic sub-problem approach took approximately $12-24$ hours depending on the example while the DBSDEF training runs took $12-60$ hours. In the DBSDEF case, the shorter runs were in one dimension with the easier problem of applying the DBSDEF on an Ornstein–Uhlenbeck process, while the longer runs were in the high-dimensional spring mass system case. Computing a particle filter of 10^5 particles in one dimension took approximately 2 hours. The computations were enabled by resources provided by

Chalmers e-Commons at Chalmers University of Technology.

Lastly, as a side note, it could be worthwhile exploring other loss functions. Since the values in the target function are either from a probability distribution or an estimation of a probability distribution, we likely have small values. Hence, one could motivate to test how the Mean Absolute Error (MAE) works since it is more robust to outliers and could improve convergence speed. However, this is empirical, and to theoretically motivate the use of the MAE loss one would have to argue that a small MSE loss leads to a small MAE loss, *i.e.* a form of bound.

6.3 Method Evaluation

In this section, we present the methods that are used to evaluate the sub-problem approach and the Deep BSDE filter. To evaluate the convergence of the sub-problem approach we use two metrics over the normed probability space $L^2(\mathbb{R}^d;\mathbb{R})$ with the norm $||f||_{L^2(\mathbb{R}^d;\mathbb{R})} = \sqrt{\int_{\mathbb{R}^d} |f(x)|^2 \,\mathrm{d}x}$. They are the L^2- and the L^∞-error, and are the average and the supremum of the difference between the approximative probability distribution and the reference probability distribution in the probability space respectively. These metrics needs to be extended with the observation space to be useful in the stochastic setting. In this case, the errors are measured in the normed probability space $L^2(\Omega; L^2(\mathbb{R}^d;\mathbb{R}))$. The combination of the two spaces has the norm $||X||_{L^2(\Omega;L^2(\mathbb{R}^d;\mathbb{R}))} = \sqrt{E[||X||^2_{L^2(\mathbb{R}^d;\mathbb{R})}]}$. In this space the metrics L^2L^∞- and the L^2L^2-error are used to evaluate the convergence of the DBSDEF. The L^2L^∞-error measures the maximum deviation of the approximation from the true probability distribution, while the L^2L^2-error measures the average deviation of the approximation from the true probability distribution, where the supremum and average are taken over the inner L^2 space respectively.

Since the sub-problem approach is only tested for the Ornstein–Uhlenbeck process, its closed form distribution at time t_k conditioned on the starting distribution p_0 is what it is evaluated against. Let p_k be the closed form probability distribution function of the SDE at time t_k and let $\widehat{p}_k$ be the approximation of p_k, which in our case will be the trained neural network u_k^*. We then have that the L^∞- and L^2-error are given by

$$L^\infty\text{-error}_k = \sup_{x\in\mathbb{R}^d} \left|p_k(x) - \widehat{p}_k(x)\right|, \tag{6.1}$$

$$L^2\text{-error}_k = \sqrt{\int_{\mathbb{R}^d} \left|p_k(x) - \widehat{p}_k(x)\right|^2 \,\mathrm{d}x}. \tag{6.2}$$

In the case of the one dimensional deterministic sub-problem approach, it is simple to evaluate the metrics (6.1)-(6.2). A grid based technique was used on a grid of 10^3 evenly spaced points on the interval $[-10, 10]$. Let $E = \{x_i\}_{i=0}^N$ denote the sorted grid points. The supremum in (6.1) is approximated by taking the maximum over the grid E. Define $\Delta x_i = x_{i+1} - x_i$ and $d_i = \left(p_k(x_i) - \widehat{p}_k(x_i)\right)^2$, then, the integral in (6.2) is approximated by

$$\int_{\mathbb{R}^d} |p_k(x) - \widehat{p}_k(x)|^2 \, \mathrm{d}x \approx \frac{1}{x_N - x_0} \, \frac{1}{N} \sum_{i=0}^{N-1} \frac{\Delta x_i}{2}(d_i + d_{i+1}). \tag{6.3}$$

For grid based calculations as in (6.3) it is important that the majority of the probability mass distribution of the process lies mostly within the interval of the grid. Otherwise, the approximation likely loses most of its accuracy. Hence, it is important that the process is relatively concentrated to one area.

For the filter case, the stochastic behaviour of each observation motivates the extension of the metrics for the L^2- and L^∞-error by an additional observation space. In this space, the L^2L^∞-error and the L^2L^2-error are calculated instead. The main difference is that, rather than computing the error for a single sample, the error is calculated across multiple samples, averaged, and then the square root is taken. This is a Monte Carlo method, a concept we recall from Section 2.4. Now we let p_k^m denote the reference filter density of the the sample m at time t_k and let $\widehat{p}_k^m$ denote the approximative filter we seek to calculate the metric for. Then the L^2L^∞- and the L^2L^2-error are approximated with the Monte Carlo method as follows

$$L^2L^\infty\text{-error}_k \approx \sqrt{\frac{1}{M'} \sum_{m=1}^{M'} \sup_{x \in \mathbb{R}^d} |p_k^m(x) - \widehat{p}_k^m(x)|^2}, \quad k \in \{0, \ldots, K\}, \tag{6.4}$$

$$L^2L^2\text{-error}_k \approx \sqrt{\frac{1}{M'} \sum_{m=1}^{M'} \int_{x \in \mathbb{R}^d} |p_k^m(x) - \widehat{p}_k^m(x)|^2 \, \mathrm{d}x}, \quad k \in \{0, \ldots, K\}. \tag{6.5}$$

When the underlying process S is one dimensional, the integral

$$\int_{x \in \mathbb{R}^d} |p_k^m(x) - \widehat{p}_k^m(x)|^2 \, \mathrm{d}x \tag{6.6}$$

is calculated over a grid for each observation sample m like in (6.3), but now with p_k^m and $\widehat{p}_k^m$ instead of p_k and $\widehat{p}_k$. As for the normalization in Section 5.3 grid based calculations becomes infeasible for $d > 2$ since the computational cost increases exponentially in dimension. This is true even if the process has a predictable long-term mean that will be within the interval of the grid. Therefore importance sampling is used from a distribution $q_k = \mathcal{N}(\mu_k^m, 2\Sigma_k^m)$, where μ_k^m and Σ_k^m are given by the Kalman filter or the Extended Kalman filter for the observation sample m at time t_k.

We employ the Monte Carlo method to approximate the right hand side of (6.7) over sufficiently many samples from the distribution q_k^m. The supremum in (6.4) is approximated by taking the supremum over the set of samples generated by importance sampling.

$$\int_{x \in \mathbb{R}^d} |p_k^m(x) - \widehat{p}_k^m(x)|^2 \, \mathrm{d}x = \mathbb{E}_{X \sim q_k^m} \left[\frac{|p_k^m(X) - \widehat{p}_k^m(X)|^2}{q_k^m(X)} \right]. \tag{6.7}$$

Lastly, it is crucial to select an appropriate reference filter p_k^m based on the specific setting. As explained in detail in Chapter 4, the Kalman filter provides a closed-form solution to the Bayesian filtering equations when the state and measurement dynamics are linear and Gaussian. Therefore, in Chapter 7 containing the numerical experiments, for two linear examples the Kalman filter is used as the reference solution. The first linear example is the Ornstein–Uhlenbeck and the second linear is a spring-mass system. An example of a bimodal SDE is also presented, which is not linear. As a result, the Extended Kalman Filter only offers an approximate solution. For this reason, a particle filter with 10^5 particles is used as a reference solution.

7
Results

In this chapter we present the results obtained from implementing the Deep BSDE filter. As the sub-problem approach is the main component in the Deep BSDE filter, a convergence analysis of it is first conducted. As we have seen in Chapter 5, the sub-problem approach does not incorporate observations which we call a deterministic setting. Its convergence is evaluated in three cases depending on how the long-term mean of the underlying Ornstein–Uhlenbeck process evolves. Then, we present the results of the Deep BSDE filter in three different cases. First the DBSDEF is applied on two one dimensional processes, an Ornstein–Uhlenbeck process and a nonlinear bimodal SDE. Then it is tested in a higher dimensional problem on a spring mass system with 2 masses where the model describes the position and velocity of each mass. In the Ornstein–Uhlenbeck and the spring-mass system examples, the reference filter is the optimal solution given by the Kalman filter while in the bimodal example it is a particle filter with 10^5 particles. Additionally, in each example, the solution from the DBSDEF is compared to particle filters. If not otherwise specified we will let $t_0 = 0$, $T = 1$ and $p_0 \sim \mathcal{N}(0, 1)$.

7.1 The Sub-Problem Approach

In this section we consider one dimensional Ornstein–Uhlenbeck (OU) processes, which are solutions to Langevian equations of the form

$$\mathrm{d}X_t = \theta(\mu - X_t)\ \mathrm{d}t + \sigma\ \mathrm{d}W_t, \quad t \in [0, T], \tag{7.1}$$

for some parameters $\mu, \theta \in \mathbb{R}, \sigma \in \mathbb{R}$. The OU process describes a mean-reverting process with long-term mean μ. The speed of mean reversion is governed by the parameter θ, while the volatility of the process is controlled by σ. Furthermore, we will consider three sets of parameters, $\mu_1 = 0$, $\mu_2 = 2.5$, $\mu_3 = 5$, $\theta_1 = \theta_2 = \theta_3 = 0.7$ and $\sigma_1 = \sigma_2 = \sigma_3 = 0.1$. Since $S_0 \sim p_0$ has mean 0 and we have different long-term means μ_1, μ_2, μ_3, the mean will evolve differently during the time interval $[0, T]$. In the case of $\mu_1 = 0$, the mean will not evolve, which we call *no evolution*. However, for μ_2 and μ_3, we will have an evolving mean from initial mean 0 to a long-term mean of 2.5 and 5 respectively. In the case of $\mu_2 = 2.5$, we say we have *moderate evolution*, while for $\mu_3 = 5$ we say we have *high evolution*. This study numerically examines whether the sub-problem approach mitigates convergence issues when the intersection of p_0 and the probability distribution of X_T has a small probability mass. Additionally, the intersection between distributions in the sub-problem approach is governed by the initial distribution of X_0, $\widehat{p}_0$, which is not necessarily p_0. However to simplify, we let $p_0 = \widehat{p}_0$. Another option to mitigate the convergence issues is to select a wider $\widehat{p}_0$, for instance $\widehat{p}_0 \sim \mathcal{N}(0, 5^2)$. This benefits the Deep BSDE method without the sub-problem split, since it creates a larger intersection between the distributions and hence making the optimization numerically easier.

