Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2024-0201

Vol. **40**, No. 1, pp. 1-26
July 2026

Score-fPINN: Fractional Score-Based Physics-Informed Neural Networks for High-Dimensional Fokker-Planck-Lévy Equations

Zheyuan Hu[1,*], Zhongqiang Zhang[2], George Em Karniadakis[3,4] and Kenji Kawaguchi[1]

[1] *Department of Computer Science, National University of Singapore, Singapore, 119077.*
[2] *Department of Mathematical Sciences, Worcester Polytechnic Institute, Worcester, MA 01609 USA.*
[3] *Division of Applied Mathematics, Brown University, Providence, RI 02912, USA.*
[4] *Advanced Computing, Mathematics and Data Division, Pacific Northwest National Laboratory, Richland, WA, United States.*

Received 20 August 2024; Accepted (in revised version) 8 January 2025

Abstract. We introduce an innovative approach for solving high-dimensional Fokker-Planck-Lévy (FPL) equations in modeling non-Brownian processes across disciplines such as physics, finance, and ecology. We utilize a fractional score function and Physical-informed neural networks (PINN) to lift the curse of dimensionality (CoD) and alleviate numerical overflow from exponentially decaying solutions with dimensions. The introduction of a fractional score function allows us to transform the FPL equation into a second-order partial differential equation without fractional Laplacian and thus can be readily solved with standard physics-informed neural networks (PINNs). We propose two methods to obtain a fractional score function: fractional score matching (FSM) and score-fPINN for fitting the fractional score function. While FSM is more cost-effective, it relies on known conditional distributions. On the other hand, score-fPINN is independent of specific stochastic differential equations (SDEs) but requires evaluating the PINN model's derivatives, which may be more costly. We conduct our experiments on various SDEs and demonstrate numerical stability and effectiveness of our method in dealing with high-dimensional problems, marking a significant advancement in addressing the CoD in FPL equations. Code is available at `https://github.com/zheyuanhu01/Score-fPINN`.

AMS subject classifications: 65M75, 65C30, 68T07

Key words: Physics-informed neural networks, high dimensional partial differential equations, curse of dimensionality, Fokker-Planck equations, Fokker-Planck-Lévy equations, score function, score matching.

*Corresponding author. *Email addresses:* `e0792494@u.nus.edu` (Z. Hu), `zzhang7@wpi.edu` (Z. Zhang), `george_karniadakis@brown.edu` (G. E. Karniadakis), `kenji@nus.edu.sg` (K. Kawaguchi)

1 Introduction

The Fokker-Planck-Lévy (FPL) equation, also known as the fractional Fokker-Planck equation, is a generalization of the traditional Fokker-Planck (FP) equation to incorporate Lévy processes, particularly those involving jumps or heavy-tailed distributions. The FPL equation is used in fields like physics for anomalous diffusion in complex systems, finance for pricing derivatives when the underlying asset exhibits jumps or heavy tails, and ecology for animal movement patterns that involve sudden, long-range moves. The classic Fokker-Planck equation describes the time evolution of the probability density function of the velocity of a particle under the influence of forces and Gaussian white noises. However, many physical and economic phenomena exhibit jumps and heavy tails, which are not adequately described by Gaussian processes. Lévy processes, which include a broader class of stochastic processes characterized by stable distributions and jumps, offer a more appropriate mathematical framework for such scenarios. The FPL equation extends the traditional Fokker-Planck equation by incorporating fractional derivatives, which can model these non-local, jump-like dynamics.

Despite its importance, obtaining numerical solutions to the FPL equation is still challenging due to the non-locality introduced by the fractional derivative, which requires special numerical schemes that can handle integral terms effectively and the need for handling both the small-scale behavior driven by the diffusion term as well as the large discrete changes introduced by the jump term.

The higher-dimensional FPL equations of interest in this paper pose more significant challenges owing to the curse of dimensionality (CoD), where traditional grid-based methods fail due to the exponential increase in computational requirements with the dimensionality of the PDE, rendering them unrealistic. Consider another branch of the traditional method, namely Monte Carlo simulation, which can tackle the CoD in certain PDEs. Though Monte Carlo methods can solve the FP equation with the Feynman–Kac formula and the corresponding stochastic differential equation (SDE), they solve the problem at one point and are also expensive when the solution at a large region is desired.

Physics-informed neural networks [40] have become popular in solving high-dimensional PDEs and tackling the CoD thanks to neural networks' strong universal approximation property [1], generalization capacities [21], robust optimization [28], and PINN's meshless plus grid-free training. Multiple methods [22, 23] recently have proposed to scale up and speed up PINNs to very high dimensions using random sampling. Although PINN offers the possibility of addressing the CoD in certain cases, in FPL equations, PINN accuracy is limited to moderately high dimensions (e.g., less than ten dimensions [9]) for computing probability density functions (PDFs) of interest in the FPL equations. They also exhibit significant numerical errors at higher dimensions, rendering them impractical. Specifically, FPL equations model PDFs and the most common Gaussian PDFs associated with Brownian motion exhibit exponential decay in numerical values as dimensionality increases. This phenomenon easily surpasses the numerical precision of computer simulations, leading to significant errors in PINN numerical solvers. The

heavy-tailed nature of Levy noise results in rare events with extremely small PDF values, further amplifying the numerical errors of vanilla PINNs. Furthermore, the above high-dimensional PINNs [22,23] works are based on integer-order derivatives and employ automatic differentiation to compute integer-order derivatives. However, fractional-order derivatives are not yet covered by automatic differentiation libraries. Approximating the *high-dimensional* fractional Laplacian is still an open problem. Therefore, tackling high-dimensional FPL equations remains a challenging problem, whether for traditional methods or emerging techniques like PINNs.

To this end, we propose utilizing a fractional score-based SDE/FPL equation solver to fit the fractional score function in the SDE, which broadens the definition of the score function used in the FP equation [25,27,44]. We demonstrate its numerical stability and the fundamental role of fractional score in solving the FPL equation's SDE, indicating its capability to accurately infer the log-likelihood (LL) and PDF of interest. Concretely, with the obtained fractional score, we can eliminate the fractional Laplacian in the FPL equation, transforming it into an FP equation that can be solved using standard PINNs [22,23,40]. The second-order FP equation enables subsequent fitting of LL and PDF via standard PINNs afterward.

We introduce two methods for fitting the crucial fractional score function: fractional score matching (FSM) [49] and score-fPINN. FSM relies on the conditional distribution modeled by the SDE and it minimizes the mean squared error between the fractional score model and the true conditional fractional score along SDE trajectories. One can prove that this is equivalent to minimizing the mean squared error between the fractional score model and the true fractional score [49]. When the conditional distribution modeled by the SDE is unknown, we use Score-fPINN which is independent of the SDE form. Specifically, Score-fPINN first utilizes Sliced Score Matching (SSM) [43] to obtain the conventional score function, which is then inserted into the FPL equation to simplify it. Then, the only unknown will be the fractional score function, which can be obtained by enforcing the PINN's PDE loss. Once the score function is obtained through one of the above methods, we simplify the FPL equation to a second-order FP equation, allowing easy calculation of the LL using standard PINN methods. In other words, our fractional score-based SDE/FPL equation solver consists of two stages. The first stage involves FSM or Score-fPINN to obtain the fractional score function. The second stage requires using the obtained fractional score function to solve LL or PDF via a second-order FP equation, which is obtained from plugging the fractional score function into the original FPL equation. In comparison between these two methods, FSM is more concise and efficient but requires the known conditional distribution from the SDE. Score-fPINN employs PINN loss and thus necessitating the calculation of derivatives of the neural network model with respect to the input. Thus, Score-PINN is more expensive while score-fPINN applies to a broader range of SDE types.

We evaluate the fractional score-based SDE/FPL equation solver on different SDEs. We test the basic case, namely, an anisotropic SDE with both Brownian and Lévy noise, and then test more challenging problems by complicating its diffusion and drift coefficients.

We also vary the initial distribution to assess the capability of our framework to fit different distributions. Experimental results demonstrate the stability of the fractional score-based SDE solver in various experimental settings. The proposed methods are sublinear in speed, and their performance remains stable across dimensions, which demonstrates the ability of the fractional score-based SDE solver to overcome CoD in FPL equations.

To the best of our knowledge, we introduce the concept of fractional score function and the Score-fPINN in solving high-dimensional FPL equations to the scientific machine learning community for the first time.

2 Related work

2.1 PINN for FP and FPL equations

FP and FPL equations are prevalent in statistical mechanics, and their high dimensionality poses significant challenges to traditional analytical methods such as finite difference [10, 41]. In contrast, machine learning techniques, particularly physics-informed neural networks (PINNs), offer a promising mesh-free solution capable of addressing the CoD and integrating observational data smoothly. Research by Chen et al. [9] involved using PINNs to address both forward and inverse issues associated with Fokker-Planck-Lévy equations. Similarly, Zhang et al. [52] applied deep KD-tree methods to tackle FP equations in scenarios with sparse data. Zhai et al. [51] and Wang et al. [48] utilized deep learning for solving steady-state FP equations, while Lu et al. [36] focused on learning high-dimensional multivariate probability densities modeled by FP equations using normalizing flows. Furthermore, Feng et al. [11] and Guo et al. [17] both employed normalization flow techniques for FP equations, and Tang et al. [46] introduced an adaptive deep density approximation method based on normalizing flows for steady-state FP equations. Hu et al. [25] introduced a score-based SDE solver for FP equations, and herein we introduce their methodology to the fractional score and FPL equation settings. Zeng et al. [50] propose an adaptive normalizing flow for fractional Fokker-Planck equation solutions, which approximates the fractional Laplacian by either Monte Carlo or Gaussian radial basis functions, together with a solution alternative refinement algorithm. Feng et al. [12] apply the temporal normalizing flow for Fokker-Planck equations based on mesh-free PDE loss function and conduct extensive experiments with linear or nonlinear drift terms.

