SIAM J. MATH. DATA SCI.
Vol. 7, No. 4, pp. 1541–1567

Generative Modeling of Lévy Area for High Order SDE Simulation*

Andraž Jelinčič†, Jiajie Tao‡, William F. Turner§, Thomas Cass§¶, James Foster†, and Hao Ni‡

Abstract. It is well known that when numerically simulating solutions to stochastic differential equations (SDEs), achieving a strong convergence rate better than $O(\sqrt{h})$ (where h is the step-size) usually requires the use of certain iterated integrals of Brownian motion, commonly referred to as its "Lévy areas," However, these stochastic integrals are difficult to simulate due to their non-Gaussian nature. and for a d-dimensional Brownian motion with $d > 2$, no fast almost-exact sampling algorithm is known. In this paper, we propose LévyGAN, a deep-learning-based model for generating approximate samples of Lévy area conditional on a Brownian increment. Due to our "bridge-flipping" operation, the output samples match all joint and conditional odd moments exactly. Our generator employs a tailored graph neural network (GNN)-inspired architecture, which enforces the correct dependency structure between the output distribution and the conditioning variable. Furthermore, we incorporate a mathematically principled characteristic-function-based discriminator. Lastly, we introduce a novel training mechanism, termed "Chen-training," which circumvents the need for expensive-to-generate training data-sets. This new training procedure is underpinned by our two main theoretical results. For four-dimensional Brownian motion, we show that LévyGAN exhibits state-of-the-art performance across several metrics which measure both the joint and marginal distributions. We conclude with a numerical experiment on the log-Heston model, a popular SDE in mathematical finance, demonstrating that a high-quality synthetic Lévy area can lead to high order weak convergence and variance reduction when using multilevel Monte Carlo (MLMC).

Key words. generative modeling, Lévy area, adversarial learning, probability theory, stochastic analysis, rough path theory, numerical approximation

*Received by the editors November 3, 2023; accepted for publication (in revised form) April 11, 2025; published electronically October 3, 2025.

https://doi.org/10.1137/23M161077X

Funding: The first author was supported by the Public Scholarship, Development, Disability and Maintenance Fund of the Republic of Slovenia during his undergraduate and master's study through the Ad Futura scholarship. The third author was supported by the EPSRC Centre for Doctoral Training in Mathematics of Random Systems: Analysis, Modelling and Simulation (EP/S023925/1). The fourth author acknowledges the support of the Erik Ellentuck Fellowship at the Institute of Advanced Study, Princeton. The fifth author was supported by the Department of Mathematical Sciences at the University of Bath, the Maths4DL programme under EPSRC grant EP/V026259/1 and The Alan Turing Institute. The sixth author was supported by The Alan Turing Institute under the EPSRC grant EP/N5101 and by the Ecosystem Leadership Award under the EPSRC grant OobfJ22\100020. The fourth, fifth, and sixth authors were also supported by the EPSRC Programme grant "DataSig" EP/S026347/1.

†Department of Mathematical Sciences, University of Bath, Bath BA2 7AY, UK (aj2382@bath.ac.uk, jmf68@bath.ac.uk).

‡Department of Mathematics, University College London, London WC1E 6BT, UK (jiajie.tao.21@ucl.ac.uk, h.ni@ucl.ac.uk).

§Department of Mathematics, Imperial College London, London SW7 2AZ, UK (william.turner17@imperial.ac.uk, thomas.cass@imperial.ac.uk).

¶Institute for Advanced Study, Princeton, NJ 18540 USA.

MSC codes. 65C30, 60H35, 68T99

DOI. 10.1137/23M161077X

1. Introduction.

The numerical simulation of stochastic differential equations (SDEs) is a ubiquitous task encountered in a wide variety of fields, ranging from mathematical finance [42] and systems biology [1] to molecular dynamics [26] and data science [29]. Real-world phenomena arising in these areas are often described well by SDEs formulated through *Itô calculus* and of the general form

$$dX_t = f(X_t)dt + \sum_{i=1}^{d} g_i(X_t)dW_t^{(i)}, \quad X_0 = x_0, \tag{1.1}$$

where the solution $X = \{X_t\}_{t\in[0,T]}$ takes values in $\mathbb{R}^e$, $W = (W^{(1)},\dots,W^{(d)})$ denotes a standard d-dimensional Brownian motion, and $f, g_i : \mathbb{R}^e \to \mathbb{R}^e$ are suitably regular vector fields on $\mathbb{R}^e$. In practice, one is often concerned with estimating quantities of the form $\bar{\varphi} := \mathbb{E}[\varphi(X) \,|\, X_0 = x_0]$, where the function φ may depend on the whole trajectory of X or simply on the terminal value X_T, such as the payoff of a European call-option. On occasion, it may be possible to obtain $\bar{\varphi}$ by solving certain partial differential equations (PDEs), e.g., through the backward Kolmogorov equation or the Feynman–Kac formula.

However, the standard approach is to use Monte Carlo simulation, where one uses a discretization scheme to generate approximate sample paths $\{\widehat{X}_i\}_{i=1}^N$ of X, which can be used to approximate $\bar{\varphi}$ by taking the average of $\{\varphi(\widehat{X}_i)\}_{i=1}^N$. Given the importance of Monte Carlo simulation in applications, there is a rich literature concerning numerical methods for SDEs and their theoretical properties. A broad range of discretization schemes are available, such as the classical Euler–Maruyama and Milstein schemes as well as the higher order Talay [44] and Ninomiya–Victoir [38] schemes. For more details on the numerical simulation of SDEs, we refer the reader to [24, 39].

There are two standard measures for the effectiveness of numerical schemes: strong error (or mean squared error-MSE) and weak error. It is a well-known result of Clark and Cameron [4] that numerical schemes using only increments of Brownian motion are limited in general to a strong convergence rate of at most $O(\sqrt{h})$, where h denotes the step- size. Furthermore, to the best of our knowledge, all numerical schemes achieving second order weak convergence require the generation of random variables in addition to the Brownian increments. In particular, the Talay scheme [44] and the Ninomiya–Victoir scheme [38] require the generation of Rademacher random variables. Further examples include the Ninomiya–Ninomiya scheme [37] and a stochastic Runge–Kutta method due to Rößler [40]. In all of these cases, the additional random variables are generated to replace certain second order iterated integrals of Brownian motion, which are collectively referred to as its *Lévy area;* see Figure 1 for a visual interpretation of Lévy area.

Definition 1.1. *The Lévy area of a d-dimensional Brownian motion over $[s,t]$ is a $d \times d$ antisymmetric matrix whose (i,j)th entry is entries given by*

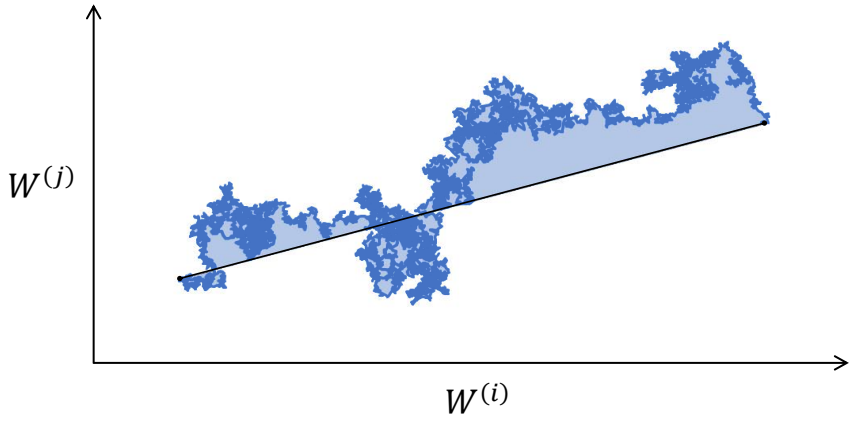


Figure 1. *Each entry $A^{(i,j)}$ is the area between the independent Brownian motions $W^{(i)}$ and $W^{(j)}$ (diagram adapted from [10]).*

$$A_{s,t}^{(i,j)} := \frac{1}{2}\left(\int_s^t (W_r^{(i)} - W_s^{(i)})\, dW_r^{(j)} - \int_s^t (W_r^{(j)} - W_s^{(j)})\, dW_r^{(i)} \right).$$

When the vector fields of the SDE (1.1) do not satisfy the commutativity condition $[g_i, g_j] = 0$ (where $[g_i, g_j](x) := g_j'(x)g_i(x) - g_i'(x)g_j(x)$ denotes the standard Lie bracket of vector fields), schemes that achieve high order Strong convergence, such as the Milstein and log-ODE methods, require the simulation of Lévy area. Consequently, the approximation of Lévy area has received much interest in recent decades, with the view towards both high order weak and strong convergence.

Approximations to Lévy area have been well studied [6, 7, 14, 24, 25, 10, 12, 35, 47], with the majority of approximations concerning strong estimation. Strong estimators aim to approximate Lévy area by minimizing the mean-squared error so that the resulting estimator may be incorporated into the strong analysis of the discretization scheme. Typically, such estimators rely on truncated expansions of Brownian motion, such as the Fourier series expansion [24, 25, 20], the Karhunen–Loève expansion [30], and, more recently, the polynomial expansion [12, 13]. These estimators are often improved by estimating the tail sum of the expansion in an appropriate manner; see, for example, [35, 47].

To the best of our knowledge, there is no known scheme which simulates Lévy area exactly, with even the "rectangle-wedge-tail" algorithm of Gaines and Lyons [14] requiring numerical integration. Moreover, this approach is only applicable in $d = 2$. On the other hand, the main drawback of the truncated expansion methods is the cost of simulation. In practice, one is often required to generate millions of Lévy area samples, and the aforementioned methods often require a high truncation level to achieve good performance. Consequently, in recent years, there has been a renewed focus on approximations of Lévy area that are suitable for weak discretization schemes, where the estimator is less costly to generate. The aim of a weak estimator is to match some moments of the Lévy area given a Brownian increment. These estimators differ in complexity depending on their intended usage. Basic estimators include the Rademacher random variables that appear in the Talay scheme [44] and Davie's approximation [6, 9] which uses a Gaussian random variable with the correct variance

(this may be improved to have the correct conditional variance given the Brownian increment [10]). Perhaps the most sophisticated weak approximation is the one due to Foster [10] that matches the first five conditional moments of Lévy area given the Brownian increment when $d \leq 3$.

In this article, we provide a new approach to the construction of weak estimators of Lévy area given a Brownian increment through the use of generative modeling techniques. To the best of our knowledge, this is the first time that the powerful toolkits provided by modern machine learning have been applied to the problem of Lévy area simulation. Arguably, the main obstacle to this approach is the computational effort required to generate considerable amounts of precise samples of Lévy area. We note this is, in principle, possible through the use of a truncated Fourier series method [20], with other options also possible. However, in the context of Lévy area generation, we present a novel training algorithm based on Chen's relation [2] that allows for the training of a generative model without access to a data-set of Lévy area samples.

1.1. Our contributions. In this article we present LévyGAN, a deep-learning-based generative model that simulates the Lévy area of arbitrary dimensional Brownian motions. Deep-learning- based generative models have been widely used for data synthesis, where a parametric model is trained to learn the target distribution. Among the variety of generative models, GAN-type models [34, 18] have been particularly noteworthy for their performances. GAN, short for generative adversarial network, operates on the compelling principle of adversarial training. This approach consists of two neural networks—a generator and a discriminator—that are trained simultaneously. The generator's task is to create synthetic data, while the discriminator's role is to distinguish between real and generated data.

With no exception, GAN-type models also possess drawbacks like other generative models. The necessity of real data as reference sets for training is one of them. Machine learning models are often data-driven and sometimes data-greedy; normally, practitioners collect real-world data and approximate its distribution using an empirical distribution outputted by the generator. This is especially pertinent in the context of Lévy area generation, where a competitive method needs to achieve very high accuracy, which incurs a high statistical complexity, and thus requires large amounts of data. In contrast to standard GANs, score-based diffusion models, and variational autoencoders, LévyGAN is designed to learn the target distribution without requiring any samples from it. We term this approach "Chen-training,"because it is theoretically underpinned by the unique invariance of the joint law of Brownian motion and Lévy area under Chen's relation; see Theorem 4.5.

