Analyzing an Application of Neural SDEs in Finance and the Challenges in Synthetic Data Generation

Analyzing an Application of Neural SDEs in Finance and the Challenges in Synthetic Data Generation

Master Thesis

to be defended publicly on
Thursday 6 , July 2023 at 10.00

by

Gianmarco Gargiulo

Financial Engineering
Master Applied Mathematics
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Student ID:	5622867
Project Duration:	2022.12.10 - 2023.07.06
Thesis committee:	
Prof. Dr. A. Papapantoleon	TU Delft, Full Professor
Prof. Dr. N. Parolya	TU Delft, Assistant Professor
Prof. Dr. F. Barsotti	TU Delft, Assistant Professor

Contents

Preface

I would like to take this opportunity to express my heartfelt gratitude to those who have been an integral part of my journey and have contributed to the completion of this thesis.

First and foremost, I would like to express my deepest appreciation to my family. Their unwavering support, love, and encouragement have been fundamental. I would also like to extend my sincere gratitude to my friends. Despite the geographical distance, their presence has been constant. I am profoundly grateful to my supervisor, Dr. Papapantoleon, for his guidance, patience, and expertise. Furthermore, I would like to extend my thanks to Dr. Parolya and Dr. Barsotti for being a member of my thesis committee

Gianmarco Gargiulo
Delft, June 2023

1 Introduction

The Neural Stochastic Differential Equations (NSDEs) connect ideas of stochastic differential equations and neural networks.

Formalized by Itô for the first time in [1], stochastic differential equations (SDEs) are ordinary deterministic differential equations with a noise factor added. These equations are useful in many contexts where predictable changes are paired with noisy fluctuations.

Neural networks, on the other hand, emerged in the 1940s and 1950s as computational models inspired by biological nervous systems. Comprising interconnected artificial neurons, they possess the capability to learn from data and perform complex tasks.

Chapter 2 lays the groundwork by providing an overview of generative models. In Chapter 3, we provide a comprehensive literature review, with particular emphasis on two approaches: one outlined in [2] and another in [3]. These approaches are selected due to their links with generative models. Additionally, we explore conditions for the existence and uniqueness of solutions in stochastic differential equations, aiming to identify analogous properties in the field of NSDEs. Through extensive research, we find that certain techniques, such as weight clipping [3] and scalar normalization [4], yield the desired results.

Financial modeling heavily relies on SDE-based models. This thesis explores the potential application of neural SDEs in this domain. However, the question of which models to rely on arises. Rough volatility models, which employ fractional Brownian motion instead of Brownian motion as noise, have gained popularity. While empirical evidence supporting this approach has been identified in [5], conflicting opinions regarding its reliability persist.

Chapter 4 focuses on replicating and explaining the results supporting the rough volatility hypothesis. Additionally, we propose counterarguments, particularly highlighting the findings presented in [6] and [7]. These studies not only provide valuable insights but also offer a new perspective on the topic. Furthermore, we replicate some of the results presented in these articles, which raises questions about the validity of the empirical evidence used to support the rough volatility models.

Motivated by the previous chapter's discussion, we shift our focus to Markovian-type models. Our research centers on the joint calibration of SPX and VIX options, as well as SPX option calibration using Markovian-type neural SDEs. Chapter 4 also includes a literature survey of Markovian models addressing joint calibration. In the case of joint calibration, we generate synthetic data following the idea described in [8], while conducting calibration tests on real data.

Considering the issue of data scarcity, the final chapter is dedicated to exploring interesting methods for generating synthetic data. We focus on the works of [9] and [10], assessing their advantages, effectiveness, and suitability for generating financial data. The first paper we examine, [9], is built on the idea of a signature, which we briefly discuss. Additionally, we introduce optimal transport, the underlying principle in [10]. Lastly, we propose a potential new method that incorporates GAF, a technique that allows to transform time series in images, and diffusion models

2 Background Material: Generative Models

A generative model is a statistical or machine learning model that is designed to capture and learn the underlying structure or distribution of a given dataset. It generates new samples that closely replicate the original data on which it was trained.

They are particularly valuable in situations where data may be scarce or when there is a need to generate synthetic data for testing or simulation purposes.

Examples of generative models include generative adversarial networks (GANs), variational autoencoders (VAEs), and restricted Boltzmann machines (RBMs). Each of these models employs distinct techniques to learn from the data and generate new samples.

In terms of a concise taxonomy of generative models based on the maximum likelihood approach, following [11] they can be broadly categorized into two groups: Implicit Models and Explicit Models. Techniques such as generative adversarial networks (GANs) that learn to produce samples by training a generator network to fool a discriminator network are examples of implicit models. Explicit Models are further classified based on the tractability of the underlying probability distribution. Tractable Models are those in which the probability distribution can be efficiently computed, whereas Untractable Models are those in which finding the exact distribution is infeasible. Normalising Flow is an example of a Tractable Model, whereas the Variational Autoencoder (VAE) is an example of an Untractable Model .

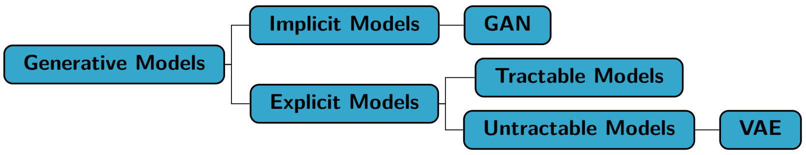


Figure 2.1.: Generative modelling taxonomy based on the maximum likelihood from [12]

In this section, Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) will be introduced. These generative models have significant implications and play a crucial role in the thesis. To provide a concise explanation, we will follow the description in [12], [13] for the GAN and [14] for the VAE.

2.1. GAN

Generative Adversarial Networks (GANs) are a class of generative models that aim to generate realistic samples that correspond to a specific input distribution without explicitly modeling the probability density or distribution of the input data. Introduced by Goodfellow et al. [12], GANs learn intricate data distributions and generate high-quality samples. In the GAN architecture we have a generator G and a discriminator D, both are typically implemented as neural networks. The generator takes in a fixed-dimensional input and generates an output with a variable dimensionality that matches the target data's characteristics. On the other hand, the discriminator receives input samples and produces a scalar output that represents the probability of the input being a real sample. The discriminator's output can be regarded as a binary classification task, with values closer to 1 indicating that the input is real, and values closer to 0 indicating that the input is fake. Through the adversarial interaction, the model iterates and refines its performance, resulting in greater performance and the generation of more realistic and diverse samples.

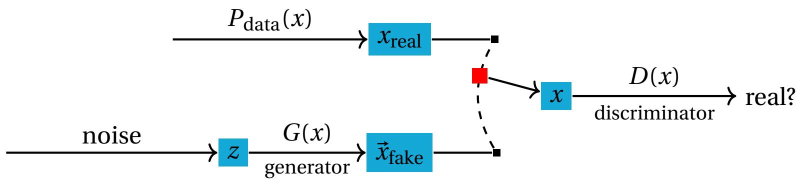


Figure 2.2.: Example of GAN behaviour generated using the Tikz library [15]

Particularly, one of the important developments in the field of Generative Adversarial Networks (GANs), the Wasserstein GAN proposed in [16] will be useful in the thesis. Leveraging the relationship with optimal transport theory, in the Wasserstein GAN, the Earth-Mover's distance (EM), also known as the Wasserstein distance, is utilized for training the model instead of the Jensen-Shannon divergence used in the GAN.

2.2. Variational Autoencoders

The Variational Autoencoders (VAEs) are a class of generative models that aim to learn a latent representation of the input data, which captures the underlying structure and distribution of the data, and use this representation to generate new data samples that resemble the input data distribution.

Proposed by Kingma et al in [14], a crucial role is played in VAEs by the latent space, which acts as a lower-dimensional representation that captures the essential characteristics of the input data. It serves as a bottleneck layer in the VAE architecture, compressing the information while preserving the key features. The latent space is defined by a probability distribution, often a multivariate Gaussian distribution, which allows for sampling and generating new data points. The encoder neural network maps the input data to this latent distribution by outputting the mean and variance parameters. These parameters are then used to sample latent points from the distribution. The sampled latent points are then passed through the decoder network, which is symmetrical to the encoder network. The decoder aims to reconstruct the original input data from the latent representation, progressively generating outputs that resemble the input data distribution.

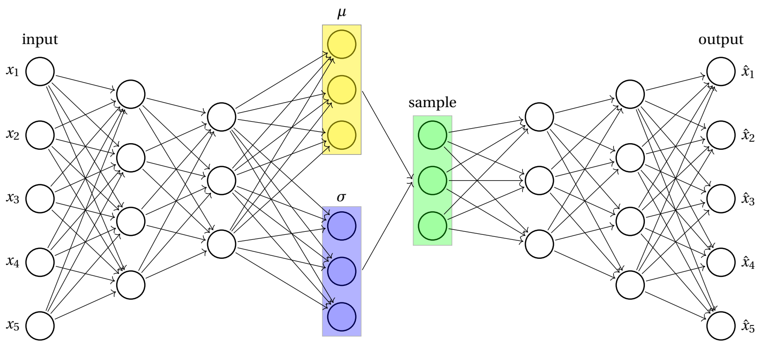


Figure 2.3.: Example of VAE architecture generated using the Tikz library [15]

In VAEs, the loss function is formed by two elements: the reconstruction loss and the regularization loss. It is typically formulated as the sum of the first and a weighted term of the second.

The capability of the VAE to recreate the initial input data from the latent space is measured by the reconstruction loss. Depending on the type of input data, binary cross-entropy (BCE) loss or mean squared error (MSE) loss are frequent options for the reconstruction loss.

The regularization loss encourages the latent distribution to resemble a predefined distribution, usually a multivariate Gaussian distribution. This regularization is achieved through the Kullback-Leibler (KL) divergence between the learned latent distribution and the target distribution. The KL divergence measures the difference between the two distributions, and by minimizing it, the VAE ensures that the latent space follows a desired distribution.

3
Neural SDEs

The two dominant modeling frameworks of neural networks and stochastic differential equations are successfully merged in neural stochastic differential equations.

The main goal of the initial phase of our research was to conduct a thorough examination of the available literature in order to enhance our comprehension of neural stochastic differential equations (NSDEs). Our focus was to comprehend the nature of these mathematical objects, identify the criteria that ensure the existence and uniqueness, and find potential applications to financial mathematics.

As a result, this chapter will begin with an overview of both neural networks and stochastic differential equations, specifically with reference to [17], [18]. The topic of neural stochastic differential equations will thereafter be covered while keeping in mind the goals that were previously outlined.

3.1. Stochastic Differential Equations

A system's evolution is described using ordinary differential equations (ODEs). When a random noise is added it becomes a stochastic differential equations (SDEs).

Let $W(t)$, $t \geq 0$, be a Brownian motion process. An equation of the form

$$dX(t) = \mu(X(t), t)dt + \sigma(X(t), t)dW(t). \tag{3.1}$$

where the given functions $\mu(x, t)$ and $\sigma(x, t)$ are called drift and diffusion, $X(t)$ is the unknown process termed a stochastic differential equation (SDE) driven by Brownian motion.

Definition 1 (Strong solution, [17]). *A process $X(t)$ is called a strong solution of the SDE (3.1) if for all $t > 0$ the integrals $\int_0^t \mu(X(s), s)ds$ and $\int_0^t \sigma(X(s), s)dB(s)$ exist, with the second being an Itô integral, and*

$$X(t) = X(0) + \int_0^t \mu(X(s), s)ds + \int_0^t \sigma(X(s), s)dB(s). \tag{3.2}$$

Proposition 1. *The random noise W and the starting data ξ can be considered as the input of a dynamical system characterized by whose output is X and is characterized by coefficients (μ, σ), [19].*

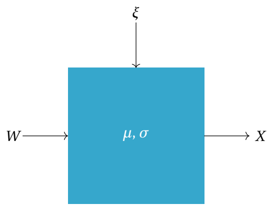


Figure 3.1.: Schematization of the behaviour of an SDE

3.1.1. Existence and Uniqueness

Let $X(t)$ satisfy (3.1)

Theorem 1 (Existence and Uniqueness). *If the following three conditions are satisfied*

1. *Coefficients are locally Lipschitz in x uniformly in t, that is, for every T and N there is a constant K depending only on T and N, such that for all $|x|,|y| \le N$ and all $0 \le t \le T$*
$$|\mu(x,t) - \mu(y,t)| + |\sigma(x,t) - \sigma(y,t)| < K|x-y|. \tag{3.3}$$
2. *Coefficients satisfy the linear growth condition*
$$|\mu(x,t)| + |\sigma(x,t)| \le K(1+|x|) \tag{3.4}$$
3. *$X(0)$ is independent of $(B(t), 0 \le t \le T)$, and $\mathbb{E}^2 X(0) < \infty$*

A unique strong solution $X(t)$ of the (3.1) exists.

As in the case of the proof for the existence of a solution to an ODE, the proof of existence of an SDE is based on the concept of Picard iterations [20]. The Lipschitz condition, which permits use of Gronwall's lemma, ensures uniqueness.

3.2. Neural Networks

Their name and construction are inspired by the human brain, with the goal of mimicking how neurons transmit signals. The multilayer perceptron is a subclass of neural networks that has shown to be one of the most useful in real-world applications.

Perceptrons, also known as nodes or units, are at the core of a neural network. Their functioning is well explained in Figure (3.2). They are arranged in layers, resulting in a layered architecture.

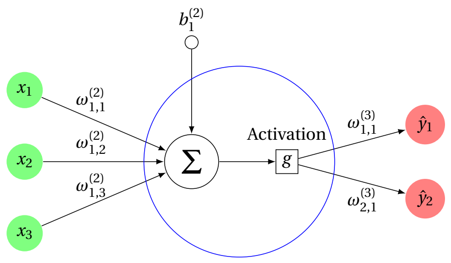


Figure 3.2.: Example of Neural Networks architecture generated using [21]

Basically, in a perceptron, a non-linear function known as the activation function transforms the weighted sum of the inputs, which are the outputs from the previous layer or are from the input layer, and a bias term, which represents a learnable constant, into an output.

The feedforward neural network, as shown in figure (3.3), is the most typical architecture, in which information goes from the input layer, through one or more hidden layers, and finally to the output layer. The input layer receives the initial data. The hidden layers are intermediate layers that perform computations on the data. They gradually learn to extract relevant features and capture complex patterns from the input. The output layer produces the final prediction or output based on the information learned by the hidden layers.

The number of hidden layers and neurons in each layer might vary depending on the problem's complexity and the amount of data provided.

Neural networks learn by iteratively modifying the weights and biases based on the difference between the predicted and actual output during the training period. This is accomplished using an optimization technique, such as gradient descent, which seeks to minimize a predetermined loss function that measures the performance of the network.

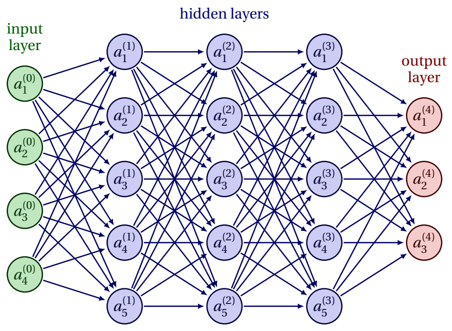


Figure 3.3.: Example of Neural Networks architecture generated using the Tikz library [22]

3.3. NEURAL STOCHASTIC DIFFERENTIAL EQUATIONS

A neural SDE is a stochastic differential equation as in Equation (3.1) of which the drift and diffusion functions μ and θ are parametrized by neural networks.

Neural networks have been shown to possess significant learning capability, and there exists a theoretical justification for utilizing them to approximate the drift and diffusion functions in the context of stochastic differential equations (SDEs). The universal approximation theorem, which is the theoretical foundation, states that, given a sufficient number of parameters, neural networks are capable of approximating any measurable function with any degree of accuracy [23], [24]. This implies that the drift and diffusion functions of any SDE can theoretically be approximated by a multi-layer neural network. In circumstances where the drift and diffusion functions are unknown or when some data is supposed to follow an SDE but the type of SDE is ambiguous, neural networks can be used to simulate the process. The identification of the drift and diffusion functions can be automated by updating the parameters in the neural networks to obtain the best fit for the corresponding SDE [25]. In complicated systems where obtaining analytical formulations for the drift and diffusion functions is challenging or impossible, this method can greatly simplify the modeling process.

3.4. EXISTENCE & UNIQUENESS OF A SOLUTION FOR NEURAL SDEs

Regarding the existence and uniqueness of solutions for Neural SDEs, it has been found in the literature that a unique solution can exist in certain circumstances. This is the case when weight clipping is used, as mentioned in [3].

The weight clipping approach involves training a neural network with weight parameters w, where the weights are restricted to a fixed range or box after each gradient update, thus ensuring that the weight parameters w remain within a compact space. This methodology has been discussed in literature in the context of Wasserstein GANs theory, for example in [16], where it has been noted that this weight clipping technique leads to all the functions f_w being K-Lipschitz, with the Lipschitz constant K depending only on the size of the weight parameter space and not on the individual weight values.

Another successful approach, inspired by the theory of Wasserstein GANs, is spectral normalization. In [26] is provided a formal explanation of how weight normalization guarantees the existence and uniqueness of the solution of a neural SDEs. Given the the interest for our thesis, we decided to replicate it. In order to do that we need to define the concept of spectral normalization

Consider a neural network with the input $\mathbf{x}$ and the following structure:

$$f(\mathbf{x}, \Theta) = \theta^{L+1} \alpha_L(\theta^L \alpha_{L-1}(\theta^{L-1}(\cdots \alpha_1 \theta^1 \mathbf{x} \cdots)))), \tag{3.5}$$

where $\Theta := \{\theta^1, ..., \theta^L, \theta^{L+1}\}$ is the learning parameters set, $\theta^l \in \mathbb{R}^{d^l \times d^{l-1}}$, $\theta^{L+1} \in \mathbb{R}^{1\times d_L}$, and α_l is an element-wise non-linear activation function. For reasons of simplicity, are excluded the bias term of each layer.

3.4.1. SPECTRAL NORMALIZATION

Spectral normalization, introduced in the work of Miyato et al. [4], is a technique employed to regulate the Lipschitz constant of a function f by directly restricting the spectral norm of each layer $\phi : \Gamma_{in} \mapsto \Gamma_{out}$.

Definition 2 (Spectral norm)**.** *The spectral norm of matrix M, $\sigma(M)$ can be expressed as:*

$$\sigma(M) = \max_{\Gamma:\Gamma\neq 0} \frac{||M\Gamma||_2}{||\Gamma||_2} = \max_{||\Gamma||_2 \leq 1} ||M\Gamma||_2 \tag{3.6}$$

which is equivalent to the largest singular value of M

By leveraging the definition of $||\phi||_{\text{Lip}}$ as $\sup_\Gamma \sigma(\nabla\phi(\Gamma))$ and (2), we can further simplify the analysis. For a linear layer $\phi(\Gamma) = \theta\Gamma$, the following holds:

$$||\phi||_{\text{Lip}} = \sup_\Gamma \sigma(\nabla\phi(\Gamma)) = \sup_\Gamma \sigma(\theta) = \sigma(\theta) \tag{3.7}$$

Defining $||\alpha_l||_{Lip} = \kappa_l$, and noting that $||\phi_1 \cdot \phi_2||_{\text{Lip}} \leq ||\phi_1||_{\text{Lip}} \cdot ||\phi_2||_{\text{Lip}}$ we have:

$$\begin{aligned} ||f||_{\text{Lip}} \leq ||(\Gamma_L \mapsto \theta^{L+1}\Gamma_L)||_{\text{Lip}} \cdot ||\alpha_l||_{\text{Lip}} \cdot ||(\Gamma_{L-1} \mapsto \theta^L\Gamma_{L-1})||_{\text{Lip}} \cdots \\ ||\alpha_1||_{\text{Lip}} \cdot ||(\Gamma_0 \mapsto \theta^1\Gamma_0)||_{\text{Lip}} = \kappa \prod_{l=1}^{L+1} ||(\Gamma_{l-1} \mapsto \theta^1\Gamma_{l-1}||_{\text{Lip}} = \kappa \prod_{l=1}^{L+1} \sigma(\theta^l) \end{aligned} \tag{3.8}$$

where $\kappa = \prod_{l=1}^L \kappa_l$

Spectral normalization is employed to normalize the spectral norm of the weight matrix θ such that it satisfies the Lipschitz constraint $\sigma(\theta) = 1$:

$$\tilde{\theta}_{SN}(\theta) := \frac{\theta}{\sigma(\theta)} \tag{3.9}$$

By normalizing each θ^l using $\tilde{\theta}_{SN}(\theta)$, we can utilize the inequality mentioned above and the fact that $\sigma(\frac{\tilde{\theta}_{SN}}{(\theta)}) = 1$ to see that $||f||_{\text{Lip}}$ is bounded from above by κ .

Theorem 2. *Let μ_θ and σ_θ be neural networks with L and K number of layers respectively and assume that it holds, for both μ_θ and σ_θ, that each activation function is in each corresponding layer is a global Lipschitz function.Therefore applying spectral normalization μ_θ and σ_θ satisfies the conditions, (1), to guarantee existence and uniqueness of a neural SDE.*

Proof. If we now apply spectral normalization to each weight matrix at our neural network, for assumed finite $L, \kappa \leq \infty$, it holds that

$$||\mu_\theta(t,x)||_{Lip} = \frac{||\mu_\theta(t,x) - \mu_\theta(t,y)||}{||x-y||} \leq \kappa \rightarrow ||\mu_\theta(t,x) - \mu_\theta(t,y)|| \leq \kappa||x-y|| \tag{3.10}$$

for any $x, y\, \mathbb{R}^N$.

Knowing that $||\mu_\theta(t,x) - \mu_\theta(t,y)|| \leq \kappa||x-y||$ for some constant κ and for all $x, y \in \mathbb{R}^N$, then consider

$$\begin{aligned} ||\mu_\theta(t,x)|| = ||\mu_\theta(t,x) - \mu_\theta(t,0) + \mu_\theta(t,0)|| \leq ||\mu_\theta(t,x) - \mu_\theta(t,0)|| + ||\mu_\theta(t,0)|| \\ = \kappa||x-0|| + ||\mu_\theta(t,0)|| = \kappa||x|| + ||\mu_\theta(t,0)|| = \kappa||x|| + D \end{aligned} \tag{3.11}$$

where $\kappa = \prod_{l=1}^{L} \kappa_l$

The proof provided for μ_θ also holds true for σ_θ.

□

3.5. THEORETICAL INTERPRETATION OF NEURAL SDEs

In [2] a concise survey of the current state-of-the-art with regards to neural stochastic differential equations is provided. As this survey served as a fundamental reference for determining which methodologies and works to adopt in the present section, a brief summary of its content is herein reported. Two approaches in particular will be thoroughly investigated due to their interesting mathematical frameworks. In general, the formulations and applications of neural stochastic differential equations (SDEs) can be classified into two major categories. One approach involves utilizing SDEs as a means to systematically introduce stochastic perturbations into a system, with a focus on analyzing the terminal state of the SDE. The other approach, on the other hand, places emphasis on investigating the entire temporal trajectory of the SDE as the primary object of interest. In [27] and [28], the authors present a training technique that relies on optimizing a variational bound through the utilization of forward-mode autodifferentiation. The methodology they propose involves deriving deep latent Gaussian models as a continuous limit. In [29], the authors introduce the concept of neural stochastic differential equations (SDEs) by employing rough path theory to obtain the limit of random ordinary differential equations (ODEs). If until now all those mentioned have been solely focusing on the terminal value of the SDE, the following are the cases belonging to the second group. In [30], [3], and [31], the authors proposed to train a training strategy with the goal of minimizing the difference between the expected value of a given function of interest f under the learned distribution μ and ν, expressed as $\int f d\mu - \int f d\nu$, where μ and ν are respectively the learned and true distribution. This methodology, in particular the approach described in [3], is discussed more in depth in (3.5.2). Multiple researchers, including [32], [25], and [29], aim to leverage stochasticity as a means to augment or regularize neural ordinary differential equation (ODE) models. The article [33] presents an analogous approach to neural ordinary differential equations (ODEs) as introduced in [34], which is considered as the foundational work for the entire theory. In their research, the authors adopt a sophisticated method that incorporates two-sided filtrations and backward Stratonovich integrals to introduce neural stochastic differential equations (SDEs).

