Data-driven generative simulation of SDEs using diffusion models

Xuefeng Gao
CUHK

Jiale Zha
CUHK

Xun Yu Zhou
Columbia University

Abstract

This paper introduces a new approach to generating sample paths of unknown stochastic differential equations (SDEs) using diffusion models, a class of generative AI models commonly employed in image and video applications. Unlike the traditional Monte Carlo methods for simulating SDEs, which require explicit specifications of the drift and diffusion coefficients, our method takes a model-free, data-driven approach. Given a finite set of sample paths from an SDE, we utilize conditional diffusion models to generate new, synthetic paths of the same SDE. To demonstrate the effectiveness of our approach, we conduct a simulation experiment to compare our method with alternative benchmark ones including neural SDEs. Furthermore, in an empirical study we leverage these synthetically generated sample paths to enhance the performance of reinforcement learning algorithms for continuous-time mean–variance portfolio selection, hinting promising applications of diffusion models in financial analysis and decision-making.

1 Introduction

Stochastic Differential Equations (SDEs) are an important class of equations for continuous-time stochastic models that have been widely employed in diverse fields, including finance, physics, operations research and machine learning. In many applications, generating sample paths of SDEs is crucial [4, 11, 17]. When an underlying SDE is known and given, i.e., its drift and diffusion coefficients are specified, Monte Carlo simulation is the classical approach for sample path generation with various methods proposed in the literature [11]. This paper instead considers the problem of simulating an SDE when the underlying equation is *unknown*, with access only to a finite set of sample paths from the unknown SDE.

Traditional approaches to learning an unknown SDE largely rely on model-based methods, where the SDE is assumed to belong to a specific class such as geometric Brownian motions. Parameters of the equation are then estimated using statistical inference techniques [8, 10]. More recent approaches use neural networks to represent drift and diffusion coefficients of the SDEs, and construct appropriate loss functions to train these networks; see e.g. [15, 25]. Once an SDE is learned, more sample paths can then be generated from it by Monte Carlo. Alternatively, several recent studies have explored model-free methods, leveraging generative models to generate samples directly from unknown SDEs, skipping the step of learning the SDEs themselves. For instance, [18] utilizes GANs [5] to simulate unknown SDEs, while [22] employs normalizing flows to generate samples from SDEs.

In this paper, we introduce a new model-free, data-driven approach to generate sample paths of SDEs using (conditional) diffusion models, a cutting-edge class of generative AI models that have achieved remarkable success in image and video applications whose performances surpass those of GANs [2, 6, 16]. The essential idea of diffusion models is to use a forward process to gradually turn the unknown target distribution into a simple noise distribution, and then reverse this process to generate new samples. To generate new sample paths from an unknown SDE, our key methodology is to

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MLxOR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making.

leverage the Markov property and train a sequence of conditional diffusion models to generate samples of stochastic increments of the SDE in an autoregressive manner. We demonstrate the effectiveness of this approach by conducting experiments on various SDEs and comparing its performance with alternative benchmark methods including neural SDEs [15]. Moreover, to illustrate the potential applications of our framework in decision making, we carry out an empirical study to show how we can make use of the synthetically generated sample paths as a market simulator to enhance the performance of model-free reinforcement learning algorithms in continuous-time mean–variance portfolio selection [9, 19].

Technically, our work is closely related to [12], as both use conditional diffusion models to autoregressively simulate SDE increments. The crucial difference lies in the estimation of the score function, a key component of the diffusion models [6, 17]. While [12] designs a Monte Carlo empirical estimator, we opt for training a score neural network which offers significantly greater scalability as demonstrated in our experiments.

In terms of financial applications, a recent working paper [1] applies generative diffusion models to simulate high-dimensional asset returns and concludes that generated data improves the accuracy of parameter estimation (e.g. mean and covariance) and hence helps solve a static portfolio optimization problem in a model-based manner. In this paper, we take a different direction. We consider continuous-time portfolio selection with model-free reinforcement learning (RL) and enrich the market simulator with synthetic paths. This allows the RL agent to experience more market scenarios in her exploration and consequently learn a more robust dynamic portfolio policy.

Finally, we mention that our work is also broadly related to time series generation using generative models (including GANs and diffusion models); see, e.g., [13, 14, 21, 23]. Our work differs from these studies in that we specifically focus on generating sample paths of SDEs that exhibit the Markov property, as well as applying the method to improving RL for dynamic financial portfolio optimization.

