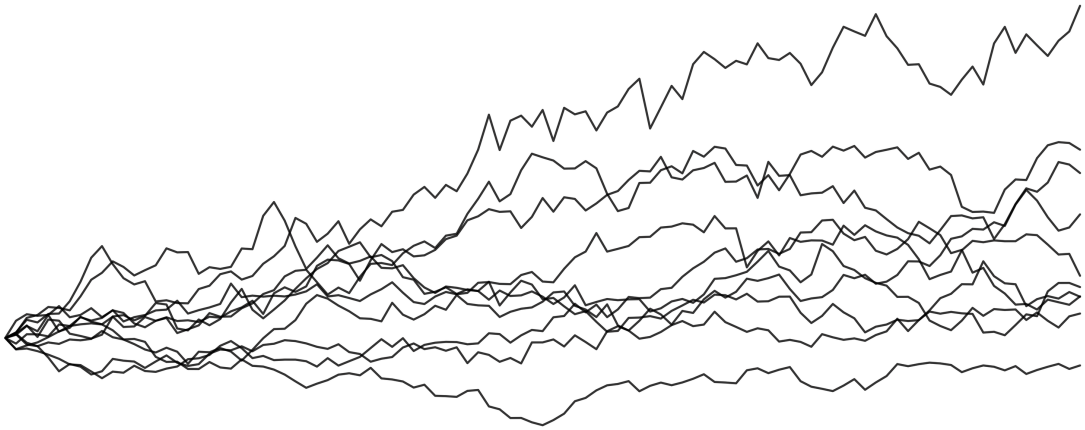


Stochastic modeling using machine learning and stochastic differential equations

From the geometric Brownian motion to stock prices

Master's thesis in Engineering Mathematics and Computational Sciences

Olof Johansson

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Stochastic modeling using machine learning and stochastic differential equations

From the geometric Brownian motion to stock prices

OLOF JOHANSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2022

Stochastic modeling using machine learning and stochastic differential equations
From the geometric Brownian motion to stock prices
OLOF JOHANSSON

Contact person Ortex Technologies Ltd.: Peter Hillerberg
Supervisor Chalmers: Mike Pereira, Mathematical Sciences
Examiner: David Cohen, Mathematical Sciences

Master's Thesis 2022
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of ten realizations of a geometric Brownian motion generated using the Euler-Maruyama method.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Stochastic modeling using machine learning and stochastic differential equations
From the geometric Brownian motion to stock prices
OLOF JOHANSSON
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Due to the presence of (unpredictable) fluctuations in the financial market, stochastic models have long been used in various financial applications. In particular, a common application is the forecasting of a given financial time series, for example stock prices. Stock prices are often assumed to follow a Geometric Brownian Motion (GBM), a specific type of stochastic differential equation. Recent studies have demonstrated promising results of using neural networks to parameterize SDEs (referred to as a neural SDE framework). Further studies have demonstrated how machine learning, specifically Recurrent Neural Networks (RNNS), can be used for predicting the future values of time-dependent data. The aim of this thesis was to investigate the possibility of combining a RNN with a neural SDE framework to forecast stock prices. In particular, three different RNNs were used, namely a Long Short-Term Memory (LSTM) model, an Echo State Network (ESN) and a Long-Short Echo State Network (LS-ESN). The results of this thesis showed that the three models considered in this thesis achieved more accurate predictions of stock prices when compared to both a traditional LSTM model and a GBM model. This was showed for both a single stock and also for 100 different stocks, where the latter also was tested for different numbers of predicted time steps ahead.

Keywords: Stochastic differential equations, machine learning, stochastic modeling, echo state networks, recurrent neural networks, financial time series

Acknowledgements

First of all, I would like to thank Peter Hillberg and Ortex Technologies Ltd. for the interest in this project and for providing the data used in this thesis. Secondly, I would like to thank my supervisor Mike Pereira for the huge engagement and support when working on this thesis. Lastly, I would to thank my family and friends for the support in rough times.

Olof Johansson, Gothenburg, June 2022

Contents

List of Figures

List of Tables

1
Introduction

This chapter provides a brief background of this thesis, followed by the scope and the aim. Thereafter, the related studies, as well as the limitations and the ethical aspects, will be presented.

1.1 Background

The term stochastic comes from the Greek word *stokhastikos*, meaning "random" or "chance". A stochastic model further refers to a model that aims to predict a set of possible outcomes from some probability distribution [1]. However, in many natural phenomena and real-world cases, events are not *in essence* stochastic nor deterministic. It is rather a choice of the observer to model a phenomenon as either stochastic or deterministic. In some cases, this choice is clear, but sometimes it is highly unclear [1]. For example, in population models, as described in [1], variations in large populations are often modeled as deterministic, despite it being agreed that many random events contribute to these variations.

Stochastic models have long been incorporated in a variety of financial applications, such as risk management, decision making in uncertain markets, asset pricing and portfolio management [2]. One reason for their extensive use is the great amount of (unpredictable) fluctuations that occur in financial markets (and particularly in stock prices). As these fluctuations are caused by factors that are usually unknown, they are often treated as stochastic processes [2]. The development of stochastic financial models has resulted in various different applications. In particular, option price models (such as the Black-Scholes model), interest rate models (such as the Hull-White and Cox-Ingersoll-Ross models) and multiple portfolio management models, all of which are still being used within financial analysis [2].

When modeled as a stochastic process, the price of a stock is often assumed to follow a Geometric Brownian Motion (GBM). The parameters of a GBM are then estimated using historical values of the stock price [2]. However, stock prices can change a lot over small time intervals, but the parameters in a GBM are constant. Thus, this does not provide a sufficient representation of the true movement of stock prices. On the other hand, the GBM has shown to provide a relatively accurate description of smaller, less complex systems [2]. Therefore, a more flexible approach with the ability to account for the relative changes in the stock prices over short time periods would be desired. The recent developments in machine learning have

demonstrated that neural networks are efficient and flexible for making predictions for various applications [3]. It could therefore be interesting to investigate how such methods could be used for stock price modeling. In particular, investigating the possibility of using a combined framework of machine learning and stochastic models could be of interest. Moreover, as stock prices are time-dependent, historical changes of the stock price could be valuable when considering how it could change in the future. Therefore, it would also be desired for a model to be able to account for historical values when making future predictions of the price of a stock. Since Recurrent Neural Networks (RNNs) are designed to model time-dependent data and learn from historical values, they are interesting to consider for this task [4]. One such model is the Long Short-Term Memory (LSTM) model.

The LSTM model has shown to outperform previously developed RNNs in various prediction tasks involving time-dependent data. It has further been widely used in financial applications [4]. A disadvantage of the LSTM model is however that it is computationally expensive to train and can suffer from convergence issues in the optimization [5]. To overcome this issue, a different kind of machine learning models called Echo State Networks (ESNs) could be used. ESNs are faster to train and have shown to be better at learning from chaotic and nonlinear data, a prominent attribute for modeling stock prices [5]. A recent development of ESNs, presented in [6], is the Long-Short Term Echo State Network (LS-ESN). The LS-ESN uses an ensemble of ESNs to improve forecasting of time-dependent data.

As previously mentioned, the price of a stock can be assumed to follow a GBM, a stochastic process that satisfies a specific type of Stochastic Differential Equation (SDE). A recent study has shown promising results of using neural networks to parametrize a SDE [7]. This is referred to as a neural SDE framework. Using such a framework has further been demonstrated to yield more accurate results (than deterministic methods) with a lower vulnerability to significant amounts of random noise in the data [7]. Since stochastic processes by definition comprise randomness, it could be interesting to use a neural SDE framework to parameterize stock prices. As previously mentioned, it could further be interesting to utilize RNNs to account for the historical data when aiming towards forecasting future stock prices. The aim of this thesis is therefore to investigate the possibility of combining the use of RNNs (more specifically a LSTM, an ESN and a LS-ESN) with a neural SDE framework in order to forecast stock prices.

1.2 Scope

As previously mentioned in Section 1.1, the main objective of this thesis is to investigate whether a combined framework of recurrent neural networks and neural SDEs can be used to make probabilistic forecasts of stock prices. Specifically, a LSTM, ESN and LS-ESN combined with the neural SDE framework (denoted LSTM-SDE, ESN-SDE and LS-ESN-SDE respectively) are used. In this thesis, these models are used to obtain a probability distribution of what the possible future values the price of a stock can be. The project was carried out during the course of six months

(January 2022 - June 2022) and was in partial collaboration with Ortex Technologies Ltd[1]. This is a company that utilizes machine learning for financial modeling and is therefore interested in the development of methods to be used for stock price modeling. All real-world data that is being used in this project, was provided by the company.

1.2.1 Aim

To fulfill the above presented scope, the aim of this thesis is to provide answers to the following question:

1. Can the LSTM-SDE, ESN-SDE and LS-ESN-SDE be used to forecast
   (a) a geometric Brownian motion,
   (b) real-world stock prices?

1.3 Related studies

A previous study in [8] used neural networks to solve Ordinary Differential Equations (ODEs), referred to as a neural ODE approach. Similarly, a study in [7] used Stochastic Differential Equations (SDEs) parameterized by neural networks, referred to as neural SDEs, for the purpose of image classification. Despite being limited to only considering images as data, the neural SDE framework in [7] was developed to make existing models more robust against perturbations in the input data. Since stochastic data by definition is subject to perturbations, this inspires to investigate the use of such a framework in predictions of stochastic data.

As also mentioned in Section 1.1, a study in [9] used a similar approach for forecasting as used in this thesis. They used a LSTM model along with a SDE block to produce a predictive mean and predictive variance of the target values. These two were produced by using two additional neural networks within the SDE block. However, with their approach, the predictive variance used in the optimization is not actually a variance, but rather some parameter used as the variance. It is therefore less intuitive what these values actually represent. Therefore, by instead incorporating the uncertainty of the final predictions within the model, by using the sample mean and sample variance from several produced predictions, the predictive parameters are both more interpretable and also represent the actual mean and variance from the predictions. Moreover, the work in [9] performed experiments on stock price data, but did so only for one time step ahead and did only consider a LSTM model as recurrent neural network. Since this model suffers from some issues as previously mentioned in Section 1.1, it is interesting to investigate other models for this task.

Lastly, as also mentioned in Section 1.1, a related study in [6] used a LS-ESN

[1]Further referred to as "the company".

model for the task of forecasting. It was shown that the LS-ESN outperformed the benchmark models on all of the different data sets used. It was however not tested on stock price data and with the obtained results, it is therefore interesting to investigate the usage in the framework of this thesis.

The work presented in this thesis was inspired by the above studies, among others, and in particular the findings of the work in [9], that combines previous studies of the LSTM model with the neural SDE framework specifically designed to make forecasting tasks more robust and accurate.

1.4 Limitations

As the aim of this thesis focuses on financial data, and in particular stock prices, applications in other domains are not considered. Furthermore, the GBM is the only stochastic process which is considered in this thesis. The reason for that is, as mentioned in Section 1.1, due to its use for stock price modeling. Moreover, since the stock price data that is used in this thesis is solely what is provided by the company, the experimental results as well as the conclusions are limited for this data specifically. Furthermore, as the results are only considered with respect to the selected benchmark models, the evaluation of the performance is limited in comparison to these models only.

1.5 Ethical aspects and considerations

When working with financial data, there are some ethical aspects to take into consideration. First of all, algorithms in financial analysis have the risk to offset human biases and negatively impact users of such an analysis. Moreover, due to the great uncertainty in financial markets, the results in this thesis are not guaranteed to hold for other stock data or for the same data in a different time period. It is therefore important to note that the results of this thesis are not a recommendation for possible investments.

2
Theory

This chapter presents the necessary theoretical concepts involved in the methods used in this thesis. The outline is to first define the fundamental concepts of stochastic processes and stochastic differential equations and then explain the construction of the models that are considered in this thesis.

2.1 Stochastic Processes

Stochastic processes appear in many real-world phenomena, such as the motion of particles in physics, modeling of infectious diseases and genetics in medicine and asset price modeling in finance [10]. It is also used to describe the noise in experimental measurements influenced by external factors [11]. The following section first defines some fundamental concepts of stochastic processes, and then presents how these processes are analytically and numerically approached.

2.1.1 Introduction to Stochastic Processes

Let $(\Omega, \mathcal{F}, \mathbb{P})$ denote a probability space where Ω is a sample space of possible outcomes, $\mathcal{F}$ is a σ-algebra and $\mathbb{P}$ is a probability measure. A stochastic process is a collection of random variables defined on such a probability space and is indexed by some set $\mathcal{T}$ [12]. Moreover, in this thesis, the set $\mathcal{T}$ is a set of (continuously or discretely) increasing values of some time interval on $\mathbb{R}$. A stochastic process indexed by $\mathcal{T}$ is further denoted as $\mathbf{X} = \{\mathbf{X}_t : t \in \mathcal{T}\}$. The random variables in a (real-valued) stochastic process are (real-valued) variables whose values are random samples from a probability distribution. The values of the random variables in $\mathbf{X}$ are therefore (for real-valued stochastic process) defined on $\mathbb{R}^N$. The stochastic process is then said to be a N-dimensional stochastic process such that $\mathbf{X}_t \in \mathbb{R}^N$ [12].

An important type of stochastic process is the Wiener process which is used to describe various different phenomena within physics, finance and biology. Let $W = \{W_t : t \in [0, T]\}$ be a stochastic process defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ on the continuous time interval $[0, T]$ for some $T > 0$. Then W is a Wiener process if it satisfies

1. $W_0 = 0$ (with probability one),

2. for any $n \geq 2$ and $0 < t_1 < \cdots < t_n \leq T$, the (non-overlapping) increments

$W_{t_2} - W_{t_1}, \cdots, W_{t_n} - W_{t_{n-1}}$ are independent,

3. for any $0 \leq s < t \leq T$, the increment $W_t - W_s$ is normally distributed with mean 0 and variance $t - s$,

4. $t \mapsto W_t$ is continuous (with probability one),

then W_t is a Wiener process [12]. As the Wiener process originated from being a model to describe a Brownian motion, it is often referred to as a Brownian motion process, or even just a Brownian motion [12]. In the remaining of this thesis, a Brownian motion will be referred to as a Wiener process as presented above. Moreover, in this thesis, the notation $\mathbf{X}_t$ will denote the value of a stochastic process at time t and $\mathbf{X}(t)$ will denote the value of a time-dependent function (not necessary stochastic) at time t.

2.1.2 Stochastic Differential Equations

Stochastic Differential Equations (SDEs) are constructed using the theory of stochastic calculus and, in this thesis, Itô calculus. Before presenting the type of SDEs considered in this thesis, the notion of an Itô integral needs to be described [13]. To do this, first consider the function $\mathbf{h} : \mathbb{R}^N \times [0, T] \longrightarrow \mathbb{R}$ for some $T > 0$. Then by recalling that, for some real-valued function of time $\mathbf{y}$ such that $\mathbf{y}(t) \in \mathbb{R}^N$ for $t \in [0, T]$, the Riemann integral $\mathbf{R}$, for $\mathbf{h}$ over $t \in [0, T]$ given as

$$\mathbf{R}(t) = \int_0^t \mathbf{h}(\mathbf{x}(s), s)\mathrm{d}s, \quad t \in [0, T],$$

can be written as the limit of a Riemann sum as

$$\mathbf{R}(T) = \int_0^T \mathbf{h}(\mathbf{x}(t), t)\mathrm{d}t = \lim_{n \longrightarrow \infty} \sum_{k=0}^{n} \mathbf{h}(\mathbf{x}(t_k^*), t_k^*)(t_{k+1} - t_k), \tag{2.1}$$

where $0 = t_0 < t_1 \cdots < t_n = T$ is a partition of the interval $[0, T]$ and $t_k^* \in [t_k, t_{k+1}]$ is an arbitrary element in the subinterval $[t_k, t_{k+1}]$ of the partition for all $k \in \{0, 1, \ldots, N-1\}$. The value of the Riemann integral is then defined by the limit of the Riemann sum if the upper and lower sums converge to the same value [13].

Consider instead the N-dimensional stochastic processes $\mathbf{Y} = \{\mathbf{Y}_t : t \in [0, T]\}$ and $\mathbf{X} = \{\mathbf{X}_t : t \in [0, T]\}$ and let $h : \mathbb{R}^N \longrightarrow \mathbb{R}^N$. Then, let $\mathbf{Y}_t$ be the integral such that for all $t \in [0, T]$,

$$\mathbf{Y}_t = \int_0^t \mathbf{h}(\mathbf{X}_s, s)\mathrm{d}W_s,$$

where $W = \{W_t : t \in [0, T]\}$ is a Brownian motion. For this case, the Riemann sum cannot be used in Equation (2.1) as a definition of the integral $\mathbf{Y}$. This is because the Brownian motion $t \mapsto W_t$ is non-differentiable and because of the values $W_{t_k^*}$ are unbounded. Therefore, as described in [13], by instead fixing $t_k^* = t_k$ for each $t_k \in \{t_0, t_1, \cdots, t_n\}$, $\mathbf{Y}$ can be defined as the unique limit in the L^2 sense, of

$$\mathbf{Y}_T = \int_0^T \mathbf{h}(\mathbf{X}_t, t)\mathrm{d}W_t = \lim_{n \longrightarrow \infty} \sum_{k=1}^{n-1} \mathbf{h}(\mathbf{X}_{t_k}, t_k)(W_{t_{k+1}} - W_{t_k}). \tag{2.2}$$

The integral equation of $\mathbf{Y}$ in Equation (2.2) is referred to as the Itô integral. As described in [13], this Itô integral is also a stochastic process defined on $(\Omega, \mathcal{F}, \mathbb{P})$. The Itô integral in Equation (2.2) will be used in defining of the type of SDEs used in this thesis. First, let $\mathbf{f} : \mathbb{R}^N \times [0,T] \longrightarrow \mathbb{R}^N$ and $\mathbf{g} : \mathbb{R}^N \times [0,T] \longrightarrow \mathbb{R}^{N\times N}$. Next, let $\mathbf{X} = \{\mathbf{X}_t : t \in [0,T]\}$ be a N-dimensional stochastic process and $\mathbf{W} = \{\mathbf{W}_t : t \in [0,T]\}$ be a N-dimensional Brownian motion. Then the integral equation

$$\mathbf{X}_t = \mathbf{X}_0 + \int_0^t \mathbf{f}(\mathbf{X}_s, s)\mathrm{d}s + \int_0^t \mathbf{g}(\mathbf{X}_s, s)\mathrm{d}\mathbf{W}_s, \quad t \in [0,T] \tag{2.3}$$

defines the Itô integral equation of $\mathbf{X}_t$ for $t \in [0,T]$. The first integral in Equation (2.3) is an ordinary Lebesgue integral and the last term is a sum of Itô integrals. Moreover, a stochastic process described by an Itô integral equation as in Equation (2.3) is called an Itô process [13]. The differential notation of Equation (2.3) is written as

$$\mathrm{d}\mathbf{X}_t = \mathbf{f}(\mathbf{X}_t, t)\mathrm{d}t + \mathbf{g}(\mathbf{X}_t, t)\mathrm{d}\mathbf{W}_t, \quad t \in [0,T]. \tag{2.4}$$

The functions $\mathbf{f}$ and $\mathbf{g}$ are referred to as the drift and diffusion functions of the Itô process $\mathbf{X}_t$. When the diffusion function g only contains constant values, it is referred to as a diffusion matrix. Moreover, [13]. The Itô integral in Equation (2.3) is referred to as the Itô SDE of $\mathbf{X}$ while the notation in Equation (2.4) are usually applied when referring to an (Itô) SDE.

The problem related to Equation (2.4) is to find its solution $\mathbf{X}$ given an initial condition $\mathbf{X}_0$. However, in order for a unique solution to exist, some restrictions are required for the drift and diffusion functions $\mathbf{f}$ and $\mathbf{g}$. These are stated in the following theorem, as according to [14].

Theorem 2.1. *Let* $\mathbf{f} : \mathbb{R}^N \times [0,T] \longrightarrow \mathbb{R}^N$ *and* $\mathbf{g} : \mathbb{R}^N \times [0,T] \longrightarrow \mathbb{R}^{N\times N}$*, then if there exists two constants* $L, C \geq 0$ *such that for all* $t \in [0,T]$ *and for all* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$*,*

$$||\mathbf{f}(\mathbf{x},t) - \mathbf{f}(\mathbf{y},t)|| \leq L||\mathbf{x}-\mathbf{y}||, \quad \textit{and} \quad ||\mathbf{g}^{(i)}(\mathbf{x},t) - \mathbf{g}^{(i)}(\mathbf{y},t)|| \leq C||\mathbf{x}-\mathbf{y}||, \tag{2.5}$$

for all $i \in \{1,\ldots,N\}$ *and there exists four constants* $a, b, c, d \geq 0$ *such that*

$$||\mathbf{f}(\mathbf{x},t)|| \leq a + b||\mathbf{x}||, \quad \textit{and} \quad ||\mathbf{g}^{(i)}(\mathbf{x},t)|| \leq c + d||\mathbf{x}||, \tag{2.6}$$

for all $i \in \{1,\ldots,N\}$*. Then, for every* $\mathbf{X}_0 \in \mathbb{R}^N$*, there exists a unique solution* $\mathbf{X} \in \mathbb{R}^N$ *which solves the stochastic differential equation in Equation* (2.4)*.*

For the proof of Theorem 2.1, see [14]. The Equations (2.5) and (2.6) are the Lipschitz continuity condition and the linear growth condition respectively for the functions $\mathbf{f}$ and $\mathbf{g}$.

2.1.3 Numerical Approximation of Stochastic Differential Equations

In order to approximately solve the SDE in Equation (2.4) and find and approximation to its solution $\mathbf{X}_t$ as according to Equation (2.4), numerical methods needs

to be applied since analytical solutions are often unavailable. To define such numerical methods for an approximate solution to Equation (2.4), let first $\mathbf{X}$, $\mathbf{f}$ and $\mathbf{g}$ be as in Equation (2.4) for $t \in [0, T]$ and let $\mathbf{X}_0 = \mathbf{x}_0$ be given. Then, similarly to deterministic ordinary differential equations, consider an integer $N > 0$ and let $\Delta t = T/N$ be the stepsize of the discretization of the time interval $[0, T]$. Then, an approximation $\tilde{\mathbf{X}}$ of $\mathbf{X}$ is computed for each discretized time $t_i = i\Delta t$ as

$$\tilde{\mathbf{X}}_{t_i} = \tilde{\mathbf{X}}_{t_{i-1}} + \mathbf{f}(\tilde{\mathbf{X}}_{t_{i-1}}, t_{i-1})\Delta t + \mathbf{g}(\tilde{\mathbf{X}}_{t_{i-1}}, t_{i-1})\Delta \mathbf{W}_{t_{i-1}}, \tag{2.7}$$

where $\Delta \mathbf{W}_{t_{i-1}} = \mathbf{W}_{t_i} - \mathbf{W}_{t_{i-1}}$ is a N-dimensional vector of independent and normal distributed variables with a mean zero and variance Δt. Hence, $\Delta \mathbf{W}_{t_{i-1}}$ can be generated as $\boldsymbol{\xi}\sqrt{\Delta t}$, where $\boldsymbol{\xi}$ is a N-dimensional vector of independent and normal distributed variables with mean zero and unit variance. Therefore, Equation (2.7) is evaluated as

$$\tilde{\mathbf{X}}_{t_i} = \tilde{\mathbf{X}}_{t_{i-1}} + \mathbf{f}(\tilde{\mathbf{X}}_{t_{i-1}}, t_{i-1})\Delta t + \mathbf{g}(\tilde{\mathbf{X}}_{t_{i-1}}, t_{i-1})\boldsymbol{\xi}\sqrt{\Delta t}, \tag{2.8}$$

for $t_i = i\Delta t$, $i \in \{1, \cdots, N\}$. The numerical method for approximating $\mathbf{X}$ by $\tilde{\mathbf{X}}$ as in Equation (2.8) is called the Euler-Maruyama (EM) method [15, 16].

