Learning SDE Solutions with Neural Stochastic Flows

Naoki Kiyohara[1,2] **Edward Johns**[1] **Yingzhen Li**[1]
[1]Imperial College London [2]Canon Inc.
{n.kiyohara23, e.johns, yingzhen.li}@imperial.ac.uk

Abstract

We introduce *neural stochastic flows*, a novel framework for learning solutions of stochastic differential equations (SDEs). It addresses the challenge of simulating neural SDEs by directly learning the weak solution of the underlying SDE governing the data generation process. Key to our approach is the introduction of a flow-based, time-dependent neural network to directly compute the conditional probability densities of its associated SDE at any time interval. Unlike neural SDEs, neural stochastic flows avoid time-consuming simulations of SDE solvers and enable single-step sampling from the learnt stochastic process. Our work contributes to stochastic process modelling, offering a flexible method that has the potential to improve computational efficiency in various applications.

1 Introduction

Neural stochastic differential equations (SDEs) [5–7, 9–11, 19] provide a flexible and principled approach for modelling complex systems with inherent randomness [12], but their application is often hindered by computational challenges, particularly fine discretisation [6, 7, 10, 11]. Not surprisingly, the challenge of time-consuming simulations using solvers also applies to neural ordinary differential equations (ODEs) [3]. In this regard, neural flows [2] are presented as a solution to this challenge by directly learning the solutions of the underlying ODE governing the data generation process. The key idea is to use a flow-based neural network to compute the solution of the initial value problem directly.

Inspired by recent advances in neural flows [2], we present neural stochastic flows, extending the idea of direct modelling of the solution space to neural SDEs. However, unlike the ODE case, SDEs have both *strong* and *weak* solutions. A strong solution for an SDE provides direct computation of the sample path given a specific realisation of the noise process, while a weak solution models the probability distribution of the stochastic process rather than individual trajectories. Since many applications require only weak solutions, we focus on modelling the weak solutions of Itô SDEs and propose a flow-based neural network architecture to satisfy key properties from stochastic flows of diffeomorphisms whilst providing access to conditional probability densities.

This paper provides the theoretical foundations for neural stochastic flows, reformulating stochastic flow conditions in terms of probability distributions. Our empirical results with known stochastic processes demonstrate strong temporal generalisation capabilities. Our work contributes to neural SDEs, offering a flexible and efficient method for modelling stochastic processes. Neural stochastic flows bridge stochastic calculus and neural networks, offering a potential for faster SDE inference.

2 Background

Neural ODEs / Neural Flows. ODEs and their integral solutions, known as flows, have been a significant focus in machine learning research. Neural ODEs [3] define the vector field using neural networks, offering high interpretability but requiring discretisation for both training and inference. This approach has been extended to latent ODEs [15], which model the dynamics of latent representations. In contrast, neural flows [2] directly learn the flow, representing the ODE's solution space by designing the architecture and/or loss function to fulfil the flow properties. This approach allows for solutions in a single step, avoiding iterative simulation.

Probability Flow ODEs. Recent research has explored the relationship between diffusion models and SDEs, proposing methods to transform the SDE of a diffusion model into a *probability flow ODE* [18]. This approach enables single-step image generation by training a neural network to predict the ODE's endpoint [17]. While this shares similarities with our work in terms of single-step sampling from SDEs, our approach aims to generalise this concept to arbitrary SDEs and any time point.

Stochastic Flows of Diffeomorphisms. An appealing prospect is to apply neural flows to Itô SDEs for direct learning of the solution space. Consider an Itô SDE:

$$d\boldsymbol{x}_t = \boldsymbol{\mu}(\boldsymbol{x}_t, t)dt + \boldsymbol{\sigma}(\boldsymbol{x}_t, t)d\boldsymbol{W}_t \tag{1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^n$ is the state vector, $\boldsymbol{\mu} : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}^n$ is the drift term, $\boldsymbol{\sigma} : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}^{n\times m}$ is the diffusion term, and $\boldsymbol{W}_t$ is an m-dimensional Wiener process. For an Itô SDE with smooth drift and diffusion terms, the solution space forms a *stochastic flow of diffeomorphisms* [8]. This is defined by a measurable map $\boldsymbol{\varphi}_{s,t} : \mathbb{R}^n \times \Omega \to \mathbb{R}^n$ where $0 < s < t < +\infty$ and Ω is the sample space in terms of probability measure, satisfying:

1. **Diffeomorphism**: For any $s < t$ and $\boldsymbol{\omega} \in \Omega$, the map $\boldsymbol{x} \mapsto \boldsymbol{\varphi}_{s,t}(\boldsymbol{x}, \boldsymbol{\omega})$ is almost surely a diffeomorphism.
2. **Independence**: Maps $\boldsymbol{\varphi}_{t_1,t_2}, \ldots, \boldsymbol{\varphi}_{t_{n-1},t_n}$ for any sequence $0 \le t_1 \le t_2 \le \ldots \le t_n$ are independent.
3. **Flow property**: For any $0 \le t_1 \le t_2 \le t_3$, any point $\boldsymbol{x} \in \mathbb{R}^n$, and any $\boldsymbol{\omega} \in \Omega$:
$$\boldsymbol{\varphi}_{t_1,t_3}(\boldsymbol{x}, \boldsymbol{\omega}) = \boldsymbol{\varphi}_{t_2,t_3}\left(\boldsymbol{\varphi}_{t_1,t_2}(\boldsymbol{x}, \boldsymbol{\omega}), \boldsymbol{\omega}\right).$$
4. **Identity Property**: For any $t \ge 0$, any point $\boldsymbol{x} \in \mathbb{R}^n$, and any $\boldsymbol{\omega} \in \Omega$: $\boldsymbol{\varphi}_{t,t}(\boldsymbol{x}, \boldsymbol{\omega}) = \boldsymbol{x}$.

Furthermore, when the SDE is autonomous, where the drift and diffusion terms are independent of time, an additional property holds:

5. **Stationarity**: For any $s \le t$, $r \ge 0$, and $\boldsymbol{\omega} \in \Omega$, the maps $\boldsymbol{\varphi}_{s,t}(\boldsymbol{x}, \boldsymbol{\omega})$ and $\boldsymbol{\varphi}_{s+r,t+r}(\boldsymbol{x}, \boldsymbol{\omega})$ have the same probability distribution.

By constructing neural networks that inherently satisfy these properties, we develop efficient models for stochastic systems that overcome discretisation limitations. Additionally, these models provide direct access to probability distribution functions of SDE-governed processes.

3 Methods

In this section, we develop an architecture that fulfils the criteria for stochastic flow of diffeomorphisms. We focus on modelling the probability distribution of the stochastic process evolving from a deterministic initial condition over time according to the SDE.

For a given stochastic flow $\boldsymbol{\varphi}$, the conditional probability distribution representing a weak solution of an SDE is defined as:

$$p\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}; t_i, t_j - t_i\right) := \int_\Omega \delta\left(\boldsymbol{x}_{t_j} - \boldsymbol{\varphi}_{t_i,t_j}\left(\boldsymbol{x}_{t_i}, \boldsymbol{\omega}\right)\right) p(\boldsymbol{\omega}) d\boldsymbol{\omega} \tag{2}$$

where $\delta(\cdot)$ denotes the Dirac delta function. To approximate this distribution, we employ a neural stochastic flow—a parametric model $\boldsymbol{f}_{\boldsymbol{\theta}}$ that transforms Gaussian samples into the desired distribution via diffeomorphic mappings, akin to normalising flows [14]. Our model is defined as:

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}; t_i, t_j - t_i\right) = \int \delta(\boldsymbol{x}_{t_j} - \boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_i}, t_i, t_j - t_i, \boldsymbol{\varepsilon}\right)) \mathcal{N}\left(\boldsymbol{\varepsilon} | \boldsymbol{0}, \boldsymbol{I}\right) d\boldsymbol{\varepsilon} \tag{3}$$

where $\boldsymbol{\varepsilon}$, of the same dimension as $\boldsymbol{x}_t$, is sampled from a standard Gaussian distribution. The function $\boldsymbol{f}_{\boldsymbol{\theta}}$ serves as our parametric model to approximate the SDE's weak solution.