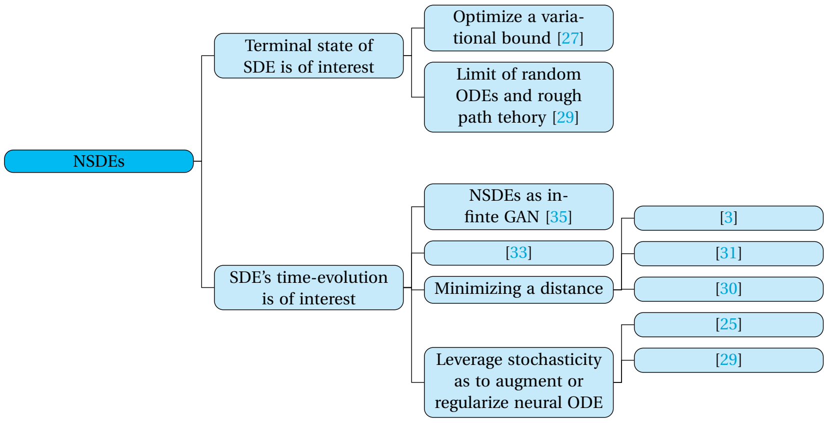

NSDEs
Terminal state of SDE is of interest
Optimize a variational bound [27]
Limit of random ODEs and rough path tehory [29]
SDE's time-evolution is of interest
NSDEs as infinte GAN [35]
[33]
Minimizing a distance
Leverage stochasticity as to augment or regularize neural ODE
[3]
[31]
[30]
[25]
[29]

Figure 3.4.: Representation of the State of the Art

3.5.1. NEURAL SDES AS INFINITE-DIMENSIONAL GANS

Within the context of the two aforementioned categories, we turn our attention to a method that, in our opinion, merits further analysis: the approach presented in [2]. This method falls into the second classification, it builds distributions on path space by using stochasticity. Our decision to investigate this particular method is based on the fact that it serves as a direct expansion of the well-established classical approach. In this case, a perspective based on Wasserstein GANs can be used to expand the current traditional approach for fitting SDEs. A generator-discriminator pair made up of a neural SDE and a neural CDE is combined to accomplish this.

Since the Wasserstein loss function possesses a unique global minimum, it becomes possible to learn arbitrary SDEs in the infinite data limit.

Furthermore, this is the first SDE modelling approach that does not rely on pre-specified statistics or the utilization of density functions.

Going into more detail, the Neural SDEs considered in [35] are as follows:

$$y(0) = \zeta_\theta(v),$$

$$dy(t) = \mu_\theta(t, y(t))dt + \sigma_\theta(t, y(t)) \cdot dw(t),$$

$$x(t) = \alpha_\theta y(t) + \beta_\theta, \tag{3.12}$$

where $\zeta_\theta, \mu_\theta, \sigma_\theta$ can be any type of neural networks, such as a simple feedforward network. $\zeta_\theta, \mu_\theta, \sigma_\theta, \alpha_\theta, \beta_\theta$ are all parametrised by θ and $v \sim N(0, \mathbb{1})$

The choice of passing the initial condition through ζ_θ is intended to introduce an additional source of noise.

The hidden state is represented by the variable X, which is not meant to be the output. Future evolution would follow to a Markov property if X were the output, even if this might not always be the case. Hence, an additional readout operation is performed to obtain the variable Y.

The approach described here addresses the challenge that arises in the context of solving stochastic differential equations (SDEs). When it comes to obtaining a solution for ordinary differential equations (ODEs), we typically have the necessary data and aim to minimize a distance or a loss as for example the mean squared error. However, in the case of SDEs, both the data and equations themselves are inherently random. Consequently, when comparing a sample of both there is no mathematical basis to expect a close correspondence in terms of distances.

The idea is to achieve a distributional match, that is, to have the distribution of the solutions of the SDEs be approximately equal to the distribution of the observed data. There are two main schools of thought to achieve this match.

The first approach involves matching certain statistics, such as the difference between the expected value of a function F evaluated at the solutions of the SDE and the expected value of F evaluated at the data. The choice of the function F is crucial, and depends on the specific problem at hand. For instance, if F belongs to a family with a Lipschitz norm of at most one, then the Wasserstein distance can be used to measure the distance between the distributions. On the other hand, if F belongs to a family defined by a kernel, then the maximum mean discrepancy (MMD) can be employed.

Alternatively, the second approach involves utilizing ideas from variational inference. In this approach, the evidence lower bound (ELBO) is maximized to achieve a match between the two distributions.

In accordance with the article's methodology, the neural SDEs are trained to minimize the 1-Wasserstein distance, or $W(\cdot,\cdot)$. A critical decision is the choice of the function F, which is parametrized as F_ϕ.

Let the law of the model x be indicated by $\mathbb{P}_x$. If we define $\mathbb{P}_\gamma$ as the (empirical) law of the data γ, the model is trained by optimizing the following :

$$\min_{\theta} W(\mathbb{P}_x, \mathbb{P}_y), \tag{3.13}$$

To do that they make use of the theory of Wasserstein GAN [16], exploiting the fact that F is parametrized, to train the model using a minmax approach.

$$\min_{\theta} \max_{\phi} (\mathbb{E}_x[F_\phi(x)] - \mathbb{E}_y[F_\phi(y)]). \tag{3.14}$$

Finding a function F that can correctly discriminate between real and generated data is the subject of the second problem that is addressed. In this case, each sample generated by the generator corresponds to a continuous path that is infinite-dimensional, and thus, the discriminator must be capable of accepting such paths as inputs.

Their solution involves utilizing a neural controlled differential equation as discriminator.

Definition 3 (Neural controlled differential equation, [35]). *A neural controlled differential equation is defined as the solution of the Control Differential Equation*

$$y(0) = \zeta_\theta(x(0)), \;\; y(t) = y(0) + \int_0^t f_\theta(y(s))dx(s) \;\; t \in (0, T]. \tag{3.15}$$

where ζ_θ, f_θ are both neural networks

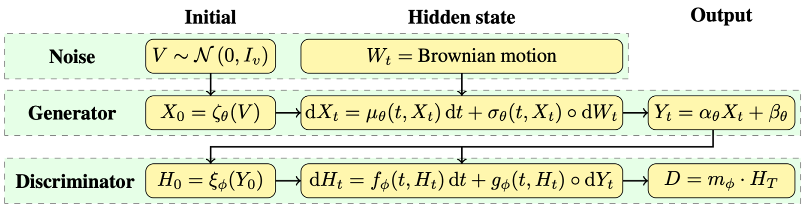

Initial
Hidden state
Output
Noise
$V \sim \mathcal{N}(0, I_v)$
W_t = Brownian motion
Generator
$X_0 = \zeta_\theta(V)$
$\mathrm{d}X_t = \mu_\theta(t, X_t)\,\mathrm{d}t + \sigma_\theta(t, X_t) \circ \mathrm{d}W_t$
$Y_t = \alpha_\theta X_t + \beta_\theta$
Discriminator
$H_0 = \xi_\phi(Y_0)$
$\mathrm{d}H_t = f_\phi(t, H_t)\,\mathrm{d}t + g_\phi(t, H_t) \circ \mathrm{d}Y_t$
$D = m_\phi \cdot H_T$

Figure 3.5.: Summary of equations adopted from [2]

3.5.1.1. EXPERIMENTS

To test this approach the authors conducted an analysis on various datasets, including Stocks, Beijing Air Quality, and Weights, using three different metrics. These metrics were then compared against the Latent ODE model [36] and the continuous-time flow process (CTFP) [37]

In their analysis, the researchers observed that Neural SDEs consistently outperformed the Latent ODE model and CTFPs in terms of both predictive accuracy and the maximum mean discrepancy (MMD) metric. Specifically, Neural SDEs exhibited substantially better results compared to the other models.

When examining the Stocks dataset, which is a domain where SDE models have traditionally been applied, Neural SDEs demonstrated superior performance not only in predictive accuracy and MMD but also in a classification metric. This suggests that Neural SDEs excel in capturing the underlying dynamics and random fluctuations present in these datasets, which are not purely driven by a deterministic drift.

The toy example presented in Figure (3.6) is an application of the method to a one-dimensional Ornstein-Uhlenbeck process, using [38]

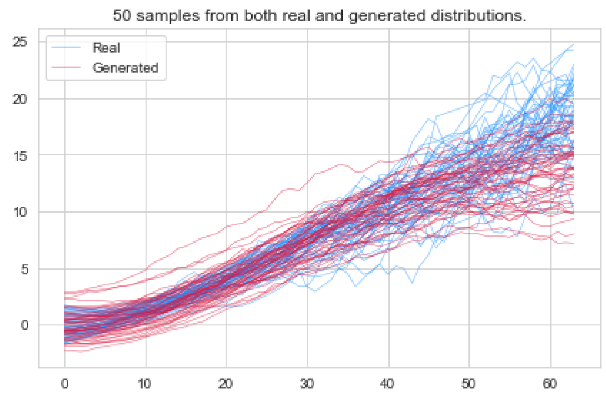


Figure 3.6.: Example of Generated paths

3.5.2. NEURAL SDEs AS GENERATIVE MODELS

The second approach, which is proposed in [3], offers a valuable perspective on the connection between generative models and stochastic differential equations (SDEs). It is particularly relevant to our thesis as we made use of this approach in conducting experiments. In the context of generative model we usually have a source distribution, denoted as μ, and a target distribution, denoted as ν, which represents the input data. The idea is to construct a mapping, denoted as $T_{\#\mu} = \nu$, that pushes the source distribution μ onto the target distribution ν.

As indicated in equation (1), stochastic differential equations (SDEs), including neural SDEs, can be interpreted as generative models. To gain a deeper understanding of this approach, a case related to finance is presented. Let denote the true martingale measure as $\mathbb{Q}^{\text{market}}$. Under this measure, all liquid derivatives are perfectly calibrated, meaning that their expected values, denoted as $\mathbb{E}^{\mathbb{Q}^{\text{market}}}[\Psi_i]$, match their market prices, denoted as $\mathfrak{p}(\Psi_i)$, for all $i = 1, \ldots, M$.

In the case of Neural SDEs this translate into parametrize the mapping function, allowing to the Neural SDEs to effectively map the source distribution μ of the underlying financial variables, S_0, as well as the Wiener measure on $C([0,T];\mathbb{R}^n)$, to the target distribution $Q^\theta = (T_t^\theta)_{\#\mu}$.

The objective is to find the optimal parameter values, denoted as θ^*, such that the mapping $T_{\#\mu}^{\theta^*}$ provides a good approximation of the target martingale measure $\mathbb{Q}^{\text{market}}$

according to a metric. In the context of this paper, the focus is on a specific metric, the mean squared error.

$$D(T^{\theta}_{\#\mu}, \mathbb{Q}^{\text{market}}) := \sum_{i=1}^{M} \mathscr{L}\left(\int \psi_i(\omega)(T^{\theta}_{\#\mu})(d\omega), \int \psi_i(\omega)\mathbb{Q}^{\text{market}}(d\omega)\right). \tag{3.16}$$

3.6. CONCLUSION

In this chapter, we have obtained answer to the questions that arose concerning neural SDEs. For their potential applications in financial mathematics, we will specifically draw inspiration from the studies conducted in [3] and [31]. However, prior to start this work, we need to determine the models to reference and the problems to address. These matters will be discussed in the following chapters.

4
Volatilty, Rough or not?

Managing model selection and calibration are essential processes in the field of mathematical finance .

Initially, for describing the dynamics of an asset price the Geometric Brownian Motion (GBM) model was proposed, given by:

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{4.1}$$

where μ and σ are constants.

The only unknown parameter in this model is σ, which is considered to be constant and deterministic. In the Black-Scholes model, which relies on this Geometric Brownian Motion, the implied volatility surface across various strike prices and time to maturity would appear flat. However, empirical evidence shows that the volatility surface is never flat. Thus, this type of models fails to adequately capture the dynamics of volatility.

Therefore, a commonly adopted method in finance is modelling the dynamic of the volatility. There are two primary approaches to this representation:

1. Local Volatility: In this approach, volatility $\sigma(t, S_t)$ is modeled to depend on both the current asset level S_t and time t. This allows for a dynamic estimation of volatility based on the specific characteristics of the underlying asset.

2. Stochastic Volatility Models: This approach considers the dynamics of σ_t as a stochastic process driven by a Brownian motion. The equation governing the evolution of σ_t is given by:

$$d\sigma_t = \mu(t, \sigma_t)dt + \Lambda(t, \sigma_t)dW_t \tag{4.2}$$

Various models fall under this category, including the Hull and White model, the Heston model, and the SABR model, among others.

Usually these models were typically developed through a three-step process. First, statistical properties of the underlying time series, known as stylized facts, were gathered.

Then, a parsimonious model was handcrafted to effectively capture the desired market characteristics without unnecessary complexity. Finally, the handcrafted model was calibrated and validated.

The application of Neural Stochastic Differential Equations (NSDEs) represents a completely different approach. Rather than imposing a pre-defined model structure, the data are allowed to dictate the model while still maintaining a strong prior. This is achieved by using SDEs for the model dynamics but adopting an overparameterized neural network to define the drift and diffusion terms, rather than fixed parametrization. In this way, calibration and model selection are performed simultaneously, driven by the data [3].

The main question addressed in this chapter is which type of stochastic process is more suitable to use as a prior: the one driven by traditional Brownian motion or the one driven by fractional Brownian motion. Initially, the theory was concentrated on stochastic processes driven by Brownian motion. However, the introduction of models driven by fractional Brownian motion, as proposed by Comte and Reanult [39], has garnered considerable attention in recent times.

Definition 4 (Fractional Brownian motion). *A fractional Brownian motion (W_t^H) with Hurst parameter* $H \in (0,1)$ *is a continuous Gaussian process with covariance structure*

$$Cov(W_t^H, W_s^H) = \left(t^{2H} + s^{2H} - |t-s|^{2H}\right), \quad t, s \in \mathbb{R}. \tag{4.3}$$

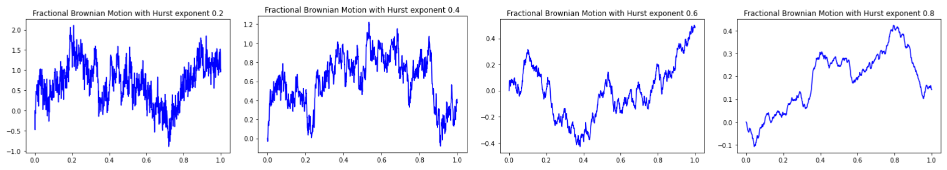


Figure 4.1.: Examples of fractional Brownian motion with different Hurst exponent

Originally, financial models primarily focused on Hurst parameters satisfying $H > \frac{1}{2}$. However, the models introduced by Gatheral et al. [5], which are driven by a fractional Brownian motion characterized by a Hurst parameter below $\frac{1}{2}$, have gained more popularity and become increasingly favored in the field.

The motivation behind this approach comes from empirical evidence suggesting that volatility exhibits a specific type of roughness, known as Hölder roughness, which is strictly less than $\frac{1}{2}$.

This increased attention is based on the potential advantages and improved performance they offer compared to traditional models. However, a major challenge arises due to the computational complexity of rough volatility models. The non-Markovian nature of these processes makes them computationally demanding, necessitating specialized numerical techniques or approximations for efficient calibration and simulation. When combined with the use of Neural SDEs, the computational cost is further amplified, making the modeling and estimation processes even more challenging. For this reason in this chapter is conducted a thorough and meticulous analysis of the supporting results, while also acknowledging the existence of conflicting opinions surrounding these findings. The intention is to gain valuable insights into the problem at hand, and at the same time develop a solid justification for picking the model type to use.

4.1. Is Volatility rough?

As mentioned, this theory is based on empirical evidence, specifically the ability of these models to recreate properties of the implied volatility surface and properties of the historical time series of volatility.

In exploring the empirical evidence of rough volatility models, this section will primarily focus on the comprehensive examination and analysis of the main reference, [5].

4.1.1. The implied volatility surface

The correlation between the smile and the time to expiration is intricately connected to the underlying dynamics. Given a stochastic volatility model the term structure of at-the-money (ATM) volatility skew , $\psi(\tau)$, defined as

$$\psi(\tau) := \left|\frac{\partial \sigma_{imp}(k,\tau)}{\partial K}\right|_{k=0} \tag{4.4}$$

is highly sensitive of the volatility models chosen. The term "skew" refers to the rate at which implied volatility changes with respect to log-moneyness k , τ represents the time to expiration and σ_{imp} the implied volatility.

The fact that the term-structure of ATM skew exhibits a power-law-like decay for a large range of maturities has been seen as a sign that volatility is rough. Moreover, in [41] is shown the ATM volatility skew is of the form $\psi(\tau) \sim \tau^{H-\frac{1}{2}}$ for model where a fractional Brownian motion with Hurst index H drives the volatility.

4.1.2. Characteristics of the historical time series of volatility

The second reason is associated with the characteristics of historical time series of volatility. Instantaneous volatility cannot be directly observed, but it can be estimated by integrating the variance over a day. Estimating integrated variance is conceptually straightforward and various statistical methods are available for this purpose. However, the practical challenge lies in data cleaning, which is performed by the Oxford-Man Institute. Therefore, daily estimates of realized variance from the Oxford-Man Institute are used as proxy measures to represent the actual spot variance. The methodology presented refer to the work [5],[40].

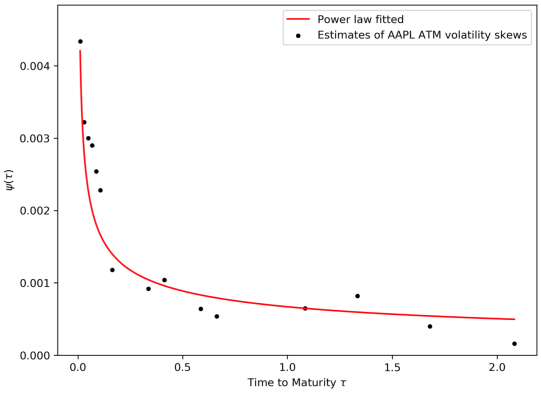


Figure 4.2.: Estimates of the AAPL volatility skews as of May 18 2020 with the fitted power-law $\psi(\tau) = A\tau^{-0.4}$, adopted from [40]

As a first step, we examine the q-th sample moment of log volatility increments at a given lag Δ. This can be defined as follows:

Let $N = \lfloor \frac{T}{\Delta} \rfloor$ be the number of observations on the time grid, where T is the total time period of observation and Δ is the time interval between observations.

The log volatility increments can be calculated as

$$m(q,\Delta) = \frac{1}{N} | \sum_{k=1}^{N} \log(\sigma_{k\Delta}) - \log(\sigma_{(k-1)\Delta}) |^q \tag{4.5}$$

where $\sigma_{i\Delta}$ represents the volatility at the i-th observation.

As shown in Figure (4.3), the plot of $m(q,\Delta)$ against $\log(\Delta)$ underlines a clear linear relationship.

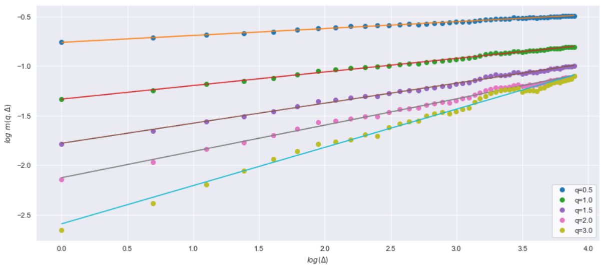


Figure 4.3.: Plot of $\log(m(q,\Delta))$ vesus $\log(\Delta)$, S&P 500

Hence, in the second step, we perform a linear regression of $\log(m(q,\Delta))$, resulting in the equation $\log(m(q,\Delta)) = \zeta_q \log(\Delta) + z_q$, where ζ_q and z_q are constants dependent on q. From this equation, we can observe that it is equivalent to expressing $m(q,\Delta)$ as $m(q,\Delta) = \Delta^{\zeta_q} e^{z_q}$.

In distribution $\log(\sigma_{k\Delta}) - \log(\sigma_{(k-1)\Delta}) \sim \log(\sigma_\Delta)$; the almost surely convergence $(m(q,\Delta)) \xrightarrow[N\mapsto\infty]{} \mathbb{E}[|\log(\sigma_\Delta) - \log(\sigma_0)|^q]$ is guaranteed by the Strong Law of Large Numbers. Combining all the results we have that:

$$\mathbb{E}[|\log(\sigma_\Delta) - \log(\sigma_0)|^q] = \Delta^{\zeta_q} e^{z_q} \tag{4.6}$$

From the plot (4.4) of ζ_q versus q it is possible to observe that $\zeta_q \sim H_q$ with $H \approx 0.13$ for S&P 500

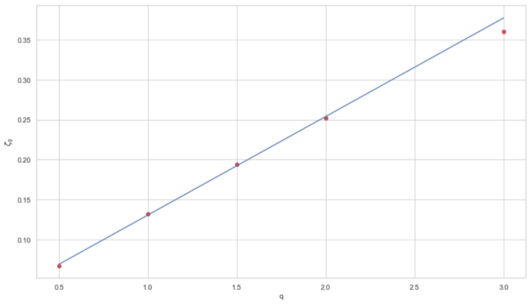


Figure 4.4.: Plot of ζ_q versus q, S&P 500

4.2. Is Rough volatility the best choice?

This chapter's goal goes beyond just presenting a comprehensive study of rough volatility models. We want to provide a study of the existing literature that critically evaluate the empirical evidence just presented.

The first question that arises is whether the observed behavior can be accurately described by a power-law decay.

Two recent empirical studies, [42],[43], have shed new light on the behavior of the at-the-money (ATM) volatility skew. Interestingly, the findings from both studies are consistent with each other and challenge the conventional belief that the ATM volatility skew follows a power-law decay.

The first noteworthy result from these studies is that the ATM volatility skew does not exhibit a precise power-law decay pattern. Additionally, both studies indicate the existence of Markovian models that can reasonably describe the ATM volatility

In the empirical study conducted in [43], an analysis of two years of S&P 500, Eurostoxx 50, and DAX data revealed an interesting observation regarding the behavior of the at-the-money (ATM) skew. The researchers propose a non-explosive parametrization to describe the ATM skew, taking into account its behavior across different maturities.

A key finding of the study is that the ATM skew generally follows a power-law shape across a significant range of maturities. However, there is an interesting exception observed for very short maturities, where the ATM skew does not display the expected blow-up or explosive behavior associated with a power-law decay.

In the study conducted in in [42] , an analysis of the S&P 500 data from 2007 to 2015 led to the proposal of a parametrization that incorporates a tamed explosion rate.

The results highlighted by Rømer further support the idea that rough volatility models may not be the most suitable for capturing the dynamics of the at-the-money (ATM) volatility term structure. In his study [44], Rømer affirm that “the fractional kernel as used in the rough Bergomi and rough Heston models is not flexible enough to separate the short and long time scale properties of volatility that is implied by quoted SPX options;” that “the fractional kernel [...] lacks flexibility in decoupling the short and long lag volatility autocorrelations;” and that “the volatility autocorrelation structure is better captured by a classical but two-factor volatility model.”

These results seem to lead to the conclusion that there are models that are not based on the assumption that volatility is rough that can recreate the empirical evidence regarding ATM term structure.

In relation to the second empirical evidence, concerns have been raised and valuable contributions have been made in two notable works, [6], [7].

In [6], Rogers proposes a different perspective that challenges the notion of rough models being highly non-Markovian. According to Rogers, in order to make predictions, it is needed to consider the entire history of the data, as rough models do not exhibit true Markovian behavior. This raises the question of what economic narrative would justify a model where knowledge of the complete historical context is necessary to predict the future.

Furthermore, the value of the Hurst parameter varies across different financial indices, indicating that there is no universal law or principle that can be universally applied to all assets.

Looking for example at a generic daily volatility estimates of the S&P500 he observed "a plot which fluctuates strongly on small time scales, but on longer time scales the level seems to be changing" [6]. Therefore his idea is to model the price process as an energetic OU process mean-reverting to a slower one for volatility:

$$dY_t = \sigma_Y dW_t' - \beta Y_t dt, \;\; dX_t = \sigma_X dWt + \lambda(Y_t - X_t)dt \tag{4.7}$$

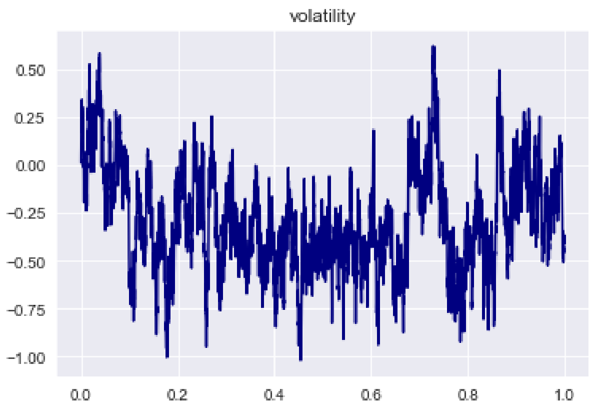


Figure 4.5.: Example with $\sigma_Y^2 = 0.625, \beta = 2.5\ \sigma_X^2 = 20, \lambda = 210$

Given that it is a bivariate Gaussian diffusion, it is considerably simpler to use. Additionally, the figure (4.6) shows that the proposed model fits the data and accurately reproduces scaling behavior as the rough volatility models.