We have designed both the generator and discriminator by exploring the features of the joint law of Brownian motion and its Lévy area [12, 13, 10]. In particular, for the generator, we ensure that the joint distribution of our generator is permutation invariant and that each component of the generated area depends only on the relevant components of the Brownian increment. We also ensure that all odd cross moments of our generator are exact through the multiplication by certain Rademacher random variables. For the discriminator, we have chosen a characteristic-function-based discriminator, initially proposed in [8, 27]. Inspired by this approach and the more general method of [31], we define the unitary characteristic function of a random variable as a generalization of the characteristic function onto higher degree Lie algebra.

For our numerical results, we train the LévyGAN in $d = 4$. It is noteworthy that the model is able to generate the Lévy area for arbitrary Brownian dimensions, with no loss in performance guaranteed in $d' < d$ and empirically strong performance for $d' > d$. Empirically, we show that LévyGAN attained the best performance among other weak estimators, such as those found in [6, 10], in terms of distributional metrics, and is comparable in generation speed to the method in [10]. Finally, we provide an application of weak approximations to Lévy area to high order multilevel Monte Carlo numerical schemes. In this example, we demonstrate that the inclusion of an approximate Lévy area term in the Strang splitting method achieves higher order variance reduction and weak convergence. Moreover, we provide evidence that an estimator that only matches the variance of Lévy area (such as the Rademacher random variables found in the Talay scheme) is not appropriate for this application. The LévyGAN implementation, together with a trained model for $d = 4$, can be found at https://github.com/andyElking/LevyGAN.

1.2. Outline and common notation. This article is divided into five main sections. In section 2 we recall the standard setup of generative adversarial networks and discuss why the traditional approach to generative modeling is not easily applicable to our setting. In section 3 we outline the structure of our generator. This section focuses on the symmetries of the joint law of Brownian motion and Lévy area that we hard code into our generator. This includes so-called "bridge-flipping," a precise multiplication by certain Rademacher random variables to ensure all joint odd moments are correctly estimated and to help the generator train evenly across all quadrants in space. In subsection 3.2 we introduce a network architecture dubbed "pair-net" inspired by graph neural networks that ensures the correct dependence structure between the coordinates of Brownian motion and the coordinates of Lévy area. The structure of our discriminator is outlined in section 4. Here we discuss two alternatives for a loss function based on the analytical form of the joint characteristic function of Lévy area and Brownian motion and a generalization termed the "unitary characteristic function" proposed in [31]. Our novel training approach, "Chen-training," is covered in subsection 4.2 before the whole training procedure is summarized in section 5. Finally, in section 6 we compare the distributional performance of our generator to the state-of-the-art Foster method [10] and demonstrate the applicability of weak estimators for Lévy area to multilevel Monte Carlo. To conclude this introduction, we outline as follows some common notation to be used throughout the article:

(1) $\mathbb{P}_{X|Y=y}$ stands for the conditional distribution of X given $Y = y$.
(2) With abuse of notation, $(X \mid Y = y) \overset{d}{=} (Z \mid Y = y)$ if $\mathbb{P}_{X|Y=y} = \mathbb{P}_{Z|Y=y}$.
(3) $(W_t)_{t\in[0,1]}$ stands for a d-dimensional Brownian motion, and the process $(A_t)_{t\in[0,1]}$ with $A_t = \{A_t^{(i,j)}\}_{1\le i<j\le d}$ a $\frac{d(d-1)}{2}$-dimensional vector representing the flattened upper triangle of the Lévy area matrix associated with the Brownian motion. Unless stated otherwise, we denote by a the dimension of Lévy area, i.e., $a = \frac{d(d-1)}{2}$.
(4) For any process $(X_t)_{t\in[0,1]}$, X_t denotes the process evaluated at time t.
(5) $\mathbb{P}_{(W_t,A_t)}$ stands for the joint law of Brownian motion and Lévy area at time t.
(6) $\mathcal{N}^d(\mu, \sigma^2)$ stands for a d-dimensional Gaussian distribution with independent coordinates, each with mean μ and variance σ^2.

(7) $\mathrm{Rad}^d(p)$ stands for a d-dimensional Rademacher random variable with independent entries, each of which takes 1 and -1 with probability p and $1-p$, respectively.
(8) Φ_X stands for the characteristic function of a random variable X.
(9) $W_{s,t} := W_t - W_s$ stands for the increments of a Brownian motion $(W_t)_{t\in[0,1]}$, and we use $W_t = W_{0,t}$ interchangeably.
(10) For any *tensor,* i.e., an element $\mathbf{x} \in \mathbb{R}^{\cdot\times d}$, we always denote by $x^{(i)} \in \mathbb{R}^{\cdot}$ the ith coordinate of the second dimension.

2. The GAN architecture. The goal of this article is to build an efficient and accurate estimator that approximates the conditional law $\mathbb{P}_{A_t|W_t}$, and hence the joint law, $\mathbb{P}_{(W_t,A_t)}$, of a Brownian increment and its Lévy area. Thanks to the scaling property of Brownian motion, it is enough to consider the problem when $t = 1$.

We adopt a conditional GAN (generative adversarial network) approach as proposed by [34]. GAN-type models, initially proposed by [18], consist of a pair of competing neural networks—the generator and the discriminator. The aim of the generator is to create "fake" data $\widetilde{\mathbf{x}}$ from some noise distribution, trying to mimic a target distribution, while the discriminator will be given both $\widetilde{\mathbf{x}}$ and data $\mathbf{x}$ from the true distribution and will try to distinguish the ground truth between them. The dynamics between the generator and discriminator are controlled by a min-max game acting on a loss function, which usually represents the distance between two distributions. In a conditional GAN, the generator is given samples from not only the noise distribution but also the conditioning variable—in our case that will be the Brownian increment W or, later, the space-time Lévy area H (see subsection 3.1). Here we provide the description of a classical conditional GAN adapted to our interest.

Definition 2.1 (classical conditional GAN for Lévy area generation). *Let $d \geq 2$ be the Brownian dimension, and le t $a := \frac{d(d-1)}{2}$ be the dimension of the associated Lévy area vector. Assume z is an n-dimensional noise vector distributed according to $\mathbb{P}_z$. The conditional generator G_θ and the discriminator D_η are maps*

$$G_\theta : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}^d \times \mathbb{R}^a \quad D_\eta : \mathbb{R}^d \times \mathbb{R}^a \to \mathbb{R}$$

that are parameterized by θ and η, respectively. When restricted to the first d-coordinates of the domain and codomain, G_θ is enforced to be identity, i.e., $G_\theta(w,z) = (w, \widetilde{A})$. Let $\mathbb{P}_{(W_1,A_1)}$ be the ground truth distribution of the coupled process. An example loss function might be given by

$$L(\theta,\eta) := \mathbb{E}_{(w,A)\sim\mathbb{P}_{(W_1,A_1)}}\left[D_\eta(w,A)\right] - \mathbb{E}_{w\sim\mathbb{P}_{W_1},z\sim\mathbb{P}_z}\left[D_\eta(G_\theta(w,z))\right],$$

where one restricts D_η to be at most 1-Lipschitz. The models are trained using the min-max game $\min_\theta \max_\eta L(\theta,\eta)$ until convergence. The generator obtained is then used to simulate Lévy areas.

Since there are no known methods for exact simulation of Lévy areas in $d > 2$, the "true" Lévy area samples $\mathbf{A} \sim \mathbb{P}_{A_1|W_1}$ must themselves be obtained through approximate simulation. In particular, we might generate the "true" Lévy area samples using the Julia-language package created by Kastner and Rössler [20], which complements their research on approximate

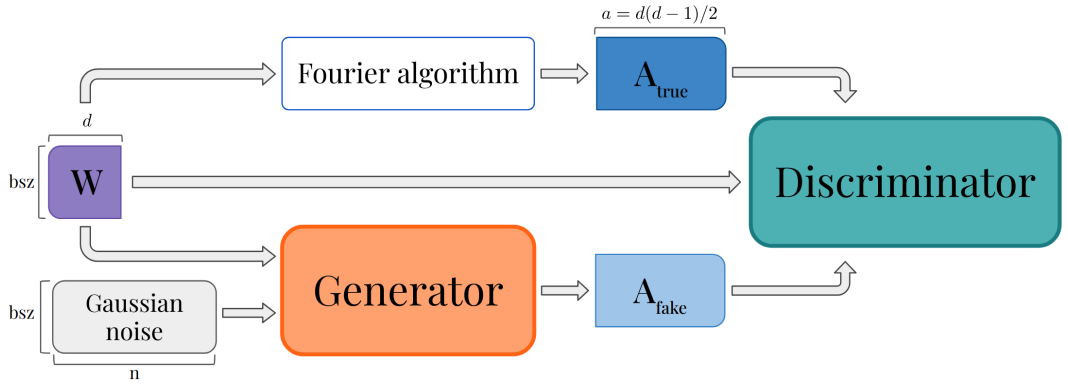


Figure 2. *A schematic of Lévy generation for classical conditional GAN. Throughout this article,* `bsz` *represents the training batch size.*

strong simulation of Lévy area through truncated Fourier series methods. We also present the flowchart of this methodology in Figure 2.

The main drawback of this methodology is the need for simulating approximate real samples of Lévy area. Using real data not only slows the training procedure but also introduces simulation error (due to the truncation in the Fourier series) and finite sample error (leading to overfitting). In order to address this problem, we propose a novel approach, LévyGAN, which completely excludes real samples from the training process and is well justified by our two main theoretical contributions, Theorem 4.5 in the main text and Theorem SM2.4 in the supplementary material.

3. Generator. In order to improve the accuracy of the generated distribution, we can consider the symmetries of the Lévy area distribution and hard code them into the generator itself. This way, the generator will consist of both a neural net and additional operations applied to the network's output. One symmetry of Lévy area that is desirable to reflect is the fact that its distribution is mean zero when conditioned on any increment $W_{0,1} = w$. We do this through two operations which we combine and term "bridge-flipping."

3.1. Bridge-flipping. We would like to hard code the symmetry of Lévy area about zero into the generator's architecture. Even though each dimension of Lévy area $A^{(i,j)}$ is symmetric about zero, its joint distribution is not invariant to multiplying any individual dimension by -1. The dimensions of the underlying Brownian motion, however, are independent and hence can be mirrored separately without violating their joint law.

The goal is hence to find a symmetry of Lévy area corresponding to independently flipping individual dimensions of the Brownian motion. Notice, however, that we are trying to generate Lévy area conditional on a fixed input $W_{0,1} = w$, and so we do not wish to flip the increment of Brownian motion itself. We can circumvent this issue by considering the polynomial expansion of Lévy area [12], which decomposes the Brownian motion into the components dependent on w and components independent of w, the latter of which can then be flipped independently. To this end, we first define the Brownian bridge and its accompanying "space-time" and "space-space" Lévy areas.

Definition 3.1 (Brownian bridge). *Let $0 \le s \le t < \infty$. Then the Brownian bridge of W on $[s,t]$ returning to zero at time t is defined as*

$$B_{s,u} := W_{s,u} - \frac{u-s}{t-s} W_{s,t} \quad \text{for } u \in [s,t].$$

The "space-time" Lévy area $H_{s,t} \in \mathbb{R}^d$ and "space-space" Lévy area $b_{s,t} \in \mathbb{R}^{d \times d}$ of B over $[s,t]$ are

$$H_{s,t}^{(i)} := \frac{1}{t-s} \int_s^t B_{s,u}^{(i)} \, du = \frac{1}{t-s} \int_s^t W_{s,u}^{(i)} - \frac{u-s}{t-s} W_{s,t}^{(i)} \, du \quad \text{for } 1 \le i \le d, \tag{3.1}$$

$$b_{s,t}^{(i,j)} := \int_s^t B_{s,u}^{(i)} \circ dB_u^{(j)} \quad \text{for } 1 \le i,j \le d. \tag{3.2}$$

Whenever $s = t$, we define $b_{s,t} = H_{s,t} = 0$. We write H_t and b_t for $H_{0,t}$ and $b_{0,t}$, respectively.

It turns out that (H,b) and $W_{s,t}$ are independent, the marginal distribution of H is Gaussian, and the marginal of b is logistic.