Conditions. Given this framework, we reformulate the conditions as follows:

1. **Diffeomorphism**: When the probability distribution is a delta function limit, i.e., when $\boldsymbol{f_\theta}$ is independent of $\boldsymbol{\varepsilon}$, the function $\boldsymbol{f_\theta}$ must be diffeomorphic with respect to $\boldsymbol{x}_{t_i}$.[3]
2. **Independence**: For any sequence $0 \le t_1 \le t_2 \le \ldots \le t_n$, the conditional probabilities $p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_2} | \boldsymbol{x}_{t_1}; t_1, t_2 - t_1\right), \ldots, p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_n} \middle| \boldsymbol{x}_{t_{n-1}}; t_{n-1}, t_n - t_{n-1}\right)$ must be independent.
3. **Flow property**: For any $0 \le t_i \le t_j \le t_k$, the joint distribution must satisfy the Chapman-Kolmogorov equation:

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_i}; t_i, t_k - t_i\right) = \int p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_j}; t_j, t_k - t_j\right) p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}; t_i, t_j - t_i\right) d\boldsymbol{x}_{t_j}. \quad (4)$$

4. **Identity property**: At the same time t, the distribution must reduce to a delta function, indicating no change: $p_{\boldsymbol{\theta}}\left(\boldsymbol{x} | \boldsymbol{x}_{t_i}; t_i, 0\right) = \delta\left(\boldsymbol{x} - \boldsymbol{x}_{t_i}\right)$.

For autonomous SDEs, we restate the stationarity condition in terms of our parametric model:

5. **Stationarity**: The conditional distributions must exhibit stationarity such that for any $t_i, t_j, r \ge 0$, $p_{\boldsymbol{\theta}}\left(\cdot \mid \boldsymbol{x}; t_i, t_j - t_i\right) = p_{\boldsymbol{\theta}}\left(\cdot \mid \boldsymbol{x}; t_i + r, t_j - t_i\right)$.

Architecture. To satisfy these conditions (except for the flow property, which is addressed separately) and to ensure computational tractability of the probability density function, similar to normalising flows, we propose the following model architecture (see Appendix A.1 for detailed architecture):

$$\boldsymbol{f_\theta}\left(\boldsymbol{x}_{t_i}, t_i, t_j - t_i, \boldsymbol{\varepsilon}\right) = \boldsymbol{G_\theta}\left(\boldsymbol{F_\theta}\left(\boldsymbol{x}_{t_i}, t_j - t_i; t_i\right) + \boldsymbol{s_\theta}(\boldsymbol{x}_{t_i}, t_i, t_j - t_i) \odot \boldsymbol{\varepsilon}, t_j - t_i; t_i\right). \quad (5)$$

Here, $\boldsymbol{F_\theta}$ and $\boldsymbol{G_\theta}$ are "conditional" coupling flows, extending coupling flows—a type of neural flow using affine coupling [4] for analytical invertibility—to model time-dependent drift and diffusion terms by incorporating t_i. These flows satisfy $\boldsymbol{F_\theta}(\boldsymbol{x}, 0; \cdot) = \boldsymbol{x}$ and $\boldsymbol{G_\theta}(\boldsymbol{x}, 0; \cdot) = \boldsymbol{x}$. Additionally, $\boldsymbol{s_\theta}(\boldsymbol{x}_{t_i}, t_i, t_j - t_i)$ is any smooth trainable function that meets the condition $\boldsymbol{s_\theta}(\boldsymbol{x}_{t_i}, t_i, 0) = \boldsymbol{0}$ for any $\boldsymbol{x}_{t_i}$ and $t_i > 0$. This architecture satisfies conditions 1, 2, and 4 by construction (see Appendix B for details). It allows for complex state-dependent noise structures, which are effective for capturing intricate stochastic dynamics.

For autonomous SDEs, we can simplify this architecture by removing the t_i condition from $\boldsymbol{F_\theta}$, $\boldsymbol{G_\theta}$, and $\boldsymbol{s_\theta}$, which then also satisfies condition 5.

To satisfy the flow property, we introduce a soft constraint approach. We add a discrepancy term $\mathcal{L}_{\text{flow}}$ to the loss function, which measures the difference between the distributions in Equation (4). This term is designed to bound the bidirectional KL divergence. We use an auxiliary variational distribution $q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)$ to estimate this discrepancy (see Appendix C for details).

The total loss function combines the negative log-likelihood of the data with the flow property constraint:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = - \mathop{\mathbb{E}}_{(\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_j}, t_i, t_j) \sim \mathcal{D}} \left[\log p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}; t_i, t_j - t_i\right)\right] + \mathcal{L}_{\text{flow}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \quad (6)$$

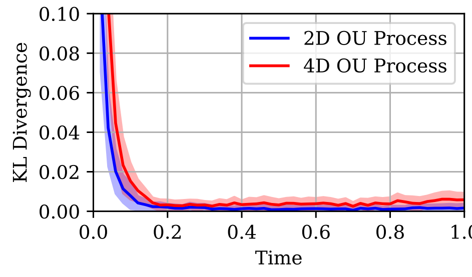
where $\mathcal{D}$ represents the dataset. The model parameters $\boldsymbol{\theta}$ and auxiliary model parameters $\boldsymbol{\phi}$ are optimised using gradient-based methods to minimise this loss function (see Appendix D for details).

4 Experiments

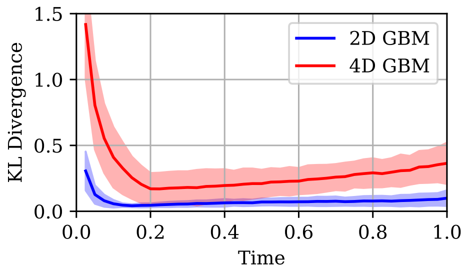
We evaluate our neural stochastic flows approach on two well-known Itô SDEs—the Ornstein-Uhlenbeck (OU) process and Geometric Brownian Motion (GBM)—in both 2D and 4D settings. Each dimension is treated independently to assess the model's performance across varying complexities. The OU process is given by $dx_t = \theta(\mu - x_t)dt + \sigma dW_t$, and GBM by $dx_t = \mu x_t dt + \sigma x_t dW_t$, where W_t is a Wiener process. We randomly sample parameters and initial values (see Appendix E for specifics) to ensure robustness across various configurations.

Our training dataset consists of triplets $(\boldsymbol{x}_0, \boldsymbol{x}_t, t)$, with t specifically set to 0.2, 0.4, and 0.8. However, due to the incorporation of the flow loss in our training procedure, our model demonstrates the ability to predict probability distributions at arbitrary time points, not just those seen during training.

[3]When $\boldsymbol{\varphi}_{t_i, t_j}$ depends on $\boldsymbol{\omega}$, transitioning to transformed probability densities as in Equation (2), the diffeomorphism condition is relaxed.



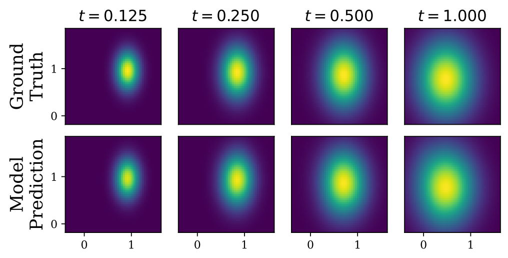
(a) OU Process



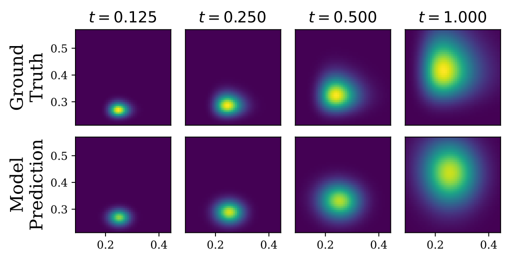
(b) GBM

Figure 1: KL divergence between true and predicted distributions over time for randomised conditions. Solid lines show means; shaded areas indicate ± 1 standard deviation (See Appendix E.5 for details).