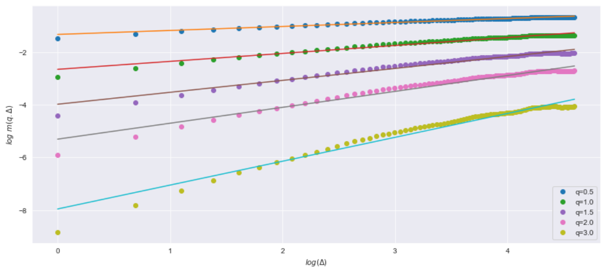


Figure 4.6.

In agreement with the views expressed by Rogers there are the results presented in [7] by Cont and Das. Their article identifies the microstructure noise as the true source of the observed irregularity in volatility. Those findings raise doubts as to whether the evidence provided by high-frequency volatility estimates supports the hypothesis of 'rough volatility'.

All their findings are based on a new non-parametric way to estimate the Hurst exponent of a path, i.e via normalized p-variation statistic: $W(L, K, \pi, p, t, X)$. All the following definitions are taken from [7].

Definition 5 (Normalized p-th variation)**.** *Let* π *a sequence of partitions of* $[0, T]$ *with* $|\pi^n| \mapsto 0$ *and* $\pi^n = \left(0 = t_0^n < t_1^n < \cdots < t_{N(\pi_n)}^n = T\right)$, $x \in V_\pi^p([0,T], \mathbb{R})$ *is said to have normlized p-th variation along* π *if there exists a continuous function* $w(x, p, \pi) : [0, T] \mapsto \mathbb{R}$ *such that*

$$\forall t \in [0,T] \sum_{\pi^n \in [0,t]} \frac{|x(t^n_{j+1}) - x(t^n_j)|^p}{[x]^p_\pi(t^n_{i+1}) - [x]^p_\pi(t^n_i)} \times (t^n_{i+1} - t^n_i) \xrightarrow[n\mapsto\infty]{} w(x,p,\pi)(t) \tag{4.8}$$

The 'normalized p-th variation statistic' is the discrete equivalent of the normalized p-th variation. Given observations on a refining time partition π^L normalized p-th variation statistic' is defined as:

$$W(L,K,\pi,p,t,X) = \sum_{\pi^K \in [0,t]} \frac{|X(t^K_{i+1}) - X(t^K_i)|^p}{\sum_{\pi^L \in [t^K_n, t^K_{n+1}]} |X(t^L_{j+1}) - X(t^L_j)|^p} \times (t^K_{i+1} - t^K_i) \tag{4.9}$$

The statistic (4.9) considers two frequencies, denoted as K and L, where L is significantly larger than K. These frequencies are referred to as the block frequency and the sampling frequency, respectively.

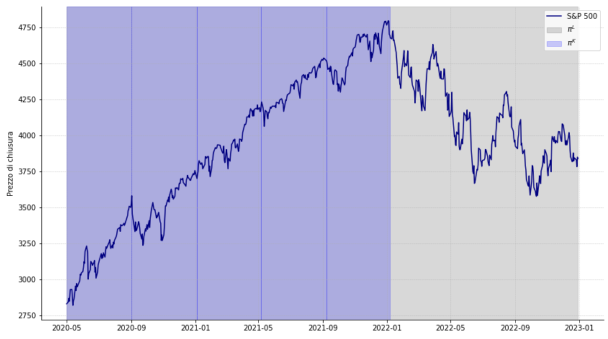


Figure 4.7.: Example of block and sampling frequency

Increasing the sampling and block frequency the statistic (4.9) converges to the normalized p-th variation (4.8).

$$\lim_{K\mapsto\infty} \lim_{L\mapsto\infty} W(L,K,\pi,p,t,x) = w(x,p,\pi)(t) \tag{4.10}$$

Definition 6 (Variation index). *The variation index of a path x along a partition sequence π is defined as the smallest $p \geq 1$ for which x has finite p-th variation along π:*

$$p^\pi(x) = \inf\{p \geq 1 : x \in V^\pi_p([0,T],\mathbb{R})\}. \tag{4.11}$$

To obtain the variation index estimator $\hat{p}_{L,K}(X)$ the approach is to calculate, for different values of p, the quantity $W(L,K,\pi,p,t,X)$ solving the following equation for $p^\pi_{L,K}(X)$,

$$W(L,K,\pi,\hat{p}^\pi_{L,K}(X),T,X) = T \tag{4.12}$$

Finally the Hurst index is calculated as

$$\hat{H}^\pi(x) = \frac{1}{\hat{p}^\pi_{L,K}(X)} \tag{4.13}$$

The tests presented utilizes the function $W(L,K,\pi,p,t,X)$ to showcase the inconsistency between the roughness observed in Realized Volatility (RV) data and instantaneous volatility data. The experiments will be run using stochastic volatility models that simulate price trajectories with varying levels of "roughness."

For the calculations of the Realized Volatility (RV) is used the following definition

Definition 7 (Realized Volatility (RV)). *The realized volatility of a price process S over time interval $[t,t+\Delta]$ sampled along the time partition π^n is defined as:*

$$RV(\pi^n)_{t,t+\Delta} = \sqrt{\sum_{\pi^n \cap [t,t+\Delta]} (X(t^n_{i+1}) - X(t^n_i))^2} \tag{4.14}$$

where $X = \log S$

The first test is conducted on the following model:

$$dS_t = \sigma_t S_t dB_t, \quad \sigma_t = \sigma_t d_t + dB_t, \tag{4.15}$$

where $\sigma_0 = 1$, B is a Brownian motion and the volatility is an Ornstein Uhlenbeck process. Simulation is carried on an interval $T = [0,1]$, with $\Delta t = \frac{1}{30000}$ observations.

The estimated roughness index for the realized volatility is significantly smaller than the roughness index for the instantaneous volatility, as shown in (4.8), where the logarithm of $W(K = 300, L = 300 \times 300, \pi, p, t = 1, X)$ is plotted against $H = 1/p$ for both the instantaneous volatility and the realized variance.

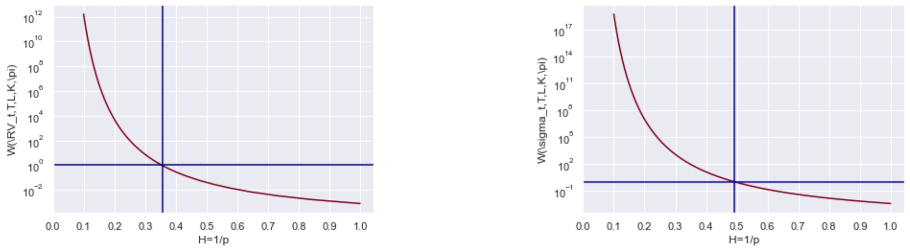


Figure 4.8.: Example of calculation of Hurst index

To more reliably analyze the first experiment's conclusion, which appears to show that realized volatility has a rougher behavior than instantaneous volatility, we run tests with the following stochastic volatility models

$$dS_t = \sigma_t S_t dB_t, \quad \sigma_t = e^{B^H_t} \tag{4.16}$$

where B and B^H are respectively a classical and a fractional Brownian motion with Hurst index $H \in (0,1)$.

In Figure (4.9), the price process, realized volatility, and the instantaneous volatility from Model (4.16) with Hurst index values of $\{0.2, 0.4, 0.5, 0.6, 0.8\}$ can be observed.

Similarly to what has been observed in the case of volatility described by a fractional Ornstein-Uhlenbeck process by Cont and Das, it is notable that for smaller H values, the instantaneous volatility appears rougher than the realized volatility. However, as the H value increases, the realized volatility exhibits significantly rougher behavior compared to the instantaneous volatility,

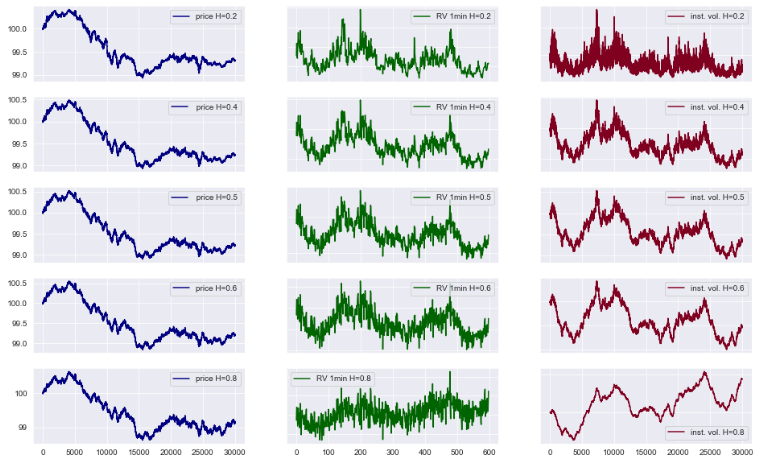


Figure 4.9.: On the left the fractional GBM in the center its realized variance and on the right the instantaneous variance for $H = \{0.2, 0.4, 0.5, 0.6, 0.8\}$

The experimental results obtained for the calculation of the Hurst index exhibit a remarkable consistency with the findings of Cont and Das in their tests involving volatility modeled by a fractional Ornstein-Uhlenbeck process. This results led them to consider realized volatility a poor estimate for the Hurst index.

4.3. CONCLUSION

One issue we wanted to examine is that of the neural calibration of SPX options.

We have chosen to continue using Markovian models for that after careful examination of both counterparts. The issues mentioned and the remedies suggested in the [6],[7] served as the basis for our conclusion. We also considered the length of time required and the complexity of implementation.

The other issues we wanted to examine is that of the neural joint calibration of SPX and VIX options. However, we lacked the necessary information to reach a decision . As a result, we dedicated the following chapter to discuss the issue of joint calibration and explore insights that will help us make a rational decision.

5
JOINT CALIBRATION OF SPX AND VIX

5.1. INTRODUCTION

The CBOE Volatility Index (VIX), often referred as the "fear gauge" of the stock market, serves as a valuable tool for understanding investors' expectations regarding the volatility of the S&P 500 index (SPX) over a 30-day timeframe. It provides key insights into market sentiment and plays a critical role in evaluating market risk [45].

Volatility indices, such as the VIX index, have additional important functions other from functioning as a market-implied measure of volatility. Cboe[1] has offered futures on the VIX since March 24, 2004, and options were added in 2006. These derivatives serve as valuable tools for market participants to hedge their exposure to volatility and safeguard their portfolios from unforeseen market movements [46].

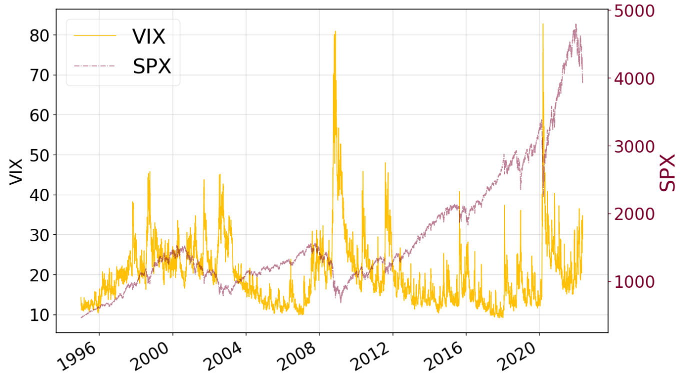


Figure 5.1.: S&P and VIX

[1]https://www.cboe.com

The SPX options market and the VIX options market are closely related because of the VIX index's structure. The market for VIX index products has continuously increased liquidity, which has highlighted the necessity to work on the joint calibration. Joint calibration of SPX and VIX options may result in improved market prediction performance. By merging information from both markets, analysts may take use of the distinct risk-neutral information available in each market, as noted in [47].

As noted by Guyon in [48], the generation of a model capable of simultaneously capturing the very steep short SPX skews and the right level of VIX implied volatilty is one of the primary challenges while working on the joint calibration. In the same article he found that the convex ordering is a necessary condition for solving the joint calibration problem. This results in a volatility process with a high mean-reversion speed and a significant negative correlation to the S&P 500 index when applied to models with continuous trajectories.

The joint calibration problem has been addressed through various methodologies, and one promising approach for studying it involves the application of neural stochastic differential equations. While the existing literature (5.2) discusses several approaches, two potential candidates that stand out for the application of neural stochastic differential equations are the Rough Volatility models and Markovian models. After thorough consideration of the motivations presented in Chapter 4, it is evident that the decision has been made to proceed with the Markovian models

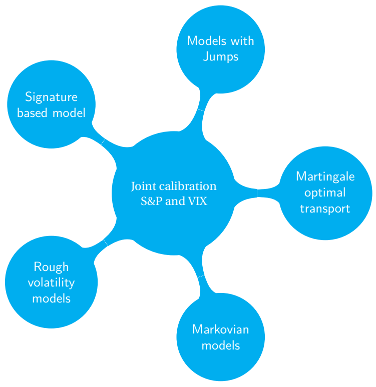


Figure 5.2.: Representation of the existing methodology for the joint calibration

Based on the analysis conducted in the previous chapter, we provide a state of the art for models that express volatility using Markovian models. Several findings provide further support for this approach. The first finding, presented in [49], examines the unskewed rough Bergomi model and reveals "a 20% difference between the [vol-of-vol] parameter obtained through VIX calibration and the one obtained through SPX. This suggests that the volatility of volatility in the SPX market is 20% higher when compared to VIX" . If the authors of the article, Jacquier et al., suggest potential data inconsistencies or arbitrage opportunities, Guyon in his work [50] highlights that the observed disparity in implied volatilities of volatility between the VIX and SPX markets should not be immediately construed as evidence of an arbitrage opportunity. Instead, it implies that the employed model may not be fully consistent with the available market data In addition to this there are also the findings highlighted in [44], which illustrate the reasons why simple rough models such as Rough Heston, Rough Bergomi, and Extended Rough Bergomi fail in joint calibration and which will be expanded in (5.2.1).

5.2. STATE OF ART

Drawing upon the comprehensive and essential overview presented in [51], which covers the existing methods used for joint calibration, we have developed state-of-the-art for Markovian models.

The initial approach involved the utilization of a double constant elasticity of variance model (CEV) as proposed in [52]. Despite the model's flexibility, it was not adequately fitting the implied volatilities of both SPX and VIX options in a joint calibration setting.

To tackle the joint calibration problem, continuous stochastic volatility models using Markovian semimartingales have been explored. However, calibration on the Heston model with stochastic volatility-of-volatility was limited to maturities exceeding four months, as VIX options exhibit lower liquidity for shorter maturities [53].

In recent developments, Jaber et al. proposed a novel model belonging to the category of continuous Markovian models in [54, 55]. This model represents volatility using a fifth-order polynomial within a single Ornstein-Uhlenbeck (OU) process.

For the state of the art two publications by Guyon merit attention.

The first paper [56], in collaboration with Mustapha, introduces a successful joint calibration approach using a neural stochastic differential equation model. This innovative methodology demonstrates the effectiveness of incorporating neural networks into the calibration process.

In the second paper [57], Guyon and Lekeufack conduct empirical and statistical analyses, along with joint calibration, for a family of models where volatility depends on the asset's paths. These models, although continuous, may not strictly adhere to Markovian properties. However, they can be transformed into Markovian models by substituting general kernels with exponential kernels.

Moreover, in the second article, a 4-factor Markovian PDV model is presented. This model achieves an impressive joint fit to the SPX and VIX smiles, indicating its robustness and accuracy in capturing market dynamics.

Finally, extremely valuable for this state of the art is the work done by Rømer's [44] which earned him the title of Risk quant of the year in 2021. Given the special interest in the following section, the key steps are presented

5.2.1. EMPIRICAL ANALYSIS OF ROUGH AND CLASSICAL STOCHASTIC VOLATILITY MODELS

Rømer conducted an extensive calibration study using SPX options data from 2004 to 2019. The focus of the study was on calibrating the rough Bergomi model, an extended version of it, and the original Heston model.

Rømer's observations revealed that the models under consideration, were unable to achieve a perfect fit across all market scenarios.

One important limitation of these models is their inability to accurately reproduce the ATM skew structure. Furthermore, these models might not sufficiently heavy-tailed to capture the short-term behavior of volatility effectively.

His primary conclusions from this first phase of the investigation, in brief, are that the fractional kernels employed in the rough Bergomi and rough Heston models lack the necessary flexibility to separate the short and long-term properties of volatility, as implied by SPX options prices. These limitations highlight the need for alternative approaches or modifications to better capture the complex dynamics of volatility in financial markets.

Following the identification of the limitations in the previous models, Rømer proceeded to conduct a secondary analysis that involved testing more advanced models. These included various two-factor volatility models and a quadratic rough Heston model.

In selecting these models, Rømer considered the insights gained from the previous analysis. It was determined that a two-factor model would be more suitable for capturing volatility dynamics. Specifically, one factor should exhibit lower negative correlation with the S&P 500 index and have a stronger influence on short expiries. Furthermore, this factor was expected to be noisier and demonstrate faster mean-reversion characteristics.

Upon comprehensive analysis of the results, it was discovered that the advanced models outperformed the initial models across a broader range of market conditions. The best outcomes were obtained by a volatility model driven by two Ornstein-Uhlenbeck processes using a non-standard transformation function. Calibrating this model to SPX options yielded almost perfect fits, and calibrating it jointly to SPX and VIX options resulted in highly satisfactory fits. Here we presented it:

$$V_t = \mu X_{1,t} + (1-\mu) X_{2,t} + c \tag{5.1}$$

$$X_{i,t} = f_{\text{hyp}}(Z_{1,t}^2 - d_i) + f_{\text{hyp}}(-d_i) \tag{5.2}$$

$$Z_{i,t} = \eta_i \delta_i (\theta_i Y_{1,t} + (1-\theta_i) Y_{2,t}) \tag{5.3}$$

$$Y_{i,t} = \zeta_{i,0}(t) + \int_0^t K_i(t-s)\,dW_{i+1,s} \quad i = 1,2 \quad t \geq 0 \tag{5.4}$$

where $\mu, \theta_1, \theta_2 \in [0,1]$, $\eta_1, \eta_2, c \geq 0$ $d_1, d_2 \in \mathbb{R}$

The choices made in this study are detailed in [44]. However, we would like to highlight three key choices that are particularly interesting.

Firstly, the selection of the gamma kernel $K(t) = e^{-\lambda t} t^{\alpha}$, where $t > 0$ and $\lambda \geq 0$. In this specific case, the kernel reduces to the exponential kernel with $\alpha = 0$. The choice of the gamma kernel guarantees asymptotically bounded variance when $\lambda > 0$. Furthermore, it provides separate control over the singular part (via α) and the long-term behavior (via λ).

The second important choice is the use of the hyperbolic transformation $f_{\text{hyp}}(x) := \sqrt{x^2+1} + x$, inspired by [58]. This approach was adopted to address the issue of heavy-tailed distributions in the resulting calibrations. The hyperbolic transformation offers a more suitable alternative to the exponential transformation, leading to more accurate estimation results.

Lastly, the use of ζ is also important. Any time-dependence exhibited by ζ can be interpreted as an expression of the historical path-dependence that has occurred before time zer

5.3. How to calculate VIX and VIX options

By definition, the VIX index is a derivative of the SPX index S, which may be expressed as

$$\text{VIX}_t = -\sqrt{2\mathbb{E}[\log \frac{(S_{t+\Delta})}{S_t} | \mathscr{F}_t]} \times 100 \tag{5.5}$$

In this case, Δ is equal to one month and $\mathbb{E}$ is the risk-neutral expectation. [59].

The VIX future at time $t \in [0, T]$ with maturity T is given by

$$F_{t,T}^{\text{VIX}} = \mathbb{E}[\text{VIX}_t | \mathscr{F}_t] \tag{5.6}$$

VIX options are formally defined as options on the VIX future at time T maturing at the same time. Therefore Calls and Puts on VIX respectively are simply:

$$\text{Call}_t^{\text{VIX}} = \mathbb{E}[(F_{T;T} - K)^+ | \mathscr{F}_t], \quad \text{Put}_t^{\text{VIX}} = \mathbb{E}[(K - F_{T;T})^+ | \mathscr{F}_t] \tag{5.7}$$

5.4. Conclusion

We are now prepared to move on to the experimental phase after conducting an extensive literature review and being convinced of the effectiveness of Markovian models in joint calibration.

6 Experiments Neural Joint and SPX Calibration

This chapter focuses on employing neural stochastic differential equations (NSDEs) to face the calibration of SPX and the joint calibration.

Calibration is important because, while easily tradable assets like call and put options have their market prices determined by supply and demand, less common and illiquid options may not have readily available quoted market prices.

Calibration's role is to determine the model parameters that best fit the available market data. These calibrated models can then be used to effectively price illiquid derivatives.

According to the approach and notation specified in the section (3.5.2), the calibration process is reduced to determining the optimal model parameters, indicated as θ_*, that minimize the loss function defined by:

$$\theta_* = \arg\min_{\theta \in \Theta} \sum_{i=1}^{M} \mathscr{L}(\mathfrak{p}(\Psi_i), \mathbb{E}^{\mathbb{Q}(\theta)}[\Psi_i]) \tag{6.1}$$

To model the trajectory of the underlying securities for the specified options S_t, we define the following Neural SDEs:

$$dS_t = rS_t dt + \sigma^S(t, S_t, V_t, \delta) S_t dB_t^S, \quad S_0 = s_0 \tag{6.2}$$

In case of the calibration we will test considering the volatility V_t following two different dynamics, the first defined as follows:

$$dV_t = \gamma^V(V_t, \alpha) dt + \Lambda^V(V_t, \beta) dB_t^V, \quad V_0 = v_0, \tag{6.3}$$

here, are simultaneously optimized the neural networks' parameters, α, β, δ as well as $\rho, v_0 \in \mathbb{R}$

The second volatility is the one described in (5.1). This second case involves assuming both ζ as neural networks, d, η, μ, α, $v_0 \in \mathbb{R}$ $\rho \in \mathbb{R}^3$, as parameters. The value of c is set to 0, as advised in [44]. As in the first case, the model's parameters and the neural network's parameters are both optimized simultaneously.

We hereafter refer to the first volatility model as the "S-V." Regarding the second model, we use similar notation to the one employed in [44]. As a result, we called it "S-M-2f-QHYP".

In both volatility models, the Brownian motions driving the dynamics of the assets are correlated with the Brownian motions driving the volatility. To incorporate and address this correlation, we can employ the Cholesky decomposition and leverage the transformed noise $\tilde{W}$ within the algorithm.

The Cholesky decomposition is a method that allows us to express a covariance matrix as the product of a lower triangular matrix and its transpose.

In the first volatility model, we have two correlated Brownian motions. We can define a vector $Z = (Z_1, Z_2)$ of independent Brownian motions, applying the Cholesky decomposition we can construct the vector $\tilde{W} = (\tilde{W}_1, \tilde{W}_2)$ as follows:

$$\tilde{W}_1 = Z_1 \quad \tilde{W}_2 = \rho_{1,2} Z_1 + \sqrt{1-\rho_{1,2}^2}\, Z_2$$

Here, $\rho_{1,2}$ represents the correlation coefficient between the two Brownian motions.

In the second volatility model, we have three correlated Brownian motions. Therefore in this case define a vector $Z = (Z_1, Z_2, Z_3)$ of independent Brownian motions and applying the Cholesky decomposition we can construct the vector $\tilde{W} = (\tilde{W}_1, \tilde{W}_2, \tilde{W}_3)$ as follows:

$$\tilde{W}_1 = Z_1$$

$$\tilde{W}_2 = \rho_{1,2} Z_1 + \sqrt{1-\rho_{1,2}^2}\, Z_2$$

$$\tilde{W}_3 = \rho_{1,3} Z_1 + \frac{\rho_{2,3} - \rho_{1,2}\rho_{1,3}}{\sqrt{1-\rho_{1,2}^2}} Z_2 + \sqrt{1-\rho_{1,3}^2 - \left(\frac{\rho_{2,3} - \rho_{1,2}\rho_{1,3}}{\sqrt{1-\rho_{1,2}^2}}\right)^2}\, Z_3$$

Here, $\rho_{1,2}$, $\rho_{1,3}$, and $\rho_{2,3}$ represent the correlation coefficients between the different pairs of Brownian motions.

Let us proceed to a more detailed examination of the algorithm's structure from [3], whose theoretical foundation is explained in (3.5.2) and which represents the basis for our experiments. The first step of the implementation consists of simulate trajectories from the neural SDEs using the tamed Euler scheme (8) and calculate a Monte Carlo estimator for $\mathbb{E}^{\mathbb{Q}(\theta)}[\Psi]$.