2.2 High-dimensional PDE solvers

Numerous techniques have been developed to overcome the curse of dimensionality (COD) in solving high-dimensional partial differential equations (PDEs): physics-informed neural networks (PINNs) [40, 42], backward stochastic differential equations (BSDE) [19], and the Multilevel Picard method [3]. Specifically, using the PINN framework, He et al. [20] introduced the randomized smoothing PINN (RS-PINN), which leverages Monte

Carlo simulations and Stein's identity to evaluate derivatives, thus bypassing expensive automatic differentiation techniques. Subsequently, Zhao et al. [53] suggested replacing Monte Carlo simulations with sparse quadratures to lower variance, while Hu et al. [24] investigated the bias-variance dilemma in RS-PINN. Stochastic dimension gradient descent (SDGD) [23] reduces memory usage and hastens convergence by sampling subsets of dimensions for gradient descent when training PINNs. Meanwhile, Hutchinson trace estimation (HTE) [22] offers an alternative to the high-dimensional Hessian by a Hessian vector product based on HTE in the PINN loss function to speed up the process. Regarding BSDE, the backward stochastic differential equations method [19] and deep splitting approach [2] integrate deep learning with traditional techniques for solving parabolic PDEs, allowing for the modeling of unknown functions within these established frameworks. Lastly, the multilevel Picard method [3, 26] addresses nonlinear parabolic PDEs under specific regularity conditions for convergence.

2.3 Fractional PDE solvers

Pang et al. [37] propose fractional PINN (fPINN), adopting neural networks as solution surrogates and discretizing the fractional derivative for supervision. Guo et al. [16] propose Monte Carlo fPINN (MC-fPINN) estimating Caputo-type time-fractional derivatives and fractional Laplacian in the hyper-singular integral representation using Monte Carlo from Beta distributions. Firoozsalari et al. [13] consider Gaussian quadrature to compute the numerical integral related to fractional derivative. Leonenko and Podlubny [33] propose a Monte Carlo-based estimator for the Grünwald–Letnikov fractional derivative. While these methods, especially MC-fPINN [16], effectively approximate the high-dimensional fractional Laplacian in FPL equations, they are still constrained by the numerical instability caused by the exceedingly small values of the PDF modeled by the FPL equation.

2.4 Score-based generative models

Song et al. [44] highlighted the relationship between diffusion generative models and SDEs. These SDEs inject noise into data sets, such as images and texts, converting them to a pure Gaussian state. The reverse process of the SDE then removes the noise to restore the original data distribution. Central to this mechanism is the derivation of the score function, which is the gradient of the log-likelihood of a distribution, an approach known as score matching (SM). Various techniques for score matching have been developed. For instance, Song et al. [44] developed methods for matching the conditional score function, a technique that is mathematically on par with direct score matching [27]. Moreover, Song et al. [43] introduced sliced score matching (SSM), achieving objectives similar to direct score matching but without necessitating knowledge of the underlying distribution. Finite difference score matching [38] further lessens the computational burden associated with sliced score matching by sidestepping the intensive computation of gradients in the score function in SSM. Additionally, Lai et al. [31] explored the partial differential equation that

governs the score function in FP equations, suggesting the use of physics-informed neural networks (PINNs) [40] to streamline the score-matching optimization process. Boffi and Vanden-Eijnden [5,6] adopt score-based solvers to time-dependent and time-independent Fokker-Planck equations. [49] extend the score matching based on SDE with Brownian motion to the Lévy process, transforming the data distribution to stable distributions, and further propose the corresponding fractional score matching, which is used to reverse the Lévy process for data generation from random Lévy noise.

3 Proposed method

This section presents the methodology of employing a fractional score-based model to address SDE forward problems with Brownian and Lévy noises. We list abbreviations and notations in Tables 1 and 2.

Table 1: List of abbreviations.

Abbreviation	**Explanation**
CoD	Curse-of-Dimensionality
PDE	Partial Differential Equation
ODE	Ordinary Differential Equation
SDE	Stochastic Differential Equation
PDF	Probability Density Function
LL	Logarithm Likelihood
PINN	Physics-Informed Neural Network
fPINN	Fractional Physics-Informed Neural Network
HTE	Hutchinson Trace Estimation
HJB	Hamilton-Jacobi-Bellman
FP	Fokker-Planck
FPL	Fokker-Planck-Lévy
SM	Score Matching
SSM	Sliced Score Matching
FSM	Fractional Score Matching
OU	Ornstein–Uhlenbeck
AI	Artificial Intelligence

Table 2: List of notations.

Notation	Explanation
$\mathcal{S}\alpha\mathcal{S}^d(\gamma)$	d-dimensional α-stable distribution with parameter γ
$p_t(\boldsymbol{x})$	Probability density function (PDF) concerning time t and input $\boldsymbol{x}$
$\boldsymbol{f}(\boldsymbol{x},t)$	Drift coefficient of the SDE
$\boldsymbol{G}(\boldsymbol{x},t)$	Diffusion coefficient of the SDE
$\sigma(t)$	Coefficient for the Lévy noise in the SDE
$\boldsymbol{A}^\alpha(\boldsymbol{x},t)$	$\boldsymbol{A}^\alpha(\boldsymbol{x},t):=\boldsymbol{f}(\boldsymbol{x},t)-\frac{1}{2}\nabla\cdot\left[\boldsymbol{G}(\boldsymbol{x},t)\boldsymbol{G}(\boldsymbol{x},t)^{\mathrm{T}}\right]-\sigma(t)\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x})$
$\boldsymbol{w}_t$	Brownian motion
$\boldsymbol{L}_t^\alpha$	Lévy process
$q_t(\boldsymbol{x})$	Logarithm likelihood (LL) of $p_t(\boldsymbol{x})$, i.e., $q_t(\boldsymbol{x})=\log p_t(\boldsymbol{x})$
$q_t(\boldsymbol{x};\phi)$	PINN model parameterized by ϕ to model LL
$\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x})$	Fractional score function defined as $\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x})=\frac{(-\Delta)^{\frac{\alpha-2}{2}}\nabla p_t(\boldsymbol{x})}{p_t(\boldsymbol{x})}$
$\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x};\theta)$	Fractional score function PINN model parameterized by θ, i.e., Score-fPINN

3.1 Problem definition and background

3.1.1 Stable distribution

Definition 3.1. (α-stable Lévy distribution) If a random variable $X\sim\mathcal{S}\alpha\mathcal{S}^d(\gamma)\in\mathbb{R}^d$, then its characteristic function $\mathbb{E}[\exp(i\langle\boldsymbol{k},X\rangle)]=\exp(-\gamma^\alpha\|\boldsymbol{k}\|^\alpha)$.

Here are some examples of analytical Lévy distributions:

- If $\alpha=2$, then $\mathcal{S}\alpha\mathcal{S}^d(\gamma)=\mathcal{N}(0,2\gamma^2 I)$ is Gaussian.
- If $\alpha=1$, then $\mathcal{S}\alpha\mathcal{S}^d(\gamma)$ is the Cauchy distribution with the PDF [30,32]:

$$p(\boldsymbol{x};\gamma)=\frac{\Gamma\left(\frac{1+d}{2}\right)}{\Gamma\left(\frac{1}{2}\right)(\gamma^2\pi)^{\frac{d}{2}}\left[\left(1+\frac{\|\boldsymbol{x}\|^2}{\gamma^2}\right)\right]^{\frac{d+1}{2}}}. \tag{3.1}$$

  Its characteristic function is $\phi(\boldsymbol{k})=\exp(-\gamma\|\boldsymbol{k}\|)$.

- If $\alpha=1.5$, then $\mathcal{S}\alpha\mathcal{S}^d(\gamma)$ is the Holtsmark distribution whose PDF can be expressed via hypergeometric functions.
- For stable distribution with other α, the PDF does not have a closed-form expression.

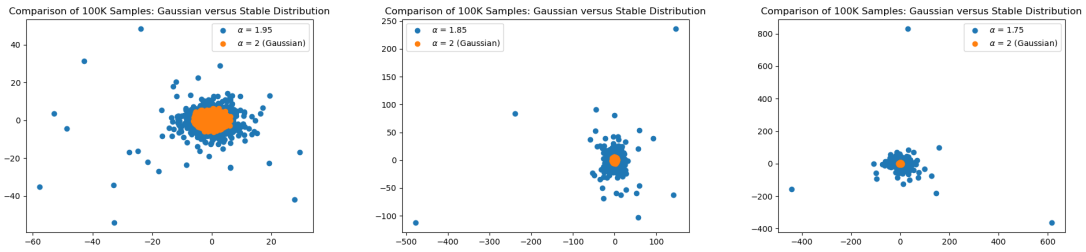

Figure 1: We generate 100,000 random samples from stable distributions with various α in 2D for visualization. The long tail of $\alpha<2$ stable distributions exhibit a much larger support than the short tail of Gaussian with $\alpha=2$. Rare events far from the original point are also common for stable distributions with $\alpha<2$.

When $\alpha=2$, the Gaussian distribution has an exponentially short tail, e.g., for the 1D Gaussian $X\sim\mathcal{N}(0,\sigma^2)$, its tail bound is $\mathbb{P}(X\geq r)\leq\exp\left(-\frac{r^2}{2\sigma^2}\right)$. Modeling stable distributions is difficult due to its long tail property when $\alpha<2$. When $\alpha<2$, let $\boldsymbol{X}$ follow the d-dimensional α-stable distribution and it has a long tail, i.e., $\mathbb{P}(\|\boldsymbol{X}\|_2\geq r)\sim r^{-\alpha}$. Thus, rare events are relatively common for stable distributions compared with the short-tailed Gaussian, as shown in Fig. 1. Due to the long tail, for $\alpha<2$, the variance of α-stable distribution is infinite.We mainly focus on $1<\alpha<2$.

3.1.2 Lévy process

Furthermore, a Lévy process $\boldsymbol{L}_t$ is a stochastic process that generalizes Brownian motion by allowing non-Gaussian increments.

Definition 3.2. (Lévy Process) An $\mathbb{R}^d$-valued stochastic process $\boldsymbol{L}_t$ is a Lévy process if

1. $\boldsymbol{L}_0=0$ almost surely.
2. $\boldsymbol{L}_t$ has independent increments.
3. $\boldsymbol{L}_t$ has stationary increments.
4. The sample paths of the process are stochastically continuous.