A central type of SDE used in financial modeling for which the EM method can be applied is the geometric Brownian motion (GBM). The GBM is defined for the one-dimensional case as

$$\mathrm{d}X_t = \mu X_t \mathrm{d}t + \sigma X_t \mathrm{d}W_t, \tag{2.9}$$

where μ and σ is the drift and diffusion constants. The GBM is used to model the movement of stock prices and is used in the Black-Scholes option pricing formula [17]. The GBM is a continuous stochastic process, but can be used to model data that are discretely observed, such as stock prices. By using the EM method, sample paths of the GBM can be simulated in discrete steps. A desired property of the GBM is that the ratio of its values between consecutive time steps are log-normal distributed. Hence, $\log X_t/X_{t-\Delta t}$ are normally distributed with mean $(\mu - \sigma^2/2)\Delta t$ and variance $\sigma^2 \Delta t$, where Δt is the time step size. Moreover, the logarithmic ratios are also independent and identically distributed (i.i.d) as according to [18].

There exist several ways to estimate the parameters of a GBM using historical values of the stock price. One method utilizes the previously mentioned result to estimate the parameters, as according to [18], as

$$\begin{aligned} \hat{\mu} &= \sum_{t=1}^{N-1} \frac{X_{t+1} - X_t}{X_t} \\ \hat{\sigma} &= \sqrt{\frac{1}{N-1} \sum_{t=1}^{N-1} \left(\frac{X_t - X_{t-1}}{X_t} - \hat{\mu} \right)^2}, \end{aligned} \tag{2.10}$$

where X_t denotes the stock price observed at some time t for all t in the interval $\{0, \ldots, N-1\}$. When the parameters of a GBM have been estimated, the EM method can be applied to simulate sample paths over the future values of X_t. An

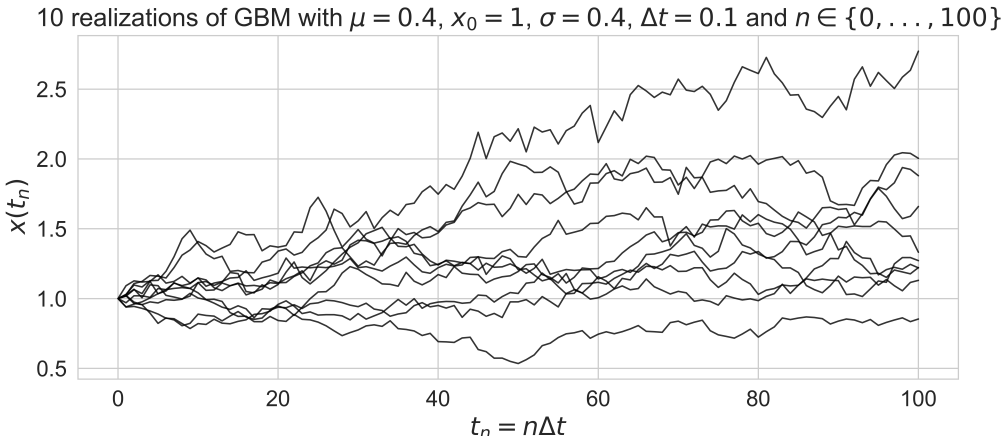


Figure 2.1: 10 *realizations of a GBM with* $\mu = 0.4$, $x_0 = 1$, $\sigma = 0.4$, $\Delta t = 0.1$ *for* $n \in \{0, \ldots, 100\}$ *generated using the EM method.*

illustration of 10 realizations of the GBM simulated using the EM method is shown in Figure 2.1 with $\mu = 0.4$, $\sigma = 0.4$, $x_0 = 1$, $\Delta t = 0.1$ for the time steps $n \in \{0, \ldots, 100\}$ where $N = 101$.
The pseudocode for the implementation of the EM method for approximating a stochastic process, given as a GBM, is shown below.

Algorithm 1: Euler-Maruyama method for a geometric Brownian motion

Data: μ, σ, x_0, t_0, T, N
Result: $\tilde{X}(t) \approx X(t)$, $t \in [t_0, T]$
$\Delta t \leftarrow (T - t_0)/N$;
$\tilde{X} \leftarrow$ zero array of size N;
$\tilde{X}(0) \leftarrow x_0$;
for $i = 1$ *to* N **do**
  $\xi \leftarrow$ randomly generated number from $\mathcal{N}(0, 1)$;
  $X \leftarrow \tilde{X}(i - 1)$;
  $\tilde{X}(i) \leftarrow X + \mu X \Delta t + \sigma X \sqrt{\Delta t} \xi$;
end

2.2 Stochastic Modeling

Consider an input data set, $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ where $\mathbf{x}_i \in \mathbb{R}^{D_{obs}}$ for all $i = 1, \ldots, k$ and k is the total number of samples for some dimension $D_{obs} > 0$ of the observed values. Let further $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k)$, where $\mathbf{y}_i \in \mathbb{R}^{D_{targ}}$ for all $i = 1, \ldots, k$, denote the corresponding target data set for some dimension $D_{targ} > 0$ of the target values. By assuming that values of the input and target data are driven by some underlying process such that their values are samples from a probability distribution. Then the connection between the input and target samples, assumed that such a connection exists, can be described by a conditional probability distribution, $p_{true}(\mathbf{y}_i|\mathbf{x}_i)$. The objective of a stochastic model (when the underlying process is

unknown) is to find an approximative probability distribution, $p_{appr}(\mathbf{y}_i|\mathbf{x}_i, \theta)$, with parameters θ for all $i = 1, \ldots, k$, which best approximates the true distribution, $p_{true}(\mathbf{y}_i|\mathbf{x}_i)$ [19, 20].

When using a forecasting model, prior information is used to predict future outcomes. Consider for example the time-development of a stock price. The future price of the stock can be estimated by first finding an approximate probability distribution, where later observations are conditioned on the prior observations. Assume that the price of the stock has been observed for T number of equitemporal steps. This time-series could further be divided into T_{sub} equal length subintervals, where $2 \leq T_{sub} < T$. One method for approximating the probability distribution, is to condition each interval on the prior. This distribution could then be used to forecast future, not yet observed, prices of the stock. This could be achieved by simply using the latest observed data as input and estimating the output using the approximate conditional probability distribution. However, to estimate p_{appr} can be difficult. A special case is when the drift and diffusion functions are known. Then, the conditional probability distribution can be approximated for any type of SDE under the assumption that the distribution is Gaussian. This case will be more deeply presented in the the following section.

2.2.1 Cubature Integration Sigma Point Approximation

Suppose that some N-dimensional discretely observed data $\mathbf{x} = \{\mathbf{x}(t) : t \in \{0, \cdots, T\}\}$ for some $T > 0$ corresponds to the N-dimensional Itô process $\mathbf{X} = \{\mathbf{X}_t : t \in [0, T]\}$ as

$$\mathrm{d}\mathbf{X}_t = \mathbf{f}(\mathbf{X}_t, t)\mathrm{d}t + \mathbf{g}(\mathbf{X}_t, t)\mathrm{d}\mathbf{W}_t, \quad t \in [0, T] \tag{2.11}$$

where $\mathbf{f} : \mathbb{R}^N \times [0, T] \longrightarrow \mathbb{R}^N$ and $\mathbf{g} : \mathbb{R}^N \times [0, T] \longrightarrow \mathbb{R}^{N \times N}$ are the drift and diffusion functions respectively and $\mathbf{W}$ is an N-dimensional Brownian motion. The values of the process $\mathbf{X}$ at the time steps $t \in \{0, \cdots, T\}$ is then equal to the values of the observations at that time t. Then the conditional distribution, $p(\mathbf{x}(t)|\mathbf{x}(s))$, of $\mathbf{x}(t)$ given $\mathbf{X}_s = \mathbf{x}(s)$, evaluated at $\mathbf{X}_t = \mathbf{x}(t)$ for some $t \geq s$, is given as the solution to the Fokker-Planck-Kolmogorov (FPK) equation [21]

$$\begin{aligned} \frac{\partial p(\mathbf{x}(t)|\mathbf{x}(s))}{\partial t} = &- \sum_{i=1}^{N} \frac{\partial}{\partial x_i} \left[\mathbf{f}_i(\mathbf{x}(t), t) p(\mathbf{x}(t)|\mathbf{x}(s))\right] \\ &+ \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\partial^2}{\partial x_i \partial x_j} \left\{ [\mathbf{g}(\mathbf{x}(t), t)\mathbf{g}(\mathbf{x}(t), t)^{\mathrm{T}}]_{i,j} p(\mathbf{x}(t)|\mathbf{x}(s)) \right\}, \end{aligned} \tag{2.12}$$

with some initial value $p(\mathbf{x}(s)|\mathbf{x}(s)) = p_s$ and where superscript T denotes the transpose. Then, by considering the expectaction, $\langle \mathbf{X}_t \rangle_p$ of $\mathbf{X}_t$, where $\langle \cdot \rangle_p$ is expectation with respect to the conditional distribution p, the conditional mean and conditional covariance matrices, $\mathbf{m}(t)$, $\mathbf{S}(t)$ can be denoted by

$$\begin{aligned} \mathbf{m}(t) &= \langle \mathbf{X}_t \rangle_p \\ \mathbf{S}(t) &= \left\langle (\mathbf{X}_t - \mathbf{m}(t))(\mathbf{X}_t - \mathbf{m}(t))^{\mathrm{T}} \right\rangle_p, \quad t \in [0, T], \end{aligned}$$

and are called the first and second order moments of $\mathbf{X}_t$ with respect to p. As a consequence of Itô's formula, $\mathbf{m}(t)$ and $\mathbf{S}(t)$ are given as the solution to the ordinary differential equations of

$$\frac{\mathrm{d}\mathbf{m}(t)}{\mathrm{d}t} = \langle \mathbf{f}(\mathbf{X}_t, t) \rangle_p \,, \quad t \in [0, T], \tag{2.13}$$

and

$$\begin{aligned} \frac{\mathrm{d}\mathbf{S}(t)}{\mathrm{d}t} &= \left\langle \mathbf{f}(\mathbf{X}_t, t)(\mathbf{X}_t - \mathbf{m}(t))^{\mathrm{T}} \right\rangle_p \\ &+ \left\langle (\mathbf{X}_t - \mathbf{m}(t))\mathbf{f}(\mathbf{X}_t, t)^{\mathrm{T}} \right\rangle_p \\ &+ \left\langle \mathbf{g}(\mathbf{X}_t, t)\mathbf{g}(\mathbf{X}_t, t)^{\mathrm{T}} \right\rangle_p \,, \quad t \in [0, T]. \end{aligned} \tag{2.14}$$

Since these are given as expectations with respect to the conditional distribution, which is given by the solution of the FPK in Equation (2.12), the differential equations in (2.13) and (2.14) cannot be explicitly solved in most cases. In the case when $\mathbf{X}$ is Gaussian distributed they can completely characterize the solution [21]. This is however most often not the case, but despite this, with the use of the first and second order moments, the conditional distribution can be approximated for nonlinear SDEs. In fact, by approximating the conditional distribution of $\mathbf{X}$ at $\mathbf{X}_t = \mathbf{x}(t)$ given $\mathbf{X}_s = \mathbf{x}(s)$ as Gaussian as

$$p(\mathbf{x}(t)|\mathbf{x}(s)) \simeq \mathcal{N}(\mathbf{x}(t)|\mathbf{m}(t), \mathbf{S}(t)),$$

where $\mathcal{N}$ denotes a Gaussian (or normal) distribution and $\simeq$ denotes that p is assumed to be equal the Gaussian distribution $\mathcal{N}$. The expectations of the first and second order moments are therefore taken with respect to $\langle \cdot \rangle_{\mathcal{N}}$ instead of $\langle \cdot \rangle_p$. The moments in Equation (2.13) and (2.14) can then be numerically solved [22]. This type of approximation is referred to as Gaussian-process approximation or Gaussian-density approximation [23, 24, 22].

There are several numerical methods to solve the ODEs in Equation (2.13) and (2.14). For example by using linearization approximation where the drift is linearized around the conditional mean after which the diffusion is approximated as

$$\begin{aligned} \mathbf{f}(\mathbf{x}(t), t) &\approx \mathbf{f}(\mathbf{m}(t), t) + \mathbf{J}_{\mathbf{x}}(\mathbf{f})(\mathbf{m}(t), t)(\mathbf{x}(t) - \mathbf{m}(t)) \\ \mathbf{g}(\mathbf{x}(t), t) &\approx \mathbf{g}(\mathbf{m}(t), t), \end{aligned}$$

where $\mathbf{J}_{\mathbf{x}}$ is the Jacobian operator of $\mathbf{f}$ with respect to its first argument $\mathbf{x}$. These approximations can then be used to approximate the expectations in the ODEs of the moments [22]. A different way of approximating Equation (2.13) and (2.14) is to directly approximate the expectations with respect to $\mathcal{N}(\mathbf{x}(t)|\mathbf{m}(t), \mathbf{S}(t))$. This can be done using quadrature weighted sums as

$$\langle \mathbf{f}(\mathbf{X}_t, t) \rangle_{\mathcal{N}} \approx \sum_{i=1}^{2N} W^{(i)} \mathbf{f}(\mathbf{s}^{(i)}(t), t), \tag{2.15}$$

where $\mathbf{s}^{(i)}(t)$ are referred to as sigma-points and $W^{(i)}$ referred to as the corresponding weights of the sigma-points and $i \in \{1, 2, \ldots, 2N\}$. Here, N is the dimension of $\mathbf{x}$. Different methods can be used to select these sigma-points and weights. A common method [25] is to choose $\mathbf{s}^{(i)}(t)$ as

$$\mathbf{s}^{(i)}(t) = \mathbf{m}(t) + \sqrt{\mathbf{S}(t)}\boldsymbol{\xi}_i, \tag{2.16}$$

where $\sqrt{\mathbf{S}(t)} \in \mathbb{R}^{N\times N}$ such that $\mathbf{S}(t) = \sqrt{\mathbf{S}(t)}\sqrt{\mathbf{S}(t)}^{\mathrm{T}}$ and

$$\boldsymbol{\xi}_i = \begin{cases} \sqrt{N}\mathbf{e}_i, & i \in \{1, 2, \ldots, N\} \\ -\sqrt{N}\mathbf{e}_{i-N}, & i \in \{N+1, N+2, \ldots, 2N\}, \end{cases} \tag{2.17}$$

$$W^{(i)} = \frac{1}{2N}, \quad i \in \{1, 2, \ldots, 2N\}, \tag{2.18}$$

where $\mathbf{e}_i$ is the N-dimensional unit vector as

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \quad \cdots, \quad \mathbf{e}_N = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}.$$

Using this sigma point approximation of Gaussian integrals with the sigma-points and corresponding weights being chosen as in Equation (2.16), (2.17) and (2.18) is referred to as cubature integration sigma-point approximation (CISPA) [22] and is commonly used in filtering theory for approximation of Gaussian integrals. The approximation of the equations for the first and second order moments are then evaluated as

$$\frac{\mathrm{d}\mathbf{m}(t)}{\mathrm{d}t} = \sum_{i=1}^{2N} W^{(i)}\mathbf{f}(\mathbf{m}(t) + \sqrt{\mathbf{S}(t)}\boldsymbol{\xi}_i, t), \tag{2.19}$$

and

$$\begin{aligned} \frac{\mathrm{d}\mathbf{S}(t))}{\mathrm{d}t} =& \sum_{i=1}^{2N} W^{(i)}\mathbf{f}(\mathbf{m}(t) + \sqrt{\mathbf{S}(t)}\boldsymbol{\xi}_i, t)\boldsymbol{\xi}_i^{\mathrm{T}}\sqrt{\mathbf{S}(t)}^{\mathrm{T}} \\ &+ \sum_{i=1}^{2N} W^{(i)}\boldsymbol{\xi}_i\sqrt{\mathbf{S}(t)}\mathbf{f}^{\mathrm{T}}(\mathbf{m}(t) + \sqrt{\mathbf{S}(t)}\boldsymbol{\xi}_i, t) \\ &+ \sum_{i=1}^{2N} W^{(i)}\mathbf{g}(\mathbf{m}(t) + \sqrt{\mathbf{S}(t)}\boldsymbol{\xi}_i, t)\mathbf{g}^{\mathrm{T}}(\mathbf{m}(t) + \sqrt{\mathbf{S}(t)}\boldsymbol{\xi}_i, t). \end{aligned} \tag{2.20}$$

The ODEs in Equation (2.19) and (2.20) can be solved using usual numerical methods for ODE such as the Euler method. Finding the conditional mean and conditional covariance for the approximation of $p(\mathbf{x}(t)|\mathbf{x}(s))$, then the initial conditions of $\mathbf{m}$ and $\mathbf{P}$ is set to $\mathbf{m}(s) = \mathbf{x}(s)$ and $\mathbf{S}(s) = \mathbf{0}$. Then, the interval $\{s, \ldots, t\}$ is discretized the into n partitions with some discretization step $\Delta t = (t - s)/n$. The conditional mean and conditional covariance are then iteratively solved using Equation (2.19) and (2.20) at each discretized time step $n\Delta t$ [22].

The approximations used in this section aim at approximating the conditional distribution of nonlinear SDEs, but can also be used for the linear case. However, this method assume that the original SDE is actually known, which is not always the case. If some observed data is considered to follow a SDE whose drift and diffusion are unknown, assumptions must be put on the drift and diffusion in order to apply the approximations presented in this section. For financial applications, like modeling stock prices, a common assumption is that the stock price can be modeled as Geometric Brownian Motion (GBM), as mentioned in Section 2.1.2. However, estimating the parameters of the GBM, such that the process best fits the observed data, does not always represent the actual data that is being observed. Therefore, some kind of transformation of the data is often required, such that the transformed data provides more accurate modeling. One method that aims to approximate the conditional distribution of some observed data (by doing such a transformation) is the Free-Energy Approximation (FEA). This is presented in the proceeding section.

2.2.2 Free-Energy Approximation

Let $\mathbf{x}$ be some N-dimensional observed data $\mathbf{x} = \{\mathbf{x}(t) : t \in \{0, \dots, T\}\}$ for some $T > 0$ as in the previous section. Then assume that $\mathbf{x}$ corresponds to a stochastic process, $\mathbf{X}$, such that the values of $\mathbf{x}$ are sampled from probability distribution $p(\mathbf{X}_t)$ at each $t \in \{0, \dots, T\}$. Then, a connection between the values of $\mathbf{X}$ that have been observed, and the possible values of future observations of $\mathbf{X}$, can be obtained by approximating the conditional distribution of the observed values of $\mathbf{X}$. One method from [26] and [27], referred to as free-energy approximation (FEA), aims to approximate the posterior distribution of the values from some observed data. This is done by modeling the observed data by a SDE of a time-continuous process and find the posterior distribution by fitting another approximating SDE to the first one. More concretely, assume first that $\mathbf{X}$ is some noisy observed values from a hidden N-dimensional process $\mathbf{Z} = \{\mathbf{Z}_t : t \in [0, T]\}$, such that

$$\begin{aligned} &\mathrm{d}\mathbf{Z}_t = \mathbf{f}(\mathbf{Z}_t, t)\mathrm{d}t + \mathbf{\Sigma}\mathrm{d}\mathbf{W}_t, \quad t \in [0, T] \\ &\mathbf{X}_t \sim \mathcal{N}(\mathbf{Z}_t, r^2\mathbf{I}), \quad t \in [0, T] \end{aligned} \tag{2.21}$$

where $\mathbf{f} : \mathbb{R}^N \times [0, T] \longrightarrow \mathbb{R}^N$ is some drift function, $\mathbf{\Sigma} := \mathrm{diag}\{\sigma_1^2, \dots \sigma_N^2\}$ is the diagonal diffusion matrix, $\mathbf{\Sigma} \in \mathbb{R}^{N \times N}$, $\mathbf{W}_N$ is an N-dimensional Brownian motion. Moreover, r is the noise parameter, a fixed parameter that describes the variance of the noise of the observed data. Then, given the observations $\mathbf{x}(t)$ for all $t \in \{0, \dots, T\}$ of $\mathbf{X}$, the aim is to approximate the posterior distribution $p(\mathbf{Z}_t | \mathbf{x}(t), t)$ for all $t \in \{1, \dots, T\}$. As proposed by [26] and [27], this is done by considering the SDE

$$\begin{aligned} &\mathrm{d}\mathbf{Z}_t = \mathbf{g}(\mathbf{Z}_t, t) + \mathbf{\Sigma}\mathrm{d}\mathbf{W}_t, \\ &\mathbf{g}(\mathbf{Z}_t, t) = -\mathbf{A}(t)\mathbf{Z}_t + \mathbf{b}(t), \end{aligned} \tag{2.22}$$

where $\mathbf{A}(t) \in \mathbb{R}^{N \times N}$ and $\mathbf{b}(t) \in \mathbb{R}^N$ are the functions of the linear drift (drift that is linear with respect to the input values) $\mathbf{g} : \mathbb{R}^N \times [0, T] \longrightarrow \mathbb{R}^N$. The reason for having a linear SDE to approximate the posterior distribution p is that the

solution of the SDE in Equation (2.22) correspond to a Gaussian process (GP) and thus Gaussian distributed. As mentioned in the preceding section, this means that its first and second order moments fully characterize the solution of its posterior distribution given from the FPK equation. As shown in [27], the ODEs for the first and second order moments for Equation (2.22) is

$$\begin{aligned} \frac{\mathrm{d}\mathbf{m}(t)}{\mathrm{d}t} &= -\mathbf{A}(t)\mathbf{m}(t) + \mathbf{b}(t) \\ \frac{\mathrm{d}\mathbf{S}(t)}{\mathrm{d}t} &= -\mathbf{A}(t)\mathbf{S}(t) - \mathbf{S}(t)\mathbf{A}^{\mathrm{T}}(t) + \boldsymbol{\Sigma}, \end{aligned} \tag{2.23}$$

for some $s, t \in [0, T]$ where $s \leq t$. Hence, the interest is to find $\mathbf{A}(t)$ and $\mathbf{b}(t)$ such that the distribution of $\mathbf{Z}_t$ under the SDE in Equation (2.22) approximates the posterior distribution of $\mathbf{Z}_t$ under the enforced SDE in Equation (2.21). By denoting the distribution of $\mathbf{Z}_t$ under the SDE in Equation (2.22) as $q(\mathbf{Z}_t, t)$, this is done by minimizing the so called variational free-energy which is defined, as according to [26], as

$$\mathcal{F}^{FE}(\mathbf{A}, \mathbf{b}) = \int_0^T E_{sde}(t)\mathrm{d}t + \int_0^T E_{obs}(t) \sum_{k=0}^{T} \delta(t-k)\mathrm{d}t + \mathrm{KL}_D[q(\mathbf{Z}_0)||p(\mathbf{Z}_0)], \tag{2.24}$$

where KL_D is the Kullback-Leibler divergence, $\delta(\cdot)$ is the Dirac's delta function and k is the index of the discrete time interval $\{0, \ldots, T\}$ for which the values of $\mathbf{x}$ are observed. Moreover,

$$\begin{aligned} E_{sde}(t) &= \frac{1}{2}\left\langle (f(\mathbf{Z}_t, t) - g(\mathbf{Z}_t, t))^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (f(\mathbf{Z}_t, t) - g(\mathbf{Z}_t, t)) \right\rangle_q, \quad t \in [0, T], \\ E_{obs}(t) &= \frac{1}{2}\left\langle (\mathbf{X}_t - \mathbf{Z}_t)^{\mathrm{T}} (r^2\mathbf{I})^{-1} (\mathbf{X}_t - \mathbf{Z}_t) \right\rangle_q \\ &\quad + \frac{K \ln 2\pi}{2} + \frac{\ln |r^2\mathbf{I}|}{2}, \quad t \in \{0, \ldots, T\}. \end{aligned} \tag{2.25}$$