Proposition 3.2 (distribution of Brownian bridge Lévy area [13, 10]). *For fixed $0 \le s < t < \infty$, the process $\{(H_{s,u}, b_{s,u})\}_{u \in [s,t]}$ and the increment $W_{s,t}$ are independent. Furthermore, H is distributed as a d-dimensional Gaussian with independent coordinates, and the marginal distribution of each Brownian bridge Lévy area is logistic:*

$$H_{s,t} \sim \mathcal{N}^d\left(0, \frac{1}{12}(t-s)\right) \quad \text{and} \quad b_{s,t}^{(i,j)} \sim \text{Logistic}\left(0, \frac{1}{2\pi}(t-s)\right).$$

This yields the first two terms of the polynomial expansion of Lévy area.

Proposition 3.3 (polynomial expansion of Lévy area [12]). *The Lévy area of a d-dimensional Brownian motion W has the following decomposition:*

$$A_{s,t} = H_{s,t} \otimes W_{s,t} - W_{s,t} \otimes H_{s,t} + b_{s,t},$$

where $\otimes$ denotes the outer-product of vectors.

This decomposition reduces the conditional generative task to the estimation of the Brownian bridge Lévy area conditional on the Brownian increment, where the target distribution and conditioning variable are independent. This approach may be generalized to the estimation of the tail sum of the polynomial expansion of Lévy area truncated at a higher level. Since B and the increment $W_{0,1} = w$ are independent, the conditional distribution $(B_t \mid W_{0,1} = w)$ is symmetric around 0. Furthermore, each dimension of B is its own independent process, so we can flip each individually without affecting the distribution, as established in the following lemma.

Lemma 3.4. *Let W be a d-dimensional Brownian motion on $[0,1]$, and let B, H, b be the corresponding derived processes from Definition 3.1. Fix some $\xi \in \{-1,1\}^d$, and let H', and b' be the space-time and space-space Lévy area processes associated with the process $\xi \odot B = \{\xi \odot B_t\}_{t \in [0,t]}$, where $\odot$ denotes the Hadamard (coordinatewise) product. Then*

$$\{B_t\}_{t \in [0,1]} \overset{d}{=} \{\xi \odot B_t\}_{t \in [0,1]}, \quad H' = \xi \odot H, \quad b' = (\xi \otimes \xi) \odot b, \quad \text{and} \quad (H', b') \overset{d}{=} (H, b).$$

We also include a multiplication of a final independent random variable $\xi_0 \sim \mathrm{Rad}(\frac{1}{2})$, whose role is explained in Proposition SM3.3 of the supplementary material. Combining this with Lemma 3.4, we obtain the "bridge-flipping" function.

Definition 3.5. *Let $w, h, \xi \in \mathbb{R}^d$, $b \in \mathbb{R}^{d\times d}$, and $\xi_0 \in \mathbb{R}$. Then the bridge-flipping function is defined as*

$$\mathrm{BF}(w, h, b, \xi_0, \xi) := \xi_0 \left((\xi \odot h) \otimes w - w \otimes (\xi \odot h) + (\xi \otimes \xi) \odot b\right). \tag{3.3}$$

For this function we have the following as a consequence of Proposition 3.3 and Lemma 3.4.

Theorem 3.6 (bridge-flipping). *Let $\xi_0, \dots, \xi_d \overset{i.i.d.}{\sim} \mathrm{Rad}(\frac{1}{2})$ be random variables so that W, (H, b), and $(\xi_0, \dots, \xi_d)$ are independent. Write $\xi = (\xi_1, \dots, \xi_d)$, and fix some $w \in \mathbb{R}^d$. Let H, b be as in Definition 3.1. Then for every $t \in [0, 1]$,*

$$(A_{0,t} \mid W_{0,1} = w) \overset{d}{=} (\mathrm{BF}(W_{0,t}, H_{0,t}, b_{0,t}, \xi_0, \xi) \mid W_{0,1} = w).$$

For our purpose, we will utilize the result for $t = 1$,

$$(A_{0,1} \mid W_{0,1} = w) \overset{d}{=} \mathrm{BF}(w, H_{0,1}, b_{0,1}, \xi_0, \xi).$$

Recall that $H_{s,t} \sim \mathcal{N}^d\big(0, \frac{1}{12}(t-s)\big)$ and that b and H are correlated, but that (H, b) and $W_{0,1}$ are independent. Hence, given a neural net $\mathrm{NN}_\theta : \mathbb{R}^{d+n} \to \mathbb{R}^d$, we define the "Bridge-flipping generation":

This construction has the following desirable properties:

(1) Informally speaking, the use of ξ effectively makes the generator behave identically on all orthants of $\mathbb{R}^d$, and hence any learning done in one orthant transfers equally to the other orthants. This speeds up training and significantly improves the generator's accuracy, as it now perfectly mimics the symmetric structure of Lévy area.

(2) The neural net can be trained directly on the distribution of $(b_{0,1} \mid H_{0,1} = h)$ and is then used for generation of $(A_{0,1} | W_{0,1} = w)$ using the BF (bridge-flipping) algorithm.

(3) The structure of BF allows for efficient implementation of back-propagation.

The use of the extra Rademacher random variable ξ_0 is to guarantee that all odd joint and conditional moments are correctly matched. This is summarized in Proposition SM3.3 of the supplementary material. Having $\widetilde{A}$ unbiased means that some of the usual error analysis from stochastic numerics can be applied, such as in the proof of Theorem 4.5, where one of the requirements is that $\widetilde{A}$ be unbiased. Recall that the Milstein scheme requires a subroutine which generates samples of Lévy area given a Brownian increment. Since $\widetilde{A} \sim \mathbb{P}^{\theta,w}_{\mathrm{BF}}$ is unbiased, one can establish theoretical guarantees on the convergence of Milstein's method with the BF generator as this subroutine. The following result can be proven by applying [33, Theorem 1.1] with $p_1 = \frac{3}{2}$, $p_2 = 1$.

Proposition 3.7. *Given a time horizon $T > 0$, and a step size $h = \frac{T}{N}$ where $N \geq 1$ denotes the number of steps, let $\{\widetilde{X}_n\}_{n\in\{0,\dots,N\}}$ be the output of Milstein's scheme (see subsection SM7.2.1 in the supplementary material for the definition) applied to the SDE*

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t,$$

Algorithm 3.1. Area generation using bridge-flipping.

Input: θ - neural net parameters, d - Brownian dimension, w - Brownian increment, n - dimension of noise vector

$\xi_0 \leftarrow \mathrm{Rad}(1/2)$; $\xi \leftarrow \mathrm{Rad}^d(1/2)$

$h \leftarrow \mathcal{N}^d\big(0, \frac{1}{12}(t-s)\big)$; $z \leftarrow \mathcal{N}^n(0,1)$

$\widetilde{b} \leftarrow \mathrm{NN}_\theta(h,z)$ ▷ we want $\mathrm{NN}_\theta(h,z) \overset{d}{\approx} (b_{0,1} \mid H_1 = h)$

return $\mathrm{BF}(w, h, \widetilde{b}, \xi_0, \xi)$

where the Lévy areas provided as input to Milstein's scheme were generated by the BF generator. Then if μ, σ are continuous and globally Lipschitz, there exists a constant $C > 0$ such that for sufficiently small h,

$$\sup_{0 \le n \le N} \mathbb{E}\left[\left|\widetilde{X}_n - X_{nh}\right|^2\right]^{\frac{1}{2}} \le C h^{\frac{1}{2}}.$$

Remark 3.8. Although Theorem 3.6 and Algorithm 3.1 lead to the generation of $(A_{0,1}| W_{0,1} = w)$, we emphasize that they can be generalized to the generation of $(A_{0,t}|W_{0,1} = w)$ for any $t \in [0,1]$ by using the scaling property of Brownian motion and its Lévy area. In particular, given w we can do the following:

(1) Sample w' from the distribution of $(W_t|W_1 = w)$, i.e., $w' \sim \mathcal{N}(t\,w,\, t(1-t))$;
(2) rescale $w'' = \frac{w'}{\sqrt{t}}$ and sample $a'' \sim (A_{0,1} \mid W_1 = w'')$;
(3) finally, rescale again $a' = t\,a''$ which has the desired distribution $a' \sim (A_{0,t} \mid W_1 = w)$.

3.2. The pair-net generator. It is clear that for any $1 \le i, j \le d$, both $b_1^{(i,j)}$ and $A_1^{(i,j)}$ depend only on the paths of $\{W_t^{(i)}\}_{t\in[0,1]}$ and $\{W_t^{(j)}\}_{t\in[0,1]}$ and not on $\{W_t^{(k)}\}_{t\in[0,1]}$ for $k \notin \{i,j\}$. This dependency structure can be well described using graphs, encouraging us to employ model architectures reminiscent of graph neural networks (GNNs) [41].

Consider a clique on d nodes, where each node corresponds to one dimension of the Brownian motion, and each edge (i,j) is associated to $A^{(i,j)}$ (or $b^{(i,j)}$). Unlike GNNs, where it might be desirable for information to propagate throughout the entire graph, we want edge (i,j) to never see information at node $k \notin \{i,j\}$. Hence, our architecture should function like a 1-step GNN with edgewise outputs.

So that $A^{(i,j)}$ only depends on $W^{(i)}$ and $W^{(j)}$, we generate a separate noise vector for each dimension of Brownian motion. We can interpret this as some embedding of the entire path $\{W_t^{(i)}\}_{t\in[0,1]} \mapsto \mathrm{noise}^{(i)}$, but in practice we use Gaussian noise.

Definition 3.9 (pair-net). *The pair-net is defined as the mapping*

$$PairNN_\theta : (\mathbb{R}, \mathcal{Z}) \times (\mathbb{R}, \mathcal{Z}) \to \mathbb{R}; \quad (H, Z) \times (H', Z') \mapsto \widetilde{\mathbf{b}},$$

where $\mathcal{Z}$ is the space of latent noise. Consider $H_{0,1}$ and $b_{0,1}$ from Definition 3.1. We approximate $\mathbb{P}_{b_{0,1}|H_{0,1}=h}$ via

$$\widetilde{\mathbf{b}}^{(i,j)} = PairNN_\theta((h^{(i)}, \mathbf{z}^{(i)}), (h^{(j)}, \mathbf{z}^{(j)})), \quad 1 \le i < j \le d,$$

Algorithm 3.2. Area generation using pair-net generator and bridge-flipping.

Input: θ - neural net parameters, d - dimension of Brownian motion, w - Brownian increment, n - dimension of noise vector associated with each coordinate of Brownian motion.

1: $\xi_0 \leftarrow \mathrm{Rad}(1/2)\,; \quad \xi \leftarrow \mathrm{Rad}^d(1/2)$
2: $\mathbf{z}^{(i)} \leftarrow \mathcal{N}^n(0,1)$ for $1 \le i \le d$; $\quad h \leftarrow \mathcal{N}^d\big(0, \frac{1}{12}(t-s)\big)$
3: $\widetilde{b}^{(i,j)} \leftarrow \mathrm{PairNN}_\theta((h^{(i)}, \mathbf{z}^{(i)}), (h^{(j)}, \mathbf{z}^{(j)}))$ for $1 \le i < j \le d$
4: **return** $\mathrm{BF}\,(w, h, \widetilde{b}, \xi_0, \xi)$

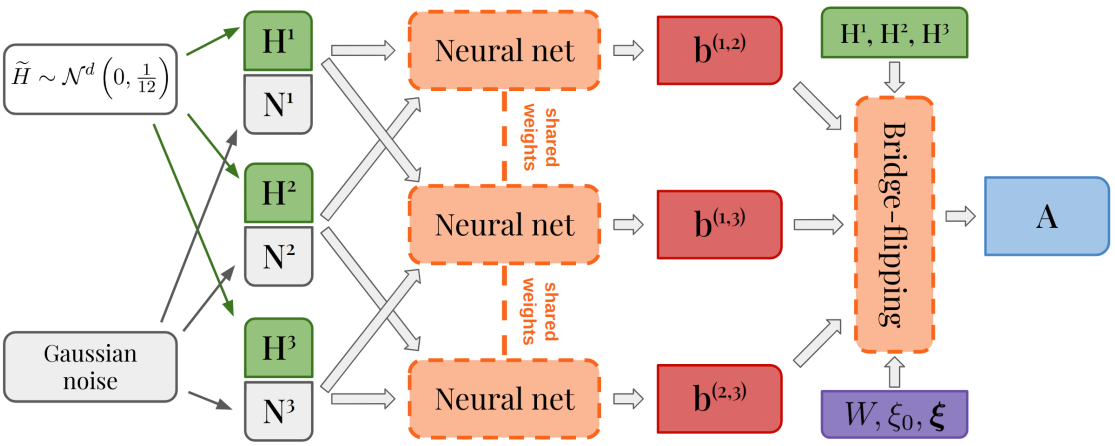


Figure 3. *A schematic of the pair-net architecture when* $d = 3$.

for any $\mathbf{z} \in \mathcal{Z}^d$. *In practice, we choose* $\mathcal{Z}$ *to be* $\mathbb{R}^n$ *and let* $\mathbf{z}^{(i)}$ *be an* n*-dimensional Gaussian noise for* $1 \le i \le d$. *Combining this with bridge-flipping, we describe the generation algorithm and flowchart in Algorithm* 3.2 *and Figure* 3.

