ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Enhancing Equity Data Availability using Synthetic Time Series Generation

Master Thesis

Azamat Zhaksylykov

July 28, 2025

Advisors: Dr. Florian Ofenheimer-Krach[3] , Damian Tschirky[1], Dr. Elizabeth Ren Rui Xing[2], Prof. Dr. Josef Teichmann[3],

IHH, [1] Zürcher Kantonalbank
LBCH, [2] Zürcher Kantonalbank
Department of Mathematics, (3), ETH Zürich

Abstract

Recent advances in neural network techniques have opened new avenues in financial modeling, particularly in quantitative finance. A data-driven hedging models, such as Deep Hedging (Bühler et al., 2019) , have gained popularity for their ability to learn and capture complex market dynamics directly from data. However, these models require large datasets for training, which are often scarce or inconsistent in historical market data. This thesis addresses the data scarcity issue by investigating the feasibility of adapting the Neural Jump Ordinary Differential Equations (NJ-ODEs) framework (F. Krach et al., 2022), originally designed for prediction, into a novel generative model. The proposed generative model was evaluated using two standard stochastic processes: the Geometric Brownian motion and the Ornstein-Uhlenbeck process.

Acknowledgements

My deepest thanks to Florian for his mentorship throughout this thesis. Many of the core ideas would not have emerged without his input. I greatly appreciate his consistent availability and openness. He was always willing to make time for discussion and support whenever needed.

Many thanks to Damian for making this industry thesis possible within the derivative trading team at Zürcher Kantonalbank. This was my first time working in the bank. I am very thankful for his support, feedback, and guidance during the thesis.

I am grateful to Elizabeth for her support and discussions during our weekly meetings. Her support during the onboarding process was invaluable. I appreciated her help, even with small things like finding a place to sit.

I am also grateful to Prof. Teichmann for the opportunity to explore a novel research topic. Presenting preliminary results at the Friday seminars proved to be a valuable experience for me. The feedback I received deepened my understanding of the topic and motivated me.

Contents

Chapter 1

Introduction

The last decade has seen remarkable progress in machine learning and artificial intelligence. In quantitative finance, these advancements have opened new possibilities for financial modeling and managing derivatives portfolio. Neural networks, with their ability to model complex, nonlinear relationships, have become powerful tools for pricing and hedging financial instruments (Bühler et al., 2019). Their growing popularity is driven by their potential to learn market dynamics directly from historical data, offering an alternative to traditional models that rely on parametric assumptions.

Neural network-based financial models, rely heavily on large, high-quality datasets to ensure accuracy and reliability (Ruf and Wang, 2020). However, obtaining extensive historical data in financial markets is challenging due to issues like non-stationarity, regime shifts, sparse data, irregular sampling intervals, and structural breaks . Moreover, financial data are often proprietary, costly, and restricted by privacy constraints, worsening the data scarcity problem. These barriers limit the practical use and effectiveness of neural network models in finance (Bühler et al., 2020, Potluru et al., 2023). Therefore, addressing data limitations while preserving statistical accuracy is an important research direction.

Addressing this limitations, this master's thesis attempts to adapt the NJ-ODEs framework into a generative model. NJ-ODEs, as defined by Herrera et al. (2021), are models for prediction and filtering of continuous-time stochastic processes, that ensures theoretical optimality guarantees. This research specifically leverages the Path-Dependent Neural Jump Ordinary Differential Equations (PD-NJ-ODEs), an extension of NJ-ODEs capable of handling non-Markovian and discontinuous stochastic processes by utilizing the signature transform. From now on, when referring to NJ-ODEs in this thesis, we mean the extended PD-NJ-ODEs framework, as opposed to the original NJ-ODE model.

This thesis investigates the generative potential of the NJ-ODEs framework, through empirical evaluations. The model's performance was tested using standard benchmark stochastic processes widely used in financial modeling, namely Geometric Brownian Motion (GBM) and the Ornstein-Uhlenbeck (OU) process. These processes are foundational in quantitative finance due to their analytical tractability and relevance to financial applications (Black and Scholes, 1973, Uhlenbeck and Ornstein, 1930, Vasicek, 1977). The experiments evaluated the NJ-ODE framework's ability to capture the key dynamics and statistical properties of these processes. They aimed to confirm its potential for generating realistic synthetic financial data.

The rest of this thesis is structured as follows. Chapter 2 provides a theoretical background, covering synthetic time-series generation in finance, neural network architectures, mathematical foundations, and the NJ-ODE framework. Chapter 3 describes the methodologies developed to adapt the NJ-ODE framework into generative modeling. Chapter 4 and Chapter 5 detail the experimental setups, implementations, and results for two distinct generative approaches. Chapter 6 explores improvements to the NJ-ODE model by data sparsity, testing different loss functions and network architectures. Finally, Chapter 7 concludes by summarizing the key findings and challenges encountered throughout the research.

Chapter 2

Background*

This chapter opens with a concise introduction to synthetic time-series generation research in finance. It proceeds with a brief overview of feedforward, residual, and recurrent neural networks. It then introduces the key mathematical concepts relevant to this thesis, followed by a presentation of the NJ-ODE framework.

2.1 Synthetic Time Series Generation in Finance

The generation of synthetic financial time series has become increasingly important in modern quantitative finance. It facilitates various tasks such as model validation, stress testing, hedging derivatives portfolio, and the development of algorithmic trading strategies (Horvath et al., 2025). Traditionally, such data has been simulated using parametric models, such as the Black–Scholes model (Black and Scholes, 1973), the Heston model (Heston, 1993), ARIMA (Box et al., 2015), and GARCH(Bollerslev, 1986). These models depend on strict structural assumptions about the underlying stochastic processes and require calibrating a limited set of parameters to historical market data. Although these models provide analytical interpretability, they often struggle to reproduce the complex nonlinear dependencies and stylized features commonly observed in empirical financial data(Horvath et al., 2025).

Recent advances in machine learning have enabled the development of nonparametric, data-driven generative models that seek to learn the underlying distribution of financial time series directly from historical observations, without relying on rigid distributional assumptions(Bühler et al., 2020). They are called Market Generators. They span a variety of neural architectures, including Generative Adversarial Networks (GANs)(Goodfellow

[1]Some parts of this chapter are adapted from prior work, including Herrera et al. (2021), F. T. O. Krach (2025) and Andersson (2024).

et al., 2014), Variational Autoencoders (VAEs)(Kingma and Welling, 2019), and transformer-based models (Vaswani et al., 2017). The representational power of these models has demonstrated encouraging results in capturing the complex dynamics of financial markets. This holds true even in scenarios with limited data availability or non-stationarity (Horvath et al., 2025).

2.2 Neural Networks

Neural networks constitute a class of computational models inspired by the structure and functioning of biological neural systems. They have gained widespread usage across machine learning, artificial intelligence, and financial modeling due to their capacity to identify and model complex nonlinear relationships in datasets. Their strength lies in their ability to learn complex, nonlinear relationships from data through iterative training(Goodfellow et al., 2016).

2.2.1 Feedforward Neural Networks

Feedforward neural networks, also known as multi-layer perceptrons (MLPs), are foundational neural network architectures widely used in various applications, including financial modeling tasks. These networks consist of an input layer, multiple hidden layers, and an output layer, with data flowing unidirectionally from input to output without looping back (Goodfellow et al., 2016).

Mathematically, a feedforward neural network with n hidden layers can be expressed as:

$$h_{(i)} = \sigma\left(W_{(i)}h_{(i-1)} + b_{(i)}\right), \quad i = 1, \ldots, n, \tag{2.1}$$

where $h_{(0)} = x$ is the input vector, $W_{(i)}$ are weight matrices, $b_{(i)}$ are bias vectors, and σ is a nonlinear activation function, such as sigmoid, tanh, or ReLU.

For example, a two-layer feedforward network is defined as:

$$y = W_{(2)}\sigma\left(W_{(1)}x + b_{(1)}\right) + b_{(2)}. \tag{2.2}$$

The training of such networks involves adjusting parameters ($W_{(i)}$ and $b_{(i)}$) to minimize a loss function. This optimization is typically done using gradient-based methods like stochastic gradient descent (SGD) or adaptive techniques such as Adam (Kingma and Ba, 2014). The backpropagation algorithm (Rumelhart et al., 1986) efficiently computes the gradients needed for parameter updates. This facilitates the widespread use of feedforward neural networks in practical applications.

2.2.2 Residual Neural Networks

Despite their ability to model complex patterns, deep feedforward networks frequently encounter challenges such as vanishing and exploding gradients, particularly in very deep architectures (Goodfellow et al., 2016). Residual neural networks (ResNets), introduced by He et al. (2016), address these challenges through skip or residual connections. These connections allow gradients to bypass certain layers directly, making it easier to train very deep networks.

A residual network updates layer representations according to:

$$h_{(i+1)} = h_{(i)} + \sigma\left(W_{(i)}h_{(i)} + b_{(i)}\right). \tag{2.3}$$

This formulation allows gradients to propagate more effectively through deeper networks, enabling the training of models with many layers without performance degradation (Goodfellow et al., 2016).

2.2.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) are designed to handle sequential data, making them well-suited for time-series prediction tasks common in finance. RNNs maintain an internal state, or memory, that captures temporal dependencies in data(Goodfellow et al., 2016).

A standard recurrent network is defined as follows:

$$h_{i+1} = \sigma\left(Ux_{i+1} + Vh_i + b\right),\ y_{i+1} = g\left(h_{i+1}\right), \tag{2.4}$$

where x_i is the input at time t_i, h_i is the hidden state, U and V are input-to-hidden and hidden-to-hidden weight matrices, respectively, and g maps hidden states to outputs(Goodfellow et al., 2016).

For this thesis, we adopt the following notation. The RNN employs a neural network ρ_θ, where θ represents the trainable parameters, to update a discrete latent variable h using discrete observations x_i on a regular grid, according to the update rule $h_{i+1} := \rho_\theta(h_i, x_{i+1})$.

While traditional RNNs often experience difficulties with capturing long-term dependencies due to gradient instability. Advanced variants, such as Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Cho et al., 2014), effectively mitigate these challenges through specialized gating mechanisms. These enhancements improve model performance on tasks involving lengthy temporal sequences.

2.2.4 Neural Ordinary Differential Equations

A key limitation of traditional RNNs is their inability to handle irregularly sampled data or model continuous-time dynamics. To address these challenges, Chen et al. (2018) introduced the Neural Ordinary Differential Equations (Neural ODEs) framework. They define the evolution of hidden states through a continuous-time differential equation:

$$\frac{dh_t}{dt} = f(h_t, t, \theta), \tag{2.5}$$

where h_t denotes the hidden state at time t, and $f(\cdot, \cdot, \theta) = f_\theta$ is a neural network parameterized by trainable weights θ.

The hidden state at any time $t \geq t_0$ can then be obtained by solving the initial value problem:

$$h_t = h_{t_0} + \int_{t_0}^{t} f(h_s, s, \theta)\, ds, \tag{2.6}$$

which enables continuous-time modeling of latent dynamics.

This solution is computed using numerical ODE solvers such as Euler or Runge–Kutta methods ((Chen et al., 2018)). The update rule is written as:

$$h_t := \text{ODESolve}(f, h_{t_0}, (t_0, t)), \tag{2.7}$$

where `ODESolve` represents a chosen numerical integration method applied over the time interval $[t_0, t]$.

2.3 Mathematical setup

We consider a continuous-time stochastic process $X = (X_t)_{t \in [0,T]}$ taking values in $\mathbb{R}^d$. The process X is assumed to be càdlàg, allowing for both continuous evolution and abrupt jumps.

We typically do not observe the full trajectory of X. Instead, we observe its values at a finite, random number of irregularly spaced time points. Let $n \in \mathbb{N}$ denote the number of observation times, and let $\{t_i\}_{i=1}^{n} \subset [0, T]$ denote the corresponding sequence of observation times.

To account for missing data, we introduce an observation mask $M = (M_i)_{i \in \mathbb{N}}$, where each $M \in \{0, 1\}^d$ is a random variable indicating which components of X_{t_i} are observed at time t_i. Specifically, if $M_{i,j} = 1$, then the j-th component $X_{t_i,j}$ is observed at time t_i. Otherwise, the corresponding component is considered missing. Furthermore, let $\tau(t) := \max\{t_i : 0 \leq k \leq n, t_i \leq t\}$ represent the most recent observation time (or zero if no observations have been made) before time $t \in [0, T]$.

We define the available information up to time t as the σ-algebra generated by all observations made up to and including time t. A natural filtration $(\mathcal{A}_t)_{t\in[0,T]}$ representing all available information given by

$$\mathcal{A}_t := \sigma\left(X_{t_i,j}, t_i, M_i | i \leq \kappa(t), j \in \{1 \leq l \leq d | M_{i,l} = 1\}\right)$$

We denote the conditional expectation process of X by $\hat{X} := (\hat{X}_t)_{t\in[0,T]}$, where

$$\hat{X}_t := \mathbb{E}\left[X_t \,|\, \mathcal{A}_t\right].$$

As shown by Herrera et al. (2021), the NJ-ODE framework yields predictions that are L^2 optimal, as they approximate this conditional expectation.

All mathematical objects introduced in this section are formally defined in the appendix for clarity and reference.

2.4 Neural Jump Ordinary Differential Equations

In this section, I summarize the NJ-ODE framework developed by Florian Krach and collaborators. For a more detailed exposition is available at: https://floriankrach.github.io/njode/.