The variational free-energy in Equation (2.24) is henceforth referred to only as the free-energy. To optimize $\mathbf{A}$ and $\mathbf{b}$ such that the free-energy is minimized, the ODEs for the first and second order moments in Equation (2.23) can be used as consistency constraints for the means and covariances along sample paths of Equation (2.22). These constraints can be enforced by introducing the Lagrangian, $\mathcal{L}$, defined, according to [26] and [27], as

$$\begin{aligned} \mathcal{L} =& \mathcal{F}^{FE}(\mathbf{A}, \mathbf{b}) - \int_0^T \boldsymbol{\lambda}^{\mathrm{T}}(t)\Big(\frac{\partial \mathbf{m}}{\partial t}(t) + \mathbf{A}(t)\mathbf{m}(t) - \mathbf{b}(t)\Big)\mathrm{d}t \\ & - \int_0^T \mathrm{Tr}\left[\boldsymbol{\Psi}(t)\Big(\frac{\partial \mathbf{S}}{\partial t}(t) + 2\mathbf{A}(t)\mathbf{S}(t) - \boldsymbol{\Sigma}\Big)\right]\mathrm{d}t, \end{aligned} \tag{2.26}$$

where $\boldsymbol{\lambda}(t) \in \mathbb{R}^N$ and $\boldsymbol{\Psi}(t) \in \mathbb{R}^{N \times N}$ are the time dependent Lagrange multipliers with $\boldsymbol{\Psi}(t)$ being symmetric and $\mathrm{Tr}[\cdot]$ refers to the trace of a matrix. The interest is to minimize the free-energy with respect to the independent variations of $\mathbf{A}(t)$, $\mathbf{b}(t)$, $\mathbf{m}(t)$ and $\mathbf{S}(t)$ subject to the constraints in Equation (2.23). Therefore, the

stationary points in Equation (2.26) are desired. To allow for this, integration by parts is used for Equation (2.26) resulting in

$$\begin{aligned}\mathcal{L} =&\mathcal{F}^{FE}(\mathbf{A},\mathbf{b}) - \int_0^T \left[\boldsymbol{\lambda}^{\mathrm{T}}(t)(\mathbf{A}(t)\mathbf{m}(t)-\mathbf{b}(t)) - \frac{\partial \boldsymbol{\lambda}^{\mathrm{T}}}{\partial t}(t)\mathbf{m}(t)\right]\mathrm{d}t \\ &- \int_0^T \mathrm{Tr}\left[\boldsymbol{\Psi}(t)(2\mathbf{A}(t)\mathbf{S}(t)-\boldsymbol{\Sigma}) - \frac{\partial \boldsymbol{\Psi}}{\partial t}(t)\mathbf{S}(t)\right]\mathrm{d}t \\ &- \boldsymbol{\lambda}^{\mathrm{T}}(T)\mathbf{m}(T) + \boldsymbol{\lambda}^{\mathrm{T}}(0)\mathbf{m}(0) - \mathrm{Tr}[\boldsymbol{\Psi}(T)\mathbf{S}(T)] + \mathrm{Tr}[\boldsymbol{\Psi}(0)\mathbf{S}(0)].\end{aligned} \tag{2.27}$$

However, at the final time T, only the changes for $\mathbf{A}(T)$ and $\mathbf{b}(T)$ are considered. Therefore, as according to [27], $\boldsymbol{\Psi}(T) = \boldsymbol{\lambda}(T) = \mathbf{0}$. Moreover, the initial values of $\mathbf{S}(0)$ and $\mathbf{m}(0)$ are also fixed, meaning that they are not under subject of optimization. The gradients with respect to $\mathbf{A}(t)$ and $\mathbf{b}(t)$ of $\mathcal{L}$ are therefore, as recalled from [27], evaluated by taking the functional derivatives as

$$\nabla_{\mathbf{A}}\mathcal{L} = \nabla_{\mathbf{A}}E_{sde}(t) - \boldsymbol{\lambda}(t)\mathbf{m}^{\mathrm{T}}(t) - 2\boldsymbol{\Psi}(t)\mathbf{S}(t) \tag{2.28}$$

$$\nabla_{\mathbf{b}}\mathcal{L} = \nabla_{\mathbf{b}}E_{sde}(t) + \boldsymbol{\lambda}(t), \tag{2.29}$$

where

$$\nabla_{\mathbf{A}}E_{sde}(t) = \boldsymbol{\Sigma}^{-1}\left[\langle\nabla_{\mathbf{Z}}f(\mathbf{Z}_t,t)\rangle_q + \mathbf{A}(t)\right]\mathbf{S}(t) - \nabla_{\mathbf{b}}E_{sde}(t)\mathbf{b}(t)\mathbf{m}^{\mathrm{T}}(t), \tag{2.30}$$

$$\nabla_{\mathbf{b}}E_{sde}(t) = \boldsymbol{\Sigma}^{-1}\left[-\langle f(\mathbf{Z}_t,t)\rangle_q - \mathbf{A}(t)\mathbf{m}(t) + \mathbf{b}(t)\right], \tag{2.31}$$

of which $\langle f(\mathbf{Z}_t,t)(\mathbf{Z}_t-\mathbf{m}(t))^{\mathrm{T}}\rangle_q = \langle\nabla_{\mathbf{Z}}f(\mathbf{Z}_t,t)\rangle_q\mathbf{S}(t)$ has been used in Equation (2.30), as according to [26]. Thereafter, obtaining the gradients with respect to $\mathbf{m}(t)$ and $\mathbf{S}(t)$ implies taking the functional derivatives of $\mathcal{L}$ which yields

$$\nabla_{\mathbf{m}}\mathcal{L} = \frac{\partial\boldsymbol{\lambda}}{\partial t}(t) + \nabla_{\mathbf{m}}E_{sde}(t) - \mathbf{A}^{\mathrm{T}}(t)\boldsymbol{\lambda}(t), \tag{2.32}$$

$$\nabla_{\mathbf{S}}\mathcal{L} = \frac{\partial\boldsymbol{\Psi}}{\partial t}(t) + \nabla_{\mathbf{S}}E_{sde}(t) - 2\boldsymbol{\Psi}(t)\mathbf{A}(t), \tag{2.33}$$

with the corresponding jump conditions that are enforced when observations occur,

$$\boldsymbol{\lambda}(t^+) = \boldsymbol{\lambda}(t^-) - \nabla_{\mathbf{m}}E_{obs}(t)|_{t\in\{0,\dots,T\}}, \tag{2.34}$$

$$\boldsymbol{\Psi}(t^+) = \boldsymbol{\Psi}(t^-) - \nabla_{\mathbf{S}}E_{obs}(t)|_{t\in\{0,\dots,T\}}, \tag{2.35}$$

where the $|_{t\in\{0,\dots,T\}}$ means that the term is only evaluated for when the time t is equal to the time of the observations which are observed at the discrete times $t \in \{0,\dots,T\}$. Setting Equation (2.32) and (2.32) to zero yields the ODEs of the time derivative of $\boldsymbol{\lambda}(t)$ and $\boldsymbol{\Psi}(t)$. The optimization goal is then to update $\mathbf{A}(t)$ and $\mathbf{b}(t)$ with the use of the explicit gradients in Equation (2.28) and (2.29). Though, since $\boldsymbol{\lambda}(t)$, $\boldsymbol{\Psi}(t)$, $\mathbf{m}(t)$ and $\mathbf{S}(t)$ are also dependent on $\mathbf{A}(t)$ and $\mathbf{b}(t)$, the implicit gradients must also be considered. However, if the consistency constraints in Equation (2.23) and the jump constraints in Equation (2.34) and (2.35) are satisfied, the implicit gradients vanish [26]. To obtain this, a forward propagation of solving the first and second order moments in Equation (2.23) using the initial conditions $\mathbf{m}(0)$, $\mathbf{S}(0)$ and the current values of $\mathbf{A}$ and $\mathbf{b}$ are performed. Then, a backward

solver that solves for the Lagrange multipliers with the current values of $\mathbf{m}$, $\mathbf{S}$, $\mathbf{A}$ and $\mathbf{b}$ along with the initial conditions $\boldsymbol{\lambda}(T) = \boldsymbol{\Psi}(T) = \mathbf{0}$ are performed using the ODEs obtained by setting Equation (2.32) and (2.33) to zero. Then finally, $\mathbf{A}(t)$ and $\mathbf{b}(t)$ is updated for all $t \in [0, T]$ using the gradients $\nabla_{\mathbf{A}}\mathcal{L}$ and $\nabla_{\mathbf{b}}\mathcal{L}$ in Equation (2.30) and (2.31) respectively as

$$\mathbf{A}(t) \leftarrow \mathbf{A}(t) - \eta\nabla_{\mathbf{A}}\mathcal{L}, \tag{2.36}$$

$$\mathbf{b}(t) \leftarrow \mathbf{b}(t) - \eta\nabla_{\mathbf{b}}\mathcal{L}, \tag{2.37}$$

with gradient step size $\eta \in (0, 1]$ [27]. That is, using gradient descent. The pseudo code for finding the optimal posterior distribution q using the free-energy approximation is shown in Algorithm 2.

Algorithm 2: Finding the optimal posterior distribution q using free-energy approximation.

Data: $\mathcal{D}$, $\mathbf{m}(0)$, $\mathbf{S}(0)$, Σ, η, K, T
$\Delta t \leftarrow T/K$;
Initialize $\{\mathbf{A}(k), \mathbf{b}(k)\}_{k\geq 0}$;
repeat
  for $k = 0, \ldots, K-1$ **do**
    $\mathbf{m}(k+1) \leftarrow \mathbf{m}(k) - (\mathbf{A}(k)\mathbf{m}(k) - \mathbf{b}(k))\Delta t$;
    $\mathbf{S}(k+1) \leftarrow \mathbf{S}(k) - (\mathbf{A}(k)\mathbf{S}(k) + \mathbf{S}(k)\mathbf{A}^{\mathrm{T}}(k) - \Sigma)\Delta t$;
  end
  for $k = K, \ldots, 1$ **do**
    $\boldsymbol{\lambda}(k-1) \leftarrow \boldsymbol{\lambda}(k) + (\nabla_{\mathbf{m}} E_{sde}|_{t=t_k} - \mathbf{A}^{\mathrm{T}}(k)\boldsymbol{\lambda}(k))\Delta t$;
    $\boldsymbol{\Psi}(k-1) \leftarrow \boldsymbol{\Psi}(k) + (\nabla_{\mathbf{S}} E_{sde}|_{t=t_k} - 2\boldsymbol{\Psi}(k)\mathbf{A}(k))\Delta t$;
    if *observation at* t_{k-1} **then**
      $\boldsymbol{\lambda}(k-1) \leftarrow \boldsymbol{\lambda}(k-1) + \nabla_{\mathbf{m}} E_{obs}|_{t=t_{k-1}}$;
      $\boldsymbol{\Psi}(k-1) \leftarrow \boldsymbol{\Psi}(k-1) + \nabla_{\mathbf{S}} E_{obs}|_{t=t_{k-1}}$;
    end
  end
  compute $\{\tilde{\mathbf{A}}(k), \tilde{\mathbf{b}}(k)\}_{k\geq 0}$;
  update $\{\mathbf{A}, \mathbf{b}\}_{k\geq 0}$ using Equation (2.36) and (2.37);
until *minimum of* $\mathcal{L}$ *is obtained*;

The free-energy approximation aims to find the optimal posterior distribution of some observed data under the assumption that the observed data follows some SDE. However, there are several issues involved in doing so. Firstly, the choice of the SDE for the observed data needs to be pre-defined and the results may vary a lot based on this choice. Secondly, both SDEs assume a constant diffusion term, meaning that the noise of the observations are assumed to keep the same variance, independent of time and the values themselves. Thirdly, the computations required for the gradients can become difficult for more complicated choices of the parameterized drift f. Moreover, the complexity of the algorithm is of order $\mathcal{O}(MKD_{hid}^3)$, where M is the number of iterations of the outer loop, hence it can become computationally expensive for higher dimensions and larger number of discretization steps [27]. However, with the use of machine learning, approximating the observed data can be done with neural

networks. Hence, instead of relying on the pre-defined SDE, the approximations of the drift and diffusion terms of the SDE can be optimized to best fit the observed data with respect to the objective. This is described in the following section.

2.2.3 Neural SDEs

A neural SDE is a stochastic differential equation as in Equation (2.4) of which the drift and diffusion functions $\mathbf{f}$ and $\mathbf{g}$ are parametrized by neural networks. Given some data $\mathbf{x}(t) \in \mathbb{R}^N$ for $t = \{0, 1, \dots, T\}$, a single layer neural network $f^*(\cdot; \Theta_f)$ with weight matrix $\Theta_f \in \mathbb{R}^{N_{nn} \times N}$ is defined as

$$\mathbf{f}^*(\mathbf{x}(t); \Theta_f) = \alpha(\Theta_f \mathbf{x}(t) + \mathbf{b}), \tag{2.38}$$

where $\mathbf{b} \in \mathbb{R}^{N_{nn}}$ is the bias vector, $\alpha : \mathbb{R}^{N_{nn}} \longrightarrow \mathbb{R}^{N_{nn}}$ is the corresponding activation function and N_{nn} is the so called width of the neural network, also referred to as its dimension. Therefore, let $\mathbf{f}^* : \mathbb{R}^N \longrightarrow \mathbb{R}^{N_{nn}}$ and $\mathbf{g}^* : \mathbb{R}^N \longrightarrow \mathbb{R}^{N_{nn}}$ be the single layer drift and diffusion neural networks, then the neural SDE of some process $\mathbf{X}$ that corresponds to the data $\mathbf{x}$, takes the form

$$\mathrm{d}\mathbf{X}_t = \mathbf{f}^*(\mathbf{X}_t; \boldsymbol{\Theta}_f)\mathrm{d}t + \mathbf{g}^*(\mathbf{X}_t; \boldsymbol{\Theta}_g) \odot \mathrm{d}\mathbf{W}_t, \quad \mathbf{X}_0 \in \mathbb{R}^N, \tag{2.39}$$

where Θ_f and Θ_g are the corresponding weights of the networks $\mathbf{f}^*$ and $\mathbf{g}^*$ respectively and $\mathbf{W}$ is a N_{nn}-dimensional Brownian motion and $\odot$ is the elementwise product (Hadamard). Apart from the demonstrated learning ability of neural networks, a theoretical motivation for using neural networks to approximate the drift and diffusion functions is a result referred to as the universal approximation theorem. The universal approximation theorem states that neural networks are universal approximators. This means that a neural network of multiple layers can theoretically approximate any measurable function to any degree of accuracy [28, 29]. Therefore, in situations when the drift and diffusion functions are unknown, or when some data is assumed to follow a SDE, but the identification of the type of SDE is uncertain, neural networks can be applied to approximate this process. Hence, the identification of the drift and diffusion functions can be automated by updating the parameters in the neural networks to best fit the solution of the corresponding SDE [7].

Moreover, a neural SDE as in Equation (2.39), can be seen as using multiple residual connections, a method which is commonly used in many deep neural network topologies such as DenseNets and ResNets which have achieved state-of-the-art performance [30]. Residual connections were introduced to approach the problems of training extremely deep neural networks such as vanishing and exploding gradients, saturated performance and high computational cost. By including an additional bypass around the activation functions between layers, the information is allowed to flow without attenuation [30]. Such a bypass is called a residual connection, or a skip connection, and is defined as

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{f}^*(\mathbf{x}(t); \Theta_f), \quad t \in \{0, \dots, T-1\}. \tag{2.40}$$

To connect residual connections to neural SDEs, first consider the deterministic case of having zero diffusion in Equation (2.39). Then the neural SDE evaluates as a neural ODE with

$$\mathrm{d}\mathbf{X}_t = \mathbf{f}^*(\mathbf{X}_t; \Theta_f)\mathrm{d}t, \quad t \in \{0, \dots, T\},$$

which can be numerically approximated as

$$\mathbf{X}_{t+\Delta t} = \mathbf{X}_t + \mathbf{f}^*(\mathbf{X}_t; \Theta_f)\Delta t. \tag{2.41}$$

Hence, comparing to Equation (2.40), the residual connection and the neural ODE in Equation (2.41) take a similar form [31]. The neural SDE can thus be viewed as a stochastic residual connection between the time steps in $\{0, \dots, t\}$.

In order for unique solution to exist for Equation (2.39), the neural networks $\mathbf{f}^*$ and $\mathbf{g}^*$ must satisfy the conditions in Theorem 2.1. To achieve this, a certain transformation can be made upon the weight matrices of the neural networks. Specifically, for arbitrary sizes of the neural networks, assume that $\mathbf{f}^*$ and $\mathbf{g}^*$ are neural networks with L and K number of layers respectively, that is,

$$\begin{aligned} \mathbf{f}^*(\mathbf{x}(t); \boldsymbol{\Theta}_f) = W_f^{(L)} \circ \cdots \circ W_f^{(1)}(\mathbf{x}(t)), \quad \text{where} \quad W_f^{(l)} : \mathbb{R}^{d_{l-1}} \longrightarrow \mathbb{R}^{d_l}, \\ \text{s.t} \quad W_f^{(l)}(\mathbf{d}) = \alpha_f^{(l)}(\Theta_f^{(l)}\mathbf{d} + \mathbf{b}_f^{(l)}), \quad \text{for any } \mathbf{d} \in \mathbb{R}^{d_{l-1}}, \end{aligned} \tag{2.42}$$

and similarly

$$\begin{aligned} \mathbf{g}^*(\mathbf{x}(t); \boldsymbol{\Theta}_g) = W_g^{(K)} \circ \cdots \circ W_g^{(1)}(\mathbf{x}(t)), \quad \text{where} \quad W_g^{(k)} : \mathbb{R}^{d_{k-1}} \longrightarrow \mathbb{R}^{d_k}, \\ \text{s.t} \quad W_g^{(k)}(\mathbf{d}) = \alpha_g^{(k)}(\Theta_g^{(k)}\mathbf{d} + \mathbf{b}_g^{(k)}), \quad \text{for any } \mathbf{d} \in \mathbb{R}^{d_{k-1}}, \end{aligned} \tag{2.43}$$

where $l \in \{1, \dots, L\}$, $k \in \{1, \dots, K\}$, $\boldsymbol{\Theta}_f = (\Theta_f^{(1)}, \dots, \Theta_f^{(L)})$ where $\Theta_f^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ and $\boldsymbol{\Theta}_g = (\Theta_g^{(1)}, \dots, \Theta_g^{(K)})$ where $\Theta_g^{(K)} \in \mathbb{R}^{d_k \times d_{k-1}}$ are the weight matrices for f^* and g^* respectively and $\mathbf{b}_f^{(l)} \in \mathbb{R}^{d_l}$, $\mathbf{b}_g^{(k)} \in \mathbb{R}^{d_k}$ are the corresponding bias vectors for l:th respectively k:th layer and d_l, d_k is the corresponding dimension [32]. If the networks f^* and g^* shall satisfy the conditions in Theorem 2.1, then transformations are needed on all the weight matrices $\boldsymbol{\Theta}_f$ and $\boldsymbol{\Theta}_g$. This is done by applying spectral normalization to $\boldsymbol{\Theta}_f$ and $\boldsymbol{\Theta}_g$ as originally proposed in [32]. This is stated below.

Theorem 2.2. *Let $\mathbf{f}^*$ and $\mathbf{g}^*$ be as in Equation (2.42) and (2.43) respectively and assume that it holds, for both $\mathbf{f}^*$ and $\mathbf{g}^*$, that each activation function is in each corresponding layer satisfies*

$$||\alpha_f^{(l)}(\mathbf{x}) - \alpha_f^{(l)}(\mathbf{y})|| \leq C_f^{(l)}||\mathbf{x} - \mathbf{y}||,$$

for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{d_{l-1}}$, and $l \in \{1, \dots, L\}$ and some constant $C_f^{(l)} \geq 0$, and the same for g^*. Define the spectral norm of a matrix $W \in \mathbb{R}^{N \times N}$ as

$$\sigma(W) := \max_{\mathbf{h} \in \mathbb{R}^N : \mathbf{h} \neq \mathbf{0}} \frac{||W\mathbf{h}||}{||\mathbf{h}||} = \max_{||\mathbf{h}|| \leq 1} ||W\mathbf{h}||,$$

then, by applying the transformation

$$\tilde{\Theta}_f^{(l)} = \frac{\Theta_f^{(l)}}{\sigma(\Theta_f^{(l)})}, \quad \textit{and} \quad \tilde{\Theta}_g^{(k)} = \frac{\Theta_g^{(k)}}{\sigma(\Theta_g^{(k)})}, \quad l \in \{1, \cdots, L\}, \quad k \in \{1, \cdots, K\}, \tag{2.44}$$

to each weight matrix in $\mathbf{\Theta}_f$ *and* $\mathbf{\Theta}_g$, $\mathbf{f}^*$ *and* $\mathbf{g}^*$ *satisfy the conditions in Theorem 2.1.*

Proof. First, define the Lipschitz norm of $\mathbf{f}^*$, $||\mathbf{f}^*||_{\text{Lip}}$ as

$$||\mathbf{f}^*||_{\text{Lip}} := \frac{||f^*(\mathbf{x}) - \mathbf{f}^*(\mathbf{y})||}{||\mathbf{x} - \mathbf{y}||},$$

for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$ and the corresponding Lipschitz constant as the smallest value c such that

$$||\mathbf{f}^*||_{\text{Lip}} \leq c.$$

Furthermore, let $\mathbf{f}_i^*$ be i:th layer in the network f^* such that $\mathbf{f}_i^* : \mathbf{h}_{in}^{(i)} \mapsto \mathbf{h}_{out}^{(i)}$ where $\mathbf{h}_{in}^{(i)}$ and $\mathbf{h}_{out}^{(i)}$ is the input respectively output vector corresponding to layer i. It holds by definition that $||\mathbf{f}_i^*||_{\text{Lip}} = \sup_{\mathbf{h}} \sigma\left(\nabla f_i^*(\mathbf{h})\right)$ and so for a linear mapping $\mathbf{f}_i^*(\mathbf{h}) = \Theta_f^{(i)}\mathbf{h} + \mathbf{b}_f^{(i)}$, the Lipschitz norm is given by $||\mathbf{f}_i^*||_{\text{Lip}} = \sup_{\mathbf{h}} \sigma(\nabla f_i^*(\mathbf{h})) = \sup_{\mathbf{h}} \sigma(\Theta_f^{(i)}) = \sigma(\Theta_f^{(i)})$. Moreover, from assumption it holds that the Lipschitz constant of each activation function corresponding to layer i is equal to $C_f^{(i)}$ such that $||\alpha_f^{(i)}||_{\text{Lip}} \leq C_f^{(i)}$ [1]. Then by using the inequality

$$||\mathbf{f}_i^* \circ \mathbf{f}_{i+1}^*||_{\text{Lip}} \leq ||\mathbf{f}_i^*||_{\text{Lip}} \cdot ||\mathbf{f}_{i+1}^*||_{\text{Lip}},$$

a bound on $||\mathbf{f}^*||_{\text{Lip}}$ is achieved as

$$\begin{aligned}
||\mathbf{f}^*||_{\text{Lip}} \leq & ||\alpha_f^{(1)}||_{\text{Lip}} \cdot ||\tilde{\Theta}_f^{(1)}\mathbf{h}^{(0)} + \mathbf{b}_f^{(1)}||_{\text{Lip}} \cdot ||\alpha_f^{(2)}||_{\text{Lip}} \cdot ||\tilde{\Theta}_f^{(2)}\mathbf{h}^{(1)} + \mathbf{b}_f^{(2)}||_{\text{Lip}} \\
& \cdots ||\alpha_f^{(L)}||_{\text{Lip}} \cdot ||\tilde{\Theta}_f^{(L)}\mathbf{h}^{(L-1)} + \mathbf{b}_f^{(L)}||_{\text{Lip}} = C_f \prod_{l=1}^{L} ||\tilde{\Theta}_f^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}_f^{(l)}||_{\text{Lip}} \\
= & C_f \prod_{l=1}^{L} \sigma(\tilde{\Theta}_f^{(l)}) = C_f \prod_{l=1}^{L} \sigma\left(\frac{\Theta_f^{(l)}}{\sigma(\Theta_f^{(l)})}\right) = C_f \cdot 1 = C_f,
\end{aligned}$$

where $C_f = \prod_{l=1}^{L} C_f^{(l)}$. For assumed finite L, $C_f < \infty$. Thus, it holds that

$$||\mathbf{f}^*||_{\text{Lip}} = \frac{||\mathbf{f}^*(\mathbf{x}) - \mathbf{f}^*(\mathbf{y})||}{||\mathbf{x} - \mathbf{y}||} \leq C_f,$$

for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$. The proof of the second condition in Theorem 2.1 of Equation (2.6), is not provided by [32], but can easily be shown using the result of the first

[1] For many common activation functions, $||\alpha||_{\text{Lip}} \leq 1$, such as for the ReLU and tanh-functions [32].

condition. Knowing that $||\mathbf{f}^*(\mathbf{x}) - \mathbf{f}^*(\mathbf{y})|| \leq C||\mathbf{x} - \mathbf{y}||$ for some constant C and for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$, then consider

$$\begin{aligned} ||\mathbf{f}^*(\mathbf{x})|| =& ||\mathbf{f}^*(\mathbf{x}) - \mathbf{f}^*(\mathbf{0}) + \mathbf{f}^*(\mathbf{0})|| \leq ||\mathbf{f}^*(\mathbf{x}) - \mathbf{f}^*(\mathbf{0})|| + ||\mathbf{f}^*(\mathbf{0})|| \\ &\leq C||\mathbf{x} - \mathbf{0}|| + ||\mathbf{f}^*(\mathbf{0})|| = C||\mathbf{x}|| + ||\mathbf{f}^*(\mathbf{0})|| = C||\mathbf{x}|| + K, \end{aligned}$$

for some constants C and K and where $\mathbf{0} \in \mathbb{R}^N$, which satisfies the second condition in Equation (2.6) in Theorem 2.1. Hence, by applying the spectral normalization in Equation (2.44) to each weight matrix in the network $\mathbf{f}^*$ then $\mathbf{f}^*$ satisfy the conditions in Theorem 2.1. The proof for $\mathbf{g}^*$ is identical. ■

From the results in Theorem 2.2, it is showed that applying the spectral normalization in Equation (2.44) to each weight matrix in the networks $\mathbf{f}^*(\cdot; \boldsymbol{\Theta}_f)$ and $\mathbf{g}^*(\cdot; \boldsymbol{\Theta}_g)$, the networks satisfy the conditions in Theorem 2.1 required for the SDE in Equation (2.39) to be solvable when approximating the drift and diffusion functions with $\mathbf{f}^*$ and $\mathbf{g}^*$ respectively.