Figure 1 illustrates model performance across stochastic processes. For OU processes, low KL divergence (< 0.01) is achieved except at small t, demonstrating robust interpolation and extrapolation. GBM exhibits higher divergence, indicating lower accuracy for non-Gaussian distributions and complex evolutions. Increasing KL divergence with system dimensionality suggests reduced accuracy for higher-dimensional distributions. While the model matches ground truth at $t = 0$, higher KL divergence for small t likely results from slower entropy decrease (see Appendix F.1).



(a) OU Process



(b) GBM

Figure 2: Evolution of probability densities for 2D system. The upper panels show contour plots of probability densities for ground truth at $t = 0.125, 0.25, 0.5, 1$. Lower panels display the corresponding model-predicted probability densities at the same time points.

Figure 2 showcases the model's proficiency in capturing the evolving probability distributions for both the OU process and GBM in 2D. The contours accurately reflect the changing means and variances over time, including at untrained time points. However, for the GBM at $t = 1$, which represents extrapolation ($t > 0.8$), the distribution's centre and shape exhibit slight deviations, likely due to propagated errors from interpolated regions.

Our results demonstrate the model's efficacy in capturing a range of stochastic processes across varying dimensions and time scales. The approach models both OU and GBM processes, providing accurate probability density estimates at arbitrary time points, including those unseen during training. This capability stems from the flow loss, which ensures the learnt stochastic flow adheres to the Chapman-Kolmogorov equation. Moreover, the model's discretisation-free nature has the potential to enable rapid predictions, offering a computationally efficient method for SDE distribution forecasting.

5 Conclusion and Future Work

In this paper, we have presented a method that enables one-step sampling from learnable SDEs, providing a potential approach to streamline certain SDE-based models. This technique may open new possibilities for applications requiring efficient SDE processing. Future work will explore the potential of this technique across different domains, conducting comprehensive comparisons with other neural SDE approaches, and investigating its implications for the broader field of stochastic processes and machine learning.

References

[1] Felix V. Agakov and David Barber. An auxiliary variational method. In *Neural Information Processing, 11th International Conference, ICONIP 2004, Calcutta, India, November 22-25, 2004, Proceedings*, volume 3316 of *Lecture Notes in Computer Science*, pages 561–566. Springer, 2004.

[2] Marin Bilos, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. Neural flows: Efficient alternative to neural ODEs. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 21325–21337, 2021.

[3] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6572–6583, 2018.

[4] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[5] Junteng Jia and Austin R. Benson. Neural jump stochastic differential equations. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9843–9854, 2019.

[6] Patrick Kidger, James Foster, Xuechen Li, and Terry J. Lyons. Efficient and accurate gradients for neural SDEs. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 18747–18761, 2021.

[7] Patrick Kidger, James Foster, Xuechen Li, and Terry J. Lyons. Neural SDEs as infinite-dimensional GANs. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5453–5463. PMLR, 2021.

[8] H. Kunita. *Stochastic Flows and Stochastic Differential Equations*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1990.

[9] Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 3870–3882. PMLR, 2020.

[10] Xuanqing Liu, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. Neural SDE: stabilizing neural ODE networks with stochastic noise. *CoRR*, abs/1906.02355, 2019.

[11] YongKyung Oh, Dongyoung Lim, and Sungil Kim. Stable neural stochastic differential equations in analyzing irregular time series data. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[12] Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications (Universitext)*. Springer, 6th edition, 2003.

[13] Rajesh Ranganath, Dustin Tran, and David M. Blei. Hierarchical variational models. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 324–333. JMLR.org, 2016.

[14] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1530–1538. JMLR.org, 2015.

[15] Yulia Rubanova, Tian Qi Chen, and David Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5321–5331, 2019.

[16] Tim Salimans, Diederik P. Kingma, and Max Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1218–1226. JMLR.org, 2015.

[17] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 32211–32252. PMLR, 2023.

[18] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[19] Belinda Tzen and Maxim Raginsky. Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. *CoRR*, abs/1905.09883, 2019.

A Detailed Model Architecture and Sampling Procedure

In this appendix, we provide a comprehensive description of our model's architecture, focusing on the conditional coupling flows utilised, and we detail the procedure for sampling from the model during inference.

A.1 Model Architecture

Our model aims to approximate the conditional probability distribution $p\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}; t_i, t_j - t_i\right)$ of the solution of an SDE at time t_j, given the state at time t_i. To achieve this, as given in Equation (5), we employ a neural stochastic flow defined as follows:

$$\boldsymbol{f_\theta}\left(\boldsymbol{x}_{t_i}, t_i, t_j - t_i, \boldsymbol{\varepsilon}\right) = \boldsymbol{G_\theta}\left(\boldsymbol{F_\theta}\left(\boldsymbol{x}_{t_i}, t_j - t_i; t_i\right) + \boldsymbol{s_\theta}\left(\boldsymbol{x}_{t_i}, t_i, t_j - t_i\right) \odot \boldsymbol{\varepsilon}, t_j - t_i; t_i\right),$$

where $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, $\odot$ denotes element-wise multiplication, and $\boldsymbol{\theta}$ represents the model parameters.

The architecture comprises three main components:

- **Nonlinear Transformations $\boldsymbol{F_\theta}$ and $\boldsymbol{G_\theta}$**: These are neural flows conditioned on the initial time t_i. Specifically, we choose to implement $\boldsymbol{F_\theta}$ and $\boldsymbol{G_\theta}$ using *conditional coupling flows* (see Section A.1.1 for details). The use of coupling flows allows for efficient maximum likelihood estimation during training, as they possess analytical invertibility. A key property of neural flows is that when $t_j - t_i = 0$, the output is identical to the input $\boldsymbol{x}_{t_i}$ [2]. This conditioning enables the transformations to adapt based on the initial time and time difference, effectively capturing the temporal dynamics of the stochastic process.

  The function $\boldsymbol{F_\theta}$ serves as a nonlinear transformation from the input space $\boldsymbol{x}_{t_i}$ to an intermediate space where noise is added, while $\boldsymbol{G_\theta}$ transforms this noisy intermediate representation to the output space $\boldsymbol{x}_{t_j}$. When there is no noise, i.e., when $\boldsymbol{s_\theta}$ is zero, the composition $\boldsymbol{G_\theta} \circ \boldsymbol{F_\theta}$ becomes a deterministic mapping, which is a diffeomorphism due to the invertibility property of the chosen coupling flows.
- **Noise Scaling Function $\boldsymbol{s_\theta}$**: This function scales the noise term $\boldsymbol{\varepsilon}$ based on the time difference $t_j - t_i$ and potentially the input state $\boldsymbol{x}_{t_i}$ and the initial time t_i. It is essential that $\boldsymbol{s_\theta}(\boldsymbol{x}_{t_i}, t_i, 0) = \boldsymbol{0}$ for any $\boldsymbol{x}_{t_i}$ and t_i to ensure no noise is added when there is no time evolution. While $\boldsymbol{s_\theta}$ does not necessarily have to depend on the input state or the initial time—any function satisfying $\boldsymbol{s_\theta}(\boldsymbol{x}_{t_i}, t_i, 0) = \boldsymbol{0}$ suffices (e.g., $\boldsymbol{s_\theta} = \sqrt{t_j - t_i}$)—we find empirically that including dependencies on $\boldsymbol{x}_{t_i}$ and t_i often leads to improved performance.

