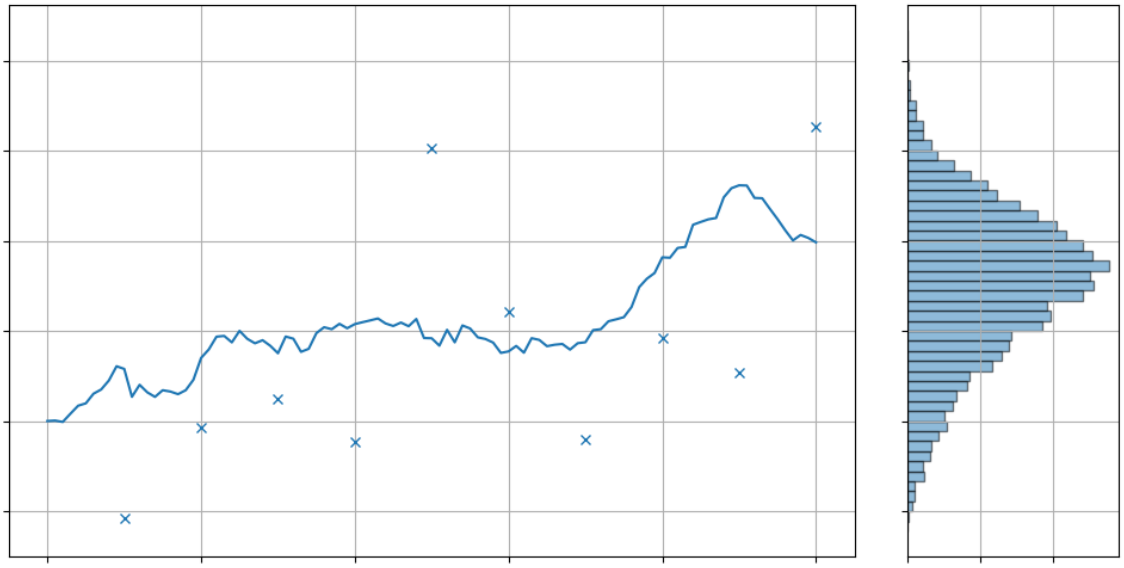


A WGAN Based Method for Stochastic Filtering

Solving Conditional Distributions by Means of Combining Neural Networks in a WGAN Setting

Master's thesis in Physics

Joel Axelsson

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

A WGAN Based Method for Stochastic Filtering

Solving Conditional Distributions by Means of Combining Neural Networks in a WGAN Setting

Joel Axelsson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
Gothenburg, Sweden 2026

A WGAN Based Method for Stochastic Filtering
Solving Conditional Distributions by Means of Combining Neural Networks in a WGAN Setting
Joel Axelsson

Supervisor: Patrik Albin, Department of Mathematical Sciences
Examiner: Adam Andersson, Department of Mathematical Sciences/SAAB AB

Master's Thesis 2026
Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

A WGAN Based Method of Stochastic Filtering
Solving Conditional Distributions By Means of Combining Neural Networks in a WGAN Setting
Joel Axelsson
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

The problem of extracting information about a state from incomplete noisy measurement is knowns as a ”filtration problem” in the field of stochastic processes. In this thesis the information extracted corresponds to an estimate of the posterior conditional distribution of a stochastic process. Recent development in generative adversarial networks allows for such filtering problems to be solved using a network class called the WGAN. In this thesis a method of implementing the WGAN for filtration is investigated. The method is tested for a linear SDE scaled in dimensions and on a non-linear SDE of singular dimension. both examples were observed linearly with additive Gaussian noise.

The method was benchmarked against filtering methods not based on machine learning. In summary it can be stated the the results indicated that some merit to the method could be deducted. It was however the result that the method was outperformed by most of its peers. An investigation into where the method could be improved was conducted.

Acknowledgements

I would like to express my sincere gratitude towards my examiner Adam Andersson and my supervisor Patrik Albin. Thank you Adam for providing me with this thesis and for your continuous support throughout the project. Thank you Patrik for your insights about the thesis which have been of great help. Lastly I would like to extend my gratitude to my friends and family. All of which have been supportive all throughout my education. Without you all it would not have been possible.

Joel Axelsson, Gothenburg, January 2026

Contents

1

Introduction

Classical physics postulates that the laws governing life around us are determinate. This means that any given action should in theory result in the same reaction. This is however often not the case. In reality most processes which in theory are determinate are in fact not. A good physicist might try to calculate the trajectory of a tennis ball after it bounces but will be surprised to learn that sometimes unexpected outcomes occur. The physicist will argue that there are no randomness involved, that the perfect prediction would be possible every time if just all information was available. This might be true, it is however seldom the case that perfect knowledge of a system is available. In many engineering practices these unknown dynamics are explained as noise. The problem to solve becomes how we negate this noise in order to obtain the true results. In applied mathematics, the problem of determining information about a true state given incomplete observation is called ”filtering”. With the rise of computer assisted control systems, interest in computational filtering rose. In the 1960's, Rudolf E. Kálmán lead the development of filtering methods, which now share his name[11]. After Kálmán proposed his filter many more advanced methods in filtering have been successfully developed to more reliably solve more complex problems. To this day the development of new filtering methods continues.

In 2014 a new approach to generative neural networks was proposed by Goodfellow et al. in [6]. This new proposed method proved to be a huge improvement in the area of generative networks. Generative networks are networks which are capable of reproducing distributions and the newly approach gave neural networks the ability to be used to solve filtering problems. In 2017 Gulrajani et al. proposed an improved network, called the WGAN [7]. This thesis aims to explore the idea of implementing a WGAN network to try to solve filtering problems.

The structure of the thesis is as follows. In Chapter 2 an introduction of the underlying theory useful for understanding this thesis is presented. In Chapter 3 an explanation to machine learning and WGANs is provided. In Chapter 4 the method for implementing the WGAN used in this thesis is presented. Chapter 5 other Bayesian filtering methods used for comparison are explained. Chapter 6 provides the SDEs generating the stochastic processes used to evaluate the WGAN. Chapter 7 provides the metrics of evaluation along with the results and error analysis of the method.

2

Theory

This chapter gives an introduction to the theory necessary for this thesis. First some notation is explained. Thereafter concepts from stochastic analysis and probability are introduced.

2.1 Notation

By $\mathbb{R}^d$ denote the d-dimensional Euclidean space. Let $\langle \cdot, \cdot \rangle$ denote the standard inner product

$$\langle a, b \rangle = \sum_{i=1}^{d} a_i b_i,$$

for two vectors $a, b \in \mathbb{R}^d$ and $\|\cdot\|$ denote the corresponding norm

$$\|a\| = \sqrt{\langle a, a \rangle}.$$

Denote the set of real numbers on a d-dimensional unit sphere by $\mathbb{S}^{d-1} = \{x \in \mathbb{R}^d : \|x\| = 1\}$.

2.2 Probability theory and filtration

In the field of stochastic processes, filtration is the area which deals with extracting information from incomplete data, often subject to noise. Many different ways of solving filtration problems exist. The basis of all such solutions lies in the existence of optimal solutions to many such problems. This section aims in building up theory which can be used in proving the existence of solutions to filtration problems.

2.2.1 Probability spaces and measures

A sample space, from the field of probability, is the set of all possible outcomes of a random trial. For a random trial to be valid, the sample space cannot be empty. A sample space can be either continuous or discrete. Building upon the sample space, following is the definition of a σ-algebra.

Definition 2.2.1 (Definition of a σ-algebra)**.** Let Ω be a non-empty set (for example a sample space). A collection of subsets of Ω is called a σ-algebra, denoted $\mathcal{F}$, if

1. $\Omega, \emptyset \in \mathcal{F}$
2. $A \in \mathcal{F} \implies A^c \in \mathcal{F}$
3. If $A_1, A_2, A_3 ... \in \mathcal{F} \implies \bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$

From solely this definition of a σ-algebra, not much can be inferred about its usefulness in probability theory. It is however essential in defining probability spaces. The final definition before probability spaces can be introduced is that of a probability measure. Below is the definition of a probability measure.

Definition 2.2.2 (Definition of a probability measure)**.** If a real-valued set function P on an σ-algebra $\mathcal{F}$ is countably additive, that is it satisfies

$$P\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} P(A_n),$$

for all pairwise disjoint sets A_n in $\mathcal{F}$ such that $\bigcup_{n=1}^{\infty} A_n \in \mathcal{F}$, then it is called a measure. Such a measure defined on a σ-algebra of subsets of a space Ω is called a probability measure if $P \geq 0$ and $P(\Omega) = 1$.

Using the previous definitions the probability space is defined as a triple consisting of a sample space Ω, a σ-algebra and a probability measure, denoted $(\Omega, \mathcal{F}, P)$. Furthermore, a measurable space is defined as a tuple consisting of an measure space E and a σ-algebra $\mathcal{E}$ denoted $(E, \mathcal{E})$.

Both the probability space and the measurable space are the corner stones in defining random variables. As stated earlier, the sample space should be seen as all the possible outcomes. The purpose of the σ-algebra in not self evident from its definition. Another name for the σ-algebra is the event space, which gives some indication of its purpose. The specifics of the event space is what dictates what information can be deducted after a random trial. Furthermore, σ-fields can be subsets of one another. In this case a smaller σ-algebra offers less detailed information than the bigger. It can also be the case that they are incomparable. Then information from the outcome in one σ-algebra leads to no information in the other σ-algebra. Finally the probability measure is the function which assigns probabilities to each event.

For a probability space $(\Omega, \mathcal{F}, P)$ and a measurable space $(E, \mathcal{E})$ a random variable is defined as a measurable function $X : \Omega \to E$ mapping the sample space to the measure space. This thesis deals only with real-valued random variables, meaning that the measure space is some Euclidean space. In this case the formal definition of measurability is:

Definition 2.2.3 (Measurability)**.** Let $(\mathbb{R}, \mathcal{E})$ be a measurable space, that is a space

with a σ-algebra. A real-valued function $f : \mathbb{R} \to \mathbb{R}$ is called measurable with respect to $\mathcal{E}$(or $\mathcal{E}$-measurable) if $\{x : f(x) < c\} \in \mathcal{E}$ for every $c \in \mathbb{R}$.

Remark. The previous definition of a real-valued measurable function is easily expanded from function in $\mathbb{R}$ onto functions $f : \mathbb{R}^n \to \mathbb{R}^n$. This function can be thought of as multiple $\mathbb{R}$ function as $f = (f_1, f_2, ..., f_n)$. Thus if all of these functions f_i are are measurable, then f is measurable.

Lastly, any random variable X is said to be L^p if

$$\mathbf{E}[\|X\|^p] < \infty.$$

Formally the L^p space is defined as

$$L^p(\Omega, \mathcal{F}, P) = \{X : \Omega \to \mathbb{R}^d \ ; \ \mathbf{E}[\|X\|^p] < \infty\}.$$

This section only provides a fundamental explanation of probability spaces and measures. For a more detailed explanation the reader is referred to [13].

2.2.2 Conditional expectation of random variable

A stochastic process is a collection of random variables on a common probability space $(\Omega, \mathcal{F}, P)$. In this notation a random variable is a measurable function $X : \Omega \to \mathbb{R}$ mapping the probability space to the measurable space. Considering continuous random variables, and assuming $X \in L^1(\Omega, \mathcal{F}, P)$. Let $\mathcal{G} \subseteq \mathcal{F}$ be a sub-σ-algebra, then a conditional expectation of X given $\mathcal{G}$ is any $\mathcal{G}$-measurable stochastic variable $Z : \Omega \to E$ satisfying

$$\int_G Z dP = \int_G X dP \quad \text{for all } G \in \mathcal{G}.$$

We denote such Z by $\mathbf{E}[X|\mathcal{G}]$. For a conditional expectation value with respect to a σ-algebra generated by a stochastic variable the notation $\mathbf{E}[X|Y] := \mathbf{E}[X|\sigma(Y)]$ is used.

In the following section the assumption is that of real-valued continuous random variables $X : \Omega \to \mathbb{R}^d$, which are also L^2. Supposing X has mean $\mu \in \mathbb{R}^d$ and variance $\sigma^2 \in \mathbb{R}^{d \times d}$ then by definition

$$\min_{x \in \mathbb{R}^d} \mathbf{E}\left[\|X - x\|^2\right]$$

is minimized by $x = \mu$ and the minima is given by σ^2. If the minimization is instead done over functions g and h as to minimize

$$\min_{h \text{ measurable}} \mathbf{E}[\|g(X) - h(Y)\|^2]$$

then it can be proven that the $\sigma(Y)$-measurable function h which minimizes the previous mentioned object is $h(Y) = \mathbf{E}[g(X)|Y]$. The proof of the existence and uniqueness of such a function is given in Theorem 2.2.1.