By using neural SDEs, drift and diffusion functions can be optimized to fit $\mathbf{x}(t)$ for all $t \in [0, T]$. However, this approach alone does not account for historical values $\mathbf{x}(s)$ for some $0 \leq s < t$, $s, t \in \{0, \ldots, T\}$, but rather only the current value $\mathbf{x}(t)$. In order to make the model also consider historical values, an additional type of machine learning methods can be applied, namely recurrent neural networks. These take a sequence of observations as input and aims to also consider the time-dependencies between the different time steps of the input [33]. Incorporating this in the neural SDE framework, results into what in this thesis is called, a latent variable neural SDE. This is presented in to proceeding section.

2.2.4 Latent Variable Neural SDEs

The term latent variable refers to a variable that is unobserved and, in the use of machine learning, often meaning a representation of some input data. Latent variables are often mentioned in methods for dimensionality reduction [34] and in generative models [35]. The idea is to map the observed input data to latent variables, referred as a latent mapping. The aim of this is to create a more meaningful representation of the observed input data. For generative models, the goal is to optimize this mapping so that new data can be generated from the latent variables, such that the new data belongs to the same distribution as the input data. If so, the latent variables are said to be a well representation of the observed data [35]. For dimensionality reduction models, the objective is to map the observed input data of higher dimensions down to latent variables of lower dimensions, $D_{obs} > D_{lat}$, while maintaining any valuable information in the observed data [34, 36]. This is often used for visualization purposes, clustering tasks or for improved classification in classification tasks. The common goal is, however, to optimize the latent mapping such that the latent variables best represents the observed input data in the sense that no meaningful information of the observed data is lost [20].

A specific type of model that utilizes latent variables under the assumption that

the observed input data is sampled from an unknown underlying process is Variational Autoencoders (VAEs) [20]. VAEs are generative models that consist of a recognition model and a generative model. The former aims to learn a conditional distribution of the latent variables conditioned on the observed input data, usually approximated by neural networks. Let $\mathbf{x}(t) \in \mathbb{R}^{D_{obs}}$ be some observed input data and $\mathbf{y}(t) \in \mathbb{R}^{D_{targ}}$ be some observed target data, both discretely observed for $t \in \{0, \dots, T\}$ for some $T > 0$. Moreover, let $\mathbf{z}(t) \in \mathbb{R}^{D_{lat}}$ denote some latent variables for all $t \in \{0, \dots, T\}$. By letting $q_{\boldsymbol{\theta}}(\mathbf{z}(t)|\mathbf{x}(t))$ denote the recognition model with some parameters $\boldsymbol{\theta}$ and $p_{\boldsymbol{\phi}}(\mathbf{y}(t)|\mathbf{z}(t))$ denote the generative model with some parameters $\boldsymbol{\phi}$ for all $t \in \{0, \dots, T\}$. The aim of these models is to optimize $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ such that the latent variables sampled from $q_{\boldsymbol{\theta}}$ can be used to reconstruct $\mathbf{y}(t)$ in the generative model $p_{\boldsymbol{\phi}}$ [20, 33]. This has empirically proven to yield better results comparing to traditional, deterministic autoencoders [37, 38, 20].

Applying the same principles as for the VAEs, but for a sequence of observed input values, Recurrent Neural Networks (RNNs) can be used. This is referred to as Variational Recurrent Neural Networks (VRNNs). A RNN is a neural network that also processes information between the different time steps of the input data [33]. This enables the model to account for previously observed values of $\mathbf{x}(t)$, and not just the value $\mathbf{x}(t)$ at some $t \in \{1, \dots, T\}$. For VRNNs, this implies that time-dependencies between time steps are induced in the latent variables. Similar with VAEs, VRNNs have been demonstrated to achieve a better performance compared to traditional RNNs in various tasks, such as speech and text modeling [33]. To connect VRNNs to stochastic modeling, assume once again that some sequence of input data is driven by some underlying process. Then a similar paradigm as for VRNNs can be applied to approximate this process. This can be compared to the free-energy approximation method described in Section 2.2.2. Here, the connection between the observed data and the underlying process is fixed. Therefore, the latent variable approach provides more flexibility as the latent mapping is subject to optimization. Therefore, rather than considering an approximate SDE of some hidden process as in the free-energy approximation, a process of latent variables can be considered.

Let $\mathbf{x} = \{\mathbf{x}(t) : t \in \{0, \dots, T\}\}$ be some observed values for some $T > 0$, where $\mathbf{x}(t) \in \mathbb{R}^{D_{obs}}$ for all $t \in \{0, \dots, T\}$ and some $D_{obs} > 0$. Assume that the values in $\mathbf{x}$ is a sample from an unknown underlying process. For the task of forecasting, the aim is to approximate the conditional distribution for some subsequential values of $\mathbf{x}$. That is, finding an approximation to $p(\mathbf{y}(T)|\mathbf{x}(0), \dots, \mathbf{x}(T))$, where $\mathbf{y}(T) = \{\mathbf{x}(T+1), \dots, \mathbf{x}(T+h)\}$ for some $h \geq 1$. Here, $D_{targ} = D_{obs}$. To do this, first define a RNN model that will be used to map the sequence of observed input values, $\mathbf{x}$, to a latent variable, $\mathbf{z} \in \mathbf{R}^{D_{lat}}$. A commonly used RNN is the Long Short-Term Memory (LSTM) model, due to its ability to learn longer dependencies compared to traditional RNNs. An illustration of a LSTM model is shown in Figure 2.2. Each layer in the LSTM model consists of four so called gates; an input gate, a forget gate, a cell gate and an output gate. For each element in the input sequence,

that is, for each $\mathbf{x}(t)$ for $t \in \{0, \dots, T\}$, one layer of the LSTM model computes

$$\begin{cases} \mathbf{i}_t = \text{sigmoid}(\Theta_{ii}\mathbf{x}(t) + \mathbf{b}_{ii} + \Theta_{hi}\mathbf{h}_{t-1} + \mathbf{b}_{hi}) \\ \mathbf{f}_t = \text{sigmoid}(\Theta_{if}\mathbf{x}(t) + \mathbf{b}_{if} + \Theta_{hf}\mathbf{h}_{t-1} + \mathbf{b}_{hf}) \\ \mathbf{g}_t = \tanh(\Theta_{ig}\mathbf{x}(t) + \mathbf{b}_{ig} + \Theta_{hg}\mathbf{h}_{t-1} + \mathbf{b}_{hg}) \\ \mathbf{o}_t = \text{sigmoid}(\Theta_{io}\mathbf{x}(t) + \mathbf{b}_{io} + \Theta_{ho}\mathbf{h}_{t-1} + \mathbf{b}_{ho}) \\ \mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\ \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \end{cases} \tag{2.45}$$

where the subscripts i, f, c and o refers to the input, forget, cell and output gates respectively. $\Theta_{if} \in \mathbb{R}^{D_{obs} \times D_{LSTM}}$ further refers to the weight matrix that acts on the input in gate f whilst $\Theta_{hf} \in \mathbb{R}^{D_{LSTM} \times D_{LSTM}}$ refers to the weight matrix that acts on the hidden vector h_{t-1} in gate f. Similarly, the bias vectors $\mathbf{b}_{iq} \in \mathbb{R}^{D_{LSTM}}$ and $\mathbf{b}_{hq} \in \mathbb{R}^{D_{LSTM}}$ acts on the input respectively hidden vector in gate $q \in \{i, f, g, o$. Moreover, $\odot$ refers to the elementwise product (Hadamard). For $t = 1$, $\mathbf{h}_{t-1}$ and $\mathbf{c}_{t-1}$ are usually zero initiated [39]. Here, D_{LSTM} refers to the dimension of the LSTM network. For the final time step, the output latent variable is $\mathbf{h}_T$, which can then be processed through an additional linear layer that maps from $\mathbb{R}^{D_{LSTM}}$ to $\mathbb{R}^{D_{lat}}$ to produce the latent variable $\mathbf{z}_0$. In the remainder of this thesis, the last linear layer will be considered as an integrated part of a LSTM model.

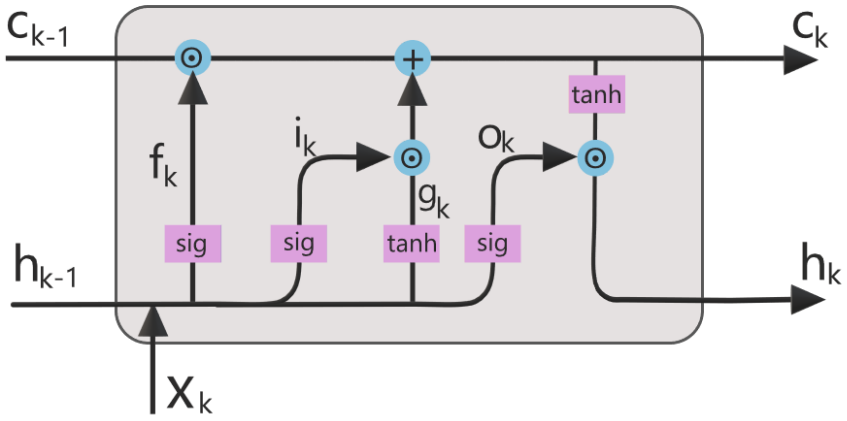


Figure 2.2: *Information flow of a LSTM model for a single time step k where sig refers to the sigmoid function and tanh refers to the tanh-function. The blue dots represent either the elementwise product (circle with dot) or regular addition (plus). The values of each function* $\mathbf{f}_k$, $\mathbf{i}_k$, $\mathbf{g}_k$, $\mathbf{o}_k$ *as well as the hidden and cell vectors,* $\mathbf{h}_k$ *and* $\mathbf{c}_k$ *respectively, are computed as in Equation* (2.45).

Let $L(\cdot; \mathbf{\Theta}_L)$, where $L : \mathbb{R}^{D_{obs}} \times \mathbb{R} \longrightarrow \mathbb{R}^{D_{lat}}$, denote the LSTM model with corresponding weight matrices $\mathbf{\Theta}_L$. Then, the latent variable from this LSTM model can be treated as an initial condition for some latent variable neural SDE. That is, a stochastic process of latent variables with drift and diffusion functions as presented in Section 2.2.3. Moreover, since it is assumed that the values in $\mathbf{x}$ are samples from an unknown underlying process, the same assumption holds for the values in

$\mathbf{y}$. A realization of the latent variable process, $\mathbf{z} = \{\mathbf{z}_t : t \in \{0, \ldots, T_{sde}\}\}$ for some $T_{sde} > 0$ and where $\mathbf{z}_0 = L(\mathbf{x}(t); \boldsymbol{\Theta}_L)$, can be generated using numerical approximation methods, such as the EM method described in Section 2.1.3. The values in the latent variable process can then be used to approximate the conditional distribution of the values in $\mathbf{y}$. However, in order to obtain a mapping from the latent variables to the values in $\mathbf{y}$, an additional network, referred to as the target network, can be applied to the latent variables. This target network is a linear neural network, denoted by $L_T(\cdot; \Theta_{L_T})$ with $\Theta_{L_T} \in \mathbb{R}^{D_{targ} \times D_{lat}}$ being the corresponding weight matrix. and D_{targ} is the dimension of the values in $\mathbf{y}$. Moreover, to not restrict the number of generated latent variables in the latent process to be equal the number of predicted time steps h, T_{sde} can be selected as a positive multiple , $k_{targ} \in \{1, 2, \ldots, \}$, of h. Then, every k_{targ}:th latent variable is processed through the target network to predict the target values in $\mathbf{y}$. The weights in the latent mapping, drift and diffusion networks as well as the target network can further be optimized to best fit the target values in $\mathbf{y}$ [40]. Recalling from the VAE paradigm, this is similar to the generative model of which the aim is to approximate the conditional distribution $p(\mathbf{y}|\mathbf{z})$. Since this distribution, under the assumption that it exists, is unknown, additional assumptions need to be made. This is presented in the following section.

2.2.4.1 Gaussian Approximation Assumption

Selecting an appropriate distribution for some data for which the true distribution is unknown can be complicated. However, as seen in both Section 2.2.1 and Section 2.2.2, a common choice of approximation is the Gaussian approximation. The Gaussian distribution is one of the most commonly used distributions within machine learning and statistics and has proven to be effective in various different methods [41]. One reason for its wide usage is due to its parameters (the mean and the variance), being highly descriptive of the most fundamental properties of a distribution. The mean, μ, and variance, σ^2, correspond to the expectations

$$\mu = \mathbb{E}[y], \quad \sigma^2 = \mathbb{E}[y^2] - \mathbb{E}[y]^2,$$

for some Gaussian random variable y [41]. Recall that it is assumed that $\mathbf{x}(t)$ is driven by an underlying unknown process for all $t \in \{0, \ldots, T\}$. Further assume that $\mathbf{x}(t)$ is a Gaussian vector. That is, a vector such that any linear combination of its values is Gaussian distributed. Then, it follows that the conditional distribution $p(\mathbf{y}(t)|\mathbf{x}(t))$ is also (multivariate) Gaussian for all $t \in \{1, \ldots, h\}$. Recall that h is the number of subsequential values from $x(t)$. The aim is then to approximate the parameters in this conditional distribution. That is, the conditional mean vector $\boldsymbol{\mu}(t) \in \mathbb{R}^{D_{targ}}$ and conditional covariance matrix $\boldsymbol{\Sigma}(t) \in \mathbb{R}^{D_{targ} \times D_{targ}}$. This can be done by using a Monte-Carlo (MC) approach.

More concretely, let $\mathbf{z} = \{\mathbf{z}_t : t \in \{0, \ldots, T_{sde}\}\}$ be some generated latent variable process and let $\hat{\mathbf{y}}(t) = L_T(\mathbf{z}_t; \Theta_{L_T})$ where $t \in \{0, \ldots, h\}$ be the output of the model. In order to approximate $\boldsymbol{\mu}(t)$ and $\boldsymbol{\Sigma}(t)$, a set of N output vectors $\{\hat{\mathbf{y}}_n(t)\}_{n=1}^N$ can be generated by simulating several latent variable variable paths, $\{\mathbf{z}_n\}_{n=1}^N$ in the model. From this set of outputs, the sample mean, $\hat{\boldsymbol{\mu}}(t) \in \mathbb{R}^{D_{targ}}$, and sample

covariance, $\hat{\boldsymbol{\Sigma}}(t) \in \mathbb{R}^{D_{targ} \times D_{targ}}$, can then be computed as

$$\hat{\boldsymbol{\mu}}(t) = \frac{1}{N} \sum_{n=1}^{N} \hat{\mathbf{y}}_n(t), \tag{2.46}$$

$$\hat{\boldsymbol{\Sigma}}(t) = \frac{1}{N-1} \sum_{n=1}^{N} (\hat{\mathbf{y}}_n(t) - \hat{\boldsymbol{\mu}}(t))(\hat{\mathbf{y}}_n(t) - \hat{\boldsymbol{\mu}}(t))^{\mathrm{T}}, \tag{2.47}$$

for all $t \in \{1, \ldots, h\}$. These can then be treated as estimates for the true conditional mean vector and true conditional covariance matrix.

For a perfect MC approximation of which it is known that $\hat{\mathbf{y}}(t)$ belongs to the same distribution as $\mathbf{y}(t)$, and that the samples $\{\hat{\mathbf{y}}_n(t)\}_{n=1}^{N}$ are independent and identically distributed, it is implied by the strong law of large numbers that $\hat{\boldsymbol{\mu}}(t)$ converges almost surely to $\mathbb{E}[\mathbf{y}(t)]$, thus

$$\lim_{N \longrightarrow \infty} \hat{\boldsymbol{\mu}}(t) \overset{w.p.1}{=} \mathbb{E}[\mathbf{y}(t)],$$

where $w.p.1$ refers to 'with probability one'. Moreover, the root-mean-square error (RMSE) of the estimate $\hat{\boldsymbol{\mu}}(t)$ is proportional to $1/\sqrt{N}$ implied by the central limit theorem [42]. However, in this case when the set of outputs are generated from the latent variable neural SDE framework, it is not guaranteed that $\hat{\mathbf{y}}(t)$ and $\mathbf{y}(t)$ have the same distribution. The neural networks of the latent mapping, drift and diffusion networks and the target target network may result in the estimates in Equation (2.46) and (2.47) being biased. The bias is given as the difference between the expectation of an estimator and the true value of the parameter that is being estimated [43]. Thus, in this case, the parameter that is being estimated is the mean vector of the true conditional distribution of all $\mathbf{x}(t)$. That is, $\mathbb{E}[\mathbf{y}(t)] = \boldsymbol{\mu}(t)$. Therefore, the bias of the estimator $\hat{\boldsymbol{\mu}}(t)$ of $\mathbb{E}[\mathbf{y}(t)]$ is given as

$$\mathbb{E}\left[\hat{\boldsymbol{\mu}}(t)\right] - \mathbb{E}\left[\mathbf{y}(t)\right].$$

Moreover, the bias is related to the mean-square-error (MSE) and as well as the variance it the estimator $\hat{\mu}(t)$ as

$$\mathbb{E}\left[(\hat{\boldsymbol{\mu}}(t) - \mathbb{E}[\mathbf{y}(t)])^2\right] = \mathrm{Var}[\hat{\boldsymbol{\mu}}(t)] + (\mathbb{E}[\hat{\boldsymbol{\mu}}(t)] - \mathbb{E}[\mathbf{y}(t)])^2, \tag{2.48}$$

for $t \in \{1, \ldots, h\}$ and where the left hand side is the MSE of $\hat{\boldsymbol{\mu}}(t)$.

2.2.4.2 Optimization Procedure

In the previous sections, only a single sequence of observed values has been considered. However, during the optimization procedure of a machine learning model, referred to as the training of the model, it is desired to have several sequences of both input values and target values. To do this, let the total number of observations be T, such that $\mathbf{x}(t)$ for $t \in \{1, \ldots, T\}$. Let further $h > 0$ be the number of prediction steps in the forecast. Then, the sequence of total observations is divided into D pairs of subsequences of input and target values, denoted $\{\mathbf{x}^{(d)}, \mathbf{y}^{(d)}\}$ for $d \in \{1, \ldots, D\}$.

Let $T_{sub} < T$ denote the number of input values in each subsequence, $\mathbf{x}^{(d)}$. Then, the number of input and target pairs of subsequences is the smallest positive integer $D := \min_{c \in \mathbb{N}}\{c = (T - T_{sub})/h\}$. The input values in each subsequence can further be obtained as $\mathbf{x}^{(d)}(t) = \mathbf{x}((d-1)h+t)$, where $t \in \{1, \ldots, T_{sub}\}$. Similarly, the target values in each subsequence can be obtained by taking the h consecutive values of $\mathbf{x}^{(d)}$, as $\mathbf{y}^{(d)}(t) = \mathbf{x}((d-1)h + t)$ where $t \in \{T_{sub} + 1, T_{sub} + 1 + h\}$. An illustration of dividing the observed sequence $\mathbf{x}(t)$ for $t \in \{1, \ldots, T\}$ in to pairs of subsequences of input values and output values $\{\mathbf{x}^{(d)}, \mathbf{y}^{(d)}\}$ for $d \in \{1, \ldots, D\}$ is shown in Figure 2.3.