A.1.1 Conditional Coupling Flows

Coupling flows, a type of neural flow, utilise affine coupling techniques known for their analytical invertibility [2]. These flows operate by dividing input variables into two subsets and applying an invertible transformation to one subset, conditioned on the other [4]. We extend this concept by introducing additional conditioning to accommodate time-dependent drift and diffusion terms.

Consider an input vector $\boldsymbol{x}$ partitioned into two disjoint subsets, $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$. The coupling flow layer transforms $\boldsymbol{x}_A$ using scale and shift operations conditioned on both $\boldsymbol{x}_B$ and temporal variables:

$$\boldsymbol{y}_A = \boldsymbol{x}_A \odot \exp\left(\boldsymbol{u}(\boldsymbol{x}_B; t_i)\, \phi_u(t_j - t_i)\right) + \boldsymbol{v}(\boldsymbol{x}_B; t_i)\, \phi_v(t_j - t_i), \tag{7}$$

$$\boldsymbol{y}_B = \boldsymbol{x}_B, \tag{8}$$

where:

- $\boldsymbol{u}$ and $\boldsymbol{v}$ are arbitrary functions, typically implemented as neural networks, that output scale and shift parameters, respectively.
- ϕ_u and ϕ_v are time embedding functions satisfying $\phi_u(0) = \phi_v(0) = 0$.
- t_i represents the initial time, and t_j represents a subsequent time point.

The use of the exponential function in the scaling operation ensures the invertibility of the transformation. Since $\boldsymbol{x}_B$ remains unchanged, the Jacobian determinant of the transformation is straightforward to compute.

A key feature of this formulation is the behaviour of the time embedding functions ϕ_u and ϕ_v. Their property of $\phi_u(0) = \phi_v(0) = 0$ ensures that when $t_j = t_i$, the transformation reduces to the identity transformation:

$$\boldsymbol{y}_A = \boldsymbol{x}_A \odot \exp\left(\boldsymbol{u}(\boldsymbol{x}_B; t_i) \cdot 0\right) + \boldsymbol{v}(\boldsymbol{x}_B; t_i) \cdot 0 = \boldsymbol{x}_A \tag{9}$$

This characteristic guarantees that the flow preserves the original input without transformation when there is no time difference.

To construct highly expressive invertible transformations, we stack multiple coupling flow layers, alternating the subsets A and B between layers. The conditioning on t_i allows the transformation to adapt based on the initial time, capturing temporal dependencies in the data. The time embeddings $\phi_u(t_j - t_i)$ and $\phi_v(t_j - t_i)$ enable the transformation to vary smoothly with the time difference $t_j - t_i$, whilst ensuring no transformation occurs when the time difference is zero.

In our model, conditional coupling flows serve as one instance for the functions $\boldsymbol{F_\theta}$ and $\boldsymbol{G_\theta}$. However, other invertible transformations satisfying the necessary properties can also be employed, depending on the specific requirements and characteristics of the data at hand.

It is worth noting that while this formulation of conditional coupling flows allows for handling general SDEs with time-dependent drift and diffusion terms (time-inhomogeneous SDEs), the experiments presented in the main body of this paper focus on a simpler case. Specifically, we consider autonomous SDEs, where the drift and diffusion terms do not explicitly depend on time. As a result, the experiments do not utilise the t_i conditioning described above. The more general formulation presented here illustrates the full capabilities of our approach and its potential for future extensions to time-inhomogeneous SDEs, while the main paper demonstrates its effectiveness on the common and important class of time-homogeneous SDEs.

A.2 Sampling from the Model

To sample from the conditional distribution, we employ our model's architecture in a straightforward manner. The process involves applying the transformation $\boldsymbol{F_\theta}$ to the initial state $\boldsymbol{x}_{t_i}$, adding scaled Gaussian noise, and then applying the transformation $\boldsymbol{G_\theta}$. The noise scaling factor is determined by $\boldsymbol{s_\theta}$, which depends on the initial state and time parameters. This procedure generates samples that reflect the learnt conditional distribution of the stochastic process.

A.3 Computing the Conditional Densities

The computation of conditional densities is crucial for our model, as it enables both maximising log-likelihood and minimising flow loss combined with a reparametrisation trick during training. To evaluate the conditional density $p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}; t_i, t_j - t_i\right)$, we utilise the change of variables formula:

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}; t_i, t_j - t_i\right) = p_{\boldsymbol{\varepsilon}}\left(\boldsymbol{\varepsilon}\right) \left|\det\left(\frac{\partial \boldsymbol{f_\theta}}{\partial \boldsymbol{\varepsilon}}\right)\right|^{-1}. \tag{10}$$

Due to the structure of the flows used, the determinant of the Jacobian $\partial \boldsymbol{f_\theta} / \partial \boldsymbol{\varepsilon}$ can be efficiently computed.

The Jacobian determinant is given by:

$$\left|\det\left(\frac{\partial \boldsymbol{f_\theta}}{\partial \boldsymbol{\varepsilon}}\right)\right| = \left|\det\left(\frac{\partial \boldsymbol{G_\theta}}{\partial \boldsymbol{u}}\right)\right| \times \left|\det\left(\mathrm{diag}(\boldsymbol{s})\right)\right|, \tag{11}$$

where $\boldsymbol{u} = \boldsymbol{z} + \boldsymbol{s} \odot \boldsymbol{\varepsilon}$. Since $\boldsymbol{s}$ is a diagonal scaling, $\det(\mathrm{diag}(\boldsymbol{s})) = \prod_i s_i$, and the determinant of $\partial \boldsymbol{G_\theta} / \partial \boldsymbol{u}$ is computed depending on the specific invertible transformation used for $\boldsymbol{G_\theta}$.

B Theoretical Foundations of the Proposed Model

This appendix offers a rigorous theoretical exposition for our proposed model, demonstrating adherence to the essential mathematical properties outlined in Section 3. Key properties include diffeomorphism, independence, identity property, and stationarity for autonomous SDEs.

Diffeomorphism. Consider the mapping $\boldsymbol{f}_{\boldsymbol{\theta}}$ defined for fixed $0 < t_i \leq t_j$ and any $\boldsymbol{\varepsilon}$, as given in Equation (5):

$$\boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_i}, t_i, t_j - t_i, \boldsymbol{\varepsilon}\right) = \boldsymbol{G}_{\boldsymbol{\theta}}\left(\boldsymbol{F}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_i}, t_j - t_i; t_i\right) + \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_i}, t_i, t_j - t_i) \odot \boldsymbol{\varepsilon}, t_j - t_i; t_i\right).$$

Here, $\boldsymbol{F}_{\boldsymbol{\theta}}$ and $\boldsymbol{G}_{\boldsymbol{\theta}}$ are diffeomorphisms with respect to their first variables. In the case where $\boldsymbol{f}_{\boldsymbol{\theta}}$ is independent of $\boldsymbol{\varepsilon}$, we have $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_i}, t_i, t_j - t_i) = \mathbf{0}$. Consequently, $\boldsymbol{f}_{\boldsymbol{\theta}}$ reduces to a composition of diffeomorphisms:

$$\boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_i}, t_i, t_j - t_i, \boldsymbol{\varepsilon}\right) = \boldsymbol{G}_{\boldsymbol{\theta}}\left(\boldsymbol{F}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_i}, t_j - t_i; t_i\right), t_j - t_i; t_i\right). \tag{12}$$

As the composition of diffeomorphisms is itself a diffeomorphism, $\boldsymbol{f}_{\boldsymbol{\theta}}$ is a diffeomorphism with respect to $\boldsymbol{x}_{t_i}$.

Independence. For any sequence of times $0 \leq t_1 \leq t_2 \leq \ldots \leq t_n$, the model ensures that the conditional probabilities $p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_2} | \boldsymbol{x}_{t_1}; t_1, t_2 - t_1\right), \ldots, p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_n} \middle| \boldsymbol{x}_{t_{n-1}}; t_{n-1}, t_n - t_{n-1}\right)$ are independent. This independence arises because in Equation (3), $\boldsymbol{\varepsilon}$ is sampled independently.