We recall the time discretization introduced in Section 2.1 with K coarse time intervals $[t_k, t_{k+1}]$ and N finer intervals between each coarser interval $[t_{k,n}, t_{k,n+1}]$, making in total KN steps. In the context of the sub-problem approach, this means we have K sub-problems, each having N prediction steps. Initially we fix $KN = 32$, and vary the number of subproblems in 5 different discretization setups by considering $(K, N) = (2, 16), (4, 8), (8, 4), (16, 2), (32, 1)$. The obtained approximative solution at the time t_k is denoted $\widehat{p}_k^{K,N}$. They are compared with the Deep BSDE method with the L^2- and L^∞-error from Chapter 6 evaluated against the closed-form Ornstein–Uhlenbeck distribution conditioned on the initial distribution p_0. Then, for the moderate and high evolution cases, we continue to analyse the convergence of the sub-problem approach for $K = 4$ at different N. In the case of no evolution $\mu_1 = 0$, we omit this type of convergence analysis since it offers no additional insights.

7.1.1 OU 1: No Evolution

This example demonstrates the sub-problem approach applied on a Ornstein–Uhlenbeck process with parameters $\mu = 0$, $\theta = 0.7$ and $\sigma = 0.1$. The L^2- and L^∞- error for the sub-problem approach and the original Deep BSDE method are calculated over the time interval $[0, T]$ and are shown in Figure 7.1. When N increases, the number of sub-problems decreases since $KN = 32$ is constant. Therefore with larger N we have that the first sub-problem density is obtained at a later time point, which can be seen in the figure. For instance for $(K, N) = (2, 16)$, the first error is at the time $t = 0.5$. The solutions of the sub-problem approach perform similarly to each other, however $(K, N) = (2, 16)$ and $(K, N) = (4, 8)$ perform slightly better than $(K, N) = (16, 2)$ and $(K, N) = (32, 1)$. The original Deep BSDE method outperformed all setups of the sub-problem approach. As noted in Chapter 5, we know from [17] that the Deep BSDE method converges finer discretization and since it was the best method in the no evolution case, no further analysis of its convergence is made. Lastly, it is worth mentioning that as the sub-problems become more numerous, for instance $(K, N) = (32, 1)$ the training becomes unstable, which is seen in the spiking behaviour in Figure 7.1. In Appendix A we also include an example with 10^4 training samples per epoch compared to the 10^7 used throughout the sub-problem section in this chapter. It shows faster convergence for the sub-problem approach compared to the Deep BSDE method.

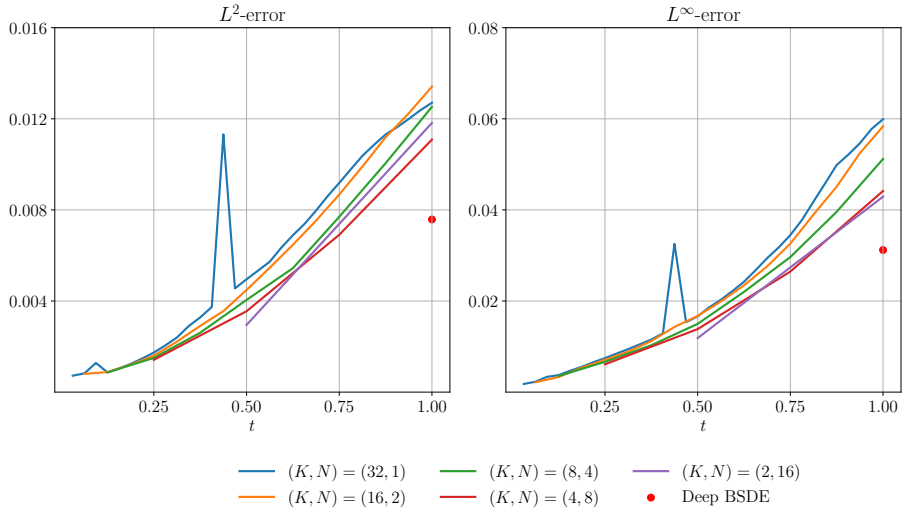

Figure 7.1: Comparison of L^2- and L^∞- error for the no evolution case.

7.1.2 OU 2: Moderate Evolution

In the moderate evolution case the mean increases from 0 at time $t = 0$ towards the long-term mean of $\mu = \mu_2 = 2.5$. The same sub-problem setups are analysed as in the previous example, namely $(K, N) = (2, 16), (4, 8), (8, 4), (16, 2), (32, 1)$. Figure 7.2 presents the true distribution p_T and the approximations at the end time $t = T = 1$ obtained from the sub-problem setups as well as from the Deep BSDE. Looking at the figure we see more variance in the accuracy of the solutions. It is clear that the $(K, N) = (32, 1), (16, 2)$ models have failed to converge, and rather diverged. This is probably due to error accumulation in each target function in each sub-problem. The Deep BSDE method has also failed to learn the distribution p_T, being flat instead. This is likely due to the discussed mismatch in distributions of p_0 and X_T, which is part of the motivation behind the sub-problem approach.

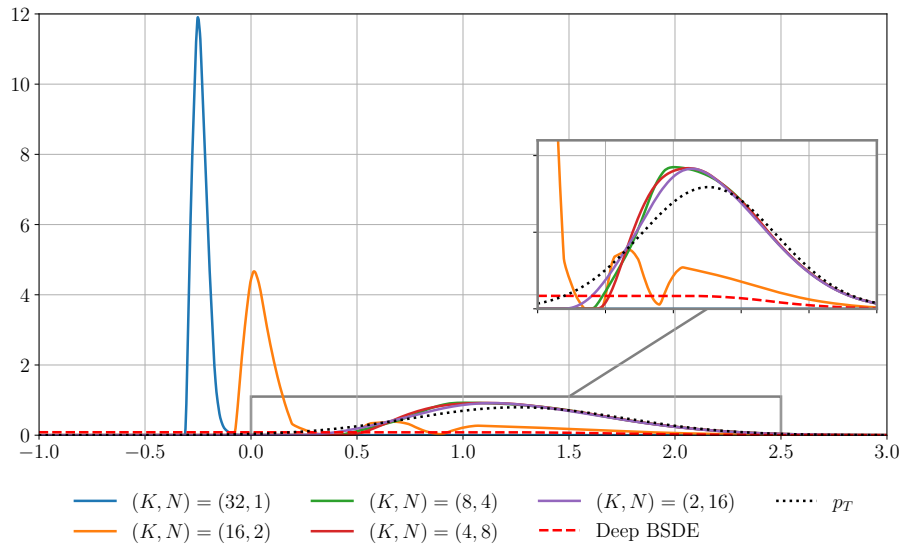

Figure 7.2: The sub-problem approach at $KN = 32$ for $N = 1, 2, 4, 8, 16$ compared to the Deep BSDE method at time $t = T = 1$, for the moderate evolution case.

The L^2- and L^∞- error in the time interval $t \in [0, T]$ are presented in Figure 7.3. The sub-problem approach is more accurate when $(K, N) = (8, 4), (4, 8), (2, 16)$ and looking at the errors in the figure it can be seen that they are similar. There appears to be a significant error accumulation for the sub-problem setups with many sub-problems, *i.e.* large K. This can be seen in the errors of $(K, N) = (32, 1), (16, 2)$ which is initially aligned with the other sub-problem setups, but diverges at $t \approx 0.4$ and $t \approx 0.6$ respectively.

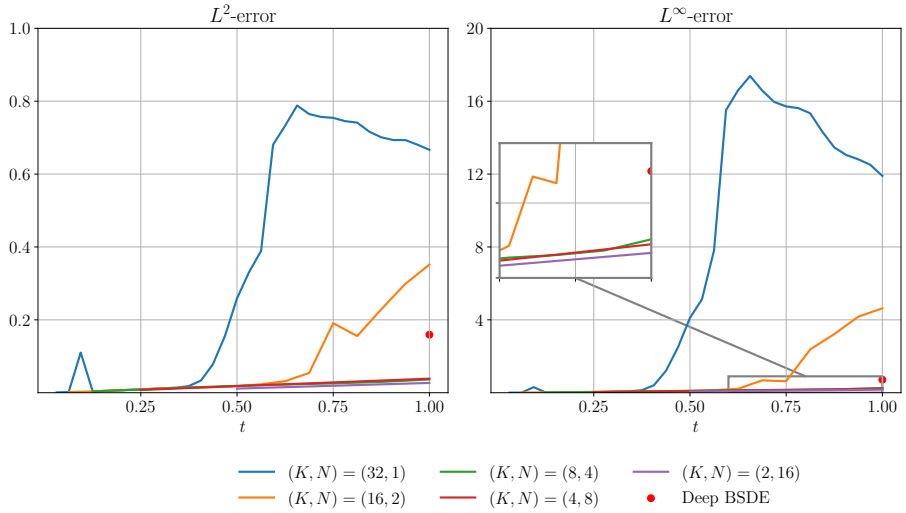

Figure 7.3: Comparison of L^2-error and L^∞-error for the moderate evolution case.

As $K = 4$ sub-problems seem to be enough in the moderate evolution case, we seek to analyse the convergence of the sub-problem approach for $K = 4$. This is done by running the sub-problem approach with $(K, N) = (4, 1), (4, 2), (4, 4), (4, 8), (4, 16), (4, 32)$. Their solutions are compared to the original Deep BSDE method at $N = 128$, which can be likened to the sub-problem approach with $(K, N) = (4, 32)$ since they have the same Δt. The resulting approximation obtained from the sub-problem setups and the original Deep BSDE, as well as the true distribution p_T can be seen in Figure 7.4. In the figure, it is apparent that for $(K, N) = (4, 2), (4, 4), (4, 8), (4, 16), (4, 32)$, the sub-problem approach outperforms the Deep BSDE method, and that we have convergence as N increases.

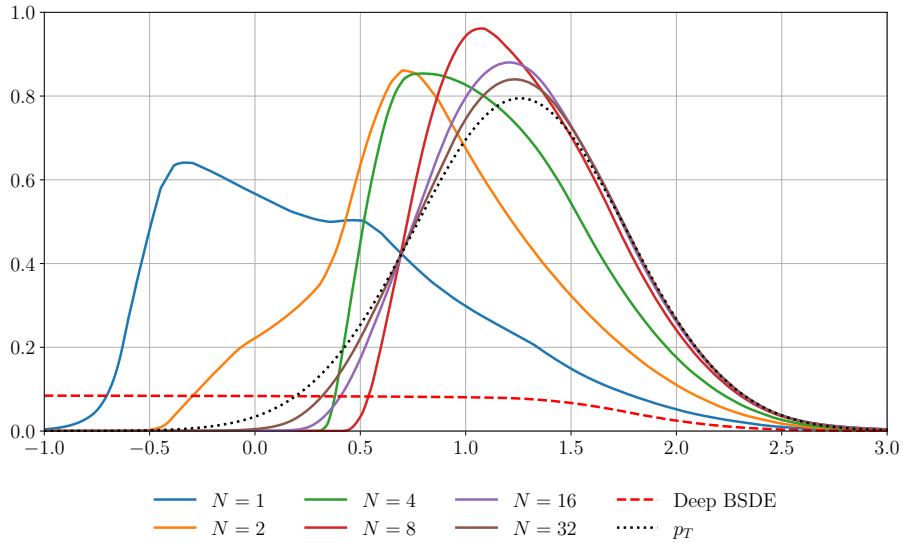

Figure 7.4: The sub-problem approach using $K = 4$, $N = 1, 2, 4, 8, 16, 32$ and the Deep BSDE method with $N = 128$, at $t = T = 1$, in the moderate evolution case.