2.4.1 Introduction

The Neural Jump Ordinary Differential Equations (NJ-ODE) framework is a model for learning the latent dynamics of time-evolving systems from irregular and partially observed data. NJ-ODEs estimate the conditional expectation of future states by combining continuous latent evolution with discrete jumps at observation times (Herrera et al., 2021). The NJ-ODE architecture consists of three modular neural networks, each responsible for a distinct aspect of the system dynamics:

- **ODE Dynamics Network:** $f_{\theta_1} : \mathbb{R}^{d_H} \times \mathbb{R}^{d_X} \times [0,T] \times [0,T] \to \mathbb{R}^{d_H}$, governing the continuous evolution of the hidden state between observation times.
- **Jump Network:** $\rho_{\theta_2} : \mathbb{R}^{d_X} \to \mathbb{R}^{d_H}$, responsible for discrete updates of the hidden state at observation times, introducing a jump in the latent dynamics.
- **Readout Network:** $g_{\theta_3} : \mathbb{R}^{d_H} \to \mathbb{R}^{d_Y}$, mapping the hidden state to the observable output space.

Here, $\mathbb{R}^{d_X}$, $\mathbb{R}^{d_H}$, and $\mathbb{R}^{d_Y}$ denote the input, latent, and output spaces, respectively, with $d_X, d_H, d_Y \in \mathbb{N}$.

2.4.2 The Model Framework

The NJ-ODE model is defined as follows:

$$
\begin{aligned}
H_0 &= \rho_{\theta_2}(X_0, 0), \\
dH_t &= f_{\theta_1}\Big(H_{t-}, X_{\tau(t)}, \tau(t), t - \tau(t)\Big)\, dt + \left(\rho_{\theta_2}(X_t, H_{t-}) - H_{t-}\right) du_t, \\
Y_t &= g_{\theta_3}(H_t),
\end{aligned}
$$

where $H_t \in \mathbb{R}^{d_H}$ denotes the hidden (latent) state process and $Y_t \in \mathbb{R}^{d_Y}$ represents the model's output. The function $u_t := \sum_{i=1}^{n} 1_{[t_i,\infty)}(t)$ is a counting process that increases at discrete observation times t_i, while $\tau(t)$ denotes the most recent observation before time t. The set of parameters $\theta := (\theta_1, \theta_2, \theta_3)$ encompasses all learnable weights across the three neural networks and optimized during training.

This formulation allows the model to evolve continuously between observations via the ODE network f_{θ_1}, while the jump network ρ_{θ_2} deterministically updates the hidden state at each observation time, enabling the NJ-ODE to capture complex dynamics.

Another way to write the NJ-ODE:

$$
\begin{aligned}
h_{t_{i+1}^-} &:= \text{ODESolve}(f_{\theta_1}, (h_t, x_{t_i}, t_i, t - t_i), (t_i, t_{i+1})), \\
h_{t_{i+1}} &:= \rho_{\theta_2}(x_{t_{i+1}}).
\end{aligned}
$$

Herrera et al. (2021) demonstrated that, when trained using suitable objectives, the NJ-ODE effectively approximates the conditional expectation of the underlying process, yielding optimal L^2 predictions. This ensures that the model not only fits observed data points but also generalizes for forecasting tasks.

Building on this framework, F. Krach et al. (2022) introduced the path-dependent NJ-ODE, extending the original model to capture temporal dependencies beyond the current input. This extension is motivated by applications in which the dynamics of a system depends not only on the most recent input but also on the entire history of the input trajectory. To model such dependencies, the authors incorporate the signature transform from rough path theory. This technique provides a compact, non-parametric representation of the history of a path, allowing the model to capture complex temporal dependencies(Chevyrev and Kormilitzin, 2016).

The resulting path-dependent NJ-ODE is formulated as:

$$
H_0 = \rho_{\theta_2}(0, 0, \pi_m(0), M_0 \odot X_0),
$$

$$
\mathrm{d}H_t = f_{\theta_1}\big(H_{t^-}, t, \tau(t), \pi_m(\tilde{X}_{\leq \tau(t)})\big)\, \mathrm{d}t + \big[\rho_{\theta_2}\big(H_{t^-}, t, \pi_m(\tilde{X}_{\leq \tau(t)}), M_t \odot X_t\big) - H_{t^-}\big]\, \mathrm{d}u_t,
$$

$$Y_t = g_{\theta_3}(H_t),$$

where $\pi_m(\tilde{X}\leq \tau(t))$ denotes the truncated signature of the interpolated input path $\tilde{X}\leq \tau(t)$ up to order m. The path $\tilde{X}_{\leq\tau(t)}$ is constructed via forward-fill interpolation, which preserves causality by ensuring that no future information leaks into the model at any time t. The element-wise product $M_t \odot X_t$ allows for masking certain input features, providing model to incorporate irregularly sampled or missed data.

2.4.3 Objective Function

The training objective of the NJ-ODE framework is to approximate the conditional expectation of the target process X, which corresponds to the optimal predictor in the L^2-norm sense. The objective function Ψ is defined as follows:

$$\Psi(Y) := \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\frac{1}{n}\sum_{i=1}^{n}\left(\|M_i \odot (X_{t_i} - Y_{t_i})\|_2 + \left\|M_i \odot (Y_{t_i} - Y_{t_i^-})\right\|_2\right)^2\right], \quad (2.8)$$

where Y is A-adapted process, M_i is a masking operator that accounts for missing observations, and the expectation is taken over the product probability space $(\Omega \times \tilde{\Omega}, \mathcal{F} \otimes \tilde{\mathcal{F}}, \mathbb{P} \times \tilde{\mathbb{P}})$.

The training loss function $\mathcal{L}$ is defined by evaluating this objective on the model's output for a given parameter configuration θ:

$$\mathcal{L}(\theta) := \Psi(Y^\theta(X)),$$

where $Y^\theta(X)$ represents the output of the NJ-ODE model with parameters θ, conditioned on the observed data from the process X.

This loss function incorporates two key components:

- The *jump term* $\|X_{t_i} - Y_{t_i}\|_2$, which encourages the jump network ρ_{θ_2} to produce accurate state updates upon receiving new observations.
- The *continuity term* $\left\|Y_{t_i} - Y_{t_i^-}\right\|_2$, which penalizes discontinuities in the model output. This term drives the ODE dynamics network f_{θ_1} to evolve the latent state H_t smoothly over time to closely follow the conditional expectation path.

Both terms also contribute to training the readout network g_{θ_3}, which maps the latent state to the output space. The model is thus trained to maintain accurate predictions both at and between observation times.

This formulation corresponds to the loss function referred to as `"standard"` in the original NJ-ODE implementation.

Chapter 3

Methodology

This chapter presents the methodological framework developed to adapt the NJ-ODE model for generative purposes. We begin by introducing the Itô diffusion process, which serves as the model for the dynamics of underlying stochastic process. We then describe the modifications made to the original NJ-ODE framework in order to enable synthetic data generation. In particular, we detail how the model was adapted to estimate both the drift and diffusion components of stochastic processes, a key step in extending NJ-ODE from a forecasting tool to a generative framework.

3.1 Itô Diffusion Processes

Itô diffusion processes form a fundamental class of continuous-time stochastic processes that are widely used to model systems evolving under both deterministic trends and random perturbations. These processes are described by stochastic differential equations (SDEs) of the general form:

$$dX_t = \mu(t, X_{[0,t]})\, dt + \sigma(t, X_{[0,t]})\, dW_t, \tag{3.1}$$

Let $\{W_t\}_{t\in[0,T]}$ be a d_W-dimensional Brownian motion defined on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$, and let $X := (X_t)_{t\in[0,T]}$ be the solution process, where $X_t \in \mathbb{R}^{d_X}$ denotes the state at time t. The measurable functions $\mu : [0,T] \times \mathbb{R}^{d_X} \to \mathbb{R}^{d_X}$ and $\sigma : [0,T] \times \mathbb{R}^{d_X} \to \mathbb{R}^{d_X \times d_W}$ are referred to as the drift and diffusion coefficients, respectively (Bass, 2011; Björk, 2009; Shreve, 2004). Further details are provided in the Appendix.

The drift term $\mu(t, X_t)$ encodes the deterministic direction of movement, while the diffusion term $\sigma(t, X_t)\, dW_t$ captures the randomness introduced by the Brownian motion. The dependence on the entire path history X_t allows for non-Markovian dynamics, extending the modeling flexibility to incorporate memory effects.

Itô diffusions are central to many applications in science and engineering, particularly in quantitative finance, where they are used to model asset prices, interest rates, and volatility surfaces (Björk, 2009). The ability to accommodate both temporal and historical path dependence makes Itô diffusion processes a powerful framework for modeling stochastic systems.

Geometric Brownian Motion

The GBM is one of the fundamental stochastic processes in quantitative finance, particularly for modeling the evolution of asset prices. Its popularity arises from its analytical tractability. Additionally, it plays a central role in classical models like the Black–Scholes framework for option pricing (Black and Scholes, 1973).

The dynamics of a GBM are described by the SDE

$$dX_t = \mu X_t \, dt + \sigma X_t \, dW_t,$$

where $X_t \in \mathbb{R}_{>0}$ represents the asset price at time t, $\mu \in \mathbb{R}$ is the drift coefficient reflecting the expected rate of return, $\sigma > 0$ is the volatility parameter, and W_t is a standard Brownian motion.

The solution to this SDE has a closed-form expression:

$$X_t = X_0 \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right) t + \sigma W_t\right),$$

which shows that the logarithm of X_t is normally distributed, implying that X_t itself follows a log-normal distribution. This property makes GBM particularly appealing for modeling financial quantities that must remain strictly positive, such as stock prices.

Despite its simplicity and assumptions like constant drift and volatility, GBM is a cornerstone of financial mathematics. It acts as a benchmark for more complex models, such as those incorporating stochastic volatility and jump-diffusion processes (Björk, 2009; Shreve, 2004).

Ornstein–Uhlenbeck Process

The OU process is a classical example of a mean-reverting stochastic process. In finance, the OU process has been used to model interest rates, stochastic volatility, and other mean-reverting quantities (Björk, 2009; Uhlenbeck and Ornstein, 1930; Vasicek, 1977).

The dynamics of the OU process are governed by the SDE:

$$dX_t = k(m - X_t) \, dt + \sigma \, dW_t,$$

where $X_t \in \mathbb{R}$ denotes the state of the process at time t, $k > 0$ is the rate of mean reversion, $m \in \mathbb{R}$ is the long-term mean level, $\sigma > 0$ is the volatility parameter, and W_t is a standard Brownian motion.

The term $k(m - X_t)$ represents the mean-reverting drift, which pulls the process toward the long-term mean m at rate k.

The exact solution of the OU process is given by:

$$X_t = X_0 e^{-kt} + m(1 - e^{-kt}) + \sigma \int_0^t e^{-k(t-s)}\, dW_s,$$

which illustrates its dependence on both the initial condition and the stochastic evolution driven by Brownian motion.

The OU process plays a central role in various financial models. For instance, it forms the basis of the Vasicek interest rate model (Vasicek, 1977).

3.2 NJ-ODE as Generative Method I

In this section, we present a method for extracting instantaneous drift and diffusion estimates from the predictions of a neural network trained to approximate the conditional first and second moments of a stochastic process. Our approach assumes the underlying dynamics follow an Itô diffusion of the general form:

$$dX_t = \mu(t, X_t)\, dt + \sigma(t, X_t)\, dW_t, \tag{3.2}$$

where $X_t \in \mathbb{R}^d_X$ is the state of the process at time t, μ is the drift vector, σ is the diffusion matrix, and W_t is a standard d_W-dimensional Brownian motion. The functions μ and σ may depend on the full path history X_t, allowing for non-Markovian dynamics.

To approximate this process numerically, we discretize the dynamics using the Euler–Maruyama scheme(Kloeden and Platen, 1992) :

$$X_{t+\Delta t} \approx X_t + \mu_t \Delta t + \sigma_t \Delta W_t, \tag{3.3}$$

where $\mu_t = \mu(t, X_t)$, $\sigma_t = \sigma(t, X_t)$, and $\Delta W_t \sim \mathcal{N}(0, \Delta t \cdot I_{d_W})$ represents a Brownian increment. Our goal is to estimate μ_t and σ_t based on the conditional expectations of $X_{t+\Delta t}$ and $X_{t+\Delta t} X_{t+\Delta t}^\top$, learned by our NJ-ODE model.