Identity Property. The model preserves identity transformation when there is no time difference, i.e., $t_j = t_i$. Under these conditions, by construction, $\boldsymbol{F}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_i}, 0; t_i) = \boldsymbol{x}_{t_i}$, $\boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_i}, 0; t_i) = \boldsymbol{x}_{t_i}$, and $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_i}, t_i, 0) = \mathbf{0}$. Therefore,

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_i}, t_i, 0, \boldsymbol{\varepsilon}) = \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{F}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_i}, 0; t_i) + \mathbf{0}; 0, t_i) = \boldsymbol{x}_{t_i}, \tag{13}$$

which satisfies the identity property.

Stationarity. In the case of autonomous SDEs, the model exhibits stationarity. This is ensured by the architecture's independence from the specific values of times t_i and t_j, focusing instead on the time difference $t_j - t_i$. The conditional distributions thus remain invariant under any shift in time, which satisfies the stationarity condition for autonomous SDEs.

C Derivation of the Flow Loss

This appendix elaborates on the derivations crucial for understanding the flow loss used in the neural stochastic flow model.

In our research, we utilise an auxiliary variational distribution [1, 13, 16] to constrain the upper bounds of bidirectional KL divergences. This approach effectively encourages the model to adhere to the flow property. By minimising these upper bounds, we guide the model towards satisfying the Chapman-Kolmogorov relation, as shown in Equation (4), thereby improving its consistency with the theoretical requirements of stochastic flows of diffeomorphisms. Specifically, we define $\mathcal{L}_{\text{flow}}$ as the expectation of a weighted sum of two components:

$$\mathcal{L}_{\text{flow}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathop{\mathbb{E}}_{p(t_i, t_j, t_k)}\left[\lambda_{\text{1-to-2}} \mathcal{L}_{\text{flow, 1-to-2}}(\boldsymbol{\theta}, \boldsymbol{\phi}; t_i, t_j, t_k) + \lambda_{\text{2-to-1}} \mathcal{L}_{\text{flow, 2-to-1}}(\boldsymbol{\theta}, \boldsymbol{\phi}; t_i, t_j, t_k)\right], \tag{14}$$

where $\lambda_{\text{1-to-2}}$ and $\lambda_{\text{2-to-1}}$ are weighting factors. The two components, $\mathcal{L}_{\text{flow, 1-to-2}}$ and $\mathcal{L}_{\text{flow, 2-to-1}}$, correspond to the upper bounds of the one-step to two-step and two-step to one-step KL divergences, respectively. In our experiments, we set both $\lambda_{\text{1-to-2}}$ and $\lambda_{\text{2-to-1}}$ to 0.4, balancing the contribution of each component to the overall flow loss.

In our experiment, which focuses on the autonomous case, the triplet (t_i, t_j, t_k) is sampled according to the following procedure:

- The initial time t_i is sampled from the data $\mathcal{D}$, representing the starting times in the dataset.[4]
- The final time t_k is sampled from a mixture distribution $p(t_k) = \frac{1}{2}\mathcal{U}(t_i, t_i + t_{\text{max}}) + \frac{1}{2}p_{\text{data}}\left(t_k | t_i\right)$, where $\mathcal{U}(t_i, t_i + t_{\text{max}})$ denotes the uniform distribution over the interval $[t_i, t_i + t_{\text{max}}]$, and $p_{\text{data}}\left(t_k | t_i\right)$ is the conditional data distribution of end times corresponding to the sampled initial time t_i. Here, t_{max} represents the maximum time horizon for model predictions. In our experiments, we use $t_{\text{max}} = 1$.

[4]For non-autonomous SDEs, sampling t_i uniformly or similarly to t_k ensures the Chapman-Kolmogorov equation is satisfied across all time points, which is crucial as transition probabilities depend on absolute time. This approach helps capture the full dynamics of non-autonomous systems.

- The intermediate time t_j is uniformly sampled from the interval $[t_i, t_k]$, i.e., $t_j \sim \mathcal{U}(t_i, t_k)$.

This sampling strategy is designed to reflect the structure of the data, where each sample in the dataset is a tuple consisting of the starting time, its corresponding state, the ending time, and the state at that time.

The inclusion of $p_{\text{data}}\left(t_k | t_i\right)$ in the mixture distribution for sampling t_k is motivated by our observation that the model achieves higher accuracy in regions where data exists. By incorporating this data-driven component, we aim to enhance the efficiency of the learning process, leveraging the model's improved performance in data-rich areas.

By employing this method of sampling, we ensure that the flow loss $\mathcal{L}_{\text{flow}}$ is computed in a manner that is consistent with the temporal dynamics represented in the data, thereby promoting a more accurate approximation of the true stochastic process.

C.1 One-step to Two-step KL Divergence

We begin with the derivation of the KL divergence from a one-step computation to a marginalised two-step computation. To clarify the computation, subscripts $p^1_{\boldsymbol{\theta}}$ and $p^2_{\boldsymbol{\theta}}$ distinguish between the distributions for 1-step sampling from $\boldsymbol{x}_{t_i}$ to $\boldsymbol{x}_{t_k}$ and two-step sampling from $\boldsymbol{x}_{t_i}$ to $\boldsymbol{x}_{t_k}$ via $\boldsymbol{x}_{t_j}$. The key steps are outlined as follows:

$$
\begin{aligned}
& D_{\text{KL}}\left[p^1_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_i}\right) \middle\| \int p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_j}\right) p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}\right) d\boldsymbol{x}_{t_j}\right] && (15)\\
&= \int p^1_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_i}\right) \log\left[\frac{p^1_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_i}\right)}{\int p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_j}\right) p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}\right) d\boldsymbol{x}_{t_j}}\right] d\boldsymbol{x}_{t_k} && (16)\\
&= \int p^1_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_i}\right) \log\left[\frac{p^1_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_i}\right) p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)}{p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_j}\right) p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}\right)}\right] d\boldsymbol{x}_{t_k} && (17)\\
&= \iint p^1_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_i}\right) q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right) \Bigg[\log\left[\frac{p^1_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_i}\right) q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)}{p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_j}\right) p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}\right)}\right] \\
&\qquad + \log\left[\frac{p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)}{q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)}\right]\Bigg] d\boldsymbol{x}_{t_j} d\boldsymbol{x}_{t_k} && (18)\\
&= \mathop{\mathbb{E}}_{p^1_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_i}\right) q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)} \Bigg[\log\left[\frac{p^1_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_i}\right) q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)}{p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_j}\right) p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}\right)}\right] \\
&\qquad + D_{\text{KL}}\left[q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right) \| p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)\right]\Bigg] && (19)\\
&\le \mathop{\mathbb{E}}_{p^1_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_i}\right) q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)} \left[\log\left[\frac{p^1_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_i}\right) q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)}{p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_j}\right) p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}\right)}\right]\right] && (20)\\
&=: \mathcal{L}_{\text{flow, 1-to-2}}(\boldsymbol{\theta}, \boldsymbol{\phi}; t_i, t_j, t_k). && (21)
\end{aligned}
$$

The equation from the second line to the third line is obtained using Bayes' theorem and the Markov property:

$$
\int p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_j}\right) p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}\right) d\boldsymbol{x}_{t_j} = \frac{p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} | \boldsymbol{x}_{t_j}\right) p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}\right)}{p^2_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j} | \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)}. \tag{22}
$$

Note that for clarity, the time arguments are omitted from the probability distributions within the equations.