Under the α-stable Lévy distribution, the isotropic α-stable Lévy process $\boldsymbol{L}_t^\alpha$ under consideration satisfies that for all $s<t$, $\boldsymbol{L}_t^\alpha-\boldsymbol{L}_s^\alpha=\boldsymbol{L}_{t-s}^\alpha\sim\mathcal{S}\alpha\mathcal{S}^d\left((t-s)^{1/\alpha}\right)$ in distribution.

The PDF takes extremely small values in most regions and decays exponentially with dimensions and thus it causes numerical error to conventional PINNs [40]. Moreover, The heavy-tail property of α-stable Lévy noise leads to the huge support of the PDFs, making domain truncation impractical.

3.1.3 Fractional SDE and Fokker-Planck-Lévy equation

Then, we consider SDEs with Brownian and α-stable Lévy noises:

$$d\boldsymbol{X} = \boldsymbol{f}(\boldsymbol{X},t)dt + \boldsymbol{G}(\boldsymbol{X},t)d\boldsymbol{w}_t + \sigma(t)d\boldsymbol{L}_t^{\alpha}, \tag{3.2}$$

where $\boldsymbol{L}_t^{\alpha}$ is the α-stable Lévy process in $\mathbb{R}^d$ and $\boldsymbol{X} \in \mathbb{R}^d$ is the state variable, $t \in [0,T]$ where T is the terminal time. Here all the coefficients are known: $\boldsymbol{f}(\boldsymbol{x},t): \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, $\boldsymbol{G}(\boldsymbol{x},t): \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^{d\times d}$, and $\sigma(t): \mathbb{R} \to \mathbb{R}$. We want to solve the forward problem, i.e., given the initial distribution $p_0(\boldsymbol{x})$ at $t=0$ and the SDE coefficients, we solve the evolution of the distribution of $\boldsymbol{x}$ on $[0,T]$. The PDF for $\mathbf{X}$ satisfies the following fractional FPL equation:

$$\begin{aligned}\partial_t p_t(\boldsymbol{x}) = &-\sum_{i=1}^{d} \frac{\partial}{\partial \boldsymbol{x}_i}[\boldsymbol{f}_i(\boldsymbol{x},t)p_t(\boldsymbol{x})] + \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j}\left[\sum_{k=1}^{d} \boldsymbol{G}_{ik}(\boldsymbol{x},t)\boldsymbol{G}_{jk}(\boldsymbol{x},t)p_t(\boldsymbol{x})\right] \\ &-\sigma(t)(-\Delta)^{\frac{\alpha}{2}} p_t(\boldsymbol{x}),\end{aligned} \tag{3.3}$$

where the fractional Laplacian $(-\Delta)^{\frac{\alpha}{2}}$ is formally defined as follows using the Fourier transform [34,45].

Definition 3.3. Denote the Fourier transform of the function f as $\mathcal{F}\{f\}(\boldsymbol{k}) = \int_{\mathbb{R}^d} \exp(i\langle \boldsymbol{x},\boldsymbol{k}\rangle) f(\boldsymbol{x})d\boldsymbol{x}$. The Fractional Laplacian for $\alpha \in (0,2)$ is defined as

$$(-\Delta)^{\frac{\alpha}{2}} f(\boldsymbol{x}) = \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} \|\boldsymbol{k}\|^{\alpha} \exp(-i\langle \boldsymbol{x},\boldsymbol{k}\rangle)\mathcal{F}\{f\}(\boldsymbol{k})d\boldsymbol{k}. \tag{3.4}$$

Direct computation of the fractional Laplacian to solve the fractional FPL equation corresponding to the Lévy process suffers from the curse of dimensionality due to the high-dimensional integral for traditional numerical schemes [14].

3.2 Fractional score function

We introduce the fractional score function $\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x})$ that generalizes the conventional score function [27,44] to enable the solution of the LL and circumvent CoD. Specifically, the fraction score functions for an underlying PDF $p_t(\boldsymbol{x})$ are defined as follows [49]:

$$\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x}) = \frac{(-\Delta)^{\frac{\alpha-2}{2}}\nabla p_t(\boldsymbol{x})}{p_t(\boldsymbol{x})}. \tag{3.5}$$

If $\alpha = 2$, it becomes a vanilla score function, and the Lévy process is Brownian, and $\boldsymbol{S}_t^{(2)}(\boldsymbol{x}) = \nabla \log p_t(\boldsymbol{x}) = \frac{\nabla p_t(\boldsymbol{x})}{p_t(\boldsymbol{x})}$. The vanilla score function of multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ is $-\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$. It converts the inseparable multivariate Gaussian PDF to a linear score function. Similarly, the fractional score of any Lévy distribution is also a linear function.

Theorem 3.1 (Yoon et al. [49]). *If a random variable $\boldsymbol{x} \sim \mathcal{S}\alpha\mathcal{S}^d(\gamma) \in \mathbb{R}^d$ and its PDF is denote $p(\boldsymbol{x})$, then its fractional score is $\boldsymbol{S}^{(\alpha)}(\boldsymbol{x}) = \frac{(-\Delta)^{\frac{\alpha-2}{2}}\nabla p(\boldsymbol{x})}{p(\boldsymbol{x})} = -\frac{\boldsymbol{x}}{\alpha\gamma^{\alpha}}$.*

Compared with PDF, whose value decays exponentially with SDE dimensionality, the fractional score function's scale is invariant with dimension, making it numerically stable and preventing the PINN training from numerical overflow.

Another pivotal advantage of the fractional score is that the fractional Laplacian can be simplified as the negative of the divergence operator. By Definition 3.3 of the fractional Laplacian and by the Fourier transform (see also Lemma C.1 in [49]),

$$(-\Delta)^{\frac{\alpha}{2}} p_t(\boldsymbol{x}) = -\nabla\cdot\Big((-\Delta)^{\frac{\alpha-2}{2}}\nabla p_t(\boldsymbol{x})\Big) = -\nabla\cdot\Big(p_t(\boldsymbol{x})S_t^{(\alpha)}(\boldsymbol{x})\Big). \tag{3.6}$$

Then, we can rewrite the FPL equation as follows

$$\begin{aligned}\partial_t p_t(\boldsymbol{x}) = &-\nabla_{\boldsymbol{x}}\cdot\Big[\Big(\boldsymbol{f}(\boldsymbol{x},t)-\sigma(t)^{\alpha}\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x})\Big)p_t(\boldsymbol{x})\Big]\\ &-\frac{1}{2}\sum_{i,j=1}^{d}\frac{\partial^2}{\partial \boldsymbol{x}_i\partial \boldsymbol{x}_j}\left[\sum_{k=1}^{d}\boldsymbol{G}_{ik}(\boldsymbol{x},t)\boldsymbol{G}_{jk}(\boldsymbol{x},t)p_t(\boldsymbol{x})\right].\end{aligned} \tag{3.7}$$

Taking ∇ at both sides and dividing by $p_t(\boldsymbol{x})$ gives a second-order PDE (LL-PDE) for the LL $q=q_t(\boldsymbol{x})=\log p_t(\boldsymbol{x})$:

$$\partial_t q = \frac{1}{2}\nabla_{\boldsymbol{x}}\cdot(\boldsymbol{G}\boldsymbol{G}^{\mathrm{T}}\nabla_{\boldsymbol{x}}q)+\frac{1}{2}\|\boldsymbol{G}^{\mathrm{T}}\nabla_{\boldsymbol{x}}q\|^2-\langle\boldsymbol{A}^{\alpha},\nabla_{\boldsymbol{x}}q\rangle-\nabla_{\boldsymbol{x}}\cdot\boldsymbol{A}^{\alpha}:=\mathcal{L}_{\text{LL-PDE}}\left[q,\boldsymbol{S}_t^{(\alpha)}\right], \tag{3.8}$$

where

$$\boldsymbol{A}^{\alpha}(\boldsymbol{x},t)=\boldsymbol{f}(\boldsymbol{x},t)-\frac{1}{2}\nabla\cdot\left[\boldsymbol{G}(\boldsymbol{x},t)\boldsymbol{G}(\boldsymbol{x},t)^{\mathrm{T}}\right]-\sigma(t)\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x}). \tag{3.9}$$

Here, we define the LL-PDE operator $\mathcal{L}_{\text{LL-PDE}}\left[q,\boldsymbol{S}_t^{(\alpha)}\right]$ since $\boldsymbol{f},\boldsymbol{G},\sigma(t)$ are known and only the LL q and the fractional score function $\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x})$ will be the input variables. Hence, it is evident that once we acquire the fractional score, solving the LL can be realized by using PINNs for this second-order PDE. It is then crucial to obtain the fractional score and we will introduce two methods to obtain it: fractional score matching and score fractional PINN (Score-fPINN).