Drift Estimation from First Moments

The NJ-ODE model is trained to predict the conditional expectation $\mathbb{E}[X_{t+\Delta t} \mid \mathcal{A}_t]$, where $\mathcal{A}_t$ represents the available information up to time t, and noting that $\mathbb{E}[\Delta W_t \mid \mathcal{A}_t] = 0$, we have:

$$\mathbb{E}[X_{t+\Delta t} \mid \mathcal{A}_t] \approx X_t + \mathbb{E}[\mu_t \mid \mathcal{A}_t] \Delta t. \tag{3.4}$$

Rearranging terms yields an estimate for the conditional drift:

$$\hat{\mu}_t := \mathbb{E}[\mu_t \mid \mathcal{A}_t] \approx \frac{\mathbb{E}[X_{t+\Delta t} \mid \mathcal{A}_t] - X_t}{\Delta t}. \tag{3.5}$$

Diffusion Estimation from Second Moments

To estimate the instantaneous diffusion, we consider the conditional expectation of second-order products. Applying Itô's formula to the product $X_t^i X_t^j$ yields:

$$d(X^i X^j)_t = X_t^j dX_t^i + X_t^i dX_t^j + d[X_t^i, X_t^j].$$

Substituting the SDE terms gives:

$$d(X^i X^j)_t = X_t^j \mu_t^i dt + X_t^j \sigma_t^i dW_t + X_t^i \mu_t^j dt + X_t^i \sigma_t^j dW_t + \sigma_t^i \sigma_t^j dt.$$

Discretising using the Euler–Maruyama scheme and taking conditional expectations leads to:

$$\mathbb{E}[(X^i X^j)_{t+\Delta t} \mid \mathcal{A}_t] \approx (X^i X^j)_t + X_t^j \hat{\mu}_t^i \Delta t + X_t^i \hat{\mu}_t^j \Delta t + \mathbb{E}[\sigma_t^i \sigma_t^j \mid \mathcal{A}_t] \Delta t.$$

Rearranging terms yields the estimate of the covariance matrix $\Sigma_t = \sigma_t \sigma_t^\top$ as:

$$(\hat{\Sigma}_t)_{ij} := \mathbb{E}[\sigma_t^i \sigma_t^j \mid \mathcal{A}_t] \approx \frac{\mathbb{E}[(X^i X^j)_{t+\Delta t} \mid \mathcal{A}_t] - (X^i X^j)_t}{\Delta t} - X_t^i \hat{\mu}_t^j - X_t^j \hat{\mu}_t^i. \tag{3.6}$$

To generate a new point using the Euler–Maruyama scheme, we require a matrix $\hat{\sigma}_t$ such that $\hat{\sigma}_t \hat{\sigma}_t^\top = \hat{\Sigma}_t$. If $\hat{\Sigma}_t$ is positive semi-definite, this can be computed via the Cholesky decomposition. Theoretically, this condition is satisfied since, for any non-zero vector $x \in \mathbb{R}^{d_X}$,

$$x^\top \hat{\Sigma}_t x = \mathbb{E}[x^\top \Sigma_t x \mid \mathcal{A}_t] \geq 0.$$

However, due to estimation errors and numerical approximations, the computed matrix $\hat{\Sigma}_t$ may not be positive semi-definite in practice.

A common strategy to ensure positive definiteness is to apply a regularization by adding a small multiple of the identity matrix(Ledoit and Wolf, 2004):

$$\hat{\Sigma}_t^{(\alpha)} = \hat{\Sigma}_t + \alpha I_d, \quad \text{for some } \alpha > 0.$$

In our experiments, the estimated covariance matrices occasionally had negative eigenvalues, violating the positive semi-definiteness required for Cholesky decomposition. Attempts to resolve this issue by adding a small multiple of the identity matrix did not yield satisfactory results.

3.3 NJ-ODE as a Generative Method II

In this section, we present a second generative modeling approach using the NJ-ODE framework, which enables the direct estimation of the instantaneous covariance structure of a stochastic process. Unlike Method I, which derives diffusion estimates from second-order moment predictions, Method II only uses first-order moment predictions.

Direct Estimation of Diffusion

It can be shown that Equation (3.6) can be equivalently written as

$$\hat{\Sigma}_t = \frac{1}{\Delta t} \mathbb{E}\left[(X_{t+\Delta t} - X_t)(X_{t+\Delta t} - X_t)^\top \,\middle|\, \mathcal{A}_t \right],$$

which expresses the instantaneous covariance in terms of the conditional second moment of the process increments.

Based on this representation, we define a process Z_t as

$$Z_t := (X_{t+\Delta t} - X_t)(X_{t+\Delta t} - X_t)^\top.$$

The conditional expectation $\mathbb{E}[Z_t \mid \mathcal{A}_t]$ directly yields an estimate of $\Delta t \cdot \hat{\Sigma}_t$.

The above definition is valid under the assumption that the process X is fully observed. However, in practical scenarios where observations of X are incomplete or irregularly spaced, we generalize the definition of Z_t by replacing X_t with the most recent observation available before time t, denoted $\tau(t)$. The generalized process is given by

$$Z_t := (X_t - X_{\tau(t)})(X_t - X_{\tau(t)})^\top.$$

The goal is to train the NJ-ODE model to estimate the conditional expectation of Z_t, using an architecture that ensures the positive semi-definiteness of the covariance matrix. It is important to note that if $X_t \in \mathbb{R}^{d_X}$, then the corresponding process $Z_t \in \mathbb{R}^{d_X \times d_X}$ is a matrix-valued quantity. To enable processing within the NJ-ODE framework, each Z_t matrix is first flattened into a vector of length d_X^2, which serves as the model input. The NJ-ODE model then outputs a vector of the same dimension, which is subsequently reshaped back into a $d_X \times d_X$ matrix denoted as Y_t, representing the model's output at time t. The estimated covariance is defined as:

$$\hat{\Sigma}_t := \frac{Y_t Y_t^\top}{\Delta t}.$$

This formulation guarantees that the estimated covariance matrix is symmetric and positive semi-definite by construction.

New Loss Function

To guide the NJ-ODE model during training, we begin by recalling the original loss function, which is designed to approximate the conditional expectation of the target process X in the sense of the L_2-norm. The loss functional is given by:

$$\Psi(X, Y) := \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\frac{1}{n}\sum_{i=1}^{n}\left(\|M_i \odot (X_{t_i} - Y_{t_i})\|_2 + \left\|M_i \odot (Y_{t_i} - Y_{t_i^-})\right\|_2\right)^2\right],$$

where $M_i \in \{0,1\}^d$ is a binary mask that encodes which components of the process are observed at time t_i. The first term penalizes the model for inaccurate predictions at observed time points, while the second term regularizes the jump size between consecutive model outputs.

In the generative setting, our goal extends beyond estimating the conditional expectation of X. We also seek to estimate the instantaneous volatility. Since the volatility structure is captured by the Z_t process, we adapt the loss function accordingly.

To that end, we replace the target X with Z and compare it against the product of the model output, $Y_t Y_t^\top$. The modified loss function used to train the model to recover the diffusion component is thus given by:

$$\mathcal{L}_{\text{vol}}(\theta) := \Psi(Z, YY^\top).$$

This formulation allows the NJ-ODE framework to learn second-order structure from the data. The flexibility of this objective is crucial for developing a generative model that accurately captures both drift and diffusion dynamics of stochastic processes.

Drift and Volatility Estimation

The drift and volatility components are learned using two separate NJ-ODE models. The first model is trained on the original process X to predict the conditional mean $\mathbb{E}[X_{t+\Delta t} \mid \mathcal{A}_t]$, from which the drift is estimated as:

$$\hat{\mu}_t \approx \frac{\mathbb{E}[X_{t+\Delta t} \mid \mathcal{A}_t] - X_t}{\Delta t}.$$

The second model targets the diffusion term, learning to predict the conditional expectation of Z_t. The model is trained by minimizing the loss function $\mathcal{L}_{\text{vol}}(\theta)$, such that the output Y_t and $Y_t Y_t^\top$ approximates $[Z_{t+\Delta t} \mid \mathcal{A}_t]$.

Given the model output Y_t, the instantaneous covariance matrix is estimated as:

$$\hat{\Sigma}_t := \frac{Y_t Y_t^\top}{\Delta t}.$$

The corresponding volatility $\hat{\sigma}_t$ is then obtained as the Cholesky factor of $\hat{\Sigma}_t$.

The advantage of this separation lies in the ability to tailor each model's architecture and loss to the specific nature of the quantity being learned.

Generating Synthetic Data

Once the models for $\hat{\mu}_t$ and $\hat{\Sigma}_t$ are trained, the generative process proceeds via the Euler–Maruyama scheme:

$$X_{t+\Delta t} = X_t + \hat{\mu}_t \Delta t + \hat{\sigma}_t \Delta W_t,$$

where $\Delta W_t \sim \mathcal{N}(0, \Delta t \cdot I_{d_W})$, and $\hat{\sigma}_t$ is obtained as as the Cholesky factor of $\hat{\Sigma}_t = \frac{Y_t Y_t^\top}{\Delta t}$. This formulation overcomes the issues related to negative eigenvalues in the covariance matrix.

By iteratively applying this scheme, the NJ-ODE model generates synthetic sample paths. This approach provides a framework for generative modeling of continuous-time stochastic processes.

Chapter 4

Experiments for Method I

All implementations were done using PyTorch. The code is available at: https://github.com/zhaksylykov/thesis.

4.1 Implementation Details

4.1.1 Datasets

Multiple datasets were generated by simulating $N = 10{,}000$ sample paths of a 3-dimensional OU process and a 1-dimensional GBM. The Euler-Maruyama discretisation scheme was used for the simulations. Each simulation was carried out on an equidistant time grid with step size $\Delta t = 0.01$ over the interval $[0, T]$, with $T = 1$. This results in 101 time points per path.

All sample paths are fully observed at every time point, meaning the trajectories contain no missing data.

Each dataset was subsequently split into training and testing subsets using an 80%/20% partition. The specific parameters used for a 3-dimensional OU process and a 1-dimensional GBM are detailed below.

Ornstein-Uhlenbeck 3D

- **SDE:**
$$dX_t = -\Theta(X_t - \mu)\,dt + \Sigma\,dW_t,$$
where $X_t \in \mathbb{R}^3$, $\Theta \in \mathbb{R}^{3\times 3}$ is the mean-reversion matrix, $\mu \in \mathbb{R}^3$ is the long-term mean vector, $\Sigma \in \mathbb{R}^{3\times 3}$ is the volatility matrix, and $W_t \in \mathbb{R}^3$ is a 3-dimensional Brownian motion.
- **Conditional expectation:**
$$\mathbb{E}(X_{t+s} \mid \mathcal{A}_t) = e^{-\Theta s} X_t + \left(I - e^{-\Theta s}\right)\mu.$$

- **Parameters used:**

$$\mu = \begin{bmatrix} 1.2 \\ 1.0 \\ 1.5 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 0.2 & 0.1 & 0.1 \\ 0.1 & 0.25 & 0.1 \\ 0.1 & 0.1 & 0.3 \end{bmatrix}, \quad \Theta = \begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 0.3 \end{bmatrix}, \quad X_0 = \begin{bmatrix} 1.0 \\ 1.5 \\ 2.0 \end{bmatrix},$$

Geometric Brownian Motion 1D

- **SDE:**

$$dX_t = \mu X_t \, dt + \sigma X_t \, dW_t,$$

  where W_t is a 1-dimensional Brownian motion.

- **Conditional expectation:**

$$\mathbb{E}(X_{t+s} \mid \mathcal{A}_t) = X_t e^{\mu s}.$$

- **Parameters used:** $\mu = 2$, $\sigma = 0.3$, $X_0 = 1$.

4.1.2 Architecture

In all experiments, the dimension of the latent (hidden) state is set to $d_H = 10$. The functions f_{θ_1}, $\tilde{g}_{\theta_3}$, and $\tilde{\rho}_{\theta_2}$ are implemented as feedforward neural networks, each comprising two hidden layers with 50 units per layer and using the tanh activation function. This activation function is chosen due to its ability to bound outputs within a fixed range. This helps stabilize training when the hidden state h and input observation x can take on large or unbounded values.

The readout network g_{θ_3} and the jump network ρ_{θ_2}, are then constructed as residual versions of $\tilde{g}_{\theta_3}$ and $\tilde{\rho}_{\theta_2}$.

To prevent overfitting and improve generalization, dropout with a rate of 0.1 is applied after each nonlinearity within the networks.

The continuous-time dynamics of the latent state are solved using the explicit Euler method, which is the simplest approach for numerically integrating the ODE component of the model.

4.1.3 Training

The neural networks were trained using the Adam optimizer with a learning rate of 0.001. Training was conducted over 100 epochs with a batch size of 200. All network parameters were initialized randomly. No additional hyperparameter tuning was performed. We employed a modified version of the original NJ-ODE loss function, referred to as the `"easy"` loss in the implementation. It is defined as:

$$\Psi(Y) := \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\frac{1}{n}\sum_{i=1}^{n}\left(\|M_i \odot (X_{t_i} - Y_{t_i})\|_2 + \left\|M_i \odot (X_{t_i} - Y_{t_i^-})\right\|_2\right)^2\right], \quad (4.1)$$

This loss differs from the standard formulation by replacing the term $\|Y_{t_i} - Y_{t_i^-}\|_2$ with $\|X_{t_i} - Y_{t_i^-}\|_2$. The adjustment mitigates the risk of local minima and enhances training outcomes. This was empirically demonstrated by (F. T. O. Krach (2025)).

By directly comparing both the model's jump prediction Y_{t_i} and the pre-jump state $Y_{t_i^-}$ against the true observation X_{t_i}, this formulation penalizes discrepancies more effectively. In particular, it discourages the model from producing intermediate values $Y_{t_i} \neq X_{t_i}$ that reduce the loss artificially. As a result, the `"easy"` loss improves stability during training and promotes better convergence to the true conditional expectation (F. T. O. Krach, 2025).

4.2 Data Generation: Ornstein-Uhlenbeck 3D

We employ Method I to generate data based on a 3-dimensional OU process $X = (X_1, X_2, X_3)$, where $X \in \mathbb{R}$ with d = 3 İn the initial setup, the NJ-ODE model was trained directly on the raw process values of X, such that the input and output dimensions matched the process dimensionality. While this configuration is sufficient for modeling the first-order dynamics of X_t, it is inadequate for capturing the second-order dynamics of X_t. Modeling dynamics of $(XX^\top)_t$ is central to our generative modeling approach.

To address this limitation, we extended the input representation to include pairwise products and squared terms of the components of X, thereby capturing both linear and second-order interactions. Specifically, we define the augmented input vector as

$$\tilde{X} = (X_1, X_2, X_3, X_1^2, X_2^2, X_3^2, X_1X_2, X_1X_3, X_2X_3),$$

which consists of all unique monomials of degree up to two. This transformation results in an input and output dimensionality of $\frac{d^2+3d}{2}$; for $d = 3$, this yields 9 features.