2 Problem Formulation and Methodology

Consider the d-dimensional *target* SDE defined on a filtered probability space $\big(\Omega, \mathcal{F}, \mathbb{P}, \{\mathcal{F}_t\}_{0\le t\le T}\big)$:

$$\mathrm{d}\mathbf{X}(t) = \mu(t, \mathbf{X}(t))\mathrm{d}t + \sigma(t, \mathbf{X}(t))\mathrm{d}\mathbf{W}(t),\ t \in [0, T], \tag{1}$$

where $\mu : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ is the drift coefficient, $\sigma : [0, T] \times \mathbb{R}^d \to \mathbb{R}^{d\times m}$ is the diffusion coefficient, and $\mathbf{W}$ is a standard m-dimensional Brownian motion. We assume the SDE starts from a fixed known initial state $\mathbf{X}(0) \equiv \mathbf{x}(0) \in \mathbb{R}^d$, but both functions μ *and* σ *are unknown.* However, we have access to a set of i.i.d. sample paths of the SDE (1) on $[0, T]$. Assuming that the SDE (1) has a unique strong solution, *our goal is to generate additional new sample paths that exhibit (approximately) the same distributional/statistical properties to those of the SDE* (1). In practice, only discrete-time observations of (1) are available. Without loss of generality, we consider a uniform grid $\mathcal{T} := \{t_n : t_n = n\Delta t \text{ for } n = 0, 1, ..., N_T\}$ with the fixed time step $\Delta t > 0$ and assume that the number of observation slots $N_T = T/\Delta t$ is an integer. The dataset available to us consists of $H > 1$ sample paths of $\{\mathbf{X}(t) : t \in [0, T]\}$ observed at grid points in $\mathcal{T}$, denoted by

$$\left\{\mathbf{x}^{(i)}(t_0), \mathbf{x}^{(i)}(t_1), \cdots, \mathbf{x}^{(i)}(t_{N_T}),\quad i = 1, ..., H\right\}. \tag{2}$$

To generate new sample paths, we train conditional diffusion models to simulate the stochastic increment of the target SDE (1) with the fixed time step Δt. Our method can be easily extended to the case when $\mathbf{X}(0)$ follows some unknown initial distribution and the time grid $\mathcal{T}$ is non-uniform. In this short version of the paper, we consider the simpler setting for illustrative propose.

We now give a brief review of the conditional diffusion models, which are built on *Denoising Diffusion Probabilistic Models* (DDPM) [3, 6]. Given a condition $\mathbf{c} \sim \mathbf{C}$, the conditional DDPM first noises the training data $\mathbf{Y}(0) \sim q_0 = p_{\text{target}}(\cdot|\mathbf{c})$ through iteratively adding Gaussian noise for K steps, i.e., $q(\mathbf{Y}(k)|\mathbf{Y}(k-1)) = \mathcal{N}\big(\mathbf{Y}(k); \sqrt{\alpha_k}\mathbf{Y}(k-1), (1-\alpha_k)I_{d_{\text{target}}}\big)$, where $\mathcal{N}\big(\cdot; \mu, \Sigma\big)$ is the Gaussian density with mean vector μ and covariance matrix Σ, and $\{\alpha_k\}_{k=1}^K$ are certain pre-specified noise schedulers. As $k \to K$, the distribution of $\mathbf{Y}(k)$ converges to a standard Gaussian distribution. The generation (or denoising) process is to reverse the above process; namely it starts from the Gaussian noise $\hat{\mathbf{Y}}(K) \sim \mathcal{N}(\cdot; \mathbf{0}, I_{d_{\text{target}}})$ and iteratively employs the following update via a parametric transition

distribution:

$$p_\theta(\hat{\mathbf{Y}}(k-1)|\hat{\mathbf{Y}}(k),\mathbf{c}) = \mathcal{N}\big(\hat{\mathbf{Y}}(k-1); \mu_\theta(\hat{\mathbf{Y}}(k),k,\mathbf{c}),(1-\alpha_k)I_{d_{\text{target}}}\big), \tag{3}$$