3.3 Fractional Score Matching (FSM)

Fractional Score Matching (FSM) aims to parameterize $\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x};\theta)$ to approximate the fractional score function $\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x})$, and minimize their L_2 distance:

$$L^{\alpha}_{\text{Oracle-SM}}(\theta)=\mathbb{E}_{t\sim\text{Unif}[0,T]}\mathbb{E}_{\boldsymbol{x}\sim p_t(\boldsymbol{x})}\left[\left\|\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x};\theta)-\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x})\right\|^2\right]. \tag{3.10}$$

However, the exact fractional score $\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x})$ is given by the exact SDE solution, which is unknown. So, computing the objective function $L^{\alpha}_{\text{Oracle-SM}}(\theta)$ is unrealistic. Yoon et al. [49]

demonstrate that minimizing $L^{\alpha}_{\text{Oracle-SM}}(\theta)$ is equivalent to minimizing conditional score matching loss function.

$$L^{\alpha}_{\text{Cond-SM}}(\theta)=\mathbb{E}_{t\sim\text{Unif}[0,T]}\mathbb{E}_{\boldsymbol{x}_0\sim p_0(\boldsymbol{x})}\mathbb{E}_{\boldsymbol{x}|\boldsymbol{x}_0\sim p_{0t}(\boldsymbol{x}|\boldsymbol{x}_0)}\left[\left\|\boldsymbol{S}^{(\alpha)}_t(\boldsymbol{x};\theta)-\boldsymbol{S}^{(\alpha)}_t(\boldsymbol{x}|\boldsymbol{x}_0)\right\|^2\right], \tag{3.11}$$

where $\boldsymbol{S}^{(\alpha)}(\boldsymbol{x}|\boldsymbol{x}_0)=\frac{(-\Delta)^{\frac{\alpha-2}{\alpha}}p_{0t}(\boldsymbol{x}|\boldsymbol{x}_0)}{p_{0t}(\boldsymbol{x}|\boldsymbol{x}_0)}$ is the conditional score function and $p_{0t}(\boldsymbol{x}|\boldsymbol{x}_0)$ is the conditional distribution given the starting point $\boldsymbol{x}_0$ at $t=0$. Conditioned on the starting point $\boldsymbol{x}_0$, $p_{0t}(\boldsymbol{x}|\boldsymbol{x}_0)$ can be analytically obtained for common SDEs, such as Lévy processes and the widely-used Ornstein–Uhlenbeck (OU) processes in image generation [49]. Specifically, these stochastic processes typically exhibit α-stable distributions as their conditional distributions, with their conditional fractional score functions being linear. After obtaining the fractional score, we solve the LL-PDE (3.8) with the standard PINN.

Although we can analytically obtain the conditional score function, we cannot obtain the unconditioned exact score function for Lévy and the Ornstein–Uhlenbeck (OU) processes. Even if we can derive the exact solution from the conditional score function, this process usually involves extensive Monte Carlo simulation to transform the conditional one into the unconditioned one. It becomes extremely costly due to the massive sampling of Gaussian and Lévy noises and Euler-Maruyama SDE discretization, preventing real-time fast inference. On the other hand, PINN is trained once-for-all and can conduct fast prediction after one round of training. In contrast, if there is no analytical conditional distribution, traditional methods fail due to high dimensions [4, 18], and the only solution becomes our Score-fPINN, which works for general SDEs without known conditional distributions. Score-fPINN will be introduced in the next subsection.

3.4 Score fractional PINN (Score-fPINN)

As FSM works only when conditional distributions are known, we propose the Score-fPINN to solve general SDEs regardless of the conditional distributions. Specifically, the Score-fPINN initially employs the Sliced Score Matching (SSM) to obtain the vanilla score function of the $p_t(\boldsymbol{x})$. Subsequently, this vanilla score function is integrated into the FPL equation, enabling the derivation of the fractional score via the standard PINN approach.

Observing that $\boldsymbol{S}^{(2)}_t(\boldsymbol{x})=\nabla q=\nabla\log p_t(\boldsymbol{x})$, we obtain from the LL-PDE (3.8) that

$$\begin{aligned}\partial_t\boldsymbol{S}^{(2)}_t(\boldsymbol{x})=&\nabla_{\boldsymbol{x}}\left[\frac{1}{2}\nabla_{\boldsymbol{x}}\cdot(\boldsymbol{G}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{S}^{(2)}_t)+\frac{1}{2}\|\boldsymbol{G}^{\mathrm{T}}\boldsymbol{S}^{(2)}_t\|^2-\langle\boldsymbol{A}^{\alpha},\boldsymbol{S}^{(2)}_t\rangle-\nabla_{\boldsymbol{x}}\cdot\boldsymbol{A}^{\alpha}\right]\\ :=&\mathcal{L}_{\text{Score-fPDE}}\left[\boldsymbol{S}^{(2)}_t,\boldsymbol{S}^{(\alpha)}_t\right],\end{aligned} \tag{3.12}$$

where $\boldsymbol{A}^{\alpha}$ is defined in Eq. (3.9). We call this PDE Score-fPDE, where $\boldsymbol{f}$, $\boldsymbol{G}$, and σ are known. The operator $\mathcal{L}_{\text{Score-fPDE}}$ takes the two scores as input variables. Thus, once we know $\boldsymbol{S}^{(2)}_t(\boldsymbol{x})$, then $\boldsymbol{S}^{\alpha}_t(\boldsymbol{x})$ inside $\boldsymbol{A}^{\alpha}$ can be solved by learning from the PDE above since it is the only unknown in the PDE. Fortunately, the vanilla score $\boldsymbol{S}^{(2)}_t(\boldsymbol{x})$ can be readily

learned using a well-developed technique [43], namely Sliced Score Matching (SSM). Concretely, the SSM loss function to obtain a vanilla score function model $\boldsymbol{S}_t^{(2)}\left(\boldsymbol{x};\theta^{(2)}\right)$ parameterized by $\theta^{(2)}$ such that $\boldsymbol{S}_t^{(2)}\left(\boldsymbol{x};\theta^{(2)}\right)\approx \boldsymbol{S}_t^{(2)}(\boldsymbol{x})$ is given by:

$$\theta^{(2)}=\arg\min_{\theta^{(2)}}\mathbb{E}_{t\sim\text{Unif}[0,T]}\mathbb{E}_{\boldsymbol{x}\sim p_t(\boldsymbol{x})}\left[\frac{1}{2}\left\|\boldsymbol{S}_t^{(2)}\left(\boldsymbol{x};\theta^{(2)}\right)\right\|^2+\nabla_{\boldsymbol{x}}\cdot\boldsymbol{S}_t^{(2)}\left(\boldsymbol{x};\theta^{(2)}\right)\right]. \tag{3.13}$$

Note that SSM does not assume anything about the SDE type. SSM only needs SDE samples from $p_t(\boldsymbol{x})$, which can be obtained from any numerical SDE discretization scheme [8,47].

After obtaining the vanilla score function $\boldsymbol{S}_t^{(2)}\left(\boldsymbol{x};\theta^{(2)}\right)\approx \boldsymbol{S}_t^{(2)}(\boldsymbol{x})$ via SSM. We solve for the fractional score $\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x})$ through score-fPINN on the score-fPDE:

$$\begin{aligned}&L_{\text{Score-fPINN}}(\theta)\\ =&\mathbb{E}_{t\sim\text{Unif}[0,T]}\mathbb{E}_{\boldsymbol{x}\sim p_t(\boldsymbol{x})}\left[\left(\partial_t\boldsymbol{S}_t^{(2)}\left(\boldsymbol{x};\theta^{(2)}\right)-\mathcal{L}_{\text{Score-fPDE}}\left[\boldsymbol{S}_t^{(2)}\left(\boldsymbol{x};\theta^{(2)}\right),\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x};\theta)\right]\right)^2\right].\end{aligned} \tag{3.14}$$

After obtaining the fractional score, we solve Eq. (3.8) via the standard PINN.

3.5 Methods summary and comparison

We have introduced fractional score matching (FSM) or score-fPINN to obtain the fractional score function. Then, we can infer LL by solving the LL PDE in Eq. (3.8) equipped with the fractional score. Similar to Score-PINN for FP equation [25], our fractional score-based SDE/FPL equation solver contains two stages. Algorithm 1 summarizes the above methodology. Following Score-PINN [25], we parameterize the score and LL models differently.

In comparison, these two methods' second steps are the same. After obtaining the fractional score by either FSM or score-fPINN in the first step, we plug it into the FPL equation and transformed FPL into a second-order PDE. Hence, FSM's and score-fPINN's computational costs are the same in the second stage, while only their first stages differ.

Regarding their first stages, FSM is more concise and efficient as its computational and loss function in Eq. (3.11) only requires the fractional score function model inference. However, it requires the known conditional distributions modeled by the SDE. In contrast, Score-fPINN does not require such conditional distributions. But it employs PINN loss in Eq. (3.14) and thus necessitates computing derivatives of the neural network model with respect to the input, which results in higher computational cost. Besides, the SSM loss in Eq. (3.13) to obtain the vanilla score function also requires the first-order derivative of the score function model. Thus, score-fPINN is much more expensive by design, while score-fPINN applies to a broader range of SDE types. Empirically, our experiments in the next section showed that FSM is generally 2 to 10 times faster than Score-fPINN, depending on the setting.

Algorithm 1 Fractional Score-based SDE solver.

1: Obtain the approximated fractional score function $\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x};\theta)\approx \boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x})$ via one of the two approaches:
   - Fractional score matching (FSM): $\theta=\operatorname{argmin}_\theta L^{\alpha}_{\text{Cond-SM}}(\theta)$ in Eq. (3.11) if the SDE conditional distribution is tractable.
   - Score-fPINN: First obtain the vanilla score function $\boldsymbol{S}_t^{(2)}\left(\boldsymbol{x};\theta^{(2)}\right)\approx \boldsymbol{S}_t^{(2)}(\boldsymbol{x})$ from SSM loss in Eq. (3.13). Then, optimize score-fPINN $\theta=\operatorname{argmin}_\theta L_{\text{Score-fPINN}}(\theta)$ in Eq. (3.14).
2: Parameterize the LL model $q_t(\boldsymbol{x};\phi)$.
3: Obtain the LL by solving the LL-PDE (3.8) with $\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x};\theta)$ as the score function, $\phi=\operatorname{argmin}_\phi L_{\text{LL-PDE}}(\phi)$, where

$$\begin{aligned} L_{\text{LL-PDE}}(\phi)=&\lambda_{\text{initial}}\cdot\mathbb{E}_{\boldsymbol{x}\sim p_0(\boldsymbol{x})}\left[\left(q_0(\boldsymbol{x};\phi)-\log p_0(\boldsymbol{x})\right)^2\right]\\ &+\lambda_{\text{residual}}\cdot\mathbb{E}_{t\sim\text{Unif}[0,T]}\mathbb{E}_{\boldsymbol{x}\sim p_t(\boldsymbol{x})}\left[\left(\partial_t q_t(\boldsymbol{x};\phi)-\mathcal{L}_{\text{LL-PDE}}\left[q_t(\boldsymbol{x};\phi),\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x};\theta)\right]\right)^2\right]. \end{aligned} \tag{3.15}$$

3.6 Technical details

Due to the heavy-tail of α-stable distribution as shown in Fig. 1, the FPL equation with Lévy noise and fractional Laplacian poses more challenges than the classical FP.