This extended feature representation allows the model to learn the evolution of second-moment structure over time. The training performance on test data using the augmented input representation is illustrated in the figure below. The close alignment between predictions and ground truth suggests successful learning of the process dynamics.

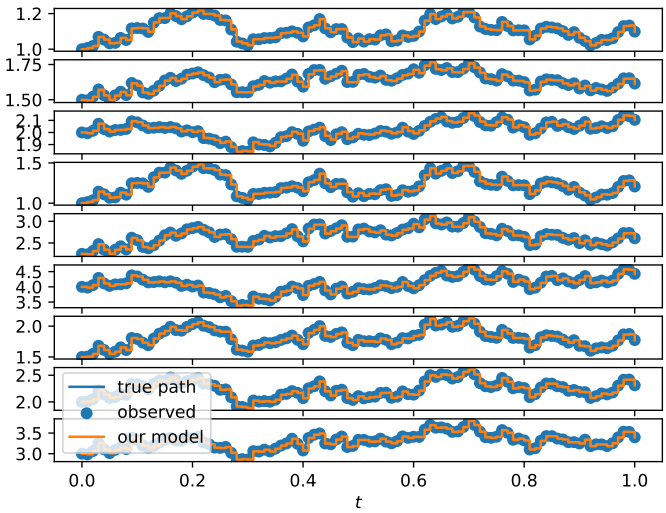


Figure 4.1: Training results for the NJ-ODE model with augmented input features.

To iteratively generate new sample paths, we begin from a given initial state X_0 and evolve the process forward in time using the conditional drift and covariance estimated by the trained NJ-ODE model. At each time step t, the model produces an estimate of the conditional mean $\hat{\mu}_t$ and the conditional covariance matrix $\hat{\Sigma}_t$. New data points are then generated using the Euler–Maruyama discretisation scheme:

$$X_{t+\Delta t} = X_t + \hat{\mu}_t \Delta t + \hat{\sigma}_t \Delta W_t,$$

where $\Delta W_t \sim \mathcal{N}(0, \Delta t \cdot I)$ denotes a Brownian increment and $\hat{\sigma}_t$ is a matrix satisfying $\hat{\sigma}_t \hat{\sigma}_t^\top = \hat{\Sigma}_t$. In practice, $\hat{\sigma}_t$ is computed via the Cholesky decomposition of $\hat{\Sigma}_t$, which requires $\hat{\Sigma}_t$ to be positive semi-definite.

Example

To illustrate the data generation procedure based on the NJ-ODE model, consider the first iteration, starting from the initial condition

$$X_0 = \begin{bmatrix} 1.0 \\ 1.5 \\ 2.0 \end{bmatrix}.$$

From this input, we construct the augmented input vector $\tilde{X}_0 \in \mathbb{R}^9$, incorporating both linear and second-order features as follows:

$$\tilde{X}_0 = (1.0, 1.5, 2.0, 1.0, 2.25, 4.0, 1.5, 2.0, 3.0)\,.$$

This vector is then provided as input to the trained NJ-ODE model, which outputs a predicted vector $\hat{\hat{X}}_0 \in \mathbb{R}^9$:

$$\hat{\hat{X}}_0 = (1.0002, 1.4997, 1.9997, 1.0004, 2.2492, 3.9987, 1.4999, 2.0003, 2.9990)\,.$$

From the first three components of $\hat{\hat{X}}$, we extract the model's estimate of the conditional mean:

$$\mathbb{E}[X_{0.01} \mid \mathcal{A}_0] = \begin{bmatrix} 1.0002 \\ 1.4997 \\ 1.9997 \end{bmatrix}.$$

The remaining six components of $\hat{\hat{X}}$ correspond to second-order statistics. Using them, we reconstruct the estimated second-moment matrix:

$$\mathbb{E}\left[X_{0.01}X_{0.01}^\top \mid \mathcal{A}_0\right] = \begin{bmatrix} 1.0004 & 1.4999 & 2.0003 \\ 1.4999 & 2.2492 & 2.9990 \\ 2.0003 & 2.9990 & 3.9987 \end{bmatrix}.$$

We compute the drift and covariance estimates using the following formulas:

$$\hat{\mu}_0 = \frac{\mathbb{E}[X_{0.01} \mid \mathcal{A}_0] - X_0}{0.01}, \quad (\hat{\Sigma}_0)_{i,j} = \frac{\mathbb{E}[X_i X_j \mid \mathcal{A}_0] - X_i X_j}{0.01} - X_i\hat{\mu}_j - X_j\hat{\mu}_i.$$

Evaluating this expression numerically yields:

$$\hat{\mu}_0 = \begin{bmatrix} 0.0205 \\ -0.0347 \\ -0.0345 \end{bmatrix}, \qquad \hat{\Sigma}_0 = \begin{bmatrix} 0.0002 & -0.0038 & 0.0224 \\ -0.0038 & 0.0235 & 0.0171 \\ 0.0224 & 0.0171 & 0.0880 \end{bmatrix}.$$

These estimates of the conditional mean and covariance form the basis for generating the next point in the trajectory using the Euler–Maruyama scheme. However, care must be taken to ensure that $\hat{\Sigma}_0$ is positive semi-definite. In this example, the initial covariance estimate was found to have eigenvalues approximately equal to (–0.023, 0.017, 0.038), indicating a violation of this condition. Consequently, the Cholesky decomposition is not feasible at this step.

This issue highlights a practical challenge in modeling conditional covariance structures with neural networks, while ensuring that the predicted matrices remain symmetric and positive semi-definite. To address this, one may consider applying regularisation techniques or enforcing explicit constraints during training to guarantee the desired properties.

4.3 Data Generation: Geometric Brownian Motion 1D

In this section, we apply Method I to generate synthetic data based on a 1-dimensional GBM process $X \in \mathbb{R}^1$. The goal is to model both the process X_t and its square X_t^2 at each time step. In this section, we consider two approaches. First is a joint model that learns both X and X^2 simultaneously. And the second one is a separate modeling approach where two independent models are trained for X and X^2, respectively.

Joint Modeling of X and X^2

In the first approach, we generate a dataset containing the values of X_t at each time step. The NJ-ODE framework supports flexible input and output dimensionalities, which allows us to extend the model's input representation. Specifically, we utilize the function called `'func_appl_X'` in the NJ-ODE framework. By setting it to `"power-2"`, automatically augments the input by including second-order monomials such as X_t^2.

With this configuration, the model is trained on the augmented dataset that includes both the original process X_t and its square X_t^2. The resulting architecture thus learns to jointly model both the first-order and the second-order moments of the process.

Figure 4.2 displays the training results on the test set. The close agreement between predicted and actual trajectories indicates that the NJ-ODE model has successfully captured the dynamics of the process.

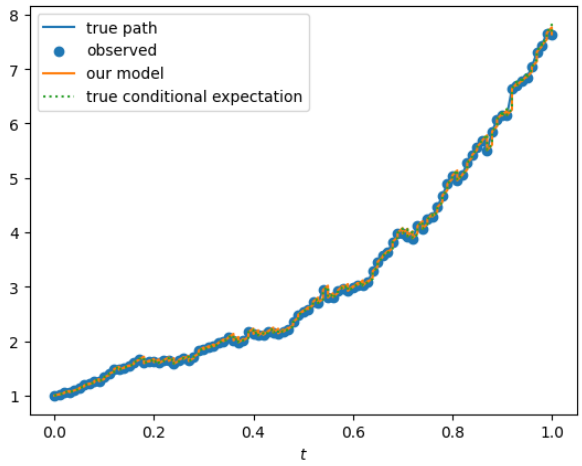


Figure 4.2: Training results of the NJ-ODE model for joint modeling of X and X^2.

After training the NJ-ODE model on the augmented dataset containing both X_t and X_t^2, we use the learned model to generate synthetic data. Starting

from an initial value X_0, the model is used to estimate $\hat{\mu}_t$ and $\hat{\Sigma}_t$ at each time step. These estimates are then used within the Euler–Maruyama scheme to simulate the next point in the trajectory:

$$X_{t+\Delta t} = X_t + \hat{\mu}_t \Delta t + \hat{\sigma}_t \Delta W_t,$$

where $\hat{\sigma}_t$ is obtained via taking square root of $\hat{\Sigma}_t$, and $\Delta W_t \sim \mathcal{N}(0, \Delta t)$ denotes a standard Brownian increment.

This process is repeated iteratively until the final time horizon $T = 1$ is reached. At each step, the model conditions its prediction on previously generated values, enabling the construction of fully synthetic trajectories.

To evaluate the quality of the generated data, we compare trajectories sampled from the trained NJ-ODE model to those generated from the true GBM dynamics using known parameters. Figure 4.3 displays five synthetic (`"fake"`) and five reference (`"real"`) trajectories.

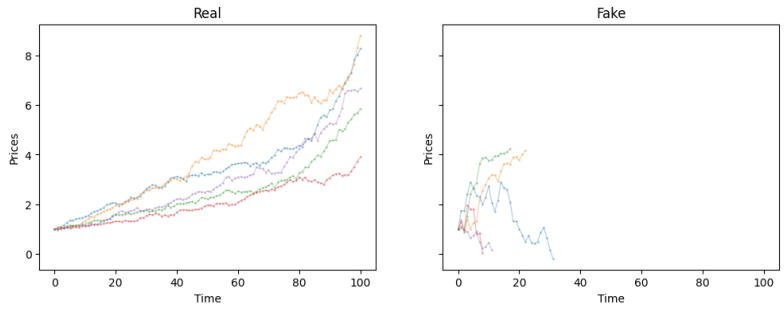


Figure 4.3: Comparison of 5 real and generated paths

As evident from the figure, the synthetic trajectories display higher volatility than the reference GBM paths. This suggests that the model tends to overestimate the variability of the process. Additionally, many of the generated trajectories terminate prematurely, failing to reach the terminal time.

This early termination is attributed to numerical instabilities encountered during sampling. In particular, some estimated covariance matrices $\hat{\Sigma}_t$ are not positive. As a result, the computation of $\hat{\sigma}_t$ fails, which in turn prevents the generation of the next point in the trajectory via the Euler–Maruyama scheme.

An important empirical finding from our experiments is that the dimensionality of the hidden state in the NJ-ODE architecture has an effect on the volatility of the generated trajectories. Models with larger hidden states tend to produce trajectories with reduced volatility. This effect can be attributed to the increased representational capacity of larger latent spaces.

Figure 4.4 illustrates the effect of increasing the hidden size to 50. Compared to Figure 4.3, the generated paths exhibit lower volatility.

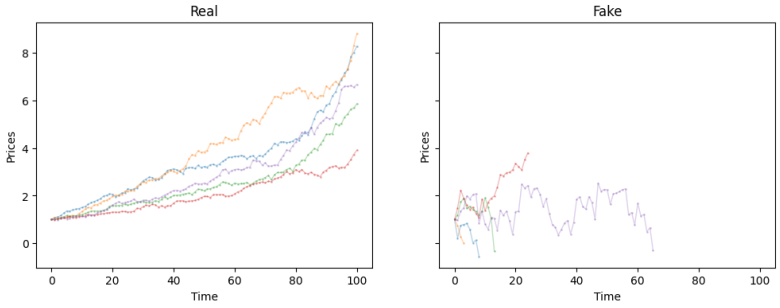
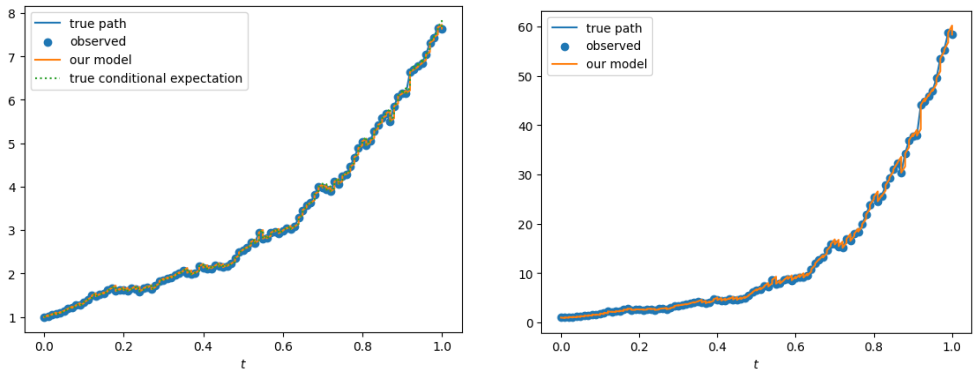


Figure 4.4: Comparison of 5 real and 5 generated paths with hidden size equal to 50.

4.3.1 Separate Modeling of X and X^2

In this approach, we construct two distinct datasets. One containing the values of the process X_t at each discrete time step, and another containing the squared values X_t^2 at the corresponding time steps. The goal is to develop separate models for the first moment of the corresponding process using the NJ-ODE framework.

We train two independent NJ-ODE models. One on the dataset corresponding to X_t, and the other on the dataset corresponding to X_t^2. This decoupled modeling strategy allows each network to specialize in learning the temporal dynamics of its respective target.



(a) NJ-ODE prediction of X_t.

(b) NJ-ODE prediction of X_t^2.

Figure 4.5: Training results for NJ-ODE models trained separately on X_t and X_t^2.