Another benefit of the pair-net architecture is that the neural net PairNN_θ can now be significantly smaller since it does not need to capture the relationship between all the dimensions. This comes with the downside of requiring more passes through it. (As shown in Figure 3, we use a single net to generate all b's; each dimension of b requires its own forward pass.) However, suitable indexing and batching can make this efficient. Indeed, using pair-net results in both a speed-up and greater accuracy (see section 6). Furthermore, we emphasize that the pair-net structure allows us to generate Lévy area for any dimension d', even though we train the model in a lower dimension $d < d'$. From now on, we will take nsz to be the total noise dimension for the complete Lévy area generation, namely, $\mathrm{nsz} = n \times d$ where n is the latent noise dimension and d denotes the Brownian dimension. In the framework of Definition 2.1, our generator is now $G_\theta : \mathbb{R}^d \times \mathbb{R}^{\mathrm{nsz}} \to \mathbb{R}^d \times \mathbb{R}^a$ and the evaluation follows Algorithm 3.2.

4. Discriminator. One potential discriminator is the characteristic function GAN (CF-GAN) approach introduced in [8, 27], which aims to learn the law of an underlying process by approximating its characteristic function. We compare this to traditional GANs list some advantages of this approach below:

(1) The characteristic function always exists and is uniformly bounded.
(2) The characteristic function fully describes the law of the random variable, hence offering good theoretical support.

In this section, we introduce two different choices for the characteristic function. First, we assess the distance between two distributions using the characteristic function distance.

Definition 4.1 (characteristic function distance [8]). *Let $\mathbb{P}_X$ and $\mathbb{P}_Y$ be the distributions of two $\mathbb{R}^d$-valued random variables X and Y, respectively. The characteristic function distance (CFD) between X and Y associated with an $\mathbb{R}^d$-valued random variable $\Lambda \sim \nu$ is given by*

$$CFD_\Lambda(X,Y) = \mathbb{E}_{\Lambda\sim\nu}\left[\|\Phi_X(\Lambda) - \Phi_Y(\Lambda)\|\right],$$

where $\Phi_X(\Lambda) := \mathbb{E}_{X\sim\mathbb{P}_X}[\exp i\langle\Lambda, X\rangle]$.

The characteristic function distance has the following properties:
(1) Definiteness: If the support of Λ is $\mathbb{R}^d$, then $\mathrm{CFD}_\Lambda(X,Y) = 0$ iff $\mathbb{P}_X = \mathbb{P}_Y$.
(2) The distance is bounded and differentiable almost everywhere.
(3) For certain ν, $\mathrm{CFD}_\Lambda(X,Y_n) \to 0 \implies Y_n \xrightarrow{d} X$ (see [43] as well as Proposition SM1.2 of the supplementary material).

If we restrict ν to be a member of the family of certain well-known distributions, for example, Gaussian, we can parameterize the distribution ν by learnable coefficients, and it will play the role of discriminator in the GAN setting. The back-propagation on these coefficients is well understood by techniques such as the reparameterization trick. In practice, we approximate the characteristic function by an empirical measure: if $\Lambda \sim \nu$, and $x_1, \ldots, x_N$ are samples from $\mathbb{P}_X$, then we estimate $\Phi_X(\Lambda)$ by

$$\widehat{\Phi}_X(\Lambda) = \frac{1}{N}\sum_{i=1}^{N} \exp\left(i\langle\Lambda, x_i\rangle\right).$$

The use of the empirical characteristic function in place of the analytical characteristic function is justified in Proposition SM1.1 of the supplementary material. However, if $X = (W_t, A_t)$, i.e., the joint process of d-dimensional Brownian motion and the corresponding Lévy area at any time t, we may obtain $\Phi_{(W_t,A_t)}$ analytically; see Theorem SM3.1 in the supplementary material.

4.1. Unitary characteristic function. In this subsection, we introduce an extension of the classical characteristic function of a random variable, originally proposed in [3, 31]. We denote by U_n the set of unitary matrices of dimension n; then U_n is a matrix Lie group with the group operation of matrix multiplication. The Lie algebra of U_n, denoted by $\mathfrak{g}_n$, is the set of anti-Hermitian matrices, i.e., $\mathfrak{g}_n := \{A \in \mathbb{C}^{n\times n} : A^* + A = 0\}$. Next, we give a definition of the unitary representation of a random variable and its unitary characteristic function.

Definition 4.2 (unitary characteristic function). *Let $X \sim \mathbb{P}_X$ be an $\mathbb{R}^d$-valued random variable. Let $n \geq 1$ be an integer. Denote by U_n and $\mathfrak{g}_n$ the unitary matrix Lie group of degree n and its corresponding Lie algebra. Let $M \in \mathcal{L}(\mathbb{R}^d, \mathfrak{g}_n)$; the unitary representation function of X is given by the mapping*

$$\mathcal{U}_M(X) := \exp(M(X)), \tag{4.1}$$

where exp *denotes the matrix exponential. The unitary characteristic function of* X *is defined as a mapping from* $L(\mathbb{R}^d, \mathfrak{g}_n)$ *to* $GL(n)$ *such that*

$$UCF_n(X)(M) := \mathbb{E}_{X\sim\mathbb{P}_X}[\mathcal{U}_M(X)]. \tag{4.2}$$

In practice, $UCF_n(X)(M)$ *is approximated by Monte Carlo. Let* $\mathbf{X} = \{X_i\}_{i=1}^N$ *be* N *samples from* $\mathbb{P}_X$*. The empirical unitary characteristic function is given by*

$$EUCF_n(\mathbf{X})(M) := \frac{1}{N}\sum_{i=1}^{N}\mathcal{U}_M(X_i).$$

Remark 4.3. If $n = 1$, then $\mathfrak{g}_1$ is just the set of pure imaginary numbers, and M can be identified with an element $\lambda \in \mathbb{R}^d$ such that $M(x) = i\langle\lambda, x\rangle$ for all $x \in \mathbb{R}^d$, where $\langle\cdot,\cdot\rangle$ denotes the standard inner product. Then, $\mathrm{UCF}_1(X)$ recovers the standard characteristic function of a d-dimensional random variable.

Similarly to Definition 4.1, for any $n \geq 1$, we can define the *unitary characteristic function distance* (UCFD) for two random variables X and Y as

$$\mathrm{UCFD}_n^2(X,Y) = \mathbb{E}_{M\sim\mathbb{P}_{\mathcal{M}}}\left[\|\mathrm{UCF}_n(X)(M) - \mathrm{UCF}_n(Y)(M)\|_{HS}^2\right],$$

where $\mathbb{P}_{\mathcal{M}}$ denotes the distribution of the linear mapping M, and $\|\cdot\|_{HS}$ denotes the Hilbert–Schmidt norm. As in the case of the discriminator for the standard characteristic function, here we parameterize the law of linear mappings $\mathbb{P}_{\mathcal{M}}$ by an empirical measure

$$\mathbb{P}_{\mathcal{M}} = \frac{1}{N}\sum_{i=1}^{N}\delta_{M_i},$$

where δ denotes the Dirac measure, and each M_i can be parameterized as learnable coefficients and optimized using gradient-based methods. For computation and optimization details of UCFD, we refer the reader to [31]. Let $X = (W_1, A_1)$; then the benefits of UCFD compared to the standard characteristic function distance include the following:

(1) Standard uniqueness results hold as $\mathrm{UCF}_1(X)$ recovers the standard characteristic function.
(2) Since $\mathfrak{g}_1$ is a subspace of $\mathfrak{g}_n$ for any $n > 2$, $\mathcal{U}_M$ possesses a richer structure than $\mathbb{C}$ for any linear mapping M into $\mathfrak{g}_n$. Although using $\mathfrak{g}_1$ already encodes enough information to determine the random variable, embedding the random variable into Lie algebra of a higher degree appears to provide a more efficient way of representing the information that characterizes the random variable. Empirically, using the unitary characteristic function led to a more stable training procedure compared to the standard one.

4.2. Chen-training. While Brownian motion over different intervals can be concatenated simply via addition, i.e., $W_{0,t} = W_{0,s} + W_{s,t}$, concatenation of Lévy areas requires an additional term, specified by Chen's relation. In its general version within rough path theory, Chen's relation establishes the homomorphism property of path signatures under concatenation [32, Theorem 2.9]. However, we will present just the special case relating to Lévy area.

Proposition 4.4 (Chen's relation [2]). *For times $0 < s < t$,*

$$A_{0,t}^{(i,j)} = A_{0,s}^{(i,j)} + A_{s,t}^{(i,j)} + \frac{1}{2}\left(W_{0,s}^{(i)} W_{s,t}^{(j)} - W_{0,s}^{(j)} W_{s,t}^{(i)}\right).$$

We now present the main theoretical contribution of this section, which can be viewed as a partial converse of the above proposition.

Theorem 4.5 (distributional uniqueness of Lévy area under Chen's relation). *Suppose μ is a mean zero probability distribution on $\mathbb{R}^d \times \mathbb{R}^{d\times d}$, where the first marginal has finite second moment. Let $(V_i, Z_i) \overset{i.i.d.}{\sim} \mu$ for $i = 1, 2$; if it holds that*

$$V_3 := \frac{1}{\sqrt{2}}(V_1 + V_2), \quad Z_3 := \frac{1}{2}Z_1 + \frac{1}{2}Z_2 + \frac{1}{4}(V_1 \otimes V_2 - V_2 \otimes V_1) \tag{4.3}$$

is also distributed according to μ, then μ is the distribution of $(W_{0,1}, A_{0,1})$ where A is the Lévy area process associated with a d-dimensional Brownian motion W.

Including a finite-variance assumption on the measure μ, and a Gaussian assumption on the first marginal, provides an alternative proof using Wasserstein distances (see Theorem SM2.3 in the supplementary material). The preceding motivates the following procedure, which takes samples Z_1, Z_2 from some distribution and concatenates them using Chen's relation to produce samples that are closer in distribution to $A_{0,1}$. With abuse of notation, if $\mathbf{X} \in \mathbb{R}^{m\times d}$ consists of m samples of a d-dimensional random variable, we denote by $\mathbf{X}^{(i)} \in \mathbb{R}^m$ the ith coordinate of each sample. Adopting this notation, we describe the Chen-combine operation in Algorithm 4.1.

We also note that the proof of Theorem SM2.3 shows that the repeated application of Chen-combine gives convergence of order at least $\frac{1}{2}$ in the 2-Wasserstein metric. Faster rates can be proven assuming a suitable starting distribution, such as Davie's approximation [6, 9].

Algorithm 4.1. Chen-combine.

Input: m - batch size, d - Brownian dimension, a- Lévy dimension, $(\mathbf{W}, \widetilde{\mathbf{A}})_{\text{first}} \in \mathbb{R}^{m\times(d+a)}$, $(\mathbf{W}, \widetilde{\mathbf{A}})_{\text{second}} \in \mathbb{R}^{m\times(d+a)}$ - Brownian increments and Lévy area samples.