Lemma 1. *For two stochastic variables $X, Y \in L^2$ and a function g such that $g(X) \in L^2$, it holds that*

$$\mathbf{E}\left[(g(X) - \mathbf{E}[g(X)|Y]) \cdot h(Y)\right] = 0,$$

for all functions h measurable with respect to $\sigma(Y)$.

Proof. The proof follows directly from applying the towering identity of conditional expectation, doing so we can show that

$$\mathbf{E}[(g(X) - \mathbf{E}[g(X)|Y]) \cdot h(Y)] = \mathbf{E}[(g(X)h(Y)] - \mathbf{E}[\mathbf{E}[g(X)|Y]h(Y)].$$

From the tower identity of conditional expectation values we can show that

$$\mathbf{E}[\mathbf{E}[g(X)|Y]h(Y)] = \mathbf{E}[\mathbf{E}[g(X)h(Y)|Y]] = \mathbf{E}[g(X)h(Y)].$$

Thus we have shown that

$$\mathbf{E}[(g(X) - \mathbf{E}[g(X)|Y]) \cdot h(Y)] = 0.$$

□

Theorem 2.2.1. *Let X,Y be two L^2 variables and g,f are two functions such that $g(X)$,$f(Y)$ are also L^2 then the function f, measurable with respect to $\sigma(Y)$, that minimizes*

$$\min_{f \ measurable} \mathbf{E}[(g(X) - f(Y))^2]$$

is $f(Y) = \mathbf{E}[g(X)|Y]$.

Proof. Let $f^*(Y) = \mathbf{E}[g(X)|Y]$ and f be another $\sigma(Y)$ measurable function. Then we wish to show that

$$\mathbf{E}[(g(X) - f(Y))^2] \geq \mathbf{E}[(g(X) - f^*(Y))^2].$$

To show this we expand $(g(X) - f(Y))^2$ as

$$\begin{aligned}(g(X) - f(Y))^2 &= (g(X) - f^*(Y) + f^*(Y) - f(Y))^2 \\ &= (g(X) - f^*(Y))^2 + 2\,(g(X) - f^*(Y))\,(f^*(Y) - f(Y)) \\ &\quad + (f^*(Y) - f(Y))^2.\end{aligned}$$

Now taking the exception value of this, using Lemma 1 to show that the expection of the cross term is zero, gives us

$$\mathbf{E}[(g(X) - f(Y))^2] = \mathbf{E}[(g(X) - f^*(Y))^2] + \mathbf{E}[(f^*(X) - f(Y))^2].$$

In this equation the second exception value is non-negative. Thus we see that the left-hand side of the equation obtains its minima if $\mathbf{E}[(f^*(X) - f(Y))^2]$ is zero, which is obtained for $f(Y) = f^*(Y) = \mathbf{E}[g(X)|Y]$. □

The existence of a minimizing function $h(Y) = \mathbf{E}[X|Y]$ is of great consequence in the field of particle filters. Theorem 2.2.1 states that of any possible function the conditional expectation $\mathbf{E}[X|Y]$ is the best mean squared error predictor of X given the observation Y. This can be interpreted as all the information available from the measurement are being extracted, leaving the residual $X - \mathbf{E}[X|Y]$ uncorrelated with any function of Y. A more rigorous discussion of the concepts of this section is provided in chapter 13 in [17].

2.3 Stochastic differential equations

A Wiener process is a real-valued stochastic process continuous in time. Formally the Wiener process is said to be any process satisfying the characteristics presented in Definition (2.3.1).

Definition 2.3.1 (Definition of Wiener process)**.** A Wiener process is a process W_t which satisfies

1. $W_0 = 0$ is almost surely.

2. W_t has independent increments: the future increments $W_{t+u} - W_t$, $t, u \geq 0$ are independent of past values W_s, $t > s \geq 0$.

3. W_t has Gaussian increments: $W_{t+u} - W_t \sim \mathcal{N}(0, u)$.

4. W_t has almost surely continuous paths.

The definition above is that of a one-dimensional Wiener process. The definition can be extended to a multidimensional Wiener process $W = (W_t^1, W_t^2, \ldots, W_t^d)$, where all the constituents satifsy the definition of a Wiener process. Building upon the definition of the Wiener process, a stochastic differential equation (SDE) is an equation which involve at least one stochastic element. In differential terms the general form of an SDE can be written as

$$\mathrm{d}X_t = \mu(t, X_t)\mathrm{d}t + \sigma(t, X_t)\mathrm{d}W_s, \quad t \in (0, T], \quad X_0 = x_0, \tag{2.1}$$

where $\mu : [0, T] \times \mathbb{R}^n \to \mathbb{R}^n$ is the drift coefficient and $\sigma : [0, T] \times \mathbb{R}^n \to \mathbb{R}^{n \times m}$ is the diffusion coefficient. For the sake of clarity, the aforementioned dimensions of μ and σ results in $X \in \mathbb{R}^n$ and $W \in \mathbb{R}^m$. Worth noting is that this definition is not strictly rigorous due to it not being Stieltjes integrable since W_t is not of finite variation. Nonetheless it is commonly used to refer to the more rigorous stochastic integral equation, which will be introduced shortly. The interpretation of Equation (2.1) should be that the dynamics of a variable X is governed by both deterministic and stochastic processes. The solution to the equation is itself a stochastic process. Formally the solution is a process adapted to the filtration $\mathbb{F}$, genenerated by W_t on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F}, \mathrm{P})$. Such a filtration is defined $\mathbb{F} = \sigma(W_s : 0 \leq s \leq t)$ and is the smallest σ-algebra containing all the events of the Wiener process

up to time t. A more rigorous definition of Equation (2.1) is written in the form of an stochastic integral equation as

$$X_t = x_0 + \int_0^t \mu(s, X_s)\mathrm{d}s + \int_0^t \sigma(s, X_s)\mathrm{d}W_s, \tag{2.2}$$

where x_0 is the initial value.

The previous definition of a SDE leads to the question of whether a solution exists. Existence of solutions to SDEs such as described in Equation (2.2) are non-trivial but do exist under a certain conditions. In general it is the case that SDEs have stricter requirements on its constituents for existence of solutions than regular PDEs. When discussing solutions of SDEs two common condition are Lipschitz continuity and linear growth conditions discribed in the following prepositions.

Definition 2.3.2 (Lipschitz continuity)**.** A function $\mu : \mathbb{R}^n \times [0, T] \to \mathbb{R}^m$ is said to be locally Lipschitz continuous on a subset $D \subseteq \mathbb{R}^n$ uniformly in t if there exists a constant $L \geq 0$ such that

$$\|\mu(x, t) - \mu(y, t)\| \leq L \|x - y\|$$

for all $x, y \in D$, $t \in [0, T]$. If the previous statement hold for the whole domain $D = \mathbb{R}^n$ then the Lipschitz condition is said to be global.

Definition 2.3.3 (Linear growth bound)**.** A function $\mu : \mathbb{R}^n \times [0, T] \to \mathbb{R}^m$ is said to satisfy linear growth on a subset $D \subseteq \mathbb{R}^n$ uniformly in t if there exists a constant $L \geq 0$ such that

$$\|\mu(x, t)\| \leq L(1 + \|x\|)$$

for all $x \in D$, $t \in [0, T]$. If $D = \mathbb{R}^n$ then the growth bound is said to be global.

The previous prepositions can be used to prove the existence of a strong unique solution for every SDE, given that it satisfies three conditions. Following is the theorem stated without proof.

Theorem 2.3.1 (Existence and uniqueness of solutions)**.** *For the SDE described in Equation* (2.2)*, if the following three conditions are satisfied*

1. *The coefficents μ and σ are locally Lipschitz.*

2. *The coefficents μ and σ satisfy the linear growth condition.*

3. *X_0 is independent of $(W_t, 0 \leq t \leq T)$, and $\mathbf{E}[(X_0)^2] < \infty$.*

Then there exists a unique strong solution X_t to the SDE. Moreover the SDE has continuous paths.

A proof of Theorem 2.3.1 is given by Theorem 5.4 in [5]. As is stated in the theorem the solution will not only exist but be unique. For those unfamiliar with SDEs, the

meaning of the term "unique solution" differs slightly from that of regular PDEs. The solution is adapted to the filtration of the Wiener process. This means that when solving the SDE numerically two independent solutions will not necessarily have equal paths. These solutions do however still belong to the same unique solution due to the fact that they belong to the same filtered probability space.

2.3.1 Euler-Maruyama method

SDEs like the one in Equation (2.2) rarely have a closed form solution. Therefore the solution to an SDE is often calculated through numerical methods. One such method is the Euler-Maruyama method which is used in this thesis. The method was introduced by G. Maruyama as an extension to Euler's method of solving differential equations into stochastic equations. The premise of the method is the partition of the time-intervall $[0, T]$ into N subintervalls $0 = \tau_0 < \tau_1 < ... < \tau_N = T$, where $\tau_{n+1} - \tau_n = T/N = \Delta t$. For this partition Equation (2.2) can then be written

$$X_{\tau_{n+1}} = X_{\tau_n} + \int_{\tau_n}^{\tau_{n+1}} \mu(s, X_s)\mathrm{d}s + \int_{\tau_n}^{\tau_{n+1}} \sigma(s, X_s)\mathrm{d}W_s.$$

For a small enough partition an approximation of $\mu(s, X_s) \approx \mu(\tau_n, X_{\tau_n})$ and $\sigma(s, X_s) \approx \sigma(\tau_n, X_{\tau_n})$ for $s \in [\tau_n, \tau_{n+1}]$ can be done making the integrations independent of the integrands giving the solution:

$$X_{\tau_{n+1}} = X_{\tau_n} + \mu(\tau_n, X_{\tau_n})\Delta t + \sigma(\tau_n, X_{\tau_n})\Delta W_{\tau_n}, \tag{2.3}$$

where $\Delta W_{\tau_n} = W_{\tau_{n+1}} - W_n$. The fundamental characteristics of a Wiener process dictates that the process W_t is independent of the past and that increments are Gaussian distributed as $W_{\tau_{n+1}} - W_{\tau_n} \sim \mathcal{N}(0, \tau_{n+1} - \tau_n)$. For a given discretisation of the Ito process $X = \{X_t, 0 \leq t \leq T\}$ given by $\bar{X}_n = X_{\tau_n}$ Equation (2.3) can be formulated as an iterative method of solving a SDE as

$$\bar{X}_{n+1} = \bar{X}_n + \mu(\tau_n, \bar{X}_n)\Delta t + \sigma(\tau_n, \bar{X}_n)\Delta W_{\tau_n}. \tag{2.4}$$

This scheme is the so called Euler-Maruyama method of solving SDEs. The statement of the so called weak convergence of the Euler-Maruyama method is presented in the following theorem.

Theorem 2.3.2. *Assuming that μ and σ satisfy global Lipschitz condition and have bounded derivatives. Let X and $\bar{X}$ be the solution to Equations* (2.2) *and* (2.4) *respectively with $X_0 = \bar{X}_0$. Then for a function g it holds that*

$$\left\| \mathbf{E}[g(X_T)] - \mathbf{E}[g(\bar{X}_T)] \right\| \leq C\Delta t,$$

for some constant $C \geq 0$.

Proof of Theorem 2.3.2 can be found in Theorem 14.1.5 in [3]. The implication of this theorem is that solving the expectation of a SDE can be made arbitrarily close to the real solution given a small enough step-length Δt.

3

Artificial neural networks

Recent advancements in machine learning has sparked an interest in machine learning in society as a whole. Although it is now common knowledge what neural networks are capable of doing, not everybody knows about the underlying theory. This chapter provides an introduction to concepts from machine learning relevant for this thesis. The chapter is split into two parts: Multilayered perceptrons (MLP) and Wasserstein Generative Adversarial Networks (WGAN).