Figure 4.5 shows the training performance of both models on the test set. The close alignment between predicted and actual trajectories suggests that

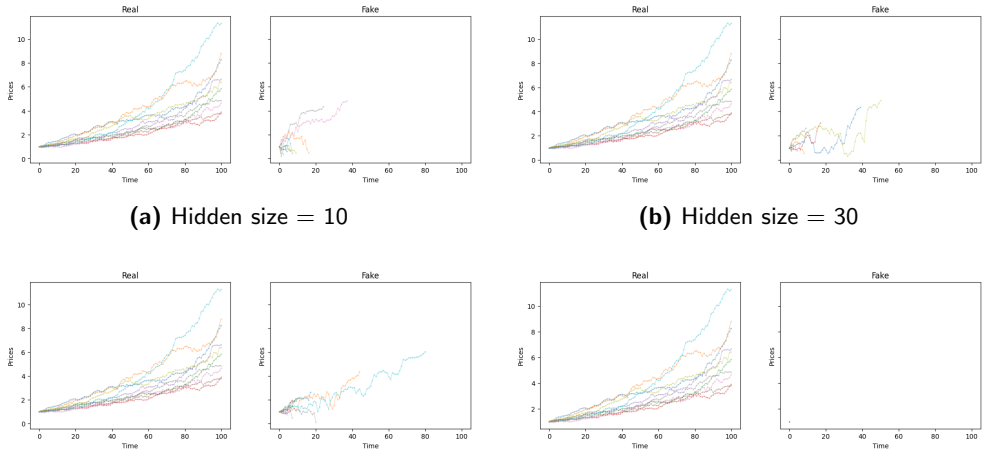
each model successfully captured the underlying structure of its respective process.

Following training, the NJ-ODE models are used to estimate the instantaneous drift $\hat{\mu}_t$ and diffusion $\hat{\Sigma}_t$ at each time step. These quantities are then employed within the Euler–Maruyama scheme to simulate synthetic trajectories.

The generation procedure begins from an initial condition X_0. At each time step, the model estimates $\hat{\mu}_t$ and $\hat{\Sigma}_t$, which are used to compute the next value $X_{t+\Delta t}$. This process is repeated iteratively until the final time $T = 1$ is reached, resulting in a full synthetic path.

For each configuration, we compare 10 generated trajectories to 10 reference paths simulated from the true GBM model. The results are shown in Figure 4.6.

To assess the impact of model capacity on the quality of generated paths, we performed this generative experiment using NJ-ODE models with varying hidden state dimensions: 10, 30, 40, and 50. For each configuration, we compare 10 generated trajectories to 10 reference paths simulated from the true GBM model. The results are shown in Figure 4.6.



(a) Hidden size = 10 **(b)** Hidden size = 30

(c) Hidden size = 40 **(d)** Hidden size = 50

Figure 4.6: Comparison of 10 real and generated paths for varying hidden state sizes

As illustrated in Figure 4.6, the hidden state dimension of the NJ-ODE model affects the variability of the generated paths. Specifically, increasing the hidden size leads to reduced volatility. This can be attributed to the enhanced expressive power of deeper latent representations, which allows the model to capture more nuanced temporal dependencies.

However, for the model with hidden size 50, a critical issue arises. The estimated initial covariance matrix $\hat{\Sigma}_0$ is not positive semi-definite, which prevents the computation of its Cholesky decomposition and thus halts the trajectory generation at the very first step.

This limitation motivates the development of a modified approach. The next chapter presents a modified generative approach that explicitly enforces the positive semi-definiteness of $\hat{\Sigma}_t$ during training by redefining the modeling and loss function design.

Chapter 5

Experiments for Method II

All implementations were done using PyTorch. The code is available at: https://github.com/zhaksylykov/thesis.

5.1 Implementation Details

5.1.1 Datasets

In this chapter, we adopt the same dataset simulation procedures as described previously, including the generation of sample paths for a 3-dimensional OU process and a 1-dimensional GBM. Additionally, we introduce a new dataset based on a 1-dimensional OU process.

To support the second generative approach, we construct the process Z. By construction, each Z path contains only 100 time points, while each X path has 101 time points. To ensure compatibility with the NJ-ODE framework, we prepend a zero matrix to the beginning of each Z path, setting $Z_0 = 0$. This is consistent with the fact that at least two observations of X are required to compute the first non-trivial value of Z.

Ornstein–Uhlenbeck 1D

- **SDE:**
$$dX_t = -k(X_t - m)\,dt + \sigma\,dW_t,$$
where W_t is a 1-dimensional Brownian motion.
- **Conditional expectation:**
$$\mathbb{E}(X_{t+s} \mid \mathcal{A}_t) = X_t e^{-ks} + m(1 - e^{-ks}).$$
- **Parameters used:** $k = 0.3$, $m = 1.5$, $\sigma = 0.3$, $X_0 = 2$.

5.1.2 Architecture and Training

The model architecture and training procedure used in this method for process X follow exactly the setup described in the previous chapter.

A modification to the loss function is required when training the model on the Z dataset. Unlike the model trained on X, which uses the standard NJ-ODE loss function, the model for Z is trained using a new loss function. In particular, we adopt the proposed loss function:

$$\mathcal{L}_{\text{vol}}(\theta) = \Psi(Z, YY^\top),$$

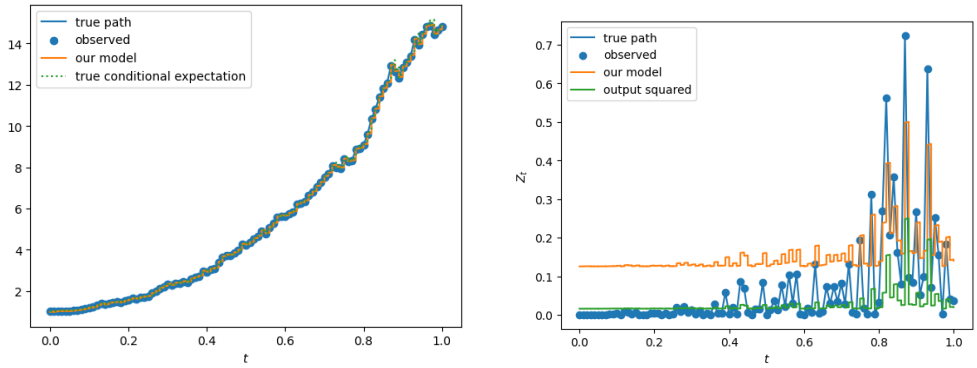
where Y_t is the network output and $Y_t Y_t^\top$ serves as the estimate for the predicted covariance matrix times Δt. This formulation ensures that the predicted volatility estimate remains symmetric and positive semi-definite by construction. In the NJ-ODE codebase, this loss function is named *"easy_vol"*.

5.2 Data Generation: Geometric Brownian Motion 1D

In this section, we generate synthetic data using Method II. While the overall procedure follows the same steps as described in the previous chapter, the key difference lies in the use of a two-model approach. One NJ-ODE model is trained to learn the dynamics of the process X, and a separate model is trained to estimate the process Z.

Separate Training of X and Z

Figure 5.1 presents the training results for the two independent NJ-ODE models. While prior results have demonstrated the model's ability to capture the dynamics of X, this is the first instance illustrating training performance for the Z process. Since the model is trained to output a matrix Y_t such that $Y_t Y_t^\top \approx Z_t$, a direct comparison between the raw model output and the target is not meaningful. We visualize the squared model output $Y_t Y_t^\top$ in green for interpretability.



(a) NJ-ODE prediction of X_t **(b)** NJ-ODE prediction of Z_t

Figure 5.1: Training results for NJ-ODE models trained separately on X_t and Z_t^2.

As seen in Figure 5.1, the Z process is characterized by sharp, non-smooth behavior, posing additional difficulties for accurate modeling. After training, both models were used to generate synthetic data.

Evaluation Under Fully Observed X

To evaluate the generative quality, we compare 10 synthetic sample paths produced by the NJ-ODE framework with 10 reference paths from the true GBM process. Figure 5.2 presents the comparison.

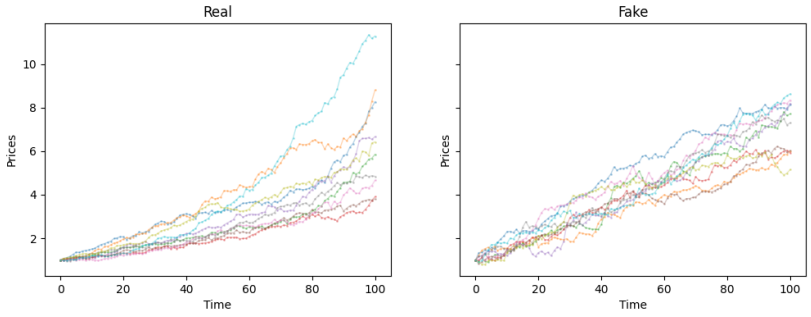


Figure 5.2: Comparison of 10 real and generated GBM paths.

The generated paths fail to exhibit the characteristic exponential growth of GBM. Instead, the synthetic trajectories more closely resemble those of a Brownian motion with constant drift. This can be attributed to how drift is estimated from conditional expectations:

$$\text{Drift estimate at time } t := \frac{\mathbb{E}[X_{t+\Delta t} \mid \mathcal{A}_t] - X_t}{\Delta t}.$$

For GBM:

$$dX_t = \mu X_t\, dt + \sigma X_t\, dW_t \quad \Rightarrow \quad \mathbb{E}[X_{t+\Delta t}] = X_t e^{\mu \Delta t},$$

yielding:

$$\text{drift} = \frac{X_t(e^{\mu \Delta t} - 1)}{\Delta t}.$$

For Brownian Motion with Constant Drift:

$$dX_t = \mu\, dt + \sigma\, dW_t \quad \Rightarrow \quad \mathbb{E}[X_{t+\Delta t}] = X_t + \mu \Delta t,$$

yielding:

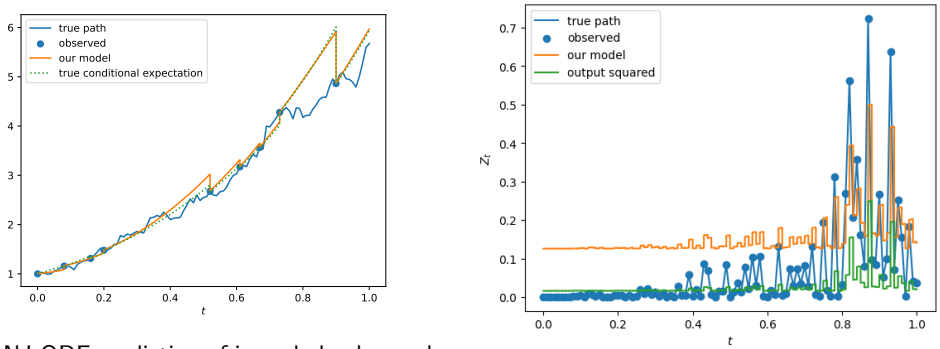
$$\text{drift} = \mu.$$

The NJ-ODE model appears to learn the second form of drift, failing to capture the multiplicative structure of GBM. The model's failure likely arises from training on fully observable data, which may bias it toward short-term linear patterns. Therefore, to address this limitation, we explore training the NJ-ODE model on partially observed data for the X process.

5.2.1 Training with Irregularly Observed X

For modeling of long-term behavior, we randomly subsample the observation times for X_t. Each path is observed at a subset of the 100-point time grid, with each point included independently with probability 0.1. The initial time $t_0 = 0$ is always included. The corresponding Z process remains fully observed.

After training NJ-ODE models for X and Z, Figure 5.3 shows the training performance on the test set.



(a) NJ-ODE prediction of irregularly observed X_t

(b) NJ-ODE prediction of Z_t

Figure 5.3: Training results for NJ-ODE models trained separately on X_t and Z_t.

We then generate 10,000 sample paths using the learned models. Figure 5.4 compares 10 generated and 10 true GBM paths. The synthetic data now better reflects exponential growth, indicating improved drift estimation under irregular observations.

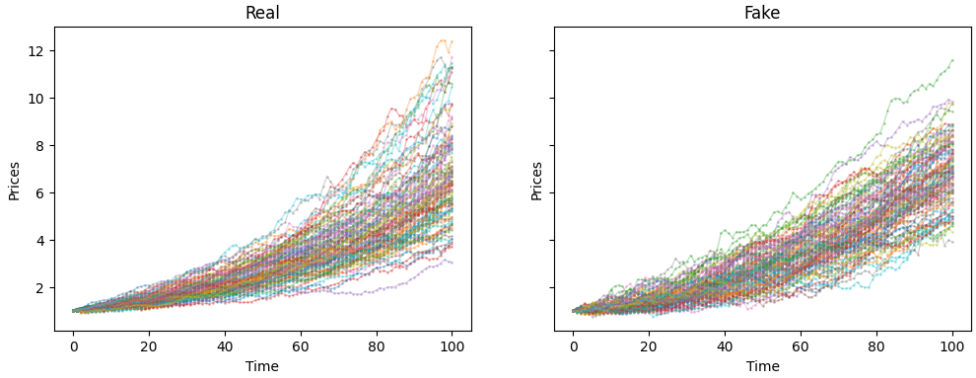


Figure 5.4: Comparison of 10 true and generated GBM paths (irregularly observed training).

Parameters Estimation for GBM

We generated 10,000 synthetic paths of a 1-dimensional GBM. To estimate the drift and volatility parameters of the GBM, we followed a methodology similar to that described in Azimzadeh (2020).

Initially, the log-returns for each path were calculated at each time step. These were then aggregated across both time steps and paths.