In Figure 7.5, the errors over time are presented. Since there are 4 sub-problems,

the first distribution for all the sub-problem setups is obtained at the time $t = 0.25$, and hence the first L^2- and L^∞- error are also calculated at the time $t = 0.25$. Convergence can be seen in the figure for each increasing N at all time points, for both the metrics.

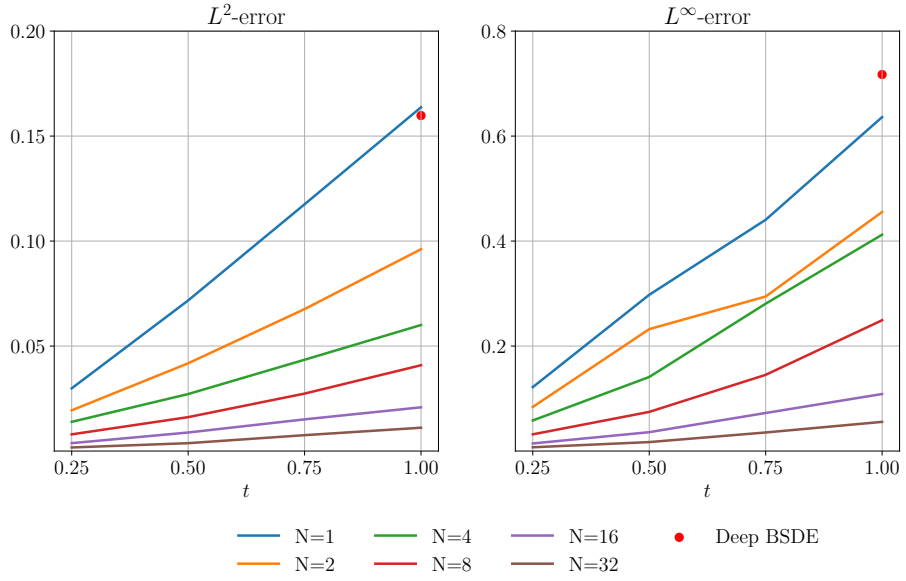

Figure 7.5: L^2-error and L^∞-error for the sub-problem approach with $K = 4$ for $N = 1, 2, 4, 8, 16, 32$ compared to the Deep BSDE method at $N = 128$.

In Figure 7.6 the L^2- and the L^∞- error at the end time $t = T = 1$ for different N are shown. The figure also includes $O(N^{-1})$ as a reference. The errors from the sub-problem approach setups decreases at approximately the same rate as the reference line which indicates that it has an approximative convergence rate at the end time of 1 in both norms.

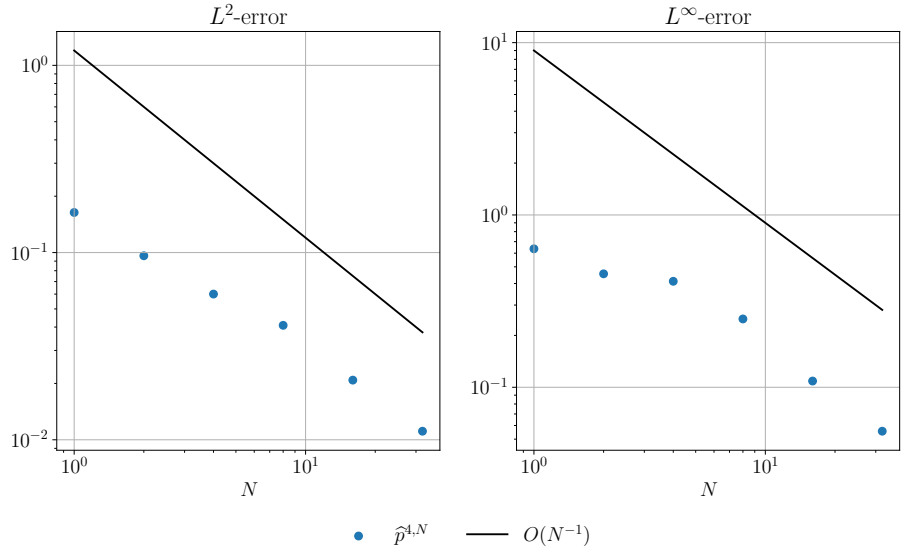

Figure 7.6: L^2- and L^∞- error for the sub-problem approach at $t = T = 1$ for $K = 4$ and $N = 1, 2, 4, 8, 16, 32$.

7.1.3 OU 3: High Evolution

In Section 5.2 we motivated why the original Deep BSDE method might perform poorly when the distribution p_0 has a small intersection with the distribution of X_T and that the sub-problem approach could help mitigate this issue, which is what this high evolution case aims to demonstrate. The same sub-problem setups with $KN = 32$ are used as in the previous two examples. Figure 7.7 shows the true distribution p_T and the approximations of the sub-problem setups and the Deep BSDE method at the end time $t = T = 1$. In the figure it can be seen that the Deep BSDE method has difficulty accurately learning the distribution p_T and is flat instead. The same can be said for $(K, N) = (2, 16)$, *i.e.* for the solution with two sub-problems, which also stays relatively flat. Considering that the Deep BSDE method failed in the moderate evolution case, the fact that the two sub-problem solution in the high-evolution case failed is not unexpected. For $(K, N) = (8, 4), (4, 8)$ we see better approximations, but still inaccurate ones. The solutions of $(K, N) = (32, 1), (16, 2)$ did not perform good either. This is likely due to significant error accumulation between each sub-problem, which was seen in the moderate evolution case as well.

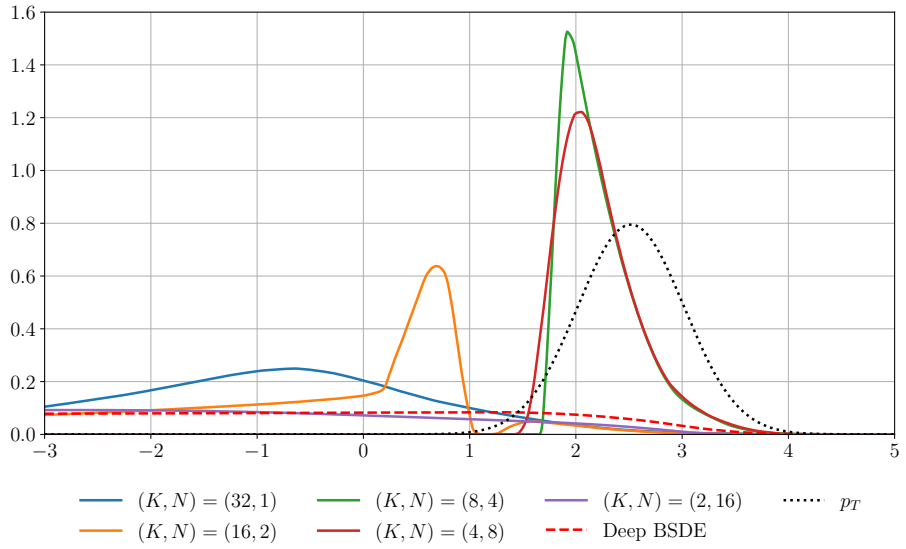

Figure 7.7: The sub-problem approach at $KN = 32$ for $N = 1, 2, 4, 8, 16$ compared to the Deep BSDE method at time $t = T = 1$, for the high evolution case.

The L^2- and L^∞- error over time are shown in Figure 7.8. The errors in the figure are hard to interpret since they are all substantial. The behaviour of $(K, N) = (32, 1), (16, 2)$ is unexpected, but can be explained. They evidently start off learning the true distributions accurately. After a number of sub-problems are trained, the error accumulation starts and we see the error metrics increase drastically. After this, the subsequent sub-problems learn from the wrong target functions, since the previous sub-problems diverged, and end up becoming flat everywhere, which can be seen for $(K, N) = (32, 1)$ in Figure 7.7. A flat distribution has a lower error than a diverged curve with the main modality centred in the wrong coordinate. This explains how the error first increases and then decreases. Nevertheless, it is clear

in Figure 7.7 that there are solutions that are more accurate than others, like the mentioned $(K, N) = (8, 4), (4, 8)$.

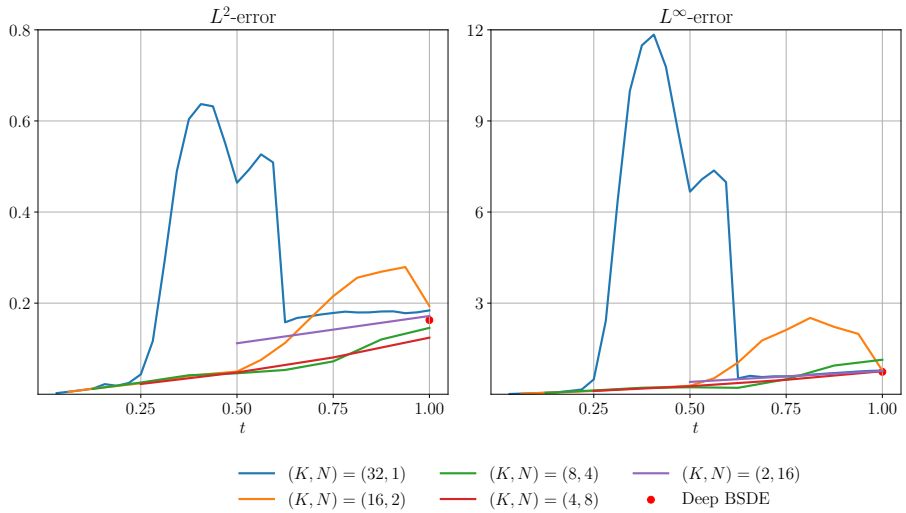

Figure 7.8: Comparison of L^2-error and L^∞-error for the high evolution case.

From the analysis done so far in the no, moderate and high evolution cases, the conclusion is that there is a trade-off between the number of sub-problems and error-accumulation. We want a large enough intersection between the distributions at hand without inducing error accumulation from too many sub-problems. It is not straight forward to choose the best combination, since it likely depends on many factors such as dataset size, the underlying process and the discretization KN. Further experiments need to be made to develop a framework to navigate this trade-off.