First, the fractional score of a common distribution is usually unknown due to the computationally costly fractional Laplacian. When $\alpha=2$, the vanilla score function of the initial distribution $\boldsymbol{S}_{t=0}^{(2)}(\boldsymbol{x})$ is usually known and can serve as the hard constraint to regularize the vanilla score function PINN model. When $\alpha<2$, the fractional score requires numerical computations and thus introduces extra errors.

Second, we use **smooth L_1 loss** instead of L_2 loss to robustify the optimization with extreme values/rare events produced by Lévy noise following Yoon et al. [49]. Concretely, the smooth L_1 loss has a hyperparameter β whose form is:

$$\ell(\hat{y},y)=\begin{cases}|\hat{y}-y|^2 & \text{if } |\hat{y}-y|<\beta,\\ 2\beta|\hat{y}-y|-\beta^2 & \text{if } |\hat{y}-y|\geq\beta.\end{cases} \tag{3.16}$$

Here, $\hat{y}$ denotes the model prediction, and y is the ground truth label. When $\beta\to\infty$, it converges to L_2 loss. When the prediction and label diverge significantly ($|\hat{y}-y|\geq\beta$), such as when rare events or extreme values occur in the Lévy process, employing an L_1 loss generates gradients with a smaller scale, ensuring stability. Conversely, when the prediction and label do not differ significantly ($|\hat{y}-y|<\beta$), a standard L_2 loss is used.

Last, we may match both $\boldsymbol{S}_t^{(2)}(\boldsymbol{x})$ and $\boldsymbol{S}_t^{(\alpha)}(\boldsymbol{x})$ like Score-PINN [25] to transform the FPL equation into ODE further. However, learning two score functions and plugging them into

the FPL equation will lead to substantial error. In practice, the inherent error introduced by inserting two score functions is significant to the extent that PINN cannot solve for the solution of the corresponding LL-ODE. Conversely, matching only one fractional score function proves accurate enough, enabling PINN to extract the likelihood from the corresponding LL-PDE.

3.7 Comparison with related work by Yoon et al.

We compare our fractional score-based SDE solver with the generative model based on the fractional score by Yoon et al. [49]. We employ the fractional score to solve the PDF and LL modeling by the SDE and FPL equations. On the other hand, the generative model generates data through the SDE without explicitly focusing on the PDF and LL, emphasizing the generated data's quality. Furthermore, when solving the SDE, we already know the SDE's initial conditions and form, particularly the drift and diffusion coefficients, which may be nonlinear or more complex. In contrast, in the generative model, the initial conditions are unknown data distributions. The generative model utilizes the SDE to transform the data distribution into pure Lévy noise and then reverses the SDE. The reverse SDE serves as a pathway from noise to images, enabling image generation, with its drift coefficient associated with the fractional score function. Hence, the generative model matches the fractional score to invert the SDE and generate data. Finally, the generative model employs relatively simple SDEs, such as Ornstein-Uhlenbeck (OU) processes and Lévy processes, to generate images since these processes provable convert any unknown data distribution to pure Lévy noises, while we can address more general SDEs. The comparisons and differing methodologies above also apply to the distinction between our SDE/FPL equation solver and other score-based generative models [44].

3.8 Comparison with alternative methods

We compare our method with other existing approaches, primarily traditional Monte Carlo methods, and some other machine learning techniques. Monte Carlo methods are essentially similar to FSM in that they can only be used when the conditional distribution of the SDE is known. Furthermore, Monte Carlo methods require large-scale sampling of Gaussian and Lévy noise during prediction, and the Euler-Maruyama discretization of the SDE leads to a large amount of sampling, making Monte Carlo methods very slow. In contrast, once our method is trained, it is capable of making fast predictions once and for all.

When compared to other machine learning approaches [9,11,13,16,17,36,46,48,51,52], our key contribution lies in the use of fractional score functions. These fractional score functions are numerically stable, especially in very high-dimensional cases, allowing them to be effectively handled by existing machine learning systems, such as Python's PyTorch [39]. Other machine learning methods directly handle probability density functions (PDF), whose values decay exponentially as the dimensionality of the problem increases.

This causes the values to quickly diminish beyond the computer's precision, rendering simulations computationally infeasible. Consequently, these other methods often suffer from numerical overflow and diverge. On the other hand, our method has been validated through extensive high-dimensional experiments (up to 100 dimensions) to demonstrate its stability and effectiveness.

Regarding benchmarking, we primarily focus on extremely high-dimensional cases, particularly those exceeding 10 dimensions. Existing machine learning methods often focus on lower-dimensional problems, typically in the range of 3-10 dimensions. In their cases, learning the probability density function (PDF) remains numerically stable because the dimensionality is relatively low. However, we extend our analysis up to 100 dimensions, where the scale of the probability density function becomes very small. Only our method is capable of handling such high-dimensional problems without suffering from numerical overflow.

4 Computational experiment

In this section, we conduct numerical experiments to evaluate the performance of our fractional score-based SDE solver under various experimental settings, including anisotropic Ornstein-Uhlenbeck (OU) processes and stochastic processes with dual noise components of Brownian and Lévy, ultimately testing SDE with nonlinear drift, which have a clever, analytical solution via elementary coordinate transformations, but whose conditional distributions are unknown.

High-dimensional Fokker-Planck-Lévy (FPL) equations are difficult to benchmark due to the lack of baseline methods. Traditional grid-based methods suffer from the curse of dimensionality [4, 18], and general FPL equations lack analytical solutions. Hence, we test analytically solvable SDEs or SDEs with a special structure, which we can solve accurately. We further manipulate various initial distributions to test the fractional score-matching approach. We also validate the effectiveness of details designed in the methodology, such as the smooth L_1 loss.

For the experiment in Sections 4.1 to 4.3, we tested SDE with known conditional distribution where both methods are applicable. In Section 4.4, we tested SDE with nonlinear drift without known conditional distribution, where only Score-fPINN can be applied.

We take $\alpha \in \{1.95, 1.85, 1.75\}$, dimension d in $\{10, 20, 50, 100\}$. We will investigate the effect of dimension and α on our methods' performances. The score models in FSM and score-fPINN are all four-layer fully connected networks whose input and output dims are the same as the SDE dimensionality d, while the hidden dimension is 128 for all cases. The LL model is also a 4-layer fully connected network whose input dimension is d, output dimension is 1, and hidden dimension is 128 for all cases. We adopt the following model structure to satisfy the initial condition with hard constraint and to avoid the boundary/initial loss [35] for the LL model $q_t(\boldsymbol{x};\phi) = \mathrm{NN}(\boldsymbol{x},t;\phi)t - \log p_0(\boldsymbol{x})$. and

$S_t^{(2)}\left(\boldsymbol{x};\theta^{(2)}\right)=\mathrm{NN}(\boldsymbol{x},t;\theta^{(2)})t-\nabla\log p_0(\boldsymbol{x})$.

In the smooth L_1 loss, we take β=1. In Section 4.1, we also test the influence of different choices of β. The score and LL models are trained via Adam [29] for 100K and 10K epochs, respectively, with an initial learning rate of 1e-3, which exponentially decays with a decay rate of 0.9 for every 10K epochs. We select 10K random residual points along the SDE trajectory with uniform time steps t at each Adam epoch for all methods in training score and LL. We randomly sample 10K fixed testing points along the SDE trajectory. For the test points, we delete test points of rare points far away from $\mathbf{0}$, see Fig. 1 for examples, which is mathematically defined as those with the lowest 10% LL.

All experiments are conducted on an NVIDIA A100 GPU with 80GB of memory. Due to its efficient automatic differentiation, we implement our algorithm using JAX [7]. All experiments are computed five times with five independent random seeds for reproducibility.

4.1 High-dimensional anisotropic basic SDE

4.1.1 SDE formulation

We consider a basic SDE with an anisotropic multivariate Gaussian as the initial condition:

$$d\boldsymbol{x}=d\boldsymbol{w}_t+d\boldsymbol{L}_t^{\alpha},\quad p_0(\boldsymbol{x})\sim\mathcal{N}(0,\boldsymbol{\Sigma}),\tag{4.1}$$

where $\boldsymbol{\Sigma}=\mathrm{diag}(\lambda_1,\lambda_2,\cdots,\lambda_d)$ where $\lambda_{2i}\sim\mathrm{Unif}[1,2]$ and $\lambda_{2i+1}=1/\lambda_{2i}$ are randomly generated for an anisotropic initial condition and SDE. This SDE gradually injects Gaussian and Lévy noises into the initial condition, and its exact solution is the sum of a Gaussian $\mathcal{N}(0,\boldsymbol{\Sigma}+t\boldsymbol{I})$ and the Lévy noise $\boldsymbol{L}_t^{\alpha}$. Despite its simple form, this SDE is still anisotropic due to the initial condition and effectively high-dimensional due to the inseparable Lévy noise, i.e., this SDE cannot be simplified to low-dimensional cases. Thus, traditional methods cannot solve it.

4.1.2 Hyperparameter setting

Here we set the terminal time $T=1$. The networks are $q_t(\boldsymbol{x};\phi)=\mathrm{NN}(\boldsymbol{x},t;\phi)t-\frac{d}{2}\log(2\pi)-\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$, and $S_t^{(2)}\left(\boldsymbol{x};\theta^{(2)}\right)=\mathrm{NN}(\boldsymbol{x},t;\theta^{(2)})t-\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$, where both networks are fully-connected neural networks as described at the beginning of the section. The exact distribution is the one for the sum of a Gaussian $\mathcal{N}(0,\boldsymbol{\Sigma}+t\boldsymbol{I})$ and the Lévy noise $\boldsymbol{L}_t^{\alpha}$. The distribution is computed via Monte Carlo simulation with 10^7 samples. The evaluation criterion is the relative L_2 error of the LL.