Typically, Euler scheme is employed for path simulations. However, when dealing, as in this case, with situations where the volatility term can grow super-linearly a modification proposed in [60],[61] is utilized (8).

Definition 8 (tamed Euler scheme)**.** *Given a generic stochastic model of the form:*

$$dX^{\theta} = \gamma(X_t^{\theta}, t, \theta)dt + \Lambda(X_t^{\theta}, t, \theta)dW(t) \tag{6.4}$$

The tamed Euler scheme reads

$$X_{t_{k+1}}^{\theta} = X_{t_k}^{\theta} + \frac{\gamma(X_{t_k}^{\theta}, t_k, \theta)}{1+|\gamma(X_{t_k}^{\theta}, t_k, \theta)|\sqrt{\Delta t_k}}\Delta t_k + \frac{\Lambda(X_{t_k}^{\theta}, t_k, \theta)}{1+|\Lambda(X_{t_k}^{\theta}, t_k, \theta)|\sqrt{\Delta t_k}}\Delta W_{t_k} \tag{6.5}$$

This scheme ensures that the moments do not experience a blow-up. It is employed for the simulation of the underlying asset and for both volatility processes.

Finally, by considering a set of N independent and identically distributed (i.i.d.) samples of (6.2) generated using the modified equation (8) the empirical approximation of $\mathbb{Q}(\theta)$ can be defined as follows :

$$\mathbb{Q}_N(\theta) := \frac{1}{N}\sum_{i=1}^{N}\delta_{S_i^{\theta}} \tag{6.6}$$

By applying the Law of Large Numbers, as N tends to infinity, the empirical expectation $\mathbb{E}^{\mathbb{Q}^N(\theta)}[\Psi]$ converges in probability to the true expectation $\mathbb{E}^{\mathbb{Q}(\theta)}[\Psi]$. This means that, on average, the empirical approximation becomes increasingly accurate as the sample size increases.

Furthermore, the Central Limit Theorem tells that:

$$\left(\mathbb{E}^{(\theta)}[\Psi] \in \left[\mathbb{E}^{\mathbb{Q}_N(\theta)}[\Psi] - z_{\alpha/2}\frac{\sigma}{\sqrt{N}}, \mathbb{E}^{\mathbb{Q}_N(\theta)}[\Psi] + z_{\alpha/2}\frac{\sigma}{\sqrt{N}}\right]\right) \mapsto 1 \text{ as } N \mapsto \infty$$

The value $z_{\alpha/2}$ is determined in such a way that the cumulative distribution function (CDF) of the standard normal distribution Z evaluated at $z_{\alpha/2}$ is equal to $1 - \frac{\alpha}{2}$ and σ represents the standard deviation, which is calculated as the square root of the variance of the random variable Ψ

As N increases, the empirical approximation $\mathbb{Q}_N(\theta)$ becomes more reliable in estimating the true measure $\mathbb{Q}(\theta)$ but we increase the computational cost. Hence the attempt is to apply variance reduction techniques. Using the martingale representation theorem is possible to define.

$$\Psi_{cv} = \Psi - \int \Xi(s)dW(s) \tag{6.7}$$

In this context, the hedging strategy $\Xi(s)$ can be obtained by parameterizing it as a neural network, incorporating a dependency on the stock path.

Specifically, $\Xi((X_{s\wedge t}^{\theta})_{t\in[0,T]}, s, \xi)$, where $\xi \in \mathbb{R}^d$, represents the neural network's input, taking into account the stock path up to time s. By employing this approach, the expectation remains unchanged, while the variance tends to zero. The algorithm is built around a two-stage optimization problem. The first stage addresses the calibration problem, while the second stage focuses on adapting the abstract hedging strategy to minimize variance. The calibration problem can be expressed as follows

$$\theta^* \in \arg\min \sum_{i=1}^{M} \left(\mathbb{E}^{\mathbb{Q}_N} \left[\Psi_i((S_t^\theta)_{t\in[0,T]}) - \int_0^T \hat{\Xi}((S_{s\wedge t}^\theta)_{t\in[0,T]}, s, \xi_i) d\hat{S}^\theta(s) \right] - \mathfrak{p}(\Psi_i)|\mathscr{F}_0 \right)^2 \quad (6.8)$$

During this stage, the algorithm optimizes the parameters of the Neural SDEs while keeping the hedging strategy parameters fixed. The formulation of the variance reduction problem is as follows:

$$\xi^* \in \arg\min \left[\mathrm{Var}^{\mathbb{Q}_N} \left[\Psi_i((S_t^\theta)_{t\in[0,T]}) - \int_0^T \hat{\Xi}((S_{s\wedge t}^\theta)_{t\in[0,T]}, s, \xi) d\hat{S}^\theta(s) |\mathscr{F}_0 \right] \right. \quad (6.9)$$

During this stage, the algorithm optimizes the parameters of the hedging strategy while keeping the Neural SDEs parameters fixed

The use of $d(\hat{S}_t^\theta) = e^{-rt}\sigma^S(t, X_t^\theta, \theta)dW_t$ and $\hat{\Xi} = e^{-rt}\Xi_t\sigma^S(t, X_t^\theta, \theta)dW_t$ is motivated by the desire to achieve better hedge results, even if it results in a potentially worse variance reduction.

In the following is reported the entire algorithm taken from [62]

Algorithm 1 Neural SDE Calibration with Empirical Risk Minimization

```
Input: $\{t_0, t_1, ..., t_n\}$ time grid; $\Psi$ vector of option payoffs; $\mathfrak{p}(\Psi)$ market option price vector
Initialization:
    - Choose initial values for the parameters $\theta \in \Theta$ for the neural SDE.
    - Set the number of epochs $N_{\text{epochs}}$.
    - Choose the number of Monte Carlo (MC) paths $N_{\text{MC}}$.
    - Set the learning rate $\alpha$ and other hyperparameters for the optimizer for the Adam optimizer.
for epoch i in 1 to $N_{\text{epochs}}$ do
    Forward pass: Generate $N_{MC}$ paths $S_t^j$ using the tamed Euler scheme with the current parameter values $\theta$.
    Calculate the empirical expected value $\mathbb{E}^{Q^{N_{\text{MC}}}}$ of the payoff function $\Psi(S_T)$ over the $N_{\text{MC}}$ paths:
    Backward pass: Update the parameter $\theta \in \Theta$ using the Adam optimizer according to the following rule
    During one epoch Freeze $\xi$, use Adam to update $\theta$, where
        $\theta = \arg\min_{\theta\in\Theta} \left\| \mathbb{E}^{Q^{N_{\text{MC}}}} \left[ \Psi_j(S^{\pi,\theta}) - \sum_{k=0}^{N_{\text{steps}}-1} \hat{\Xi}(t_k, X_{t_k}^{\pi,\theta}, \xi_i)\Delta\hat{S}_{t_k}^\pi \right] - \mathfrak{p}(\Psi) \right\|_2^2$
    During one epoch Freeze $\xi$, use Adam to update $\theta$, where
        $\xi = \arg\min_{\xi} \mathrm{Var}^{Q^{N_{\text{MC}}}} \left[ \Psi_j(X^{\pi,\theta}) - \sum_{k=0}^{N_{\text{steps}}-1} \hat{\Xi}(t_k, X_{t_k}^{\pi,\theta}, \xi_i)\Delta\hat{S}_{t_k}^\pi \right]$
end for
return $\theta$ for all prices $\Psi$
```

6.1. Computation of VIX

Exploiting that Vix is the square-root of a conditional expectation the initial idea was to proceed with the Nested Montecarlo simulation.

6.1.1. NESTED MONTE CARLO

The strategy suggested in [63] involves estimating VIX using the Nested Monte Carlo (NMC) simulation technique.

The main goal of the algorithm is to create a simulation of N values of the VIX, as expressed in Equation (5.5). By using Monte Carlo simulation and repeatedly applying the simulation procedure, the algorithm generates a nested structure of simulations, enabling the estimation of the conditional expectation and facilitating the pricing of VIX derivatives.

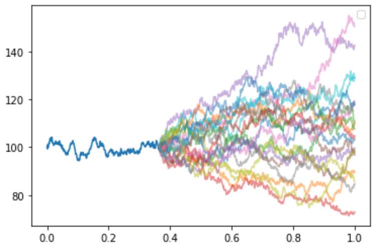


Figure 6.1.: Example of behaviour of Nested Montecarlo Simulation

Given the parameters and initial state variables, the NMC method may simulate the model's state variables across a time horizon T. The authors assume that the variance between time T and T + Δ, as well as the integral $\int_T^{T+\Delta_T} v_u du$, can be calculated from the time T state variables.

In the provided algorithm of Nested Monte Carlo from [63], the notation ψ is used to represent the model parameters, while the initial state variables are represented as γ.

Algorithm 2 NestedMonteCarlo

```
function MC($\psi$, $\gamma$, $t$)                          ▷ Provided by the model
    function NMC($\psi$, $\gamma$, $t$,$N$,$M$)
        for $i \leftarrow 1$ to $N$ do                          ▷ Outerpaths
            $\beta'_i \leftarrow$ MC($\psi$, $\gamma$, $T$)              ▷ Simulates outer path
            for $j \leftarrow 1$ to $M$ do                      ▷ Innerpaths
                $S_{i,j} \leftarrow$ MonteCarlo($\psi$, $\gamma'_i$, $\Delta$)   ▷ Simulates inner path
                $X_{i,j} \leftarrow$ InnerX($S_{i,j}$)            ▷ Inner estimate
            end for
            $Y_i \leftarrow$ Mean($X_i$)                    ▷ Averages inner estimates
        end for
        return $\beta'$, $Y$              ▷ State variables and target variable
    end function
```

The NMC (Nested Monte Carlo) method, as observed from the algorithm, can be computationally slow. This is primarily because it requires performing additional Monte Carlo simulations for each outer path. As a result, the algorithm often becomes impractical due to its time-consuming nature.

The idea therefore is to use the method of the Itô-Taylor expansions, presented in [64], for computing the VIX. The computer resources required can be significantly lowered through this method and it offers a highly accurate representation of the VIX due to its convergence properties.

The VIX represents the 30-day expectation of future volatility, which implies that it is influenced by small-time dynamics. The Itô-Taylor expansion, being a small-time expansion technique, converges rapidly in this context.

6.1.2. ITÔ-TAYLOR EXPANSION

Following the methodology used in [64] for a stochastic variance process given by

$$dV_t = \mu(V_t)dt + \sigma(V_t)dW_t, \tag{6.10}$$

it is possible to express the expectation $\mathbb{E}[f(V_t)]$ using a Taylor-like formula (6.11) by iteratively applying Dynkin's formula (as described in [65]).

The Taylor-like formula is given by

$$\mathbb{E}[f(V_t)] = \sum_{n=0}^{N} \mathscr{L}_n f(V_0)\frac{t^n}{n!} + \delta_{N+1}, \tag{6.11}$$

$\forall N \in \mathbb{N}$, $f(\cdot) \in \mathbb{R}$ is a smooth function and $\mathscr{L}$ is defined as the infinitesimal generator of the diffusion in (6.10)

$$\mathscr{L}f(v) = \mu(v)\frac{\partial f(v)}{\partial v} + \frac{\sigma^2(v)}{2}\frac{\partial^2 f(v)}{\partial v^2} \tag{6.12}$$

As demonstrated in [66], [67], [68] $\sum_{n=0}^{N} \mathscr{L}_n f(V_0)\frac{t^n}{n!}$ represents an approximation of the expectation $\mathbb{E}[f(V_t)]$.

By combining (5.5) with (6.11) and ignoring the term δ_{N+1} we have:

$$VIX_T^2 \approx \frac{1}{\tau}\int_T^{T+\tau} \sum_{n=0}^{N} \mathscr{L}_n f(V_T)\frac{(\gamma - T)^n}{n!}d\gamma \times 100^2. \tag{6.13}$$

The equation can be further simplified by factoring out the variable γ from the infinitesimal generator, as it does not appear in the generator. This simplification allows us to rewrite the integral as follows:

$$VIX_T^2 \approx \frac{1}{\tau}\sum_{n=0}^{N} \mathscr{L}_n f(V_T)\frac{(\tau^2)^{n+1}}{n+1!} \times 100^2. \tag{6.14}$$

To further enhance the computational efficiency of computing the VIX, we can adopt the approach presented in [69]. In this approach, we simplify the calculations by considering the identity function f and stopping at the first order. This means that we only need to compute $\mathscr{L}^0 f(V_T) = V_T$ and $\mathscr{L}^1 f(V_T) = \mu(V_T)$.

$$VIX_T^2 \approx V_T + \frac{\mu(V_T)}{2}\tau \times 100^2 \tag{6.15}$$

6.2. EXPERIMENTS

We have implemented the methodology described in the report using Python and various libraries such as Numpy [70], PyTorch [71], and Scipy [72]. The architectures of the experiments can be found in A.1. We made use of the GPU machines provided by the Delft High-Performance Computing Centre [DHPC2022] for our computations.

Each calibration test was conducted multiple times to evaluate the consistency and reliability of the results. The Adam optimization algorithm was employed for each iteration of the calibration process. During each iteration, a total of 4×10^4 Monte Carlo trajectories were simulated, and the time interval for each month was discretized into 8 equally spaced time steps

6.2.1. TOY TEST

As first test we replicate the example proposed in the paper. The calibration has been done with synthetic data generated using the Heston model:

$$\begin{aligned} dS_t &= rS_t dt + \sqrt{V_t} S_t dW_1 \\ dV_t &= \kappa(\theta - V_t) dt + \sigma \sqrt{V_t} dW_2 \end{aligned} \tag{6.16}$$

The parameters used are $x_0 = 1, r = 0.025, k = 0.78, \mu = 0.11, \eta = 0.68, V_0 = 0.04, \rho = 0.044$. The call prices were acquired from the Heston model by means of Monte Carlo simulation utilizing a total of 10^7 Brownian trajectories.

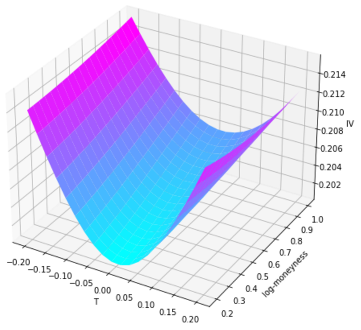


Figure 6.2.: Heston Implied Volatility

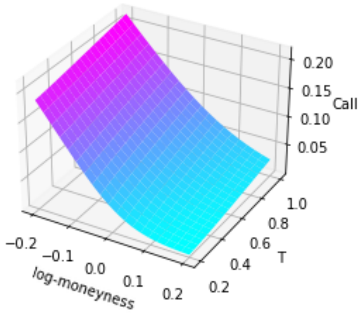


Figure 6.3.: Heston call prices

Initially, we performed a test on the calibration using six different maturities over 500 epochs. The obtained MSE error of $\mathcal{O}(10^{-8})$ aligns well with the results reported in the referenced paper. Furthermore, we conducted a benchmark test with just two maturities, which provides a consistent comparison for the calibration of real data. In this case, we performed 100 epochs and achieved an MSE error in the range of $\mathcal{O}(10^{-6})$-$\mathcal{O}(10^{-7})$.

Although slightly higher compared to the previous test, this error level is still considered satisfactory and indicative of a robust calibration.

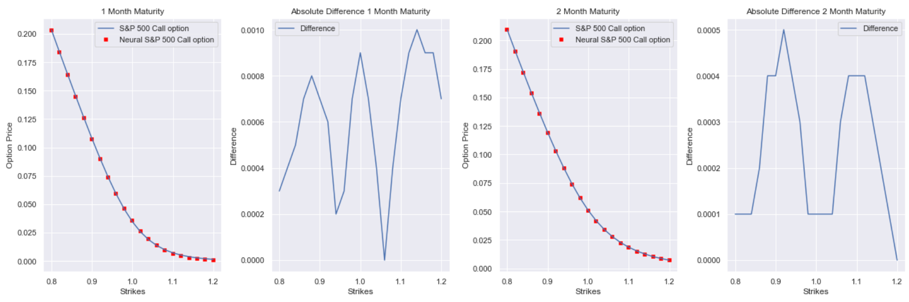


Figure 6.4.: Calibration Heston prices using S-V

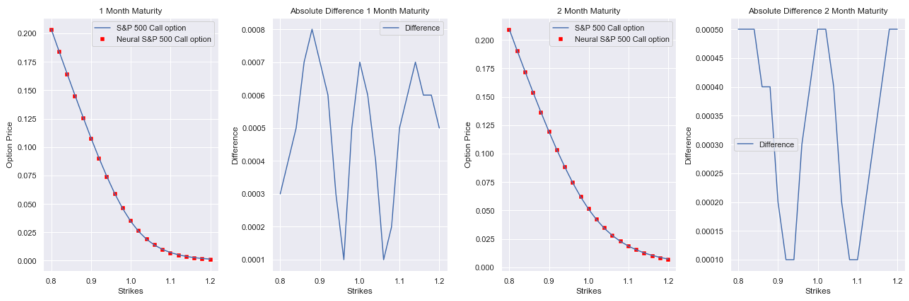


Figure 6.5.: Calibration Heston prices using S-M-2f-QHYP

6.2.2. SPX Calibration

6.2.2.1. First Test with Real Data

The first test was conducted on a data set downloaded from https:// historicaloption-data.com. The test were conducted studying the problem of the calibration of 6 bi-monthly maturity with 22 Strikes, as shown in the figure (6.6).

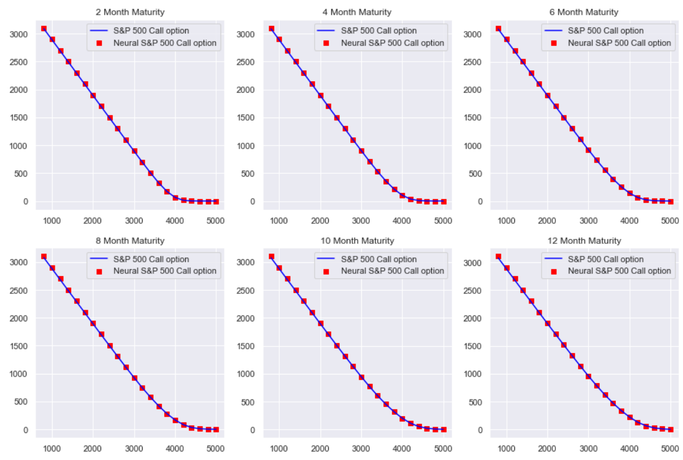


Figure 6.6.: Calibration SPX for 6 Maturity

Then, we tested for the case of just the first two maturities, as reported:

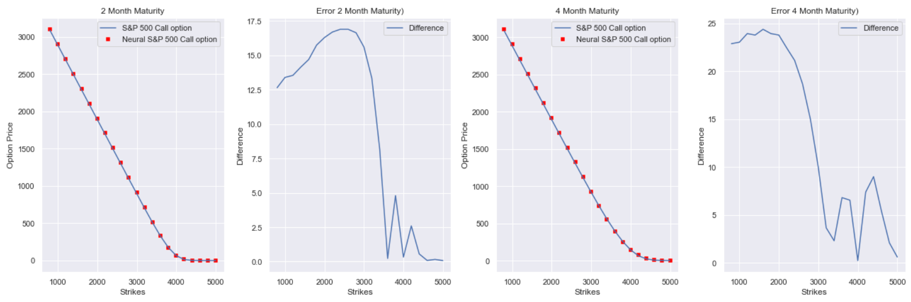


Figure 6.7.: Calibration with S-V

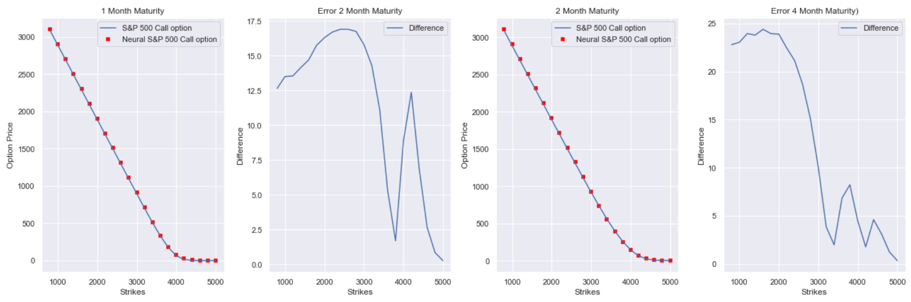


Figure 6.8.: Calibration with S-M-2f-QHYP

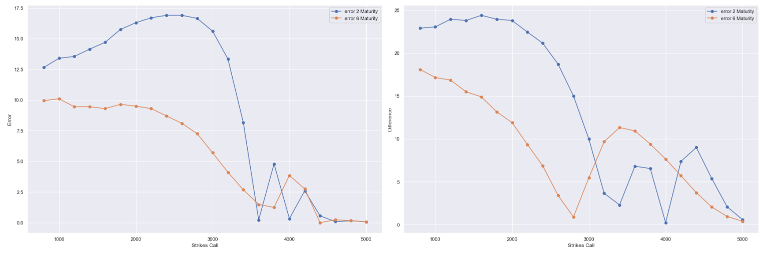


Figure 6.9.: Example of behaviour loss 6 Maturity and 2 Maturity

6.2.2.2. SECOND TEST WITH REAL DATA

The second test is carried out using the data set from the github [62]:

First we train the data using 6-Month Maturity with 23 different strikes.

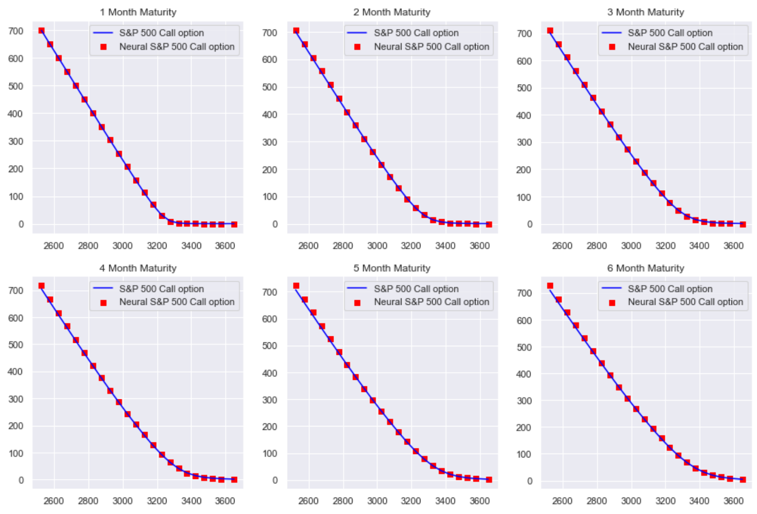


Figure 6.10.: Calibration SPX for 6 Maturity

Secondly we concentrated as before with the first two maturities, using both S-V and S-M-2f-QHYP training both 23 strikes and 46 strikes.

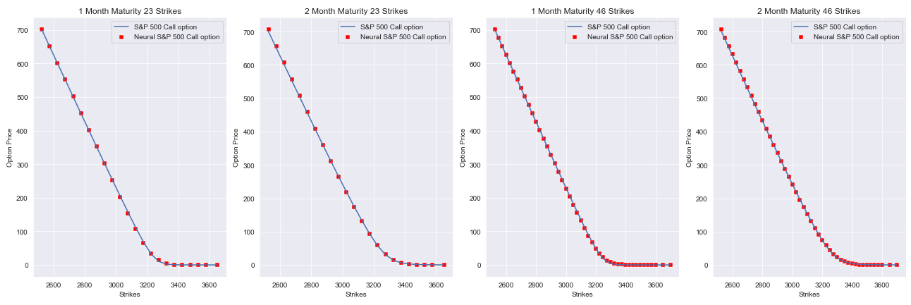


Figure 6.11.: Calibration with S-V for 23 and 46 Strikes

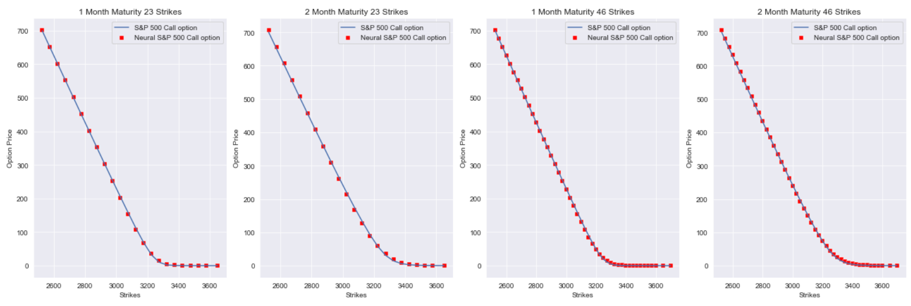


Figure 6.12.: Calibration with S-M-2f-QHYP for 23 and 46 Strikes

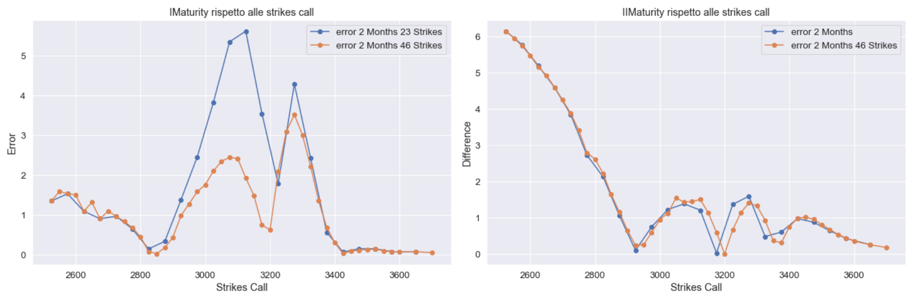


Figure 6.13.: Example of behaviour loss 23 Strikes and 46 Strikes

6.2.3. ANALYSIS OF THE EXPERIMENTS

Given the limited availability of financial data, especially in terms of option pricing, our objective was to evaluate the performance of a calibration method that produces excellent loss results in scenarios with minimal data. The obtained results from the three types of experiments, namely the 6-Maturity experiment and those with 2 Maturity with varying numbers of strikes, are consistent with the findings using the toy model.