However, 4 sub-problems seem to be enough in the high evolution case and therefore the same analysis is conducted for the high evolution case as was done in the moderate evolution case. The model is trained with increasingly finer time discretization $(K, N) = (4, 1), (4, 2), (4, 4), (4, 8), (4, 16), (4, 32)$. Looking at Figure 7.9 we see that the sub-problem approach for $(K, N) = (4, 1), (4, 2), (4, 4)$ fails to approximate the distribution p_T. However, significant improvement can be seen at $(K, N) = (4, 8)$ and it further improves its accuracy for each discretization $(K, N) = (4, 16), (4, 32)$.

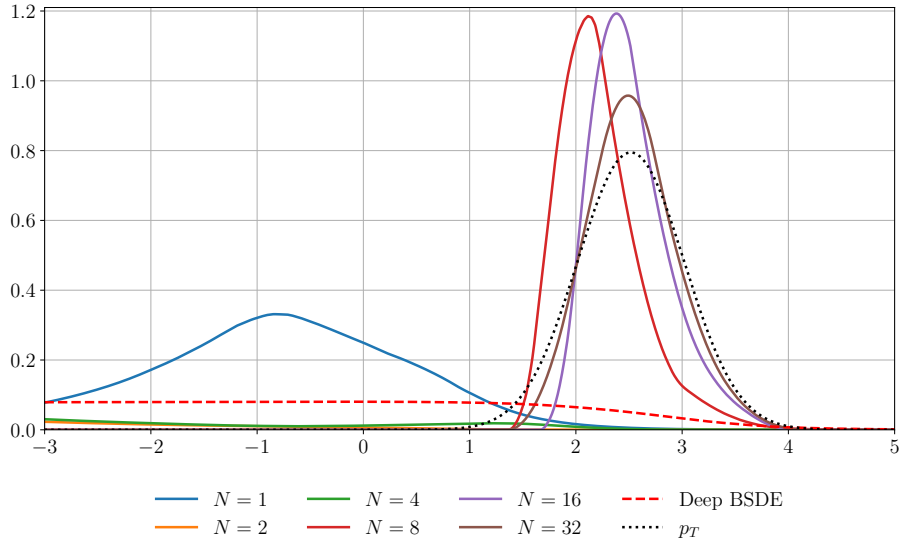

Figure 7.9: Results from the sub-problem approach using $K = 4$ sub-problems and $N = 1, 2, 4, 8, 16, 32$ compared to the Deep BSDE method with $N = 128$, at time $t = T = 1$, for the high evolution case.

The L^2- and the L^∞-error over time are shown in Figure 7.10. It is evident from the figure that there is convergence as the discretization becomes finer, *i.e.* N increases.

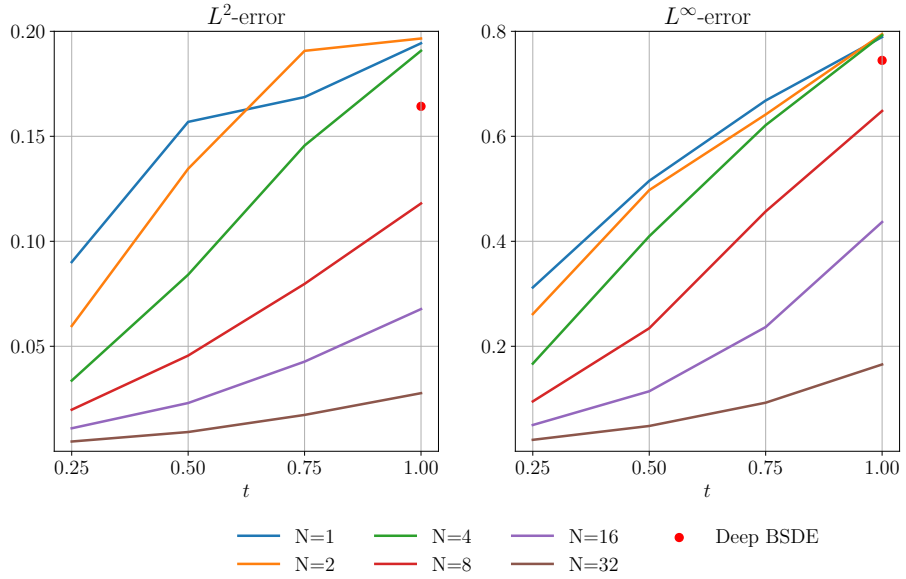

Figure 7.10: L^2-error and L^∞-error for the sub-problem approach with $K = 4$ for $N = 1, 2, 4, 8, 16, 32$ compared to the Deep BSDE method at $N = 128$.

In Figure 7.11, we see the convergence rate at the end time $t = T = 1$ with $O(N^{-1})$ as a reference. The convergence is not as obvious as in the moderate evolution case, however, from $N = 4$ a converging trend starts, which is also indicated by the patterns in figures 7.9-7.10. The later start of the convergence could be explained that the problem in the high evolution case is more challenging, and that

the sub-problem setups $(K, N) = (4, 1), (4, 2), (4, 4)$ are all insufficient to learn. The phenomenon that convergence starts later is not uncommon, often seen with approximations involving the Euler–Maruyama method.

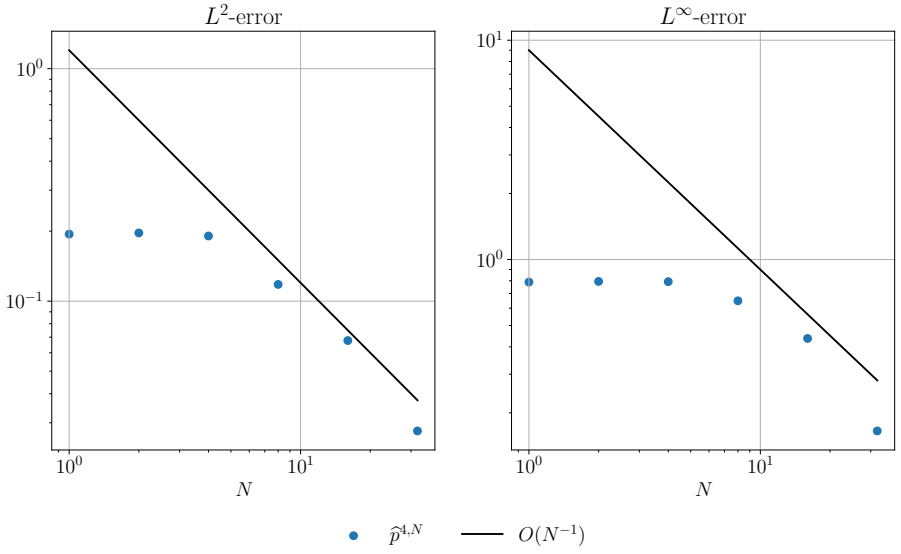

Figure 7.11: L^2- and L^∞- error for the sub-problem approach at $t = T = 1$ for $K = 4$ and $N = 1, 2, 4, 8, 16, 32$.

To summarize the results from the sub-problem approach we note that there is a trade-off between number of sub-problems and error accumulation. The no, moderate and high evolution cases have exemplified that the best solution in one case might not be the best solution in another. As mentioned in the no evolution case in Section 7.1.1, the sub-problem approach might converge faster. We can therefore only conclude that the sub-problem approach has the potential to outperform the Deep BSDE, but that the number of sub-problems needs to be chosen carefully since increasing the number of sub-problems is not necessarily preferable as it introduces additional error accumulation. In the case of no evolution, it was even clear that it is not always necessary to introduce any sub-problems, as the original Deep BSDE method performed the best. Lastly, the sub-problem approach converged for $K = 4$ in the moderate and high evolution, approximately up to order 1 in both norms. The convergence of the sub-problem approach verifies it as a sensible component in the Deep BSDE filter, which is analysed next in three examples.

7.2 Ornstein–Uhlenbeck Filtering

In this section, the Deep BSDE filter is applied on the same one dimensional OU process as in Section 7.1 with the no evolution case. That is, we have an OU process with parameters $\mu = 0, \theta = 0.7, \sigma = 0.1$. Introducing observations, we let the measurement model be $h(x) = x$, and the additive noise be Gaussian with mean 0 and variance 1. The reference filter is the optimal solution given by the Kalman filter. The DBSDEF is compared with particle filters having $10^2, 10^3$, 10^4 and 10^5 particles. A Monte Carlo sample of 10^3 solutions was made to calculate the L^2L^2- and L^2L^∞-error, which were introduced in Chapter 6. Six models were trained with different number of prediction steps $N = 1, 2, 4, 8, 16, 32$, with each model taking 10 observations. Since the state-dimension is 1 and we have a stationary long-term mean, grid based normalization was applied during the training with a grid of 1000 evenly spaced points in the interval $[-10, 10]$.

How the models' errors develop over time is shown in Figure 7.12. As can be seen in the figure, there is convergence as the number of prediction steps per observation increases. It is also noteworthy that the error stays flat at first, only to start increasing later at an inflection point. The inflection point for each model comes later for larger N, indicating that larger N leads to a flat error for a longer time. This can be seen by looking at $t = 0.4$ in Figure 7.12 where the models with $N = 4, 8, 16, 32$ all have relatively low error while the models with $N = 1, 2$ have already started diverging.

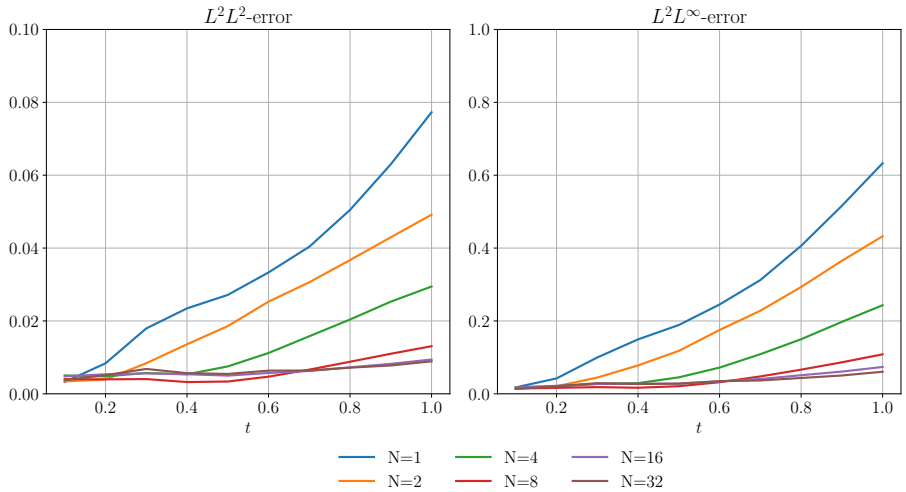

Figure 7.12: The L^2L^2- and L^2L^∞-error for the DBSDEF in the Ornstein–Uhlenbeck example.

The models' errors at time $t = T = 1$ are shown in 7.13, where a reference line is included for comparison. As can be seen in the figure, it is reasonable to think that DBSDEF probably converges with at least a rate of $O(N^{-1/2})$.