where the mean μ_θ is a neural network with parameter θ. To ensure the generated sample $\hat{\mathbf{Y}}(0)$ is distributed (approximately) as $p_{\text{target}}(\cdot|\mathbf{c})$, μ_θ is trained via maximizing the evidence lower bound on the log-likelihood of the data $\mathbb{E}_{\mathbf{c}\sim\mathbf{C},\mathbf{Y}(0)\sim p_{\text{target}}(\cdot|\mathbf{c})}[\log p_\theta(\mathbf{Y}(0)|\mathbf{c})]$, where $p_\theta(\cdot|\mathbf{c})$ is the time-0 marginal distribution of $\hat{\mathbf{Y}}(0)$ of the denoising process (3). The objective is equivalent to [6]:

$$\max_\theta \mathbb{E}_{\mathbf{c}\sim\mathbf{C},\mathbf{Y}(0)\sim p_{\text{target}}(\cdot|\mathbf{c}),\mathbf{Y}(k)\sim q(\cdot|\mathbf{Y}(k-1))}\left[\sum_{k=1}^{K}\log\frac{p_\theta(\mathbf{Y}(k-1)|\mathbf{Y}(k),\mathbf{c})}{q(\mathbf{Y}(k)|\mathbf{Y}(k-1))}\right]. \tag{4}$$

We do not apply other training techniques such as the classifier (free) guidance [2, 7], because they are designed for image generation and do not provide an improvement in performance in our experiments.

For our desired generative model of SDEs, due to the Markov property of the SDE (1), we construct it in an auto-regressive manner. Specifically, we simulate the conditional increment $\Delta\mathbf{X}(t;\mathbf{x}) := \mathbf{X}^{t,\mathbf{x}}(t+\Delta t) - \mathbf{x}$ defined by the solution to (1) conditional on $\mathbf{X}^{t,\mathbf{x}}(t) = \mathbf{x}$. We then set condition $\mathbf{c}_n = (t_n, \mathbf{x}(t_n))$ sampled from $\mathbf{C}_n = (t_n, \mathbf{X}(t_n))$ and train a series of generative models $\{G(\mathbf{x}, t_n; \theta_n)\}_{n=1}^{N_T}$, such that each $G(\mathbf{x}, t_n; \theta_n)$ consists of a parametric transition kernel p_{θ_n} and generates new samples $\hat{\mathbf{Y}}_n(0)$ distributed as $\Delta\mathbf{X}(t_n;\mathbf{x})$ by running (3), i.e.,

$$\hat{\mathbf{Y}}_n(0) \overset{d}{\approx} \Delta\mathbf{X}(t_n;\mathbf{x}) := \mathbf{X}(t_{n+1}) - \mathbf{X}(t_n) = \mathbf{X}(t_{n+1}) - \mathbf{x}. \tag{5}$$

In the generation phase, we start from $\hat{\mathbf{x}}(t_0) = \mathbf{x}(0)$ and recursively produce

$$\hat{\mathbf{x}}(t_n) = \hat{\mathbf{x}}(t_{n-1}) + \hat{\mathbf{Y}}_{n-1}(0), \quad \text{for } n = 1, ..., N_T.$$

3 Simulation Study

In this section, we numerically evaluate our generative model using training paths that are simulated from the underlying SDEs [4]. We test our method on two distinct tasks: generating paths from a one-dimensional Ornstein-Uhlenbeck (OU) process and a 100-dimensional geometric Brownian motion (GBM). We benchmark our model against two alternative ones: the first is also an autoregressive generative model [12] built on score-based diffusion models but with a Monte Carlo score estimator (which we term SDM-MC). The second is based on the Neural SDE method proposed in [15]. Due to space limitation, here we only present results with representative performance metrics.

Example 1. The one-dimensional OU process is governed by

$$\mathrm{d}X(t) = \theta(\mu - X(t))\mathrm{d}t + \sigma\mathrm{d}W(t),\ X(0) = x(0). \tag{6}$$

We set $x(0) = 1.5, \mu = 1.2, \theta = 1, \sigma = 0.3$, and $T = 1, \Delta t = 0.05$, resulting in $N_T = 20$ observation slots in total, and there are $H = 100$ real OU trajectories in the training dataset (2).

Table 1 presents the KL-divergence between the joint distributions of discrete observations of 100 real and synthetic OU trajectories, estimated over 5,000 repeated experiments using the method from [20]. The results clearly show that our model produces paths with a distribution much closer to the target OU process, significantly outperforming both the SDM-MC and Neural SDE benchmarks.