In general, we observed a slight improvement when incorporating additional data points, such as increasing the number of strikes or extending the maturity period, as evidenced in (6.9) and (6.13).

Nevertheless, it is important to emphasize that even in the base case scenario, where both data sets were utilized, the results were already deemed satisfactory. This outcome suggests that this calibration method exhibits a remarkable adaptability and effectiveness, yielding reliable calibration outcomes even under conditions of limited data availability.

The behaviour of the loss values are consistent with those obtained in the toy test, as shown in (6.14). It is important to note that during the simulation, both prices and strikes were scaled down. Therefore, the losses presented in (6.14) need to be rescaled, considering the values used for the downsizing, taking in mind that the MSE loss function was utilized to find losses in the original problem.

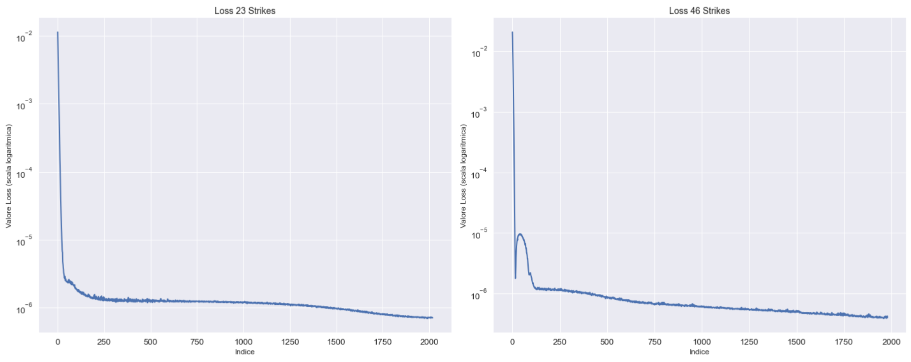


Figure 6.14.: Example loss's behaviour for 23 and 46 strikes

However, a remark regarding the behavior of the loss function arises in the experiments with a two-month maturity, particularly when using the S-M-2f-QHYP volatility model. There is a certain level of rigidity present. Although the learning process initially progresses smoothly, it tends to slow down after a certain number of epochs, as evidenced by the figures below

In order to address the issue mentioned above and improve the algorithm's performance, we conducted various tests and explored potential areas for enhancement.

One observations made during these tests was that the value object of the second optimization (6.9) often becomes very small, leading to a reduced learning capacity of the algorithm. To overcome this challenge, we propose reducing the learning rate specifically for the optimization related to this value object (6.9). By doing so, the calibration process becomes slower but can improve results due to increased sensitivity.

To illustrate the effect of a different learning rate, we provide an example in Figure (6.15).

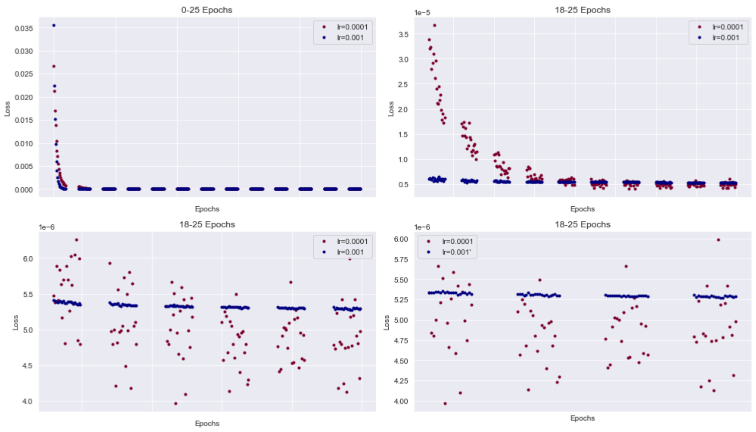


Figure 6.15.: Example of the effect of a different learning rate

Furthermore, additional techniques such as dropout or regularization could also be explored to further enhance the algorithm's performance. In terms of computational cost, the S.V model outperforms the S-M-2f-QHYP model. For instance, when considering a case with 23 strikes, a complete iteration using the S.V model typically takes less than 1 second. On the other hand, the S-M-2f-QHYP model requires approximately 1 to 2 seconds to complete a similar iteration.

6.2.3.1. JOINT CALIBRATION

The first challenge encountered is the lack of data for studying joint calibration. However, following the approach suggested in [69], it is possible to overcome this issue by generating synthetic data for joint calibration using the Heston model.

6.2.3.2. GENERATION SYNTHETIC DATA

In [8], the authors propose to represent the dynamics of the S&P 500 index following the standard analysis found in the literature (e.g., [73]; [74]; [75]; [76]) as follows:

$$dS_t = S_t(r_t + \gamma_t)dt + S_t\sqrt{V_t}dW_t^S + d\left(\sum_{n=1}^{N_t}\left[e^{Z_n} - 1\right]S_{\tau_n^-}\right) - S_t\tilde{\mu}\lambda dt \tag{6.17}$$

$$dV_t = \kappa(\theta - V_t)dt + \sigma_V\sqrt{V_t}dW_t^V + d\left(\sum_{n=1}^{N_t} Z_n^v\right) \tag{6.18}$$

For our purpose, we eliminate the jumps in both the dynamics and simplify to a straightforward Heston model.

Leveraging how the VIX is defined, in [77] and [78] is developed a formula to explicitly calculate assuming the dynamics previously presented:

$$VIX_t^2 = 100 \times \sqrt{(aV_t + b)}. \tag{6.19}$$

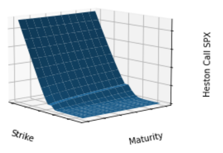
where

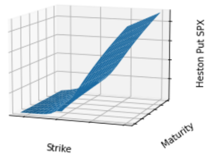
$$\begin{cases} a = \frac{1-e^{-\kappa\tau}}{k\tau}, \ \tau = \frac{30}{365}. \\ \\ b = \left(\theta + \frac{\lambda\mu_V}{k}\right)(1-a) + \lambda c, \\ \\ c = 2\left[\tilde{\mu} - (\mu_S + \rho_J\mu_V)\right], \end{cases}$$

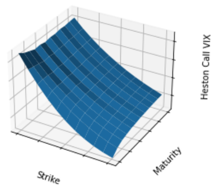
Since we reduced to the case without jumps, it reads as

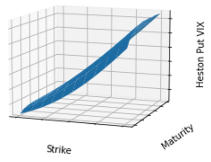
$$\begin{cases} a = \frac{1-e^{-\kappa\tau}}{k\tau}, \ \tau = \frac{30}{365}. \\ \\ b = \theta(1-a) + \lambda c, \end{cases}$$

Finally the prices of the options can be calculated using different computational methods.









6.2.4. Results

We performed multiple tests by varying the number of generated Brownian motions for the simulations and the number of maturities. The results presented here are based on 70,000 simulations and six maturities. As expected, we observed that increasing either one or both parameters led to a more accurate calibration.

The total MSE error after 20 epochs is of $\mathcal{O}(10^{-4})$ for SPX and $\mathcal{O}(10^{-1})$ for VIX. For the forward pass it takes from 3 to 5 second for each iteration.

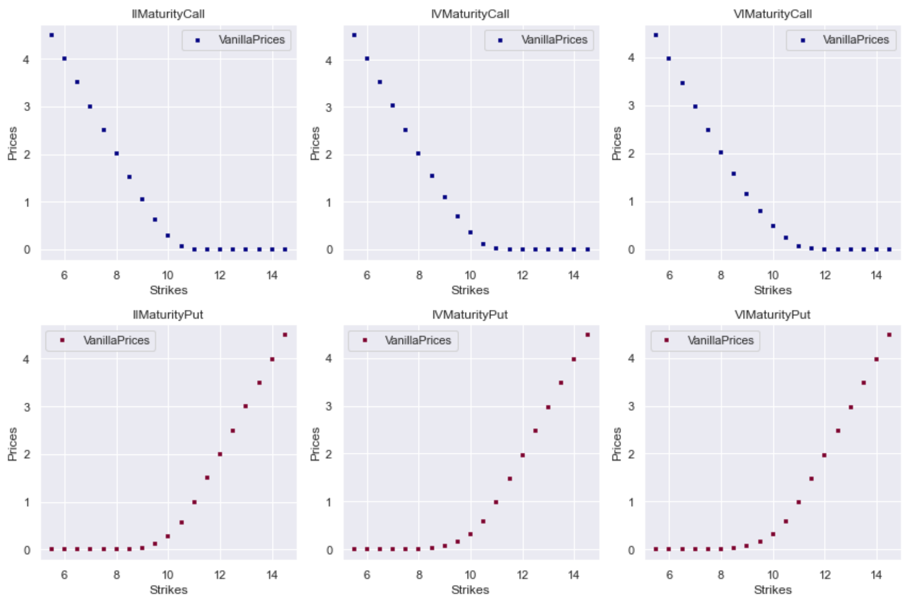


Figure 6.16.: Results of the joint calibration for SPX options

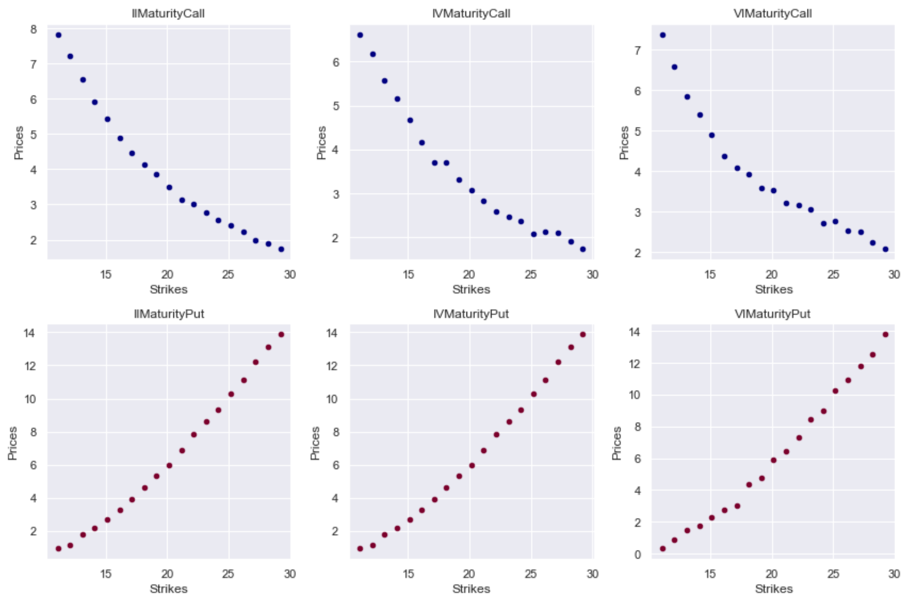


Figure 6.17.: Results of the joint calibration for VIX options

We also questioned whether the results we obtained were limited to the synthetic data

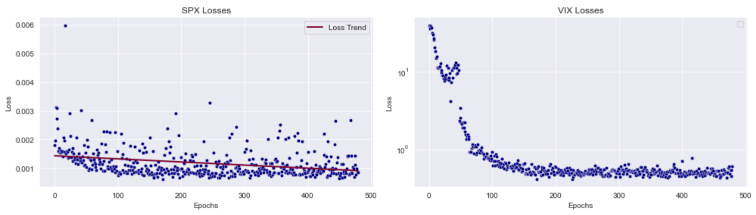


Figure 6.18.: On the left SPX's losses and on the right VIX's losses are presented

generated in the described manner. In our experiment, we utilized the 4-factor Markovian PDV model mentioned in the state of the art. We generated options data for the SPX and VIX with a 1-month maturity.

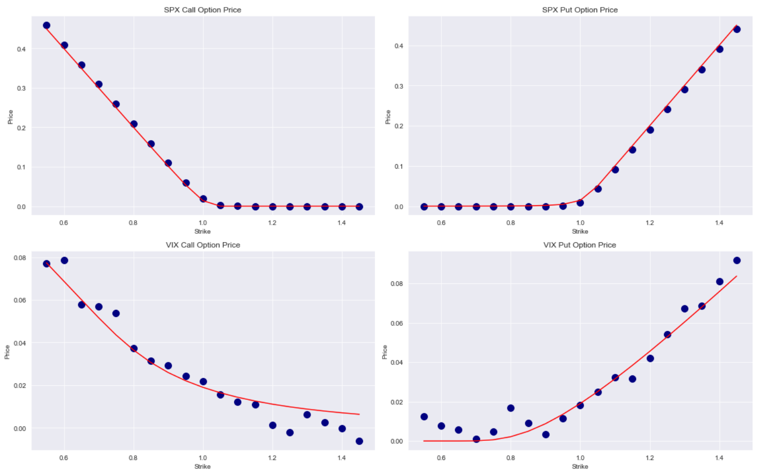


Figure 6.19.: Example of calibration using 4-factor Markovian PDV model

What we observed regarding the loss functions is that both the models exhibit a similar order of Mean Squared Error (MSE) loss during the calibration of 1-Month maturity options.

7 SYNTHETIC GENERATION OF DATA

Synthetic data is a fascinating and rapidly evolving field that has captured the attention of various disciplines [79], [80], [81], [82]. Although the concept of synthetic data may be easy we consider theoretical definition given in [11] :

Definition 9. *Synthetic data is data that has been generated using a purpose- built mathematical model or algorithm, with the aim of solving a (set of) data science task(s).*

Data scarcity poses a significant challenge for modern methodologies, especially those related to machine and deep learning, and this issue was also encountered during the course of this research. Synthetic data offers a fundamental tool for addressing this challenge in mathematical finance, and has already been widely adopted, consider, for instance, the fundamental role of Monte Carlo simulations.

The primary goal of this chapter is to examine some of the most powerful techniques for creating synthetic data with a specific focus on the financial ones. More specifically, the chapter delves into the generation of financial time series using advanced methodologies. It is not extensively proposed a discussion on the potential applications and advantages of using synthetic data to address privacy concerns; for which is advised to refer to [82], [11] .

In light of the growing interest and development of generative models, this chapter examines two specific methodologies for generating synthetic financial data. The first method employs Generative Adversarial Networks (GANs). The second approach leverages Variational Autoencoders (VAEs). For essential explanation of both this generative models we refer to the chapter 2. This chapter provides an in-depth analysis of the strengths and limitations of each approach. By doing so, the chapter seeks to provide valuable insights into the state-of-the-art techniques for generating synthetic financial data and their possible applications.

Additionally, a potential new, at least to the best of our knowledge, approach is introduced that combines the theory of Gramian Angular Fields with diffusion models.

7.1. VAE FOR GENERATION OF TIME SERIES

Generating synthetic market paths poses a significant challenge due to the fact that the distribution of paths is defined on an infinite-dimensional space, whereas current generative modelling tools are limited to finite dimensions. To overcome this challenge, the approach is to project the infinite-dimensional space of paths onto a suitable finite-dimensional space, where standard generative modelling methods can be applied.

An approach that has proven effective in projecting the infinite-dimensional space of paths to a finite-dimensional space is through the use of the signature or log-signature, as proposed in [9]. The signature transforms the continuous path into a sequence of statistics, represented by an infinite-dimensional vector of signature entries. These statistics provide a complete characterization of the original path, taking into account its time parametrization, and are described in a concise manner even by just the first few entries of the signature vector. The truncation error at level N decreases rapidly, with a factorial decay rate of $\mathcal{O}(\frac{1}{N!})$ [83].

This section is divided in two parts, the aim of the first part is to provide a concise and self-contained account of the fundamental concepts and results of rough path theory. Those who seek a more detailed understanding of this field are advised to refer to the extensive body of work of Terry Lyons and his research group, whose articles, conference presentations, and books are the basis upon which this section is founded and which offers a more comprehensive discussion of the theory of rough paths.

The second involves analyzing the experiment using the methodology described in [9].

7.1.1. SIGNATURE OF A PATH

The concept of signatures, as introduced in the reference [83], is based on the framework of rough path theory. The signature is a powerful method to represent data. At a mathematical level the signature is a faithful transform of a multidimensional time series.

Definition 10 (Signature of a path, [84]). *Let $E := \mathbb{R}^d$ and J be a compact time interval. Let $X : J \mapsto E$ be a continuous path of finite p-variation such that the following integration makes sense.*

The signature $S(X)$ of X over the time interval J is defined as follows $X_J = (1, X^1, \dots, X^n, \dots)$, where for each integer $n \geq 1$,

$$X^k = \int \dots \int_{u_1 < \dots u_k, u_1, \dots, u_k \in J} dX_{u_1} \otimes \dots \otimes dX_{u_k} \tag{7.1}$$

The truncated signature of X of order n is denoted by $S^n(X)$, i.e. $S^n(X) = (1, X^1, ..., X^n)$, for every integer $n \geq 1$.

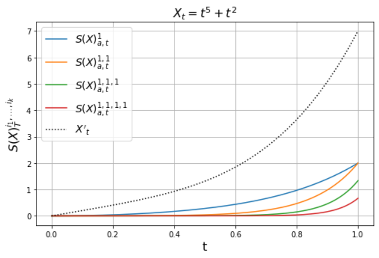


Figure 7.1.: Example of Signature at the 1st level

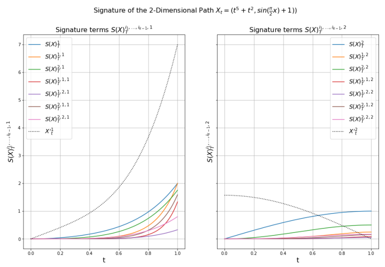


Figure 7.2.: Example of Signature at the 2nd level

7.1.1.1. Origin

The concept of a signature of a path is closely related to the idea of a controlled differential equation in the form of

$$dY_t = g(Y_t)dX_t, \qquad Y_0 = y_0, \tag{7.2}$$

In this form, the state of a complex system, represented by Y, evolves over time as a function of its present state and the incremental changes in the driving signal X, which is controlled by the function g. To solve this equation, one can use Picard iteration, where a sequence of $\{Y^n : [0,T] \mapsto W := \mathbb{R}^d\}_{n\geq 1}$ is defined recursively for every $t \in [0,T]$ assuming smooth paths. Specifically, the starting point of the sequence is $Y^0_t = y_0$, and the following terms are obtained by iteratively integrating the previous term up to time t, [84].

$$Y^0_t = y_0; \quad Y^{n+1}_t = y_0 + \int_0^t g(Y^n_s)dX_s \tag{7.3}$$

By using the fixed point argument, it can be shown that Y^n converges in 1-variation norm and the limit of Y^n is a solution to (7.2) under some appropriate conditions.

Let $X : J \mapsto E := \mathbb{R}^d$ be a d-dimensional path where J is a compact interval. The set of any continuous path $X : J \mapsto E$ of finite p-variation is denoted by $\mathcal{V}^p(J,E)$. Assuming as done, in [84], $X \in \mathcal{V}^1([0,T],E)$ and $g : W \mapsto L(E,W)$ is a bounded linear map, it can be deduced that

$$Y^n_t = \Big(I + \sum_{k=1}^{n} g^{\otimes k} \int \cdots \int_{u_1<\ldots<u_n} dX_{u_1} \otimes \ldots \otimes dX_{u_n}\Big) y_0 \tag{7.4}$$

and Y^n_t converges to Y_t as n tends to infinity, and Y_t yields the following representation

$$Y_t = \Big(I + \sum_{k=1}^{\infty} g^{\otimes k} \int \cdots \int_{u_1<\ldots<u_n} dX_{u_1} \otimes \ldots \otimes dX_{u_n}\Big) y_0 \tag{7.5}$$

where $u_1 \ldots u_n \in J$ and $g^{\otimes k}(x_1 \times \cdot x_k) = g(x_1) \cdot \cdot g(x_k)$

The solution of the problem is exclusively determined by $g^{\otimes k}$ and the set of iterated integrals in X.

In a broader sense, based on Chen's finding [85], it can be stated that for any given smooth function g, which does not necessarily have to be linear, the solution is entirely determined by the signature of X.

Therefore, as observed as [86], the signature can be seen as a reliable and powerful tool with the ability to accurately describe a system's dynamics as it responds to an input signal. In light of the fact that the signature does not necessitate the specification of any precise set of parameters, this understanding provides a promising indication of why it is often viewed as a non-parametric strategy for summarizing a path.

Furthermore, this suggests that a similar approach to that of reservoir computing can be applied to comprehending the data. This involves analyzing the data's impact on the signature of a specific segment of the data. By doing so, it becomes possible to characterize the data themselves.

The properties that will be discussed in the sections that follow are the most important for comprehending the significance of the signature path in the context of the article [9].

Proposition 2. *Signatures offer a way to represent functions on the space of curves, without explicitly modeling or parameterizing the curve itself. In contrast to Fourier transforms and wavelets, which use a functional basis to model a curve as a linear combination, signatures provide a basis of functions for a functional on path space, [87].*

Theorem 3 (**Uniqueness of signature**, [88])**.** *Let $X \in \mathcal{V}^1([0,T], E)$. Then the signature of X determines X up to the tree-like equivalence. .*

According to Levin et al. ([84]), a tree-like path can be heuristically interpreted as a null-path under a specific control. The definition of tree-like equivalence is precisely given in Hambly and Lyons' work ([88]).

Proposition 3 (**Invariance under reparametrisation,** [89])**.** *Let $X : [0,T] \mapsto V$ be a piecewise smooth path, and let $\varphi : [0,T] \mapsto [0,T]$ be a reparametrisation. Then, the signature of X coincides with the signature of $X \cdot \varphi$*

To better understand, it is convenient to think about the connection with the controlled differential equations explained in (7.1.1.1). Indeed if one moves along the control faster he simply goes along the response faster. Therefore looking at the path and after at the response over all the interval does not matter how quickly you move, you get the same response. This factors out an infinite dimensional group of symmetry.

As for instance, when representing an object, such as a numerical value, using time series data generated at different speeds, it results in distinct time series representations. However, the property of invariance under reparametrization offers a solution to this problem.

Proposition 4 (Robustness to missing Data and irregular sampling, [90])**.** *The signature feature set is capable of handling time series with varying lengths and unequal time spacing. The signature feature transformation provides a fixed dimension descriptor for any d-dimensional time series, regardless of its length or time spacing.*

This property is particularly valuable in finance where data can often be incomplete or difficult to obtain at regular intervals

Proposition 5 (Universality, [91])**.** *The universal nonlinearity property asserts that for any given $\epsilon > 0$ and under certain assumptions, there exists a linear function L that can be used to approximate any function f mapping data x to labels y.*

$$||f(x) - L(Sig(x))|| \leq \epsilon \tag{7.6}$$

This property holds due to the fact that linear functionals on the signature are densely defined in the set of functions on x

Remark 1 (Itô-Lyons Map, [92])**.** *In stochastic process theory, the integral map $(X;Y) \mapsto \int_0^T Y_s dX_s$ lacks continuity. This means that we cannot define a meaningful topology on the space of paths such that the integral map is continuous [[93], Prop 1.1]. However, by extending the given path X with its rough path structure, denoted as $\mathscr{X} = (X;\mathbb{X})$, and considering the integration of Y with respect to $\mathscr{X}$, we can overcome this limitation.*

In this scenario, we observe that, according to the rough path metric, the integral becomes a continuous function. This continuity of the integral is proven in [[92], Thm 3.2.2], [[93], Thm 4.10].