4.1.3 Main results

The main results are shown in Table 3. FSM and score-fPINN can achieve low errors ($\sim 1\%$) in high dimensions up to 100D. They perform similarly in general. The two methods' total running time grows sublinearly across different dimensions thanks to the grid-less and mesh-free PINN and score-matching training. This sublinear cost growth

Table 3: Main results for the basic SDE (4.1) up to 100D with various α. We present the relative L_2 errors of FSM and Score-fPINN, which perform similarly. We also present the total running time for both algorithms to demonstrate their scalability.

Method	Dim	alpha	Time	L_2	Method	Dim	alpha	Time	L_2
FSM	100	1.75	12min	2.81E-2	Score-fPINN	100	1.75	59min	2.83E-2
		1.85		2.48E-2			1.85		2.70E-2
		1.95		2.48E-2			1.95		2.47E-2
	50	1.75	11min	2.16E-2		50	1.75	56min	2.17E-2
		1.85		1.94E-2			1.85		1.98E-2
		1.95		1.52E-2			1.95		1.44E-2
	20	1.75	7min	1.23E-2		20	1.75	50min	1.28E-2
		1.85		1.02E-2			1.85		1.09E-2
		1.95		1.02E-2			1.95		1.08E-2
	10	1.75	4min	6.59E-3		10	1.75	42min	6.72E-3
		1.85		6.21E-3			1.85		6.19E-3
		1.95		4.43E-3			1.95		4.58E-3

mitigates the curse of dimensionality [4,18] where traditional mesh-based methods' costs grow exponentially with the PDE dimension. But score-fPINN is more expensive than FSM, as discussed in Section 3.5. As dimension increases, the PDF/LL value becomes smaller, decaying exponentially, and tends to be more unstable, and thus, the prediction error becomes larger, As α decreases, the α-stable distribution's support becomes wider, and there are more rare events and extreme values, posing an additional challenge to PINN. These are all reflected in the results. Nevertheless, in all test cases, our methods achieve a less than 5% test relative error in the highest 100D PDE. Overall, the main results demonstrate the stable performance and scalability of our proposed fractional score-based SDE/FPL equation solver.

4.1.4 Additional study: Effect of smooth L_1 loss

This additional study investigates the effect of β in the smooth L_1 loss. We kept all the same hyperparameters as in the main results, except we manipulated β in the smooth L_1 loss in $\{0.01,0.1,1,10,100\}$. We only study the most difficult 100D case with $\alpha=1.75$ and test the FSM method since the two methods perform similarly.

The results are shown in Table 4. With increasing β in the smooth L_1 loss in $\{1,10,100\}$, we notice a drop in the final relative L_2 error performance on LL. This illustrates the importance of the smooth L_1 loss on the robustness of optimization in the presence of extreme values and rare events produced by stable distributions. With larger β, the smooth L_1 loss will gradually converge to the L_2 loss and become more sensitive to extreme values, thus deteriorating the performance. On the other hand, if we choose a small β, e.g., $\beta=0.01$ and $\beta=0.1$, its optimization effect will deteriorate too due to the

Table 4: Results for the effect of β in the smooth L_1 loss. The best performance is achieved with $\beta=1$, which is bolded.

Method	Dimension	alpha	Beta	L_2
FSM	100	1.75	0.01	3.08E-2
	100	1.75	0.1	2.88E-2
	100	1.75	1	**2.81E-2**
	100	1.75	10	3.30E-2
	100	1.75	100	7.20E-2

following disadvantages of L_1 loss. The gradient of the L_1 loss being constant (except at zero, where it is undefined) can indeed impact the convergence speed of the optimization process. Since the gradient does not scale with the error, significant errors are penalized no more than small errors in gradients. This can lead to inefficient convergence in scenarios where the errors vary significantly in magnitude. When the prediction exactly matches the target (i.e., error = 0), the derivative of the L_1 loss is undefined, which can cause issues in gradient-based optimization methods. The non-differentiability at zero can make the optimization landscape challenging, especially for optimization algorithms that rely heavily on the smoothness of the function, such as some variants of gradient descent.

In conclusion, this additional demonstrates the effectiveness and importance of employing smooth L_1 loss and underscores the importance of choosing an appropriate β. The selection guidelines and underlying mechanisms are also explained.

4.2 High-dimensional anisotropic SDE with complicated diffusion coefficient

4.2.1 SDE formulation

We consider the anisotropic SDE with Brownian and Lévy noises and unit Gaussian as the initial condition:

$$d\boldsymbol{x}=(\boldsymbol{B}+t\boldsymbol{I})d\boldsymbol{w}_t+d\boldsymbol{L}_t^{\alpha},\quad p_0(\boldsymbol{x})\sim\mathcal{N}(0,\boldsymbol{I}). \tag{4.2}$$

The exact solution is the sum of a Gaussian $p_t(\boldsymbol{x})\sim\mathcal{N}(0,\boldsymbol{\Sigma}_t)$ and the Lévy noise $\boldsymbol{L}_t^{\alpha}$, where

$$\boldsymbol{\Sigma}_t=\boldsymbol{I}+\int_0^t\left[(\boldsymbol{B}+s\boldsymbol{I})(\boldsymbol{B}+s\boldsymbol{I})^{\mathrm{T}}\right]ds=\left(1+\frac{t^3}{3}\right)\boldsymbol{I}+t\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}+\frac{t^2}{2}(\boldsymbol{B}+\boldsymbol{B}^{\mathrm{T}}). \tag{4.3}$$

Here, we generate $\boldsymbol{B}=\boldsymbol{Q}\boldsymbol{\Gamma}$ where $\boldsymbol{Q}$ is a random orthogonal matrix and $\boldsymbol{\Gamma}$ is a diagonal matrix generated in the same way used in the previous example of basic SDE. The anisotropic Gaussian $\mathcal{N}(0,\boldsymbol{\Sigma}_t)$ covariance matrix's eigenspace evolves if $\boldsymbol{B}$ is not orthogonal and not symmetric, posing an additional challenge. Compared to the previous case study, we attempt to test more difficult and effective high-dimensional SDE with complicated diffusion coefficient $\boldsymbol{G}(\boldsymbol{x},t)=\boldsymbol{B}+t\boldsymbol{I}$. This SDE is anisotropic due to the diffusion coefficient and effectively high-dimensional due to the inseparable Lévy noise, i.e., this SDE cannot be simplified to low-dimensional cases.

4.2.2 Hyperparameter setting

Here we set the terminal time $T=1$ and present the relative L_2 errors of the LL at $T=1$. The reference, i.e., the LL of the distribution that is the sum of a Gaussian $\mathcal{N}(0,\boldsymbol{\Sigma}_t)$ and the Lévy noise $\boldsymbol{L}_t^\alpha$, is computed via Monte Carlo simulation with 10^7 samples. Unlike in Section 4.1, sampling along the SDE trajectory requires the heavy computation of $\boldsymbol{\Sigma}_t^{1/2}$. Hence, we do not resample t at each training iteration and precompute $\boldsymbol{\Sigma}_t^{1/2}$. Specifically, after fixing t, we compute $\boldsymbol{\Sigma}_t^{1/2}$ for each t as follows. We conduct eigendecomposition for $\boldsymbol{\Sigma}_t=\boldsymbol{Q}_t^{\mathrm{T}}\boldsymbol{\Gamma}_t\boldsymbol{Q}_t$ where $\boldsymbol{Q}_t$ is orthogonal and $\boldsymbol{\Gamma}_t$ is diagonal. Then, $\boldsymbol{\Sigma}_t^{1/2}$ can be computed via $\boldsymbol{\Sigma}_t^{1/2}=\boldsymbol{Q}_t^{\mathrm{T}}\boldsymbol{\Gamma}_t^{1/2}\boldsymbol{Q}_t$.

4.2.3 Main results

The main results are shown in Table 5. FSM and score-fPINN can achieve low errors ($\sim$2%) in high dimensions up to 100D. The errors are larger than the previous basic SDE due to the complicated diffusion coefficient in the current case, causing the Gaussian component in the exact solution to have varying eigenspace with time evolution. The two methods' total running time grows sublinearly across different dimensions as before but they are slightly more expensive than in the previous basic SDE (4.1) since the computation of the diffusion coefficient takes extra time. We observe that score-fPINN is more expensive than FSM, as discussed in Section 3.5. Like the previous basic SDE results, the relative L_2 error of both methods are similar and grow slightly with the increasing SDE dimension and decreasing α. Overall, the main results demonstrate the stable performance and scalability of our proposed fractional score-based SDE/FPL equation solver on SDE with a complicated diffusion coefficient.

Table 5: Main results for the SDE with a complicated diffusion coefficient (4.2) up to 100D with various α. We present the relative L_2 errors of FSM and Score-fPINN, which perform similarly. We also present the total running time for both algorithms to demonstrate scalability.

Method	Dim	alpha	Time	L_2	Method	Dim	alpha	Time	L_2
FSM	100	1.75	25min	4.98E-2	Score-fPINN	100	1.75	72min	5.00E-2
		1.85		2.95E-2			1.85		3.02E-2
		1.95		2.74E-2			1.95		2.77E-2
	50	1.75	19min	3.79E-2		50	1.75	68min	3.78E-2
		1.85		2.56E-2			1.85		2.58E-2
		1.95		1.79E-2			1.95		1.75E-2
	20	1.75	14min	1.92E-2		20	1.75	62min	1.98E-2
		1.85		1.28E-2			1.85		1.30E-2
		1.95		1.14E-2			1.95		1.23E-2
	10	1.75	12min	1.32E-2		10	1.75	57min	1.11E-2
		1.85		9.41E-3			1.85		8.99E-3
		1.95		5.21E-3			1.95		5.62E-3

4.2.4 Additional study: Effect of Laplacian initial distribution

This additional study investigates the effect of changing the initial distribution to be a unit Laplacian distribution, which causes the exact solution to be the sum of the Gaussian $\mathcal{N}(0,\boldsymbol{\Sigma}_t-\boldsymbol{I})$, the unit Laplacian, and the Lévy noise $\boldsymbol{L}_t^\alpha$. The mixed exact solution poses more challenges to the SDE solver. We kept the hyperparameters the same as in the main results, with the only difference being that we will use boundary loss with 20 weight for the LL model instead of the hard constraint due to the non-smoothness of the Laplacian PDF and LL. Also, we extend the training of the LL model from the previous 10K epochs to 100K epochs. This is because we now require the boundary loss for the non-smooth Laplacian initial condition instead of the hard constraint, which requires more epochs for the LL model to optimize the additional loss function. We test both FSM and score-fPINN in 100D.