$\widehat{\mathbf{W}} \leftarrow \mathbf{0} \in \mathbb{R}^{m\times d}$, $\widehat{\mathbf{A}} \leftarrow \mathbf{0} \in \mathbb{R}^{m\times a}$

$\mathbf{W}_{\text{first}} \leftarrow \frac{1}{\sqrt{2}}\mathbf{W}_{\text{first}}$, $\mathbf{W}_{\text{second}} \leftarrow \frac{1}{\sqrt{2}}\mathbf{W}_{\text{second}}$, ▷ Brownian scaling $W_{0,\frac{1}{2}} \overset{d}{=} \frac{1}{\sqrt{2}}W_{0,1}$

$\widetilde{\mathbf{A}}_{\text{first}} \leftarrow \frac{1}{2}\widetilde{\mathbf{A}}_{\text{first}}$, $\widetilde{\mathbf{A}}_{\text{second}} \leftarrow \frac{1}{2}\widetilde{\mathbf{A}}_{\text{second}}$, ▷ Brownian scaling $A_{0,\frac{1}{2}} \overset{d}{=} \frac{1}{2}A_{0,1}$

for $i \in \{1, \dots, d\}$ **do**

  $\widehat{\mathbf{W}}^{(i)} \leftarrow \mathbf{W}_{\text{first}}^{(i)} + \mathbf{W}_{\text{second}}^{(i)}$

  for $j \in \{i+1, \dots, d\}$ **do**

    $\mathbf{D} \leftarrow \frac{1}{2}\left(\mathbf{W}_{\text{first}}^{(i)} \odot \mathbf{W}_{\text{second}}^{(j)} - \mathbf{W}_{\text{first}}^{(j)} \odot \mathbf{W}_{\text{second}}^{(i)}\right)$

    $\widehat{\mathbf{A}}^{(i,j)} \leftarrow \widetilde{\mathbf{A}}_{\text{first}}^{(i,j)} + \widetilde{\mathbf{A}}_{\text{second}}^{(i,j)} + \mathbf{D}$, ▷ Chen's relation

return $(\widehat{\mathbf{W}}, \widehat{\mathbf{A}})$

Remark 4.6. Let $\widetilde{\mathbf{A}}$ be the estimated Lévy area given a fixed Brownian increment $\mathbf{W}$. Let $\widetilde{\mathbf{A}}_{\text{Chen}}$ be the resulting Lévy area process of Chen-combine$(\mathbf{W}, \widetilde{\mathbf{A}})$. In Theorem SM2.4 of the supplementary material we show that (informally speaking)

$$\mathcal{W}_2(\widetilde{\mathbf{A}}, \mathbf{A}_{\text{true}}) \leq (2+\sqrt{2})\mathcal{W}_2(\widetilde{\mathbf{A}}, \widetilde{\mathbf{A}}_{\text{Chen}}),$$

where $\mathcal{W}_2$ denotes the 2−Wasserstein metric. So, in order to minimize $\mathcal{W}_2(\widetilde{\mathbf{A}}, \mathbf{A}_{\text{true}})$, we need only minimize $\mathcal{W}_2(\widetilde{\mathbf{A}}, \widetilde{\mathbf{A}}_{\text{Chen}})$, which requires no "true" samples of Lévy area to compute. This allows us to modify the training objective of the GAN so that it will learn the correct distribution *without any access to externally supplied data.*

Assume $(\mathbf{W}, \widetilde{\mathbf{A}}) \in \mathbb{R}^{2m\times(d+a)}$ is a collection of $2m$ samples of a $(d+a)$-dimensional random variable. By Chen-combine$(\mathbf{W}, \widetilde{\mathbf{A}})$ we mean the following: evenly split $(\mathbf{W}, \widetilde{\mathbf{A}})$ into two blocks with equal size, denoted by $(\mathbf{W}, \widetilde{\mathbf{A}})_{\text{first}}, (\mathbf{W}, \widetilde{\mathbf{A}})_{\text{second}} \in \mathbb{R}^{m\times(d+a)}$, and apply the operation described in Algorithm 4.1. The output of Chen-combine will be an element in $\mathbb{R}^{m\times(d+a)}$.

5. LévyGAN.

In this section, we incorporate the ideas presented in sections 3 and 4 into our tailored model LévyGAN used to generate the associated Lévy area conditioned on the Brownian increments. Similarly to Definition 2.1, we provide the definition of the proposed model as follows.

Definition 5.1 (LévyGAN). *Let $PairNN_\theta$ denote a pair-net generator as defined in Definition 3.9. Given a d-dimensional Brownian increment at $t=1$, $W \sim \mathcal{N}^d(0,1)$, for all $1 \leq i < j \leq d$, we generate an estimated Lévy area associated to W as follows:*

$$\tilde{b}^{(i,j)} = PairNN_\theta((H^{(i)}, Z^{(i)}), (H^{(j)}, Z^{(j)})), \tag{5.1}$$

$$\tilde{A}^{(i,j)} = H^{(i)}W^{(j)} - H^{(j)}W^{(i)} + \tilde{b}^{(i,j)}. \tag{5.2}$$

Let $\mathbf{w} \in \mathbb{R}^{N\times d}$ be N samples of Brownian increment, and let the associated $\widetilde{\mathbf{A}} \in \mathbb{R}^{N\times a}$ be generated according to (5.1) and (5.2); then we construct new samples using Chen-combine *defined as in Algorithm 4.1:*

$$(\mathbf{w}_{Chen}, \widetilde{\mathbf{A}}_{Chen}) := \text{Chen-combine}(\mathbf{w}, \widetilde{\mathbf{A}}).$$

For $m \geq 1$, let $\mathfrak{g}_m$ be the Lie algebra of the unitary matrix group U_m. Recall $EUCF_m$ from Definition 4.2, and let $\mathcal{M} = \{M_i\}_{i=1}^M$, $M_i \in L(\mathbb{R}^{d+a}, \mathfrak{g}_m)$ be a collection of linear mappings onto $\mathfrak{g}_m$, where each is parameterized by an element in $\mathbb{R}^{(d+a)\times dim(\mathfrak{g}_m)}$. $\mathcal{M}$ will play the role of the discriminator. Finally, the training is performed with respect to the min-max game

$$\min_\theta \max_{\mathcal{M}} Loss(\theta, \mathcal{M}; \mathbf{w}),$$

where $Loss(\theta, \mathcal{M}; \mathbf{w}) := EUCFD_m((\mathbf{w}, \widetilde{\mathbf{A}}), (\mathbf{w}_{Chen}, \widetilde{\mathbf{A}}_{Chen}))$. The training algorithm and flowchart are described in Algorithm 5.1 and Figure 4.

Remark 5.2. One can interpret Chen-training as a type of adaptive training, where the "almost true" target data $\widetilde{\mathbf{A}}_{\text{Chen}}$ is always just sufficiently better than the generator's output, so that training can progress effectively. Thus, instead of requiring large data-sets of "true"

Algorithm 5.1. Training algorithm for LévyGAN.

Input: d - Brownian dimension, a - Lévy area dimension, n - noise dimension, m - Lie algebra degree, M - number of linear mappings onto $\mathfrak{g}_m$, PairNN_θ - generator, $\mathcal{M} \in \mathbb{R}^{M\times(d+a)\times\dim(\mathfrak{g}_m)}$ - discriminator, iter_d - number of discriminator updates per generator update, `bsz` - batch size, η_g, η_d - generator and discriminator learning rates.

while $\theta, \mathcal{M}$ not converge **do**
  for $i \in (1, \dots, \mathrm{iter}_d)$ **do**
    Sample $\mathbf{w} \sim \mathcal{N}^d(0,1)$, $(\mathbf{h}, \mathbf{z}) \sim \mathcal{N}^d(0, \frac{1}{12}) \times \mathcal{N}^{d\times n}(0,1)$ of size $2\,\mathtt{bsz}$.
    $\widetilde{\mathbf{b}}^{(i,j)} \leftarrow \mathrm{PairNN}_\theta((\mathbf{h}^{(i)}, \mathbf{z}^{(i)}), (\mathbf{h}^{(j)}, \mathbf{z}^{(j)}))$ for $1 \le i < j \le d$
    $\xi_0 \leftarrow \mathrm{Rad}(1/2)$, $\boldsymbol{\xi} \leftarrow \mathrm{Rad}^d(1/2)$
    $\widetilde{\mathbf{A}} \leftarrow \mathrm{BF}(\mathbf{w}, \mathbf{h}, \widetilde{\mathbf{b}}, \xi_0, \boldsymbol{\xi})$
    $\mathbf{w}_{\mathrm{Chen}}, \widetilde{\mathbf{A}}_{\mathrm{Chen}} \leftarrow$ Chen-combine$(\mathbf{w}, \widetilde{\mathbf{A}})$
    $\mathrm{Loss}(\theta, \mathcal{M}; \mathbf{w}) \leftarrow \mathrm{EUCFD}_m((\mathbf{w}, \widetilde{\mathbf{A}}), (\mathbf{w}_{\mathrm{Chen}}, \widetilde{\mathbf{A}}_{\mathrm{Chen}}))$
    $\mathcal{M} \leftarrow \mathcal{M} - \eta_d \cdot \nabla_{\mathcal{M}}(-\mathrm{Loss}(\theta, \mathcal{M}; \mathbf{w}))$ ▷ Maximize the loss
  Sample $\mathbf{w} \sim \mathcal{N}^d(0,1)$, $(\mathbf{h}, \mathbf{z}) \sim \mathcal{N}^d(0, \frac{1}{12}) \times \mathcal{N}^{d\times n}(0,1)$ of size $2\,\mathtt{bsz}$.
  $\widetilde{\mathbf{b}}^{(i,j)} \leftarrow \mathrm{PairNN}_\theta((\mathbf{h}^{(i)}, \mathbf{z}^{(i)}), (\mathbf{h}^{(j)}, \mathbf{z}^{(j)}))$ for $1 \le i < j \le d$
  $\xi_0 \leftarrow \mathrm{Rad}(1/2)$, $\boldsymbol{\xi} \leftarrow \mathrm{Rad}^d(1/2)$
  $\widetilde{\mathbf{A}} \leftarrow \mathrm{BF}(\mathbf{w}, \mathbf{h}, \widetilde{\mathbf{b}}, \xi_0, \boldsymbol{\xi})$
  $\mathbf{w}_{\mathrm{Chen}}, \widetilde{\mathbf{A}}_{\mathrm{Chen}} \leftarrow$ Chen-combine$(\mathbf{w}, \widetilde{\mathbf{A}})$
  $\mathrm{Loss}(\theta, \mathcal{M}; \mathbf{w}) \leftarrow \mathrm{EUCFD}_m((\mathbf{w}, \widetilde{\mathbf{A}}), (\mathbf{w}_{\mathrm{Chen}}, \widetilde{\mathbf{A}}_{\mathrm{Chen}}))$
  $\theta \leftarrow \theta - \eta_d \cdot \nabla_\theta \mathrm{Loss}(\theta, \mathcal{M}; \mathbf{w})$ ▷ Minimize the loss
return $\mathrm{PairNN}_\theta, \mathcal{M}$

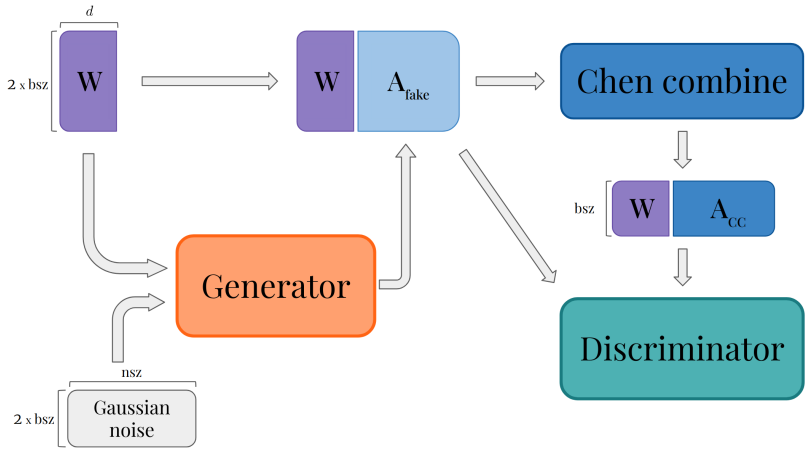


Figure 4. *A schematic of LévyGAN. Here* bsz *denotes the batch dimension, and we recall that nsz denotes the total noise dimension, namely* $\mathrm{nsz} = n \times d$.

samples, which are costly to generate and difficult to handle, we can now very efficiently generate new "true" data on the fly, of any desired quantity and of just the right precision. One could choose to iteratively apply Chen-combine several times, but we have observed that training is slightly faster and more efficient when only a single application of Chen combined

is used in Algorithm 5.1. This is because using two Chen-combines produces two fewer "true" samples, and reducing the sample size leads to a poorer estimate of the empirical unitary characteristic function distance (EUCFD).

6. Numerical experiments. We train the model in $d = 4$. Note that by the architecture of the generator, the model can be used to generate Brownian Lévy area for any $d' \leq d$. The model can be also used to generate Lévy area for any $d' > d$; however, the performance might be deteriorated as training is not done for higher dimensions.