Consider the following problem:

$$dY_t = g(Y_t)d\mathscr{X}_t, \quad Y_0 = y_0, \tag{7.7}$$

As suggested in [86], the Itô-Lyons map can be represented as the diagram (7.3), where $\tilde{\Gamma}_g : \mathbb{X} \mapsto (Y;Y')$ and $\Gamma_g : X \mapsto Y$ represent the solution map of (7.7) and the solution map of the problem reduced to only X, respectively. In this context, Y' can be interpreted as a 'derivative' of Y.

$$\begin{tikzcd}
(X,\mathscr{X}) \arrow[r, "\tilde{\Gamma}_g"] & (Y,\mathscr{Y}) \arrow[d, "\text{Project}"] \\
X \arrow[u, "\text{Lift}"] \arrow[r, dashed, "\Gamma_g"'] & Y
\end{tikzcd}$$

Figure 7.3.: Ito-Lyons map diagram

This property may have several applications, one of which is connected to the hedging strategy. Hedging strategy might be considered as an integral versus the pricing process. Hence, this property ensures that two prices exhibiting similar signatures will demonstrate comparable performance when subjected to the same hedging strategy

The memory cost of the depth-N signature is not dependent on the series length n, but on the dimensionality of the problem, and can be expressed as $(d^{N+1}-1)/(d-1)$. To address this issue, log-signatures can be used as they retain all the properties of the signature, while being of lower dimensionality when truncated to the same degree.

Definition 11 (log-signature, [94])**.** *If $\gamma_t \in E$ is a path segment and S is its Signature then*

$$S = 1 + S^1 + S^2 + \cdots \quad \forall i,\ \ S^i \in E^{\otimes i}$$

$$\log(1+x) = x - \frac{x^2}{2} + \cdots$$

$$\log(S) = (S^1 + S^2 + \cdots) - \frac{(S^1 + S^2 + \cdot\cdot)^2}{2} + \cdots \tag{7.8}$$

$\log(S) = (S^1 + S^2 + \cdots) - \frac{(S^1+S^2+\cdots)^2}{2} + \cdots$ *is referred to as the log Signature of γ.*

Defining with d the path dimension and k the truncated degree of the signature we have the following:

k \d	2	3	4	5	6	7
1	3 **2**	4 **3**	5 **4**	6 **5**	7 **6**	8 **7**
2	7 **3**	13 **6**	21 **10**	31 **15**	43 **21**	57 **28**
3	15 **5**	40 **14**	85 **30**	156 **55**	259 **91**	400 **140**
4	31 **8**	121 **32**	341 **90**	781 **205**	1555 **406**	2801 **728**

Table 7.1.: The left integer is the signature dimension whereas the right integer (in bold) is the log-signature dimension [95]

Definition 12 (Expected signature,[84])**.** *Given a probability space $(\Omega, P, \mathcal{F})$, X is a E-valued stochastic process. Suppose that for every $\omega \in \Omega$, the signature of $X(\omega)$ is well defined a.s and under the probability measure P, its expectation denoted by $\mathbb{E}[S(X(\omega)]$ is finite. We call $\mathbb{E}[S(X(\omega)]$ the expected signature of X*

The expected signature of a random path plays a similar role as that the moment generating function of a random variable does. In [95], by Proposition 6.1 in [96], the following result is presented:

Theorem 4. *Let X and Y be two $\Omega(J,\mathbb{R}^d)-$ valued random variables. If $\mathbb{E}[S(X)] = \mathbb{E}[S(Y)]$, and $\mathbb{E}[S(X)]$ has infinite radius of convergence, then $X = Y$ in the distribution sense.*

The fundamental question surrounding the expected signature, often known as the moment problem, namely whether the expected signature unequivocally determines the probability distribution of a random signature, has thus been resolved under proper conditions.

Now that a better understanding of the rough paths theory has been acquired, it becomes clearer why it is used. The properties of the theory allow for the encoding of important information about the path, and the process of vectorization, combined with the characteristics of the expected signature, facilitates the definition of a concept of distance. This distance measure enables the training of models to generate new paths and improve their performance.

Remain to establish which kind of evaulation metrics use

7.1.1.2. WHICH METRICS USE FOR THE EVALUATION

In [9] is suggested to make use of Two-sample tests as a suitable approach. These tests, as specified in the article, are used to assess whether two given data samples are from the same distribution, without describing the nature of that distribution.

The test specifically evaluates if $p \neq q$ given i.i.d. observations $\{x_1 ..., x_m\}$ and $\{y_1, ..., y_n\}$ from p and q, where p and q are the probability measures of the two random variables X and Y

Compared to distributional metrics and divergences, two-sample tests exhibit distinctive characteristics that make them more suitable as performance evaluation metrics. Distributional metrics often require inferring the underlying distribution from the available data sample before being applicable. In contrast, two-sample tests offer a direct assessment of the dissimilarity between two data samples without relying on explicit distributional assumptions.

Growing popularity between the two sample tests has the Mean Maximum Discrepancy distance proposed in [97]. The MMD is based on the largest divergence in expectations over predefined set of functions $\mathcal{H}$.

$$\mathrm{MMD}[\mathcal{H}, p, q] := \sup_{f \in \mathcal{H}} \left(\mathbb{E}_{X\sim p}[f(X)] - \mathbb{E}_{Y\sim q}[f(Y)]\right) \tag{7.9}$$

The choice of $\mathcal{H}$ is critical; in this case, considering all functions in the unit ball of a Reproducing Kernel Hilbert Space (RKHS) as $\mathcal{H}$ enables distinguishing between metrics p and q.

This approach solve, as observed in [9], the problem that the underlying distribution of the data generating model is not known explicitly.

However, this metric does not tackle the problem of the potential non-universality of features to control for in the generated time-series and the problem of generalising distributional metrics to path space

Among the different methods available for Maximum Mean Discrepancy (MMD), the authors chose the approach based on the signature, which was introduced in [98] .

Given a sample of real paths $Y_1, \ldots, Y_n$ is collected and n sample of generated paths $X_1, \ldots, X_n$ the test statistic $T(X_1, \ldots, X_n; Y_1, \ldots, Y_n)$, proposed in [98], is computed using the following procedure:

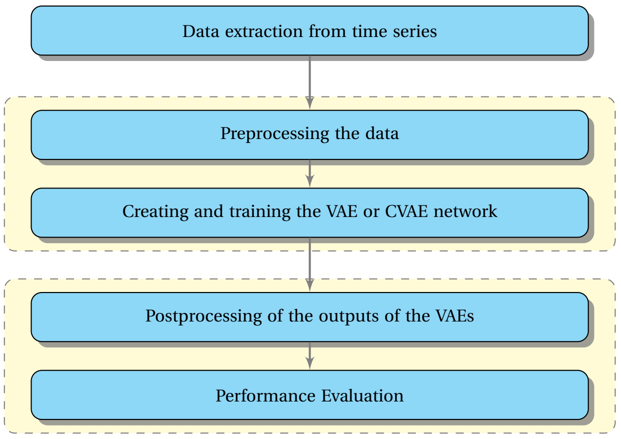
$$T(X_1,.,X_n;Y_1,.,Y_n) := \frac{1}{n(n-1)} \sum_{i,j,i\neq j} k(X_i, X_j) - \frac{2}{n^2} \sum_{i,j} k(X_i, Y_j) + \frac{1}{n(n-1)} \sum_{i,j,i\neq j} k(Y_i, Y_j) \tag{7.10}$$

where $k(\cdot,\cdot)$ is signature kernel; the recursive structure allows for an efficient and robust calculation.

The test in [9] proceeds by determining a threshold value based on a fixed confidence level, α, which lies between 0 and 1. The threshold is computed as $c_\alpha := 4\sqrt{-n^{-1}\log\alpha}$. The generative model will be considered realistic with a confidence level of α if $T^2 < c_\alpha$.

7.1.2. EXPERIMENTS

For conducting the experiments we followed the approach in [9] schematized as follows:


Data extraction from time series
Preprocessing the data
Creating and training the VAE or CVAE network
Postprocessing of the outputs of the VAEs
Performance Evaluation

where CVAE stands for Conditional Variational Autoencoder, which is the same as VAE but conditioned on some current market conditions.

It is worth noting that after dividing the original partitions into weekly or monthly path segments, the collected data samples are turned into log signatures using the already implemented esig, tosig and iisignature libraries and the lead-lag transformation method. The lead-lag transformation is defined as follows:

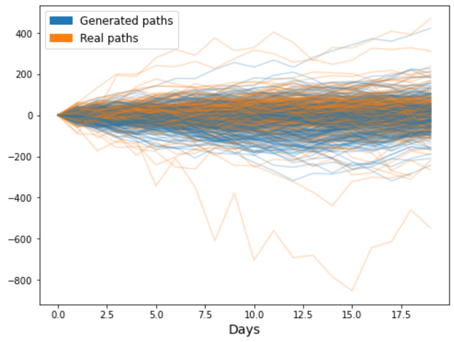
Definition 13 (Lead-lag path, [9]). *The lead-lag transformation of* $\{X_{t_i}\}_{t_i \in D}$ *is defined by the 2d-dimensional continuous path*

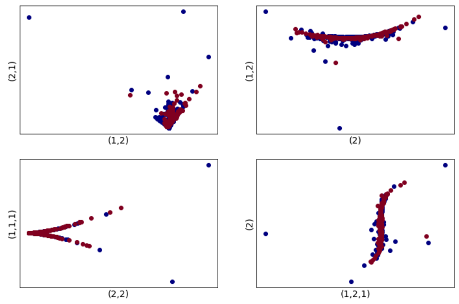
$$X_t^D = (X^{D,b}, X^{D,f}) := \begin{cases} (X_{t_k}, X_{t_{k+1}}), & t \in \left[\frac{2k}{2nT}, \frac{2k+1}{2nT}\right) \\ (X_{t_k}, X_{t_{k+1}} + 2(t-(2k+1))(X_{t_k+2} - X_{t_k+1})), & t \in \left[\frac{2k}{2nT}, \frac{2k+\frac{3}{2}}{2nT}\right) \\ (X_{t_k} + 2(t-(2k+\frac{3}{2}))(X_{t_k+1} - X_{t_k}), X_{t_{k+2}}), & t \in \left[\frac{2k+\frac{3}{2}}{2nT}, \frac{2k+2}{2nT}\right) \end{cases} \tag{7.11}$$

The component $X^{D,b}$ *is the backward or lag component, and* $X^{D,f}$ *is the forward or lead component. The signature of the lead-lag transformation will be denoted by* $X_{0,T}^{D,<\infty}$

The VAE is then trained to generate new data. However, because those data are in the signature form, they should be postprocessed before they can be evaluated.

The first test consist in generate new data using monthly paths of S&P in the period from 01-01-2000 to 01-01-2022. The projections of the high-dimensional log-signatures onto various two-dimensional subspaces are visually represented on the right plot.



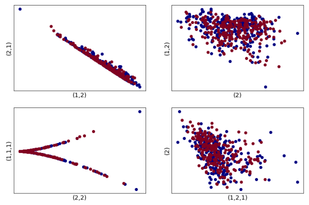


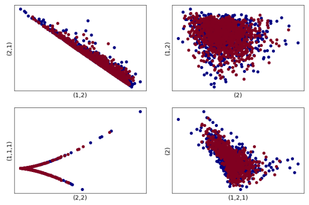
After repeatedly running the algorithm, we found that it takes approximately 30-35 minutes to complete on a MacBook Air. The table below (7.2) displays the confidence levels at which the signature-based MMD test detects a transition from similarity to dissimilarity between the two sample distributions. Higher confidence levels indicate greater similarity between the generated and original samples. The experimental results align with the results in [9].

	Weekly signature paths	Monthly signature paths
MMD signature confidence level	99.95 %	99.87 %

Table 7.2.

We created two sets of data from the S-M-2f-QHYP model, one identical to the prior test on S&P and the other substantially larger. We discovered that the generated paths had a very comparable quality based on our observations. This experiment, as done in [9] with Rough Bergomi model, aims to showcase the optimal performance of the method already on a small number, as in the case of S&P 500 data. Projections of high-dimensional log-signatures onto 2D subspaces are shown for both experiments, with the first on the left and the second on the right.





We also tested the model's capability to simulate future market behavior. The results were modest. However, we believe that by imposing constraints during the training process, such as incorporating market sentiment, better results can be achieved.

For instance, in Figure (7.4) is tested the capability of the model to simulate the future market behavior between 01-01-2022 to 01-02-2022, by estimating the error compared to the actual market behavior. It is evident that the algorithm failed to capture the upward movement in the market during that period. We think that by incorporating sentiment analysis or other relevant factors into the training process, the algorithm can be improved to better capture such movements.

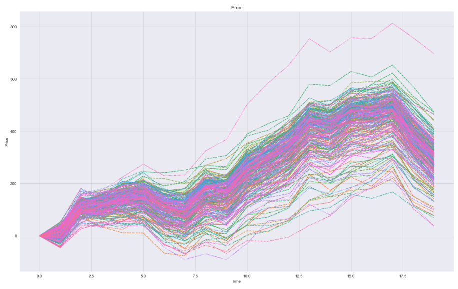


Figure 7.4.: Error generated paths

It is important to note that the primary purpose of the presented algorithm is to generate synthetic data that closely resembles market data. This synthetic data can be used for training other algorithms, such as the deep hedging algorithm. However, we think it is possible to modify and refine the algorithm to generate data that can be useful for other type of studies.

We conducted a similar test on the VIX, focusing on the same period. Our goal was to study whether the algorithm was able to capture the dependencies between the movements of the VIX and SPX. Comparing all the generated paths was challenging, so we focused on the paths that correspond to the respective means of all paths for both the VIX and SPX. Additionally, we considered the scenario where the selected paths minimize the absolute error. As shown in Figure (7.5), we found that the SPX and VIX's synthetic data combined show that interact in a way that is consistent with real-world observations.

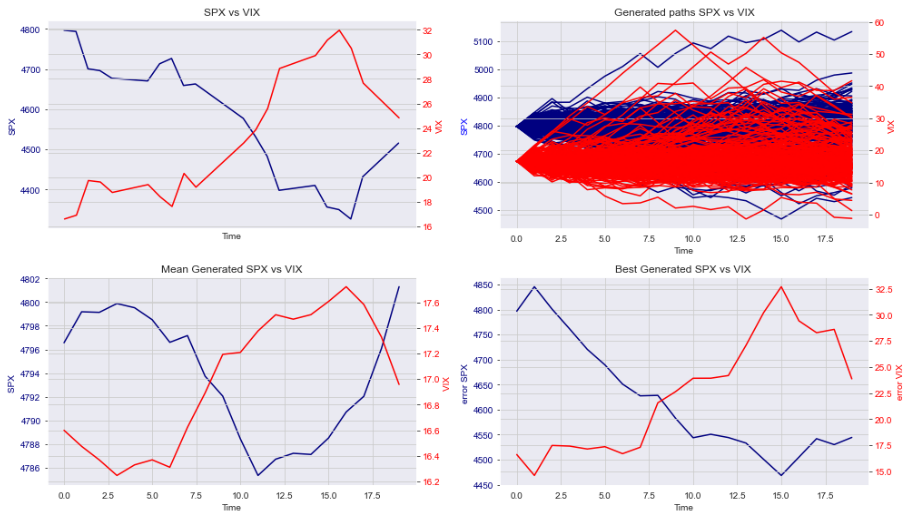


Figure 7.5.: Synthetic SPX and VIX

Finally we tested this approach on three different metrics proposed in [99]

The first two, the Discriminative Metric and the Predictive Metric, both work with a two-layer LSTM. The third metric is the Independence Metric. These metric scores are used to assess the algorithms' discriminative and predictive abilities, as well as the distance and correlation between generated data and the true distribution. In all cases, lower values yield better results. The following arrangements were taken into consideration when conducting the tests; for the predictive metric, we used the data after applying the min-max scalar. No changes were made in the other circumstances. The data used are those that were created without any condition; as a result, their initial values are all 0 for each generated path. The test were conducted ten times and in the Figure (7.6), (7.7)are reported the boxplot regarding the Discriminative and Predective metrics. Regarding the independence metrics, given the structure of the pre-posed algorithm and the available data tend to always give the same value equal to 0.0867

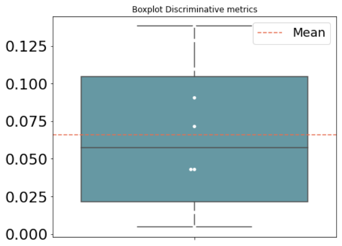


Figure 7.6.: Box plot discriminative

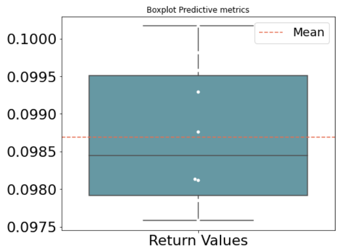


Figure 7.7.: Box plot Predective

The values provided by these tests are interesting since they are consistent with values found in the article for various data types and methods [99]. However, to provide an accurate comparison, we would need to test the other algorithms with the same quantity of data. Indeed, the tests in [99] were performed on number sequences of the order of 10^3 and involved 5 channels. In our case, we generated four different channels , presented in the Figure (7.8), representing the close, adjusted, high, and low prices of the S&P index for the same period of the first experiment. However, the number of sequences available for analysis was less than 300.

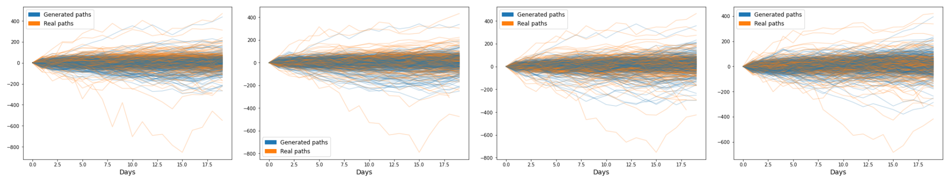


Figure 7.8.: From left to right Close, High, Adjusted, Open SPX index

7.2. GAN For Generation of Time Series

This section aims to investigate the combined use of generative models, such as Generative Adversarial Networks (GANs), with optimal transport problems. Specifically, we are interested in the method presented in the the article [10], where is proposed a new adversarial object that draws from the theory of optimal transport and introduces the mixed Sinkhorn divergence to address the issue of bias. To provide a clear and rigorous understanding of this approach, a brief introduction to the mathematical objects involved will be presented in the first part. In the second part, the method proposed in the article will be examined in greater detail, including testing it and exploring the challenges related to generating synthetic financial data

7.2.1. An introduction to optimal transport

For this section, relevant sources include the two books by C. Villani ([100], [101]), [102] and [103] that cover the topic comprehensively.

All started when Gaspard Monge began by asking himself, "How do I fill a hole with dirt as efficiently as possible?"

In other words he sought a solution that would efficiently transfer the contents of one density distribution to another, ensuring that the overall mass is conserved. This problem has since become known as the Monge's optimal transport problem.

The problem considers two densities $f, g \geq 0$ on $\mathbb{R}^d$, with $\int_{\mathbb{R}} f(x) = \int_{\mathbb{R}} g(y) = 1$, the task is to find a map T that push the density f onto the density g.

Mathematically it reads as follows:

$$\int_E g(y)dy = \int_{T^{-1}(E)} f(x)dx = 1 \tag{7.12}$$

At the same time the problem aims to minimize the quantity

$$C(x) = \int_{\mathbb{R}^d} |T(x) - x| f(x)dx \tag{7.13}$$

If in the equation (7.12) found the map $T : \mathbb{R}^d \mapsto \mathbb{R}^d$ that does exactly what required, the second equation (7.13) guarantees the efficiency by minimizing the cost function $C(T)$. In this context, the term $|T(x) - x|$ represents the transportation cost associated with moving mass from one location to another, while $f(x)dx$ quantifies the amount of mass being moved. By minimizing the cost function $C(T)$, we are essentially minimizing the total cost incurred in transporting all the mass.

The problem remained unsolved for centuries, as it was unclear whether a minimizer existed and how to characterize it. Only with Kantorovich's work was it placed into a suitable framework, which allowed for approaching and solving it. Solutions have been found, and by incorporating more general measures, spaces, and cost functions $c(x, y)$, instead of the Euclidean distance $|x - y|$ the problem has been broadly extended.

Kantorovich's innovative concept was to address Monge's problem by relating it to linear programming. Instead of assigning a specific destination, $T(x)$, to each particle initially located at x, Kantorovich proposed assigning a number of particles from x to y for each pair (x, y). This formulation allows for more flexible movements, as particles originating from a single point x can potentially move to different destinations y.

7

Considering a probability measure μ on $\mathscr{X}$, a probability measure ν on $\mathscr{Y}$, and a cost function $c : \mathscr{X} \times \mathscr{Y} \mapsto \mathbb{R}$ he formulated the classical optimal as:

$$\min_{\pi \in \Pi(\mu,\nu)} \mathbb{E}^{\pi}[c(x, y)] \tag{7.14}$$

where $\Pi(\mu, \nu)$ is the space of transport plans (couplings) between μ and ν. In addition, it is possible to define a subspace that will become fundamental later, namely the causal transport plan

Definition 14. *A transport plan $\pi \in \Pi(\mu, \nu)$ is called causal if*

$$\pi(dy_t|dx_1, \cdots, dx_T) = \pi(dy_t|dx_1, \cdots, dx_t) \;\; \forall \; t = 1, \cdots, T - 1. \tag{7.15}$$

The set of all such plans will be denoted by $\Pi^K(\mu, \nu)$.

The minimizers are defined optimal transport plans between μ and ν.

Considering now

$$\mathscr{W}_c(\mu,\nu) = \inf_{\pi\in\Pi(\mu,\nu)} \{\mathbb{E}^{\pi}[c(x,y)]\} \tag{7.16}$$

$\mathscr{W}_c$ is a distance and in particular for $c(x,y) = ||x-y||_1$ correspond to , the Wasserstein-1 distance. Solving equation (7.16), where μ and ν are supported on finite sets containing n elements, has a computational complexity that is super cubic in n according to references [104], [105], and [106]. This means it becomes computationally expensive when dealing with large data sets.

The Sinkhorn loss, an OT-based loss function, was proposed in [107] as a means of addressing these issues.

This approach utilizes two fundamental techniques: (a) entropic smoothing, which transforms the original OT loss into a more robust and differentiable quantity that can be evaluated using Sinkhorn fixed point iterations; and (b) algorithmic differentiation of these iterations, enabling seamless execution on GPUs. For an introduction to the Sinkhorn distance and algorithm, we referred to the following articles [104] its corresponding GitHub repository and [108] .

7.2.2. Sinkhorn Distance

To gain a comprehensive understanding of the Sinkhorn distance, let us examine a classic illustrative scenario that involves a transportation problem between bakeries and bars in Manhattan. Every morning, the bakeries supply croissants to the bars throughout the area. The problem is to determine the best way to transport the croissants, based on the cost associated with each Bakery and Bar. The cost of transporting croissants from bakery A to bar B is given by the function $c(A,B)$.

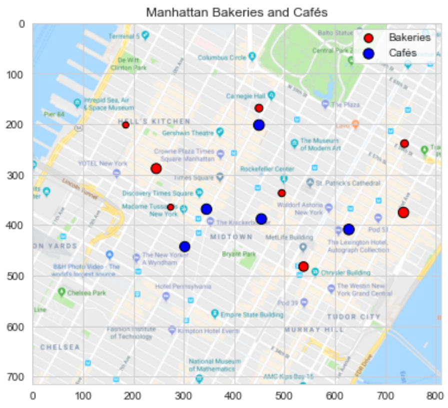


Figure 7.9.: Map of Bars and Bakery generated using [109]

The amount called transportation plan is $\pi(i,j)$, where i is one of the Bakery, and j is one of the Bar. Given any such kind of amount $\pi(i,j)$, we will have a deterministic total cost, and the cost can be written as $\sum \pi(i,j)c(i,j)$. We define α_i the amount of croissants produced by bakery i, and β_j is the amount needed by bar j. To interpret both α_i and β_j as probability distributions without any loss of generality, we impose $\sum_i \alpha_i = 1 = \sum_j \beta_j = 1$.

Instead of considering the classical optimal transport we now consider the following modified optimal transportation problem where an extra constraint is imposed to make sure that the KL divergence distance between π and $\alpha\beta^T$ is smaller to some predefined parameter.

$$\Gamma^* = \min \sum_{i,j} \pi_{i,j} c_{i,j} \tag{7.17}$$

with the following constraints:

$$\begin{cases} KL(\pi|\alpha\beta^T) \le a \\ \sum_j \pi_{i,j} = \alpha_i \\ \sum_i \pi_{i,j} = \beta_j \end{cases} \tag{7.18}$$

The Sinkhorn distances is the optimal value of the following revised optimal transportation problem.