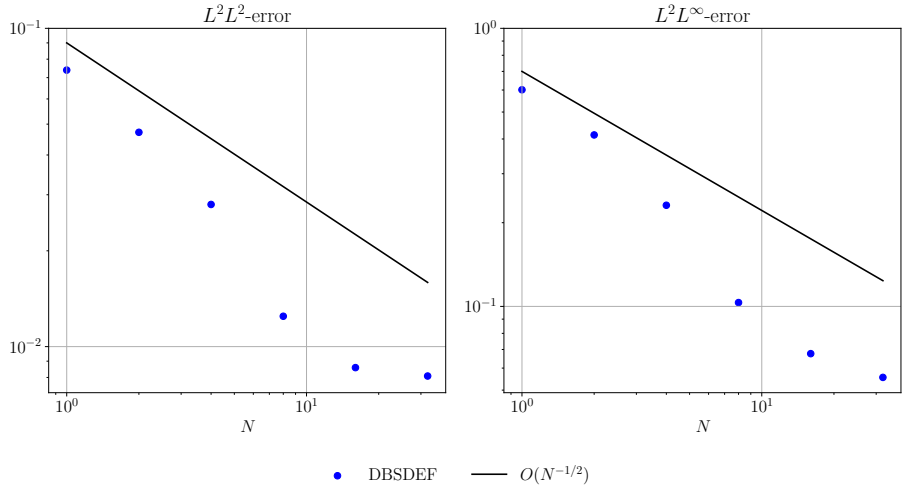

Figure 7.13: The L^2L^2- and L^2L^∞-error at $t = T = 1$ for the DBSDEF in the Ornstein–Uhlenbeck example.

Figure 7.14 shows the L^2L^2- and L^2L^∞-error of the DBSDEF with $N = 32$ and 4 bootstrap particle filters with $10^2, 10^3, 10^4, 10^5$ particles respectively at the time of each observation. The DBSDEF outperforms the particle filters with $10^2, 10^3, 10^4$ particles at all time steps for both metrics. However, as expected the particle filters converges as the number of particles increases, so the particle filter with 10^5 particles is on parity with the DBSDEF. It could be hypothesised that 10^6 particles would outperform the DBSDEF. Nevertheless, several options remain for improving the DBSDEF shown in this example, for instance increasing N, changing the architecture of the neural networks, larger training data set, and using a finer normalization grid.

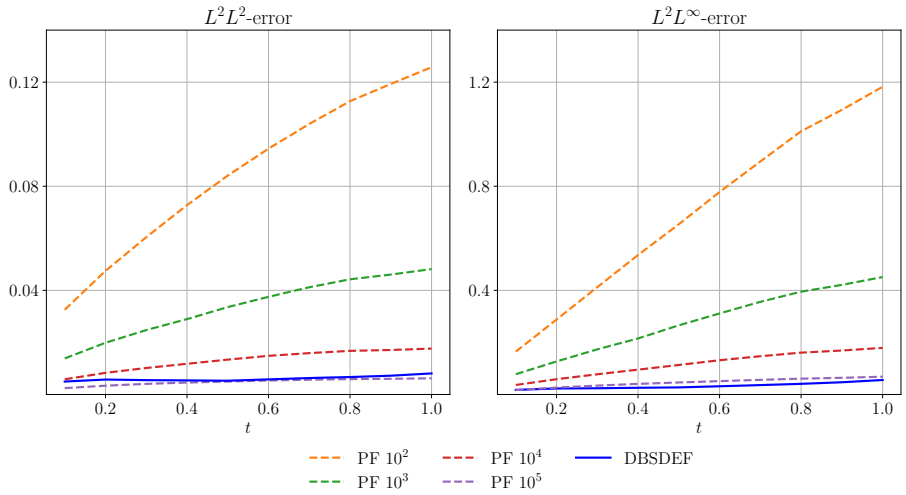

Figure 7.14: L^2L^2- and L^2L^∞-error for the Deep BSDE filter with $N = 32$ and particle filters for the Ornstein–Uhlenbeck example.

Finally, to build intuition on the Deep BSDE filter, one individual solution is shown in Figure 7.15 at five times $t = 0.2, 0.4, 0.6, 0.8, 1.0$, that is, at the time for every other

observation. One solution of the next example, the bimodal case is also included.

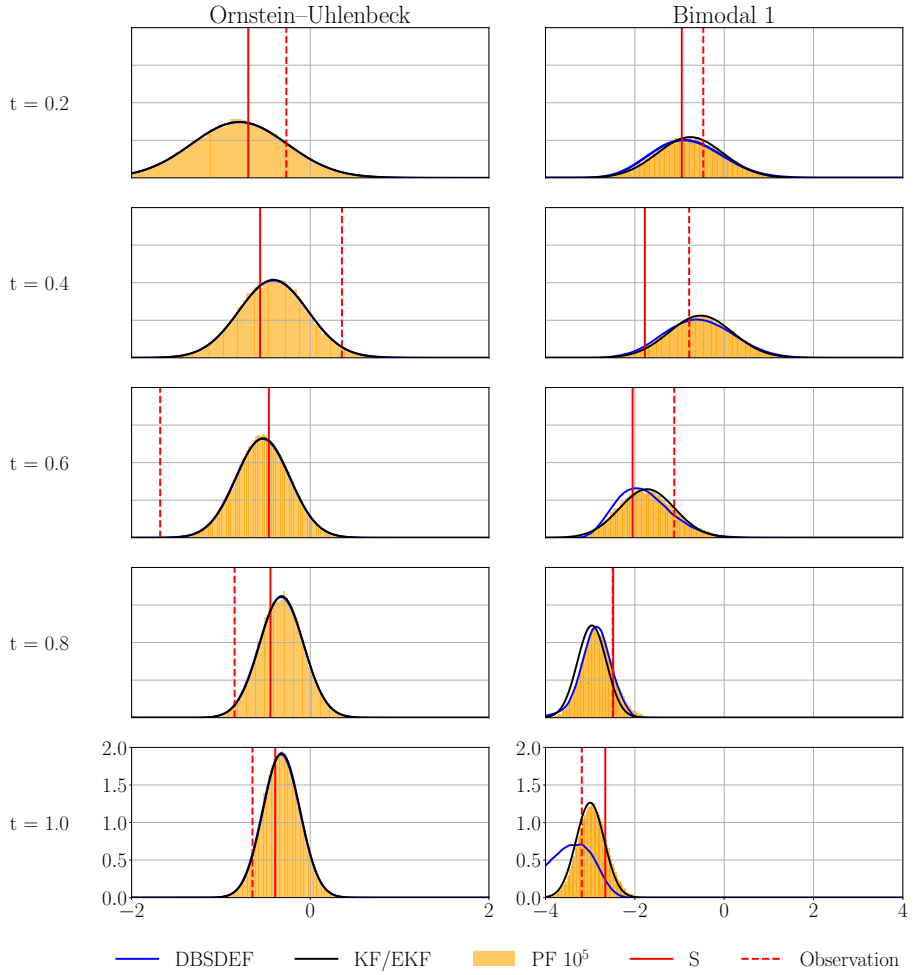

Figure 7.15: Examples of DBSDEF solutions compared with the Kalman filter and a particle filter at times $t = 0.2, 0.4, 0.6, 0.8, 1.0$. The true state S and its corresponding observations are included as vertical lines.

7.3 Bimodal Filtering

To explore how the DBSDEF performs under nonlinear dynamics we test the DBSDEF on a bimodal SDE, which is an SDE given by

$$X_t = X_0 + \int_0^t -4aX_s(X_s^2 - b^2)\,\mathrm{d}s + \int_0^t \sigma\,\mathrm{d}W_s. \tag{7.2}$$

The SDE is called bimodal since its long-term distribution has two modes. The parameter b controls the location of the two modes centered at $-b$ and b, also called wells, since in physics these correspond to points of lowest potential energy. The parameter a controls the depth and strength of the wells which means that larger a makes the force pulling the process towards the wells at $\pm b$ stronger. The diffusion term is constantly σ.

Because of the nonlinear dynamics of the Bimodal SDE, the reference filter is in this example a particle filter with 10^5 particles. Still having $d = 1$, we let $a = 0.1$, $b = 3$, and $\sigma = 1$. We use the same measurement dynamics, and distribution p_0 as in the OU filtering example from Section 7.2. The method is tested on a discretization of $N = 1, 2, 4, 8, 16, 32$ with $T = 1$ and 10 observations. That is, the same discretization and observation count as in Section 7.2. In Figure 7.16 the L^2L^2- and L^2L^∞-error over time for the different N are presented. Up to time $t = 0.5$ the DBSDEF converges quite well, after which the increase of the errors accelerates and the convergence stops. The most likely explanation for this result is that the bimodal filtering problem is harder, and that the amount of training was insufficient.

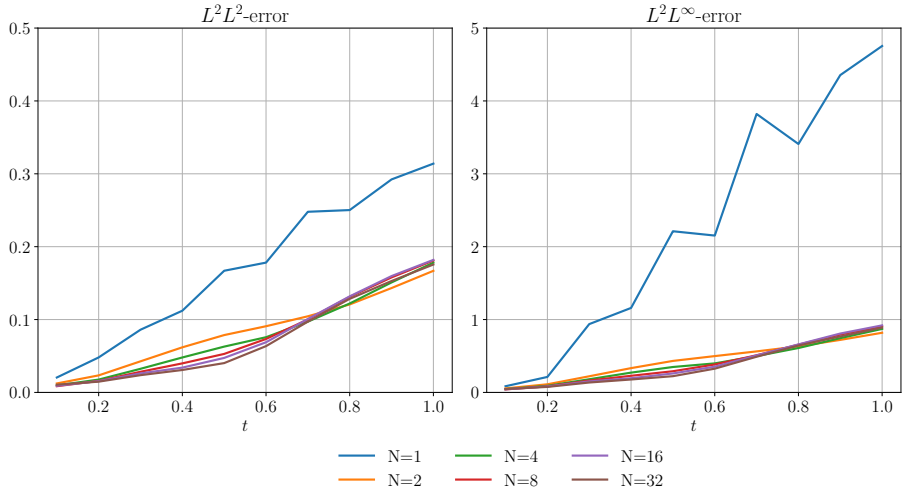

Figure 7.16: The L^2L^2- and L^2L^∞-error for the Deep BSDE filter in the first bimodal example

However, looking at the errors at $t = 0.5$, there is convergence, which is shown below in Figure 7.17. Still, it decays with a smaller order than in the Ornstein–Uhlenbeck example from the previous section.