We performed the training procedure as illustrated in Algorithm 5.1. On the generator side, we used a feed-forward neural network. The activation function is chosen to be the LeakyReLU function. On the discriminator side, we parameterize 128 linear maps onto the Lie algebra of degree 3 to mimic the empirical distribution used to compute UCFD mentioned in subsection 4.1. The total number of training iterations is set to be 2500, where we observed the convergence on the marginal 2-Wasserstein metric on real data. We optimize both the generator and discriminator using stochastic gradient descent and the Adam optimizer. We set the batch size to 2^{13} and the learning rate for generator/discriminator to 0.001/0.01, respectively. Both learning rates decay for each 500 iteration. Finally, we set iter_d to 3.

We conducted a hyperparameter grid-search (see section SM5 of the supplement ary material), evaluating the model performance according to the marginal 2-Wasserstein metric, with our optimal architecture as follows:

- Feed-forward neural network with 3 hidden layers and 16 hidden dimensions.
- LeakyReLU activation function with slope $= 0.01$.
- Gaussian noise with $n = 3$.

Finally, we assess the performance of our model on the generation of the coupled process for $d = 2,\ 3,\ 4$, and 8. We consider the following test metrics:

(1) Marginal 2-Wasserstein metric.
(2) Cross moment metric.
(3) Characteristic function distance using maximum mean discrepancy with different kernels.
(4) Empirical unitary characteristic function.

A detailed explanation of each test metric can be found in section SM4 of the supplementary material. We make comparisons with two baselines, Foster's and Davie's moment matching generator [10, 6], and we regard the truncated Fourier series [20] of Lévy area up to an L^2 precision of 10^{-4} as "true" samples. A comparison of the computational time taken to generate approximate samples of Lévy area using each method, together with a summary of the marginal distributional performance is found in Table 1. The performance of each method with respect to the joint distributional metrics is summarised in Table 2. Finally, we provide a numerical SDE example for the log-Heston model using different estimators for fake Lévy area.

6.1. SDE example. In this section, we will demonstrate how "fake" Lévy area can be used within SDE numerics to achieve both high order weak convergence as well as multilevel Monte Carlo (MLMC) variance reduction. Although the synthetic Lévy area only needs to exhibit the correct mean and covariance to give high order weak convergence, we show that the bias introduced by the MLMC estimator is negligible in practice due to the small Chen error inherent in our generative model. A secondary motivation is to compare the various Lévy area

Table 1

Marginal distribution fitting and computational efficiency for the different generative models. The generation is done using NVIDIA Quadro RTX 8000. The marginal W_2 error is calculated with respect to the joint process generated by the Fourier series. Tests are performed with 2^{20} samples. The final column contains the results of the Fourier algorithm with 19 terms in the expansion (far more terms were used to generate "true" samples). This truncation has been chosen so that the performance is comparable to LévyGAN and Foster's method.

Test Metric	LévyGAN	Foster	Davie	Fourier series
Computational time (s)	0.019	0.0071	0.002	3.1
Marginal W_2 (10^{-2})	$\mathbf{.246 \pm .013}$	$.254 \pm .010$	$2.03 \pm .013$	$.27 \pm 0.008$

Table 2

Fourth moment and MMD-based metrics across different models and Brownian dimensions.

Dim	**Test metrics**	LévyGAN	Foster	Davie
2	Fourth moment	$.004 \pm .002$	$\mathbf{.002 \pm .002}$	$.042 \pm .001$
	Polynomial MMD (10^{-5})	$\mathbf{.341 \pm .070}$	$.654 \pm .131$	$.646 \pm .188$
	Gaussian MMD (10^{-6})	$1.47 \pm .125$	$\mathbf{1.44 \pm .128}$	$34.6 \pm .683$
	EUCFD (10^{-2})	$\mathbf{1.52 \pm .213}$	$1.92 \pm .113$	$10.1 \pm .851$
3	Fourth moment	$\mathbf{.004 \pm .002}$	$\mathbf{.004 \pm .002}$	$.043 \pm .001$
	Polynomial MMD (10^{-5})	$\mathbf{2.18 \pm .568}$	$2.30 \pm .732$	$2.26 \pm .773$
	Gaussian MMD (10^{-6})	$1.87 \pm .002$	$\mathbf{1.84 \pm .001}$	$16.3 \pm .001$
	EUCFD (10^{-2})	$\mathbf{1.88 \pm .063}$	$2.03 \pm .034$	18.5 ± 1.11
4	Fourth moment	$\mathbf{.004 \pm .000}$	$.006 \pm .002$	$.043 \pm .002$
	Polynomial MMD (10^{-5})	$\mathbf{4.04 \pm .436}$	4.65 ± 1.31	$5.62 \pm .808$
	Gaussian MMD (10^{-6})	$\mathbf{1.90 \pm .001}$	$\mathbf{1.90 \pm .001}$	$263 \pm .003$
	EUCFD (10^{-2})	$\mathbf{1.92 \pm .026}$	$2.03 \pm .036$	$17.5 \pm .483$
8	Fourth moment	$\mathbf{.006 \pm .001}$	$\mathbf{.006 \pm .002}$	$.044 \pm .000$
	Polynomial MMD (10^{-2})	$\mathbf{1.13 \pm .019}$	$1.15 \pm .030$	$1.31 \pm .066$
	Gaussian MMD (10^{-6})	$\mathbf{1.91 \pm .001}$	$\mathbf{1.91 \pm .000}$	$1.92 \pm .003$
	EUCFD (10^{-2})	$\mathbf{1.99 \pm .002}$	$\mathbf{1.99 \pm .001}$	$2.05 \pm .003$

generators and show that our GAN-based approach performs indistinguishably from previous state-of-the-art methods—all while taking less time to generate samples.

Consider the Itô SDE from (1.1),

(1.1 revisited)
$$dX_t = f(X_t)dt + \sum_{i=1}^{d} g_i(X_t)dW_t^{(i)}, \quad X_0 = x_0,$$

where the solution X takes values in $\mathbb{R}^e$. To estimate the solution to (1.1), one typically uses a discretization scheme that generates approximate sample paths of the solution X. Often the objective is to approximate quantities of the form

(6.1)
$$\mathbb{E}\left[\varphi(X) \,|\, X_0 = x_0\right],$$

where φ may depend on the whole sample path $(X_t)_{t\in[0,T]}$, though commonly it is only a function of the solution X_T at the terminal time T. To measure the error of a particu-

lar discretization scheme, there are two standard metrics: weak and strong error. We will only evaluate the weak error for reasons discussed in subsection SM7.1 of the supplementary material. To accurately determine the error of various numerical schemes, we seek a multidimensional SDE and a quantity of the form (6.1) which is known semianalytically. Thankfully, such an example exists: the price of a European call-option under the log-Heston model. The stochastic volatility model is defined by the two-dimensional SDE

$$\begin{aligned} dU_t &= \left(r - \frac{1}{2}V_t\right)dt + \sqrt{V_t}dW_t^{(1)}, \quad U_0 \in \mathbb{R}, \\ dV_t &= \kappa(\theta - V_t)dt + \sigma\sqrt{V_t}dW_t^{(2)}, \quad V_0 > 0, \end{aligned} \tag{6.2}$$

for a pair of independent Brownian motions $W^{(1)}$ and $W^{(2)}$. To ensure the volatility term V remains positive, we must enforce the Feller condition $2\kappa\theta - \sigma^2 > 0$. The payoff of a European call-option for a price process S with $S := \exp(U)$ is given by

$$\varphi(S) := e^{-rT}\left(e^{U_T} - K\right)^+,$$

where r is the discount rate, K the strike price, and T the maturity. For the derivation and form of the semianalytic formula for the expected value of the above, we refer the reader to [46, 5].

6.2. Numerical results. We compare four discretization schemes combined with multilevel Monte Carlo (MLMC) [16]. We briefly recall that MLMC is based on the idea of a telescoping sum of expectations. Indeed, assume we have L levels, and that Y_l is an estimator for X, based on a discretization scheme with step-size h_l; then we may write

$$\mathbb{E}\left[\varphi(Y_L)\right] = \sum_{l=1}^{L} \mathbb{E}\left[\varphi(Y_l) - \varphi(Y_{l-1})\right],$$

with $Y_0 \equiv 0$. The MLMC estimator is then defined by

$$\bar{\varphi}_{n_1,\dots,n_L} = \sum_{l=1}^{L} \widehat{\varphi}_{n_l}, \quad \text{where} \quad \widehat{\varphi}_{n_l} = \frac{1}{n_l}\sum_{i=1}^{n_l}\left(\varphi(Y_l^{i,l}) - \varphi(Y_{l-1}^{i,l})\right), \tag{6.3}$$

where $Y_l^{i,l}$ is the ith sample of the estimator Y_l used on level l, and $Y_{l-1}^{i,l}$ is the ith sample of the estimator Y_{l-1} that is used on level l. It is important to note that the pairs $(Y_l^{i,l}, Y_{l-1}^{i,l})$ are coupled: the underlying Brownian path for each member of the pair is the same. In our case, the path on the lower level will be coarse (i.e., a large step-size) and the higher level will be fine (i.e., a small step-size). The standard condition used to ensure convergence of the telescoping sum of expectations is given by

$$\mathbb{E}\left[\varphi(Y_l^{\cdot,l})\right] = \mathbb{E}\left[\varphi(Y_l^{\cdot,l+1})\right].$$

However, when incorporating a fake Lévy area term, our coupling at each level is defined as follows.

(1) The Brownian increments for the fine path $Y_l^{\cdot,l}$ are generated with step-size h_l, with the increments of fake Lévy area generated using some estimator $\widetilde{A}^{h_l}$.
(2) The Brownian increments on the coarse path $Y_{l-1}^{\cdot,l}$ are computed by pairwise summing the increments of the fine path. The fake Lévy area used on the coarse path is computed using one iteration of Chen's identity applied to the increments and areas of the fine path.

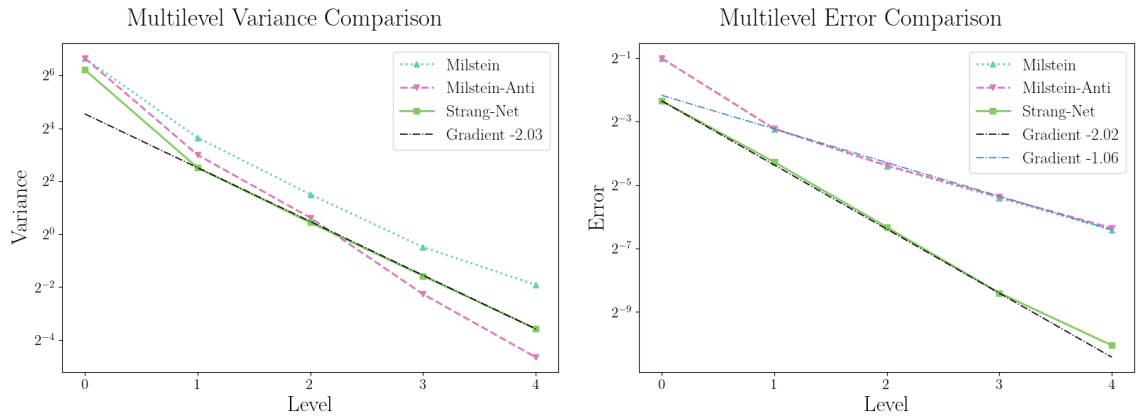
This scheme, however, introduces a bias; namely, the distribution of the fake Lévy area used for the fine path at level l will not be the same as the distribution of Lévy area used for the coarse path at level $l+1$. We may write the effect of this by amending the telescoping expectation to be

$$\sum_{l=1}^{L} \mathbb{E}\left[\varphi(Y_l(\widetilde{A}^{h_l})) - \varphi(Y_{l-1}(\widetilde{A}^{h_l}_{\mathrm{CC}}))\right] = \underbrace{\sum_{l=1}^{L} \mathbb{E}\left[\varphi(Y_l(\widetilde{A}^{h_l})) - \varphi(Y_{l-1}(\widetilde{A}^{h_{l-1}}))\right]}_{\text{desired telescoping expectation}} + \underbrace{\sum_{l=1}^{L} \mathbb{E}\left[\varphi(Y_{l-1}(\widetilde{A}^{h_{l-1}})) - \varphi(Y_{l-1}(\widetilde{A}^{h_l}_{\mathrm{CC}}))\right]}_{\text{bias term}} \tag{6.4}$$

where we have emphasized the dependence of the fine level on the fake area $\widetilde{A}^{h_l}$ and dependence of the coarse level of one Chen iteration of this fake area, denoted by $\widetilde{A}^{h_l}_{\mathrm{CC}}$. The bias introduced is exactly the second sum. We aim to show empirically that this sum is small in comparison to the size of the weak error due to the SDE discretization. In order to minimize this sum, the distribution of an estimator $\widetilde{A}$ must be as close as possible to the distribution Chen-combine$(\widetilde{A})$; this is exactly the criterion used to train our generator.