The decision to introduce an additional parameter is motivated by the fact that the Kullback-Leibler (KL) divergence is a measure of how different two probability distributions are from each other. Here it means the distance between the two distances should be close, or the optimal solution of π should be around $\alpha\beta^T$, which is a pre-defined distributions as a result of α and β being both known distributions. The underlying motivation for this constraint lies in the impact of the distribution $\alpha\beta^T$ on the transportation plan. Essentially, this plan ensures that if the Bakery produces more, it should transport a greater amount, and if the Bar requires more, it should receive proportionally more from each producer.

Expanding $KL(\pi|\alpha\beta^T)$ we have:

$$KL(\pi|\alpha\beta^T) = \sum_{i,j} \pi_{i,j} \log(\frac{\pi_{i,j}}{\alpha_i\beta_j}) = \sum_{i,j} \pi_{i,j} \log(\pi_{i,j}) - \sum_{i,j} \pi_{i,j} \log(\alpha_i) - \sum_{i,j} \pi_{i,j} \log(\beta_j) \tag{7.19}$$

Using that $\sum_i \pi_{i,j} = \alpha_i$ and $\sum_j \pi_{i,j} = \beta_i$ we have:

$$KL(\pi|\alpha\beta^T) = \sum_{i,j} \pi_{i,j} \log(\frac{\pi_{i,j}}{\alpha_i\beta_j}) = \sum_{i,j} \pi_{i,j} \log(\pi_{i,j}) - \sum_{i} \alpha_i \log(\alpha_i) - \sum_{j} \beta_j \log(\beta_j) \tag{7.20}$$

Using the definition of entropy, i.e $\sum_{i,j} \pi_{i,j} \log(\pi_{i,j}) = \mathfrak{h}(\pi)$ and $\sum_i \alpha_i \log(\alpha_i) = \mathfrak{h}(\alpha)$ we can write the constraint as simple as follows

$$\mathfrak{h}(\alpha)+\mathfrak{h}(\beta)-\mathfrak{h}(\pi)\leq a \tag{7.21}$$

Therefore looking at the dual problem we have the following:

$$\hat{\Gamma}=\sum_{i,j}\pi_{i,j}c_{i,j}-\frac{1}{\lambda}\mathfrak{h}(\pi) \tag{7.22}$$

with constraints:

$$\begin{cases}\sum_j \pi_{i,j}=\alpha_i\\ \sum_i \pi_{i,j}=\beta_j\end{cases}$$

Writing the Langrangian and looking for the optimality, i.e $\frac{\partial\mathscr{L}}{\partial\pi_{i,j}}=0$ we have the following:

$$\pi^*_{i,j}=u_i e^{-\lambda c_{i,j}}v_j \tag{7.23}$$

Equivalently

$$\pi^*=\text{diag}(u)e^{\lambda c}\text{diag}(v) \tag{7.24}$$

where u and v are two positive vectors.

7.2.2.1. Sinkhorn Algorithm

The Sinkhorn algorithm performs alternating projections between two sets: one that contains all matrices with row sums equal to α, and another that contains all matrices with column sums equal to β. Two vectors of scalar values, u and v, are iteratively updated starting from some initial values to achieve these projections. As the algorithm progresses, it converges to the intersection of the two sets, which corresponds to the set of matrices with row sums equal to α and column sums equal to β, and represents the optimal solution [110].

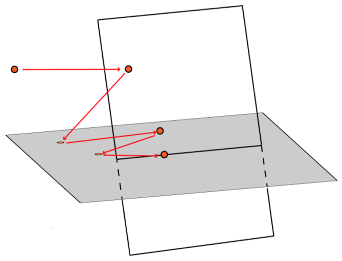


Figure 7.10.: Example of Sinkorn algortihm behaviour adopted from [110]

Finally using this approach for the example of the Bakery and Bar we have:

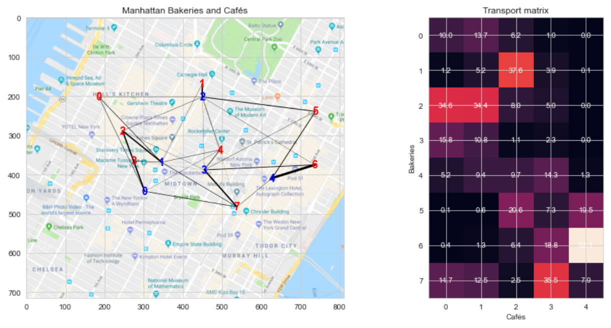


Figure 7.11.: Visual rapresentation of Optimal transport results and Transport matrix

7.2.3. COT-GAN

Having introduced OT theory, Sinkhorn distance and Sinkhorn algorithm we can now present the approach, COT GAN, taken in the [10].

The goal of COT-GAN is to generate new d-dimensional (number of channels), T -long sequences. The idea is to make use of causal optimal transport between $X = \mathbb{R}^{d\times T}$ and $Y = \mathbb{R}^{d\times T}$ in order to encode temporal dependencies in of the sequences.

To utilize the Sinkhorn algorithm, they consider the entropic regularization problem, which formulates the following problem:

$$\Gamma^{\mathcal{K}}_{c,\epsilon}(\mu,\nu) := \inf_{\pi\in\Pi^{\mathcal{K}}(\mu,\nu)} \{\mathbb{E}^{\pi}[c(x,y)] - \epsilon H(\pi)\} \tag{7.25}$$

Here μ, ν are distributions on the path space $\mathbb{R}^{d\times T}$ and $H(\pi) = \sum_{i,j} \pi(x^i, y^j)\log(\pi(x^i, y^j))$ is the Shannon entropy of π. On $X \times Y$, we denote by $x = (x_1, ..., x_T)$ and $y = (y_1, ..., y_T)$ the first and second half of the coordinates, and we let $\mathcal{F}_X = (\mathcal{F}^X_t)^T_{t=1}$ and $\mathcal{F}_X = (\mathcal{F}^Y_t)^T_{t=1}$ be the canonical filtrations. $\Pi^{\mathcal{K}}(\mu,\nu)$ is the set of causal transport plan defined in (14)

The optimizer for the mentioned problem (7.25) is denoted as $\pi^K_{c,\epsilon}(\mu,\nu)$, and the regularized COT distance is defined as follows:

$$W_{c,\epsilon}(\mu,\nu) := \mathbb{E}^{\pi^K_{c,\epsilon}(\mu,\nu)}[c(x,y)] \tag{7.26}$$

Instead of examining equation (7.25) directly, an alternative approach is adopted, which involves reformulating the problem as a maximization over non-causal problems utilizing a specific family of cost functions. This significant finding has been demonstrated in the research conducted in [111]. For that the reformulation of causality constraint given in [112] is taken in consideration

Let $\mathcal{M}(\mathcal{F}^X,\mu)$ be the set of $(X,\mathcal{F}^X,\mu)$-martingales, and define

$$\mathcal{H}(\mu) := \{(h, M) : h = (h)_{t=1}^{T-1}, h \in C_b(\mathbb{R}^{d\times t}), M = (M)_{t=1}^{T-1} \in \mathcal{M}(\mathcal{F}^X, \mu), M \in C_b(\mathbb{R}^{d\times t})\}, \tag{7.27}$$

where, as usual, $C_b(X)$ denotes the space of continuous, bounded functions on X. Then, a transport plan $\pi \in \Pi(\mu, \nu)$ is causal if and only if

$$\mathbb{E}^{\pi}[\sum_{t=1}^{T-1} h_t(y_{\leq t})\Delta_{t+1}M(x_{\leq t+1})] = 0 \tag{7.28}$$

where $x_{\leq t} := (x_1, x_2, ..., x_t)$ and similarly for $y_{\leq t}$, and $\Delta_{t+1}\mathcal{M}(x \leq t+1) := \mathcal{M}_{t+1}(x+1) - \mathcal{M}_t(x \leq t)$.

From [111] we have that $\sup_{l\in\mathcal{L}(\mu)}[l(x,y)] = \begin{cases} 0 \text{ if } \pi \text{ is casual} \\ \infty \text{ otherwise} \end{cases}$

where $\mathcal{L}(\mu, \nu) := \left\{\sum_{j=1}^{J}\sum_{t=1}^{T-1} h_t^j(y)\Delta_{t+1}M^j(x),\ \ J \in N, (h^j, M^j) \in H(\mu)\right\}$

This allows to rewrite

$$\Gamma_{c,\epsilon}^{\mathcal{K}}(\mu,\nu) = \inf_{\pi\in\Pi(\mu,\nu)} \sup_{l\in\mathcal{L}(\mu)} \left\{\mathbb{E}^{\pi}[c(x,y) + l(x,y)] - \epsilon H(\pi)\right\}$$

By leveraging the min-max theorem, it can used the convexity of $\mathcal{L}(\mu)$ and the convexity and compactness of $\Pi(\mu,\nu)$ to obtain the following result:

$$\Gamma_{c,\epsilon}^{\mathcal{K}}(\mu,\nu) = \sup_{l\in\mathcal{L}(\mu)} \inf_{\pi\in\Pi(\mu,\nu)} \left\{\mathbb{E}^{\pi}[c(x,y) + l(x,y)] - \epsilon H(\pi)\right\} = \sup_{l\in\mathcal{L}(\mu)} \Gamma_{c+l,\epsilon}(\mu,\nu) \tag{7.29}$$

Hence, in accordance with the notation in equation (7.26), the problem addresses the worst-case distance between μ and ν, defined as:

$$\sup_{l\in\mathcal{L}(\mu)} \mathcal{W}_{c+l,\epsilon}(\mu,\nu) \tag{7.30}$$

This serves as a regularized Sinkhorn distance, ensuring adherence to the causal constraint on the transport plans.

The paper addresses two main problems, with the first one focusing on the application of Sinkhorn divergence in the context of mini-batches. In this scenario, the training dataset comprises a collection of paths $\{x_i\}_{i=1}^N$, all having the same length T, where each x_i represents a sequence $(x_i^1, x_i^2, \ldots, x_i^T)$ in $\mathbb{R}^d$. Given the potentially large size of N, the approach involves approximating $\mathcal{W}_{c+l,\xi}(\mu,\nu)$ using its empirical mini-batch counterpart. Specifically, a batch size of m is chosen, and $\{x_i\}_{i=1}^m$ is sampled from the dataset, along with $\{z_i\}_{i=1}^m$ from the latent space ζ. The generated samples are denoted as $y_i^\theta = g^\theta(z_i)$, while the empirical distributions are represented as $\hat{x} = \sum_{i=1}^m \delta_{x^i}$ and $\hat{y}_\theta = \sum_{i=1}^m \delta_{y_\theta}$.

During the optimization process of the original Sinkhorn divergence over mini-batches, they have identified a potential issue that can result in biased parameter estimation. To overcome this problem, the authors propose utilizing a mixed Sinkhorn divergence at the mini-batch level, which helps improve the accuracy of parameter estimation. The mixed Sinkhorn divergence is defined for a generic cost function c as follows:

$$\mathcal{W}^{mix}_{c,\epsilon}(\hat{x},\hat{x}',\hat{y}_\theta,\hat{y}'_\theta) := \mathcal{W}_{c,\epsilon}(\hat{x},\hat{y}_\theta) + \mathcal{W}_{c,\epsilon}(\hat{x}',\hat{y}'_\theta) - \mathcal{W}_{c,\epsilon}(\hat{x},\hat{x}') - \mathcal{W}_{c,\epsilon}(\hat{y}_\theta,\hat{y}'_\theta) \tag{7.31}$$

The specific cost function employed during training is defined as follows:

$$c^K_\varphi(x,y) := c(x,y) + \sum_{j=1}^{J}\sum_{t=1}^{T-1} h^j_{\varphi_1,t}(y)\Delta_{t+1}M^j_{\varphi_2}(x) \tag{7.32}$$

where $\mathbf{h}_{\varphi_2} := (h^j_{\varphi_1})^J_{j=1}$ and $\mathbf{M}_{\varphi_2} := (M^j_{\varphi_2})^J_{j=1}$ are two separate neural networks.

The cost function mentioned in Equation (7.32) is employed to approximate the four terms described in Equation (7.31) using the Sinkhorn algorithm. By executing the Sinkhorn algorithm for a specified number of iterations L, these approximations are obtained.
Using this method, the model incorporates only the information available up to the current time step. As a result, it successfully captures the process's temporal dependencies.

The second challenge pertains to the integration of the martingale property within the algorithm. This challenge arises from the inherent difficulty of imposing constraints related to conditional expectations, as enforcing the martingale condition directly proves to be non-trivial. They propose to penalize the processes M if their increments on every time step have a non-zero average. For an $(X,\mathcal{F}^X)$-adapted process we have

$$p_{\mathcal{M}_{\varphi_2}}(\hat{x}) = \frac{1}{mT}\sum_{j=1}^{J}\sum_{t=1}^{T-1}|\sum_{i}^{m}\frac{\Delta_{t+1}M^j_{\varphi_2}(x^i)}{\sqrt{Var[M_{\varphi_2}]}+\eta}| \tag{7.33}$$

Accordingly, the adversarial objective function related to COT-GAN is defined as.

$$\mathcal{W}^{mix}_{c^K_\varphi,\epsilon}(\hat{x},\hat{x}',\hat{y}_\theta,\hat{x}') - \lambda p_{\mathcal{M}_{\varphi_2}}(\hat{x}') \tag{7.34}$$

Taking λ a positive constant we update θ to decrease this objective, and φ to increase it.

Incorporating these arrangements, they proposed a GAN consisting of a generator, and two discriminators, represented by h_{φ_1} and M_{φ_2}. Here, the generator g_θ is a parameterized function that takes a distribution ζ from a latent space and is trained to produce an induced distribution $\nu_\theta = \zeta \cdot g^{-1}$ that closely approximates the target distribution μ. The discriminators, parameterized by φ_1 and φ_2 respectively, learn to distinguish between the real data distribution μ and the generated distribution ν_θ using a robust (worst-case) distance.The algorithm is summarized in Algorithm [10].

7.2.4. Experiments

7.2.4.1. Sinusoidal Series

We utilized *s* to conduct tests on sinusoids with random phase, frequency,amplitude and dimensions, specifically 1, 5, and 10. The computational performance of the algorithm was evaluated on a MacBook Air, and each iteration for the 10-dimensional case took approximately 1-2 seconds to complete. As for the Sinkhorn Loss, the results align, as expected, with the results presented in the article, as shown in Figure (7.12). These findings provide support for the proposed approach and demonstrate its consistency with the reported results in the research paper

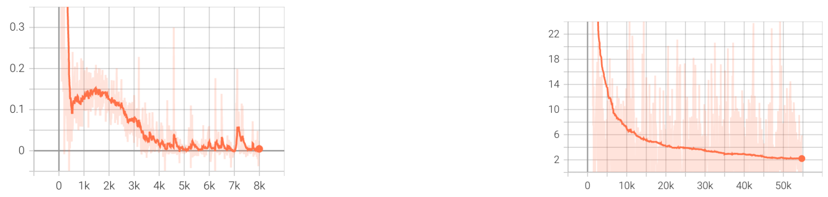


Figure 7.12.: Example of Sinkhorn Loss function for a 1-dim on the left and the analyzed (7.13) 10-dim sine series on the right

To enhance comprehension, Figure (7.13) illustrates the step-by-step progression of the process. The figure showcases different stages of the process, enabling a clearer visualization of the overall procedure.

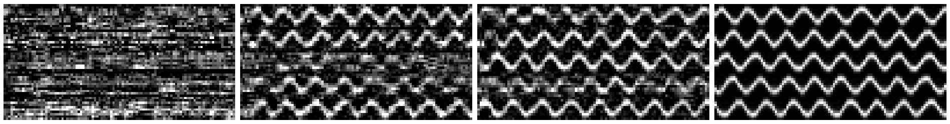


Figure 7.13.: Training of the first 5 dimensions of a 10 dimensional Sine series

7.2.4.2. Financial Time Series

We conducted some tests on different financial time series, we have arrived at findings that are in line with the results discussed in the article. The article identifies the high non-stationarity of financial data is a significant element limiting the algorithm's ability to train effectively. In light of this results, they suggested concentrating future efforts on creating an improved COT-GAN algorithm that is especially suited to handle these difficulties. It is crucial to remember that the use of causality as a technique for calculating the distances between sequences will still be essential in this improved approach.

7.3. GENERATING FINANCIAL TIME SERIES USING DIFFUSION MODELS

Time series data can be represented as images using an interesting method called the Gramian Angular Field (GAF) [113]. The idea it to apply this to transform financial time data into images.

Once the financial time series is encoded as an image using the GAF, the next step is to generate new images that are similar to the original one. For this purpose we choose to use Diffusion models.

The crucial advantage of using the GAF encoding map is its bijectivity, which implies that the generated images can be translated back into new financial time series. Therefore the idea is, by applying the inverse transformation, to convert the generated images into time series data. The expectation is that these newly generated time series will possess similarities to the initial ones.

7.3.1. GAF

Proposed by Wang and Oates in [113], the initial stage of constructing the Gramian Angular Field (GAF) involves rescaling the n real-valued observations $\mathbb{X} = \{x_1, x_2, ..., x_n\}$ of a time series. The objective is to ensure that all values are within the interval $[-1, 1]$ or $[0, 1]$ through the following process:

$$\tilde{x}_i = \frac{(x_i - \max(\mathbb{X})) + (x_i - \min(\mathbb{X}))}{\max(\mathbb{X}) - \min(\mathbb{X})} \tag{7.35}$$

The scaled series $\tilde{\mathbb{X}}$ is is transformed to a polar coordinates system by computing the angular cosine of the single components of the scaled time series:

$$\theta_i = \arccos(\tilde{x}_i) \qquad r = \frac{t_i}{n} \quad \text{where} \quad \tilde{x}_i \in \tilde{\mathbb{X}}, t_i \in N \tag{7.36}$$

Here, t_i is the time stamp and N is a constant factor to regularize the span of the polar coordinate system.

Gramian Summation Angular Field (GASF) and Gramian Difference Angular Field (GADF) can be defined as the sum or difference of the time series' points:

$$\text{GASF} = [\cos(\theta_i + \theta_j)] = \tilde{\mathbb{X}}' \cdot \tilde{\mathbb{X}} - \sqrt{I - \tilde{\mathbb{X}}'^2} \cdot \sqrt{I - \tilde{\mathbb{X}}^2} \tag{7.37}$$

$$\text{GADF} = [\sin(\theta_i - \theta_j)] = \sqrt{I - \tilde{\mathbb{X}}'^2} \cdot \tilde{\mathbb{X}} - \tilde{\mathbb{X}}' \cdot \sqrt{I - \tilde{\mathbb{X}}^2} \tag{7.38}$$

A key aspect of the Gramian Angular Fields (GAFs) is that yields one and only one outcome in the polar coordinate system with a unique inverse map. This characteristic enable us to translate the images produced back into time series data.

Moreover, GAFs possess the ability to preserve temporal dependencies. In this method time is represented by the position within the image. As we move from the top-left to the bottom-right of the image, time progressively increases. The inherent temporal relationships and ordering of the original time series are preserved in this spatial representation.

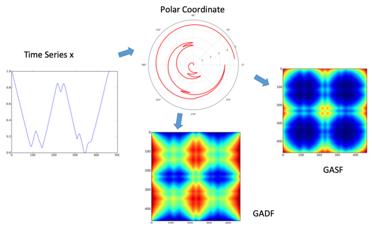


Figure 7.14.: Example of Gramian Angular Fields behavior adopted from [113]

7.3.2. Diffusion model

As defined in [114], a diffusion probabilistic model, commonly referred to as a "diffusion model," is a parameterized Markov chain trained using variational inference. The objective of a diffusion model is to generate samples that match the data distribution after a finite number of transformations. To provide a concise explanation, we will follow the description in [114], [115], [116].

Diffusion models consist of two distinct steps: the Forward and the Reverse diffusion process. In the first, Gaussian noise is gradually added to the observed data, starting from an initial distribution and progressing through a series of transformations. In the second, the original data is reconstructed based on the final latent variables obtained from the Forward process.

Mathematically, for the Forward process, given a data point sampled from a real data distribution $x_0 \sim q(x)$, is built a Markov chain that progressively introduces Gaussian noise to the data according to a variance schedule $\beta_1, ..., \beta_T$:

$$q(x_{1:T}) := \prod_{t=1}^{T} q(x_t|x_{t-1}), \quad q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{7.39}$$

However, when it comes to reconstruction, estimating $q(x_t|x_{t-1})$ directly is challenging since it requires using the entire dataset. As a result, a model is learned to approximate these conditional probabilities, allowing to perform the reverse diffusion process described below:

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \qquad p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{7.40}$$

Here, the model $p_\theta(x_{0:T})$ is a Markov chain with learned Gaussian transitions starting at $p(x_T)$, which is assumed to be a Gaussian with mean zero and unit covariance.

7.3.3. EXPERIMENTS

To preprocess the financial time series data, we adopt a partitioning approach where the original data is divided into monthly path segments. Then, as shown in the Figure (7.15), we apply the Gramian Angular Field (GAF) transformation.

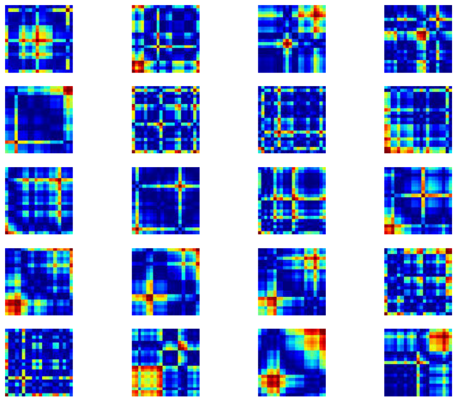


Figure 7.15.: Sample of financial time series as images

Next, we proceed to train the diffusion model described in [117] using the generated images. The Forward process, as shown in Figure (7.16), effectively deconstructs the images

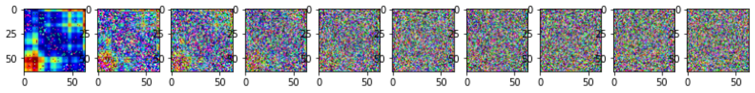


Figure 7.16.: Forward process

However, when we attempt to apply the reverse process, we encounter certain challenges. As evident from Figure (7.17),

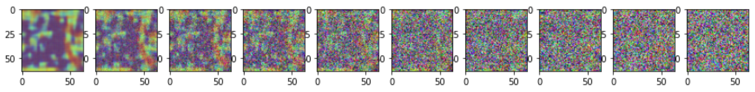


Figure 7.17.: Reverse process

To assess the algorithm's performance, we ran a number of tests across 100 epochs. However, the observed lack of similarity between the produced images can be attributed to several factors.

Firstly, the number of epochs utilized in our experiments might be insufficient. As highlighted by the author, achieving satisfactory results with this algorithm often requires a significantly larger number of epochs. Unfortunately, due to the computational demands, training for more than 100 epochs already exceeds a duration of 12 hours

Secondly, it is worth noting that the algorithm was trained on a substantially larger data set consisting of over 8,000 images, while our own data set comprises less than 300 images. This disparity in the training data size could have an impact on the algorithm's ability to generate similar images.

Lastly, the inherent structure of the images themselves might contribute to the observed results. Although we experimented with higher resolution images by incorporating more pixels, the similarity of the outcomes remained largely consistent.

8 Conclusion and Future Works

In conclusion, this thesis addressed various problems, with an initial emphasis on Neural Stochastic Differential Equations (NSDEs). The theory behind NSDEs is relatively new, but there is a growing belief in its tremendous potential for further expansion, supported by an increasing number of researchers working in this field and the publication of numerous papers.

The exploration of existing literature in this thesis gave the opportunity to gain a better understanding of NSDEs. During the model selection process, an ongoing discussion arose regarding the efficiency of rough volatility models in reproducing real-world situations compared to classical volatility models. After replicating and comparing the results from both factions, we decided to embrace the second faction. This decision was also influenced by the time constraint prevalent in real-world finance, as the algorithm based on Markovian models already consumed a significant amount of time. Consequently, using rough volatility models would have further exacerbated this problem.

The financial mathematics issues used to evaluate NSDEs were calibration and neural joint calibration. While we were convinced to employ Markovian models for calibration, we undertook extensive research before making a conclusion on joint calibration. A brief literature review confirmed the validity of our choice, and we found the results presented in the aforementioned paper [44] to be particularly interesting. The results concerning calibration are excellent, even when working with real data, which presents a different scenario compared to calibration with synthetic data. The relative errors allow for efficient calibration of Neural SDEs, even with a very small data set. To test efficiency, we performed various tests on different data sets using two types of stochastic models.