C.2 Two-step to One-step KL Divergence

Building upon the previous analysis, we now explore the derivation of the KL divergence from a marginalised two-step computation to a one-step computation. We denote this as $\mathcal{L}_{\text{flow, 2-to-1}}$. The key steps are outlined as follows:

$$
\begin{aligned}
& D_{\text{KL}}\left[\int p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_j}\right) p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}\right) d\boldsymbol{x}_{t_j} \middle\| p_{\boldsymbol{\theta}}^{1}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_i}\right)\right] && (23)\\
&= \int\left[\int p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_j}\right) p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}\right) d\boldsymbol{x}_{t_j}\right] \log\left[\frac{p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_j}\right) p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}\right)}{p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right) p_{\boldsymbol{\theta}}^{1}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_i}\right)}\right] d\boldsymbol{x}_{t_k} && (24)\\
&= \iint p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}\right) p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_j}\right) \log\left[\frac{p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}\right) p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_j}\right)}{p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right) p_{\boldsymbol{\theta}}^{1}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_i}\right)}\right] d\boldsymbol{x}_{t_j} d\boldsymbol{x}_{t_k} && (25)\\
&= \mathop{\mathbb{E}}_{p_{\boldsymbol{\theta}}^{2}(\boldsymbol{x}_{t_j}|\boldsymbol{x}_{t_i}) p_{\boldsymbol{\theta}}^{2}(\boldsymbol{x}_{t_k}|\boldsymbol{x}_{t_j})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}\right) p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_j}\right)}{p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right) p_{\boldsymbol{\theta}}^{1}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_i}\right)}\right]\right] && (26)\\
&= \mathop{\mathbb{E}}_{p_{\boldsymbol{\theta}}^{2}(\boldsymbol{x}_{t_j}|\boldsymbol{x}_{t_i}) p_{\boldsymbol{\theta}}^{2}(\boldsymbol{x}_{t_k}|\boldsymbol{x}_{t_j})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}\right) p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_j}\right)}{q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right) p_{\boldsymbol{\theta}}^{1}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_i}\right)}\right] - \log\left[\frac{p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)}{q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)}\right]\right] && (27)\\
&= \mathop{\mathbb{E}}_{p_{\boldsymbol{\theta}}^{2}(\boldsymbol{x}_{t_j}|\boldsymbol{x}_{t_i}) p_{\boldsymbol{\theta}}^{2}(\boldsymbol{x}_{t_k}|\boldsymbol{x}_{t_j})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}\right) p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_j}\right)}{q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right) p_{\boldsymbol{\theta}}^{1}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_i}\right)}\right]\right] \\
&\qquad - \mathop{\mathbb{E}}_{p_{\boldsymbol{\theta}}^{2}(\boldsymbol{x}_{t_k}|\boldsymbol{x}_{t_i})}\left[D_{\text{KL}}\left[p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right) \middle\| q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)\right]\right] && (28)\\
&\leq \mathop{\mathbb{E}}_{p_{\boldsymbol{\theta}}^{2}(\boldsymbol{x}_{t_j}|\boldsymbol{x}_{t_i}) p_{\boldsymbol{\theta}}^{2}(\boldsymbol{x}_{t_k}|\boldsymbol{x}_{t_j})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}\right) p_{\boldsymbol{\theta}}^{2}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_j}\right)}{q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right) p_{\boldsymbol{\theta}}^{1}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_i}\right)}\right]\right] && (29)\\
&=: \mathcal{L}_{\text{flow, 2-to-1}}(\boldsymbol{\theta}, \boldsymbol{\phi}; t_i, t_j, t_k). && (30)
\end{aligned}
$$

From the first line to the second line, we use the relationship in Equation (22). Although the expression inside the log appears to depend on $\boldsymbol{x}_{t_j}$ at first glance, it does not actually depend on $\boldsymbol{x}_{t_j}$ due to this relationship. Therefore, the range of integration is changed to integrate over $\boldsymbol{x}_{t_j}$ first.

By minimising $\mathcal{L}_{\text{flow}}$, we encourage the model to satisfy the flow property in both directions simultaneously. This approach provides a flexible way to balance the trade-off between model expressiveness and adherence to theoretical constraints, which is crucial for the performance and reliability of neural stochastic flow models.

C.3 Data Augmentation

Although not used in the current experiment, we observe that when the variation in time intervals within the dataset is extremely limited, a data augmentation technique can be effective. This technique is equivalent to the one-step to two-step KL divergence flow loss and can help improve the model's performance in scenarios where the dataset lacks diversity in time intervals, thereby enhancing the model's ability to generalise across different time scales.

The data augmentation loss $\mathcal{L}_{\text{aug}}$ is defined as the following negative log-likelihood:

$$
\mathcal{L}_{\text{aug}}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{(\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_j}) \sim \mathcal{D}}\left[\mathop{\mathbb{E}}_{\substack{t_k - t_j \sim \mathcal{D} \\ p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_k}|\boldsymbol{x}_{t_j})}}\left[-\log\left[p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_i}\right)\right]\right]\right]. \tag{31}
$$

Here, $(\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_j})$ are sampled from the dataset $\mathcal{D}$, time interval $t_k - t_j$ is sampled from the time intervals in $\mathcal{D}$, and $\boldsymbol{x}_{t_k}$ is then sampled from the model distribution $p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k} \middle| \boldsymbol{x}_{t_j}\right)$.

It is worth noting that we sample both $t_j - t_i$ (implicitly through $\boldsymbol{x}_{t_j}$) and $t_k - t_j$ from the dataset $\mathcal{D}$. This approach is taken because the model tends to exhibit higher accuracy for time intervals that exist in the training data. By sampling time intervals from the dataset, we focus on improving the model's performance for these more reliable time scales, which can then be generalised to other intervals.

This loss function encourages the model to generate consistent predictions over multiple time steps, even when the training data lacks diverse time intervals. It does this by comparing the direct prediction from $\boldsymbol{x}_{t_i}$ to $\boldsymbol{x}_{t_k}$ with the two-step prediction via $\boldsymbol{x}_{t_j}$.

Interestingly, this augmentation loss is closely related to the KL divergence between the marginalised two-step prediction and the direct one-step prediction:

$$D_{\mathrm{KL}}\left[\mathrm{sg}\left(\int p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k}\middle|\boldsymbol{x}_{t_j}\right) p_{\mathrm{data}}\left(\boldsymbol{x}_{t_j}\middle|\boldsymbol{x}_{t_i}\right) d\boldsymbol{x}_{t_j}\right)\middle\|\, p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_k}\middle|\boldsymbol{x}_{t_i}\right)\right] = \mathcal{L}_{\mathrm{aug}} + \mathrm{const.} \tag{32}$$

In this equation, $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator, which treats its argument as a constant with respect to gradients. This relationship shows that minimising $\mathcal{L}_{\mathrm{aug}}$ is equivalent to minimising the KL divergence between the marginalised two-step prediction and the direct one-step prediction, up to a constant term.

D Optimisation Procedure

Algorithm 1 outlines the optimisation procedure for our neural stochastic flow model. This procedure first optimises the auxiliary model parameters $\boldsymbol{\phi}$, which guide the variational distribution $q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j}\middle|\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)$ to align with the model's conditional distribution $p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j}\middle|\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)$, before updating the main model parameters $\boldsymbol{\theta}$. This sequencing ensures that the main model is updated based on an auxiliary model that closely approximates the model's own transition density, leading to more consistent parameter estimates and improved satisfaction of the flow property.