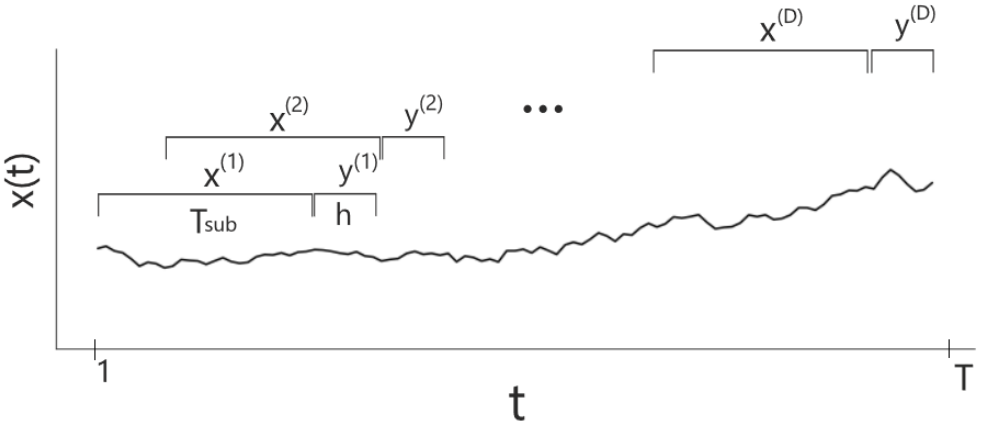


Figure 2.3: *Dividing a sequence of observed data, $\mathbf{x}(t)$, for $t \in \{1, \ldots, T\}$ into D pairs of subsequences of input data and target data, $\{\mathbf{x}^{(d)}, \mathbf{y}^{(d)}\}$ for $d \in \{1, \ldots, D\}$. Here, T_{sub} is the number of input values in each subsequence of and h is the number of target values in the target. The time step between each subsequence of $\mathbf{x}$ is h.*

The process of training a neural network is usually done through maximum likelihood estimation. Let $\mathcal{D}$ denote the set of D number of pairs of subsequences such that $\mathcal{D} = \{\mathbf{x}^{(d)}, \mathbf{y}^{(d)}\}_{d=1}^{D}$ of some observed data $\mathbf{x}(t)$ for $t \in \{1, \ldots, T\}$. Under the Gaussian assumption that the conditional distribution for each $\mathbf{y}^{(d)}(t)$ for $t \in \{1, \ldots, h\}$, conditioned on $\mathbf{x}^{(d)}(T_{sub})$ is multivariate Gaussian. Then the aim of the training procedure is to optimize the weight matrices Θ_L, Θ_f, Θ_g and Θ_{L_T} in the model such that the estimators, $\hat{\boldsymbol{\mu}}^{(d)}(t)$ and $\hat{\boldsymbol{\Sigma}}^{(d)}(t)$ for each $d \in \{1, \ldots, D\}$ and $t \in \{1, \ldots, h\}$ maximize the corresponding likelihood function for all $\mathbf{y}^{(d)}(t)$ [44]. For the Gaussian assumption, the likelihood function is

$$\mathcal{L}(\mathbf{y}(t), \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\Sigma}}(t)) = \prod_{d=1}^{D} \frac{\exp\left(-\frac{1}{2}(\mathbf{y}^{(d)}(t) - \hat{\boldsymbol{\mu}}^{(d)}(t))^{\mathrm{T}}(\hat{\boldsymbol{\Sigma}})^{-1}(\mathbf{y}^{(d)}(t) - \hat{\boldsymbol{\mu}}^{(d)}(t))\right)}{\sqrt{(2\pi)^{D_{obs}}|\hat{\boldsymbol{\Sigma}}^{(d)}(t)|}},$$

where $\mathbf{y}(t) = \{\mathbf{y}^{(d)}(t)\}_{d=1}^{D}$, $\hat{\boldsymbol{\mu}}(t) = \{\hat{\boldsymbol{\mu}}^{(d)}(t)\}_{d=1}^{D}$ and $\hat{\boldsymbol{\Sigma}} = \{\hat{\boldsymbol{\Sigma}}^{(d)}(t)\}_{d=1}^{D}$ is the target values, estimated mean vectors and estimated covariance matrices respectively for $t \in \{1, \ldots, h\}$. In practice however, for gradient based optimization methods, it is more convenient to, instead of maximizing the likelihood, minimize the negative log-likelihood (NLL) [45]. In this case, the objective instead becomes to minimize

$$\mathcal{L}_{NLL}(\mathbf{y}(t), \boldsymbol{\mu}(t), \hat{\boldsymbol{\Sigma}}(t)) = -\ln\left[\mathcal{L}(\mathbf{y}(t), \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\Sigma}}(t))\right], \tag{2.49}$$

for each $t \in \{1, \ldots, h\}$. In this thesis, only data where $D_{obs} = 1$ is considered. Hence for the sake of simplicity, let $D_{obs} = 1$. In this case, multivariate Gaussian-assumption reduces to univariate and thus the assumed conditional distribution of the input data is a univariate Gaussian. Therefore, the estimators are also univariate. Thus, in this case, the NLL is reduced to

$$\begin{aligned}
\mathcal{L}_{NL}(y(t), \hat{\mu}(t), \hat{\sigma}^2(t)) &= -\ln\left[\prod_{d=1}^{D} p_G(y^{(d)}(t)|\hat{\mu}^{(d)}(t), \hat{\sigma}^{2(d)}(t))\right] \\
&= -\ln\left[\prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\hat{\sigma}^{2(d)}(t)}} \exp\frac{-(y^{(d)}(t) - \hat{\mu}^{(d)}(t))^2}{2\hat{\sigma}^{2(d)}(t)}\right] \\
&= -\sum_{d=1}^{D} \ln\left[\frac{1}{\sqrt{2\pi\hat{\sigma}^{2(d)}(t)}} \exp\frac{-(y^{(d)}(t) - \hat{\mu}^{(d)}(t))^2}{2\hat{\sigma}^{2(d)}(t)}\right] \\
&= -\sum_{d=1}^{D} \frac{-(y^{(d)}(t) - \hat{\mu}^{(d)}(t))^2}{2\hat{\sigma}^{2(d)}(t)} - \frac{\ln\hat{\sigma}^{2(d)}(t)}{2} - C \\
&= \sum_{d=1}^{D} \frac{(y^{(d)} - \hat{\mu}^{(d)}(t))^2}{2\hat{\sigma}^{2(d)}(t)} + \frac{\ln\hat{\sigma}^{2(d)}(t)}{2} + C,
\end{aligned} \tag{2.50}$$

where the constant C is neglected. The NLL in Equation (2.50) are then summed for all $t \in \{1, \ldots, h\}$. Updating the weights is done by a gradient-based optimization methods such as stochastic gradient descent (SGD), Adam or other [45]. The pseudocode for the learning procedure of this latent variable neural SDE framework is shown in Algorithm 3.

Algorithm 3: Learning procedure for the latent variable neural SDE framework up until the loss is computed in the univariate case.

Data: $\mathcal{D}$, T_{sde}, N_{sde}, h, N
Initialize $L(\cdot;\Theta_L)$, $f^*(\cdot;\Theta_f)$, $g^*(\cdot;\Theta_g)$, $L_T(\cdot;\Theta_{L_T})$;
Apply spectral normalization to Θ_f, Θ_g as in Equation (2.44);
$\Delta t_{sde} \leftarrow T_{sde}/N_{sde}$;
$k_{targ} \leftarrow N_{sde}/h$;
for ***each*** $x^{(d)}, y^{(d)} \in \mathcal{D}$ **do**
  $\mathbf{z}_0 \leftarrow L(x^{(d)};\Theta_L)$;
  $\hat{\mathbf{y}} \leftarrow$ zero array of size $N \times h$;
  for $n = 0, \ldots, N$ **do**
    $\mathbf{z} \leftarrow$ zero array of size N_{sde};
    $\mathbf{z}(0) \leftarrow \mathbf{z}_0$;
    $\hat{\mathbf{y}}_{(i)} \leftarrow$ zero array of size h;
    $h_i \leftarrow 0$;
    for $k = 1, \ldots, N_{obs}$ **do**
      $\xi \leftarrow$ randomly generated number from $\mathcal{N}(0,1)$;
      $\mathbf{z} \leftarrow \mathbf{z}(k-1)$;
      $\mathbf{z}(k) \leftarrow \mathbf{z} + f^*(\mathbf{z};\Theta_f)\Delta t_{sde} + g^*(\mathbf{z};\Theta_g)\sqrt{\Delta t_{sde}}\xi$;
      if $(k \ mod \ k_{targ}) = 0$ **then**
        $\hat{\mathbf{y}}_{(i)}(h_i) \leftarrow L_T(\mathbf{z}(k);\Theta_{L_T})$;
        $h_i \leftarrow h_i + 1$;
      end
    end
    $\hat{\mathbf{y}}(n) \leftarrow \hat{\mathbf{y}}_{(i)}$;
  end
  $\hat{\mu} \leftarrow$ zero array of size h;
  $\hat{\sigma^2} \leftarrow$ zero array of size h;
  for $t = 1, \ldots, h$ **do**
    $\hat{\mu}(t) \leftarrow \frac{1}{N}\sum_{n=1}^{N}\hat{\mathbf{y}}(n,t)$;
    $\hat{\sigma}^2 \leftarrow \frac{1}{N-1}\sum_{n=1}(\hat{\mathbf{y}}(n,t) - \hat{\mu}(t))^2$;
  end
  $\mathcal{L} \leftarrow \sum_{t=1}^{h}\mathcal{L}_{NL}(\mathbf{y}^{(d)}(t), \hat{\mu}(t), \hat{\sigma}^2(t))$;
end

2.3 Echo State Networks

Traditional RNNs are used in various applications such as medicine, linguistics, computer vision, finance, physics amongst others and have empirically proven to be able to learn temporal dependencies in data. However, there is only a partial success in the training of these models as the backpropagation suffers from several limitations such as the exploding and vanishing gradient problem. The latter is a consequence of the backpropagation algorithm combined with the large amount of nested layers required for longer time steps [5]. Therefore, since the weights are shared among

the time steps of an input sequence and since they are updated based on the chain rule, the gradients used in the optimization can vanish since product of many small numbers tends to zero. Hence, if the gradients become sufficiently small, the weights of the network are being updated with insignificantly small factors which leads to insignificant improvements. A similar scenario can happen with exploding gradients. If the gradients in the backpropagation are large, then their values explode since the product of many large numbers tends to infinity. This results in the weights just oscillating between large values between updates and hence leading to no improvement [46]. Even though the LSTM model was developed to approach both of these problems, for long input sequences the issues remains. Further limitations in training RNN's are the computational costs due the large amount of parameters as well as high risk of overfitting [5].

Reservoir computing (RC) is a subfield of machine learning methods that uses different techniques to perform classification and regression on applications that involve processing of temporal information (data which is time-sequential) and the earliest known definition of a reservoir was presented in [47, 5]. The main part of these methods is the reservoir, an excitable system of high-dimensional temporal dynamics with fixed weights that process sequential data and outputs intermediate activation states. These values are then further processed through a readout layer. This readout layer contains the only weights subject to optimization and process the intermediate activation states and outputs the desired predictions [46]. A schematic illustration of this is shown in Figure 2.4.

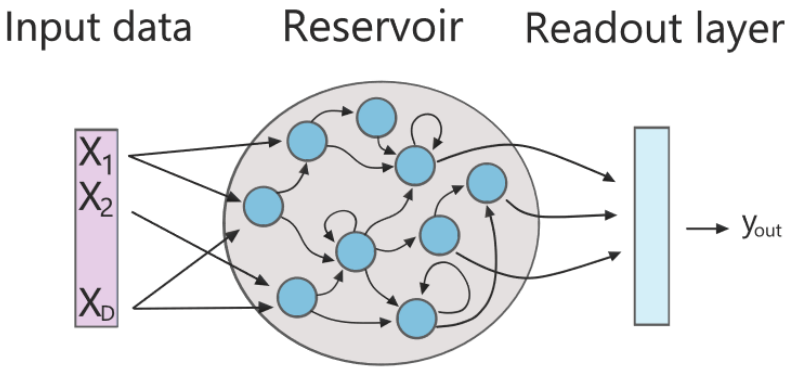


Figure 2.4: *Schematic illustration of a reservoir computer consisting of a reservoir with randomly initiated neurons which is sparsely connected and a readout layer of trainable weights. Here, some input data* $\mathbf{X} \in \mathbb{R}^D$ *are fed into the reservoir where each node (blue dots) represent the reservoir weight matrix. The arrows between two nodes represent the non-zero element weight element at the corresponding position of the two nodes. That is, the arrow between weight node* i *and node* j *represent the weight element* θ_{ij} *for all* $i, j \in [1, \dots, D_{res}]$ *where* $D_{res} \times D_{res}$ *is the size of the reservoir weight matrix. Hence, an arrow from a node into itself represent the non-zero element in the diagonal at the corresponding position.*

These models were developed as an alternative to traditional RNNs with the asset of faster training due to the fixed weights in the reservoir and also improved representations of chaotic, nonlinear and dynamical data. A specific collection of methods that belongs to the framework of reservoir computing are Echo State networks (ESNs). The schematic design of an ESN is similar to general RC models, as shown in Figure 2.5. It consists of a reservoir with sparsely connected neurons, followed by a readout layer. The weights that connect the input data to the reservoir as well as the weights within the reservoir are fixed and are randomly initiated, usually uniformly or from a binomial distribution [5, 46]. When analyzing the change in the weight values during the training of traditional RNNs of multiple layers, as done in [48], it was showed that the most prominent changes were made in the weights corresponding to the last output layers. This further encourage the use of such a reservoir, apart from being less computationally expensive due to the fixed weights. To define how the temporal information flows through an ESN, first let $\mathbf{x}(t)$ be a time-dependent process where $\mathbf{x}(t) \in \mathbb{R}^{D_{obs}}$, $t \in \{0, \cdots, T\}$. Then, the intermediate activation states out of the reservoir are computed as

$$\mathbf{u}(t) = \gamma \cdot \alpha \left(\Theta_{in}\mathbf{x}(t) + \Theta_{res}\mathbf{u}(t-1)\right) + (1-\gamma)\mathbf{u}(t-1), \tag{2.51}$$

where $\mathbf{u}(t) \in \mathbb{R}^{D_{esn}}$ is the intermediate activation states, $\Theta_{in} \in \mathbb{R}^{D_{esn} \times D_{obs}}$ is the input weights that connects the input process to the reservoir with D_{esn} being its dimension, $\Theta_{res} \in \mathbb{R}^{D_{esn} \times D_{esn}}$ is the reservoir weights, γ is a leakage rate that governs how much information to be integrated between states and α being the activation function, usually defined to either the tanh- or the *sigmoid* function. The intermediate activation states $\mathbf{u}(t)$ are then fed into the readout layer which normally

is a linear neural network $r(\mathbf{u}(t); \Theta_r) = \alpha_r(\Theta_r \mathbf{u}(t) + \mathbf{b}_r)$ with α_r, $\Theta_r \in \mathbb{R}^{D_{out} \times D_{esn}}$, $\mathbf{b}_r \in \mathbb{R}^{D_{out}}$ being the activation function, weight matrix and bias vector for the readout layer respectively [6].

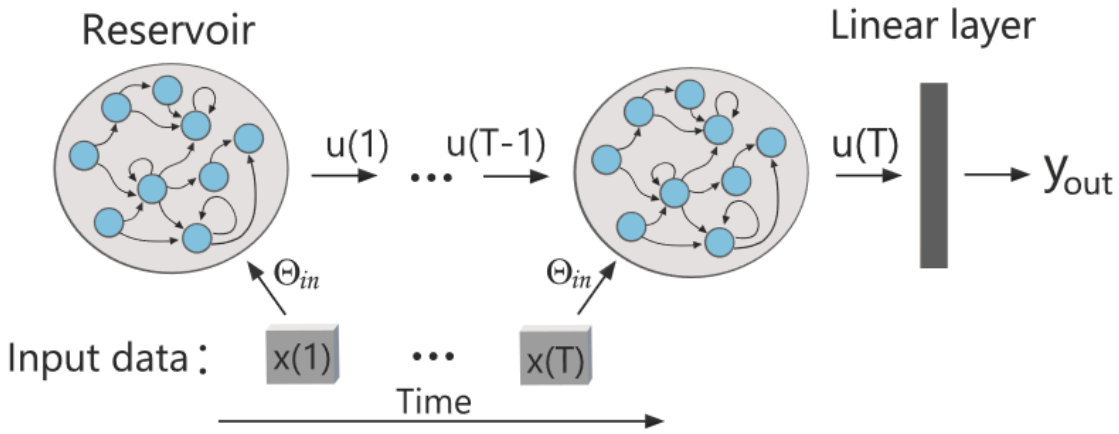


Figure 2.5: *Schematic illustration of an echo state network of which some time-dependent input data, $x(t)$, for $t \in [1, T]$, are fed into the reservoir whose time-dependent intermediate states, $u(t)$, are computed. The last state $u(T)$ are then being fed into the readout layer consisting of a linear layer which outputs the predicted values y_{out}.*

There are several steps involved for constructing an ESN [5]. Firstly, in the initialization of the input weights Θ_{in}, the input scaling parameter, $\omega_{in} \in (0, 1]$, needs to be specified such that each element, θ_{in}^{ij}, in Θ_{in} is initiated as $\theta_{in}^{ij} \sim \mathcal{U}(-\omega_{in}, \omega_{in})$, for all $i, j \in [0, D_{esn}]$, where $\mathcal{U}$ is the uniform distribution. The second step is to first specify the spectral radius parameter, ρ, used to scale the weight matrix of the reservoir, Θ_{res}, as

$$\Theta_{res} = \rho \cdot \frac{\tilde{\Theta}_{res}}{|\lambda_{max}(\tilde{\Theta}_{res})|}, \tag{2.52}$$

where $\tilde{\Theta}_{res}$ is the initialized weight matrix of the reservoir which is uniformly initialized, usually in the interval $[-0.5, 0.5]$ and $|\lambda_{max}(\tilde{\Theta}_{res})|$ is the spectral radius of $\tilde{\Theta}_{res}$, that is, the maximum absolute eigenvalue of $\tilde{\Theta}_{res}$. The last step of the initialization process of an ESN is to specify the sparsity parameter, $\beta \in (0, 1)$, which governs the proportion of elements of the reservoir weight matrix to be non-zero and then randomly setting the remaining proportion of the elements in Θ_{res} to zero. The spectral radius scaling in Equation (2.52) is performed in order to obtain the Echo State property (ESP), also referred to the asymptotic stability property [5, 6, 49].

2.3.1 Echo State Property

The ESP is an important property of the ESN which states that the reservoir will asymptotically remove the impact from initial conditions. To define the ESP, as according to [50], first let $\mathcal{X}$ be the compact set of observed data, $\mathbf{x}(t) \in \mathcal{X} \subset \mathbb{R}^N$ for all $t \in \mathbb{Z}$ and let $\mathcal{U}$ be the compact set of intermediate states, $\mathbf{u}(t) \in \mathcal{U} \subset \mathbb{R}^D$. Moreover let $A : \mathcal{X} \times \mathcal{U} \longrightarrow \mathcal{U}$ be such that

$$\mathbf{u}(t) = A(\mathbf{x}(t), \mathbf{u}(t-1)) = \alpha(\Theta_{in}\mathbf{x}(t) + \Theta_{res}\mathbf{u}(t-1)). \tag{2.53}$$

The compactness of $\mathcal{U}$ is guaranteed when the activation function, $\alpha(\cdot)$, is bounded such as tanh-function [50]. In practice, the input will also be bounded hence the compactness of $\mathcal{X}$ is assumed to hold. Then, let $\mathcal{U}^\infty := \{\mathbf{u}^\infty = (\mathbf{u}(0), \mathbf{u}(1), \dots) | \mathbf{u}(t) \in \mathcal{U}, \text{ for all } t \geq 0\}$ and $\mathcal{X}^\infty := \{\mathbf{x}^\infty = (\mathbf{x}(0), \mathbf{x}(1), \dots) | \mathbf{x}(t) \in \mathcal{X}, \text{ for all } t \geq 0\}$ be the positive infinite state and input sets respectively. Then, with the use of Definition 2.1, the ESP is defined as in Definition 2.2, as according to [50].

Definition 2.1. *If* $\mathbf{u}(t) = A(\mathbf{x}(t), \mathbf{u}(t-1))$ *for all* $t \geq 0$*, then* $\mathbf{u}^\infty$ *is compatible with* $\mathbf{x}^\infty$*.*

Definition 2.2. *The mapping* $A : \mathcal{X} \times \mathcal{U} \longrightarrow \mathcal{U}$ *as given in Equation* (2.53) *possess the Echo State property with respect to* $\mathcal{X}$ *if and only if there exists a null sequence* $(\delta_t)_{t\geq 0}$ *such that for all* $\mathbf{u}^\infty$*,* $\mathbf{y}^\infty \in \mathcal{U}^\infty$ *that are compatible with all* $\mathbf{x}^\infty \in \mathcal{X}^\infty$*, it holds that* $||\mathbf{u}(t) - \mathbf{y}(t)|| \leq \delta_t$ *for all* $t \geq 0$*.*

In Definition 2.2, the null sequence $(\delta_t)_{t\geq 0}$ is referred to a sequence such that for every $\epsilon > 0$, there exists some $N \in \mathbb{N}$ such that $|\delta_t| < \epsilon$ when $t \geq N$ [51]. By scaling the initialized weight matrix of the reservoir as in Equation (2.52) such that $|\lambda_{\max}(\Theta_{res})| < 1$, the ESN satisfies the ESP, as according to [50] and [52].

2.3.2 Long-Short Echo State Networks

To further improve the performance of ESNs, [6] proposed an ensemble of ESNs called Long-Short Echo State Networks (LS-ESNs). Specifically, the LS-ESN consist of three different ESNs, a long ESN, a short ESN and a traditional ESN. Each with different recurrent connections, such that each of them process different time steps of the intermediate activation states. This demonstrated to yield improved prediction performance when tested on various data of both synthetically generated and real-world applications [6]. The long ESN aims to capture long-time dependencies through skip connections of length k. Mathematically, the intermediate activation states for the long ESN, $\mathbf{u}_{long}(t)$, is therefore defined as

$$\mathbf{u}_{long}(t) = \gamma \cdot \alpha_{long}(\Theta_{in}\mathbf{x}(t) + \Theta_{res}\mathbf{u}_{long}(t-k)) + (1-\gamma)\mathbf{u}_{long}(t-k), \tag{2.54}$$

where k is the length of the skip connection, $\mathbf{x}(t) \in \mathbb{R}^N$ is the time-sequential input process for $t \in [0, T]$, γ is the leakage rate, $\Theta_{in} \in \mathbb{R}^{D\times N}$, $\Theta_{res} \in \mathbb{R}^{D\times D}$ is the input and reservoir weight matrices respectively and α_{long} being the corresponding

activation function. The short ESN aims to capture the short-term dependencies by only considering the m last historical states [6]. The intermediate activation states for the short ESN, $\mathbf{u}_{short}(t)$, is defined as

$$\mathbf{u}_{short}(t) = \gamma \cdot \alpha_{short}(\Theta_{in}\mathbf{x}(t) + \Theta_{res}\mathbf{u}_{short}(t-1)) + (1-\gamma)\mathbf{u}_{short}(t-1), \quad (2.55)$$

where $i \in [t-m, t]$ is the m historical time steps and α_{short} is the corresponding activation function. The intermediate activation states from the traditional ESN, long ESN and short ESN, given by Equation (2.51), (2.54) and (2.55) respectively, are then concatenated as $\mathbf{U}(t) = (\mathbf{u}(t), \mathbf{u}_{long}(t), \mathbf{u}_{short}(t)) \in R^{3D}$ of which then is being fed into the readout layer $r(\mathbf{U}(t), \Theta_r) = \alpha_r(\Theta_r\mathbf{U}(t) + \mathbf{b}_r)$ where α_r is the activation function, $\Theta_r \in \mathbb{R}^{D_{out} \times 3D}$ is the weight matrix and $\mathbf{b}_r \in \mathbb{R}^{D_{out}}$ is the bias vector. A schematic illustration of the LS-ESN model is shown in figure 2.6.