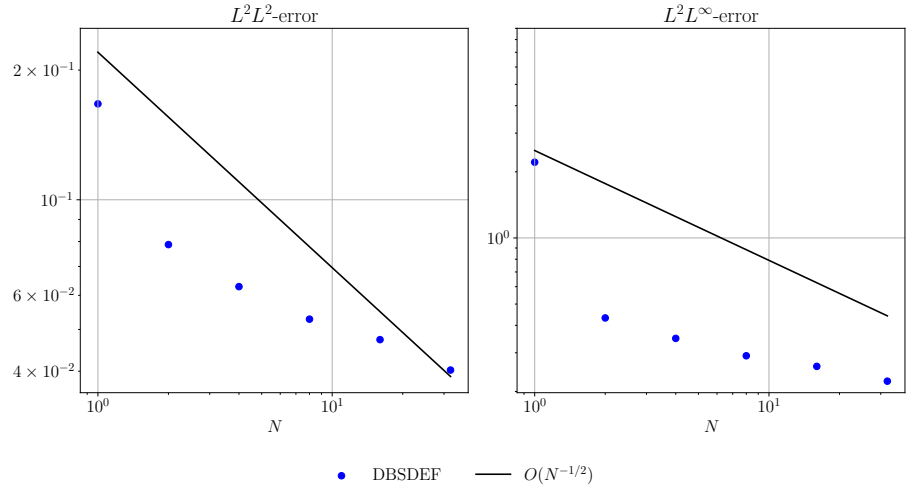

Figure 7.17: The L^2L^2- and L^2L^∞-error at time $t = 0.5$ for the Deep BSDE filter in the first bimodal example.

Since the reference filter is a particle filter with 10^5 particles, we can now compare the best DBSDEF with $N = 32$ with the Extended Kalman filter, as well as particle filters with $10^2, 10^3$ and 10^4 particles. This comparison is shown in Figure 7.18, where it can be seen that the DBSDEF is outperformed by the EKF after $t = 0.4$.

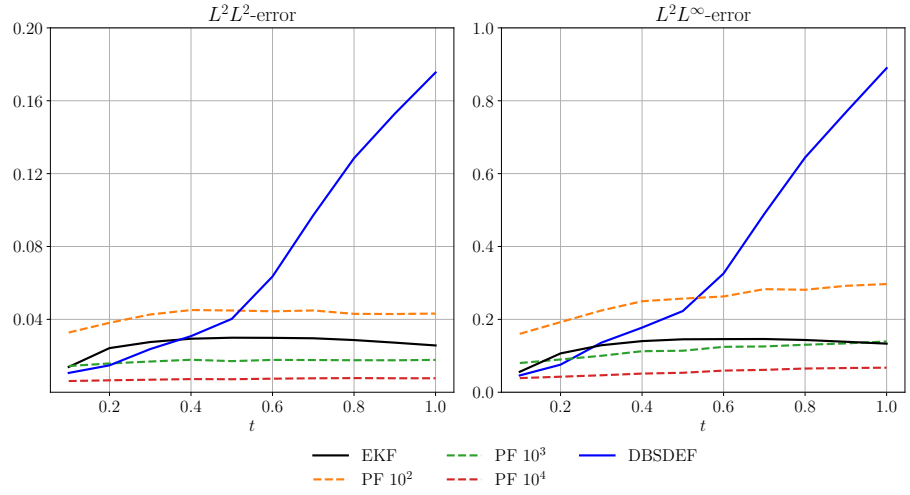

Figure 7.18: Comparison of the errors of the DBSDE, Extended Kalman filter and particle filters, for the first bimodal example.

The parameter b controls the centres of the modes, which indicates something about how fast the intersection of the probability distribution of the bimodal SDE and p_0 and the obtained approximations $\widehat{p}_k$ shrinks. Therefore, moving the modes further away from areas where p_0 and $\widehat{p}_k$ have significant probability mass likely makes the problem more difficult while moving the modes closer to areas where p_0 and $\widehat{p}_k$ has most of its probability mass likely makes the problem easier. In this next example using a bimodal process, the centres of the modes are therefore moved closer to 0, which is the centre of the Gaussian p_0. This is done by decreasing b from 3 to 2, i.e,

all other parameters being unchanged another DBSDEF was trained. The results from filtering on this second bimodal SDE can be seen in Figure 7.19. From the figure it can be seen that there is convergence for longer.

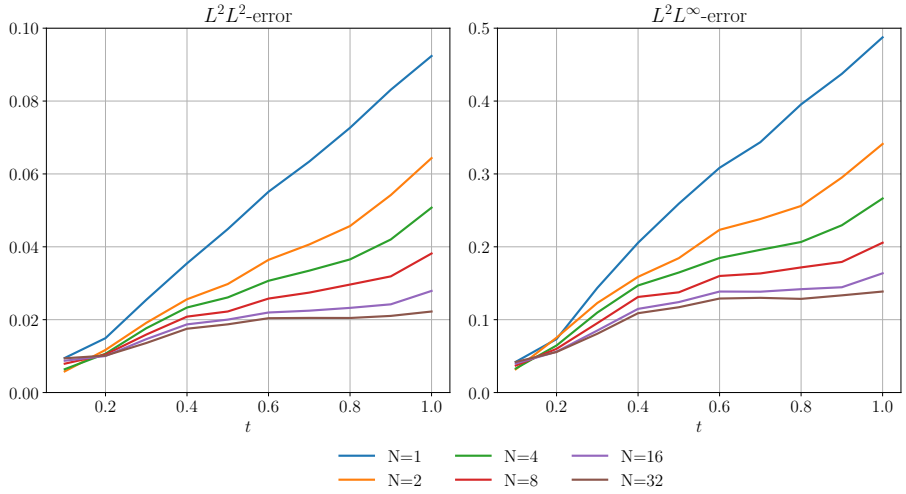

Figure 7.19: The L^2L^2- and L^2L^∞-error for the Deep BSDE filter in the second bimodal example.

Figure 7.20 shows the L^2L^2- and the L^2L^∞-errors at time $t = T = 1$ together with a reference line $O(N^{-1/2})$. In the figure it can be seen that the convergence extends to the final time $t = T = 1$, and comparing with the reference line, it can be noted that it could be of order $1/2$. Therefore it can be concluded that at least part of the reason for the poor performance in the first bimodal filtering case was due to difficulty of the problem, and that easier problem or more training likely yield better accuracy and convergence.

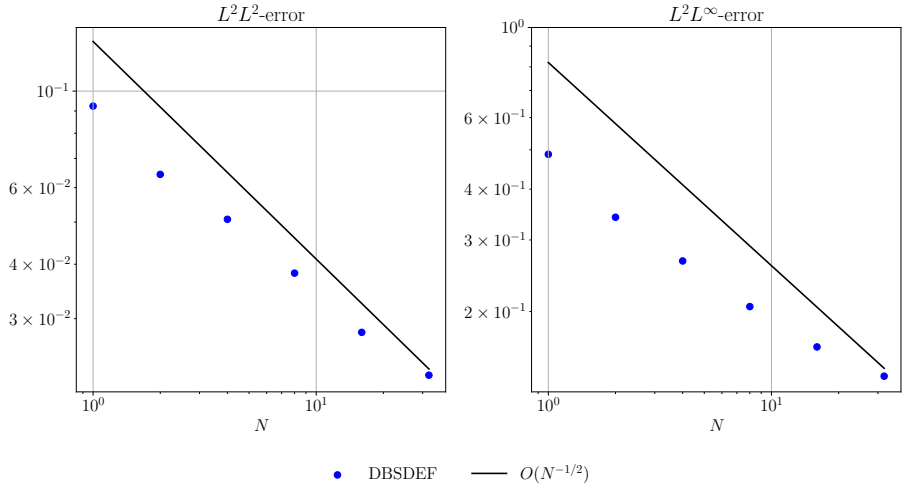

Figure 7.20: The L^2L^2- and L^2L^∞-error at $t = T = 1$ for the Deep BSDE filter in the second bimodal example.

The comparison with the EKF and particle filters is done once again and can be seen in Figure 7.21. The DBSDEF has lower L^2L^2-error than the EKF and similar

L^2L^∞-error. The particle filter with 10^4 outperforms the DBSDEF clearly on both metrics.

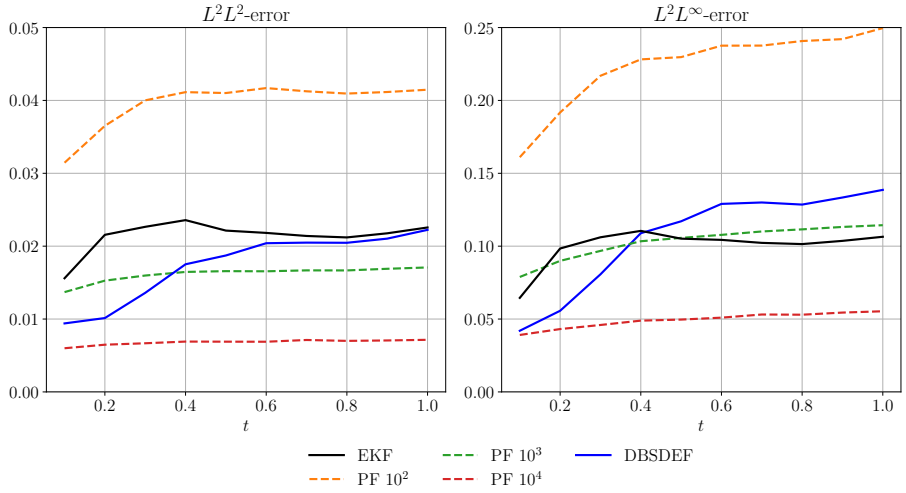

Figure 7.21: Comparison of the errors of the DBSDEF, Extended Kalman filter and particle filters in the second bimodal example.

7.4 Spring-Mass System Filtering

After two one-dimensional examples, we now consider a higher dimensional SDE, namely a spring-mass system with turbulence, similar to the one from Section 5.4.4 in [5]. It is an SDE given by

$$X_t = X_0 + \int_0^t \begin{bmatrix} 0_{M\times M} & I_{M\times M} \\ A_{21} & A_{22} \end{bmatrix} X_s \, \mathrm{d}s + \int_0^t \begin{bmatrix} \sigma_1 I_{M\times M} & 0_{M\times M} \\ 0_{M\times M} & \sigma_2 I_{M\times M} \end{bmatrix} \mathrm{d}W_s, \tag{7.3}$$

where

$$A_{21} = \begin{pmatrix} -\frac{k_1+k_2}{m_1} & \frac{k_2}{m_1} & & & \\ \frac{k_2}{m_2} & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \frac{k_M}{m_{M-1}} \\ & & & \frac{k_M}{m_M} & -\frac{k_M+k_{M+1}}{m_M} \end{pmatrix}, \quad A_{22} = \begin{pmatrix} -\frac{c_1+c_2}{m_1} & & \\ & \ddots & \\ & & -\frac{c_M+c_{M+1}}{m_M} \end{pmatrix}.$$

It is a system of springs, masses and dampers that are interacting dynamically. More specifically, m_i is the i-th mass, k_i is the spring constant of the i-th spring and c_i is the damping coefficient of the i-th damper. The state vector X will be $\mathbb{R}^{2M}$-valued where the first M components represents positions of the masses and the last M components represent velocities of the objects.