The additional results are shown in Table 6. Overall, these additional results demonstrate the stable performance and scalability of our proposed score-fPINN on an SDE with a complicated diffusion coefficient plus the Laplacian initial condition, whose exact solution is a mixture of anisotropic Gaussian, Laplacian, and stable distributions.

Table 6: Additional results for the SDE with a complicated diffusion coefficient (4.2) up to 100D with various α, whose initial distribution is changed to be Laplacian for more SDE problem complexity. We present the relative L_2 errors of FSM and Score-fPINN, which perform similarly. We also present the total running time for both algorithms to demonstrate scalability.

Method	Dim	alpha	Time	L_2	Method	Dim	alpha	Time	L_2
FSM	100	1.75	95min	6.57E-2	Score-fPINN	100	1.75	297min	6.68E-2
		1.85		6.50E-2			1.85		6.42E-2
		1.95		6.28E-2			1.95		6.19E-2

4.3 High-dimensional anisotropic OU processes with linear drift

4.3.1 SDE formulation

We consider the OU process and an anisotropic multivariate Gaussian as the initial condition:

$$d\boldsymbol{x}=-\frac{\boldsymbol{x}}{\alpha}dt+d\boldsymbol{L}_t^\alpha,\quad p_0(\boldsymbol{x})\sim\mathcal{N}(0,\boldsymbol{\Sigma}),\tag{4.4}$$

where $\boldsymbol{\Sigma}=\mathrm{diag}(\lambda_1,\lambda_2,\cdots,\lambda_d)$ where $\lambda_{2i}\sim\mathrm{Unif}[1,2]$ and $\lambda_{2i+1}=1/\lambda_{2i}$. This OU process is interesting since it gradually transforms any initial distribution to the unit α-stable distribution as $t\to\infty$. The score-based generative model [49] utilizes this OU process to transform the data distribution into pure Lévy noise and then reverses the SDE. The reverse SDE serves as a pathway from noise to images, enabling image generation, with its drift coefficient associated with the fractional score function. Mathematically, the exact

solution to this OU process is given by

$$\boldsymbol{x}_t = \exp\left(-\frac{t}{\alpha}\right)\boldsymbol{x}_0 + \mathcal{S}\alpha\mathcal{S}^d\left((1-\exp(-t))^{1/\alpha}\right), \quad \text{in distribution.} \tag{4.5}$$

The exact solution is the weighted sum of the initial condition and an α-stable process, which can be computed via Monte Carlo simulation. This SDE is still anisotropic due to the initial condition and effectively high-dimensional due to the inseparable Lévy noise, i.e., this SDE cannot be simplified to low-dimensional cases.

4.3.2 Hyperparameter setting

Here the terminal time $T=0.5$. We adopt the following models: $q_t(\boldsymbol{x};\phi)=\mathrm{NN}(\boldsymbol{x},t;\phi)t-\frac{d}{2}\log(2\pi)-\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$, where $\mathrm{NN}(\boldsymbol{x},t;\phi)$ is the fully connected neural network and $\boldsymbol{S}_t^{(2)}\left(\boldsymbol{x};\theta^{(2)}\right)=\mathrm{NN}(\boldsymbol{x},t;\theta^{(2)})t-\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$, where $\mathrm{NN}(\boldsymbol{x},t;\theta^{(2)})$ is the fully connected neural network. Both networks are described at the beginning of the section.

The reference, i.e., the LL of the distribution, the sum of a Gaussian and the Lévy noise, is computed via Monte Carlo simulation with 10^7 samples. The evaluation criterion is the relative L_2 error of the LL.

4.3.3 Main results

The main results are shown in Table 7. FSM and score-fPINN can achieve low errors ($\sim$1%) in high dimensions up to 100D. The two methods' total running time grows sublinearly across different dimensions. As dimension increases, the PDF/LL value becomes smaller, decaying exponentially, and tends to be more unstable, and thus, the prediction error

Table 7: Main results for the OU process up to 100D with various α. We present the relative L_2 errors of FSM and Score-fPINN, which perform similarly. We also present the total running time for both algorithms to demonstrate scalability.

Method	Dim	alpha	Time	L_2	Method	Dim	alpha	Time	L_2
FSM	100	1.75	21min	3.98E-2	Score-fPINN	100	1.75	68min	3.68E-2
		1.85		3.90E-2			1.85		3.57E-2
		1.95		3.87E-2			1.95		3.44E-2
	50	1.75	17min	1.23E-2		50	1.75	56min	1.34E-2
		1.85		1.15E-2			1.85		1.20E-2
		1.95		1.15E-2			1.95		1.18E-2
	20	1.75	15min	4.67E-3		20	1.75	41min	5.33E-3
		1.85		3.97E-3			1.85		4.08E-3
		1.95		2.54E-3			1.95		3.28E-3
	10	1.75	13min	3.15E-3		10	1.75	32imn	4.88E-3
		1.85		2.84E-3			1.85		4.34E-3
		1.95		2.16E-3			1.95		3.04E-3

becomes larger. As α decreases, the α-stable distribution's support becomes wider, and there are more rare events and extreme values, posing an additional challenge to PINN. These results demonstrate our methods' capability to deal with anisotropic OU processes with linear drifts.

4.4 High-dimensional SDE with complicated drift coefficient

4.4.1 SDE formulation

We have already tested several anisotropic SDEs with simple drift coefficients. Since obtaining a reference solution for general SDE with Lévy noise and complicated drift is difficult, we design a high-dimensional FPL equation with a low effective dimension, which can be solved via traditional methods for reference. An intuitive example is the simple Lévy process $d\boldsymbol{x}=d\boldsymbol{L}_t^\alpha$ with the isotropic initial condition $p_0(\boldsymbol{x})\sim\mathcal{N}(0,\boldsymbol{I})$. Since the initial condition and the injected noise are all isotropic, the solution is only a function of $r=\|\boldsymbol{x}\|$. In the polar coordinate, it becomes a 1D problem. Garofalo [15] gives the simplified fractional Laplacian in polar coordinate if the function is only about r. Thus, the intrinsically low-dimensional problem can be solved via traditional methods for score-fPINN's reference. Notably, the high-dimensional α-stable distribution/Lévy noise is inseparable, though it is isotropic across different dimensions. More generally, the following high-dimensional FPL equation can be reduced to 1D problems: $d\boldsymbol{x}=f(\|\boldsymbol{x}\|)\boldsymbol{x}dt+d\boldsymbol{L}_t^\alpha$ where f can be any smooth function. Here we consider a nonlinear hyperbolic tangent drift:

$$d\boldsymbol{x}=-\boldsymbol{x}\tanh\left(\|\boldsymbol{x}\|/\sqrt{d}\right)dt+d\boldsymbol{L}_t^\alpha,\quad p_0(\boldsymbol{x})\sim\mathcal{N}(0,\boldsymbol{I}). \tag{4.6}$$

4.4.2 Reference generation

Since the exact solution is isotropic and only concerns r in the polar coordinate, we can generate reference accurately using kernel density estimation (KDE) thanks to its low-dimensional substructure. We use the Euler-Maruyama scheme to discretize the SDE to get SDE samples/trajectories. The step size for the Euler-Maruyama scheme to discretize the SDE is 0.003. After that, we compute the norm of these samples and use KDE to estimate the distribution of the SDE samples' norms. Then, KDE provides the distribution of $r=\|\boldsymbol{x}\|$ of the SDE solution. Since the exact solution is isotropic and only concerns r, we multiply the KDE results by the normalizing constant $(d-1)\log(r)-(d/2-1)\log(2)-\log\Gamma(d/2)+d/2\log(2\pi)$ to obtain the final LL reference.

4.4.3 Hyperparameter setting

We set $\alpha\in\{1.95,1.85,1.75\}$ and $d=100$ and present the relative L_2 errors of the LL at the terminal time $T=0.3$. In this example, we select 1K random residual points along the SDE trajectory at each Adam epoch in the training score and LL and 10K fixed testing points at the terminal time only. We adopt the following models: $q_t(\boldsymbol{x};\phi)=$

$\mathrm{NN}(\boldsymbol{x},t;\phi)t-\frac{d}{2}\log(2\pi)-\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$, where $\mathrm{NN}(\boldsymbol{x},t;\phi)$ is the fully connected neural network and $\boldsymbol{S}_t^{(2)}\left(\boldsymbol{x};\theta^{(2)}\right)=\mathrm{NN}(\boldsymbol{x},t;\theta^{(2)})t-\boldsymbol{x}$, where $\mathrm{NN}(\boldsymbol{x},t;\theta^{(2)})$ is the fully connected neural network. Both networks are described at the beginning of the section.

4.4.4 Main results

Table 8 shows the main results for SDE (4.6) with a complicated drift coefficient. Note that the conditional distribution of the PDF modeled by this SDE is unknown and thus FSM is not applicable. We only present results for score-fPINN. Similar to previous experiments, score-fPINN achieves stable performances within a reasonable running time. Its relative error remained stable for tested α's in 100D. This illustrates that score-fPINN is applicable across various SDE scenarios, including basic SDE and SDEs with complicated coefficients.

Table 8: Results for SDE (4.6) with nonlinear drift coefficient in 100D with various α.