3.1 Multilayered perception

The principle of the MLP is that data is propagated through the network one layer at a time. In Figure 3.1 an example layout of the MLP is shown. In the figure, the black circles represent the input layer of the network, whereas the white circles represent neurons. All but the first and last layers in the MLP are referred to as "hidden layers". They are called hidden since they are not observable. Referring to a neural network as a black box method acknowledges the fact that there are no means of interpreting the network at this state. If the number of hidden layers are large, the network is referred to as "deep". The right-most layer of neurons represent the output layer. Each neuron consists of an activation function, a weight vector and a threshold. The weights govern the flow of data in the network. Each arrow in the figure corresponds to an element of a weight vector. In the MLP all the data flows from left to right and never backwards. This is referred to as the MLP being a "feed-forward" network. The MLP is also "fully-connected", meaning that every node from the previous layer is connected to every node in the next. For each layer μ, the value of every neuron $V_i^{(\mu)}$ is determined by

$$g(\langle a_l, x \rangle + b_l),$$

where g is the activation function, x is the incoming data, a_l and b_l are the weight vector and threshold corresponding to neuron l in layer μ. Thereafter the values of $V_i^{(\mu)}$ are propagated to the next layer in the same manner, until an output is reached. The weights and thresholds constitute the model hyperparameters Θ. [14]

Neural networks such as MLPs are adaptable to a wide range of problems. By adjusting the number of hidden layers and neurons of the network it is in theory

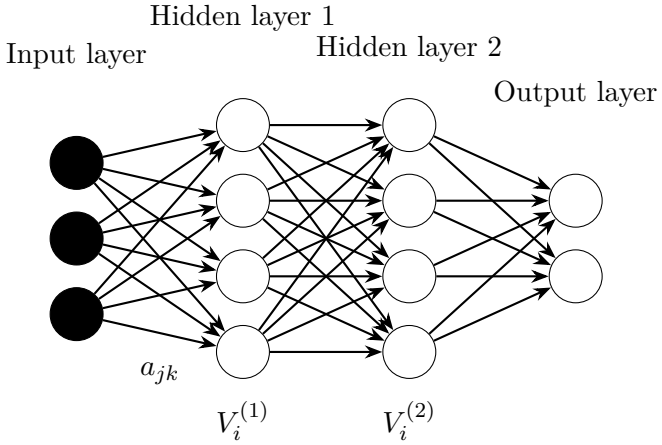


Figure 3.1: An example of how a MLP might be constructed. In this example each circle represents a neuron. The architecture consists of 3 inputs, followed by two hidden layers with 4 neurons each which are in turn connected to an output layer consisting of 2 outputs. The arrows correspond to the connection between neurons. Each connection has a specific value.

possible to approximate any function[1]. The limitation of this adaptability lies in the fact that such optimisation is non-trivial. Training of neural networks is computationally costly. This means that it is not possible to test all combinations of hidden layers and neurons for each problem. How then is the architecture chosen? The truth is that for almost all applications, architecture is chosen by intuition. It is obvious that a complex problem requires a more complex network architecture but if that corresponds to $1, 5, 10, 50...$ hidden layers is often not possible to investigate completely. The general method is thus to initiate a network which is believed to yield good results. Thereafter if the architecture proves ineffective then it it modified, this process is then repeated until an architecture yielding satisfying result is found. From this it is logical to think that most results obtained in the field of machine-learning could be improved upon by hyper optimisation but since neural networks deals with approximation this is not a problem as long as the error is negligible.

3.1.1 Activation function

The specifics of the problem play a roll in choosing an appropriate activation function. To each problem, different activation function are more suitable. If the activation function were chosen to be constant then the neural network could only approximate constant functions. In [2] proof is laid forth for the case that almost all activation functions can be used in MLPs to approximate a continuos function arbitrarily well (with the caveat that the functions meet some criteria), given that the network has sufficiently many neurons.

3.1.2 Training the MLP

As stated above, with sufficiently many neurons, approximation of any function arbitrarily well is possible. A neural network such as an MLP can thus be seen as a function dependent on both the input data but also of its hyperparameters, which consists of weights and thresholds. The goal of the network is to approximate some function, this function is called the target function, or just target. Training of the network therefore consists of tuning the hyperparameters as to obtain as good an approximation of the target as possible. The logical question then becomes how do we compare the output of the network with the target? In machine learning this is achieved by defining a loss function. A loss function is a metric which converts the difference between the output and target, which may be high-dimensional, into a scalar. The way to interpret this loss function is in the sense of a unit-less error. In this way a lesser loss function corresponds to a better model. In theory zero loss would equal a perfect approximation, however in practice this is most often not possible nor required. The choice of loss function is essential since it determines in which sense the network will be trained to mimic the target.

Once a loss function is determined, it is helpful to view it as a function $L(\Theta)$ dependent on the hyperparameters Θ. Thereafter training of the network consists of tuning the hyperparameters of the network as to minimize $L(\Theta)$. This is no easy task since Θ very well may consist of thousands of parameters. Numerically this can be done iteratively using some optimising algorithm. The algorithm is most commonly gradient decent based. The algorithm used in this thesis is Adam [9]. Once the iterative optimisation has converged, the network is trained.

3.2 Wasserstein Generative Adversarial Networks

Consider the problem of using a MLP to approximate a probability density. In theory this problem is solvable by training the network to minimise some distance to the true density. In many problems however it is not tractable to determine the true densities by pure sampling. The number of samples required to determine the posterior makes this method of training a MLP non-tractable numerically. In order to solve these problem a more advanced method is thus required. One such method relies on the Wasserstein Generative Adversarial Network (WGAN). Before this specific network is introduced its predecessor, the Generative Adversarial Network (GAN), is introduced to provide some context.

In [6] GAN was introduced as a new framework for constructing generative neural networks. The method of GANs relies on the competition between two neural networks. The two networks competing are called the generator and the discriminator. The generator is a network with the purpose of being trained to mimic a distribution, in this sense it is generative since it has the ability to generate data like the one it has been trained on. Simultaneously the discriminators purpose is providing as hard a setting as possible for the generator. The GAN method is often presented as a two player game. Simply put the game consists of the generator trying to minimize

a loss function whereas the discriminator tries to maximize the same function. This competition is advantageous in machine learning since is results in the generator being trained to perform in hard circumstances. The method was well received and seen as a breakthrough in the field of machine learning [10], it did however lack in some areas. Some of the problems include vanishing gradients. Vanishing gradient is a problem in machine learning which means that the training process stops prematurely because the gradient descent has reached a flat area. Furthermore it had the problem of mean collapse. A phenomenon where varied inputs collapse into too few targets resulting in a inadequate model [16]. It is however still worth praising the method for introducing the concept of adversarial learning.

Building upon the GAN, in [8] the WGAN is introduced. The WGAN utilises the same general structure of generator and discriminator as the GAN. The main difference between the two methods is the usage of the Wasserstein-1 metric as a loss function in the WGAN, hence its name. The Wasserstein-1 metric can be used as a metric of comparing two probability measures P_r and P_g, and is defined as

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbf{E}\left[\|x - y\| \right], \tag{3.1}$$

where $\Pi(P_r, P_g)$ denotes the set of all joint probability distributions $\gamma(x, y)$ whose marginals are respectively P_r and P_g. Another name for the Wasserstein-1 metric is the "Earth-Mover" distance. This name comes from the intuitive interpretation of $\gamma(x, y)$ representing how much (probability-)"mass" must be transported from x to y in order to transform the distribution P_r into P_g. The Wasserstein is therefore the "cost" of the optimal transformation plan γ. The usefulness of the Wasserstain-1 metric when training neural networks is evident from the following theorem:

Theorem 3.2.1. *Let P be a distribution on a compact space $\mathcal{X}$ and $(P_n)_{n\in\mathbb{N}}$ be a sequence of distributions on $\mathcal{X}$. Then for the limit $n \to \infty$ the following statements are equivalent*

1. *$W(P_n, P) \to 0$.*

2. *$P_n \xrightarrow{\mathcal{D}} P$ where $\xrightarrow{\mathcal{D}}$ represents convergence in distribution for random variables*

The proof of this theorem can be found in the aforementioned paper on WGAN [8]. Worth noting from the definition of the Wasserstein-1 metric is that it is a non-negative. Therefore the implication of Theorem 3.2.1 is that decreasing the Wasserstein-1 distance will eventually lead to matching distributions.

Although mathematically correct, numerical implementation of Equation (3.1) is not tractable due do heavy numerical costs. The Kantorovich-Rubenstein duality proves that the Wasserstein-1 distance can be written

$$W(P_r, P_g) = \sup_{\|\varphi\| \leq 1} \mathbf{E}_{x \sim P_r}[\varphi(x)] - \mathbf{E}_{x \sim P_g}[\varphi(x)],$$

where the supremum is taken over all 1-lipschits functions [4]. The result is a function which is applicable as a loss function for training a network to approximate

a distribution. Conversion of this Wasserstein-1 metric into a WGAN network is done by approximating the function φ by a neural network φ^γ parametrised by γ. Recalling the roles of the generator and discriminator in the GAN method, it becomes clear that this network φ^γ is the discriminator. Let P^θ be a parametrized measure. If the goal of this measure is to mimic another measure P then such a network can be trained by minimization of $W(P^\theta, P)$. Thus the loss function for training a neural network to mimic a distribution is done by optimising the parameters θ for the loss function

$$\inf_\theta \sup_{\|\varphi^\gamma(x)-\varphi^\gamma(y)\|\leq\|x-y\|} \mathbf{E}_{x\sim P^\theta}[\varphi^\gamma(x)] - \mathbf{E}_{x\sim P}[\varphi^\gamma(x)]. \tag{3.2}$$

As previously stated, the goal of the generator is to minimize the loss. It is therefore clear that in the above loss function it is the network parametrizing the distribution which constitutes the generator.

4

WGAN as a method of Bayesian filtration

The aim of this thesis is to investigating how a WGAN can be utilised to solve a filtration problem. This chapter provides the specifics of the filtration problem along with an explanation to how the WGAN presented in Chapter 3 can be implemented to solve this filtration problem.

4.1 General setting

Let X_t be a stochastic process. In previous chapters it has been discussed how the dynamics of stochastic processes might be modelled by SDEs. In differential form the SDE is written

$$\mathrm{d}X_t = \mu(t, X_t)\mathrm{d}t + \sigma(t, X_t)\mathrm{d}W_s, \quad t \in [0, T], \quad X_0 = x_0. \tag{4.1}$$

Solving this equation produces a function like object X_t which describes the evolution of the stochastic variable over time. This solution is most often referred to as a "path" or "trajectory". Now imagine a process in nature described by the aforementioned dynamics. In practice, observation of such a trajectory must be done discretely. Some methods might allow high-frequency observation, which emulates continuity, this thesis however deals with scarce observations. Scarce observations are observations separated in time as to no direct deduction of the continuity between observations can be made. Building upon this theory of measurements, it is also possible to imagine a system in which the trajectory is not directly observable. The measurements might instead be of the form of functions of the path. To add further likeness to the real world, such a measurement is most likely subject to some sort of measurement noise. The noise is commonly assumed to be additively Gaussian, this assumption is also applied in this thesis. Formally, for N measurements which have the true values X_{t_n}, $n = 1, 2, ..., N$, the measurements denoted Y_n are calculated through the model

$$Y_n = h(X_n) + q_n, \quad q_n \sim \mathcal{N}(0, Q_n), \quad n \in \{1, 2, ..., N\} \tag{4.2}$$

where h is the measurement function. The combination of Equations (4.1) and (4.2) is the data generating model for this thesis.

The problem of extracting information about a state X_t from incomplete noisy measurement is knowns as a "filtration problem" in the field of stochastic processes. There are many ways of formulating filtering problems, the problem in this thesis involves determining information about the final state given the noisy measurements. More specifically the problem at hand is determining the conditional distribution of the final position. In mathematical terms this would correspond to determining $P(X_T|\{Y_n\}_{n=1}^N)$, for a variable X_T and set of measurements $\{Y_n\}_{n=1}^N$ given by Equations (4.1) and (4.2) respectively. The set of measurements $\{Y_n\}_{n=1}^N$ will henceforth be referred using $Y_{1:N}$ as short-hand notation.

4.2 Implementing of WGAN

In Chapter 3 the basic structure of a WGAN is discussed in detail. To implement the WGAN it is necessary to formulate a generator and discriminator which aligns both with the specifics of the problem in addition the optimisation of Equation (3.2). Before formulating the structure of the generator and the discriminator it is helpful to convert the general loss function of the WGAN into a more problem specific one. The filtering problem at hand intends to train a network to approximate $P(X_T|Y_{1:N})$. Letting $P^\theta(X_T|Y_{1:N})$ be a neural network parametrised by θ with the purpose of learning the true distribution $P(X_T|Y_{1:N})$, then the problem specific loss function becomes

$$\inf_\theta \sup_{\|\varphi^\gamma(x)-\varphi^\gamma(y)\|\leq\|x-y\|} \mathbf{E}^\theta[\varphi^\gamma(X_T)|Y_{1:N}] - \mathbf{E}[\varphi^\gamma(X_T)|Y_{1:N}].$$

Furthermore, since the loss function depends on stochastic processes it is numerically advantageous to do an average of this when implementing it as a loss function. Doing so provides a smoother gradient leading to easier optimisation. Thus the final loss function becomes

$$\inf_\theta \mathbf{E}\left[\sup_{\|\varphi^\gamma(x)-\varphi^\gamma(y)\|\leq\|x-y\|} \mathbf{E}^\theta[\varphi^\gamma(X_T)|Y_{1:N}] - \mathbf{E}[\varphi^\gamma(X_T)|Y_{1:N}]\right]. \tag{4.3}$$

Formulating a structure for the generator and discriminator is a non-trivial task. There are numerous way of implementing. The implementation used in this thesis is presented in the upcomming sections.