Method	Ours	SDM-MC	Neural SDE
KL divergence ↓	**0.1528 ± 0.009**	0.3578 ± 0.010	0.6423 ± 0.013

Table 1: KL divergence between distributions of 100 real and synthetic OU paths.

Example 2. The multivariate GBM in $\mathbb{R}^d$ with $d = 100$ is defined as:

$$\mathrm{d}\mathbf{X}_j(t) = \mu_j\mathbf{X}_j(t)\mathrm{d}t + \sigma_j\mathbf{X}_j(t)\mathrm{d}\mathbf{W}_j(t),\ \mathbf{X}_j(0) = x_j(0), \quad j = 1, \cdots, 100, \tag{7}$$

where $\mathbf{W}_j$ represents the one-dimensional standard Brownian motion, and $\mathbf{W}_j(t)$ and $\mathbf{W}_l(t)$ have a correlation $\rho_{jl} \in [-1, 1]$ for $j, l = 1, ..., 100$. We randomly generate $\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\rho}$, and set $T = 7$ and $\Delta t = 1$. We use $H = 10,000$ real paths to train diffusion models.

Figure 1 illustrates that the mean and standard deviation of our synthetic paths closely track the training trajectories across multiple dimensions. The KL divergence between distributions of 10000 real and synthetic paths (700-dimensional) is 38.73. This performance is in sharp contrast to the SDM-MC and Neural SDE benchmarks, both of which have KL divergence over 1,000 and fail to even maintain positivity in over 90% of sample trajectories.

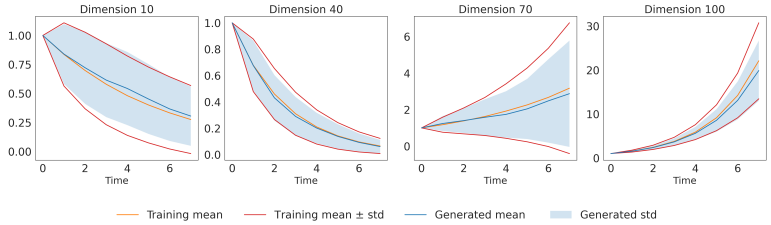

Figure 1: Mean and standard deviation evolution of training and synthetic GBM paths.

4 Empirical Study

We conduct an empirical study to demonstrate how our generative approach can be useful in portfolio optimization, despite the fact that real financial time series may not follow SDEs.

We consider the continuous-time mean–variance portfolio selection problem [19]. An agent invests in the S&P 500 index and a riskless asset with interest rate $r = 2\%$ for half a year ($T = 0.5$). She rebalances her portfolio with a strategy $a = \{a(t), t \in [0, T]\}$ in a self-financing fashion, where $a(t)$ is the dollar invested in the index at time t. The problem is to find a strategy a that minimizes the variance of the terminal wealth (denoted by $X^a(T)$) for a given target wealth level z, i.e.,

$$\min_a \text{Var}(X^a(T)), \ s.t. \ \mathbb{E}[X^a(T)] = z. \tag{8}$$

A classical model-based solution to (8) is to use historical data of S&P 500 to estimate a GBM model and then plug in the estimators into the expression of the optimal deterministic policy of (8) which is known from [24]. This approach (called *plug-in policy*) has been shown to be inferior compared with the recent model-free *RL approach* proposed by [9]. A crucial component of the RL approach in [9] is a market/environment simulator, and we show how synthetic index paths can help further improve the RL strategies by enriching this simulator.

We split daily observations of S&P 500 data from 1990-2009 into 40 half-year trajectories (called split paths). These paths are used to train our generative model. As in the SDE setting, we simulate the daily increment of S&P 500, and autoregressively generate a synthetic index path for half a year which incidentally does not require the GBM assumption. We then augment these synthetic paths to the 40 split paths originating from the data 1990-2009 and develop investment policies. Finally, we test the developed policies on S&P 500 data from 2010 to 2019.

Specifically, we test the *plug-in policy* when only the 40 split paths are used and when the 40 split and 40 synthetic paths are used to estimate the parameters of GBM. We also test the *RL policy* in [9] with three different simulators: the first one is built on the 40 split paths, the second one is based on 40 bootstrap paths as in [9], and the last one is based on the 40 split and 40 synthetic paths. The results are summarized in Table 2, which suggest that including synthetic paths fails to improve the plug-in policy. On the other hand, introducing synthetic paths to the market simulator provides a significant performance boost to the original RL policy which already decisively outperforms the plug-in policy. Indeed, when the synthetic paths are added, the RL agent achieves the highest Sharpe ratios, primarily due to reductions of the variance of the terminal wealth. Intuitively, the synthetic paths enable the RL agent to explore more possible market scenarios and learn a more robust policy.