The log-return at time t is defined as:

$$r_t := \log\left(\frac{X_{t+\Delta t}}{X_t}\right).$$

Under the GBM model, the log-returns r_t follow a normal distribution:

$$r_t \sim \mathcal{N}\left(\left(\mu - \frac{1}{2}\sigma^2\right)\Delta t,\ \sigma^2 \Delta t\right),$$

where μ is the drift coefficient and σ is the volatility of the process.

After computing all log-returns, we flatten the matrix of returns into a single vector. Let m and s denote the sample mean and standard deviation of these aggregated log-returns:

$$m := \frac{1}{n}\sum_{i=1}^{n} r_i, \quad s := \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - m)^2},$$

where n is the total number of computed returns.

From these empirical moments, we estimate the GBM parameters as follows:

- The volatility σ is estimated as:

$$\hat{\sigma} = \frac{s}{\sqrt{\Delta t}}.$$

- The drift μ is estimated as:

$$\hat{\mu} = \frac{m}{\Delta t} + \frac{1}{2}\hat{\sigma}^2.$$

The underestimation arises due to the loss function behavior. The predictions close to zero produce small losses. Furthermore, since the observed Z_t values were often small (on the order of 10^{-1} or less), the model learned to favor near-zero outputs.

Parameter	True Data	Generated Data
Estimated Drift μ	1.985	2.067
Estimated Volatility σ	0.294	0.527

Table 5.1: Estimated parameters for true and generated GBM paths.

These findings highlight the importance of the loss function and observation scheme. In the next chapter, we investigate alternative loss functions and partially observed training strategies for the Z process.

5.2.2 Training of Z with X for Volatility Estimation

In this section, we propose a training strategy for learning the instantaneous volatility by incorporating both the X and Z processes. The estimation of the drift component remains unchanged from the previous section. The NJ-ODE model is trained on a partially observed dataset for the X process using the standard framework.

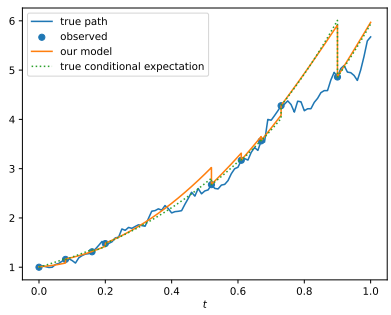
The primary modification concerns the training of the NJ-ODE model for the diffusion term. An extension of the NJ-ODE framework by Heiss et al. (2024) enables variable-sized inputs and outputs, allowing for greater flexibility in modeling. In particular, this extension permits the selection of input variables to be used during training.

We leverage this feature to train the model exclusively on the fully observed Z process, while simultaneously exposing the model to the corresponding

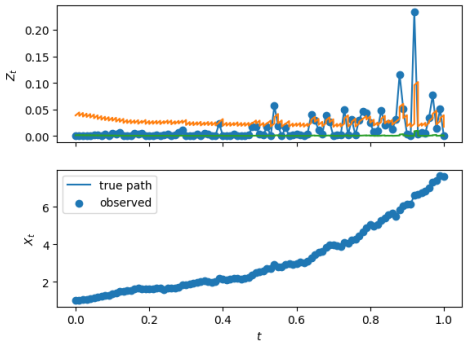
X process. This setup allows the model to learn the dependence of Z_t on the states of X, without treating X itself as a prediction target.

Furthermore, the original loss function used for input-output tasks (referred to as `"IO"` in the NJ-ODE codebase) was modified to accommodate matrix-valued outputs. This updated version, labeled `"IO_vol"`.

Figure 5.5 presents the training results. The left panel displays the NJ-ODE prediction of the partially observed X_t, while the right panel illustrates the predicted Z_t, trained using access to the X_t trajectory.



(a) NJ-ODE prediction of irregularly observed X_t



(b) NJ-ODE prediction of Z_t with X_t

Figure 5.5: Training results for NJ-ODE models on X_t (left) and Z_t with X_t (right).

After training, we used the learned drift and diffusion models to generate 1,000 synthetic GBM paths. A comparison between 10 generated trajectories and 10 true GBM trajectories is shown in Figure 5.6. While the generated paths exhibit smoother dynamics than in previous settings, they now appear to underestimate the process's true volatility.

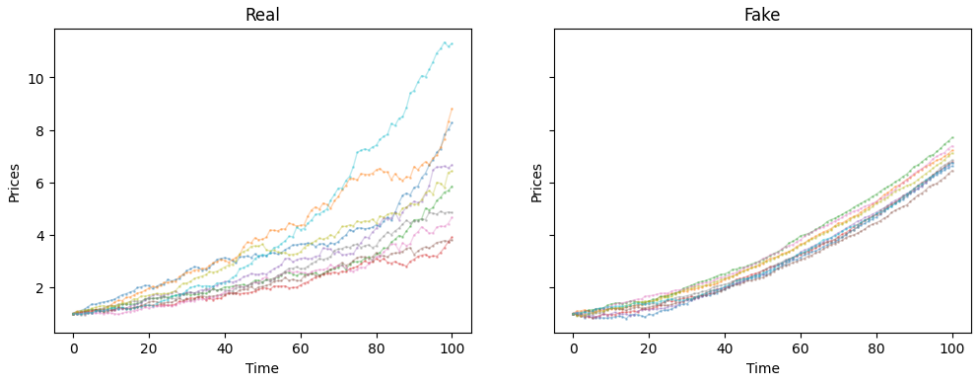


Figure 5.6: Comparison of 10 true GBM paths and 10 generated paths.

This underestimation can be traced back to the behavior of the loss function during training. Specifically, the loss remains low when the squared model output $Y_t Y_t^\top$ is close to zero or nearly constant. This effect is visible in the training curves and leads the model to favor near-zero volatility estimates. Since the training data for Z_t is derived from fully observed paths of X_t, the corresponding values of Z_t are typically small. This characteristic further amplifies the model's bias toward producing small outputs.

To quantitatively evaluate the model's performance, we computed the drift and volatility estimates from the generated data and compared them to estimates from the true GBM data. The results are shown in Table 5.2. While the drift is accurately captured, the volatility remains substantially underestimated.

Parameter	True Data	Generated Data
Estimated Drift μ	2.050	2.011
Estimated Volatility σ	0.299	0.194

Table 5.2: Estimated parameters for true and generated GBM paths (based on 1,000 samples).

These findings suggest that while conditioning on X_t lowers the volatility estimate. In the next chapter, we explore alternative loss functions and modified observation schemes to improve volatility estimation.

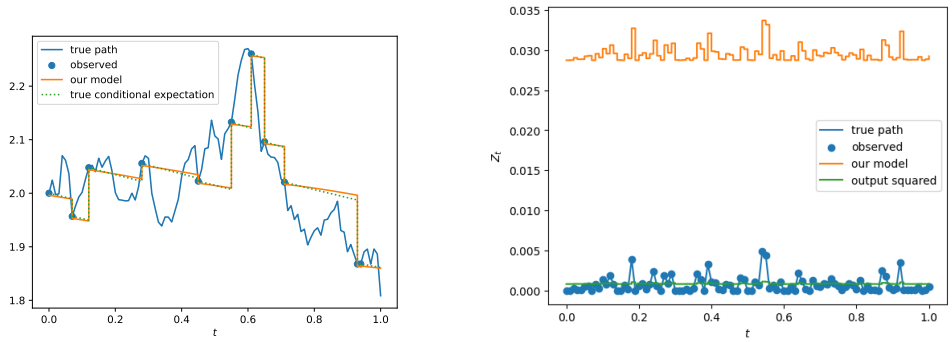
5.3 Data Generation: Ornstein-Uhlenbeck 1D

In this section, we apply the NJ-ODE framework to simulate data from a 1-dimensional OU process and evaluate the quality of the generated trajectories. The model is trained to estimate both drift and volatility components, and the resulting synthetic paths are compared against reference samples drawn from the true OU dynamics.

Model Training

We employ separate NJ-ODE models for learning the drift and volatility of the OU process. The drift component is trained directly on partially observed trajectories of the X_t process. The diffusion term is trained on fully observed paths of Z_t process.

Figure 5.7 presents the test set predictions of the trained models.



(a) Prediction of irregularly observed X_t **(b)** Prediction of Z_t

Figure 5.7: Test set performance of NJ-ODE models trained separately on X_t and Z_t.

Synthetic Data Generation

Once trained, the NJ-ODE models are used to generate 10,000 synthetic paths via the Euler–Maruyama discretisation scheme. Starting from an initial value, the drift and volatility estimates are iteratively computed at each time step, allowing for the simulation of a complete trajectory.

Figure 5.8 compares 100 generated paths with 100 reference paths from the true OU process.

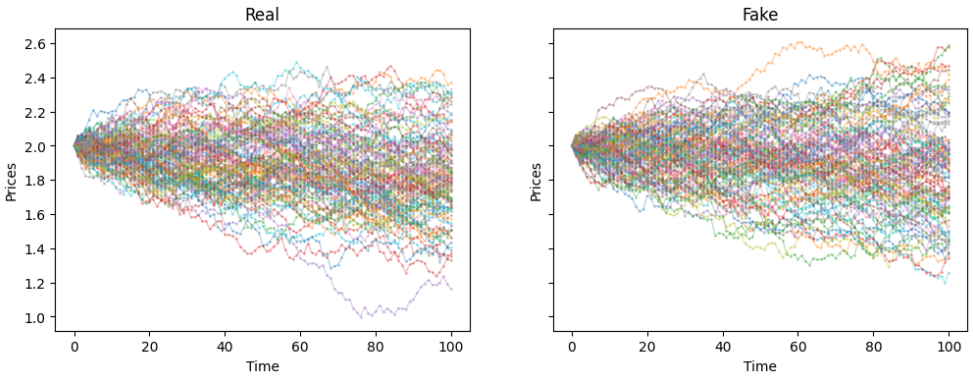


Figure 5.8: Comparison of 100 real OU paths with 100 generated paths.

Parameters Estimation for OU

We estimate the parameters of the OU process using ordinary least squares (OLS) regression. This approach relies on least squares estimation applied to the discretised OU process, as outlined in Cantaro (n.d.).

The assumed discrete-time dynamics are given by:

$$X_{t+\Delta t} = \beta X_t + \alpha + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2),$$

where β and α are regression coefficients, and ε_t represents Gaussian noise. Given simulated paths $\{X_t\}_{t=0}^{T}$, we define the vectors:

$$x = \left[X_0, X_1, \ldots, X_{T-1}\right], \quad y = \left[X_1, X_2, \ldots, X_T\right],$$

and fit the linear model $y = \beta x + \alpha + \varepsilon$ across all trajectories.

The continuous-time OU parameters are then recovered as:

- **Mean-reversion speed** $\hat{k}$:

$$\hat{k} = -\frac{1}{\Delta t}\log(\beta),$$

- **Long-term mean** $\hat{m}$:

$$\hat{m} = \frac{\alpha}{1-\beta},$$

- **Volatility** $\hat{\sigma}$:

$$\hat{\sigma} = \hat{\sigma}_\varepsilon \cdot \sqrt{\frac{2\hat{k}}{1-\beta^2}}, \quad \text{where} \quad \hat{\sigma}_\varepsilon = \sqrt{\frac{1}{N-2}\sum_{i=1}^{N}(\varepsilon_i)^2}.$$

The estimated parameters for both the real and synthetic datasets are presented in Table 5.3.

Parameter	True Data	Generated Data
Mean-Reversion Rate k	0.3189	0.8430
Long-Term Mean m	1.5392	1.3665
Volatility σ	0.3004	0.2885

Table 5.3: Estimated OU process parameters for real and generated data.

The results indicate that the NJ-ODE model accurately captures the stochastic volatility component of the OU process, as evidenced by the close match in estimated σ. However, the drift-related parameters k and m exhibit larger deviations.

Discussion

Both the visual and quantitative results confirm that the NJ-ODE framework is capable of approximating some aspects of the OU process. However, there is still substantial room for improvement. In the following chapter, we explore improvements to the volatility loss function and evaluate alternative observation schemes for Z_t. We also investigate the effect of varying the model's hidden dimension on generative performance.

Chapter 6

Additional Experiments

In this chapter, we further investigate methods to improve the generative capabilities of the NJ-ODE framework. Specifically, we explore alternative loss functions and analyze the influence of data sparsity and neural network complexity on the generation of synthetic financial data. For purposes of this part we use GBM for X process with the same parameters as in previous chapter.

6.1 Alternative Loss Functions

Previous experiments indicated difficulties in learning the dynamics of the Z process. Thus, we evaluated alternative loss functions to enhance model training.

For each observation time t_i, we define:

- At the observation point:

$$Z_{t_i} = (X_{t_i} - X_{\tau(t_i)})(X_{t_i} - X_{\tau(t_i)})^\top = \mathbf{0},$$

  since $\tau(t_i) = t_i$.

- Just before the observation:

$$Z_{t_i^-} = (X_{t_i^-} - X_{\tau(t_i^-)})(X_{t_i^-} - X_{\tau(t_i^-)})^\top = (X_{t_i} - X_{t_{i-1}})(X_{t_i} - X_{t_{i-1}})^\top,$$

  where $\tau(t_i^-) = t_{i-1}$.

All previously used datasets for the Z process were constructed using values at $Z_{t_i^-}$.