In our case, we use $M = 2$ which makes the system consist of 2 masses, 3 springs and 3 dampers. The state vector X will be $\mathbb{R}^4$-valued, of which we observe the first position and the first velocity. Hence $d' = 2$ with measurements given by

$$h(x) = Hx, \tag{7.4}$$

where H is a 2×4 matrix that is zero everywhere except at positions $(1,1),(1,3)$, where it is 1. The system is linear so the optimal filter density is given by the Kalman filter, which is therefore the reference filter. During training we use importance sampling with 5000 samples from $\mathcal{N}(\mu_k^m, 2\Sigma_k^m)$ to estimate the normalization constant, where μ_k^m, Σ_k^m are given by the Kalman filter. Because the 4-dimensional problem is significantly more computationally demanding, the number of observations is reduced from 10 to 5, and the time interval is shortened to $t \in [0, 0.5]$. This way, the observations will come during the same time-steps as previously with the end time at $T = 0.5$ instead. The shorter time interval and fewer observations are chosen to maintain quality of the results and are preferred rather than increasing the size of the training data set.

Figure 7.22 shows the L^2L^2- and the L^2L^∞-errors for the different N. Looking at the figure we can see convergence for the DBSDEF as N increases. However, it is also apparent that the DBSDEF is harder to train in higher dimension, which can be seen by seemingly slower convergence, and the errors increasing consistently. It could be that a finer partition, *i.e.* larger N would yield a flatter error curve, like was seen in the previous one-dimensional Ornstein–Uhlenbeck and bimodal examples.

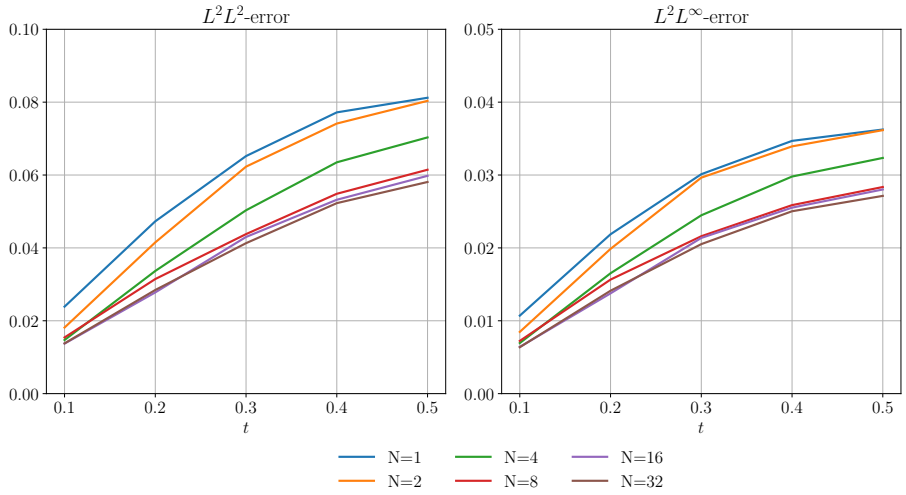

Figure 7.22: The L^2L^2- and L^2L^∞-error for the Deep BSDE filter in the spring-mass system example.

Figure 7.23 shows the L^2L^2- and the L^2L^∞- error for the DBSDEF for the different N at the end time $t = T = 0.5$. It is apparent that the rate of convergence is slower than in the one dimensional examples, but it is noteworthy that there are many parameters to tune for improvement. For instance, better training normalization, more training and other neural architectures. However, there is still improvement as N increases. Another option is that faster convergence starts at a larger N than was tested for in this example.

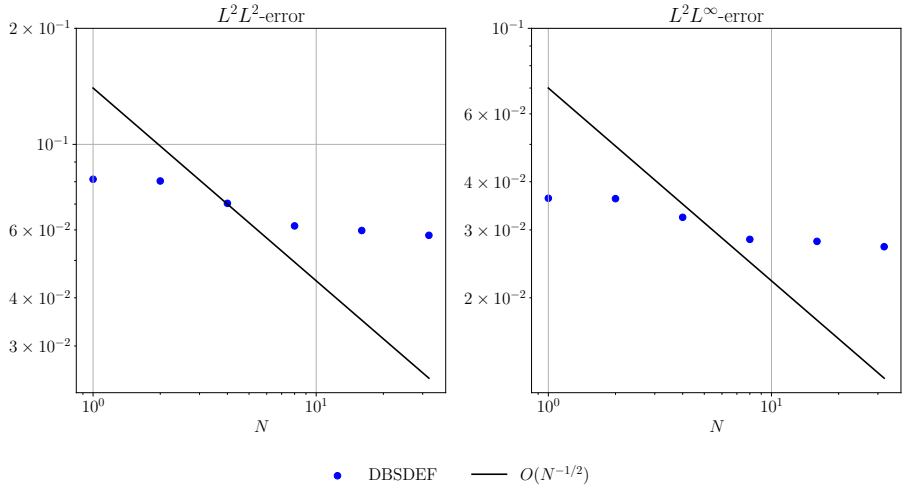

Figure 7.23: The L^2L^2- and L^2L^∞-error at $T = 0.5$ for the Deep BSDE filter in the spring-mass system example.

A comparison between the DBSDEF and bootstrap particle filters with $10^2, 10^3, 10^4, 10^5$ particles is made and their L^2L^2-errors and L^2L^∞-errors are shown in Figure 7.24. There is convergence for the particle filters, and a particle filter with 10^5 particles outperforms the DBSDEF. Like mentioned several times, there are many ways to also improve this iteration of the DBSDEF, which could have even more effect in high dimension since the potential for improvement could be higher.

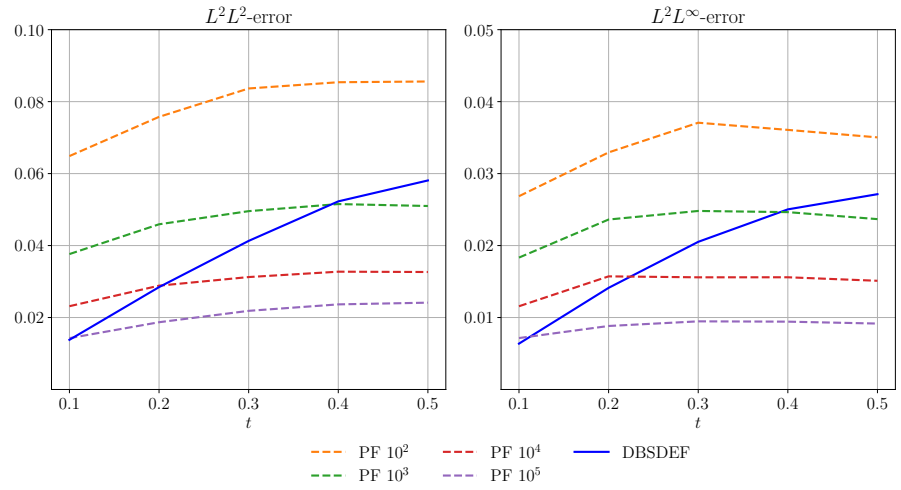

Figure 7.24: Comparison of the DBSDEF and PFs in the spring-mass system.

As a final demonstration, a single solution for the different filters is shown in Figure 7.25. This is done by computing the marginal densities of each dimension. The Kalman filter and the particle filer are aligned while the DBSDEF deviates from them. It can be noted that the DBSDEF is performing worse in the two unobserved dimensions, Position 2 and Velocity 2 than in the observed dimensions. This is expected since the filter is provided with less information in these dimensions. The issue could potentially be mitigated with more training, since the DBSDEF is

tending to the right marginal distributions.

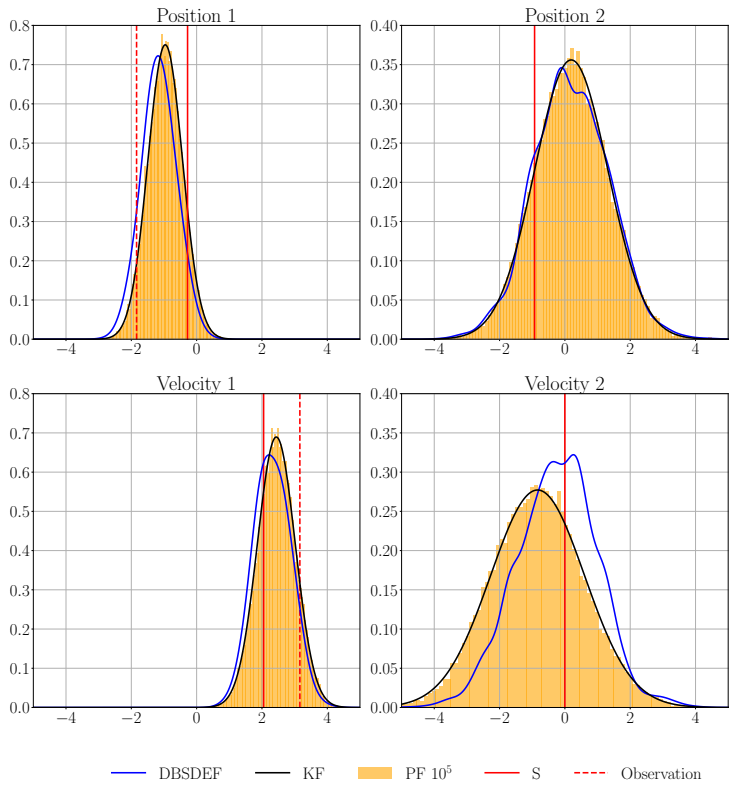

Figure 7.25: Marginal density of the DBSDEF, PF and the KF and the true state S in all dimensions and observations in the observed dimensions 1 and 3.

8
Conclusion

The purpose of the thesis was to analyse the empirical convergence rate of the novel Deep Backward Stochastic Differential Equation Filter. It was deemed important due to the need for effective filters in high dimensional nonlinear settings, where current methods either have weak performance or are computationally expensive.

The DBSDEF uses the Deep BSDE method in sequence, and hence it became a sub-objective to analyse the validity of this approach, which we referred to as the sub-problem approach. Empirical convergence was analysed in depth, shown and discussed in Section 7.1, however only for a one-dimensional Ornstein–Uhlenbeck process. The approach might be verified further by running it in higher dimension and for other equations. However, with this result and existing theory on the convergence of the Deep BSDE method [17], the sub-problem approach was regarded as sufficiently validated to continue the analysis with the DBSDEF. Preliminary results suggest that the sub-problem approach could achieve convergence with fewer computational resources than the original Deep BSDE method, but this finding requires more detailed investigation.