Target Level	Policy	SDE Paths	Mean	Variance	Sharpe Ratio ↑
$z = 1.10$	Plug-in	Split	1.1610	0.3501	0.2722
		Split + Synthetic	1.1652	0.3704	0.2715
	RL	Split	1.0857	0.0264	0.5273
		Bootstrap	1.0884	0.0264	0.5438
		Split + Synthetic	1.0860	0.0225	**0.5729**
$z = 1.20$	Plug-in	Split	1.3221	1.4004	0.2722
		Split + Synthetic	1.3371	1.4385	0.2810
	RL	Split	1.1745	0.1120	0.5215
		Bootstrap	1.1768	0.1061	0.5427
		Split + Synthetic	1.1704	0.0892	**0.5707**

Table 2: Performance of different portfolio strategies with and without synthetic paths.

References

[1] Minshuo Chen, Renyuan Xu, Yumin Xu, and Ruixun Zhang. Diffusion factor models: Generating high-dimensional returns with factor structure. *arXiv preprint arXiv:2504.06566*, 2025.

[2] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[3] Hengyu Fu, Zhuoran Yang, Mengdi Wang, and Minshuo Chen. Unveil conditional diffusion models with classifier-free guidance: A sharp statistical theory. *arXiv preprint arXiv:2403.11968*, 2024.

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

[5] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[7] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[8] Stefano M Iacus et al. *Simulation and inference for stochastic differential equations: with R examples*, volume 486. Springer, 2008.

[9] Yanwei Jia and Xun Yu Zhou. q-learning in continuous time. *Journal of Machine Learning Research*, 24(161):1–61, 2023.

[10] Mathieu Kessler, Alexander Lindner, and Michael Sørensen. Statistical methods for stochastic differential equations. *Monographs on Statistics and Applied Probability*, 124:7–12, 2012.

[11] Peter E Kloeden and Eckhard Platen. Numerical solution of stochastic differential equations. 1999.

[12] Yanfang Liu, Yuan Chen, Dongbin Xiu, and Guannan Zhang. A training-free conditional diffusion model for learning stochastic dynamical systems. *arXiv preprint arXiv:2410.03108*, 2024.

[13] Sai Shankar Narasimhan, Shubhankar Agarwal, Oguzhan Akcin, Sujay Sanghavi, and Sandeep P Chinchali. Time weaver: A conditional time series generation model. In *International Conference on Machine Learning*, pages 37293–37320. PMLR, 2024.

[14] Hao Ni, Lukasz Szpruch, Magnus Wiese, Shujian Liao, and Baoren Xiao. Conditional sig-wasserstein gans for time series generation. *Mathematical Finance*, 2023.

[15] 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*.

[16] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[17] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[18] Jorino van Rhijn, Cornelis W Oosterlee, Lech A Grzelak, and Shuaiqiang Liu. Monte carlo simulation of sdes using gans. *Japan Journal of Industrial and Applied Mathematics*, 40(3):1359–1390, 2023.

[19] Haoran Wang and Xun Yu Zhou. Continuous-time mean-variance portfolio selection: A reinforcement learning framework. *Mathematical Finance*, 30(4):1273–1308, 2020.

[20] Qing Wang, Sanjeev R Kulkarni, and Sergio Verdú. Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Transactions on Information Theory*, 55(5):2392–2405, 2009.

[21] Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant GANs: deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, 2020.

[22] Minglei Yang, Pengjun Wang, Diego del Castillo-Negrete, Yanzhao Cao, and Guannan Zhang. A pseudoreversible normalizing flow for stochastic dynamical systems with various initial distributions. *SIAM Journal on Scientific Computing*, 46(4):C508–C533, 2024.

[23] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019.

[24] Xun Yu Zhou and Duan Li. Continuous-time mean-variance portfolio selection: A stochastic LQ framework. *Applied Mathematics & Optimization*, 42(1):19–33, 2000.

[25] Aiqing Zhu and Qianxiao Li. Dyngma: a robust approach for learning stochastic differential equations from data. *Journal of Computational Physics*, page 113200, 2024.