Three alternative loss functions were considered:

Loss Function 1: Enforcing Zero at Observation Times

This loss function explicitly enforces that $Z_{t_i} = 0$ at observation points:

$$\Psi(Z, YY^\top) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left(\left|M_i \odot (YY^\top)_{t_i}\right|_2 + \left|M_i \odot (Z_{t_i^-} - (YY^\top)_{t_i^-})\right|_2\right)^2\right].$$

Loss Function 2: Pre-Jump Only

This loss function removes the zero-enforcement and trains solely on values before the jump:

$$\Psi(Z, YY^\top) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left(\left|M_i \odot (Z_{t_i^-} - (YY^\top)_{t_i^-})\right|_2\right)^2\right].$$

Loss Function 3: Two-Dataset Approach

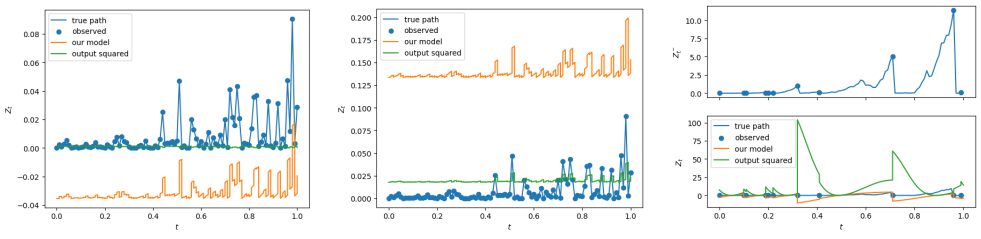
In this version, we construct two datasets. One containing Z_t and the other Z_{t^-}, and include both in the loss:

$$\Psi(Z_t, Z_{t^-}, YY^\top) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left(\left|M_i \odot (Z_{t_i} - (YY^\top)_{t_i})\right|_2 + \left|M_i \odot (Z_{t_i^-} - (YY^\top)_{t_i^-})\right|_2\right)^2\right].$$

The NJ-ODE framework was modified to allow loss functions that depend on two different input datasets simultaneously.

Training Results

The outcomes for each loss function are depicted in Figure 6.1.



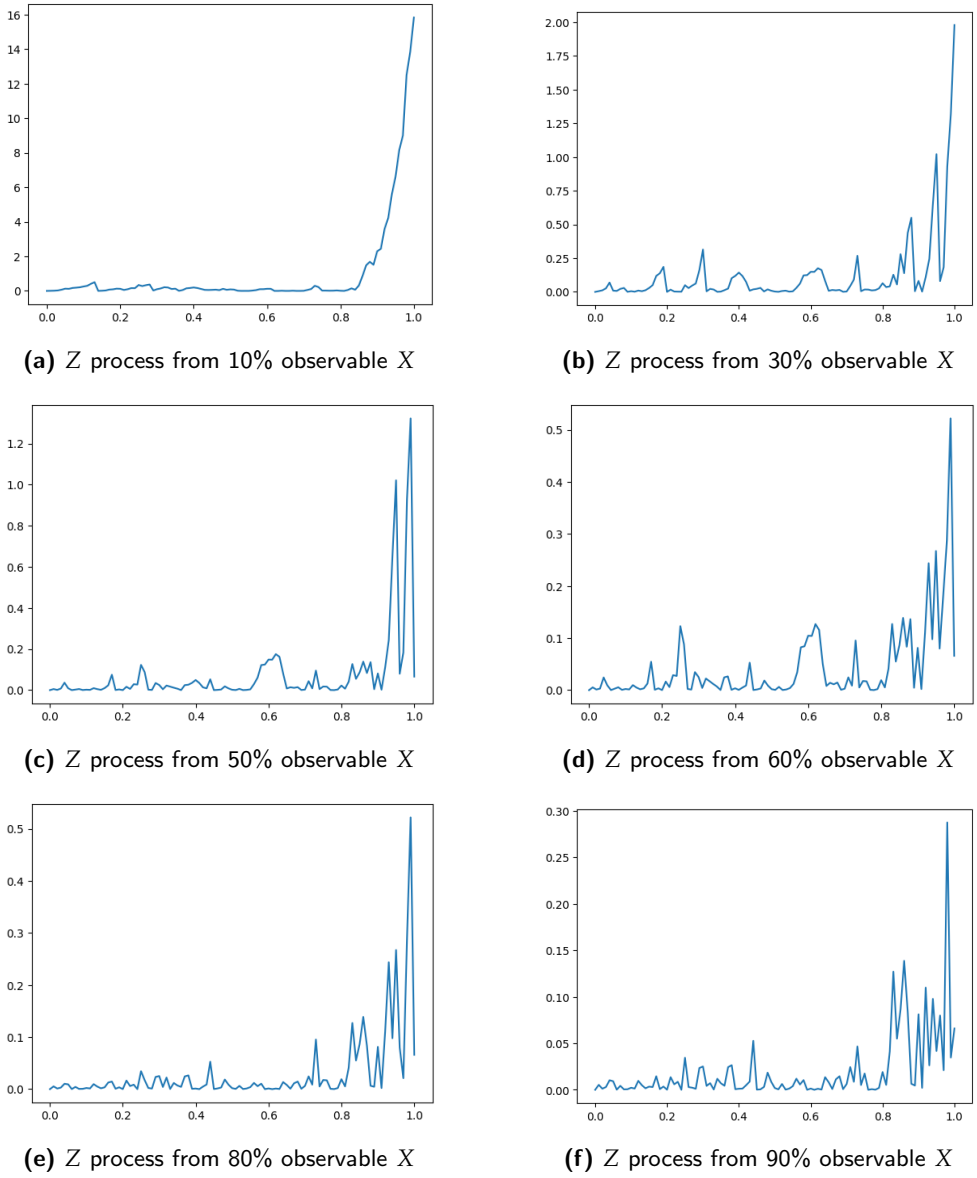
(a) Loss Function 1 **(b)** Loss Function 2 **(c)** Loss Function 3

Figure 6.1: Training results for the NJ-ODE model using three different loss functions for the Z_t process.

6.2 Impact of Observation Sparsity on the Z Process

We constructed six datasets with varying levels of observability for the X. Specifically, we simulate scenarios where X is observed at approximately 10%, 30%, 50%, 60%, 80%, and 90% of the total time grid points.

Figure 6.2 visualizes the Z processes for the different observation scenarios. Each subplot corresponds to a specific level of X observability.



(a) Z process from 10% observable X

(b) Z process from 30% observable X

(c) Z process from 50% observable X

(d) Z process from 60% observable X

(e) Z process from 80% observable X

(f) Z process from 90% observable X

Figure 6.2: Visualization of Z process under varying levels of X observability.

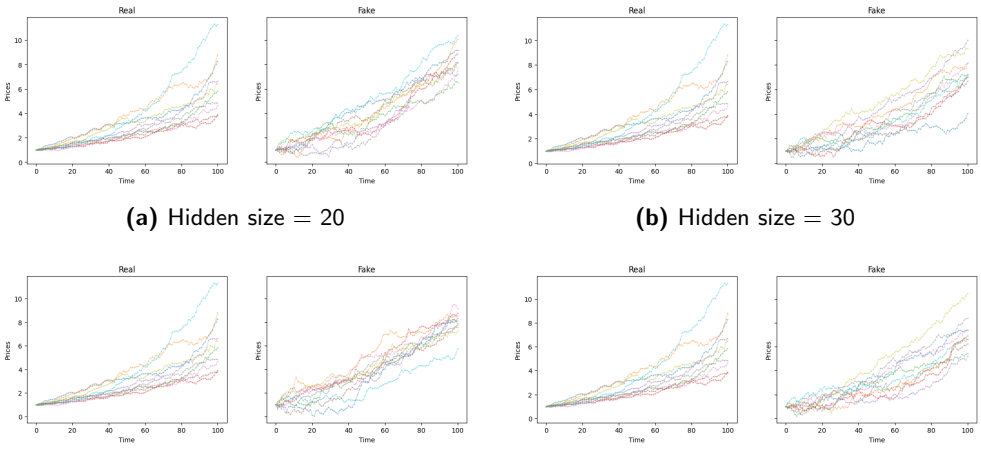
The figures reveal an increase in the magnitude of Z process with decreasing observability of the X process. For instance, when 90% of X is observable, the maximum value of Z is around 0.3. In contrast, for the 10% case, the maximum value of Z reaches approximately 16.

Empirical evidence suggests that the NJ-ODE model struggles to accurately learn the dynamics of the Z process when the X process is only sparsely observed. In particular, when the observability of X falls below 60%, the model-generated sample paths frequently become unstable or diverge. One plausible explanation is that the instantaneous volatility—estimated via the conditional expectation of the Z process—can attain excessively large values when Z itself is high. As a result, subsequent steps in the Euler–Maruyama scheme may grow rapidly, sometimes leading to NaN values.

6.3 Model Architecture Changes

The loss function number 2 was selected for further experimentation. We trained the NJ-ODE model with different hidden layer sizes: 20, 30, 40, and 50.

The generated paths for each configuration are presented in the following figure.



(a) Hidden size = 20 **(b)** Hidden size = 30

(c) Hidden size = 40 **(d)** Hidden size = 50

Figure 6.3: Generated sample paths from NJ-ODE model with varying hidden layer sizes.

In these experiments, no clear pattern in path quality was observed as the hidden size increased. This suggests that increasing model complexity alone does not necessarily enhance generative performance.

Chapter 7

Conclusion

In this thesis, we explored the potential of adapting the NJ-ODEs framework from predictive to generative modeling. Two distinct methodologies for generating synthetic data were proposed. Method I aimed to estimate the instantaneous drift and instantaneous covariance directly from the predicted conditional moments. Method II independently modeled the instantaneous drift and instantaneous diffusion terms. This approach ensured that the estimated instantaneous covariance matrices are positive semi-definite.

While Method I was conceptually straightforward, it proved impractical in practice due to the lack of guaranteed positive semi-definiteness in the estimated instantaneous covariance matrices. That led to unstable and invalid synthetic data generation. Method II, on the other hand, showed some promising results. In particular, when the model was trained on partially observable data.

Empirical assessments showed that the NJ-ODE framework faced difficulties in accurately modeling the diffusion term, which remains a significant challenge. The framework's performance was sensitive to the choice of loss function and the degree of observation sparsity. If the diffusion component can be reliably estimated, the NJ-ODE framework has the potential to generate realistic synthetic financial data.

Appendix A

Appendix

Since this thesis builds upon the NJ-ODE framework, the definitions presented in this appendix are from Florian's doctoral work (F. T. O. Krach, 2025).

A.0.1 Stochastic Process

We define the stochastic process $X := (X_t)_{t\in[0,T]} \in \mathcal{Q} \subset \mathbb{R}^{d_X}$, where $\mathcal{Q}$ is closed, convex, and bounded. The process X is càdlàg and adapted to the filtered probability space

$$(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P}).$$

Since X may exhibit jumps , we define its left-limit at time t as

$$X_{t^-} := \lim_{\varepsilon \downarrow 0} X_{t-\varepsilon}.$$

We define additional random variables on a separate probability space

$$(\tilde{\Omega}, \tilde{\mathcal{F}}, \tilde{\mathbb{F}}, \tilde{\mathbb{P}})$$

as follows:

- $n : \tilde{\Omega} \to \mathbb{N}_{\geq 0}$ is a random variable representing the random number of observations, with $\mathbb{E}_{\tilde{\mathbb{P}}}[n] < \infty$.
- $t_i : \tilde{\Omega} \to [0, T]$ for $0 \leq i \leq n$ are sorted random variables representing the random observation times.

We define τ as the time of the last observation before a given time t, expressed as:

$$\tau : [0,T] \times \tilde{\Omega} \to [0,T], \quad (t, \tilde{\omega}) \mapsto \tau(t, \tilde{\omega}) := \max\{t_i(\tilde{\omega}) \mid 0 \leq i \leq n(\tilde{\omega}), t_i(\tilde{\omega}) \leq t\}.$$

Itô diffusion

Let $X := (X_t)_{t\in[0,T]}$ be a stochastic process satisfying the following SDE:

$$dX_t = \mu(t, X_t)\, dt + \sigma(t, X_t)\, dW_t, \tag{A.1}$$

where $\{W_t\}_{t\in[0,T]}$ be a d_W-dimensional Brownian motion defined on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$, and $X_0 = x \in \mathbb{R}^{d_X}$ is starting point. The measurable functions $\mu : [0,T] \times \mathbb{R}^{d_X} \to \mathbb{R}^{d_X}$ and $\sigma : [0,T] \times \mathbb{R}^{d_X} \to \mathbb{R}^{d_X \times d_W}$ are referred to as the drift and diffusion coefficients, respectively.

The following regularity conditions are imposed:

- X is continuous and square integrable
- μ and σ are globally Lipschitz continuous their second component.
- μ is bounded and continuous in time, uniformly over space.
- The diffusion coefficient σ is càdlàg in time and square-integrable with respect to the Brownian motion W.

Information σ-algebra

We now define the information structure on the product probability space

$$(\Omega \times \tilde{\Omega}, \mathcal{F} \otimes \tilde{\mathcal{F}}, \mathbb{F} \otimes \tilde{\mathbb{F}}, \mathbb{P} \times \tilde{\mathbb{P}}),$$

where the filtration $(\mathcal{F} \otimes \tilde{\mathcal{F}})_t := \mathcal{F}_t \otimes \tilde{\mathcal{F}}_t$ for all $t \in [0,T]$ consists of the tensor-product σ-algebras.

We define the filtration of currently available information $\mathcal{A} := (\mathcal{A}_t)_{t\in[0,T]}$ by

$$\mathcal{A}_t := \sigma\left(X_{t_i,j},\, t_i \,\middle|\, t_i \le t,\; j \in \left\{\ell \in \{1,\dots,d_X\} \,\middle|\, M_{t_i,\ell} = 1\right\}\right),$$

where t_i are the random observation times, and $\sigma(\cdot)$ denotes the generated σ-algebra.