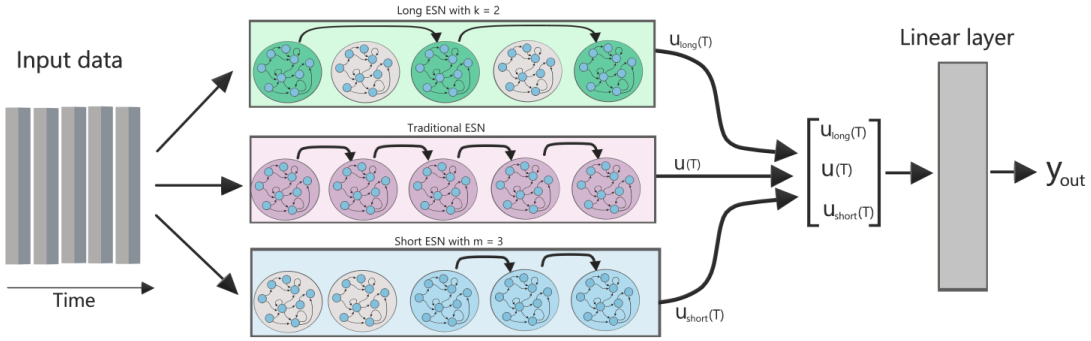


Figure 2.6: *Schematic illustration of a LS-ESN model with $k = 2$ and $m = 3$ for which time-sequential input data is fed into each of the three ESNs whose final intermediate states, $\mathbf{u}_{long}(T)$, $\mathbf{u}(T)$ and $\mathbf{u}_{short}(T)$, are concatenated before being fed into the readout layer, set to be a linear layer, and producing y_{out}.*

2.4 Evaluation metrics

In order to measure how well the models perform, appropriate evaluation metrics need to be defined. For the task of forecasting, an important property to evaluate is the accuracy of the forecast predictions. That is, how close the predictions are compared to the true target values [53]. For accuracy, a common evaluation metric in forecasting tasks is the Root Mean Square Error (RMSE), between the predicted output and the true value. For D number of predictions, $\hat{\boldsymbol{\mu}}(t) = \{\hat{\mu}^{(d)}(t)\}_{d=1}^{D}$ of D number of target values, $\mathbf{y}(t) = \{y^{(d)}(t)\}_{d=1}^{D}$, the RMSE is defined as

$$\text{RMSE}(\hat{\boldsymbol{\mu}}(t), \mathbf{y}(t)) = \sqrt{\frac{1}{D}\sum_{d=1}^{D}(\hat{\mu}(t)^{(d)} - \mathbf{y}^{(d)}(t))^2}. \quad (2.56)$$

Another evaluation metric that is used to measure the accuracy in forecasting tasks in machine learning is the coefficient of determination (R^2) metric [53]. The R^2

metric measures the square of correlation between the predictions and the target values. Mathematically, R^2 is defined as

$$R^2(\mathbf{y}(t), \hat{\boldsymbol{\mu}}(t)) = 1 - \sum_{d=1}^{D} \frac{(y^{(d)}(t) - \hat{\mu}^{(d)}(t))^2}{(y^{(d)}(t) - \bar{y}(t))^2}, \tag{2.57}$$

where $\bar{y}(t)$ is the mean of all the true values in $\mathbf{y}$. Thus, $R^2 \in [0, 1]$ where $R^2 = 0$ refers to zero correlation between the predicted and true values and $R^2 = 1$ refers to perfect correlation.

Moreover, the performance of a model can also be evaluated based on how well-calibrated it is. As according to [54], a forecast model is well-calibrated if for any $p \in [0, 1]$ then

$$\sum_{d=1}^{D} \frac{\mathbb{I}\{y^{(d)}(t) \leq F^{-1}(p; \hat{\mu}^{(d)}(t), \hat{\sigma}^{2(d)}(t))\}}{D} \longrightarrow p, \quad \text{as} \quad D \longrightarrow \infty, \tag{2.58}$$

where D is the number of prediction and target values, $\mathbb{I}$ is the indicator function and F^{-1} is the Gaussian quantile function. The left term in Equation (2.58) is, as according to [9], referred to as the empirical coverage probability denoted by $E(p, \mathbf{y}(t), \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\sigma}}^2(t))$. For cases when having uneven data distribution, using the empirical coverage probability as a calibration error may be inaccurate since the quantile deviation with higher confidence is more important than with lower confidence. Therefore, [9] proposed the confidence-weighted calibration error (CWCE) to approach this issue, defined as

$$\text{CWCE}(\mathbf{y}(t), \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\sigma}}^2(t)) = \sum_{i=1}^{n} p_i \cdot |E(p_i, \mathbf{y}(t), \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\sigma}}^2(t)) - p_i|, \tag{2.59}$$

where $p_i \in [0, 1]$ for all $i \in \{1, \cdots, n\}$. Furthermore, in this thesis, the aim is to minimize the prediction predictive variance, $\hat{\boldsymbol{\sigma}}^2(t) = \{\hat{\sigma}^{2(d)}(t)\}_{d=1}^{D}$, while still covering the true values, $\mathbf{y}(t)$. This is referred to as the concentration of the predictive distribution and is a measure of the forecasting sharpness, as according to [9]. Howver, the accuracy and calibration have often been evaluated seperately, but [9] proposed a combined metric for evaluation of accuracy and calibration, called R-CWCE. The R-CWCE metric is defined, as according to [9], as

$$\text{R-CWCE}(\mathbf{y}(t), \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\sigma}}^2(t)) = \text{CWCE}(\mathbf{y}(t), \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\sigma}}^2(t)) \cdot \sum_{d=1}^{D} \frac{(y^{(d)}(t) - \hat{\mu}(t))^2}{(y^{(i)}(t) - \bar{y}(t))^2}. \tag{2.60}$$

3
Methods

This chapter presents the methodology used in this thesis, where the first section describes the architecture of the models as well as presenting how the data is processed in these models. The second section presents what experiments that were conducted in this thesis and how they were executed. The results of these experiments are presented in Chapter 4 and discussed in Chapter 5.

3.1 Model architectures

Before describing which experiments were conducted and how they were performed, the framework used for these experiments needs to be defined. In this thesis, three main models are used. This section will present the architecture of each model. The models were implemented using the *PyTorch* framework in Python and the training procedure was done using the Adam optimizer[1].

3.1.1 LSTM-SDE Model Architecture

The first model is a latent variable neural SDE with a LSTM network as latent variable mapping. This model is denoted LSTM-SDE and is illustrated in Figure 3.1. For the LSTM-SDE model, the observed time-sequential data of dimension $D_{obs} = 1$ is fed to a single layer LSTM network of dimension D_{LSTM}, meaning that the observed data is mapped from $\mathbb{R}^{D_{obs}}$ to $\mathbb{R}^{D_{LSTM}}$ inside the LSTM network. The mapped observed data is then mapped to the initial latent variable, $\mathbf{z}_0 \in \mathbb{R}^{D_{lat}}$, through a linear layer that maps $\mathbb{R}^{D_{LSTM}}$ to $\mathbb{R}^{D_{lat}}$. The initial latent variable is then fed to a latent variable neural SDE framework as presented in Section 2.2.4, referred to as a SDE block. The SDE block generates N latent variable paths of the form $\{\mathbf{z}_0, \ldots, \mathbf{z}_{N_{sde}}\}$, where $\mathbf{z}(0) = \mathbf{z}_0$, and N_{sde} denotes the number of latent variables in each latent variable path. This is done using the EM method, as presented in Section 2.1.3, with the drift and diffusion networks f^* and g^*. Each of these networks consists of a two-layer neural network where the first layer maps $\mathbb{R}^{D_{lat}}$ to $\mathbb{R}^{2D_{lat}}$ and the second one maps $\mathbb{R}^{2D_{lat}}$ back to $\mathbb{R}^{D_{lat}}$. The activation function for both f^* and g^* is the tanh-function.

Each latent path is generated with a discretization step size $\Delta t_{sde} = T_{sde}/N_{sde}$ for some $T_{sde} > 0$. Consider the case where the number of latent variables in each path, N_{sde}, is greater than the number of target values, T_{targ}. To match the number

[1]For further information of the *PyTorch* framework, the reader is referred to [55].

of latent variables with the number of target values, every k_{targ}:th latent variable is stored into a new vector of latent variables where $k_{targ} = N_{sde}/T_{targ}$. Hence, a vector of the values $\{\mathbf{z}_{k_{targ}}, \mathbf{z}_{2k_{targ}}, \ldots, \mathbf{z}_{T_{targ}k_{targ}}\}$ from each latent variable path is obtained. Each of these vectors are then fed to a layer that maps from D_{lat} to $D_{targ} = 1$, referred to as the target path layer. The target path layer produces a target path $\{\hat{y}(1)_{(i)}, \hat{y}(2)_{(i)}, \ldots, \hat{y}(T_{targ})_{(i)}\}$ for $i = 1, \ldots, N$. The activation function for this target path layer is the tanh-function. From these target paths, the estimates, $\hat{\mu}(t)$ and $\hat{\sigma}^2(t)$, are computed for each $t \in \{1, \ldots, T_{targ}\}$.

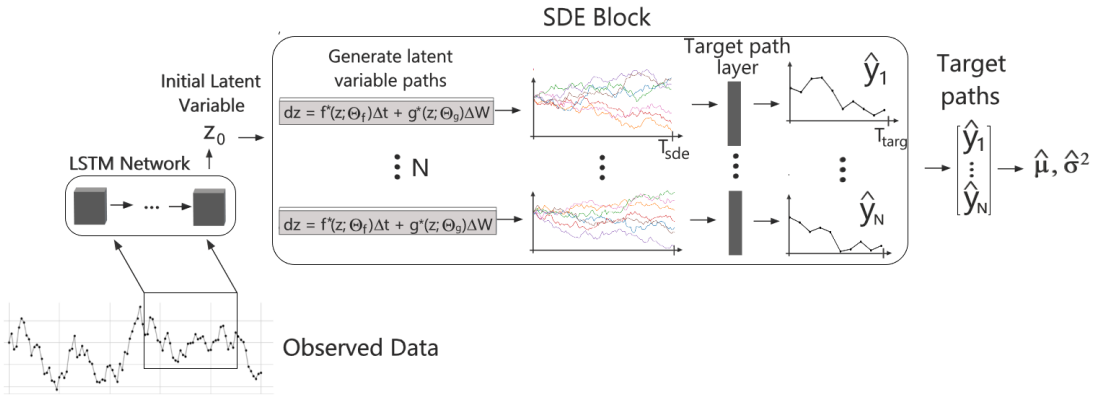


Figure 3.1: *Graphical illustration of the LSTM-SDE model.*

3.1.2 ESN-SDE Model Architecture

The second model (used in this thesis) is a latent variable neural SDE model with an Echo State network as latent variable mapping. This model is denoted ESN-SDE and is illustrated in Figure 3.2. In the ESN-SDE model, the observed data is fed into a traditional ESN whose reservoir is of dimension D_{res}. The last intermediate activation state of the ESN is then fed into a linear layer that maps from $\mathbb{R}^{D_{res}}$ to $\mathbb{R}^{D_{lat}}$ from which the initial latent variable, $\mathbf{z}_0$, is produced. The initial latent variable is then fed into the SDE block similar to the LSTM-SDE model from which the predictive mean and variance are computed.

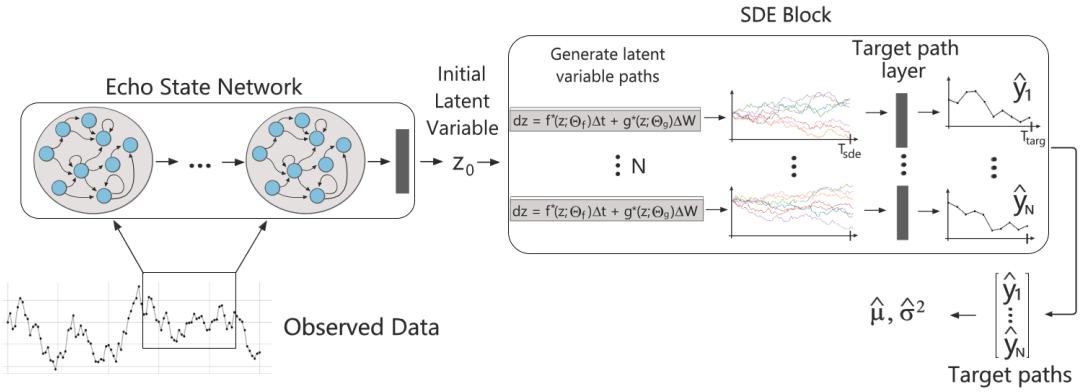


Figure 3.2: *Graphical illustration of the ESN-SDE model.*

3.1.3 LS-ESN-SDE Model Architecture

The third and last model used is similar to the ESN-SDE model in the previous section, but is instead using a Long-Short ESN as latent variable mapping. This model is denoted LS-ESN-SDE and is illustrated in Figure 3.3. The information flow for the LS-ESN-SDE model is similar to the ESN-SDE model.

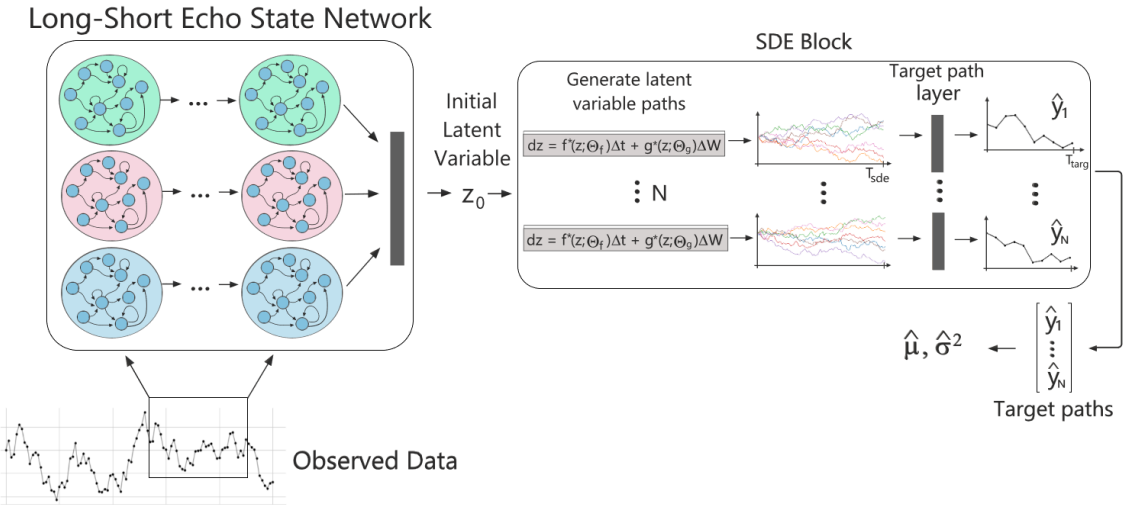


Figure 3.3: *Graphical illustration of the LS-ESN-SDE model.*

3.2 Experiments

This section describes the experiments that were conducted in this thesis, what data was used and the configuration of the models. The results for each of the experiments is presented in Chapter 4.

3.2.1 Conditional Distribution Approximation

By recalling that the models presented in Section 3.1, outputs estimates the predictive mean and the predictive variance under a Gaussian-assumption. It is therefore motivated to investigate how good these estimates actually approximate the true conditional mean and conditional variance of some data with known conditional distribution. Therefore, given some data with known solutions, the first experiment conducted, aimed towards studying the quality of the model approximations of conditional distributions compared to the true conditional distribution. For comparison, each of the models LSTM-SDE, ESN-SDE and LS-ESN-SDE was compared to the Cubature Integration Sigma-Point Approximation (CISPA) method presented in Section 2.2.1, as well as the Free-Energy Approximation (FEA) method presented in Section 2.2.2. Furthermore, the quality of the approximations for each of the models was measured based on the Jensen-Shannon Distance (JSD) between the true solution and the approximations. This was implemented using the *SciPy* package in Python. Moreover, the data used for this experiment was a Geometric Brownian Motion (GBM). The reason for using a GBM is its common usage in stock price modeling, as mentioned in Section 2.1.2. Hence, for this thesis, this process is of particular interest. Moreover, since if the approximations of the conditional mean and conditional variance are poor only for one step, then they will most likely also be poor for larger number of time steps. Therefore, the approximations of the conditional distribution were only performed for a single consecutive time step.

The parameters used for the GBM were $\mu = 0.05$ and $\sigma = 0.5$ with an initial value of $x(0) = 0.5$. The process was generated with $n \in \{1, \cdots, 100\}$ data points with a discretization step size of $\Delta t = 0.1$. The LSTM-SDE, ESN-SDE, LS-ESN-SDE and FEA models were trained on the first 99 values and tested at $x(100)$. The data used is shown in Figure 3.4.

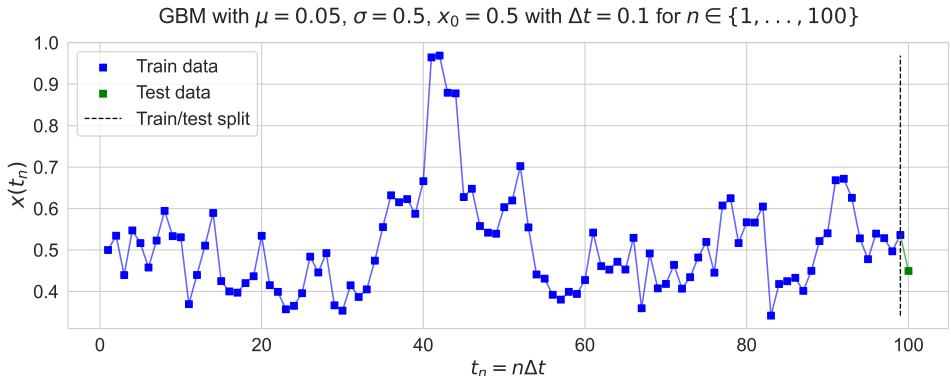


Figure 3.4: *Geometric Brownian motion with* $\mu = 0.05$, $\sigma = 0.5$, $x_0 = 0.5$, $\Delta t = 0.1$, $n \in \{0, \ldots, 100\}$.

The approximations were compared to the true conditional distribution of the GBM

which, by recalling to Section 2.1.3, are

$$\mathbb{E}[x(t+\Delta t)|x(t)] = x(t)e^{(\mu+\frac{\sigma^2}{2})\Delta t}$$
$$\mathrm{Var}[x(t+\Delta t)|x(t)] = \left(x(t)e^{(\mu+\frac{\sigma^2}{2})\Delta t}\right)^2 \left(e^{\sigma^2\Delta t}-1\right),$$

where in this case, $t = 99$ and $\Delta t = 1$. The hyperparameters used for the LSTM-SDE, ESN-SDE and LS-ESN-SDE are shown in Appendix A.1.

For the CISPA method, following the notations as presented in Section 2.2.1, the sigma-points and corresponding weights were

$$\xi_i = \pm 1, \quad W^{(i)} = \frac{1}{2}, \quad i \in \{1,2\}. \tag{3.1}$$

This implied the following equations for the conditional mean and conditional variance

$$\frac{\mathrm{d}m(t|s)}{\mathrm{d}t} = \frac{1}{2}\left(m(t|s) - \sqrt{S(t|s)}\right)\mu + \frac{1}{2}\left(m(t|s) + \sqrt{S(t|s)}\right)\mu, \tag{3.2}$$

and

$$\begin{aligned}\frac{\mathrm{d}S(t|s)}{\mathrm{d}t} &= \sqrt{S(t|s)}\left(m(t|s) - \sqrt{S(t|s)}\right)\mu - \sqrt{S(t|s)}\left(m(t|s) + \sqrt{S(t|s)}\right)\mu \\ &+ \frac{1}{2}\left((m(t|s) - \sqrt{S(t|s)})\sigma\right) + \frac{1}{2}\left((m(t|s) + \sqrt{S(t|s)})\sigma\right),\end{aligned} \tag{3.3}$$

with the initial conditions

$$m(0) = x(s), \quad S(0) = 0. \tag{3.4}$$

These were solved using Euler's method for ODEs with $x(t) = x(99) = 0.4495$ as starting point and a step size of $\Delta t_S = 10^{-5}$.

For the FEA method, following the notations as presented in Section 2.2.2, Σ was set to be equal to σ^2 and the drift of the hidden process was set to be the same as the drift for the GBM. That is, $\mathrm{d}z(t_n) = \mu z(t_n)\mathrm{d}t + \sigma^2 \mathrm{d}W(t_n)$, with $x(t_n) \sim \mathcal{N}(z(t_n), r^2)$, where the noise parameter was set to $r = 0.1$. In this setting, this implies that E_{obs} and E_{sde} evaluate into

$$\begin{aligned}E_{sde}(t_n) =& \frac{1}{2}\left\langle f(z(t_n),t_n) - g(z(t_n),t_n))^{\mathrm{T}}\Sigma^{-1}(f(z(t_n),t_n) - g(z(t_n),t_n))\right\rangle_q \\ =& \frac{1}{2\sigma^2}\left\langle(\mu z(t_n) + A(t_n)z(t_n) - b(t_n))^2\right\rangle_q \\ =& \frac{(A(t_n)+\mu)^2}{2\sigma^2}\langle z(t_n)^2\rangle_q - \frac{(A(t_n)+\mu)b(t_n)}{\sigma^2}\langle z(t_n)\rangle_q + \frac{b(t_n)^2}{2\sigma^2} \\ =& \frac{(A(t_n)+\mu)^2}{2\sigma^2}(m(t_n)^2 + S(t_n)) - \frac{(A(t_n)+\mu)b(t_n)m(t_n)}{\sigma^2} + \frac{b(t_n)^2}{2\sigma^2},\end{aligned}$$

and

$$\begin{aligned}E_{obs}(t_n) =& \frac{1}{2}\left\langle(x(t_n) - z(t_n))^{\mathrm{T}} r^{-2}(x(t_n) - z(t_n))\right\rangle_q + C \\ =& \frac{1}{2r^2}\langle(x(t_n) - z(t_n))^2\rangle_q + C \\ =& \frac{1}{2r^2}\left((m(t_n)^2 + S(t_n)) - 2x(t_n)m(t_n) + x(t_n)^2\right) + C.\end{aligned}$$

In the above evaluations, the fact that $\langle z(t_n)^2 \rangle_q = m(t_n)^2 + S(t_n)$ is used and C is a constant. This yields the following gradients of E_{obs} and E_{sde} as

$$\frac{\mathrm{d}E_{sde}}{\mathrm{d}A}(t_n) = \frac{A(t_n)+\mu}{\sigma^2}(m(t_n)^2 + S(t_n)) - \frac{b(t_n)m(t_n)}{\sigma^2} \tag{3.5}$$

$$\frac{\mathrm{d}E_{sde}}{\mathrm{d}b}(t_n) = \frac{b(t_n)}{\sigma^2} - \frac{(A(t_n)+\mu)m(t_n)}{\sigma^2} \tag{3.6}$$

$$\frac{\mathrm{d}E_{sde}}{\mathrm{d}m}(t_n) = \frac{(A(t_n)+\mu)^2 m(t_n))}{\sigma^2} - \frac{(A(t_n)+\mu)b(t_n)}{\sigma^2} \tag{3.7}$$

$$\frac{\mathrm{d}E_{sde}}{\mathrm{d}S}(t_n) = \frac{(A(t_n)+\mu)^2}{2\sigma^2} \tag{3.8}$$

$$\frac{\mathrm{d}E_{obs}}{\mathrm{d}m}(t_n) = \frac{m(t_n)}{r^2} - \frac{x(t)}{r^2} \tag{3.9}$$

$$\frac{\mathrm{d}E_{obs}}{\mathrm{d}S}(t_n) = \frac{1}{2r^2}. \tag{3.10}$$

These were implemented in the FEA algorithm described in Section 2.2.2. Moreover, the initial conditions were set to be $m(0) = 0$ and $S(0) = 0$. The gradient step size was set to $\eta = 0.1$ and the algorithm was iterated for 100 iterations. Furthermore, the number of discretization steps was set to $K = 10^5$ such that the discretization step was $\Delta t_{fea} = T/K = 10/10^5 = 10^{-4}$. The parameters of the FEA method was updated on the training data and was then tested on the test data in the Figure 3.4, as according to the algorithm described in Section 2.2.2.

After having approximated the conditional mean and the conditional variance, and obtained a predictive mean and predictive variance from the models at $x(s)$, the approximated densities were sampled by generating 10^5 Gaussian random variables with the approximated parameters. The same was done for the true distribution with the known mean and variance. Finally, the JSD was then computed using the *SciPy* package in Python on these generated random variables. The results of the transition density approximation are given in Section 4.1

3.2.2 Forecasting

Since the aim of this thesis was to use the models to predict future data values based on historical data values, the models' ability to do such forecasts were investigated. Hence, the final experiments conducted was to from some observed data. For this purpose, three different sets of data were used. For the first data set, referred to as the trivial test case, the data was generated from a GBM. For the last two data sets, real-world stock price data was used. These are therefore referred to as the real-world test cases. The motivation for performing experiments on the trivial data is to investigate the models' ability to forecast longer time steps ahead of a GBM and see if the uncertainty of the models' coincide with the uncertainty of the GBM. For the real-world test cases, a set of 100 different stocks was used, all from the same industry class of the Global Industry Classification Standard (GICS), namely the class of 1510. This data was provided by the company. The aim of the first experiment using the real-world data was to investigate the forecasting performance

on a sample stock that was randomly selected among the 100 stocks that were available. The aim of the second experiment using the real-world data used all of the 100 available stocks. The reason for using all of the stocks was to obtain a more general investigation of the forecasting performance of the models as the performance may vary a lot depending on what stock is being investigated.