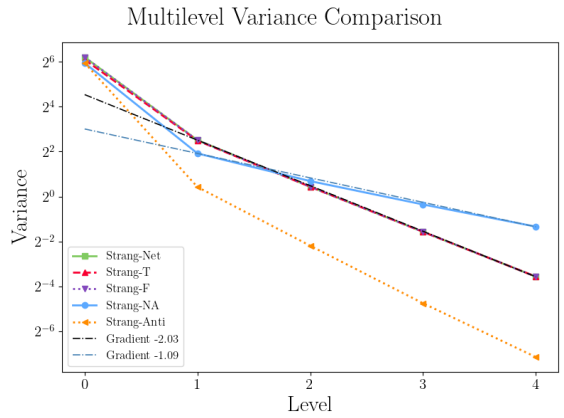
The four numerical schemes in the comparison are the no-area Milstein, antithetic Milstein [17], the Strang splitting method, and a "Strang" log-ODE method. Only the final method incorporates the fake Lévy area. For details on the schemes, see subsection SM7.2 in the supplementary material. The first two schemes were included to demonstrate that the rate of variance reduction is comparable to two popular schemes, while the weak error rate of the Strang log-ODE method is (conjectured to be) $O(h^2)$, where the other methods achieve a weak error rate of $O(h)$. The numerical simulations were performed with a constant time-step h_l on each level satisfying $h_l = \frac{1}{2}h_{l-1}$. On the coarsest level we use the time-step $h_0 = \frac{1}{2}$ for the Milstein methods and $h_0 = 1$ for the Strang methods; this results in the variance on this level being approximately equal across the three methods. The number of sample paths on each level satisfies $n_l = \frac{1}{2}n_{l-1}$ with $n_0 = 2^{27}$, so the computational effort on each level is approximately constant. We repeat the experiment 40 times and report the average result. We fix the log-Heston model parameters to be $T = 1$, $r = 0.1$, $K = 20$, $\kappa = 2$, $\theta = 0.1$, $\sigma = 0.5$, $U_0 = \log(20)$, and $V_0 = 0.4$. The following plots report the multilevel variance defined by $\mathrm{Var}[\varphi(Y_l^{\cdot,l}) - \varphi(Y_{l-1}^{\cdot,l})]$, and the empirical error given by $|\bar{\varphi}_{n_1,\ldots,n_l} - P_{\text{true}}|$, where P_{true} is the true price of the call-option under the log-Heston model.

In Figure 5a we see that the multilevelvariance of the Strang log-ODE method decreases at an approximate rate of $O(h^2)$. The rate for the Milstein antithetic scheme appears slightly higher, while the variance reduction rate for the standard Milstein method is clearly lower. As

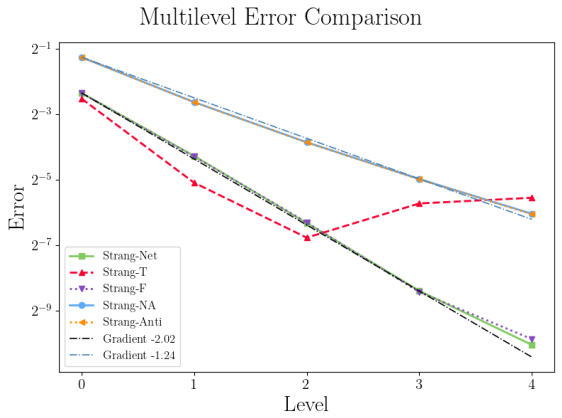


(a) Multilevel variance

(b) Empirical error



(c) Multilevel variance



(d) Empirical error

Figure 5. *Plots of multilevel variance and empirical error. The top pair of plots compare the Milstein scheme without area, Milstein antithetic, and Strang log-ODE scheme with fake Lévy area from our generator (labeled "Strang-Net"). The bottom pair compare the Strang log-ODE scheme using three different fake Lévy areas. "Strang-T" indicates that the fake Lévy area is an independent Rademacher random variable with the correct variance (the same random variable appearing in the Talay scheme), and "Strang-F" denotes Foster's approximation. The "Strang-NA" line is the usual Strang splitting method, with "Strang-Anti" being the antithetic version of this scheme.*

expected, the weak convergence rate of both the Milstein and Milstein antithetic schemes is of order $O(h)$, while the weak rate for the Strang log-ODE with fake Lévy area is approximately $O(h^2)$. It was conjectured in [11] that the Strang log-ODE method should attain this weak convergence rate, and the experiments corroborate this hypothesis.

When using a fake Lévy area in MLMC, a key factor for the performance of the scheme is how close the distribution of a Chen combined sample of Brownian motion and fake Lévy area is to the distribution before performing the combine operation. Since we employed Chen-training, our model succeeds in matching the distributions well enough to match and even

Table 3

Approximate bias introduced per level by the use of different fake Lévy areas. We use 2^{29} sample paths on each level.

Level	0	1	2	3
Strang-Net (10^{-3})	-0.402	0.313	-0.133	-0.652
Strang-F (10^{-3})	0.361	0.848	0.0714	1.31
Strang-T (10^{-3})	-17.0	4.33	-1.79	-1.10

Table 4

The average time taken across 25 runs for the Milstein and Strang log-ODE methods to attain a target RMSE. We use the algorithm of [16, section 5] to determine the number of samples on each level and the stopping condition on the number of levels used on each run. All random variables are generated in `torch` on `GPU`, with the numerical schemes implemented in `numpy` on `CPU`.

RMSE	0.1	0.0441	0.0129	0.0086	0.0057	0.0038	0.0025
Milstein (s)	**0.0097**	0.0256	0.376	1.03	2.86	8.63	23.6
LévyGAN (s)	0.0102	**0.0128**	**0.142**	**0.311**	**0.806**	**2.25**	**5.83**

outperform Foster's method. In Table 3, we record the bias introduced at each level, as in (6.4), for each of the fake Lévy areas.

For LévyGAN and Foster's method, the bias introduced on each level of the telescoping sum is of order 2^{-11}; this is far smaller than the weak error seen in Figure 5d. However, it is possible that the accumulated bias may then be of order 2^{-9} on the finest level, which may account in part for the slight deviation from the line at level 4. It is also clear that it is not enough for the fake Lévy area simply to match the mean and variance of true Lévy area, as demonstrated by the poor performance of the "Strang-T" method. However, we do note here that a scheme matching the conditional variance of Lévy area given a Brownian increment performed similarly to Foster's method in previous experiments. We may also see from the performance of the Strang splitting method that, without the fake Lévy area terms, one achieves only a weak order convergence rate of $O(h)$. It is interesting to note, however, that Figure 5c indicates that the fake Lévy area need only match the mean and variance of true Lévy area to obtain improved variance reduction at each level. Even the "Strang T" method has the same variance reduction rate as the more sophisticated techniques despite having poor empirical error.

In practice, one usually wishes to obtain some target root mean-squared error (RMSE) with minimal computational cost. In this setting, for two numerical schemes with variance reduction $O(h^\beta)$ with $\beta > 1$, the scheme with higher order weak convergence is not necessarily the preferred one. By the complexity theorem of Giles [16, Theorem 3.1], the optimal number of sample paths on each level should be asymptotically proportional to $O(h_l^{(\beta+1)/2})$. As such, the computational effort should be expended mostly on the coarse levels in the regime $\beta > 1$, driving one towards discretizations that are computationally cheap on the lower levels. Since it is difficult to measure the computational complexity of the Strang log-ODE scheme with fake Lévy area produced by a generative model, we use the following approach. We implement the algorithm of Giles [16, section 5] for the Milstein scheme and Strang log-ODE scheme and compare the average time taken to achieve a selection of target RMSEs between 0.1 and 0.0025; see Table 4 for the results.

Remark 6.1. We see from Figure 5c that the Strang-antithetic scheme achieves the highest order variance reduction with a first order weak error rate. As noted above, both factors play a role in the overall time required to achieve a desired RMSE, and it is possible that the higher order variance reduction of the Strang-antithetic scheme may outperform the high order weak error of the LévyGAN- based approach. However, it was recently shown by Iguchi et al. [19] that the antithetic method can be combined with weak estimators of Lévy area to achieve high order variance reduction. We believe that combining their approach with ours would result in a scheme with both high order variance reduction and weak error rate.

We conclude this section by reiterating that the use of extra random variables to attain higher order weak convergence has become a popular technique; see, for example, [44, 38, 37]. But, to the best of our knowledge, it has not yet been observed that the use of fake Lévy area combined with standard multilevel Monte Carlo can also achieve high order weak convergence and variance reduction.

7. Generating other integrals of Brownian motion. To demonstrate the wider applicability of the Chen-training paradigm, we turn our attention to generating the integral $C_{s,t} = \int_s^t W_{s,r}^2\, dr$ where W is a 1-dimensional Brownian motion. While a detailed discussion of this integral is beyond the scope of this paper, we note that numerical methods for scalar noise SDEs can achieve second order strong convergence if this integral is provided alongside Brownian increments and space-time Lévy areas (see [13, 45] for further details). Analogously to the procedure outlined in this article for the generation of space-space Lévy area, we may attempt to train a network to sample from the distribution $\mathbb{P}_{C_{0,1}|W_{0,1}=w,\ H_{0,1}=h}$, where H is the space-time Lévy area defined in Definition 3.1. Due to Brownian scaling, the distribution of $C_{0,1}$ satisfies the following scale invariance and modified Chen's relation.

Proposition 7.1 (scaling and Chen's relation for C). *Let W and C be defined as above, and le t $0 \le s < t$. Then*

$$C_{s,t} \overset{d}{=} (t-s)^2 C_{0,1} \qquad \text{and} \qquad C_{0,1} = C_{0,\frac{1}{2}} + C_{\frac{1}{2},1} + W_{0,\frac{1}{2}}\Big(\frac{W_{0,1}}{2} + H_{\frac{1}{2},1}\Big).$$

We note that since $C_{0,1}$ is 1-dimensional and nonnegative, neither pair-net nor bridge-flipping is required. We can then train a feed-forward neural network with 2 hidden layers, 16 hidden dimensions, Gaussian noise of dimension 3 in addition to w and h, ReLU activation function, and the absolute value applied to the output to maintain positivity. We then train using the Chen relation described in the preceding. To compare our output, we generated "true samples" for fixed pairs (w, h) by taking fine discretization of $\int_s^t W_{s,r}^2\, dr$ using the Diffrax library [21].

Our method achieved a 2-Wasserstein error of 3.27×10^{-5}. Just as for space-space Lévy area, we can generate a Gaussian variable with the correct conditional mean and variance (see [13]). However, this achieved a 2-Wassertein error of 7.67×10^{-4}, which is 20 times higher than our error. This experiment demonstrates that the Chen-training approach is not limited to only Lévy area but is applicable to other integrals of Brownian motion that have relevance in the numerical solutions of SDEs.

8. Conclusion and open directions. While stochastic analysis techniques are often used in generative deep learning, this article appears to be one of the first to show that deep-learning

methodology provides meaningful results in an application to stochastic analysis. Indeed, we have demonstrated a proof of concept that the techniques used in LévyGAN have a place in the field of numerical solutions to SDEs. We remark here though, that careful consideration of the domain-specific analytical properties was required. In particular, regardless of the network size or architecture, LévyGAN in its initial form was an order of magnitude less accurate without the inclusion of both bridge-flipping and pair-net.

One open direction for future research is a careful analysis of the conditions required on the fake Lévy area in order to achieve optimal convergence rates for the MLMC scheme discussed in subsection 6.2. More applications of fake Lévy area may be found in the field of stochastic flows and accurate approximations to the level two rough path of Brownian motion.