By the definition of τ, we have:

$$\mathcal{A}_t = \mathcal{A}_{\tau(t)} \quad \text{for all } t \in [0,T].$$

Observation mask

We define the observation mask $M = (M_k)_{0\le k\le K}$ as a sequence of random variables defined on $(\tilde{\Omega}, \tilde{\mathcal{F}}, \tilde{\mathbb{P}})$, where each M_k takes values in $\{0,1\}^{d_X}$ and is $\tilde{\mathcal{F}}_{t_k}$-measurable. The j-th component of M_k, denoted $M_{k,j}$, indicates whether the j-th component of the stochastic process X_{t_k} was observed:

$$M_{k,j} = \begin{cases} 1, & \text{if } X_{t_k,j} \text{ is observed,} \\ 0, & \text{otherwise.} \end{cases}$$

By abuse of notation, we also write $M_{t_k} := M_k$.

A.0.2 Conditional Expectation

We denote by $\hat{X} := (\hat{X}_t)_{0 \le t \le T}$ the conditional expectation process of X, defined with respect to the information filtration $\mathcal{A}_t$ as:

$$\hat{X}_t := \mathbb{E}_{\mathbb{P} \times \tilde{\mathbb{P}}} \left[X_t \,|\, \mathcal{A}_t \right].$$

We recall several fundamental properties of conditional expectations, which are central to the theory of stochastic processes. Let $X, X_1, X_2 \in \mathcal{F}$ be random variables such that $\mathbb{E}[|X|] < \infty$, and similarly for X_1 and X_2. Let $\mathcal{A} \subseteq \mathcal{F}$ be a sub-σ-algebra. Then the following results hold (see Durrett, 2010):

1. The conditional expectation $\mathbb{E}[X \mid \mathcal{A}]$ exists.
2. It is unique up to sets of measure zero under $\mathbb{P}$.
3. $\mathbb{E}[X \mid \mathcal{A}]$ is integrable.
4. If X is $\mathcal{A}$-measurable, then $\mathbb{E}[X \mid \mathcal{A}] = X$ (i.e., full information).
5. If X is independent of $\mathcal{A}$, then $\mathbb{E}[X \mid \mathcal{A}] = \mathbb{E}[X]$ (i.e., no information).
6. Linearity: For $a, b \in \mathbb{R}$,
$$\mathbb{E}[aX_1 + bX_2 \mid \mathcal{A}] = a\mathbb{E}[X_1 \mid \mathcal{A}] + b\mathbb{E}[X_2 \mid \mathcal{A}].$$
7. Monotonicity: If $X_1 \le X_2$ almost surely, then
$$\mathbb{E}[X_1 \mid \mathcal{A}] \le \mathbb{E}[X_2 \mid \mathcal{A}].$$
8. Jensen's Inequality: For any convex function φ such that $\mathbb{E}[|\varphi(X)|] < \infty$,
$$\varphi\left(\mathbb{E}[X \mid \mathcal{A}]\right) \le \mathbb{E}[\varphi(X) \mid \mathcal{A}].$$
9. Multiplicative Property: If $Y \in \mathcal{A}$ and $\mathbb{E}[|XY|] < \infty$, then
$$\mathbb{E}[XY \mid \mathcal{A}] = Y \cdot \mathbb{E}[X \mid \mathcal{A}].$$
10. Tower Property (Iterated Expectations): If $\mathcal{A}_1 \subseteq \mathcal{A}_2 \subseteq \mathcal{F}$, then
$$\mathbb{E}[\mathbb{E}[X \mid \mathcal{A}_2] \mid \mathcal{A}_1] = \mathbb{E}[X \mid \mathcal{A}_1], \quad \text{and} \quad \mathbb{E}[\mathbb{E}[X \mid \mathcal{A}_1] \mid \mathcal{A}_2] = \mathbb{E}[X \mid \mathcal{A}_1].$$

A.0.3 Signature

The signature transform provides a rich and universal representation of a path by encoding its sequential structure through an infinite collection of iterated integrals. This representation is particularly powerful because it allows any continuous functional on the path to be approximated by a linear function of the path's signature terms.

Definition A.1 (Signature) *Let $J \subset \mathbb{R}$ be a closed interval and let $X : J \to \mathbb{R}^d$ be a continuous path of finite variation. The* signature *of X over the interval J is defined as the infinite sequence:*

$$S(X) := \left(1, X_J^1, X_J^2, \dots\right),$$

where for each $m \geq 1$, the m-th level signature term is given by

$$X_J^m := \int_{u_1 < \cdots < u_m,\, u_1,\dots,u_m \in J} \mathrm{d}X_{u_1} \otimes \cdots \otimes \mathrm{d}X_{u_m} \in (\mathbb{R}^d)^{\otimes m}.$$

The map from a path X to its signature $S(X)$ is referred to as the *signature transform*. While the full signature is infinite-dimensional, in practical applications it is common to consider only a finite number of signature terms.

Definition A.2 (Truncated Signature) *Let $X : J \to \mathbb{R}^d$ be a continuous path of finite variation. The* truncated signature *of order $m \in \mathbb{N}$ is defined as:*

$$\pi_m(X) := \left(1, X_J^1, X_J^2, \dots, X_J^m\right),$$

i.e., the first $m+1$ levels of the full signature of X.

For a path taking values in $\mathbb{R}^d$, the total dimension of the truncated signature of order m is given by:

$$\dim \pi_m(X) = \begin{cases} m+1, & \text{if } d = 1, \\ \dfrac{d^{m+1}-1}{d-1}, & \text{if } d > 1. \end{cases}$$

A.0.4 Data Generation: Ornstein-Uhlenbeck 3D

Following the training of the NJ-ODE models on the 3-dimensional OU process, we evaluate the generative capabilities of the framework by simulating synthetic sample paths.

We first generate 1,000 sample paths from both real data and generated data. Figure A.1 illustrates the results, presenting a qualitative comparison of the real and generated trajectories along the first dimension. The generated paths demonstrate visual similarity to the real trajectories.

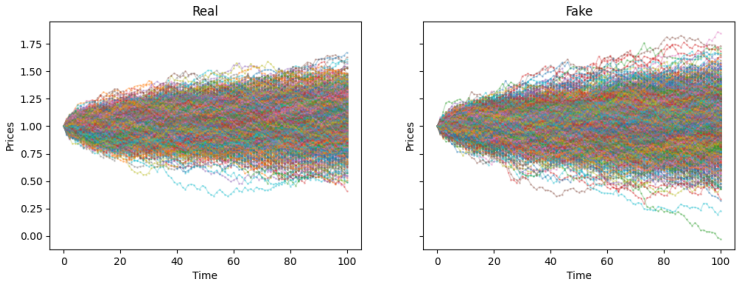


Figure A.1: Comparison of 1,000 sample paths from the real and generated 3D OU process (dimension 1).

Figure A.2 shows 10 sample paths from the real process and 10 paths generated by the NJ-ODE model, visualized side-by-side for each dimension.

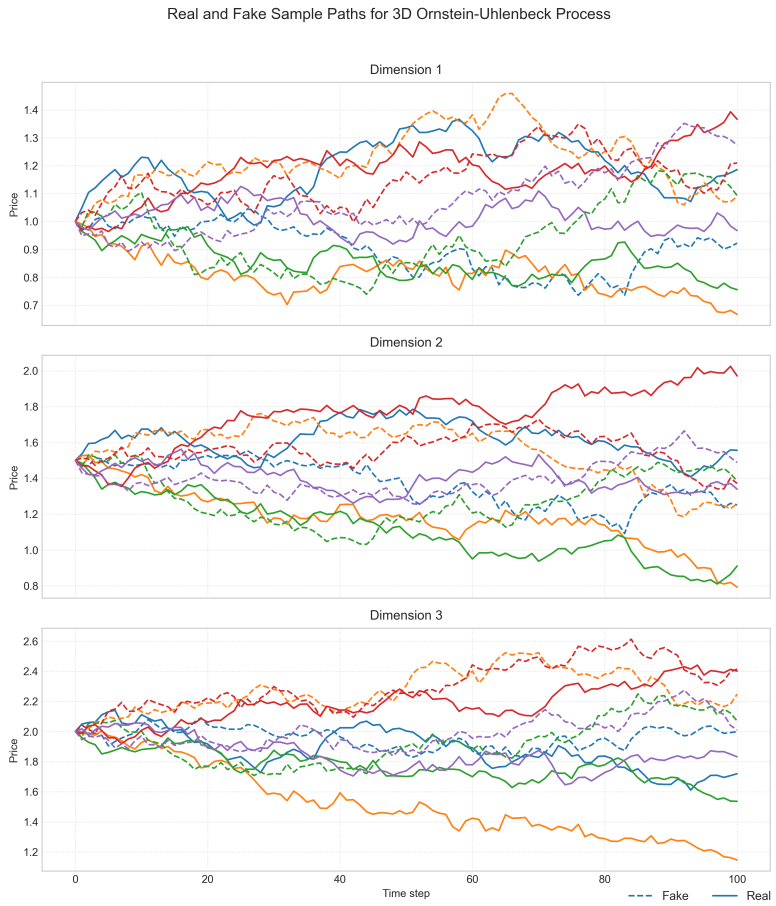


Figure A.2: Comparison of 10 real and 10 generated sample paths across all three dimensions of the 3D OU process.

Bibliography

Andersson, W. (2024). *Pd-nj-ode for predictions in convex spaces* [Master's Thesis]. ETH Zürich. https://doi.org/10.3929/ethz-b-000690272

Azimzadeh, P. (2020). Maximum likelihood estimation of geometric brownian motion parameters [Available at https://parsiad.ca/blog/2020/maximum_likelihood_estimation_of_geometric_brownian_motion_parameters/].

Bass, R. F. (2011). *Stochastic processes*. Cambridge University Press.

Björk, T. (2009). *Arbitrage theory in continuous time* (3rd ed.). Oxford University Press.

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, *81*(3), 637–654. https://doi.org/10.1086/260062

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307–327.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: Forecasting and control* (5th). John Wiley & Sons.

Bühler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep hedging. *Quantitative Finance*, *19*(8), 1271–1291. https://doi.org/10.1080/14697688.2019.1571683

Bühler, H., Horvath, B., Lyons, T., Perez Arribas, I., & Wood, B. (2020). A data-driven market simulator for small data environments. https://doi.org/10.2139/ssrn.3632431

Cantaro, C. (n.d.). Ornstein–uhlenbeck process and applications [jupyter notebook] [Accessed: 2025-04-26]. https://github.com/cantaro86/Financial-Models-Numerical-Methods/blob/master/6.1%20Ornstein-Uhlenbeck%20process%20and%20applications.ipynb

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural ordinary differential equations. *Advances in Neural Information Processing Systems (NeurIPS)*, *31*, 6571–6583.

Chevyrev, I., & Kormilitzin, A. (2016). A primer on the signature method in machine learning. https://doi.org/10.48550/arXiv.1603.03788

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. https://arxiv.org/abs/1406.1078

Durrett, R. (2010). *Probability: Theory and examples* (4th ed.). Cambridge University Press.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. https://www.deeplearningbook.org

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems (NeurIPS)*, *27*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Heiss, J., Krach, F., Schmidt, T., & Tambe-Ndonfack, F. B. (2024). Nonparametric filtering, estimation and classification using neural jump odes. https://arxiv.org/abs/2412.03271

Herrera, C., Krach, F., & Teichmann, J. (2021). Neural jump ordinary differential equations: Consistent continuous-time prediction and filtering. *International Conference on Learning Representations*. https://openreview.net/forum?id=JFKR3WqwyXR

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*, *6*(2), 327–343.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Horvath, B., Plenk, J., Vuletić, M., & Saqur, R. (2025). Generative models in finance: Market generators, a paradigm shift in financial modeling. https://doi.org/10.2139/ssrn.5284313

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization.

Kingma, D. P., & Welling, M. (2019). An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*.

Kloeden, P. E., & Platen, E. (1992). *Numerical solution of stochastic differential equations* (Vol. 23). Springer. https://doi.org/10.1007/978-3-662-12616-5

Krach, F., Nübel, M., & Teichmann, J. (2022). Optimal estimation of generic dynamics by path-dependent neural jump odes. https://arxiv.org/abs/2206.14284

Krach, F. T. O. (2025). *Neural jump ordinary differential equations* [Doctoral dissertation, ETH Zurich] [Doctoral thesis]. https://doi.org/10.3929/ethz-b-000720717

Ledoit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, *88*(2), 365–411. https://doi.org/10.1016/S0047-259X(03)00096-4

Potluru, V. K., Borrajo, D., Coletta, A., Dalmasso, N., El-Laham, Y., Fons, E., Ghassemi, M., Gopalakrishnan, S., Gosai, V., Kreačić, E., Mani, G., Obitayo, S., Paramanand, D., & Veloso, T. B. (2023). Synthetic data applications in finance. https://doi.org/10.48550/arXiv.2401.00081

Ruf, J., & Wang, W. (2020). Neural networks for option pricing and hedging: A literature review.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Shreve, S. E. (2004). *Stochastic calculus for finance ii: Continuous-time models*. Springer.

Uhlenbeck, G. E., & Ornstein, L. S. (1930). On the theory of the brownian motion. *Physical Review*, *36*(5), 823–841. https://doi.org/10.1103/PhysRev.36.823

Vasicek, O. (1977). An equilibrium characterization of the term structure. *Journal of Financial Economics*, *5*(2), 177–188. https://doi.org/10.1016/0304-405X(77)90016-2

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, *30*.