4.2.1 Discriminator

From Equation (4.3) it is clear that the discriminator must be trained to approximate conditional expectation $\mathbf{E}[\varphi(X_T)|Y_{1:N}]$. Thus the discriminator for this problem becomes the map $(\varphi, Y_{1:N}) \to \mathbf{E}[\varphi(X_k)|Y_{1:N}]$. The question then becomes how such a network can be structured to approximate a rich class of functions $\varphi : \mathbb{R}^d \to \mathbb{R}$ while simultaneously being numerically tractable. The solution lies in how the approximation of φ is done by the discriminator. Letting φ^γ be a network approximation of φ. The approximation is done by a single layered perception consisting of L neurons, in mathematical terms the approximation can be written as

$$\varphi^\gamma(x) = \sum_{l=1}^{L} c_l g(\langle a_l, x\rangle + b_l), \tag{4.4}$$

where the parameters $\gamma = (a_l, b_l, c_l) \in \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}$ are the model parameters and g is the activation function. The parameters a_l and b_l should be seen as projections and translations of the variable x. Applying this approximation in the supremum part of Equation (4.3) allows for reformulation of the expected values as

$$\begin{aligned}
&\mathbf{E}^\theta[\varphi^\gamma(X_T) \mid Y_{1:N}] - \mathbf{E}[\varphi^\gamma(X_T) \mid Y_{1:N}] \\
\approx\ &\mathbf{E}^\theta\left[\sum_{l=1}^{L} c_l\, g(\langle a_l, X_T\rangle + b_l) \mid Y_{1:N}\right] - \mathbf{E}\left[\sum_{l=1}^{L} c_l\, g(\langle a_l, X_T\rangle + b_l) \mid Y_{1:N}\right] \\
=&\sum_{l=1}^{L} c_l \left[\mathbf{E}^\theta\left[g(\langle a_l, X_T\rangle + b_l) \mid Y_{1:N}\right] - \mathbf{E}\left[g(\langle a_l, X_T\rangle + b_l) \mid Y_{1:N}\right]\right]. \qquad (4.5)
\end{aligned}$$

The first expection value in this equation can easily be calculated by sampling from the parametrised distribution $P^\theta(X_T|Y_{1:N})$. It is however too costly to calculate the second expection value by sampling. The reason for this is the fact that the probability of sampling a conditional random variable $X_T|Y_{1:N}$, with the same measurement $Y_{1:N}$, decreases exponentially with N. In order to make the supremeum tractable we therefore introduce the function

$$\Xi(a, b, Y_{1:N}) = \mathbf{E}\left[g(\langle a_l, x\rangle + b_l) \mid Y_{1:N}\right].$$

Inserting this function into Equation (4.5) allows us to write

$$\begin{aligned}
&\mathbf{E}^\theta[\varphi^\gamma(X_T) \mid Y_{1:N}] - \mathbf{E}[\varphi^\gamma(X_T) \mid Y_{1:N}] \\
\approx&\sum_{l=1}^{L} c_l \left[\mathbf{E}^\theta\left[g(\langle a_l, x\rangle + b_l) \mid Y_{1:N}\right] - \Xi(a_l, b_l, Y_{1:N})\right].
\end{aligned}$$

The function Ξ is quite simple. The method used in this thesis is to approximate this function with a neural network Ξ^ξ. For this method to work Ξ^ξ is required to be generalised for a wide range of $a \in \mathbb{R}^d$ and $b \in \mathbb{R}$. As the dimension increases it becomes increasingly difficult to train a generalised network. Thus in an attempt to make the training more tractable a is restricted to be of unit length, which is justifiable if a is seen as a projection of X_T. This restriction means that the network is a function $\Xi^\xi : \mathbb{S}^{d-1} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ approximating the map $(a, b, Y_{1:N}) \to \mathbf{E}\left[g(\langle a, X_T\rangle + b)|Y_{1:N}\right]$. Training of this network is done by minimising the function.

$$\int_{\mathbb{S}^{d-1}} \int_{\mathbb{R}} \mathbf{E}\left[|g(\langle a, X_T\rangle + b) - \Xi^\xi(a, b, Y_{1:N})|^2\right] \mathrm{d}b\ \mathrm{d}a, \qquad (4.6)$$

where $\mathrm{d}a$ is Lebesgue surface measure over $\mathbb{S}^{d-1}$ and $\mathrm{d}b$ is a measure over $\mathbb{R}$. The exception part of the loss function is proven to train the network to approximate $\mathbf{E}[g(\langle a, X_T\rangle + b)]$ by Theorem 2.2.1. The integration ensures that the network is generalised for a and b.

After the network Ξ^ξ is trained the discriminator part of the loss function consists of taking the supremum of Equation (4.3) over the parameters γ. A numerical problem

which then arises is how we ensure that the parameters γ corresponds to a function φ^γ which fulfills the 1-Lipschitz restriction. In [7] a method of enforcing Lipschitz condition is proposed consisting of adding a gradient penalty term for φ^γ. The gradient proposed is

$$\mathbf{E}_{Z\sim\nu}\left[(\|\nabla\varphi^\gamma(Z)\|-1)^2\right].$$

Introducing this additonal penalty in the supremum ensure the discriminator to generate 1-Lipschitz functions. Furthermore Ξ^ξ is only trained for translations a of unit length. Therefore to encourage a to be close to unit length in the discriminator the penalty

$$\sum_{l=1}^{L}(\|a_l\|-1)^2$$

is also introduced. This last penalty term cannot ensure that a is always of exact unit length. It is therefore advantageous to relax the constraint on a when training Ξ^ξ to allow for a thin shell.

4.2.2 Generator

A lot of the ground-work for the WGAN is layed forth when discribing how to implement the discriminator. The purpose of the generator is to approximate the conditional distribution $P^\theta(X_T|Y_{1:N})$. This is done by optimising θ for the minmax problem proposed in Equation (4.3). Parametrisation of a distribution in a tractable way can be done in a multitude of ways. The method used in this thesis is that of letting the generator be a function mapping the measurements $Y_{1:N}$ to samples from the distribution.

4.3 Numerical implementation of WGAN

Training of the WGAN requires numerical approximation of Equation (4.3). For the outer expection, use Monte-Carlo with $I \geq 1$ samples denoted by $Y_{1:N}^{(1)},...,Y_{1:N}^{(I)}$. With the Monte-Carlo Equation (4.3) simplifies to

$$\inf_\theta \frac{1}{I}\sum_{i=1}^{I}\left[\sup_{\|\varphi^\gamma(x)-\varphi^\gamma(y)\|\leq\|x-y\|}\mathbf{E}^\theta[\varphi^\gamma(X_T)|Y_{1:N}^{(i)}]-\mathbf{E}[\varphi^\gamma(X_T)|Y_{1:N}^{(i)}]\right].$$

Remembering to calculate the supreme $\gamma^{(i)}$ for each unique $Y_{1:N}^{(i)}$. Now introduce the same representation of φ^γ as in Equation (4.4). Rembering that $\Xi^\xi(a_l,b_l,Y_{1:N}^{(i)}) \approx \mathbf{E}\left[g(\langle a_l,x\rangle+b_l)\mid Y_{1:N}\right]$ to obtain

$$\inf_\theta \frac{1}{I}\sum_{i=1}^{I}\sup_{\|\varphi^\gamma(x)-\varphi^\gamma(y)\|\leq\|x-y\|}\sum_{l=1}^{L}c_l\left[\mathbf{E}^\theta[g(\langle a_l,X_T\rangle+b_l)|Y_{1:N}^{(i)}]-\Xi^\xi(a_l,b_l,Y_{1:N}^{(i)})\right].$$

For each $i\in\{1,...,I\}$ sample $J\geq 1$ sample $X_T^{i,1},...,X_T^{i,J}$ from the conditional distribution $P^\theta(X_T|Y_{1:N}^{(i)})$. This Monte-Carlo approximation the optimisation problem reads

$$\inf_\theta \frac{1}{I}\sum_{i=1}^{I}\sup_{\|\varphi^\gamma(x)-\varphi^\gamma(y)\|\leq\|x-y\|}\sum_{l=1}^{L}\frac{1}{J}\sum_{j=1}^{J}c_l\left[g(\langle a_l,X_T^{(i,j)}\rangle+b_l)-\Xi^\xi(a_l,b_l,Y_{1:N}^{(i)})\right]. \quad (4.7)$$

Training of the WGAN using the objective function thus consists of concurrent minimisation and maximisation. Including the penalties ensuring the Lipschitz and unit projection constraints the maximisation problem becomes finding $\gamma^{(i)}$ which maximises

$$\sum_{l=1}^{L}\sum_{j=1}^{J} c_l\Big[g(\langle a_l, X_T^{(i,j)}\rangle + b_l) - \Xi^{\xi}(a_l, b_l, Y_{1:N}^{(i)})\Big] \\ - \alpha\,\mathbf{E}_{Z\sim\nu}\left[(\|\nabla\varphi^{\gamma}(Z)\| - 1)^2\right] - \beta\sum_{l=1}^{L}(\|a_l\| - 1)^2,$$

for each $i \in \{1, ..., I\}$ where $\alpha, \beta > 0$ are regularization hyperparameters. Chosing the activation function g to be the SoftPlus function, given by

$$g(x) = \frac{\log(1 + \exp(\lambda x))}{\lambda},$$

the approximation of φ^{γ} becomes

$$\varphi^{\gamma}(x) = \sum_{l=1}^{L} c_l \frac{\int_{-\infty}^{\lambda(\langle a_l, x\rangle + b_l)} \sigma(s)\mathrm{d}s}{\lambda} = \sum_{l=1}^{L} c_l \frac{\log(1 + e^{\lambda x})}{\lambda}.$$

The gradient is therefore equal to

$$\nabla\varphi^{\gamma}(x) = \sum_{l=1}^{L} a_l c_l \sigma(\lambda(\langle a_l, x\rangle + b_l)),$$

where σ is there sigmoid function,

$$\sigma(x) = (1 + \exp(-\lambda x))^{-1}$$

.The maximisation problem implemented in this thesis thus becomes

$$\sum_{i=1}^{I}\sum_{l=1}^{L}\sum_{j=1}^{J} c_l \left[\frac{\log\left(1 + \exp\left(\lambda(\langle a_l, X_T^{(i,j)}\rangle + b_l)\right)\right)}{\lambda} - \Xi^{\xi}(a_l, b_l, Y_{1:N}^{(i)})\right] \\ - \alpha\,\mathbf{E}_{Z\sim\nu}\left[\left(\left\|\sum_{l=1}^{L} a_l c_l\,\sigma\left(\lambda(\langle a_l, x\rangle + b_l)\right)\right\| - 1\right)^2\right] - \beta\sum_{l=1}^{L}(\|a_l\| - 1)^2. \quad (4.8)$$

Solving the Objective (4.8) is done numerically using a stochastic gradient descent method such as Adam [9]. Left for the maximisation part of the objective is choosing an appropriate ν. The purpose of ν is to impose a Lipschitz constraint in relevant areas. In lack of a better choice, the empirical measure

$$\nu = \frac{1}{2J}\sum_{j=1}^{J}\left(\delta_{X_T^{(i,j)}} + \delta_{X_T^{(i)}}\right),$$

where $X_T^{(i)}, ..., X_T^{(I)}$ are sampled from Equation (4.1), is chosen.

Remembering that the complete training procedure include repeted maximization and minimization we proceed with the minimization. Once the supremum is calculated, insert the maximized φ^γ into the Objective (4.7). From this objective it is clear that the latter part bears no importance to the optimization of θ. Thus the infimum used to optimise the parameters θ are done by minimizing

$$\frac{1}{IJ}\sum_{i=1}^{I}\sum_{l=1}^{L}\sum_{j=1}^{J} c_l \; g(\langle a_l, X_T^{(i,j)}\rangle + b_l). \tag{4.9}$$

In order to increase gradient flow in the numerical approximation in the Objective (4.9) importance sampling is implemented. The function used for optimizing the parametrised distribution is thus

$$\frac{1}{IJ}\sum_{i=1}^{I}\sum_{l=1}^{L}\frac{P^\theta(X_T^{(i,j)}|Y_{1:N}^{(i)})}{P^{\theta_{old}}(X_T^{(i,j)}|Y_{1:N}^{(i)})}\sum_{j=1}^{J} c_l \; g(\langle a_l, X_T^{(i,j)}\rangle + b_l). \tag{4.10}$$

A summary of the method of implementation is presented in Algorithm 1.

