Neural network representation of the probability density function of diffusion processes

Wayne Isaac T. Uy[1,3], Mircea D. Grigoriu[1,2*]

[1]Center for Applied Mathematics
[2]Department of Civil and Environmental Engineering
Cornell University, Ithaca, NY 14850, USA

[3]Courant Institute of Mathematical Sciences
New York University, NY 10012, USA

wayne.uy@cims.nyu.edu, mdg12@cornell.edu

Abstract

Physics-informed neural networks are developed to characterize the state of dynamical systems in a random environment. The neural network approximates the probability density function (pdf) or the characteristic function (chf) of the state of these systems which satisfy the Fokker-Planck equation or an integro-differential equation under Gaussian and/or Poisson white noises. We examine analytically and numerically the advantages and disadvantages of solving each type of differential equation to characterize the state. It is also demonstrated how prior information of the dynamical system can be exploited to design and simplify the neural network architecture. Numerical examples show that: 1) the neural network solution can approximate the target solution even for partial integro-differential equations and system of PDEs describing the time evolution of the pdf/chf, 2) solving either the Fokker-Planck equation or the chf differential equation using neural networks yields similar pdfs of the state, and 3) the solution to these differential equations can be used to study the behavior of the state for different types of random forcings.

The probability density function of a state in a stochastic system is of paramount importance in science and engineering. It can be used to compute statistics of the state which are essential in understanding its behavior. An approach to characterize the time evolution of the pdf is by solving the Fokker-Planck equation. However, this equation may be unavailable for some types of random forcings and furthermore, finite difference and finite element numerical schemes may not be suitable for partial differential equations in high dimensions. In view of these, we derive a differential equation for the characteristic function of the state via stochastic analysis and develop a physics-informed neural network representation for the time-evolution of the pdf and the chf. The neural network architecture is designed by incorporating prior information on the stochastic differential equation. Our methodology is applied to various nonlinear dynamical systems driven by Gaussian and/or Poisson white noise. The results highlight that neural networks can adequately approximate the pdf or the

[*] both authors contributed equally to this work

chf of the state which satisfies various types of equations including systems of PDEs and integro-differential equations. The ideas established here can be directly extended to the case when the neural network is trained in a gridless manner.

1 Introduction

Let $\boldsymbol{X}(t) \in \mathbb{R}^d$ be a random vector defined on a probability space $(\Omega, \mathcal{F}, P)$ whose dynamics are goverened by

$$d\boldsymbol{X}(t) = \boldsymbol{a}(\boldsymbol{X}(t))\,dt + \boldsymbol{b}(\boldsymbol{X}(t))\,d\boldsymbol{Y}(t),\ \ t \geq 0$$

where $\boldsymbol{Y}(t) \in \mathbb{R}^k$ is a stochastic process and $\boldsymbol{a} \in \mathbb{R}^{d\times 1}, \boldsymbol{b} \in \mathbb{R}^{d\times k}$. Such stochastic differential equations (SDE) are used to model complex systems that arise in various applications of science and engineering [11]. We are interested in computing the probability density function $f(\boldsymbol{x}, t)$ of $\boldsymbol{X}(t)$ which can then be used to estimate statistics of $\boldsymbol{X}(t)$, including its moments $E[\boldsymbol{X}(t)^p]$ and probabilities of events $P(\boldsymbol{X}(t) \in A)$, $A \in \mathcal{F}$. If $\boldsymbol{Y}(t)$ is Brownian motion, $f(\boldsymbol{x}, t)$ satisfies a partial differential equation (PDE) called the Fokker-Planck equation [19]. Analytical solutions to this PDE are only available under particular conditions on the drift and diffusion matrices [15,19] and moreover, numerical solutions via the finite element method become unstable if the state $\boldsymbol{X}(t)$ has dimension $d > 3$ [16,17,23].

Other methods have been sought to solve the Fokker-Planck equation in high dimensions for special cases. In general, they approximate solutions of PDEs on high-dimensional domains. We only provide a brief survey of recent work. In [7], various algorithms are presented to solve kinetic PDEs in which the high-dimensional problem is transformed into a sequence of low-dimensional ones. Nonlinear high-dimensional PDEs are tackled in [9] by decomposing the function space into lower-dimensional nested subspaces. The work by [5,6] study the Fokker-Planck equation for high-dimensional nonlinear turbulent dynamical systems which possess conditional Gaussian structures, i.e. a set of components of the state conditioned on the trajectory of the remaining components is a Gaussian process. Finally, [4] reduces the high-dimensional Fokker-Planck equation into a 1 or 2-dimensional PDE and uses the path integral solution to solve resulting the dimension-reduced PDE.