The main conclusion of the thesis is that the DBSDEF shows promise as an alternative for filtering in nonlinear settings, and potentially in increasing state-dimension. The results achieved in the thesis indicates that the empirical convergence rate aligns with the theoretical convergence rate of $O(N^{-1/2})$. Consequently, the DBSDEF could serve as an alternative and a complementary approach to the recently introduced EBDS filter which was analysed in [5, 31]. However, there are still numerous elements to analyse and improve upon. In high dimension, further research needs to be conducted to solidify the preliminary evidence that the empirical convergence rate is aligned with the theoretical convergence rate. One way to accomplish this is to use more samples and finer normalization to have smaller approximation and statistical errors. Additionally, finer normalization in the evaluation procedure could help reduce the statistical error that is introduced while evaluating the DBSDEF.

Finally, there are several other aspects and methods to explore and fine-tune to improve the DBSDEF. As noted in Chapter 6, it could be worthwhile to replace the fully connected neural networks with convolutional neural networks in the DBSDEF. This will leverage temporal relations between observations and make the method significantly cheaper to train.

Bibliography

[1] Y. Aït-Sahalia and L. P. Hansen. *Econometric Modeling and Inference*. Cambridge University Press, 2009.

[2] François Auger, Mickaël Hilairet, Josep M. Guerrero, Eric Monmasson, Teresa Orlowska-Kowalska, and Seiichiro Katsura. Industrial applications of the kalman filter: A review. *IEEE Transactions on Industrial Electronics*, 60(12):5458–5471, 2013.

[3] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, May 1993.

[4] Michael C. Burkhart, David M. Brandman, Brian Franco, Leigh R. Hochberg, and Matthew T. Harrison. The discriminative kalman filter for bayesian filtering with nonlinear and nongaussian observation models. *Neural Computation*, 32(5):969–1017, 2020.

[5] Kasper Bågmark, Adam Andersson, and Stig Larsson. An energy-based deep splitting method for the nonlinear filtering problem. *Partial Differential Equations and Applications*, 4(2):1–27, 2023.

[6] Kasper Bågmark, Adam Andersson, Stig Larsson, and Filip Rydin. A convergent scheme for the bayesian filtering problem based on the fokker–planck equation and deep splitting. *arXiv preprint arXiv:2409.14585*, 2024. Version 2, last revised 17 Jan 2025.

[7] S. Challa and Y. Bar-Shalom. Nonlinear filter design using fokker-planck-kolmogorov probability density evolutions. *IEEE Transactions on Aerospace and Electronic Systems*, 36(1):309–315, 2000.

[8] Michael G. Crandall, Hitoshi Ishii, and Pierre-Louis Lions. User's guide to viscosity solutions of second order partial differential equations. *Bulletin of the American Mathematical Society*, 27(1):1–67, 1992.

[9] R. Daley. *Atmospheric Data Analysis*. Cambridge University Press, 1993.

[10] Bruno Demissie, Muhammad Altamash Khan, and Felix Govaers. Nonlinear filter design using Fokker–Planck propagator in Kronecker tensor format. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1–8, 2016.

[11] Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, New York, 2001.

[12] Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

[13] Robert J. Elliott. Stochastic integrals for martingales of a jump process with partially accessible jump times. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 36(3):213–226, 1976.

[14] Avner Friedman. Stochastic differential equations and applications. In J. Cecconi, editor, *Stochastic Differential Equations*, volume 77 of *C.I.M.E. Summer Schools*, page N/A. Springer, Berlin, Heidelberg, 2010.

[15] Emmanuel Gobet. *Monte-Carlo Methods and Stochastic Processes: From Linear to Non-Linear*. Chapman and Hall/CRC, Boca Raton, FL, 2016.

[16] M. S. Grewal, L. R. Weill, and A. P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration*. Wiley, 1st edition, 2001.

[17] Jiequn Han and Jihao Long. Convergence of the Deep BSDE Method for Coupled FBSDEs. *Probability, Uncertainty and Quantitative Risk*, 5(1):1–33, 2020.

[18] Kiyosi Itô. On a formula concerning stochastic differentials. *Nagoya Mathematical Journal*, 3:55–65, 1951.

[19] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(1):35–45, 1960.

[20] N. El Karoui, S. Peng, and M. C. Quenez. Backward stochastic differential equations in finance. *Mathematical Finance*, 7(1):1–71, 1997.

[21] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. `https://arxiv.org/abs/1412.6980`.

[22] Fima C. Klebaner. *Introduction to Stochastic Calculus with Applications*. Chalmers Library Print Collection, 2012.

[23] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*, volume 23 of *Stochastic Modelling and Applied Probability*. Springer, 1992.

[24] Steffen L. Lauritzen. Time series analysis in 1880: A discussion of contributions made by t.n. thiele. *International Statistical Review / Revue Internationale de Statistique*, 49(3):319–331, Dec 1981.

[25] Michael A. Nielsen. *Neural Networks and Deep Learning*, chapter 4, pages 81–104. Determination Press, 2015.

[26] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, 2nd edition, 1999.

[27] Hans C. Öttinger. *Stochastic Processes in Polymeric Fluids: Tools and Examples for Developing Simulation Algorithms*. Springer, Berlin, Heidelberg, 1996.

[28] R. M. Rangayyan. *Biomedical Signal Analysis: A Case-Study Approach*. Wiley-IEEE Press, 2002.

[29] Pablo Rebeschini and Ramon van Handel. Can local particle filters beat the curse of dimensionality? *The Annals of Applied Probability*, 25(5):2809–2866, 2015.

[30] Peter A. Ruymgaart and Tsu T. Soong. *Mathematics of Kalman-Bucy Filtering*. Springer-Verlag, 1985.

[31] Filip Rydin. A deep learning method for nonlinear stochastic filtering: Energy-based deep splitting for fast and accurate estimation of filtering densities. Master's thesis, Department of Mathematical Sciences, Chalmers University of Technology, Gothenburg, Sweden, 2024.

[32] Simo Särkkä and Lennart Svensson. *Bayesian Filtering and Smoothing*, volume 17 of *Institute of Mathematical Statistics Textbooks*. Cambridge University Press, 2nd edition, 2023.

[33] Robert H. Shumway and David S. Stoffer. State-space models. In *Time Series Analysis and Its Applications*, Springer Texts in Statistics. Springer, Cham, 2025.

[34] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30. Curran Associates, Inc., 2017.

[36] Yajie Yu, Narayan Ganesan, and Bernhard Hientzsch. Backward Deep BSDE Methods and Applications to Nonlinear Problems. *Risks*, 11(3):61, 2023.

[37] Étienne Pardoux and Shige Peng. Backward stochastic differential equations and quasilinear parabolic partial differential equations. *Stochastic Partial Differential Equations and Their Applications*, 176:200–217, 1992.

A

Appendix 1

For the sub-problem approach, we show how it behaves with fewer training samples for each epoch, that is $M = 10^4$ instead of $M = 10^7$. The results from this computationally less demanding experiment are shown in Figures A.1 and A.2. The sub-problem approach outperforms the deep BSDE method at all N. As expected with fewer training samples, the error development over time is clearly more unstable.

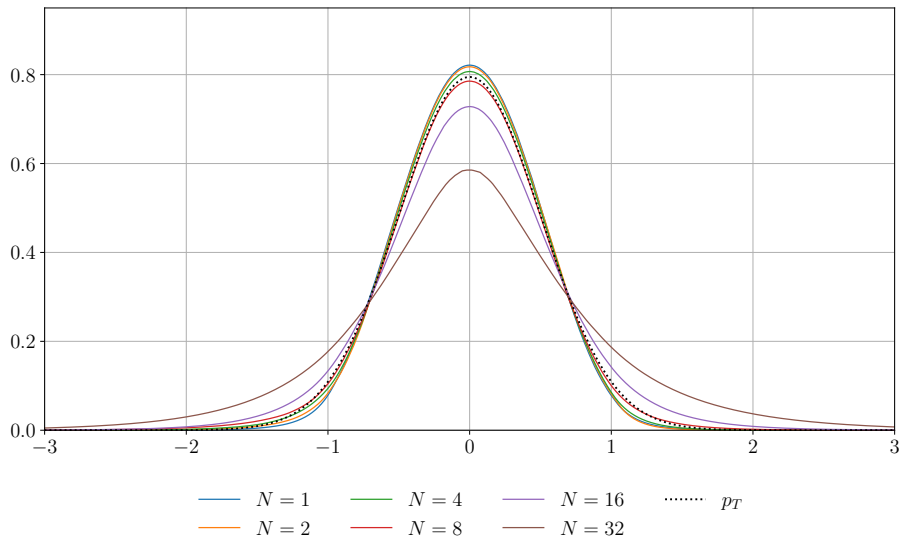

Figure A.1: Normalized results from the sub-problem approach at $KN = 32$ for $N = \{1, 2, 4, 8, 16\}$ compared to the Deep BSDE method at time $t = 1$, for the no evolution case, with $M = 10^4$ samples.

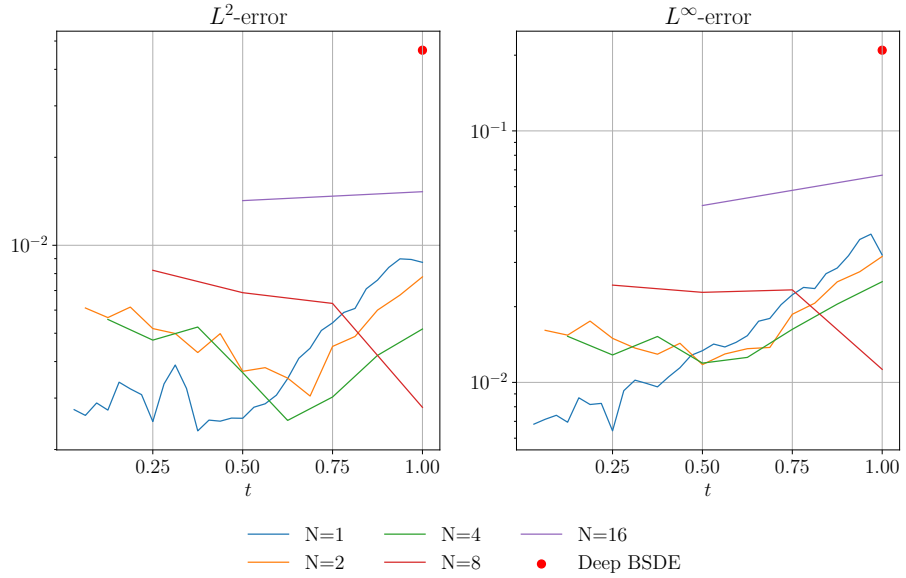

Figure A.2: Comparison of L^2-error and L^∞-error no evolution case with 10^4 samples.