An application of particular interest would be a GAN-based adaptive SDE solver, that is, a method that first generates a coarsely discretized path, before checking whether the step-size of the solver should be reduced. Such functionality is desirable for use in neural SDEs [28, 22, 23] and Logsig-RNN generators [36], which are both powerful methods for modeling noisy time series data. In the context of Lévy area generation, this would require the ability to generate Lévy area and Brownian increments over two half intervals given the Lévy area and Brownian increment over the larger interval. One approach would be to use the analytical characteristic function given in [15] which provides the joint characteristic function evaluated at multiple time points. However, we expect the training time to be rather long, since the evaluation of the characteristic function involves solving a recursive system of matrix Ricatti equations in addition to a system of independent linear matrix ODEs of order one.

Finally, it is possible to extend the Chen-training approach. This might take several forms: one might derive a Chen-type relation for higher order terms in the polynomial expansion of Brownian motion (e.g., for H and b) as in section 7; use the ordinary Chen's relation for the generation of higher order terms in the log-signature of Brownian motion; or generate Lévy areas for certain Lévy processes.

For the third application, our approach may be applicable to the generation of Lévy areas of α-stable Lévy processes, where moment matching approaches are not possible, since these processes have unbounded variance for $\alpha < 2$. Indeed, the α-stable Lévy process X_t^α satisfies the scaling property $X_t^\alpha = t^{1/\alpha} X_1^\alpha$ in addition to independent and stationary increments. Combining these properties, the distribution of its Lévy area (defined using Itô integration with jumps) should also be invariant under a suitably rescaled version of Chen's relation. One could then attempt to train a network under this Chen relation, analogously to our approach for Brownian motion.

Acknowledgments. The authors are grateful to Veronika Chronholm and Kanakira Terada for providing code which was then adapted for the numerical SDE tests in subsection 6.2. HN and JT thank Terry Lyons and Hang Lou for useful discussions.

REFERENCES

[1] A. Browning, D. Warne, K. Burrage, R. Baker, and M. Simpson, *Identifiability analysis for stochastic differential equation models in systems biology*, J. R. Soc. Interface, 17 (2020), 20200652, https://doi.org/10.1098/rsif.2020.0652.

[2] K.-T. Chen, *Integration of paths, geometric invariants and a generalized Baker–Hausdorff formula*, Ann. Math., 65 (1957), pp. 163–178, https://doi.org/10.2307/1969671.

[3] I. Chevyrev and T. Lyons, *Characteristic functions of measures on geometric rough paths*, Ann. Probab., 44 (2016), pp. 4049–4082, https://doi.org/10.1214/15-AOP1068.
[4] J. M. C. Clark and R. J. Cameron, *The maximum rate of convergence of discrete approximations for stochastic differential equations*, in Stochastic Differential Systems Filtering and Control, Springer, Berlin, Heidelberg, 1980, pp. 162–171.
[5] R. Crisostomo, *An Analysis of the Heston Stochastic Volatility Model: Implementation and Calibration Using MATLAB*, preprint, https://arxiv.org/abs/1502.02963, 2015.
[6] A. Davie, *KMT theory applied to approximations of SDE*, in Stochastic Analysis and Applications 2014, Springer Proc. Math. Statist. 100, Springer, Cham, 2014, pp. 185–201.
[7] A. S. Dickinson, *Optimal approximation of the second iterated integral of Brownian motion*, Stoch. Anal. Appl., 25 (2007), pp. 1109–1128, https://doi.org/10.1080/07362990701540592.
[8] A. Fatir Ansari, J. Scarlett, and H. Soh, *A characteristic function approach to deep implicit generative modeling*, in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, 2020, pp. 7476–7484, https://doi.org/10.1109/CVPR42600.2020.00750.
[9] G. Flint and T. Lyons, *Pathwise Approximation of SDEs by Coupling Piecewise Abelian Rough Paths*, preprint, https://arxiv.org/abs/1505.01298, 2015.
[10] J. Foster, *Numerical Approximations for Stochastic Differential Equations*, Ph.D. thesis, Oxford University, Oxford, UK, 2020.
[11] J. M. Foster, G. dos Reis, and C. Strange, *High order splitting methods for SDEs satisfying a commutativity condition*, SIAM J. Numer. Anal., 62 (2024), pp. 500–532, https://doi.org/10.1137/23M155147X.
[12] J. Foster and K. Habermann, *Brownian bridge expansions for Lévy area approximations and particular values of the Riemann Zeta function*, Combin. Probab. Comput., 32 (2023), pp. 370–397, https://doi.org/10.1017/S096354832200030X.
[13] J. Foster, T. Lyons, and H. Oberhauser, *An optimal polynomial approximation of Brownian motion*, SIAM J. Numer. Anal., 58 (2020), pp. 1393–1421, https://doi.org/10.1137/19M1261912.
[14] J. G. Gaines and T. J. Lyons, *Random generation of stochastic area integrals*, SIAM J. Appl. Math., 54 (1994), pp. 1132–1146, https://doi.org/10.1137/S0036139992235706.
[15] X. Geng and Z. Qian, *On the Finite Dimensional Joint Characteristic Function of Lévy's Stochastic Area Processes*, preprint, https://arxiv.org/abs/1206.1241, 2012.
[16] M. B. Giles, *Multilevel Monte Carlo path simulation*, Oper. Res., 56 (2008), pp. 607–617, https://doi.org/10.1287/opre.1070.0496.
[17] M. B. Giles and L. Szpruch, *Antithetic multilevel Monte Carlo estimation for multi-dimensional SDEs without Lévy area simulation*, Ann. Appl. Probab., 24 (2014), pp. 1585–1620, https://doi.org/10.1214/13-AAP957.
[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, in Advances in Neural Information Processing Systems, Vol. 27, 2014, https://proceedings.neurips.cc/paper_files/paper/2014/file/f033ed80deb0234979a61f95710dbe25-Paper.pdf.
[19] Y. Iguchi, A. Jasra, M. Maama, and A. Beskos, *Antithetic Multilevel Methods for Elliptic and Hypo-elliptic Diffusions with Applications*, preprint, https://arxiv.org/abs/2403.13489, 2024.
[20] F. Kastner and A. Rössler, *An analysis of approximation algorithms for iterated stochastic integrals and a Julia and MATLAB simulation toolbox*, Numer. Algorithms, 93 (2023), pp. 27–66, https://doi.org/10.1007/s11075-022-01401-z.
[21] P. Kidger, *On Neural Differential Equations*, Ph.D. thesis, University of Oxford, Oxford, UK, 2021.
[22] P. Kidger, J. Foster, X. Li, and T. Lyons, *Efficient and accurate gradients for neural SDEs*, in Advances in Neural Information Processing Systems, Vol. 34, 2021, pp. 18747–18761, https://proceedings.neurips.cc/paper_files/paper/2021/file/9ba196c7a6e89eafd0954de80fc1b224-Paper.pdf.
[23] P. Kidger, J. Foster, X. Li, H. Oberhauser, and T. Lyons, *Neural sdes as infinite-dimensional gans*, in Proceedings of the 38th International Conference on Machine Learning, Proceedings of Machine Learning Research 139, PMLR, 2021, pp. 5453–5463, https://proceedings.mlr.press/v139/kidger21b/kidger21b.pdf.
[24] P. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Appl. Math. (N. Y.), 23, Springer, Berlin, 1992.

[25] P. Kloeden, E. Platen, and I. Wright, *The approximation of multiple stochastic integrals*, Stoch. Anal. Appl., 10 (1992), pp. 431–441, https://doi.org/10.1080/07362999208809281.
[26] B. Leimkuhler and C. Matthews, *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*, Interdiscip. Appl. Math., 39, Springer, Cham, 2015.
[27] S. Li, Z. Yu, M. Xiang, and D. Mandic, *Reciprocal adversarial learning via characteristic functions*, in Advances in Neural Information Processing Systems, Vol. 33, 2020, pp. 217–228.
[28] X. Li, T.-K. L. Wong, R. T. Q. Chen, and D. K. Duvenaud, *Scalable gradients and variational inference for stochastic differential equations*, in Proceedings of the 2nd Symposium on Advances in Approximate Bayesian Inference, Proc. Mach. Learn. Res. (PMLR) 118, 2020, pp. 1–28.
[29] X. Li, D. Wu, L. Mackey, and M. A. Erdogdu, *Stochastic Runge-Kutta accelerates Langevin Monte Carlo and beyond*, in Advances in Neural Information Processing Systems, 2019, https://proceedings.neurips.cc/paper_files/paper/2019/file/7d265aa7147bd3913fb84c7963a209d1-Paper.pdf.
[30] M. Loève, *Probability Theory* II, 4th ed., Grad. Texts in Math. 46, Springer-Verlag, New York-Heidelberg, 1978.
[31] H. Lou, S. Li, and H. Ni, *Pcf-gan: Generating sequential data via the characteristic function of measures on the path space*, in Advances in Neural Information Processing Systems, Vol. 36, 2023, pp. 39755–39781.
[32] T. J. Lyons, M. Caruana, and T. Lévy, *Differential Equations Driven by Rough Paths*, Springer, Berlin, Heidelberg, 2007.
[33] G. N. Milstein and M. V. Tretyakov, *Stochastic Numerics for Mathematical Physics*, Springer, Berlin Heidelberg, 2004.
[34] M. Mirza and S. Osindero, *Conditional generative adversarial nets*, preprint, https://arxiv.org/abs/1411.1784, 2014.
[35] J. Mrongowius and A. Rössler, *On the approximation and simulation of iterated stochastic integrals and the corresponding Lévy areas in terms of a multidimensional Brownian motion*, Stoch. Anal. Appl., 40 (2022), pp. 397–425, https://doi.org/10.1080/07362994.2021.1922291.
[36] H. Ni, L. Szpruch, M. Sabate-Vidales, B. Xiao, M. Wiese, and S. Liao, *Sig-Wasserstein gans for time series generation*, in Proceedings of the Second ACM International Conference on AI in Finance, 2021, pp. 1–8.
[37] M. Ninomiya and S. Ninomiya, *A new higher-order weak approximation scheme for stochastic differential equations and the Runge–Kutta method*, Finance Stoch., 13 (2009), pp. 415–443, https://doi.org/10.1007/s00780-009-0101-4.
[38] S. Ninomiya and N. Victoir, *Weak approximation of stochastic differential equations and application to derivative pricing*, Appl. Math. Finance, 15 (2008), pp. 107–121, https://doi.org/10.1080/13504860701413958.
[39] E. Pardoux and D. Talay, *Discretization and simulation of stochastic differential equations*, Acta Appl. Math., 3 (1985), pp. 23–47, https://doi.org/10.1007/BF01438265.
[40] A. Rößler, *Second order Runge–Kutta methods for Itô stochastic differential equations*, SIAM J. Numer. Anal., 47 (2009), pp. 1713–1738, https://doi.org/10.1137/060673308.
[41] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, *The graph neural network model*, IEEE Trans. Neural Networks, 20 (2009), pp. 61–80, https://doi.org/10.1109/TNN.2008.2005605.
[42] S. Shreve, *Stochastic Calculus for Finance* II*: Continuous-Time Models*, Springer Finance, Springer-Verlag, New York, 2004.
[43] B. Sriperumbudurm, A. Gretton, K. Fukumizu, B. Schölkopf, and G. Lanckriet, *Hilbert space embeddings and metrics on probability measures*, J. Mach. Learn. Res., 11 (2010), pp. 1517–1561, http://jmlr.org/papers/v11/sriperumbudur10a.html.
[44] D. Talay, *Second-order discretization schemes of stochastic differential systems for the computation of the invariant law*, Stochastics and Stochastic Reports, 29 (1990), pp. 13–36, https://doi.org/10.1080/17442509008833606.
[45] X. Tang and A. Xiao, *Asymptotically optimal approximation of some stochastic integrals and its applications to the strong second-order methods*, Adv. Comput. Math., 45 (2019), pp. 813–846, https://doi.org/10.1007/s10444-018-9638-0.

[46] K. Terada, *Higher Order Methods and Multilevel Monte Carlo for SDEs in Finance*, Masters thesis, University of Edinburgh, 2022.

[47] M. Wiktorsson, *Joint characteristic function and simultaneous simulation of iterated Itô integrals for multiple independent Brownian motions*, Ann. Appl. Probab., 11 (2001), pp. 470–487, https://doi.org/10.1214/aoap/1015345301.