Method	Dimension	alpha	Time	L_2
Score-fPINN	100	1.75	50min	8.51E-3
		1.85		4.97E-3
		1.95		2.39E-3

5 Summary

We considered computational methods for high-dimensional stochastic differential equations (SDEs) with Lévy noise and their corresponding Fokker-Planck-Lévy (FPL) equations. We proposed a novel approach that accurately infers a log-likelihood (LL) solution by employing a fractional score-based SDE solver to fit the fractional score function. We demonstrated the numerical stability and the critical role of the fractional score function in modeling the SDE distribution. We introduced two fitting methods, Fractional Score Matching (FSM) and Score-fPINN, and thoroughly compare them in computational complexity and generality. Specifically, FSM is more concise and faster to train but only applies to SDEs with known conditional distributions. Score-fPINN, due to its need for computing model derivatives, is slower but more versatile. Efficient LL inference can be achieved after fitting the fractional score function. Our experiments confirm the stability and performance of the proposed SDE solver across various SDEs, demonstrating true lift of CoD. Importantly, our method outperforms traditional approaches and maintains stable computational costs as dimensions increase. The proposed method addresses challenges in high-dimensional stochastic systems, paving the way for further exploration and application in various scientific and engineering fields. Future work will include accelerating training by speeding up the sampling of stable distributions and Lévy noises, which is a current bottleneck in training such models compared to the fast sampling of conventional Gaussian distributions.

References

[1] Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[2] Christian Beck, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. Deep splitting method for parabolic PDEs. *SIAM Journal on Scientific Computing (SISC)*, 43:A3135–A3154, 2019.

[3] Christian Beck, Lukas Gonon, and Arnulf Jentzen. Overcoming the curse of dimensionality in the numerical approximation of high-dimensional semilinear elliptic partial differential equations. *arXiv preprint arXiv:2003.00596*, 2020.

[4] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[5] Nicholas M Boffi and Eric Vanden-Eijnden. Deep learning probability flows and entropy production rates in active matter. *arXiv preprint arXiv:2309.12991*, 2023.

[6] Nicholas M Boffi and Eric Vanden-Eijnden. Probability flow solution of the Fokker–Planck equation. *Machine Learning: Science and Technology*, 4(3):035012, 2023.

[7] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018.

[8] Peng Chen, Chang-Song Deng, René L Schilling, and Lihu Xu. Approximation of the invariant measure of stable sdes by an Euler–Maruyama scheme. *Stochastic Processes and their Applications*, 163:136–167, 2023.

[9] Xiaoli Chen, Liu Yang, Jinqiao Duan, and George Em Karniadakis. Solving inverse stochastic problems from discrete particle observations using the Fokker–Planck equation and physics-informed neural networks. *SIAM Journal on Scientific Computing (SISC)*, 43(3):B811–B830, 2021.

[10] Weihua Deng. Finite element method for the space and time fractional Fokker–Planck equation. *SIAM Journal on Numerical Analysis*, 47(1):204–226, 2009.

[11] Xiaodong Feng, Li Zeng, and Tao Zhou. Solving time dependent Fokker-Planck equations via temporal normalizing flow. *arXiv preprint arXiv:2112.14012*, 2021.

[12] Xiaodong Feng, Li Zeng, and Tao Zhou. Solving time dependent Fokker-Planck equations via temporal normalizing flow. *Communications in Computational Physics*, 32(2):401–423, 2022.

[13] Ali Nosrati Firoozsalari, Hassan Dana Mazraeh, Alireza Afzal Aghaei, and Kourosh Parand. deepFDEnet: A novel neural network architecture for solving fractional differential equations. *arXiv preprint arXiv:2309.07684*, 2023.

[14] Ting Gao, Jinqiao Duan, and Xiaofan Li. Fokker–Planck equations for stochastic dynamical systems with symmetric lévy motions. *Applied Mathematics and Computation*, 278:1–20, 2016.

[15] Nicola Garofalo. Fractional thoughts. *arXiv preprint arXiv:1712.03347*, 2017.

[16] Ling Guo, Hao Wu, Xiaochen Yu, and Tao Zhou. Monte Carlo fPINNs: Deep learning method for forward and inverse problems involving high dimensional fractional partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 400:115523, 2022.

[17] Ling Guo, Hao Wu, and Tao Zhou. Normalizing field flows: Solving forward and inverse stochastic differential equations using physics-informed flow models. *Journal of Computational Physics*, 461:111202, 2022.

[18] PC Hammer. Adaptive Control Processes: A Guided Tour (R. Bellman), 1962.

[19] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[20] Di He, Shanda Li, Wenlei Shi, Xiaotian Gao, Jia Zhang, Jiang Bian, Liwei Wang, and Tie-Yan Liu. Learning physics-informed neural networks without stacked back-propagation. In *International Conference on Artificial Intelligence and Statistics*, pages 3034–3047. PMLR, 2023.

[21] Zheyuan Hu, Ameya D Jagtap, George Em Karniadakis, and Kenji Kawaguchi. When do extended physics-informed neural networks (XPINNs) improve generalization? *SIAM Journal on Scientific Computing (SISC)*, 44(5):A3158–A3182, 2022.

[22] Zheyuan Hu, Zekun Shi, George Em Karniadakis, and Kenji Kawaguchi. Hutchinson trace estimation for high-dimensional and high-order physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 424:116883, 2024.

[23] Zheyuan Hu, Khemraj Shukla, George Em Karniadakis, and Kenji Kawaguchi. Tackling the curse of dimensionality with physics-informed neural networks. *Neural Networks*, 176:106369, 2024.

[24] Zheyuan Hu, Zhouhao Yang, Yezhen Wang, George Em Karniadakis, and Kenji Kawaguchi. Bias-variance trade-off in physics-informed neural networks with randomized smoothing for high-dimensional PDEs. *arXiv preprint arXiv:2311.15283*, 2023.

[25] Zheyuan Hu, Zhongqiang Zhang, George Em Karniadakis, and Kenji Kawaguchi. Score-based physics-informed neural networks for high-dimensional Fokker-Planck equations. *arXiv preprint arXiv:2402.07465*, 2024.

[26] Martin Hutzenthaler, Arnulf Jentzen, Thomas Kruse, et al. Multilevel Picard iterations for solving smooth semilinear parabolic heat equations. *Partial Differential Equations and Applications*, 2(6):1–31, 2021.

[27] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[28] Kenji Kawaguchi. On the theory of implicit deep learning: Global convergence with implicit layers. In *International Conference on Learning Representations (ICLR)*, 2021.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Samuel Kotz and Saralees Nadarajah. *Multivariate t-Distributions and Their Applications*. Cambridge University Press, 2004.

[31] Chieh-Hsin Lai, Yuhta Takida, Naoki Murata, Toshimitsu Uesaka, Yuki Mitsufuji, and Stefano Ermon. Regularizing score-based models with score Fokker-Planck equations. *arXiv preprint arXiv:2210.04296*, 2022.

[32] Hwi-Young Lee, Hyoung-Jin Park, and Hyoung-Moon Kim. A clarification of the Cauchy distribution. *Communications for Statistical Applications and Methods*, 21(2):183–191, 2014.

[33] Nikolai Leonenko and Igor Podlubny. Monte Carlo method for fractional-order differentiation extended to higher orders. *Fractional Calculus and Applied Analysis*, 25(3):841–857, 2022.

[34] Anna Lischke, Guofei Pang, Mamikon Gulian, Fangying Song, Christian Glusa, Xiaoning Zheng, Zhiping Mao, Wei Cai, Mark M. Meerschaert, Mark Ainsworth, and George Em Karniadakis. What is the fractional Laplacian? A comparative review with new results. *Journal of Computational Physics*, 404:109009, 2020.

[35] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021.

[36] Yubin Lu, Romit Maulik, Ting Gao, Felix Dietrich, Ioannis G Kevrekidis, and Jinqiao Duan. Learning the temporal evolution of multivariate densities via normalizing flows. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(3), 2022.

[37] Guofei Pang, Lu Lu, and George Em Karniadakis. fPINNs: Fractional physics-informed

neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

[38] Tianyu Pang, Kun Xu, Chongxuan Li, Yang Song, Stefano Ermon, and Jun Zhu. Efficient learning of generative models via finite-difference score matching. *Advances in Neural Information Processing Systems*, 33:19175–19188, 2020.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[40] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[41] Behnam Sepehrian and Marzieh Karimi Radpoor. Numerical solution of non-linear Fokker–Planck equation using finite differences method and the cubic spline functions. *Applied Mathematics and Computation*, 262:187–190, 2015.

[42] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[43] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, page 204, 2019.

[44] 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.

[45] Pablo Raúl Stinga. *User's Guide to the Fractional Laplacian and the Method of Semigroups*, pages 235–266. De Gruyter, Berlin, Boston, 2019.

[46] Kejun Tang, Xiaoliang Wan, and Qifeng Liao. Adaptive deep density approximation for Fokker-Planck equations. *Journal of Computational Physics*, 457:111080, 2022.

[47] Michael V Tretyakov and Zhongqiang Zhang. A fundamental mean-square convergence theorem for SDEs with locally Lipschitz coefficients and its applications. *SIAM Journal on Numerical Analysis*, 51(6):3135–3162, 2013.

[48] Taorui Wang, Zheyuan Hu, Kenji Kawaguchi, Zhongqiang Zhang, and George Em Karniadakis. Tensor neural networks for high-dimensional Fokker-Planck equations. *arXiv preprint arXiv:2404.05615*, 2024.

[49] Eunbi Yoon, Keehun Park, Sungwoong Kim, and Sungbin Lim. Score-based generative models with Lévy processes. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[50] Li Zeng, Xiaoliang Wan, and Tao Zhou. Adaptive deep density approximation for fractional Fokker–Planck equations. *Journal of Scientific Computing*, 97:1–31, 2022.

[51] Jiayu Zhai, Matthew Dobson, and Yao Li. A deep learning method for solving Fokker-Planck equations. In *Mathematical and Scientific Machine Learning*, pages 568–597. PMLR, 2022.

[52] Hao Zhang, Yong Xu, Qi Liu, Xiaolong Wang, and Yongge Li. Solving Fokker–Planck equations using deep KD-tree with a small amount of data. *Nonlinear Dynamics*, 108(4):4029–4043, 2022.

[53] Yequan Zhao, Xinling Yu, Zhixiong Chen, Ziyue Liu, Sijia Liu, and Zheng Zhang. Tensor-compressed back-propagation-free training for (physics-informed) neural networks. *arXiv preprint arXiv:2308.09858*, 2023.