Algorithm 1: Optimisation Procedure for Neural Stochastic Flows

```
Input: Dataset $\mathcal{D}$, batch size $B$, number of inner optimisation steps $K$
Output: Optimised model parameters $\boldsymbol{\theta}$, auxiliary model parameters $\boldsymbol{\phi}$
while not converged do
    Sample batch $\{(\boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_j}, t_i, t_j)_b\}_{b=1}^{B} \sim \mathcal{D}$;
    for $k = 1$ to $K$ do
        Update $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \eta_\phi \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\mathrm{flow}}(\boldsymbol{\theta}, \boldsymbol{\phi})$;
    end
    Compute $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = -\frac{1}{B}\sum_{b=1}^{B} \log p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t_j,b}\middle|\boldsymbol{x}_{t_i,b}; t_{i,b}, t_{j,b} - t_{i,b}\right) + \mathcal{L}_{\mathrm{flow}}(\boldsymbol{\theta}, \boldsymbol{\phi})$;
    Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_\theta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$;
end
return $\boldsymbol{\theta}, \boldsymbol{\phi}$
```

We train the models using the AdamW optimiser with a learning rate of 3×10^{-4} and weight decay of 10^{-5}, using a batch size of 2,048. The training and test dataset sizes are 100,000 samples each. For the auxiliary model optimisation, we perform $K = 5$ inner optimisation steps per main model update.

E Experimental Details

In our experiments, we consider two types of SDEs: the OU process and GBM. For an n-dimensional system, we employ independent equations for each dimension in both processes.

OU Process The OU process is represented as:

$$d\boldsymbol{x}_t = \boldsymbol{\Theta}(\boldsymbol{\mu} - \boldsymbol{x}_t)dt + \boldsymbol{\Sigma}^{1/2} d\boldsymbol{W}_t. \tag{33}$$

GBM The GBM is represented as:

$$d\boldsymbol{x}_t = \mathrm{diag}(\boldsymbol{\mu})\boldsymbol{x}_t dt + \mathrm{diag}(\boldsymbol{x}_t)\boldsymbol{\Sigma}^{1/2} d\boldsymbol{W}_t. \tag{34}$$

In both equations, $\boldsymbol{\Theta} = \text{diag}([\theta_1, \ldots, \theta_n])$, $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_n]^T$, $\boldsymbol{\Sigma} = \text{diag}([\sigma_1^2, \ldots, \sigma_n^2])$, and $\boldsymbol{W}_t$ is an n-dimensional Wiener process. Here, diag$(\cdot)$ denotes a diagonal matrix with the given elements on its main diagonal.

E.1 Parameter Sampling

OU Process For the OU process, we sample parameters for each dimension independently as follows:

- Mean reversion rate: $\theta_i \sim \mathcal{U}(0.1, 2.0)$
- Long-term mean: $\mu_i \sim \mathcal{N}(0, 1)$
- Volatility: $\sigma_i \sim \mathcal{U}(0.1, 1.0)$

Initial values are sampled from $\mathcal{N}(0, 1)$ for each dimension.

GBM For GBM, we use for each dimension:

- Drift: $\mu_i \sim \mathcal{N}(0, 0.5^2)$
- Volatility: $\sigma_i \sim \mathcal{U}(0.1, 0.5)$

Initial values are sampled from $|\mathcal{N}(0, 1)|$ for each dimension to ensure positivity.

E.2 Specific Parameter Instances for Numerical Experiments

Table 1 presents specific instances of the OU process and GBM used in our detailed analyses (Figures 2, 3, and 4). These parameters are single samples randomly drawn from the distributions specified in Appendix E.1. We show both 2D and 4D configurations to demonstrate our model's capability across different dimensionalities.

Table 1: Specific Instances of Parameters for OU and GBM Processes

Process	**Parameters**	**Initial Value**	**Used in**
OU (2D)	$\boldsymbol{\Theta} = \text{diag}([0.3469, 1.0038])$ $\boldsymbol{\mu} = [-0.8245, 0.6506]^T$ $\boldsymbol{\Sigma} = \text{diag}([0.4934^2, 0.8494^2])$	$\boldsymbol{x}_0 = [1, 1]^T$	Figure 2a Figure 3a
OU (4D)	$\boldsymbol{\Theta} = \text{diag}([1.5789, 0.4698, 0.9868, 0.1832])$ $\boldsymbol{\mu} = [0.4127, 0.4308, 2.1416, -0.4064]^T$ $\boldsymbol{\Sigma} = \text{diag}([0.3932^2, 0.4334^2, 0.5226^2, 0.2705^2])$	$\boldsymbol{x}_0 = [1, 1, 1, 1]^T$	Figure 3c Figure 4a
GBM (2D)	$\boldsymbol{\mu} = [0.0330, 0.5636]^T$ $\boldsymbol{\Sigma} = \text{diag}([0.2774^2, 0.1909^2])$	$\boldsymbol{x}_0 = [0.25, 0.25]^T$	Figure 2b Figure 3b
GBM (4D)	$\boldsymbol{\mu} = [0.3716, 0.2716, -0.3328, 0.1161]^T$ $\boldsymbol{\Sigma} = \text{diag}([0.2550^2, 0.2153^2, 0.3730^2, 0.1559^2])$	$\boldsymbol{x}_0 = [0.25, 0.25, 0.25, 0.25]^T$	Figure 3d Figure 4b

E.3 Analytical Solutions

For both processes, we utilise their known analytical solutions to compute the ground truth distributions:

OU Process The OU process is modelled as a Gaussian distribution with known mean and covariance. In the general multivariate case, the mean and covariance at time t are given by:

$$\mathbb{E}[\boldsymbol{x}_t] = \boldsymbol{\mu} + e^{-\boldsymbol{\Theta}t}(\boldsymbol{x}_0 - \boldsymbol{\mu}) \tag{35}$$

$$\text{Cov}[\boldsymbol{x}_t] = \int_0^t e^{-\boldsymbol{\Theta}(t-s)} \boldsymbol{\Sigma} e^{-\boldsymbol{\Theta}^T(t-s)} ds. \tag{36}$$

When each dimension of the OU process is independent, meaning that both $\boldsymbol{\Theta}$ and $\boldsymbol{\Sigma}$ are diagonal matrices, the covariance matrix simplifies considerably. Each diagonal element of the covariance matrix can be explicitly calculated, yielding:

$$\text{Cov}[\boldsymbol{x}_t] = \text{diag}\left(\frac{\sigma_1^2}{2\theta_1}(1 - e^{-2\theta_1 t}), \ldots, \frac{\sigma_n^2}{2\theta_n}(1 - e^{-2\theta_n t})\right). \tag{37}$$

GBM For the GBM, we consider the case where each dimension is independent. Assuming $x_{0,i} > 0$ for all i (which is typical in most applications), at time t, the distribution for each dimension i is given by:

$$x_{t,i} \sim \text{LogNormal}\left(\log(x_{0,i}) + (\mu_i - \frac{1}{2}\sigma_i^2)t, \sigma_i^2 t\right), \quad i = 1, \ldots, n. \tag{38}$$

Here, LogNormal(μ, σ^2) denotes the univariate log-normal distribution. If $y \sim$ LogNormal(μ, σ^2), then $\log(y) \sim \mathcal{N}(\mu, \sigma^2)$. The parameters μ and σ^2 are the mean and variance of the associated normal distribution of the logarithm of the random variable.

E.4 Model Architecture

Our neural stochastic flow model employs the architecture described in Equation (5), with the following specifications:

- $\boldsymbol{F_\theta}(\boldsymbol{x}_{t_i}, t_j - t_i)$ and $\boldsymbol{G_\theta}(\boldsymbol{x}_{t_i}, t_j - t_i)$: These are implemented as 4-layer coupling flows [2]. Each coupling flow layer uses a multi-layer perceptron (MLP) with the following structure:
  - Input: $\boldsymbol{x}_{t_i}$ and $t_j - t_i$
  - Two hidden layers of 128 units each with ReLU activation functions
  - Output layer with dimension matching $\boldsymbol{x}_{t_i}$

  We employ hyperbolic tangent rescaling [4] for numerical stability.
- $\boldsymbol{s_\theta}(\boldsymbol{x}_{t_i}, t_j - t_i)$: This is implemented as an MLP with the following structure:
  - Input: $\boldsymbol{x}_{t_i}$ and $t_j - t_i$
  - Two hidden layers of 128 units each with ReLU activation functions
  - Output layer with dimension matching $\boldsymbol{x}_{t_i}$
  - The output is passed through a softplus function and multiplied by $\sqrt{t_j - t_i}$ to ensure positivity, proper scaling with time, and to ensure it outputs a zero vector when the time difference is zero
- Auxiliary model $q_{\boldsymbol{\phi}}\left(\boldsymbol{x}_{t_j} \middle| \boldsymbol{x}_{t_i}, \boldsymbol{x}_{t_k}\right)$: The architecture of this model is chosen based on the problem setting:
  - For OU process: We employ a parameterised Gaussian distribution. Its parameters are predicted by an MLP with the following structure:
    * Input: $\boldsymbol{x}_{t_i}$, $\boldsymbol{x}_{t_k}$, $t_j - t_i$, and $t_k - t_j$
    * Two hidden layers of 128 units each with ReLU activation functions
    * Output layer split into two parts: one for the mean and one for the standard deviation (via softplus transformation)
  - For GBM: We use a conditional normalising flow approach with the following structure:
    * Input: Sample from a standard normal distribution
    * 4 coupling flow layers, each implemented as an MLP with:
      · Input: Partial vector from previous layer, $\boldsymbol{x}_{t_i}$, $\boldsymbol{x}_{t_k}$, $t_j - t_i$, and $t_k - t_j$
      · Two hidden layers of 128 units each with ReLU activation functions
      · Output layer for affine transformation parameters

As we are modelling autonomous SDEs, the t_i condition parameter is omitted from $\boldsymbol{F_\theta}$, $\boldsymbol{G_\theta}$, and $\boldsymbol{s_\theta}$, as mentioned in Section 3. These functions are conditioned sorely on the time difference $t_j - t_i$.

E.5 Evaluation Setup for KL Divergence Analysis

For the KL divergence analysis presented in Figure 1, we use the following setup:

- 8 different SDEs for each process (OU and GBM), with parameters sampled as described in Appendix E.1
- 5 distinct initial conditions for each SDE, sampled according to the method outlined in Appendix E.1
- Evaluations at 20 time points, with intervals of 0.05 from 0.05 to 1.00
- 1,000 samples from both the predicted and true distributions at each time point

The KL divergence is computed for each combination of SDE, initial condition, and time point. We then calculate the mean and standard deviation of these KL divergences across all SDEs and initial conditions for each time point.

F Experimental Results

F.1 Entropy Analysis

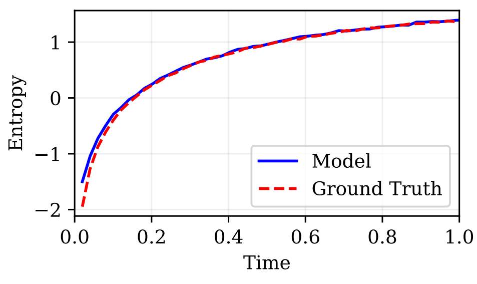
Figure 3 illustrates the temporal evolution of entropy for our OU and GBM processes in both 2D and 4D configurations. These results are obtained using the specific parameter instances described in Table 1. The plots compare the entropy values calculated from the ground truth distributions with those estimated by our model over time. The horizontal axis represents the time progression, whilst the vertical axis shows the corresponding entropy values. These entropy plots provide valuable insights into how well our model captures the dynamics of uncertainty in the OU and GBM processes.

The OU process, with its linear dynamics and Gaussian solution, shows excellent agreement between model estimates and ground truth. The GBM, despite its linear SDE, exhibits more complex behaviour due to its multiplicative noise term and log-normal solution, leading to slight deviations in model estimates. Entropy analysis suggests that increasing dimensionality from 2D to 4D does not substantially affect model performance for either process. Notably, for both processes, the model overestimates entropy at small t, resulting in higher KL divergence in this region (Figure 1). This phenomenon is likely due to the model's entropy decreasing more slowly than the ground truth as t approaches zero.

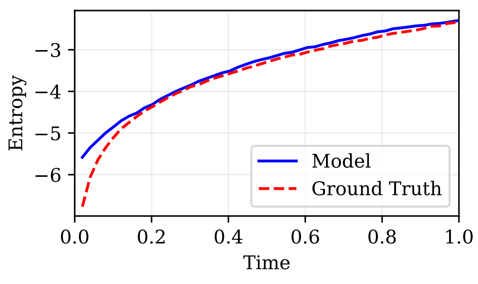
F.2 Probability Density Function Analysis

Figure 4 illustrates the estimated probability density functions for the first two dimensions of our 4D processes, as specified in Table 1. These estimations are derived using a histogram-based approach with 500,000 samples and a 50x50 grid for each axis. The horizontal axis represents the first dimension, whilst the vertical axis denotes the second dimension.

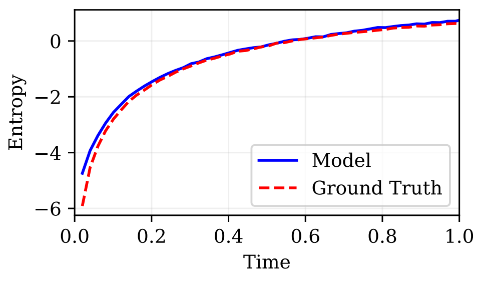
The model demonstrates robust performance in capturing the Gaussian nature of the OU process across higher dimensions. For the GBM, while the overall trend is well-captured, subtle discrepancies emerge in the estimated probability densities. Notably, despite the model's use of coupling flow layers, which theoretically allow for the representation of non-Gaussian distributions, the estimates tend to converge towards more Gaussian-like shapes. This behaviour suggests that while the model architecture is capable of expressing complex, non-Gaussian distributions like the log-normal distribution of the GBM, it appears to favour simpler, more Gaussian-like approximations. This tendency may contribute to the increased complexity in accurately modelling the log-normal distribution of the GBM, especially as dimensionality increases.



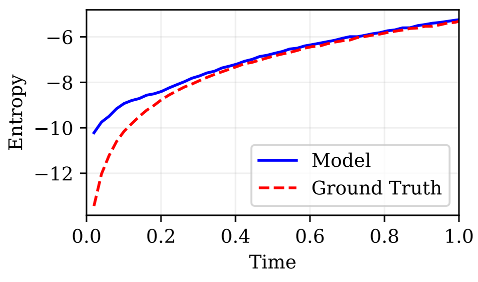
(a) 2D OU Process



(b) 2D GBM

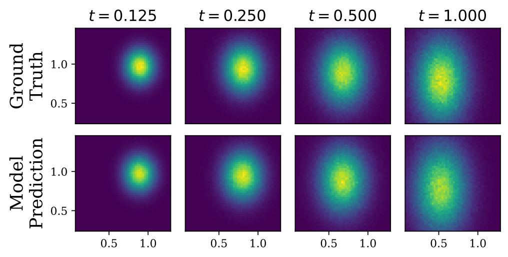


(c) 4D OU Process

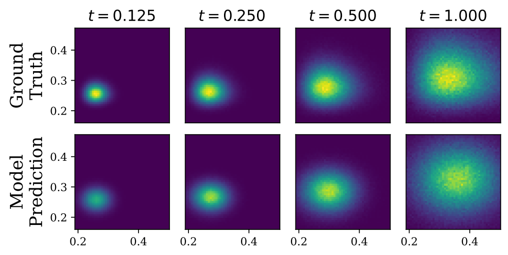


(d) 4D GBM

Figure 3: Temporal evolution of entropy for the OU and GBM processes in 2D and 4D configurations. The plots compare the ground truth entropy values with those estimated by our model over time.



(a) 4D OU Process



(b) 4D GBM

Figure 4: Temporal evolution of probability density for the 4D OU process and GBM, marginalised over the third and fourth dimensions. The colour gradient represents the density, with lighter hues indicating higher probability densities and darker shades representing lower densities.