3.2.2.1 Trivial test case

For the trivial test case, the data used was a GBM with parameters $\mu = 0.01$, $\sigma = 0.1$ and with the initial value of $x_0 = 0.1$. It was further generated for $n \in \{1, \ldots, 5000\}$ with discretization step of $\Delta t = 0.1$. Moreover, the data was divided into a training and test set using a 80/20 split ratio. The data of the GBM is shown in Figure 3.5.

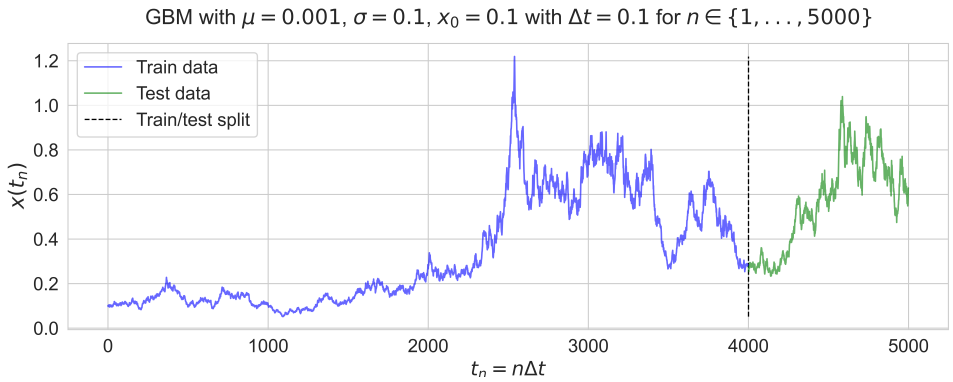


Figure 3.5: *Data of the GBM used for the trivial test case for $t_n = n\Delta t$ where $n \in \{1, \ldots, 5000\}$ and $\Delta t = 0.1$ and divided into a train set (blue) and a test set (green) with a 80/20 split ratio.*

The forecasting was performed using 10 historical time steps and 10 steps ahead were predicted.

Each model was trained on the train set in three different training periods, each with decreased learning rate. This was done in order to avoid learning plateaus and getting stuck in local minima. For each of the models, the Adam optimizer was used with a weight decay of 10^{-3}. The loss function used was the NLL loss function as described in Section 2.2.4.1. The hyperparameters used for the LSTM-SDE, ESN-SDE and the LS-ESN-SDE models for are given Appendix A.2.1. Furthermore, the models was benchmarked against a traditional machine learning model, namely a standard LSTM network. The hyperparameters used for the standard LSTM for each test case is given in Appendix A.2.1. The evaluation metrics used for measuring the forecast performance was the Root Mean Squared Error (RMSE) and the coefficient of determination (R^2) for measuring the accuracy of predictions as well as the R-CWCE for a combined evaluation measure of the accuracy and the calibration of the models. These metrics were previously presented in Section 2.4.

3.2.2.2 Real-world test cases

To investigate how the models perform in a real-world scenario, data over stock prices from 100 different stocks was used. These were randomly selected with the criteria that the time period of available data was the same for each stock. In this experiment, daily stock prices in an approximately 10 year time period from $2012-03-01$ to $2022-02-18$ were used, resulting in 2551 time steps. The list of each stock used is listed in Appendix B.

For the first real-world test case, the forecasting performance of the stock prices from a single stock was investigated using each of the LSTM-SDE, ESN-SDE and LS-ESN-SDE models. This single stock was randomly selected among the 100 available stocks and the stock used was the stock with ticker name *APD*. For this experiment, predictions were made for 10 time steps ahead and using 10 historical time steps. Moreover, the data was normalized as

$$\text{Norm}(x) = \frac{x - \min(x)}{\max(x) - \min(x)}, \quad x \in \mathbb{R}, \tag{3.11}$$

which was performed using the MinMaxScaler module in the *Scikit-learn* package in Python. After normalization, the data was also divided into a 80/20 train/test ratio split. The data over the stock price for the *APD* stock used in the first real-world test case is shown in Figure 3.6.

Figure 3.6: *Stock prices of the APD stock for* 2551 *days of data in the time period 2012-01-03 to 2022-02-18 used for the single stock test case of real-world data with a* 80/20 *split of the train (blue) and test (green) data.*

Moreover, similarly as for the trivial test case, the training procedure for each of the LSTM-SDE, ESN-SDE and LS-ESN-SDE models consisted of three training periods, each with a decreased learning rate. Furthermore, the Adam optimizer was used with a weight decay of 10^{-3} and the loss function was the NLL function described in Section 2.2.4.1. The hyperparameters used for the LSTM-SDE, ESN-SDE and the LS-ESN-SDE models for each test case are given Appendix A.2.2. For benchmarking, both a traditional LSTM and a GBM process were used. For

the GBM method, the parameters were estimated as described in Section 2.1.2 in Equation (2.10), as

$$\hat{\mu}^{(d)} = \sum_{i=0}^{s-1} \frac{S^{(d)}(i+1) - S^{(d)}(i)}{S^{(d)}(i)}$$
$$\hat{\sigma}^{(d)} = \sqrt{\frac{1}{s-1} \sum_{i=0}^{s-1} \left(\frac{S^{(d)}(i+1) - S^{(d)}(i)}{S^{(d)}(i)} - \hat{\mu}^{(d)} \right)^2},$$

where s denotes the number of historical time steps, that is $s = 10$, and $S(i)$ denotes the (normalized) stock price at each time $i \in \{0, \ldots, s\}$. After the parameters had been estimated, 100 simulated paths of the GBM were produced using the EM method described in Section 2.1.3. The step size of the discretization was set to 10^{-5} for predicting 10 days ahead and thus resulting in each simulated path having 10^6 number of data points. The predictions were then set as the mean of the final values from all of the 100 simulated paths, where the variance was also computed.

As previously mentioned, for the second experiment for the real-world test cases, data over the stock prices from all of the 100 stocks were used. The procedure was that for each of the LSTM-SDE, ESN-SDE and LS-ESN-SDE models, data from all of the stocks for each time period was used in each sample. More specifically, by letting $\{\mathbf{S}(t-s)\}_{t \in \mathcal{T}}$ denote the historical stock prices for each stock such that $\mathbf{S}(t) \in \mathbb{R}^{100}$ for each historical time period $\{t-s, \ldots, t\}$, for $t \in \mathcal{T}$ and $\mathcal{T} = \{s, s+1, \ldots, 2551\}$ where $s \in \mathbb{N}$ denotes the number of historical time steps used. Hence, each sample used in the train and test sets consisted of data from all of the stocks in that the current time period. Thus, resulting in one model trained on all of the 100 stocks, for each of the LSTM-SDE, ESN-SDE and LS-ESN-SDE models respectively. The reason for training the models on the data from all of the stocks simultaneously is that in practice, having one model per stock is highly insufficient as it requires much memory to store the trained models. Hence, it is often desired to have one model for a large set of data. Therefore, motivated by practical purposes, the models were trained and evaluated on data from all of the stocks simultaneously.

Moreover, to also investigate the models' ability to forecast longer time periods, each of the models were trained and tested for different time step predictions. More specifically, the models were investigated in predicting 10, 20, 30 and 40 time steps ahead using 10, 20, 30 and 40 historical time steps respectively. The models were evaluated using the mean of the RMSE scores for each stock. Since the value of the stock price can vary a lot between stocks, the RMSE scores were computed from evaluating on the normalized test data. Moreover, to analyze how the forecasting performance changed for a growing number of time steps predicted, a linear least squares fit was performed on the RMSE scores between the time steps for each of the model. This provided a relation between the mean RMSE scores and the number of time steps predicted for each model as

$$\text{RMSE}_M(T_{targ}) \approx \alpha + \beta T_{targ},$$

where T_{targ} is the number of predicted time steps and α and β is the residual and coefficient of the linear least square fit. This was implemented using the *optimization*

module in the SciPy package in Python. Similarly to the previous experiment for the single stock, a traditional LSTM model as well as a GBM method was used as benchmark, where the same procedure as in the previous experiment was performed for each stock. Furthermore, parameter configuration for the LSTM-SDE, ESN-SDE and LS-ESN-SDE as well as the LSTM model was also the same as in the single stock experiment. The results of the experiments conducted on the real-world stock data presented above are given in Section 4.2.2.

4

Results

This chapter presents the results of the experiments described in Section 3.2. The analysis and discussion about the results are given in Chapter 5.

4.1 Conditional distribution approximation

The results of the JSD measures for approximating the conditional distribution of the GBM presented in Section 3.2.1 for the LSTM-SDE model compared to the Cubature Integration Sigma-Point Approximation (CISPA) and the Free-Energy Approximation (FEA) method is shown in Figure 4.1. Moreover, the resulting distributions from the approximated parameters for the ESN-SDE model and the LS-ESN-SDE model compared the FEA and CISPA methods are shown in Figure 4.2a, 4.2b and 4.2c respectively.

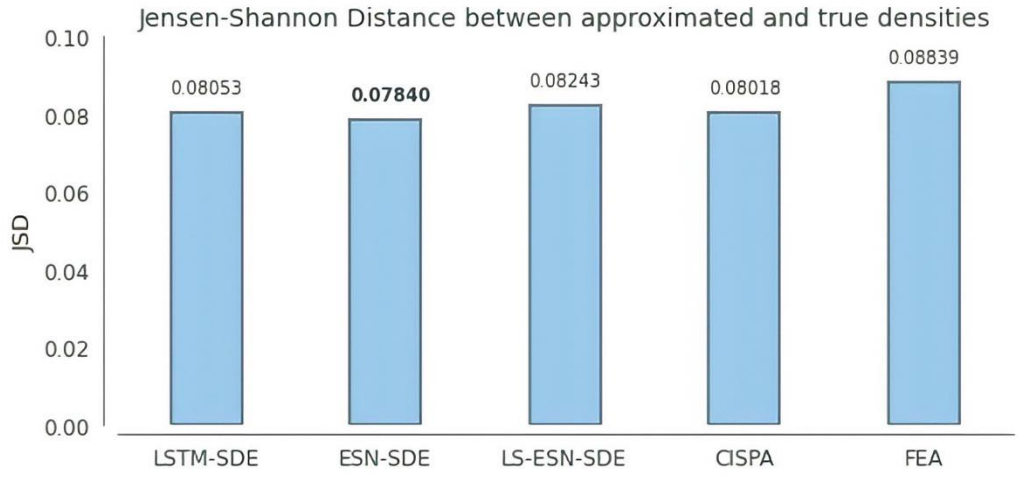
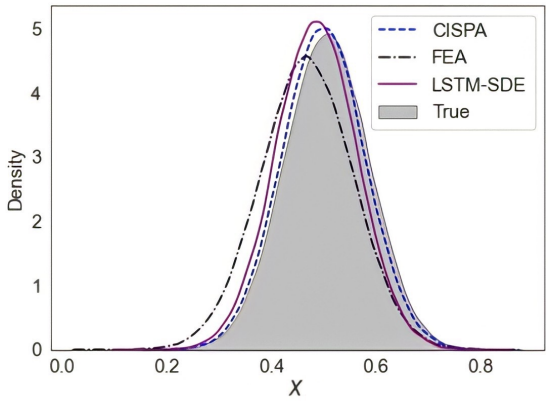
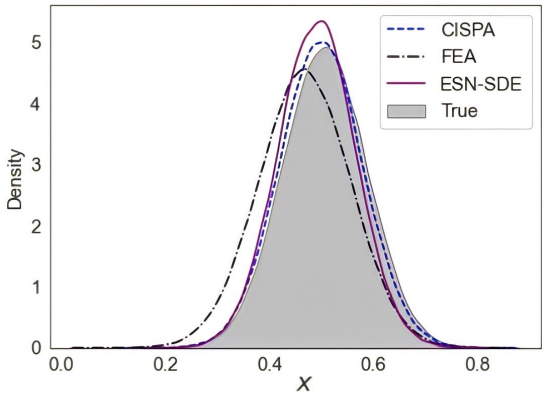


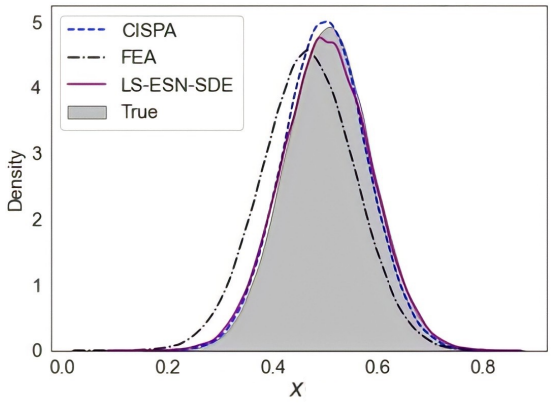
Figure 4.1: *The Jensen-Shannon distances between the approximate and true conditional distributions from generating* 10^5 *Gaussian random variables of the parameters obtained from the LSTM-SDE, ESN-SDE, LS-ESN-SDE, CISPA and FEA methods.*



(a) *Results for the LSTM-SDE model compared to the CISPA and FEA methods.*



(b) *Results for the ESN-SDE model compared to the CISPA and FEA methods.*



(c) *Results for the LS-ESN-SDE model compared to the CISPA and FEA methods.*

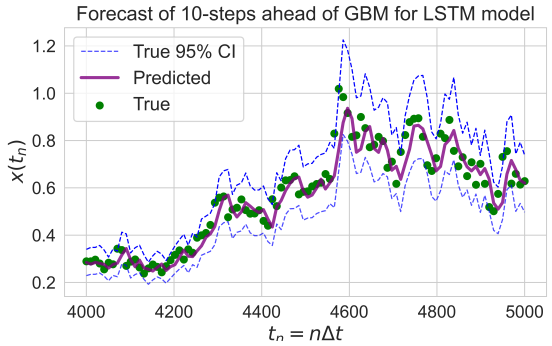
Figure 4.2: *The results of density plots estimated from generating* 10^5 *Gaussian random variables with parameters obtained for the LSTM-SDE model in* **(a)** *ESN-SDE model in* **(b)** and *LS-ESN-SDE model in* **(c)**, *each compared to the CISPA and FEA methods for the GBM.*

4.2 Forecasting

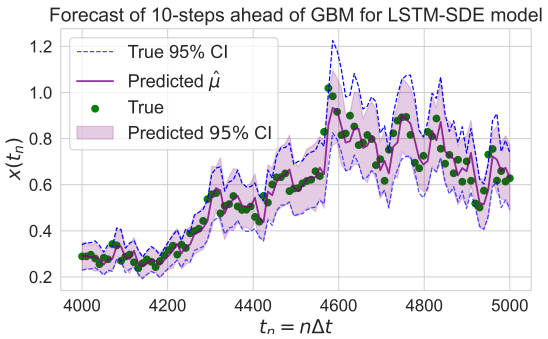
The following section presents the results for the forecasting task described in Section 3.2.2. The results for the trivial test case are first presented, followed by the results on the real-world test cases of the stock data.

4.2.1 Trivial test case

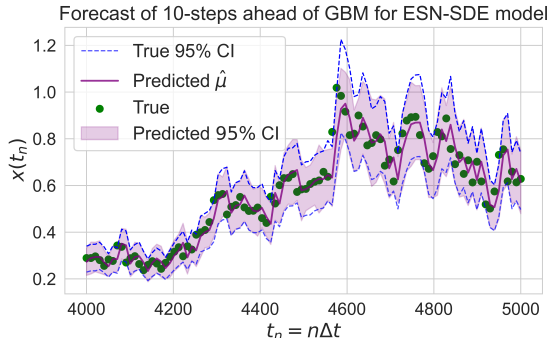
The results of the forecasts for the trivial test case as described in Section 3.2.2.1 are shown in Figure 4.3a, 4.3b, 4.3c and 4.3d for the traditional LSTM, LSTM-SDE, ESN-SDE and the LS-ESN-SDE models respectively.



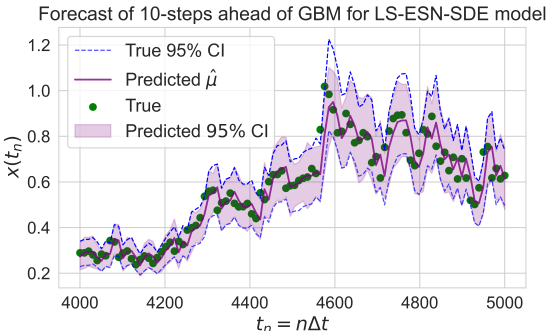
(a) *Predictions for the traditional LSTM model for the trivial test case.*



(b) *Predictions for the LSTM-SDE model for the trivial test case.*



(c) *Predictions for the ESN-SDE model for the trivial case.*



(d) *Predictions for the LS-ESN-SDE model for the trivial test case.*

Figure 4.3: *Predictions of the forecast for the trivial test case when predicting* 10 *time steps ahead and using* 10 *historical time steps for the traditional LSTM model in* **(a)**, *LSTM-SDE model in* **(b)**, *ESN-SDE model in* **(c)** *and the LS-ESN-SDE model in* **(d)**.

Moreover, the corresponding scores of the evaluation metrics for each model of the trivial case are shown in Table 4.1

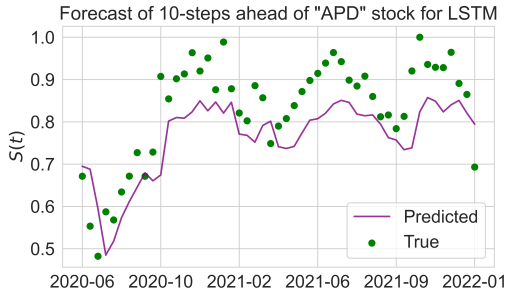
Table 4.1: Scores of the evaluation metrics from the predictions when predicting 10 steps ahead using 10 historical time steps of the trivial test case for the traditional LSTM, LSTM-SDE, ESN-SDE and LS-ESN-SDE model. The best value for each metric is shown in bold.

	LSTM	LSTM-SDE	ESN-SDE	LS-ESN-SDE
RMSE	0.0411	0.0336	0.0329	**0.0320**
R^2	0.9311	0.9475	0.9477	**0.9507**
R-CWCE	-	0.0737	**0.0436**	0.0736

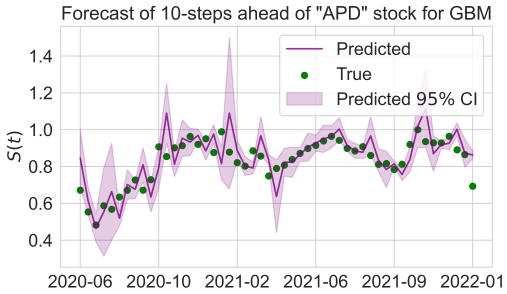
4.2.2 Stock data

The results of forecasting the *APD* stock prices for 10 steps ahead using 10 historical time steps for the traditional LSTM model, GBM method and the LSTM-SDE, ESN-SDE and LS-ESN-SDE models as described in Section 3.2.2.2, are shown in Figure

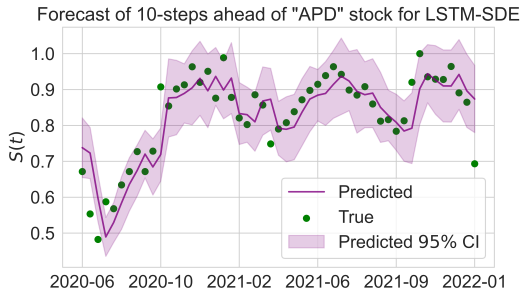
4.4a, 4.4b, 4.4c, 4.4d and 4.4e respectively.



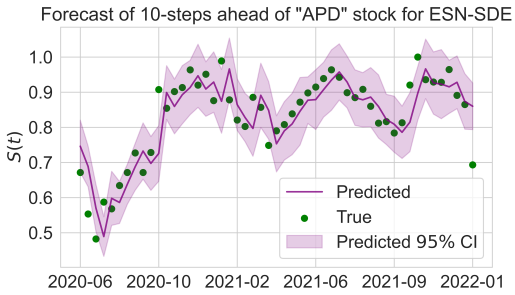
(a) *Forecast results on the APD stock for the traditional LSTM model.*



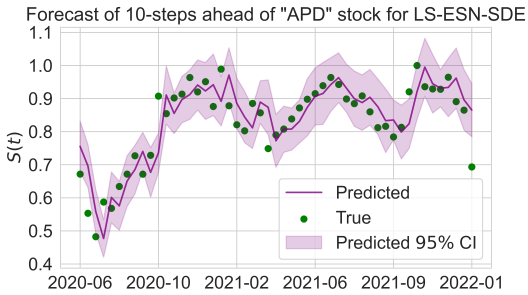
(b) *Forecast results on the APD stock for the GBM method.*



(c) *Forecast results on the APD stock for the traditional LSTM-SDE model.*



(d) *Forecast results on the APD stock for the traditional ESN-SDE model.*



(e) *Forecast results on the APD stock for the traditional LS-ESN-SDE model.*

Figure 4.4: *Forecast results on the APD stock prices when predicting* 10 *time steps ahead using* 10 *historical time steps for the traditional LSTM model* (**a**)*, GBM method* (**b**)*, LSTM-SDE* (**c**)*, ESN-SDE* (**d**) *and the LS-ESN-SDE* (**e**) *models.*

Moreover, the scores of the evaluation metrics for each of the models are shown in Table 4.2.

Table 4.2: Scores of the evaluation metrics from the predictions when predicting 10 steps ahead using 10 historical time steps of APD stock prices for the traditional LSTM model, GBM method and the LSTM-SDE, ESN-SDE and LS-ESN-SDE models. The best value for each metric is shown in bold.

	LSTM	GBM	LSTM-SDE	ESN-SDE	LS-ESN-SDE
RMSE	0.0703	0.0546	0.0402	0.0408	**0.0393**
R^2	0.5460	0.6555	0.7930	0.7983	**0.8103**
R-CWCE	-	1.5429	**1.2972**	2.1392	1.7490

The results of the mean RMSE scores for the LSTM-SDE, ESN-SDE and LS-ESN-SDE models as well as the traditional LSTM and the GBM method when predicting 10, 20, 30 and 40 time steps ahead using 10, 20, 30 and 40 historical time steps respectively for the data from all of the 100 stock prices are shown in Figure 4.5.

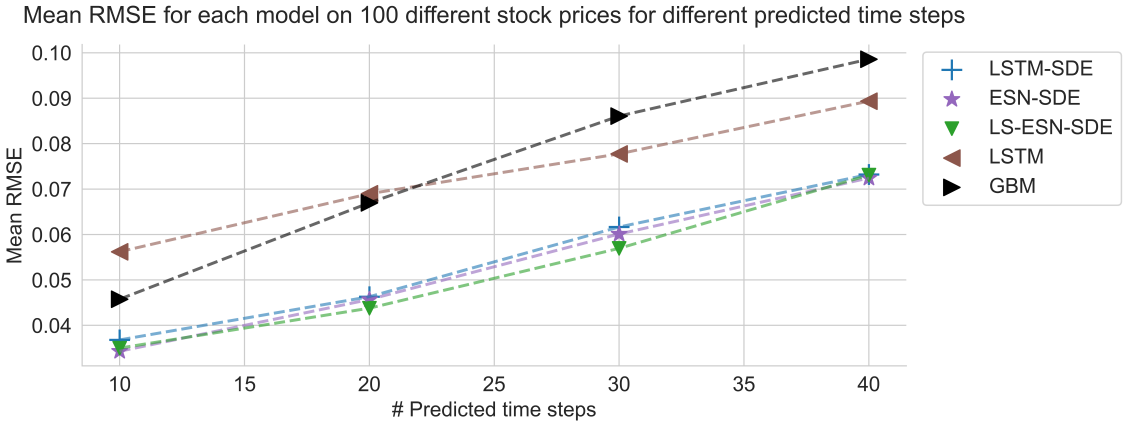


Figure 4.5: *Mean RMSE scores for the LSTM-SDE, ESN-SDE and LS-ESN-SDE model benchmarked against a traditional LSTM model and the GBM method for the* 100 *stocks when predicting* 10*,* 20*,* 30 *and* 40 *time steps ahead and equivalent amount of historical steps respectively.*

The results of the linear least squares of the change in the mean RMSE scores between the number of predicted time steps for the LSTM-SDE, ESN-SDE, LS-ESN-SDE models as well as the traditional LSTM and GBM method is shown in Table 4.3.