Regarding joint calibration, we proposed two possible approaches related to the different ways of simulating the VIX. We only tested the case where the VIX is obtained as an approximation using a simplification of the formula proposed in [64]. The results reveal that this approach achieves good calibration for the SPX and gives decent for the VIX. This partial problem may be due to multiple reasons. We believe it is because we used an approximation to calculate the VIX and that more data and more simulation numbers are needed. In fact, we tested the algorithm with a smaller amount of data and simulations and observed a significant impact on the algorithm's ability to capture the correct dependency. Therefore, we believe that using more data and employing Nested Monte Carlo simulations could yield even better results. Supporting our conviction is the work done by Guyon, who achieved optimal results by using Least Monte Carlo Squares and a significantly larger amount of data.

The issue of data played a prominent role in our thesis and is becoming increasingly common in the need to train algorithms with a larger amount of data. Therefore, we delved deeper into exploring methods for generating synthetic data. Among all the proposed methods, two in particular caught our attention: the method presented in [10] and the one presented in [9]. These methods stood out for their ability to incorporate theoretical mathematical concepts into data generation. Consequently, we extensively studied their potential applications.

Method in [9], in particular, yielded excellent results in generating financial time series. Its implementation showcased its effectiveness in capturing the dynamics of real-world financial data. On the other hand, method in [10] demonstrated great potential in generating time series but struggled when applied to non-stationary series.

Finally we proposed a possible new approach. However, we tested using a basic algorithm and better results can be reached in the future.

We were finally able to see throughout the thesis that very theoretical mathematical topics like SDEs, optimal transport, and rough path theory fit well with the new models related to deep learning and machine learning, and I believe this thesis allows us to further our understanding of how both fields can benefit from each other by further contaminating one another.

Bibliography

[1] K. Ito; *On stochastic processes.* 1941. DOI: 10.4099/jjm1924.18.0_261.

[2] P. Kidger, J. Foster, X. Li, H. Oberhauser, and T. Lyons. *Neural SDEs as Infinite-Dimensional GANs.* 2021. arXiv: 2102.03657 [cs.LG].

[3] P. Gierjatowicz, M. Sabate-Vidales, D. Šiška, L. Szpruch, and Ž. Žurič. *Robust pricing and hedging via neural SDEs.* 2020. arXiv: 2007.04154 [q-fin.MF].

[4] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. *Spectral Normalization for Generative Adversarial Networks.* 2018. arXiv: 1802.05957 [cs.LG].

[5] J. Gatheral, T. Jaisson, and M. Rosenbaum. *Volatility is rough.* 2014. arXiv: 1410.3394 [q-fin.ST].

[6] L. C. G. Rogers. *Things we think we know.* Jan. 2019.

[7] R. Cont and P. Das. *Rough volatility: fact or artefact?* 2022. arXiv: 2203.13820 [q-fin.ST].

[8] S.-P. Zhu and G.-H. Lian. *An analytical formula for VIX futures and its applications†.* 2011. DOI: 10.1002/fut.20512.

[9] H. Bühler, B. Horvath, T. Lyons, I. P. Arribas, and B. Wood. *A Data-driven Market Simulator for Small Data Environments.* 2020. arXiv: 2006.14498 [q-fin.ST].

[10] T. Xu, L. K. Wenliang, M. Munn, and B. Acciaio. *COT-GAN: Generating Sequential Data via Causal Optimal Transport.* 2020. arXiv: 2006.08571 [stat.ML].

[11] J. Jordon, L. Szpruch, F. Houssiau, M. Bottarelli, G. Cherubin, C. Maple, S. N. Cohen, and A. Weller. *Synthetic Data – what, why and how?* 2022. arXiv: 2205.03257 [cs.LG].

[12] I. Goodfellow. *NIPS 2016 Tutorial: Generative Adversarial Networks.* 2017. arXiv: 1701.00160 [cs.LG].

[13] S. Liu, O. Bousquet, and K. Chaudhuri. *Approximation and Convergence Properties of Generative Adversarial Learning.* 2017. arXiv: 1705.08991 [cs.LG].

[14] D. P. Kingma and M. Welling. *An Introduction to Variational Autoencoders.* 2019. DOI: 10.1561/2200000056. URL: https://doi.org/10.1561%5C%2F2200000056.

[15] J. Riebesell. URL: https://tikz.net/author/janosh/.

[16] M. Arjovsky, S. Chintala, and L. Bottou. *Wasserstein GAN.* 2017. arXiv: 1701.07875 [stat.ML].

[17] F. C. Klebaner. *Introduction to Stochastic Calculus with Applications.* 3rd. World Scientific, 2012, p. 452. DOI: 10.1142/p821. URL: https://doi.org/10.1142/p821.

[18] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[19] I. Karatzas and S. E. Shreve. *Brownian Motion and Stochastic Calculus*. New York: Springer, 1991.

[20] L. C. G. Rogers and D. Williams. *Diffusions, Markov Processes, and Martingales*. 2nd ed. Vol. 1. Cambridge Mathematical Library. Cambridge University Press, 2000. DOI: 10.1017/CBO9781107590120.

[21] URL: https://tex.stackexchange.com/questions/505741/architecture-neural-network-with-weights.

[22] I. Neutelings. URL: https://tikz.net/author/izaak/.

[23] K. Hornik, M. Stinchcombe, and H. White. *Multilayer Feedforward Networks Are Universal Approximators*. GBR, July 1989.

[24] H. Lin and S. Jegelka. “ResNet with One-Neuron Hidden Layers is a Universal Approximator”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS’18. Montréal, Canada: Curran Associates Inc., 2018, pp. 6172–6181.

[25] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar, and C.-J. Hsieh. *Neural SDE: Stabilizing Neural ODE Networks with Stochastic Noise*. 2019. arXiv: 1906.02355 [cs.LG].

[26] O. Johansson. *Stochastic modeling using machine learning and stochastic differential equations*. 2022.

[27] B. Tzen and M. Raginsky. *Neural Stochastic Differential Equations: Deep Latent Gaussian Models in the Diffusion Limit*. 2019. arXiv: 1905.09883 [cs.LG].

[28] B. Tzen and M. Raginsky. *Theoretical guarantees for sampling and inference in generative models with latent diffusions*. 2019. arXiv: 1903.01608 [math.PR].

[29] L. Hodgkinson, C. van der Heide, F. Roosta, and M. W. Mahoney. *Stochastic Normalizing Flows*. 2020. arXiv: 2002.09547 [stat.ML].

[30] F.-X. Briol, A. Barp, A. B. Duncan, and M. Girolami. *Statistical Inference for Generative Models with Maximum Mean Discrepancy*. 2019. arXiv: 1906.05944 [stat.ME].

[31] C. Cuchiero, W. Khosrawi, and J. Teichmann. *A Generative Adversarial Network Approach to Calibration of Local Stochastic Volatility Models*. Sept. 2020. DOI: 10.3390/risks8040101. URL: https://doi.org/10.3390%2Frisks8040101.

[32] V. Oganesyan, A. Volokhova, and D. Vetrov. *Stochasticity in Neural ODEs: An Empirical Study*. 2020. arXiv: 2002.09779 [cs.LG].

[33] X. Li, T.-K. L. Wong, R. T. Q. Chen, and D. Duvenaud. *Scalable Gradients for Stochastic Differential Equations*. 2020. arXiv: 2001.01328 [cs.LG].

[34] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. *Neural Ordinary Differential Equations*. 2019. arXiv: 1806.07366 [cs.LG].

[35] P. Kidger. *On Neural Differential Equations*. 2022. arXiv: 2202.02435 [cs.LG].

[36] Y. Rubanova, R. T. Q. Chen, and D. K. Duvenaud. “Latent Ordinary Differential Equations for Irregularly-Sampled Time Series”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/42a6845a557bef704ad8ac9cb4461d43-Paper.pdf.

[37] R. Deng, B. Chang, M. A. Brubaker, G. Mori, and A. Lehrmann. *Modeling Continuous Stochastic Processes with Dynamic Normalizing Flows*. 2021. arXiv: 2002.10516 [cs.LG].

[38] P. Kidger. *TorchSDE: Stochastic Differential Equations in PyTorch*. https://github.com/google-research/torchsde. Accessed: June 8, 2023.

[39] F. Comte and E. Renault. *Long memory continuous time models*. 1996. DOI: https://doi.org/10.1016/0304-4076(95)01735-6. URL: https://www.sciencedirect.com/science/article/pii/0304407695017356.

[40] P.-A. CORPECHOT. *Study of the joint S&P 500/VIX smile calibration problem within rough volatility models*. 2019-2020. DOI: DOINumber. URL: URL.

[41] M. Fukasawa, T. Takabatake, and R. Westphal. *Is Volatility Rough ?* 2019. arXiv: 1905.04852 [math.ST].

[42] S. D. M. Julienne Delemotte and F. Ségonne. *Yet another analysis of the SP500 at-the-money skew: crossover of different power-law behaviours*. 2023.

[43] J. Guyon and M. El Amrani. *Does the Term-Structure of Equity At-the-Money Skew Really Follow a Power Law?* Available at SSRN: https://ssrn.com/abstract=4174538 or http://dx.doi.org/10.2139/ssrn.4174538. July 2022.

[44] S. E. Rømer. *Empirical analysis of rough and classical stochastic volatility models to the SPX and VIX markets*. 2022. DOI: 10.1080/14697688.2022.2081592. eprint: https://doi.org/10.1080/14697688.2022.2081592. URL: https://doi.org/10.1080/14697688.2022.2081592.

[45] I. Guo, G. Loeper, J. Obloj, and S. Wang. *Joint Modelling and Calibration of SPX and VIX by Optimal Transport*. 2021. arXiv: 2004.02198 [q-fin.MF].

[46] Cboe Global Markets. *Cboe Volatility Index Methodology*. https://cdn.cboe.com/api/global/us_indices/governance/Volatility_Index_Methodology_Cboe_Volatility_Index.pdf. Accessed: [Insert Date].

[47] S.-L. Chung, W.-C. Tsai, Y.-H. Wang, and P.-S. Weng. *The information content of the S&P 500 index and VIX options on the dynamics of the S&P 500 index*. 2011. DOI: https://doi.org/10.1002/fut.20532. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/fut.20532. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/fut.20532.

[48] J. Guyon. *Inversion of convex ordering in the VIX market. Quantitative*. 20(10). DOI: https://doi.org/10.1080/14697688.2020.1753885.

[49] A. Jacquier, C. Martini, and A. Muguruza. *On VIX Futures in the rough Bergomi model*. 2017. arXiv: 1701.04260 [q-fin.PR].

[50] J. Guyon. *The Joint S&P 500/VIX Smile Calibration Puzzle Solved*. Apr. 2020. DOI: 10.2139/ssrn.3397382. URL: https://ssrn.com/abstract=3397382.

[51] C. Cuchiero, G. Gazzani, J. Möller, and S. Svaluto-Ferro. *Joint calibration to SPX and VIX options with signature-based models*. 2023. arXiv: 2301.13235 [q-fin.MF].

[52] J. Gatheral. "Consistent modeling of SPX and VIX options". In: *Bachelier Congress*. 2008.

[53] J.-P. Fouque and Y. F. Saporito. *Heston Stochastic Vol-of-Vol Model for Joint Calibration of VIX and S&P 500 Options*. 2017. arXiv: 1706.00873 [q-fin.PR].

[54] E. A. Jaber, C. Illand, Shaun, and Li. *Joint SPX-VIX calibration with Gaussian polynomial volatility models: deep pricing with quantization hints*. 2022. arXiv: 2212.08297 [q-fin.MF].

[55] E. A. Jaber, C. Illand, Shaun, and Li. *The quintic Ornstein-Uhlenbeck volatility model that jointly calibrates SPX VIX smiles*. 2022. arXiv: 2212.10917 [q-fin.MF].

[56] J. Guyon and S. Mustapha. *Neural Joint S&P 500/VIX smile calibration using*. 2020. DOI: 10.2139/ssrn.4309576. URL: https://www.ssrn.com/abstract=4309576.

[57] J. Guyon and J. Lekeufack. *Volatility is (mostly) path-dependent*. July 2022.

[58] C. Kahl and P. Jäckel. *Fast strong approximation Monte Carlo schemes for stochastic volatility models*. 2006. DOI: 10.1080/14697680600841108. eprint: https://doi.org/10.1080/14697680600841108. URL: https://doi.org/10.1080/14697680600841108.

[59] M. Rosenbaum and J. Zhang. *Deep calibration of the quadratic rough Heston model*. 2022. arXiv: 2107.01611 [q-fin.CP].

[60] M. Hutzenthaler, A. Jentzen, and P. E. Kloeden. *Strong convergence of an explicit numerical method for SDEs with nonglobally Lipschitz continuous coefficients*. Aug. 2012. DOI: 10.1214/11-aap803. URL: https://doi.org/10.1214%5C%2F11-aap803.

[61] Ł. Szpruch and X. Zhāng. *V-integrability, asymptotic stability and comparison property of explicit numerical schemes for non-linear SDEs*. 2018.

[62] Patgie. *Robust NSDE - IV Spread*. https://github.com/patgie/robust_nsde/blob/master/nsde_2021_calibrations/IV_Spread/SPX_call_price.npy. 2021.

[63] H. Guerreiro and J. Guerra. *Least squares Monte Carlo methods in stochastic Volterra rough volatility models*. 2021. arXiv: 2105.04511 [q-fin.PR].

[64] Z. Cui, C. Lee, and M. Liu. *Valuation of VIX Derivatives through Combined Ito-Taylor Expansion and Markov Chain Approximation*. 2021.

[65] B. Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Springer Science & Business Media, 2013, p. 228.

[66] S. Watanabe. *Analysis of Wiener functionals (Malliavin calculus) and its applications to heat kernels*. 1987.

[67] C. Li and et al. *Maximum-likelihood estimation for diffusion processes via closed-form density expansions*. 2013.

[68] D. Nualart. *The Malliavin Calculus and Related Topics*. Vol. 1995. Springer, 2006.

[69] Z. Zuric. *https://github.com/ZuricZ/NeuralSDE*.

[70] *NumPy*. https://numpy.org/.

[71] *PyTorch*. https://pytorch.org/.

[72] *SciPy*. https://www.scipy.org/.

[73] D. Duffie, J. Pan, and K. Singleton. *Transform analysis and asset pricing for affine jump-diffusions*. 2000.

[74] J. Pan. *The jump-risk premia implicit in options: evidence from an integrated time-series study*. 2002.

[75] B. Eraker. *Do stock prices and volatility jump? Reconciling evidence from spot and option prices*. 2004.

[76] M. Broadie, M. Chernov, and M. Johannes. *Model specification and risk premia: Evidence from futures options*. 2007.

[77] Y.-J. Lin. *Pricing VIX futures: Evidence from integrated physical and risk-neutral probability measures*. 2007.

[78] J.-C. Duan and C.-Y. Yeh. *Jump and volatility risk premiums implied by VIX*. 2007.

[79] J. Jordon, A. Wilson, and M. van der Schaar. *Synthetic Data: Opening the data floodgates to enable faster, more directed development of machine learning methods*. 2020. arXiv: 2012.04580 [cs.LG].

[80] O. Mendelevitch and M. D. Lesh. *Fidelity and Privacy of Synthetic Medical Data*. 2021. arXiv: 2101.08658 [cs.LG].

[81] S. M. Bellovin, P. K. Dutta, and N. Reitinger. *Privacy and synthetic datasets*. 2019.

[82] S. Assefa. *Generating synthetic data in finance: opportunities, challenges and pitfalls*. June 2020.

[83] T. L. Terry J. Lyons Michael Caruana. *Differential Equations Driven by Rough Paths*. Springer Link, 2007.

[84] D. Levin, T. Lyons, and H. Ni. *Learning from the past, predicting the statistics for the future, learning an evolving system*. 2016. arXiv: 1309.0260 [q-fin.ST].

[85] K.-T. Chen. *Iterated integrals and exponential homomorphisms*. 1954.

[86] S. Zhou. "Signatures, Rough Paths and Applications in Machine Learning". Bachelor's thesis. Utrecht University, June 13, 2019.

[87] P. Bonnier, P. Kidger, I. P. Arribas, C. Salvi, and T. Lyons. *Deep Signature Transforms*. 2019. arXiv: 1905.08494 [cs.LG].

[88] B. M. Hambly and T. Lyons. *Uniqueness for the signature of a path of bounded variation and the reduced path group*. 2010.

[89] T. J. Lyons, M. Caruana, and T. Lévy. *Differential Equations Driven by Rough Paths*. Springer, 2007.

[90] S. Liao, T. Lyons, W. Yang, and H. Ni. *Learning stochastic differential equations using RNN with log signature features*. 2019. arXiv: 1908.08286 [cs.LG].

[91] J. Morrill, A. Fermanian, P. Kidger, and T. Lyons. *A Generalised Signature Method for Multivariate Time Series Feature Extraction*. 2021. arXiv: 2006.00873 [cs.LG].

[92] T. J. Lyons. *Differential equations driven by rough signals*. 1998.

[93] P. K. Friz and M. Hairer. *A Course on Rough Paths: With an Introduction to Regularity Structures*. Universitext. Cham: Springer, 2014.

[94] T. Lyons. *Rough paths, Signatures and the modelling of functions on streams*. 2014. arXiv: 1405.4537 [math.PR].

[95] H. Ni, L. Szpruch, M. Sabate-Vidales, B. Xiao, M. Wiese, and S. Liao. *Sig-Wasserstein GANs for Time Series Generation*. 2021. arXiv: 2111.01207 [cs.LG].

[96] I. Chevyrev and T. Lyons. *Characteristic functions of measures on geometric rough paths*. Nov. 2016. DOI: 10.1214/15-aop1068. URL: https://doi.org/10.1214%2F15-aop1068.

[97] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. *A Kernel Two-Sample Test*. 2012. URL: http://jmlr.org/papers/v13/gretton12a.html.

[98] I. Chevyrev and H. Oberhauser. *Signature moments to characterize laws of stochastic processes*. 2022. arXiv: 1810.10971 [math.ST].

[99] M. Min, R. Hu, and T. Ichiba. *Directed Chain Generative Adversarial Networks*. 2023. arXiv: 2304.13131 [cs.LG].

[100] C. Villani. *Topics in Optimal Transportation*. Graduate Studies in Mathematics. American Mathematical Society, 2003.

[101] C. Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Science & Business Media, 2008. DOI: 10.1007/978-3-540-71050-9.

[102] F. Santambrogio. *Optimal Transport for Applied Mathematicians*. 2015. URL: http://cvgmt.sns.it/paper/2813/.

[103] K. DeMason. *Optimal Mass Transport and the Isoperimetric Inequality*. 2020.

[104] M. Cuturi. *Sinkhorn Distances: Lightspeed Computation of Optimal Transportation Distances*. 2013. arXiv: 1306.0895 [stat.ML].

[105] J. B. Orlin. *A faster strongly polynomial minimum cost flow algorithm*. 1993. DOI: 10.1287/opre.41.2.338.

[106] O. Pele and M. Werman. "Fast and robust Earth Mover's Distances". In: *2009 IEEE 12th International Conference on Computer Vision*. 2009, pp. 460–467. DOI: 10.1109/ICCV.2009.5459199.

[107] A. Genevay, G. Peyré, and M. Cuturi. *Learning Generative Models with Sinkhorn Divergences*. 2017. arXiv: 1706.00292 [stat.ML].

[108] J. Wang. *A Simple Introduction on Sinkhorn Distances*. 2021. URL: https://amsword.medium.com/a-simple-introduction-on-sinkhorn-distances-d01a4ef4f085.

[109] Remi Flamary, Nicolas Courty, Aurelie Boisbunon. *Introduction to Optimal Transport with Python*. https://pythonot.github.io/master/auto_examples/plot_Intro_OT.html. 2023.

[110] G. Schiebinger. *Sinkhorn's Algorithm for Entropic Optimal Transport*. URL: https://personal.math.ubc.ca/~geoff/courses/W2019T1/Lecture13.pdf.

[111] B. Acciaio, J. Backhoff-Veraguas, and J. Jia. *Cournot-Nash equilibrium and optimal transport in a dynamic setting*. 2020. arXiv: 2002.08786 [math.OC].

[112] J. Backhoff, M. Beiglböck, Y. Lin, and A. Zalashko. *Causal Transport in Discrete Time and Applications*. USA, Jan. 2017. DOI: 10.1137/16M1080197. URL: https://doi.org/10.1137/16M1080197.

[113] Z. Wang and T. Oates. *Imaging Time-Series to Improve Classification and Imputation*. 2015. arXiv: 1506.00327 [cs.LG].

[114] J. Ho, A. Jain, and P. Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[115] L. Weng. *Diffusion Models: A Gentle Introduction*. https://lilianweng.github.io/posts/2021-07-11-diffusion-models/. 2021.

[116] Y. Xie and Q. Li. *Measurement-conditioned Denoising Diffusion Probabilistic Model for Under-sampled Medical Image Reconstruction*. 2022. arXiv: 2203.03623 [eess.IV].

[117] https://colab.research.google.com/drive/1sjy9odlSSy0RBVgMTgP7s99NXsqglsUL?usp=sharing.

[118] I. Perez. *Market Simulator*. GitHub repository. URL: https://github.com/imanolperez/market_simulator.

[119] T. Xu. *CoT-GAN: Generation of Complementary Time Series*. GitHub repository. URL: https://github.com/tianlinxu312/cot-gan.

A
Architectures

A.1. Architecture Calibration

For conducting the experiments, we adopted the existing architecture proposed in [3] [62] and [69] to ensure a robust methodology and reliable results

A.1.1. Architecture Calibration and Joint Calibration in case of S.V.

Three distinct neural networks are employed to capture the behavior the two volatility γ^V, σ^S, and the volatility drift Λ^V, respectively. In both case, calibration and joint calibration these neural networks have a straightforward feed-forward architecture. In the first case they consist of four hidden layers, each containing 50 neurons, in the second they consist of three hidden layers, each containing 20 neurons. The parameterisation of the hedging strategy for the vanilla option prices is achieved by a feed-forward linear network with three hidden layers, each comprising 20 neurons. In all of the neural networks, the activation functions employed in each hidden layer are nonlinear and specifically the rectified linear unit (ReLU) functions. Moreover, for the neural network corresponding to σ^S, a non-linear activation function is applied after the output layer to ensure a positive output. Specifically, the rectifier softplus(x) $= \log(1+\exp(x))$ is utilized for this purpose.

	Input Size	Output Size	Output Layer
μ^V	2	1	Linear
Λ^V	2	1	Softplus/Sigmoid
γ^S	3	1	Softplus/Sigmoid

Figure A.1.: Summary of the neural network structures in Neural SDEs driven by a general stochastic volatility models

A.1.2. ARCHITECTURE IN CASE OF THE S-M-2F-QHYP MODEL

The selection of the neural network architecture was similar to the previous one. In this case three distinct neural networks must be employed to capture the behavior of the two function ζ, and the volatility drift μ^V, respectively. These neural networks have a straightforward feed-forward architecture, consisting of four hidden layers, each containing 50 neurons. Similarly, the parameterisation of the hedging strategy for the vanilla option prices is achieved by a feed-forward linear network with three hidden layers, each comprising 20 neurons. This ensures uniformity in the neural network architecture across the entire modeling process.

	Input Size	Output Size	Output Layer
ζ_1	1	1	Linear
ζ_2	1	1	Linear
σ^S	3	1	Softplus/Sigmoid

Figure A.2.: Summary of the neural network structures in Neural SDEs driven by S-M-2F-QHyp

A.2. ARCHITECTURE VAE

As explained in 2, VAE consists of two key components: an encoder and a decoder. We adopted the existing architecture proposed in [9], [118]. In this architecture the encoder comprises one hidden layer and two latent layers, each with 50 nodes. The activation function used is a leaky (parametric) ReLU with a parameter of $\lambda = 0.3$. The decoder, on the other hand, consists of one hidden layer with 50 units and the same leaky ReLU activation function. The output layer of the decoder employs a sigmoid activation function.

A.3. ARCHITETURE COT-GAN

We adopted the existing architecture proposed in [10],[119]. The latent state dimensionality is set to 10 at each time step, accompanied by a 10-dimensional time-invariant latent state. The generator of synthetic data employed consists of a single-layer LSTM network. The output of each time step is then fed through two layers of fully connected ReLU networks. Both the latent h and t M employ the same discriminator architecture. It consists of two layers of 1-D causal CNNs with a filter length of 5 and a stride of 1. Each layer comprises 32 synthetic data neurons per time step. The ReLU activation function is used for all layers, while the output layer utilizes the sigmoid activation function. For parameter updates, the Adam optimization algorithm is used with a learning rate of 0.001, optimizing θ and φ. The batch size is set to 32.Furthermore, the hyperparameters λ and ϵ are assigned values of 10.0 and 10, respectively.