Algorithm 1 Summary of algorithm used to train WGAN

After the number of observation I, the number of samples per observation J, the number of neurons L constituting φ^γ and the hyperparameters α, β is chosen, initialize the networks required and perform training according to the algorithm:
1: Train Ξ^ξ to minimize Objective (4.6).
2: Set $p = 0$.
3: Generate I samples $(Y^{(i)})_{i=1}^I$. For each i generate J samples $(X_T^{(i,j)})_{i\in\{1,\dots,I\},j\in\{1,\dots,I\}}$ and $(\bar{X}_T^{(i,j)})_{i\in\{1,\dots,I\},j\in\{1,\dots,I\}}$ from the conditional distribution $P^\theta(X_T|Y)$ and Equation (4.1) respectively.
4: Set $\theta_{old} = \theta_p$.
5: Increase p by 1.
6: Find $(\gamma^i)_{i=1}^I$ which maximizes Objective (4.8).
7: Find θ which minimizes Objective (4.10).
8: Repeat steps (3)-(7) until convergence.

5

Methods from Bayesian inference

In the previous chapters a framework has been put forth describing how WGANs can be applied to solve filtering problems. This chapter introduces methods from Bayesian inference which share the WGANs purpuse and are to be used as methods of comparison. The filtering methods introduced are the Kalman filter (KF), Extended Kalman filter (EKF) and the bootstrap particle filter (PF).

5.1 Time-discrete state models

The filtration problem introduced in Chapter 4 deals only with determining partial information of the state variable, namely the final distribution. The KF, EKF and PF are methods which are similar in the sense that they are intended to provide information about the state variable at the measurements. They are however not directly applicable to time-dynamical state transitional systems such as that of the state in Equation (4.1). In order for them to be used for prediction the dynamics of the state variable must be time-discretized. Different dynamical systems requires different means of time-discretisation. For filters to be applied however it is necessary that the time dynamical model is rewritten on the form

$$\begin{aligned} X_{t_{n+1}} &= f(X_{t_n}) + q_n \\ Y_n &= h(X_n) + r_n, \quad n = \{1, 2, 3, \ldots, N\} \end{aligned}$$

where f is the transition function, h is the measurement function, $X_{t_n} \in \mathbb{R}^d$ is the state, $Y_n \in \mathbb{R}^g$ is the measurement, q_n is the process noise and r_n is the measurement noise. The partition t_n should match that of the partition in time for the measurements. There is no general means of discretisation. Some problems are can be discretised exactly whereas some require approximations.

5.2 Kalman filter

The Kalman filter (KF) is the closed form analytical solution to the filtering model where transition and measurements are linear with Gaussian distributed noise. In

mathematical terms such time-discrete state dynamics are those who can be written:

$$\begin{aligned} X_n &= F_{n-1}X_{n-1} + q_{n-1} \\ Y_n &= H_{n-1}X_n + r_n, \quad n = \{1, 2, 3, ..., N\}, \end{aligned} \tag{5.1}$$

where F_{n-1} is the transition matrix, H_{n-1} is the measurement matrix, $q_{n-1} \sim \mathcal{N}(0, \boldsymbol{Q}_{n-1})$ is the process noise and $r_n \sim \mathcal{N}(0, \boldsymbol{R}_n)$ is the measurement noise while the prior distribution is Gaussian $X_0 \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{P}_0)$. In probabilistic terms the model can be written

$$\begin{aligned} P(X_n|X_{n-1}) &= \mathcal{N}(X_n|F_{n-1}X_{n-1}, \boldsymbol{Q}_{n-1}) \\ P(Y_n|X_n) &= \mathcal{N}(Y_n|H_nX_n, \boldsymbol{R}_n), \quad n = \{1, 2, 3, ..., N\}. \end{aligned}$$

The origin of the filter is the derivation of a optimal estimator in the mean square error sense [12]. Therefore, when estimating a variable governed by dynamics according to Equation (5.1), the KF if often seen as the best possible estimator.

The KF can be split into two steps, the prediction step and the update step. The prediction step estimates the next state using the known model dynamics, predicting the most likely state along with the uncertainty. The update step refines prediction using the measurements, providing a more accurate prediction and reduced uncertainty. To apply a KF to a filtering problem the following is required to be known of the system:

- An initial mean $\boldsymbol{m}_0$ and covariance $\boldsymbol{P}_0$ of the initial position.
- The transition matrix $\{F_{n-1}\}_{n=1}^N$ with the corresponding process noise $\{\boldsymbol{Q}_{n-1}\}_{n=1}^N$.
- The measurement matrix $\{H_n\}_{n=1}^N$ with the corresponding measurement noise $\{\boldsymbol{R}_n\}_{n=1}^N$.

The filter algorithm is presented in Algorithm 2. A more detailed description of the KF is provided in [15].

Algorithm 2 Kalman Filter Algorithm

The filter is initialised with an initial mean $\boldsymbol{m}_0$ and covariance $\boldsymbol{P}_0$. Thereafter the filter applies the known transition matrix $\{F_{n-1}\}_{n=1}^{N}$, measurement matrices $\{H_n\}_{n=1}^{N}$, process noise $\{\boldsymbol{Q}_n\}_{n=1}^{N}$ and measurement noise $\{\boldsymbol{R}_n\}_{n=1}^{N}$ to do determine the position $\boldsymbol{m}_n$ and uncertainty $\boldsymbol{P}_n$ at each time-step by iterative calculation of each measurement Y_n according to:

for $n = 1$ to N **do**

Prediction:

$\boldsymbol{m}^{-}{}_n = F_{n-1}\boldsymbol{m}_{n-1}$

$\boldsymbol{P}_n^- = F_{n-1}\boldsymbol{P}_{n-1}F_{n-1}^\top + \boldsymbol{Q}_{n-1}$

Update:

$\boldsymbol{v}_n = Y_n - H_n\boldsymbol{m}_n^-$

$\boldsymbol{S}_n = H_n\boldsymbol{P}_n^- H_n^\top + \boldsymbol{R}_n$

$\boldsymbol{K}_n = \boldsymbol{P}_n^- H_n^\top \boldsymbol{S}_n^{-1}$

$\boldsymbol{m}_n = \boldsymbol{m}_n^- + \boldsymbol{K}_n\boldsymbol{v}_n$

$\boldsymbol{P}_n = \boldsymbol{P}_n^- - \boldsymbol{K}_n\boldsymbol{S}_n\boldsymbol{K}_n^\top$

end for

5.3 Extended Kalman filter

The extended Kalman filter (EKF) is an extension of the classic Kalman filter. The extension of the filter allows for applying a similar filter method as the KF to non-linear state space and measurement models. The EKF is designed to filter dynamical models of the form:

$$\begin{aligned} X_n &= \boldsymbol{f}(X_{n-1}) + q_{n-1} \\ Y_n &= \boldsymbol{h}(X_n) + r_n, \quad n = \{1, 2, 3, \ldots, N\}. \end{aligned}$$

where $\boldsymbol{f}$ is the transition function, $\boldsymbol{h}$ is the measurement function, $q_{n-1} \sim \mathcal{N}(0, \boldsymbol{Q}_{n-1})$ is the process noise and $r \sim \mathcal{N}(0, \boldsymbol{R}_n)$ is the measurement noise while the prior distribution is Gaussian $X_0 \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{P}_0)$.

The idea of the EKF is the Gaussian approximation

$$P(X_n|Y_{1:n}) \approx \mathcal{N}(X_n|\boldsymbol{m}_k, \boldsymbol{P}_k)$$

to the filtering densities. The assumption here is that $X_k|Y_{1:k}$ is always Gaussian. For a generalised transition function $\boldsymbol{f}$ this is obviously not the case. Still this approximation can be beneficial if the transition function is close to linear. The derivation of the EKF is done by linearisation of the transition function and measurement function using Taylor series of the first order. The following is required to be known to apply the EKF:

- An initial mean $\boldsymbol{m}_0$ and covariance $\boldsymbol{P}_0$ of the initial position.
- The transition function $\boldsymbol{f}$ with the corresponding process noise $\{\boldsymbol{Q}_{n-1}\}_{n=1}^{N}$.

- The measurement function $\boldsymbol{h}$ with the corresponding measurement noise $\{\boldsymbol{R}_n\}_{n=1}^N$.

The filter algorithm is presented in Algorithm 3. A more detailed description of the EKF is provided in [15].

Algorithm 3 Extended Kalman Filter Algorithm

The filter is initialised with an initial mean $\boldsymbol{m}_0$ and covariance $\boldsymbol{P}_0$. Thereafter the filter applies the known transition function $\boldsymbol{f}$, measurement function $\boldsymbol{h}$, process noise $\{\boldsymbol{Q}_n\}_{n=1}^N$ and measurement noise $\{\boldsymbol{R}_n\}_{n=1}^N$ to do determine the position $\boldsymbol{m}_n$ and uncertainty $\boldsymbol{P}_n$ at each time-step by iterative calculation of each measurement Y_n according to:

for $n = 1$ to N **do**

Prediction:

$\boldsymbol{m}^-{}_n = \boldsymbol{f}(\boldsymbol{m}_{n-1})$

$\boldsymbol{P}_n^- = \boldsymbol{F}_x(\boldsymbol{m}_{n-1})\boldsymbol{P}_{n-1}\boldsymbol{F}_x^\top(\boldsymbol{m}_{n-1}) + \boldsymbol{Q}_{n-1}$

Update:

$\boldsymbol{v}_n = Y_n - \boldsymbol{h}(\boldsymbol{m}_n^-)$

$\boldsymbol{S}_n = \boldsymbol{H}_x(\boldsymbol{m}_{\boldsymbol{n}}^-)\boldsymbol{P}_n^-\boldsymbol{H}_x^\top(\boldsymbol{m}_{\boldsymbol{n}}^-) + \boldsymbol{R}_n$

$\boldsymbol{K}_n = \boldsymbol{P}_n^-\boldsymbol{H}_x^\top(\boldsymbol{m}_{\boldsymbol{n}}^-)\boldsymbol{S}_n^{-1}$

$\boldsymbol{m}_n = \boldsymbol{m}_n^- + \boldsymbol{K}_n\boldsymbol{v}_n$

$\boldsymbol{P}_n = \boldsymbol{P}_n^- - \boldsymbol{K}_n\boldsymbol{S}_n\boldsymbol{K}_n^\top$

end for

Above $\boldsymbol{F}_x(\boldsymbol{m})$ and $\boldsymbol{F}_x(\boldsymbol{m})$ denote the Jacobian of the function $\boldsymbol{f}$ and $\boldsymbol{h}$, respectivly with elements

$$[\boldsymbol{F}_x]_{j,j'} = \left.\frac{\partial f_j(\boldsymbol{x})}{\partial x_{j'}}\right|_{\boldsymbol{x}=\boldsymbol{m}},$$

$$[\boldsymbol{H}_x]_{j,j'} = \left.\frac{\partial f_j(\boldsymbol{x})}{\partial x_{j'}}\right|_{\boldsymbol{x}=\boldsymbol{m}}.$$

5.4 Particle filters

Particle filters (PF) are methods of Bayesian inference used for solving filtering problems by the means of sampling. There exists many particle filter methods, this thesis however only applies the particle filter referred to as the "Bootstrap particle filter". Since this is the only filter in this thesis it will be referred to as "particle filter" or "PF" in this thesis. The particle filter is sequential Monte-Carlo based methods, meaning they rely on sampling to determine distributions. In Bayesian inference, sampling is powerful in the sense that with a large enough sample size any distribution can be sampled. The drawback is however that for certain problems, like determining conditional distributions, the number of samples required to solve a filtering problem with pure Monte-Carlo becomes far too great. The solution to this is to use knowledge about the dynamical system to make every sample count. One of the advantages of the PF is that it is applicable to any time-discrete state

space model

$$X_n = \boldsymbol{f}(X_{n-1}) + q_{n-1}$$
$$Y_n = \boldsymbol{h}(X_n) + r_n, \quad n = \{1, 2, 3, \ldots, N\}$$

where $\boldsymbol{f}$ is the transition function, $\boldsymbol{h}$ is the measurement function, $q_{n-1} \sim \mathcal{N}(0, \boldsymbol{Q}_{n-1})$ is the process noise and $r \sim \mathcal{N}(0, \boldsymbol{R}_n)$ is the measurement noise while the prior distribution is Gaussian $X_0 \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{P}_0)$.

The idea behind the PF is to utilise the fact that the measurement noise is Gaussian additive. Using Bayes' theorem this can be used to deduct that

$$P(X_k|Y_{1:k}) \propto P(Y_k|X_k). \tag{5.2}$$

Remembering that the PF is a sample based method, the idea behind it is as follows: $X_0^{(i)}$, $i = \{1, 2, \ldots, I\}$ samples are drawn from the prior distribution. Thereafter the particles are propagated one time-step using the dynamical model. To determine how representative these states are given the measurements, their probability are determined using Equation (5.2). According to their respective probability, a new set of particles are then sampled. This step is called "importance sampling". The idea is that at each time-step, only the most probable particles survive. For the PF to be accurate a relative large number of particles are required. The number of particles required compared to regular sampling methods is however reduced greatly. The filter algorithm is presented in Algorithm 4. A more detailed description of the Bootstrap filter is given in [15].

Algorithm 4 Bootstrap filter algorithm

The filter is initialised with an prior mean $\boldsymbol{m}_0$ and covariance $\boldsymbol{P}_0$. Thereafter the filter applies the known transition function $\boldsymbol{f}$, measurement function $\boldsymbol{h}$, process noise $\{\boldsymbol{Q}_{n-1}\}_{n=1}^N$ and measurement noise $\{\boldsymbol{R}_n\}_{n=1}^N$. The filter begins with I samples $X_0^{(i)}$ being drawn from the prior distribution, thereafter estimates $\boldsymbol{m}_n$ and covariances $\boldsymbol{P}_n$ are calculated at each time-step by the following algorithm.

for $n = 1$ to N **do**

1. Draw a new set of points $X_n^{(i)}$ from the dynamic model
2. Calculate the weights $w_k^{(i)} \propto P(Y_n|X_k^{(i)})$ and normalize to unity.
3. Resample $X_n^{(i)}$, $i = \{1, 2, \ldots, I\}$ from the old set $X_n^{(i)}$ using the weights $w_k^{(i)}$ as probabilities

end for

After this iterative algorithm is performed that distribution is represented by the samples $X_n^{(i)}$ at each time-step.

6

Models

This chapter introduces the SDEs used in this thesis along with some physical interpretations of the processes.

6.1 Ornstein-Uhlenbeck process

The Ornstein-Uhlenbeck(OU) process for a stochastic variable $X_t \in \mathbb{R}^n$ is defined by the SDE

$$\mathrm{d}X_t = \Theta(\mu - X_t)\mathrm{d}t + \Sigma \mathrm{d}W_t, \tag{6.1}$$

where $\Theta \in \mathbb{R}^{n\times n}$ is the mean-reversion rate, $\mu \in \mathbb{R}^n$ is the long term mean, $\Sigma \in \mathbb{R}^{n\times m}$ is the diffusion matrix and $W_t \in \mathbb{R}^m$ is a Wiener process in m dimensions. For Equation (6.1) to represent a valid OU-process Θ is required to have strictly positive eigenvalues. This is to ensure mean-reversion of the SDE, which is a prerequisite of the OU-process. Mean-reversion in the sense of SDEs means that over time the process tends to drift towards a mean function almost surely. In the case of the OU-process it will tend towards the mean μ.

The dynamics of masses coupled by springs and dampeners under the influence of noise constitutes a OU-process. A visualisation of the model is provided in Figure 6.1. Each mass has two degrees of freedom, position and velocity. The underlying

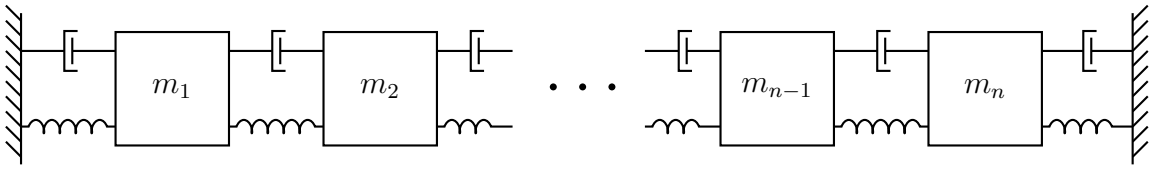


Figure 6.1: A visualisation of the spring-mass system.

assumption is that each mass interact only with its nearest neighbours. The two interacting forces present are linearly dependent on the position and velocity of the masses respectively. The force dependent on the position is calculated using Hooke's law whereas the dampening force is calculated in an equivalent manner. The dampening force can be linked to friction or equivalent phenomena in real-world problems. The equation of motion for the spring-mass system can be written

in differential form as

$$-m_i\ddot{x}_i = k_{i-1}(x_i - x_{i-1}) + k_i(x_i - x_{i+1}) + c_{i-1}(\dot{x}_i - \dot{x}_{i-1}) + c_i(\dot{x}_i - \dot{x}_{i+1}), \quad (6.2)$$

where the k are spring constants, c are dampening coefficients, m are masses, x is the position of the masses, $\dot{x}$ is the acceleration of the masses and $\ddot{x}$ is the velocity of the masses. The fact that the positions of the walls are fixed can is implemented by setting their position to zero, i.e $x_0 = x_{n+1} = 0$.

The system of equations given by Equations (6.2) is second order differential equation. Assuming a system containing n masses the solution obtain $2n$ degrees of freedom. By defining a state vector $[\mathbf{x}, \mathbf{v}]^T$, where $\mathbf{x}$ is all of the position in vector form and $\mathbf{v} = \dot{\boldsymbol{x}}$ is the velocities in vector form, then Equation (6.2) can be written as a first order differential equation:

$$\mathrm{d}\begin{bmatrix}\mathbf{x}\\ \mathbf{v}\end{bmatrix} = A\begin{bmatrix}\mathbf{x}\\ \mathbf{v}\end{bmatrix}\mathrm{d}t, \quad (6.3)$$

with A being the problem specific transition matrix. The transition matrix is a $2n \times 2n$ size matrix which in turn consists of four $n \times n$ matrices and is given by

$$A = \begin{bmatrix} 0 & I \\ M^{-1}K & M^{-1}C \end{bmatrix}.$$

Here M is a diagonal matrix containing the masses. K and C are matrices containing the spring and dampening coefficients. The symmetry of the forces results in them having similar structures. Explicitly written they have the form

$$K = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 & \cdots & 0 \\ -k_2 & k_2 + k_3 & -k_3 & \cdots & 0 \\ 0 & -k_3 & k_3 + k_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & -k_n \\ 0 & 0 & 0 & -k_n & k_n + k_{n+1} \end{bmatrix},$$

$$C = \begin{bmatrix} c_1 + c_2 & -c_2 & 0 & \cdots & 0 \\ -c_2 & c_2 + c_3 & -c_3 & \cdots & 0 \\ 0 & -c_3 & c_3 + c_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & -c_n \\ 0 & 0 & 0 & -c_n & c_n + c_{n+1} \end{bmatrix}.$$

Equation (6.3) is as of yet a ODE. To turn it into a SDE in the general form as described in Equation (2.1) a diffusion part is required. This thesis will regard the simplest case of $\sigma(t, X_t) = \sigma I$, for some constant σ. The physical interpretation of such noise could be justified but this is not necessary for this thesis. The final equation describing a noisy spring-mass system is thus given by

$$\mathrm{d}\begin{bmatrix}\mathbf{x}\\ \mathbf{v}\end{bmatrix} = A\begin{bmatrix}\mathbf{x}\\ \mathbf{v}\end{bmatrix}\mathrm{d}t + \sigma I dW_t.$$

The stochastic process which solves this equation is a so called Ornstein-Uhlenbeck process. The Ornstein-Uhlenbeck is one of the simpler stochastic processes. It has mean reverting properties, meaning that it has a set value to which it will revert indefinitely.

6.2 Bimodal

A bimodal stochastic process is a process which exhibits a probability distribution with two distinct peaks or modes. There are many examples of such processes. The dynamics for the bimodal process $X_t \in \mathbb{R}$ used in this thesis is given by the over damped Langevin equation

$$\mathrm{d}X_t = -U'(X_t)\mathrm{d}t + \mathrm{d}W_t,$$

where $W_t \in \mathbb{R}$ is a one dimensional Wiener process. The physical interpretation of the equation is that the dynamics of X_t is influenced both by a potential U and randomness simultaneously. Examples of real-world processes modeled by this dynamic is a Brownian particle being influenced by gravity. A bimodal distribution arises from the Langevin equation if the potential is a double well potential. The potential used for the bimodal example is $U(x) = -2x^2 + \frac{x^4}{4}$. This potential gives the system two minima at 2 and -2. The initial condition was set to $X_0 = 0$ and the timeframe was set to $t \in [0, 1]$ giving the dynamical model

$$\mathrm{d}X_t = (4X_t - X_t^3)\mathrm{d}t + \mathrm{d}W_t, \quad X_0 = 0, \quad t \in [0, 1]$$

where $W_t \in \mathbb{R}$ is a one dimensional Wiener process. On this process 5 discrete measurement where made at evenly spaced times. The measurements were linear and given by

$$Y_n = X_{t_n} + \mathcal{N}(0, 1), \quad n \in \{1, 2, 3, 4, 5\}.$$

In figure 6.2 the evolution of the distribution of X_t and Y_n is shown at times $t = 0.2, 0.6, 1$. The distribution of X_t becomes clearly bimodal but the distribution of the measurements less so, due to the relativly large noise. Since the measurements are not as bimodal it is logical to suspect that the conditional posterior might be somewhat bimodal.

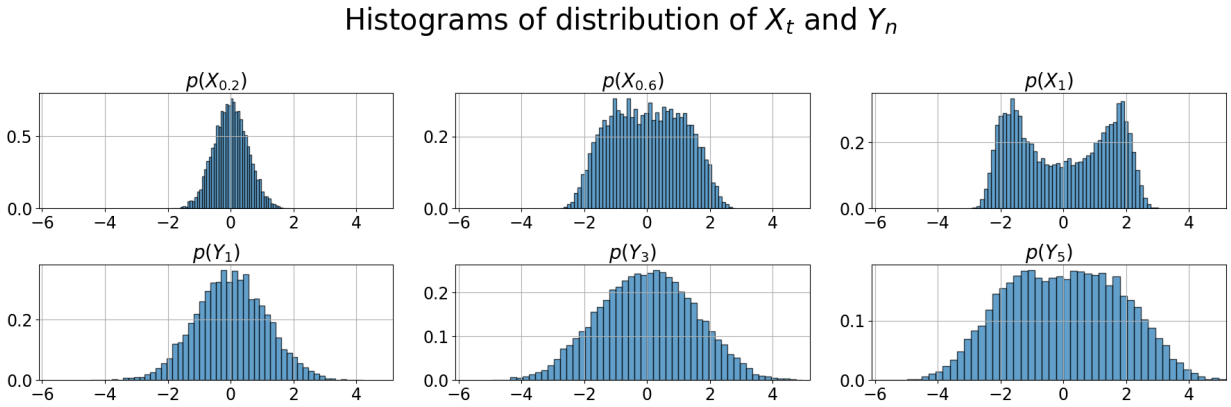


Figure 6.2: This figure illustrates how the distribution of X_t and Y_n evolves over time. From the left to right column the distribution for $t = 0.2, 0.6, 1$ is calculated using 10^4 samples. The state evolves into a bimodal distribution. The measurements however do not show the bimodality as clearly.

7

Results

This chapter provides the evaluation of the method of impleemting the WGAN. First metrics of evaluation are defined. Then the results for the trained WGAN are presented. An error analysis of the method is also provided along with a conclusion.

7.1 Evaluation metrics

This thesis aims in using neural networks, more specifically WGANs, to approximate posterior distributions. To evaluate this method, the first though which comes to mind is to calculate the true posterior distribution using a pure sampling method. This however not possible, as using such method, the number of samples required to approximate the distribution of a conditional stochastic variable $X_T|Y_{1:k}$ grows exponentially with N. Other methods of evaluation must therefore be introduced.

The first method introduced is a metric referred to as the Mean Absolute Error (MAE). It is of interest since it requires no actual distribution of comparison. The MAE measures the absolute error between the filter prediction and the true state. The metric is defined by

$$\text{MAE} = \frac{1}{M}\sum_{m=1}^{M}\left\|X_T^{(m)} - \mu_T^{(m)}\right\|,$$

where μ_T is the mean predicted by a filter and is calculated as an average of M samples. Worth noting is that this metric is not strictly an error. This means that it is not expected for the MAE to become zero, however comparison of MAEs between filter may give some indication of performance. The next two metrics are true errors, meaning that they requires a solution for comparison. As stated earlier no exact solution is available. Thus to calculate the upcoming two evaluation metrics a problem specific filter is chosen as a reference and is seen as the correct solution. The second metric of evaluation is a method called First Moment Error (FME) and is defined as

$$\text{FME} = \frac{1}{M}\sum_{m=1}^{M}\left\|\hat{\mu}_T^{(m)} - \mu_T^{(m)}\right\|,$$

where $\hat{\mu}_T$ denotes the mean calculated using the reference filter. The third and last metric is the Kullback-Leibler Divergence (KLD). This metric compares a reference

distribution $\hat{p}$ to an approximative distribution p. The metric is defined as

$$D_{\mathrm{KL}}(p||\hat{p}) = \int_{\mathbb{R}^d} p(x) \log \frac{p(x)}{\hat{p}(x)} \mathrm{d}x.$$

KLF is a distance metric comparing the similarity between two distribution. Since the accuracy of the WGAN (and other methods) might vary depending on the sample, an average of this metric is also calculated. A non-trivial fact about the KLD is that it is non-negative. It is only zero if the distributions are equal almost everywhere.

7.2 Results

This section provides the results of comparing the WGAN using the metrics defined in the previous section. The models tested are presented in Chapter 6.

7.2.1 Spring-mass system

The dynamics for the spring-mass system are presented in Chapter 6. The WGAN was tested for spring-mass systems with 1, 2 and 3 masses. In all of these cases the system was set up with masses $m_1, m_2, ... = 1$, spring coefficients $k_1, k_2, ... = 1$, dampening coefficients $c_1, c_2, ... = 1$, $\Sigma = I$ and simulated for $t \in [0, 1]$. In addition to this the initial position and velocity of the masses distributed as $\boldsymbol{x} \sim \mathcal{N}(0, I)$,$\boldsymbol{v} \sim \mathcal{N}(0, I)$ and the measurement where calculating according to

$$Y_n = X_{t_n} + \mathcal{N}(0, 1), \quad n = \{1, 2, 3, 4, 5\}.$$

This problem constitutes a linear system with linear measurements, resulting in the KF being the optimal solution and thus used as the reference filter. PFs with $N = 10^2, 10^3, 10^4, 10^5$ particles where used for comparison.

Before presenting the results some context to the problem is provided. In Figure 7.1 example trajectories are presented for each of the tested dimensions. On the right hand side of each trajectory is the conditional distribution of the final position, calculated using the WGAN, KF and a particle filter with 10^4 particles.

Evaluation of the WGAN was done using MAE, FME and KLD with a KF as the reference filter, defined in the beginning of this chapter. The average of these metrics were calculated over 1000 samples. The resulting metrics are presented in Figure 7.2. The results metrics are plotted against the dimension of the filtering problem. From the figure it is evident that the WGAN was outperformed by all filters with respect to MAE in all dimensions. MAE is not a true error, but a lower MAE does in general indicate a better filter. The filter with the lowest MAE was the KF, which is to be expected since it is the optimal solution to the problem. For the system consisting of 1 and 2 masses (corresponding to 2 and 4 dimensions) the WGAN outperformed the PF with 10^2 particles. This does to some extend validate the method since it proves that the WGAN learns to approximate the true distribution to some extend.

(a)

(b)

(c)

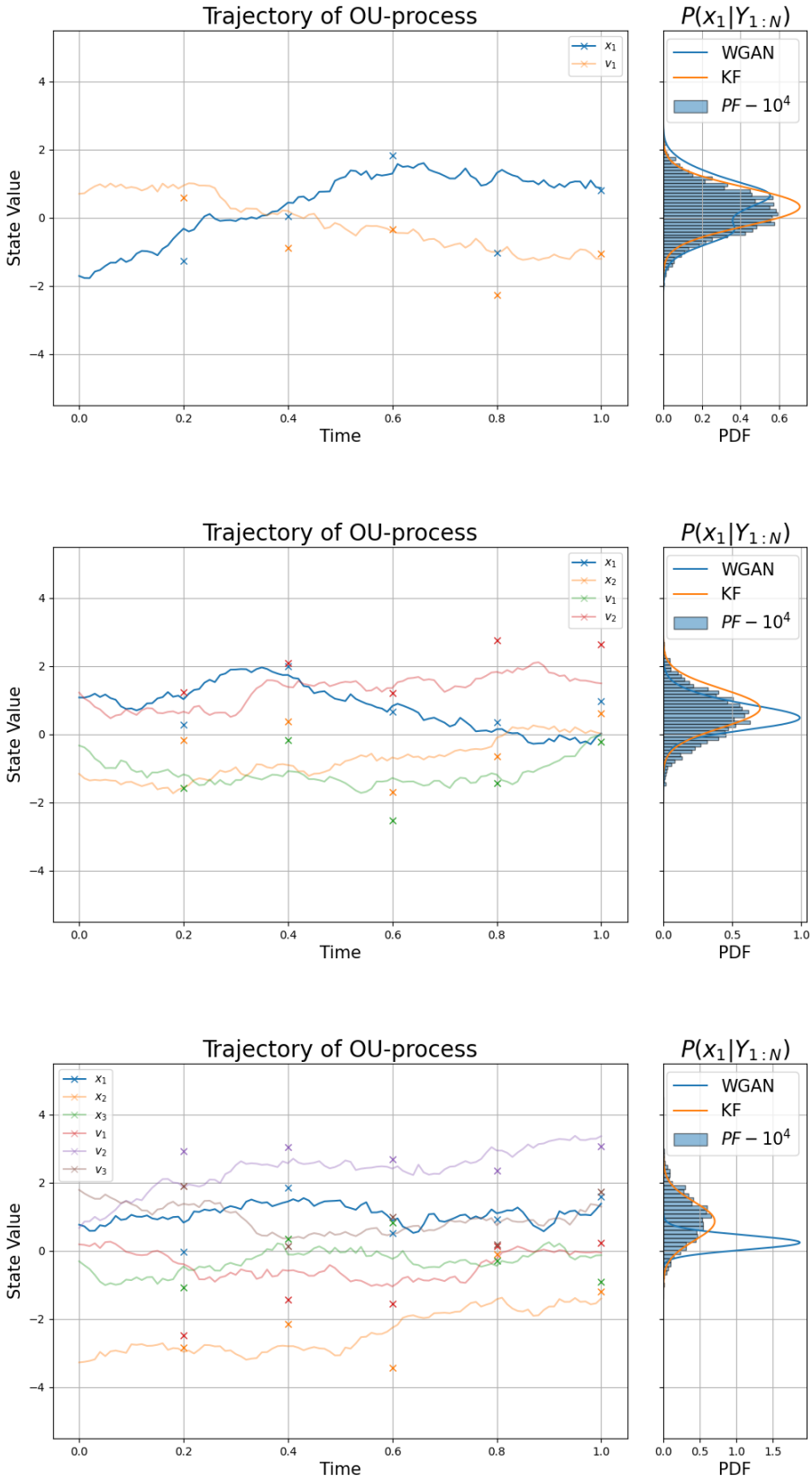


Figure 7.1: Example of data generated using the spring-mass SDE. A sample trajectory is calculated. To the right side of the trajectories the posterior distribution calculated using the WGAN, KF and PF with 10^4 is shown for the highlighted trajectory. Figure a), b) and c) shows the results for 1,2 and 3 masses respectively.

It is however worth noting that for problems of these dimension such a low number of particles are not expected to be reliable. Still it shows that the WGAN trained is better than random. For 3 masses (corresponding to 6 dimensions) the WGAN was outperformed in all metrics. Sources of error in the implementation of the WGAN is discussed later in this chapter.

The errors presented in 7.2 does increase with the dimension. This is not necessarily due to the filters performing worse as the dimension grows. It is true that higher dimension require a larger sample size for sampling based methods such as the PF to keep up performance. However, the metrics of evaluation are based on norms which are expected to grow with dimension. Thus not all the increase in error can be attributed to worse solutions. Some of the error can simply be explained by the curse of dimensionality.

7.2.2 Bimodal process

In this section the results for the bimodal system is presented. The bimodal system was calculated using Langevin dynamics, as described in Chapter 6. The observation of the process consists of ten evenly spaced measurements with normal distributed noise according to

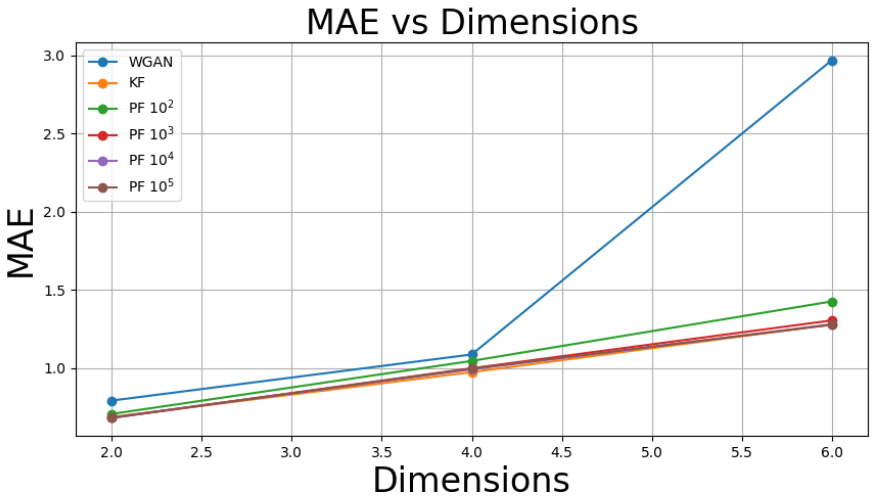
$$Y_n = X_{t_n} + \mathcal{N}(0, 1), \quad n = \{1, 2, 3, ..., 10\}.$$

The dynamics of this process is non-linear. The EKF can be used to solve the filtering problem, although it is not an exact solution. The metrics used require a reference filter, for this purpose the PF with 10^5 particles was used. In addition to the previously mentioned filter particle filters with 10^2,10^3 and 10^4 were used for comparison.

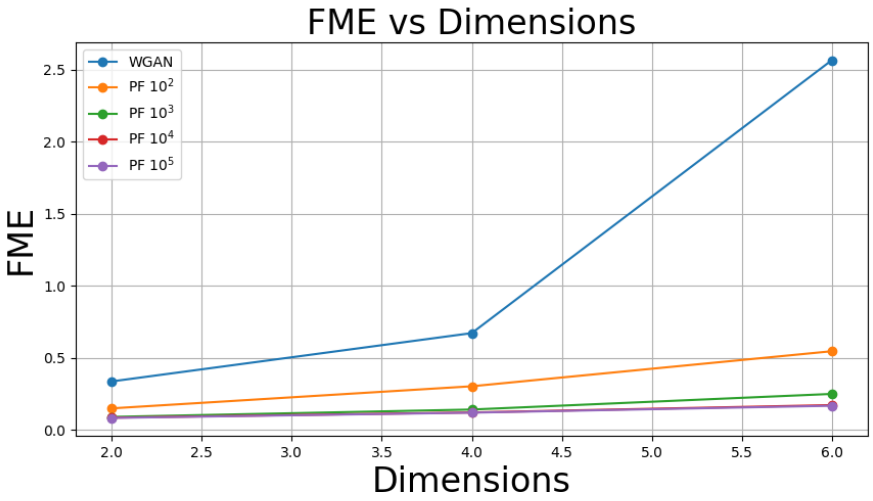
Before presenting the results some context to the problem is provided. In Figure 7.3 an example trajectory of the bimodal system is presented. The left hand figure show the trajectory of the true state along with the measurements. The right hand figure show an example of a posterior distribution $P(X_T|Y_{1:10})$ calculated using the WGAN, EKF and PF with 10^4 particles.

The WGAN was evaluated using the metrics MAE, FME and KLD, defined in the beginning of the chapter. The average of these metrics was calculated over 1000 samples. The resulting metrics are presented in Table 7.1. The WGAN was outperformed by all filters with respect to MAE. MAE is not a true error, but it is the case that a better filter should provide a smaller MAE. The filter with the lowest MAE was the PF with 10^5 particles, proving that it was a good choice as reference. As for the other metrics, the WGAN was outperformed by all filters except the particle filter with 10^2 particles. This is to some extent validation of the method since it proves that the implementation of the WGAN learns about the true distribution to some extend. It is however worth noting that 10^2 particles is far to low for a method such as the PF. Sources of error in the implementation of the WGAN is discussed later in this chapter.

(a)



(b)



(c)

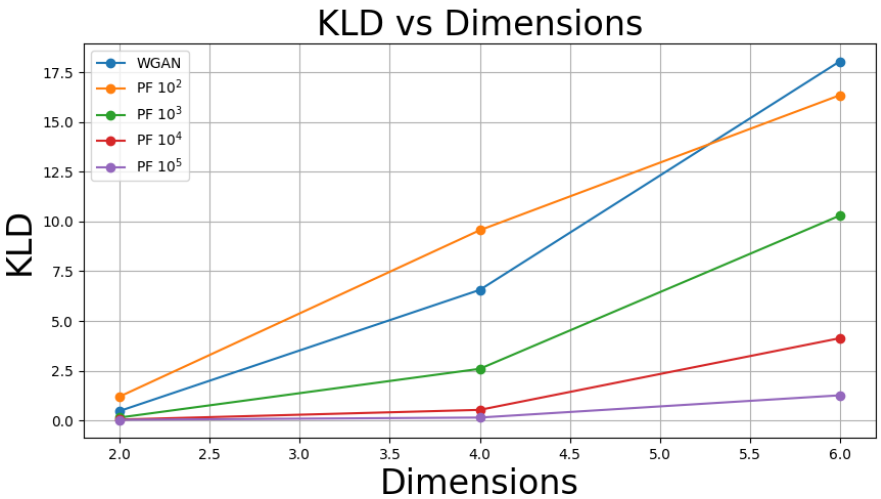


Figure 7.2: Figure showing the metrics of evaluation for 2, 4 and 6 dimensions. In a) the MAE is presented, in b) the FME is presented and in c) the KLD is presented. For the FME and KLD the KF was used as a reference. All of the metrics are averages over 1000 samples.

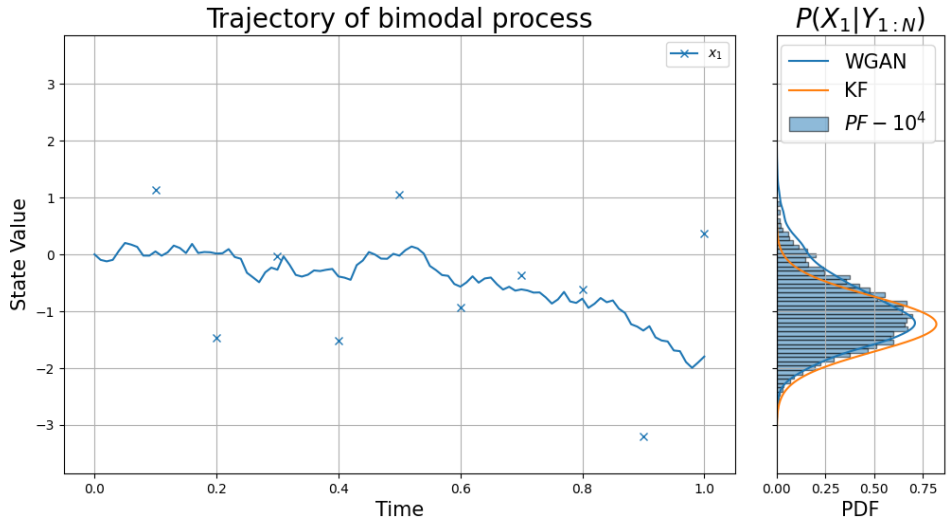


Figure 7.3: Example of data generated using the bimodal SDE. The left hand side shows the trajectory of the process and to the right is the corresponding conditional posterior distribution calculated using WGAN, KF and PF with 10^4 particles.

Table 7.1: Table showing the metrics for the WGAN, EKF and particle filters with $N = 10^2, 10^3, 10^4, 10^5$ particles. For the FME and KLD the particle filter with $N = 10^5$ was used a a reference.

	WGAN	EKF	PF-10^2	PF-10^3	PF-10^4	PF-10^5
MAE	0.4093	0.3747	0.3755	0.3669	0.3668	0.3665
FME	0.1557	0.0800	0.0899	0.0281	0.0087	–
KLD	0.1614	0.0476	0.2134	0.0177	0.0018	–

7.3 Error analysis

The method of implementing the WGAN is split into the construction of a discriminator and generator. In this thesis the generator is rather simple. The objective which the generator aims to solve does not require that much ingenuity. The method of implementing the discriminator however is somewhat unique. Thus the main area to improve is suspected to be that of the discriminator. The main objective of the discriminator is the approximation of conditional expectation values. The discriminators ability to do so is evaluated in this section

The discriminator of a WGAN is required to calculate a conditional expectation

$$\mathbf{E}[\varphi(X)|Y_{1:k}]$$

while simultaneously being able to optimise over the test function $\varphi : \mathbb{R}^d \to \mathbb{R}$. The method in this thesis relies on first approximating φ by a single layered perception network φ^γ. Combining this another network Ξ^ξ tasked with approximating

$$\mathbf{E}[g(\langle a, X\rangle + b)]$$

allows the discriminator to do the approximation

$$\mathbf{E}[\varphi(X)|Y_{1:k}] \approx \sum_{l=1}^{L} c_l \Xi^{\xi}(a_l, b_l, Y_{1:k}). \tag{7.1}$$

This method allows for easy optimisation over the function φ^{γ}. In theory this is a correct approach, it might however be the case that combining two approximation causes error to propagate, turing smaller errors larger. To test the accuracy of Equation (7.1) a network φ^{γ} was trained to approximate a target Lipschitz-1 function. This network can easily be trained by minimising

$$\int_{x \in D} |\varphi(x) - \varphi^{\gamma}(x)|^2 \mathrm{d}x,$$

where φ is the true function and D is the domain of approximation. A loss function which is motivated by Theorem 2.2.1. The approximation was evaluated by training φ^{γ} to mimic the functions:

$$\begin{aligned} \varphi(x) &= \langle a, x \rangle + b, \quad \|a\| = 1 \\ \varphi(x) &= \|x\|. \end{aligned}$$

Network for 2,4 and 6 dimensional spaces were trained. Thereafter these functions was combined with a network Ξ^{ξ}, trained for the spring-mass system explained in section 7.2.1. The method of evaluating this approximation is a modified version of the MAE given by

$$\mathrm{MAE} = \frac{1}{M} \sum_{m=1}^{M} \left| \varphi^{\gamma}(X_T^{(m)}) - \mu_T^{(m)} \right|,$$

where μ_T is the conditional mean $\mathbf{E}[\varphi^{\gamma}(X_T)|Y_{1:N}]$ calculated by a filter. The discriminators accuracy is benchmarked against the accuracy of the KF and particle filters with $N = 10^2, 10^3, 10^4$ particles. The MAE for the filter is presented in Figure 7.4

The results in figure indicate that for the linear function the MAE of the approximation (7.1) is close to that of the reference methods. This is to be expected since the expectation is a linear function. For the non-linear function however it is evident that the MAE of the approximation has greater MAE than the comparative methods. All approximations carry some error. It seems however that the error of the approximation is greater than expected. When optimising the WGAN this will probably corrupt the training since the supremum over φ has to include non-linear function for the method to be accurate. The error does not seem to grow larger in the approximation with higher dimension.

7.4 Conclusion

As a conclusion it can be stated that the method if implementing the WGAN tested in this thesis is viable to some extend. For problems of dimension 1 and 2 it succeed

in outperforming PF with 10^2 particles but not PF with more particles. The interpretation of these results is that the method validity to it and can be suspected to be improved upon. The premise of the method did however run into some errors as presented in the previous section. If improvements were to be done to Approximation 7.1 the method might prove more viable.

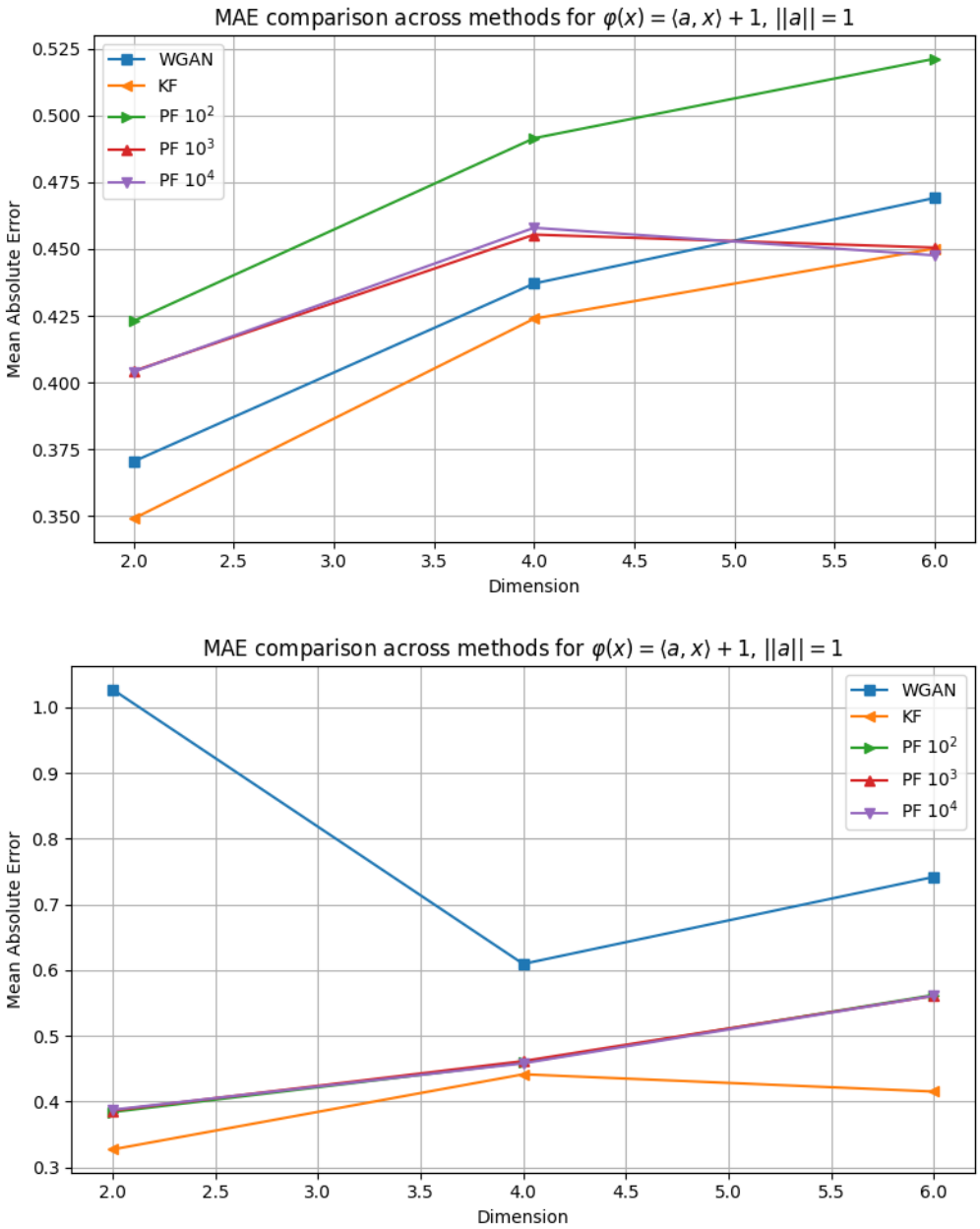


Figure 7.4: The MAE between the approximation (7.1) tested for a linear function in the upper and a non-linear function in the lower figure.

Bibliography

[1] Kurt Hornik. "Multilayer feedforward networks are universal approximators". In: *Neural Networks*. Vol. 2. 1989.

[2] Kurt Hornik. "Approximation capabilities of multilayer Feedforward Networks". In: *Neural Networks*. Vol. 4. 1991.

[3] Kloeden Peter and Platen Eckhard. *Numerical Solution of Stochastic Differential Equations*. 3rd. Springer, 1999.

[4] Cedric Villani. *Optimal transport: Old and new*. Springer, 2009.

[5] Fima C. Klebaner. *Introduction to Stochastic Calculus with Applications*. 3rd. Imperial College Press, 2012.

[6] Ian J. Goodfellow et al. "Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems*. Vol. 27. 2014.

[7] Ishaan Gulrajani et al. "Improved Training of Wasserstein GANs". In: *Advances in Neural Information Processing Systems*. Vol. 27. 2014.

[8] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. 2017.

[9] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2017). arXiv: 1412.6980.

[10] Martin Giles. *The GANfather: The man who's given machines the gift of imagination*. Accessed: 2026-01-04. Feb. 2018. URL: https://www.technologyreview.com/2018/02/12/146358/the-ganfather-the-man-whos-given-machines-the-gift-of-imagination/.

[11] Simon Godsill. "Particle Filtering: the First 25 Years and beyond". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019.

[12] Hamed Masnadi-Shirazi, Alireza Masnadi-Shirazi, and Mohammad-Amir Dastgheib. "A Step by Step Mathematical Derivation and Tutorial on Kalman Filters". In: (2019). arXiv: 1910.03558.

[13] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. 3rd. Oxford University Press, 2020.

[14] Bernhard Mehling. *Machine learning with neural networks : an introduction for scientists and engineers*. Cambridge University Press, 2022.

[15] Simo Särkkä and Lennart Svensson. *Bayesian filtering and smoothing*. Cambridge University Press, 2023.

[16] Tanujit Chakraborty et al. “Ten years of generative adversarial nets (GANs): a survey of the state-of-the-art”. In: *Machine Learning: Science and Technology*. 2024.

[17] Kenneth Shum. *Measure-theoretic probability*. Birkhäuser Cham, 2024.