Table 4.3: Results of the linear least squares of the change in the mean RMSE scores between the number of predicted time steps of the LSTM-SDE, ESN-SDE, LS-ESN-SDE, traditional LSTM and the GBM model.

	LSTM	GBM	LSTM-SDE	ESN-SDE	LS-ESN-SDE
β	**0.001087**	0.001776	0.001396	0.001408	0.001439
α	0.04601	0.02994	0.01834	0.01693	**0.01485**

5
Discussion

The following chapter presents a discussion about the results from Chapter 4.

5.1 Conditional distribution approximation

The results of the conditional distribution approximation of the GBM described in Section 3.2.1 shows that the LSTM-SDE, ESN-SDE and LS-ESN-SDE achieves good approximations of the parameters of the conditional distribution of the GBM. As shown in Figure 4.2a, the LSTM-SDE appears to be slightly shifted compared to the true distribution whereas the ESN-SDE in Figure 4.2b seems to be more centered around the true conditional mean. Similarly, the LS-ESN-SDE in Figure 4.2c seems to also be centered around the true conditional mean. Moreover, all three of the LSTM-SDE, ESN-SDE and LS-ESN-SDE models obtains a better approximation than the FEA method which appears to be both slightly shifted and have a higher variance than the conditional distribution as it has a lower peak. This could be explained by the construction of the FEA method as the diffusion part of the model is constant. As this is not the case for a GBM, whose diffusion term are dependent on the current value of the process itself, this could explain why the FEA method achieves a worse performance than the other models.

From the Jensen-Shannon distances in Figure 4.1, the ESN-SDE obtains the lowest score of 0.0784. As the peak of the distribution from the parameters of the ESN-model is higher than the peak with the true parameters, it would suggest a higher JSD score compared to the CISPA method. However, the distribution from the parameters of the CISPA method do show a slightly shifted mean than the true solution. As the peak of the distribution from the ESN-SDE model shifts at the top to a higher value, this could indicate that the mean of the ESN-SDE model is actually closer to the true mean. This could explain the lower JSD score for the ESN-SDE model compared to the CISPA method. Moreover, it is also seen that each of the three LSTM-SDE, ESN-SDE and LS-ESN-SDE models' achieved a lower JSD score than the FEA method. This coincide with the result in Figure 4.2a-4.2c. These results indicates that the LSTM-SDE, ESN-SDE and the LS-ESN-SDE obtains better approximations of the true parameters in the conditional distribution of the GBM than the FEA method. They further obtain approximations of the true parameters that are similar to the CISPA. As the conditional distribution is Gaussian, the CISPA method should be able obtain close to exact approximations. This further suggest the same for the LSTM-SDE, ESN-SDE and the LS-ESN-SDE

models.

5.2 Forecasting

The following section discusses the results of the forecasting experiments described in Section 3.2.2 of which the results of the trivial test case is first discussed followed by the discussion of the results on the real-world test cases.

5.2.1 Trivial test case

From the results in Table 4.1, it is shown that the LS-ESN-SDE achieved the lowest RMSE score as well as the lowest R^2 score among the four models'. This suggest that the LS-ESN-SDE is the most accurate in its predictions in forecasting the GBM. By comparing the RMSE scores between the LSTM-SDE, ESN-SDE and LS-ESN-SDE, it shows that the ESN and the LS-ESN provides more accurate predictions than when using a LSTM network and a ESN network for the latent mapping. This coincides with the result from the related studies of the LS-ESN. Moreover, all three of the LSTM-SDE, ESN-SDE and the LS-ESN-SDE achieves better performance in accuracy than the traditional LSTM in both the RMSE and R^2 values. This further suggests that the latent variable neural SDE framework do obtain predictions that are more correlated to the target values than the LSTM model.

It is further shown that the ESN-SDE obtains the lowest R-CWCE value compared to the LSTM-SDE and the LS-ESN-SDE. Since the R-CWCE metric is a measure of both the accuracy and the calibration of the models', this result suggest that the ESN-SDE obtains the best overall description of the data. Furthermore, it is shown in Figure 4.3b-4.3d that each of the LSTM-SDE, ESN-SDE and the LS-ESN-SDE obtain a predictive variance similar to the true conditional variance of the GBM. Comparing this with the result from the previous experiment where only one step ahead was considered. The result here therefore indicates that, even for longer time steps ahead, the models' are able to obtain a relatively close approximation of the true conditional variance. That is, for a GBM.

5.2.2 Stock data

The results of forecasting the stock prices of the *APD* stock, as presented in Section 4.2.2, shows that the traditional LSTM model provide less accurate predictions than the other four models. The LSTM achieves a RMSE of 0.0703 and a R^2 score of 0.5460 as shown in Table 4.2. Compared to the results in forecasting the data from a GBM, where the R^2 score was 0.9081, this is a significant reduction in accuracy. Moreover, the GBM method achieves a better accuracy in predictions than the traditional LSTM, but obtains a vary volatile variance. This could be explained from the data of the stock prices as shown in Figure 3.6 in Section 3.2.2. There, it is seen that time period for when the GBM obtains a very high variance, do coincide with drastic changes in the data of the stock price. Since the parameters of the GBM are estimated based on the 10 most previous time steps, any drastic change in the stock

price will result in a higher diffusion term in the GBM. Therefore, it is expected that the GBM obtains a higher variance in those time intervals. This could further explain the significant higher value for the R-CWCE metric.

For the LSTM-SDE, ESN-SDE and LS-ESN-SDE models, it is shown in Table 4.2 that the LS-ESN-SDE model achieves the lowest RMSE score with a value of 0.0393 compared to 0.0402 and 0.0408 for the LSTM-SDE and ESN-SDE respectively. This imply that these three models' are more accurate in their predictions compared to the traditional LSTM and the GBM. A similary result is shown for the R^2 scores. For the R^2 scores, the LS-ESN-SDE also achieves the best highest value of 0.8103. This is closely followed by the scores of the ESN-SDE and the LSTM-SDE models' of 0.7983 and 0.7930 respectively. This indicates a significantly higher correlation between the target values and their predictions. However, this result is just for the stock prices of a single stock and thus only provide a small indication for their ability to forecast stock prices. Hence, the results from the 100 stock prices needs to be analyzed for a more generalized analysis.

From the results in Figure 4.5, it is shown that the LSTM-SDE, ESN-SDE and the LS-ESN-SDE achieve a lower mean RMSE when forecasting on the stock prices from all of the 100 stocks compared to the traditional LSTM and the GBM model. This also for each of the predicted time steps. However, from the result in Table 4.3 it is shown in that the traditional LSTM model has the smallest rate of change of the mean RMSE scores. Hence, it has the smallest decrease in the accuracy of the predictions for growing number of predicted time steps. However, the traditional LSTM has also the highest residual of all of the models with a value of 0.04601. This implies that the accuracy of its predictions is already much higher compared to the other models. Hence, despite the low change rate of error, it is still the least accurate model among the five.

The model with the lowest residual, as shown in Table 4.3, is the LS-ESN-SDE with a value of 0.01485. This suggest that the use of the LS-ESN in the latent mapping improves the accuracy of the predictions. A similar result as in the trivial test case. Moreover, from the result from the linear least squares fit in Table 4.3, it is seen that the mean RMSE decreases fastest for the GBM model. However, it obtains a lower mean RMSE value for predicting 10 and 20 time steps ahead compared to the traditional LSTM, as seen in Figure 4.5. This suggests that the GBM model is more convenient for shorter forecasting steps. Moreover, as also seen in Figure 4.5, each of the three LSTM-SDE, ESN-SDE and the LS-ESN-SDE achives a lower mean RMSE than the two benchmark models. This indicates that the three models' do obtain more accurate predictions even in a more general case. However, because the number of stocks were limited, further experiments needs to be done in order to say how the models' would perform in a real-world setting.

5.3 Future studies

As the aim of this thesis was to investigate whether the presented methods could be used to forecast stock prices, no new theory has been developed within the scope of this thesis. Therefore, it is motivated to perform deeper studies within the theory of the used models. Specifically, within the framework of neural SDEs as the previous research of this is limited. Furthermore, the models' used for benchmark in forecast in this thesis consisted only of a traditional LSTM model and a GBM model. It is therefore motivated for further studies to compare against more advanced methods for the task of stock price predictions. Moreover, since the focus in this thesis was the application to stock prices, it is also motivated to perform future studies in how these models' can be used in other financial applications such as interest rate modeling and option price modeling. Lastly, since the scope of this thesis is within the financial domain, it would also be interesting for future studies to investigate the use of these models' in other domains and fields. Specifically in areas where stochastic models are used or where data is subject to perturbations.

6
Conclusion

The work conducted in this thesis has investigated the use of three different models for the task forecasting stock prices. The models that were considered consisted of three different types of RNNs, namely, a LSTM, a ESN and a LS-ESN. These were combined with a neural SDE framework to produce several latent variable processes. From each of these latent variable processes, a prediction was produced using an additional neural network. All of these predictions produced from the latent variable processes, was then used to compute estimates of a predictive mean and a predictive variance that were the output of the models. Under the assumption that the observations were Gaussian distributed, these estimated parameters were then used as approximations for the conditional distribution of the data. The conclusions made from this thesis is that each of the three models obtains good approximations for the conditional distribution of a GBM. Moreover, the models also achieve more accurate predictions when the outputs are used for forecasting data from a GBM when compared to a traditional LSTM model. Moreover, they also provide more accurate forecasts when being used on stock prices than both a traditional LSTM and when using a GBM model. Both on a single stock and on 100 stocks. It was further concluded that the LS-ESN did provide more accurate predictions compared to using a LSTM and a ESN. However, further studies need to be made in order to see how they compare to more advanced models to fully investigate their performance.

Bibliography

[1] M. A. Pinsky and S. Karlin, *An Introduction to Stochastic Modeling*, 4th ed. Elsevier Inc., 2011, ch. 1.

[2] S. E. Shreve, *Stochatic Calculus for Finance II, Continous-Time Models.* New York City: Springer, 2008.

[3] J. Sen, R. Sen, and A. Dutta, *Introductory Chapter: Machine Learning in Finance-Emerging Trends and Challenges*, 10 2021.

[4] G. Van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artificial Intelligence Review*, vol. 53, 12 2020.

[5] C. Sun, M. Song, S. Hong, and H. Li, "A review of designs and applications of echo state networks," 2020. [Online]. Available: https://arxiv.org/abs/2012.02974

[6] K. Zheng, B. Qian, S. Li, Y. Xiao, W. Zhuang, and Q. Ma, "Long-short term echo state network for time series prediction," *IEEE Access*, vol. 8, 2020.

[7] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar, and C. Hsieh, "Neural SDE: stabilizing neural ODE networks with stochastic noise," *CoRR*, vol. abs/1906.02355, 2019. [Online]. Available: http://arxiv.org/abs/1906.02355

[8] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *CoRR*, vol. abs/1806.07366, 2018. [Online]. Available: http://arxiv.org/abs/1806.07366

[9] P. Cui, Z. Deng, W. Hu, and J. Zhu, "Accurate and reliable forecasting using stochastic differential equations," *CoRR*, vol. abs/2103.15041, 2021. [Online]. Available: https://arxiv.org/abs/2103.15041

[10] S. Särkkä and A. Solin, *Applied Stochastic Differential Equations.* Cambridge University Press, 2019, ch. 3. [Online]. Available: https://users.aalto.fi/~asolin/sde-book/sde-book.pdf

[11] C. Mode, R. Durrett, F. Klebaner, and P. Olofsson, "Applications of stochastic processes in biology and medicine," *International Journal of Stochastic Analysis*, vol. 2013, 01 2013.

[12] E. Parzen, *Stochatic Processes.* USA: SIAM, 1999.

[13] S. Särkkä and A. Solin, *Applied Stochastic Differential Equations.* Cambridge University Press, 2019, ch. 4. [Online]. Available: https://users.aalto.fi/~asolin/sde-book/sde-book.pdf

[14] S. E. Shreve, *Stochatic Calculus for Finance II, Continous-Time Models.* New York City: Springer, 2008, ch. 4.

[15] D. J. Higham and P. E. Kloeden, *An Introduction to the Numerical Simulation of Stochastic Differential Equations.* Philadelphia: Siam, 2021, ch. 8.

[16] ——, *An Introduction to the Numerical Simulation of Stochastic Differential Equations.* Philadelphia: Siam, 2021, ch. 17.

[17] S. E. Shreve, *Stochatic Calculus for Finance II, Continous-Time Models.* New York City: Springer, 2008, ch. 3.

[18] J. C. Hull, *OPTIONS, FUTURES, AND OTHER DERIVATIVES*, 9th ed. USA: Pearson, 2015.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[20] D. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, pp. 307–392, 01 2019.

[21] S. Särkkä and A. Solin, *Applied Stochastic Differential Equations.* Cambridge University Press, 2019, ch. 5. [Online]. Available: https://users.aalto.fi/~asolin/sde-book/sde-book.pdf

[22] ——, *Applied Stochastic Differential Equations.* Cambridge University Press, 2019, ch. 9. [Online]. Available: https://users.aalto.fi/~asolin/sde-book/sde-book.pdf

[23] M. Opper and C. Archambeau, "The variational gaussian approximation revisited," *Neural computation*, vol. 21, pp. 786–92, 10 2008.

[24] J. Ala-Luhtala, S. Särkkä, and R. Piché, "Gaussian filtering and variational approximations for bayesian smoothing in continuous-discrete stochastic dynamic systems," *Signal Processing*, vol. 111, 07 2014.

[25] S. A. Juho Kokkala and S. Särkkä, "Sigma-point filtering and smoothing based parameter estimation in nonlinear dynamic systems," 2015. [Online]. Available: https://arxiv.org/abs/1504.06173

[26] C. Archambeau, M. Opper, Y. Shen, D. Cornford, and J. Shawe-Taylor, "Variational inference for diffusion processes." vol. 20, 01 2007.

[27] C. Archambeau, D. Cornford, M. Opper, and J. Shawe-Taylor, "Gaussian process approximations of stochastic differential equations." *Journal of Machine Learning Research - Proceedings Track*, vol. 1, pp. 1–16, 01 2007.

[28] H. Lin and S. Jegelka, "Resnet with one-neuron hidden layers is a universal approximator," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/03bfc1d4783966c69cc6aef8247e0103-Paper.pdf

[29] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208

[30] A. Zaeemzadeh, N. Rahnavard, and M. Shah, "Norm-preservation: Why residual networks can become extremely deep?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, pp. 1–1, 04 2020.

[31] T. Zhang, Z. Yao, A. Gholami, K. Keutzer, J. Gonzalez, G. Biros, and M. Mahoney, "Anodev2: A coupled neural ode evolution framework," 06 2019.

[32] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," 02 2018.

[33] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," *CoRR*, vol. abs/1506.02216, 2015. [Online]. Available: http://arxiv.org/abs/1506.02216

[34] N. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," *Advances in neural information processing systems*, vol. 16, 03 2004.

[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, ch. 20, http://www.deeplearningbook.org.

[36] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science (New York, N.Y.)*, vol. 313, pp. 504–7, 08 2006.

[37] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *CoRR*, vol. abs/2003.05991, 2020. [Online]. Available: https://arxiv.org/abs/2003.05991

[38] C. Doersch, "Tutorial on variational autoencoders," 2016. [Online]. Available: https://arxiv.org/abs/1606.05908

[39] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *CoRR*, vol. abs/1402.1128, 2014. [Online]. Available: http://arxiv.org/abs/1402.1128

[40] K. P. Murphy, *Probabilistic Machine Learning An Introduction*. MIT Press, 2022, ch. 1.

[41] ——, *Probabilistic Machine Learning An Introduction*. MIT Press, 2022, ch. 2.

[42] D. J. Higham and P. E. Kloeden, *An Introduction to the Numerical Simulation of Stochastic Differential Equations*. Philadelphia: Siam, 2021, ch. 2.

[43] D. D. Wackerly, W. M. III, and R. L. Scheaffer, *Mathematical Statistics with Applications*, ch. 8.

[44] K. Miura, "An introduction to maximum likelihood estimation and information geometry," *Interdisciplinary Information Sciences (IIS)*, vol. 17, 11 2011.

[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, ch. 8, http://www.deeplearningbook.org.

[46] Y. Xu, "Machine learning with echo state networks," 03 2020.

[47] L. Schomaker, "Simulation and recognition of handwriting movements," 1991.

[48] U. D. Schiller and J. J. Steil, "Analyzing the weight dynamics of recurrent learning algorithms," *Neurocomput.*, vol. 63, p. 5–23, jan 2005. [Online]. Available: https://doi.org/10.1016/j.neucom.2004.04.006

[49] J. Dai, G. K. Venayagamoorthy, and R. G. Harley, "An introduction to the echo state network and its applications in power system," in *2009 15th International Conference on Intelligent System Applications to Power Systems*, 2009, pp. 1–7.

[50] I. Yildiz, H. Jaeger, and S. Kiebel, "Re-visiting the echo state property," *Neural networks : the official journal of the International Neural Network Society*, vol. 35, pp. 1–9, 07 2012.

[51] H. Jaeger, "The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, 01 2001.

[52] C. Gallicchio, "Chasing the echo state property," 2018. [Online]. Available: https://arxiv.org/abs/1811.10892

[53] A. Jierula, S. Wang, T.-M. Oh, and P. Wang, "Study on accuracy metrics for evaluating the predictions of damage locations in deep piles using artificial neural networks with acoustic emission data," *Applied Sciences*, vol. 11, p. 2314, 03 2021.

[54] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," 2018. [Online]. Available: https://arxiv.org/abs/1807.00263

[55] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

A

Parameter configuration for the models

A.1 Conditional distribution approximation

The parameter configuration used for the LSTM-SDE, ESN-SDE and LS-ESN-SDE models for the conditional distribution approximation is shown in Table A.1

Table A.1: Hyperparameters used for the LSTM-SDE, ESN-SDE and LS-ESN-SDE models for the conditional distribution approximation.

	LSTM-SDE	ESN-SDE	LS-ESN-SDE
D_{LSTM}	32	-	-
D_{ESN}	-	10	10
D_{hidden}	8	8	8
N	$[5, 10, 50, 100, 500]$	$[5, 10, 50, 100, 500]$	$[5, 10, 50, 100, 500]$
T_{sde}	1	1	1
N_{sde}	10	10	10
Spectral radius	-	0.99	0.99
Leakage rate	-	0.99	0.99
Sparsity parameter	-	0.99	0.99
Scaling parameter	-	0.1	0.1
k_{long}	-	-	1
m_{short}	-	-	1
Learning rate	$[10^{-2}, 10^{-3}, 10^{-4}]$	$[10^{-2}, 10^{-3}, 10^{-4}]$	$[10^{-2}, 10^{-3}, 10^{-4}]$
# Epochs	$[50, 1000, 1000]$	$[50, 1000, 1000]$	$[50, 1000, 1000]$

A.2 Forecasting

A.2.1 Trivial test case

The parameter configuration used for the LSTM-SDE, ESN-SDE and LS-ESN-SDE as well as the traditional LSTM models for the forecasting in the trivial test case are shown in Table A.2

Table A.2: Hyperparameters used for the LSTM-SDE, ESN-SDE and LS-ESN-SDE models for forecasting in the trivial test case.

	LSTM-SDE	ESN-SDE	LS-ESN-SDE
D_{LSTM}	64	-	-
D_{ESN}	-	256	256
D_{hidden}	16	16	16
N	$[5, 10, 50, 100, 500]$	$[5, 10, 50, 100, 500]$	$[5, 10, 50, 100, 500]$
T_{sde}	1	1	1
N_{sde}	10	10	10
Spectral radius	-	0.99	0.99
Leakage rate	-	0.99	0.99
Sparsity parameter	-	0.99	0.99
Scaling parameter	-	0.1	0.1
k_{long}	-	-	1
m_{short}	-	-	1
Learning rate	$[10^{-2}, 10^{-3}, 10^{-4}]$	$[10^{-2}, 10^{-3}, 10^{-4}]$	$[10^{-2}, 10^{-3}, 10^{-4}]$
# Epochs	$[50, 1000, 1000]$	$[50, 1000, 1000]$	$[50, 1000, 1000]$

A.2.2 Stock prices

The parameter configuration used for the LSTM-SDE, ESN-SDE and LS-ESN-SDE as well as the traditional LSTM models for the forecasting in the trivial test case are shown in Table A.3

Table A.3: Hyperparameters used for the LSTM-SDE, ESN-SDE and LS-ESN-SDE models for the stock prices.

	LSTM-SDE	ESN-SDE	LS-ESN-SDE
D_{LSTM}	64	-	-
D_{ESN}	-	256	256
D_{hidden}	16	16	16
N	$[5, 10, 50, 100, 500]$	$[5, 10, 50, 100, 500]$	$[5, 10, 50, 100, 500]$
T_{sde}	1	1	1
N_{sde}	10	10	10
Spectral radius	-	0.99	0.99
Leakage rate	-	0.99	0.99
Sparsity parameter	-	0.99	0.99
Scaling parameter	-	0.1	0.1
k_{long}	-	-	1
m_{short}	-	-	1
Learning rate	$[10^{-2}, 10^{-3}, 10^{-4}]$	$[10^{-2}, 10^{-3}, 10^{-4}]$	$[10^{-2}, 10^{-3}, 10^{-4}]$
# Epochs	$[50, 1000, 1000]$	$[50, 1000, 1000]$	$[50, 1000, 1000]$

B

List of stock tickers

The list of all the 100 stocks used in the real-world forecasting experiment is shown in the list below.

- AAU
- AEM
- AG
- ALB
- AMRS
- AP
- APD
- ASH
- ATI
- ATR
- AU
- AUMN
- AUY
- AVY
- AXU
- BAK
- BCPC
- BLL
- CCF
- CDE
- CE
- CENX
- CLF
- CLW
- CMC
- CMP
- CX
- DD
- DRD
- ECL
- EGO
- EXK
- EXP
- FCX
- FF
- FMC
- FNV
- FOE
- FSM
- FTK
- GAU
- GEF
- GFI
- GGB
- GLT
- GOLD
- GORO
- GPK
- GPL
- GSV
- HAYN
- HL
- HMY
- HWKN
- IAG
- IFF
- IOSP
- IP
- IPI
- KALU
- KRA
- KRO
- KWR
- LIN
- LPX
- LXU
- LYB
- MAG
- MEOH
- MERC
- MLM
- MOS
- MSB
- MTRN
- MTX
- MYE
- NAK
- NEU
- NG
- NGD
- NP
- NUE
- OI
- OLN
- PAAS
- PKG
- PLG
- PLM
- PPG
- RFP
- RGLD
- RPM
- RS
- SA
- SCCO
- SCHN
- SCL
- SEE
- SHW
- SON
- SQM
- SSL
- STLD
- SVM
- SWM
- SXC
- TECK
- TG
- TGB
- THM
- TREC
- TRQ
- TRS
- TX
- UAN
- USAP
- VGZ
- VMC
- WLK
- WOR
- X
- ZEUS

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY