MULTILEVEL APPROACH TO EFFICIENT GRADIENT CALCULATION IN STOCHASTIC SYSTEMS

Joohwan Ko[1], **Michael Poli**[2], **Stefano Massaroli**[3], **Woochang Kim**[1]
[1]Department of Industrial & Systems Engineering, KAIST
[2]Department of Computer Science, Stanford University
[3]MILA
{juanko,wkim}@kaist.ac.kr
poli@stanford.edu
stefano.massaroli@mila.quebec

ABSTRACT

Gradient estimation in Stochastic Differential Equations is a critical challenge in fields that require dynamic modeling of stochastic systems. While there have been numerous studies on pathwise gradients, the calculation of expectations over different realizations of the Brownian process in SDEs is occasionally not considered. Multilevel Monte Carlo offers a highly efficient solution to this problem, greatly reducing the computational cost in stochastic modeling and simulation compared to naive Monte Carlo. In this study, we utilized Neural Stochastic Differential Equations as our stochastic system and demonstrated that the accurate gradient could be effectively computed through the use of MLMC.

1 INTRODUCTION

Gradient estimation in the dynamic modeling of stochastic systems has always been a critical challenge in various fields. In these systems, calculating the gradient of underlying Stochastic Differential Equations (SDEs) requires consideration of different realizations of the Brownian process. Despite extensive research on pathwise gradients, the calculation of expectations over different realizations of the Brownian process in SDEs is occasionally overlooked.

This is where Multilevel Monte Carlo (MLMC) comes in as a highly effective solution to this problem. By greatly reducing the computational cost in stochastic modeling and simulation compared to naive Monte Carlo (MC), it offers a more efficient way to estimate gradients. It has been shown that to achieve an accuracy of ϵ, MC simulations would require a computational cost of $O(\epsilon^{-3})$. However, with MLMC, this cost can be reduced to $O(\epsilon^{-2})$ under certain conditions (Giles, 2015), making it a much more efficient solution for gradient estimation in SDEs.

In this study, we explore the use of Neural Stochastic Differential Equations (Neural SDEs) as a stochastic system and demonstrate the effectiveness of the MLMC approach in accurately computing the gradient. Also, we show that using MLMC not only reduces the computational cost compared to naive MC simulations but also helps improve training in the LatentSDE model.

2 BACKGROUND

2.1 NEURAL SDES

SDEs are a generalization of ODEs, and Neural SDEs are often used to model the dynamics of stochastic systems (Liu et al., 2019). They are defined as input-injected SDEs in either Itô or Stratonovich forms. Our consideration of systems follows the structure in Kidger et al. (2021):

$$Z_0 = \zeta_\theta(X), \quad \mathrm{d}Z_t = f_\theta\left(t, X, Z_t\right)\mathrm{d}t + g_\theta\left(t, X, Z_t\right)\mathrm{d}W_t, \quad Y_t = \ell_\theta\left(Z_t\right), \tag{1}$$

where ζ_θ, f_θ, and g_θ are neural networks, and ℓ_θ is affine.

The training process for Neural SDEs can be generalized by minimizing the expected objective, $\mathcal{L}(Y)$, over different realizations of the Brownian process W_t. It can be achieved by solving a nonlinear program while optimizing parameters for the system, θ. We can write down the nonlinear program constrained to the Neural SDE as equation 2.

$$\begin{aligned} \min_{\theta} \quad & \mathbb{E}_x \left[\mathbb{E}_W \left[\mathcal{L}(Y)\right]\right] \\ \text{s.t.} \quad & \mathrm{d}Z_t = f_\theta\left(t, X, Z_t\right)\mathrm{d}t + g_\theta\left(t, X, Z_t\right)\mathrm{d}W_t \\ & Z_0 = \zeta_\theta(X), \quad Y_t = \ell_\theta\left(Z_t\right) \end{aligned} \tag{2}$$

We can solve equation 2 via stochastic gradient descent, where we need to calculate $\frac{\mathrm{d}}{\mathrm{d}\theta}\mathbb{E}_W\left[\mathcal{L}(Y)\right]$ (Bottou, 2010) and the SGD iteration has assumptions of $\mathcal{L}$ being Lipschitz such that $\mathbb{E}_W\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}(Y)\right] = \frac{\mathrm{d}}{\mathrm{d}\theta}\mathbb{E}_W\left[\mathcal{L}(Y)\right]$. Therefore the exact gradient for the Neural SDE can be estimated by calculating the expectation of pathwise gradients $\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}(Y)$ for each of the simulated independent Brownian process.

2.2 Pathwise Gradients Calculation for SDEs

There is an extensive amount of work on pathwise sensitivities analysis. The forward pathwise approach is an intuitive pathwise gradient calculation method for SDEs (Gobet & Munos, 2005). However, it scales poorly in computational cost with the number of parameters or state dimensions. Giles & Glasserman (2006) proposed a discrete version of the backward adjoint sensitivity method, which calculates the same value with the forward implementation but with computational savings. The stochastic adjoint method (Li et al., 2020) resolved the memory burden by generalizing the adjoint method to stochastic dynamics defined by SDEs to recover the value of the state z_t.

2.3 Multilevel Monte Carlo

Giles (2008) proposed MLMC that reduces the order of complexity of MC simulations. When we want to estimate $\mathbb{E}_W\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}_L(Y)\right]$, there is a sequence $\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}_0(Y), \ldots, \frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}_{L-1}(Y)$, with increasing accuracy and cost. Then, we can use equation 3 as an unbiased estimator for $\frac{\mathrm{d}}{\mathrm{d}\theta}\mathbb{E}_W\left[\mathcal{L}_L(Y)\right]$.

$$\mathbb{E}_W\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}_L(Y)\right] = \mathbb{E}_W\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}_0(Y)\right] + \sum_{\ell=1}^{L}\mathbb{E}_W\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}_\ell(Y) - \frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}_{\ell-1}(Y)\right] \tag{3}$$

Theorem 2.1 (Complexity Theorem). *For a real-valued random variable $\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}(Y)$, we let $\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_l(\widehat{Y})$ be the corresponding approximation using the discretization at level l, i.e. with 2^l steps of width $h_l = 2^{-l}T$.*

If there exists independent estimators $\widehat{U}_l$ of computational complexity C_l based on M_l samples and there are positive constants $\alpha \geq \frac{1}{2}\min(1, \beta), \beta, c_1, c_2, c_3$, such that

$$\begin{aligned} & A1 : \mathbb{E}\left[\widehat{U}_l\right] = \begin{cases} \mathbb{E}\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_0(\widehat{Y})\right] & \text{if } l = 0 \\ \mathbb{E}\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_l(\widehat{Y}) - \frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_{l-1}(\widehat{Y})\right] & \text{if } l > 0 \end{cases} \\ & A2 : \left|\mathbb{E}\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_l(\widehat{Y}) - \frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}(Y)\right)\right| \leq c_1 h_l^\alpha \\ & A3 : \mathbb{V}\left[\widehat{U}_l\right] \leq c_2 h_l^\beta M_l^{-1} \\ & A4 : C_l \leq c_3 M_l h_l^{-1} \end{aligned} \tag{4}$$

Then there is a constant c_4 such that for any $\epsilon < e^{-1}$, there are values for L and $\left(M_l\right)_{l=0,\ldots,L}$ resulting in a multilevel estimator $\widehat{U} = \sum_{l=0}^{L}\widehat{U}_l$ with a mean-square-error $MSE = \mathbb{E}\left[(\widehat{U} - \mathbb{E}(\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}(Y)))^2\right] < \epsilon^2$ with a complexity C bounded by

$$C \leq \begin{cases} c_4\epsilon^{-2} & \textit{if}\, \beta > 1 \\ c_4\epsilon^{-2}(\log\epsilon)^2 & \textit{if}\, \beta = 1 \\ c_4\epsilon^{-2-(1-\beta)/\alpha} & \textit{if}\, 0 < \beta < 1 \end{cases} \tag{5}$$

The theorem 2.1 (Burgos, 2014) shows its advantages in the convergence speed of V_l and guarantees improved computational complexity compared to the standard Monte Carlo for stochastic simulation, which we will discuss in detail in the next section.

3 Monte Carlo Estimation for Exact Gradients

3.1 Computational Complexity of Standard Monte Carlo

Assume we want to estimate the gradient value $V = \mathbb{E}_W\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}(Y)\right]$ with an accuracy ϵ. Let $\mathcal{L}$ be the Lipschitz loss function, and $\mathcal{L}(Y)$ be the scalar loss we want to minimize. Note that we use numerical solvers to compute the SDEs in equation 2. Those solvers, such as Euler or Milstein, use the discretization scheme to approximate the solutions of the SDEs. Therefore we denote the discretized approximation of V, as $\widehat{V} = \mathbb{E}_W\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}(\widehat{Y})\right]$. Also let its Monte Carlo estimator as $\widehat{S} = \frac{1}{M}\sum_{i=1}^{M}\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}(\widehat{Y})^{(i)}$. We can write the mean square error(MSE) of the estimation as in equation 6 (Burgos, 2014).

$$\mathbb{E}\left[(\widehat{S}-V)^2\right] = \frac{1}{M}\mathbb{V}(\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}(\widehat{Y})) + (\mathbb{E}\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}(\widehat{Y})\right] - \mathbb{E}\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}(Y)\right])^2 \tag{6}$$

The first term represents the variance of the estimator due to Monte Carlo sampling and the second term represents the bias due to the discretization. With any solver with weak order of convergence of 1, such as Milstein method (Milstein, 1994), we need $O(\epsilon^{-1})$ time steps and $O(\epsilon^{-2})$ samles to achieve MSE $O(\epsilon^2)$. Therefore, the total computational cost is $C = O(\epsilon^{-3})$(Duffie & Glynn, 1995).

3.2 Computational Complexity of Multilevel Monte Carlo

By following the basic settings of MLMC and using discretization method to approximate the gradient value, we can write the multilevel approximation of the gradient as

$$\begin{aligned} \mathbb{E}_W\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}_L(Y)\right] &\approx \mathbb{E}_W\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_L(\widehat{Y})\right] \\ &= \mathbb{E}_W\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_0(\widehat{Y})\right] + \sum_{\ell=1}^{L}\mathbb{E}_W\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_\ell(\widehat{Y}) - \frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_{\ell-1}(\widehat{Y})\right], \end{aligned} \tag{7}$$

Also, we can define the multilevel estimators as

$$\begin{aligned} \widehat{U}_0 &= M_0^{-1}\sum_{i=1}^{M_0}\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_0(\widehat{Y})^{(i)} \\ \widehat{U}_l &= M_l^{-1}\sum_{i=1}^{M_l}\left(\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_l(\widehat{Y})^{(i)} - \frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_{l-1}(\widehat{Y})^{(i)}\right) \end{aligned} \tag{8}$$

Using the complexity theorem 2.1, we can get the computational complexity of MLMC by estimating the order of convergence α and β. This can be derived analytically with some assumptions as in the theorem B.1. On the other hand, we can conduct simulation experiments to estimate the α and β. In the experiments, we approximate $\widehat{U}_l$ with $\mathbb{E}\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_f - \frac{\mathrm{d}}{\mathrm{d}\theta}\widehat{\mathcal{L}}_c\right]$ for computational efficiency, where the fine discretizations $\widehat{\mathcal{L}}_f$ uses $N_f(l) = 2^l$ fine time steps and the coarse discretizations $\widehat{\mathcal{L}}_c$ uses $N_c(l) = 2^{l-1}$ (Burgos, 2014). Let our system be a Neural SDE as in equation 1 and M_l be the

Table 1: Estimated Complexity

Estimator	α	β	**Complexity**
Loss Value	≈ 1.00	≈ 2.03	$O(\epsilon^{-2})$
Gradient for f_θ	≈ 0.97	≈ 1.87	$O(\epsilon^{-2})$
Gradient for g_θ	≈ 0.97	≈ 1.94	$O(\epsilon^{-2})$
Gradient for z_0	≈ 0.97	≈ 1.95	$O(\epsilon^{-2})$

number of path simulations for each of the levels to estimate $\mathbb{E}\left[\widehat{U}_l\right]$ and $V_l = M_l \mathbb{V}\left[\widehat{U}_l\right]$. Also by assuming

$$\begin{aligned} \mathbb{E}\left(\widehat{U}_l\right) &= O\left(h^\alpha\right) \\ V_l &= O\left(h^\beta\right), \end{aligned} \tag{9}$$

we can estimate the empirical bounds, which approximate the order of convergence, α and β, with

$$\begin{aligned} \log_2 \mathbb{E}\left[\widehat{U}_l\right] &\sim \alpha \log_2(h) \sim -l\alpha \\ \log_2 V_l &\sim \beta \log_2(h) \sim -l\beta. \end{aligned} \tag{10}$$

The table 1 shows the α and β of our system after the experiments, and the figure B.2 shows the values of $\mathbb{E}\left[\widehat{U}_l\right]$ and V_l for different levels. We can compute the complexity C with 5 and conclude that the total computational complexity is $O(\epsilon^{-2})$ for all gradients with respect to f_θ, g_θ, and z_0.

4 Experiment Results and Discussion

To compare the performance of various simulation methods in gradient calculation, we utilized the Latent SDE from Li et al. (2020) as our evaluation method. We trained the Latent SDE on a 3D Stochastic Lorenz Attractor process to confirm that the acquired posterior can accurately reproduce the training data and that the obtained priors are not deterministic. (See C for details)

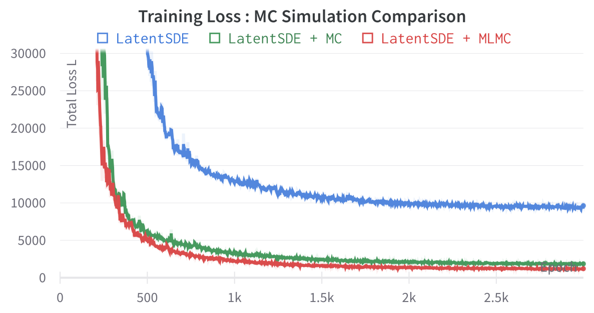

Figure 1: Training Loss: MC Simulation Comparison

Figure 4 shows that integrating MC simulations in the Latent SDE improves performance by accurately calculating gradients. The training time per epoch is similar for both Latent SDE and MC simulation, as parallelization is used to expand samples during training, which only increases memory complexity. However, by integrating MLMC, the computational cost of MC simulation is further reduced. This means that Latent SDE in combination with MLMC achieves similar or even better performance with less computational cost.

REFERENCES

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Sylvestre Burgos. *The computation of Greeks with multilevel Monte Carlo*. PhD thesis, University of Oxford, 2014.

K Andrew Cliffe, Mike B Giles, Robert Scheichl, and Aretha L Teckentrup. Multilevel monte carlo methods and applications to elliptic pdes with random coefficients. *Computing and Visualization in Science*, 14(1):3–15, 2011.

Darrell Duffie and Peter Glynn. Efficient monte carlo simulation of security prices. *The Annals of Applied Probability*, pp. 897–905, 1995.

Michael B Giles. Multilevel monte carlo path simulation. *Operations research*, 56(3):607–617, 2008.

Michael B Giles. Multilevel monte carlo methods. *Acta numerica*, 24:259–328, 2015.

Mike Giles and Paul Glasserman. Smoking adjoints: Fast monte carlo greeks. *Risk*, 19(1):88–92, 2006.

Paul Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer, 2004.

Emmanuel Gobet and Rémi Munos. Sensitivity analysis using itô–malliavin calculus and martingales, and application to stochastic optimal control. *SIAM Journal on control and optimization*, 43(5):1676–1713, 2005.

Patrick Kidger. On neural differential equations. *arXiv preprint arXiv:2202.02435*, 2022.

Patrick Kidger, James Foster, Xuechen Chen Li, and Terry Lyons. Efficient and accurate gradients for neural sdes. *Advances in Neural Information Processing Systems*, 34:18747–18761, 2021.

Peter E. Kloeden and Eckhard Platen. *Strong Taylor Approximations*, pp. 339–371. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992. ISBN 978-3-662-12616-5. doi: 10.1007/978-3-662-12616-5_10.

Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pp. 3870–3882. PMLR, 2020.

Xuanqing Liu, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.

Grigorii Noikhovich Milstein. *Numerical integration of stochastic differential equations*, volume 313. Springer Science & Business Media, 1994.

A Additional Background

A.1 Interchange of Differentiation and Expectation for $\mathcal{L}$

If the following equation is assumed to hold, then the pathwise derivative becomes an unbiased estimator for the true gradient value.

$$\mathbb{E}_W\left[\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}(Y)\right] = \frac{\mathrm{d}}{\mathrm{d}\theta}\mathbb{E}_W\left[\mathcal{L}(Y)\right] \tag{11}$$

The below assumptions(A1-A3) are sufficient to ensure that the equation 11 holds. Details of the proofs can be found in Glasserman (2004).

- A1: At each θ, $\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}(Y_t)$ exists with probability 1, for all $i = 1, \ldots, m$.
- A2: $\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}(Y)$ exists with probability 1 and is given by

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}(Y) = \sum_{i=1}^{T}\frac{\partial\mathcal{L}}{\partial Y_i}(Y(\theta))\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{L}(Y_t)$$

- A3: There exists a constant k_f such that for all $x, y \in \mathbb{R}^m$

$$|\mathcal{L}(x) - \mathcal{L}(y)| \leq k_f\|x - y\|$$

A.2 Strong Ito-Taylor Approximations

The theorem A.1 was first introduced in Kloeden & Platen (1992) to demonstrate the convergence of strong Ito-Taylor approximations. This theorem has been modified from the version presented by Burgos (2014).

Theorem A.1. *We consider the case where $Z_t \in \mathbb{R}^d$ and W_t is a 1-dimensional Brownian motion. $(f_k)_{(k=1..d)}$ and $(g_k)_{(k=1..d)}$ are the different components of the coefficients f and g in the following SDE: on $[0, T]$,*

$$\mathrm{d}Z(t) = f(Z, t)\mathrm{d}t + g(Z, t)\mathrm{d}W_t$$

We let

$$\begin{aligned} K^0 &= \frac{\partial}{\partial t} + \sum_{k=1}^{d} f_k\frac{\partial}{\partial Z_k} + \frac{1}{2}\sum_{k,l=1}^{d} g_k g_l\frac{\partial^2}{\partial Z_k\partial Z_l} \\ K^1 &= \sum_{k=1}^{d} g_k\frac{\partial}{\partial Z_k} \end{aligned} \tag{12}$$

Assuming that

- *A1: $f(Z, t)$ is $\mathcal{C}^{(1,1)}\left(\mathbb{R}^d \times \mathbb{R}^+\right)$ and $g(Z, t)$ is $\mathcal{C}^{(2,1)}\left(\mathbb{R}^d \times \mathbb{R}^+\right)$*
- *A2: (uniform Lipschitz condition): there exists a constant $C_1 > 0$ such that for all $x, y \in \mathbb{R}^d$*

$$\|f(y, t) - g(x, t)\| + \|f(y, t) - g(x, t)\| + \left\|K^1 g(y, t) - K^1 g(x, t)\right\| \leq C_1\|y - x\|$$

- *A3(linear growth bound): There exists a constant C_2 such that for $x \in \mathbb{R}^d$,*

$$\begin{gathered}\|f(x, t)\| + \|K_0 f(x, t)\| + \|K_1 f(x, t)\| + \|g(x, t)\| + \\ \|K_0 g(x, t)\| + \|K_1 g(x, t)\| + \|K_0 K_1 g(x, t)\| + \|K_1 K_1 g(x, t)\| \leq C_2(1 + |x|)\end{gathered}$$

- *A4(additional Lipschitz-like condition): there exists a constant C_3 such that for all $x \in \mathbb{R}^d$ and $s, t \in \mathbb{R}^+$, $\|g(x, t) - g(x, s)\| \leq C_3(1 + |x|)\sqrt{|t - s|}$*

Then for each $m > 0$*,*

$$\mathbb{E}\left(\sup_{0<t<T} \|Z_t\|^m\right) < \infty$$

Using the Milstein discretisation $\left(\widehat{Z}_n\right)_{n=0,\ldots,N}$*, we define the following continuous time interpolant on each time step* $[t_n, t_{n+1}] \quad (n = 0, \ldots, N-1)$ *and each dimension* $k = 1, \ldots, d$ *as*

$$\widehat{Z}_{P_k}(t) = \widehat{Z}_{n_k} + f_k\left(\widehat{Z}_n, t_n\right)(t - t_n) + g_k\left(\widehat{Z}_n, t_n\right)(W_t - W_n)$$
$$+ \frac{1}{2}\left(\sum_{l=1}^{k} g_l \frac{\partial g_k}{\partial Z_l}\left(\widehat{Z}_n, t_n\right)\right)\left((W_t - W_n)^2 - (t - t_n)\right)$$

then for each $m > 0$*, there exists a constant* C_m *such that*

$$\mathbb{E}\left(\sup_{0<t<T} \left\|Z_t - \widehat{Z}_P(t)\right\|^m\right) < C_m h^m$$

and

$$\mathbb{E}\left(\sup_{0<t<T} \left\|\widehat{Z}_P(t)\right\|^m\right) < C_m$$

Proof. See theorem 10.6.3 and corollary 10.6.4 in Kloeden & Platen (1992). □

B Details of Computational Complexity for MLMC

B.1 Complexty Theorem

The theorem 2.1 was originally presented by Giles (2008) and later improved upon by Cliffe et al. (2011). The version of the theorem that appears in this paper, which was taken from Burgos (2014), replaces the original payoff, P, with $\frac{d}{d\theta}\mathcal{L}(Y)$ to align with the notations used in our work.

B.2 Details of Numerical Simulation

We constructed a minimal Neural SDE to evaluate the expected value and variance of the multilevel estimator for computational complexity in our numerical experiments. In both figures, B.2, the term $L_f - L_c$ represents the multilevel estimator for the true loss value $\mathcal{L}$. For the gradient estimators, each of $dL_f/df - dL_c/df$, $dL_f/dg - dL_c/dg$, and $dL_f/dX_0 - dL_c/dX_0$ represents the estimator for the gradient of the drift(f), the diffusion(g), and the input to the SDE, X_0, respectively.

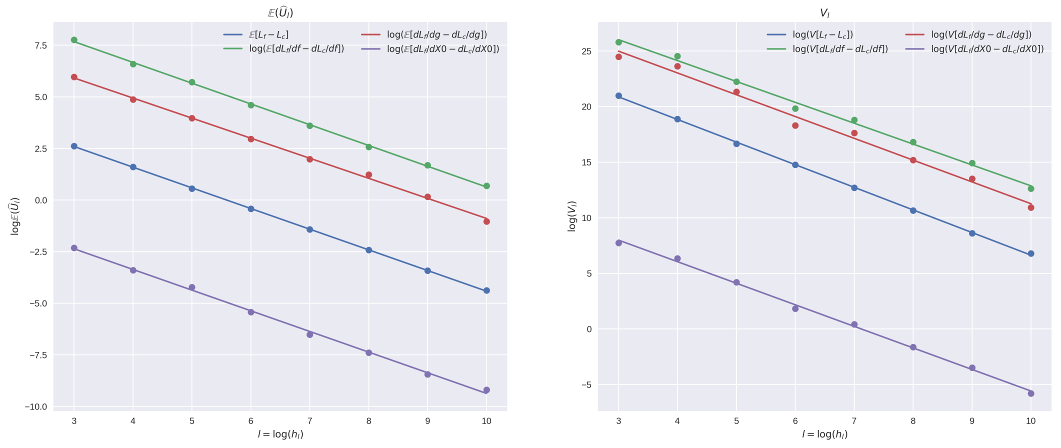

Figure 2: Calculating $\mathbb{E}(\widehat{U}_l)$ and V_l for Estimating Complexity

B.3 Analytical Derivation of Computational Complexity

Theorem B.1. *We consider an output of a stochastic system Z_t on the time interval $[0, T]$ and a smooth Lipschitz loss function $\mathcal{L}$ with a loss value of $\mathcal{L}(Z_T)$. We assume that Z_t follows an Itô process as described by equation 1, that the coefficients of the diffusion f_θ and g_θ satisfy conditions A1 to A4 of theorem A.1, and that $g_\theta > 0$. Then, the multilevel estimators of the gradients with respect to the loss $\mathcal{L}(Z_T)$ have an accuracy ϵ at a cost $O(\epsilon^{-2})$*

Proof. See chapter 4 in Burgos (2014). □

C Details on Experiments for LatentSDE

The prior and the approximated posterior are parameterized using SDEs as

$$\begin{aligned} \mathrm{d}\tilde{Z}_t &= h\left(\tilde{Z}_t, t\right)\mathrm{d}t + g\left(\tilde{Z}_t, t\right)\mathrm{d}W_t \\ \mathrm{d}Z_t &= f\left(Z_t, t\right)\mathrm{d}t + g(Z_t, t)\mathrm{d}W_t, \end{aligned} \tag{13}$$

where h, f, and g are Lipschitz and both have same initial values of z_0.

With prior and approximate posterior defined as in Equation 13, the latent SDE can be constructed as outlined in 1. The KL divergence between the prior and approximate posterior, which both have the same diffusion function, g, is finite and can be calculated by sampling paths from the approximate posterior process, as shown in Li et al. (2020). The formula for the evidence lower bound (ELBO) is

$$\log(p(x_1, x_2, ..., x_N|\theta) \geq \mathbb{E}_{Z_t}\left[\sum_{i=1}^{N} \log p\left(x_{t_i} \mid z_{t_i}\right) - \int_0^T \frac{1}{2}\left|u\left(z_t, t\right)\right|^2 \mathrm{d}t\right],$$

where $u : \mathbb{R}^d \times [0, T] \to \mathbb{R}^m$ satisfies

$$\sigma(z, t)u(z, t) = h_\phi(z, t) - h_\theta(z, t),$$

and the expectation is calculated based on the approximate posterior process that has been defined. Other detailed explanations of the training process, such as the choice of optimizer and architecture of latent SDE, can be found in Kidger (2022).

Algorithm 1 Latent SDE

```
Input: Data $\{x_i\}_{i=1,...,N}$ and times $\{t_i\}_{i=1,...,N}$
$z_0' =$ Encoder($\{x_i\}$)
$\mu_{z_0}, \sigma_{z_0} =$ Linear($z_0'$)
$z_0 \sim \mathcal{N}(\mu_{z_0}, \sigma_{z_0})$
$\{z_i\}, D_{kl} = \mathrm{SDESolve}(f, h, g, z_0, (t_0 \ldots t_N))$
$\tilde{x_i} =$ Linear($\{z_i\}$) for all $i = 1, ..., N$
Return: $\{\tilde{x}_i\}_{i=1...N}$
```

We trained on the dataset sampled from stochastic Lorenz attractor SDE defined as:

$$\begin{aligned} y &\sim \mathcal{N}\left(0, I_{3\times 3}\right) \\ \mathrm{d}y_1(t) &= a_1\left(y_2(t) - y_1(t)\right)\mathrm{d}t + b_1 y_1(t)\mathrm{d}w(t) \\ \mathrm{d}y_2(t) &= \left(a_2 y_1(t) - y_1(t)y_3(t)\right)\mathrm{d}t + b_2 y_2(t)\mathrm{d}w(t) \\ \mathrm{d}y_3(t) &= \left(y_1(t)y_2(t) - a_3 y_3(t)\right)\mathrm{d}t + b_3 y_3(t)\mathrm{d}w(t), \end{aligned} \tag{14}$$

with $a_1 = 10, a_2 = 28, a_3 = \frac{8}{3}, b_1 = 0.1, b_2 = 0.28, b_3 = 0.3$ following the work of Kidger (2022). We obtained samples at intervals of 0.05 from time 0 to 2. To conform with the preprocessing techniques used in previous studies Li et al. (2020), we normalized the data by computing the mean and standard deviation for each dimension. We then added Gaussian noise with a mean of zero and a standard deviation of 0.01 to corrupt the data.