As an alternative to the above dimension-reduction approaches and the traditional methods for solving PDEs, neural networks have been proposed to solve nonlinear and/or high-dimensional PDEs in scientific, engineering, and financial applications [18, 20]. A neural network is used to represent the solution to the PDE while its parameters are obtained via optimization. The objective function enforces the neural network approximation to satisfy the governing equation of the PDE together with the initial and boundary conditions. This idea has been pursued in [18] to solve the Schrödinger, Allen-Cahn, KdV, and Burgers' equation in 1 dimension and the Navier Stokes equation in 2 dimensions which are nonlinear PDEs whose solution may exhibit nearly discontinuous behavior. The work [20] successfully estimated free boundary PDE solutions on domains of up to 200 dimensions and solutions to the high-dimensional Hamilton-Jacobi-Bellman PDE.

By utilizing the above methodology, a neural network-based approximation to the Fokker-Planck equation has been undertaken in [2,3,24]. The 1-dimensional time-varying PDE was tackled in [2,3] where it was noticed that it was necessary to incorporate the constraint that the Fokker-Planck solution integrates to 1 for all times in the optimization step. Otherwise, the authors showed that the neural network approximation could not recover the analytical solution. In [24], the steady-state PDE of up to 3 dimensions was addressed and strategies were presented to account for the normalization constraint.

Building on existing work, we investigate how neural networks can be used to represent the pdf of a state vector satisfying a stochastic differential equation. In contrast to [24], we seek the pdf over a time interval instead of the steady-state solution. In addition, while [2,3,24] are only concerned with the Fokker-Planck equation, we also consider an alternative differential equation which describes the time evolution of the characteristic function of the state. The pdf and chf offer identical information about $\boldsymbol{X}(t)$ such that both can be used to compute its statistics, however, the latter is complex-valued. We study the advantages and disadvantages of solving the Fokker-Planck equation or the differential equation for the chf in order to approximate the pdf of $\boldsymbol{X}(t)$ from an analytical and numerical perspective. In particular, we highlight situations in which solving the Fokker-Planck equation may not be favorable regardless of the solution method employed. Strategies are then outlined on how the neural network architecture can be designed and simplified by exploiting probabilistic information from the SDE. The numerical examples feature the capabilities of the neural network solution to match the target solution for various dynamical systems subject to different types of noise. This work serves as a proof of concept of our objectives and adapts the methodology of [18]. Extensions to the high-dimensional situations can be accomplished following [20]. The differential equations dealt with here are different from the examples presented in [18]; for instance, the time evolution of the chf may be represented by a partial integro-differential equation in which the highest order of the partial derivative can exceed 2.

A brief survey of neural networks and its application to solving PDEs in the spirit of [18] is presented in Section 2. In Section 3, we derive a differential equation for the characteristic function of the state subject to commonly used random forcing via stochastic analysis. A comparison between this differential equation and the Fokker-Planck equation is also performed. A neural network-based solution for these differential equations is then presented in Section 4. Finally, Section 5 showcases the approximation properties of neural networks for a variety of applications aligned with our objective.

2 Physics-informed neural networks

A brief survey of the physics-informed neural networks framework [18] for approximating solutions to PDEs is outlined in this section. Section 2.1 describes the components of the neural network architecture and training of its parameters as employed in machine learning. Section 2.2 then elaborates how neural networks can be trained to represent solutions to PDEs.

2.1 Review of neural networks

Neural networks traditionally employed in machine learning construct an approximation $\widetilde{\boldsymbol{f}}(\boldsymbol{x}, t)$ to an unknown mapping $(\boldsymbol{x}, t) \in \mathbb{R}^{d+1} \mapsto \boldsymbol{f}(\boldsymbol{x}, t) \in \mathbb{R}^m$ from data on the input and the output. Several types of neural network architectures exist [10]; in this work, we only focus on feedforward neural networks. Denote the components of $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{f} \in \mathbb{R}^m$ by $\boldsymbol{x} = (x_1, \ldots, x_d)$ and $\boldsymbol{f} = (f_1, \ldots, f_m)$ respectively, and suppose that N data points $\{(\boldsymbol{x}_i, t_i, \boldsymbol{f}(\boldsymbol{x}_i, t_i))\}_{i=1}^N$ of the unknown function $\boldsymbol{f}$ are available. The neural network architecture is comprised of an input layer with $m_0 := d + 1$ neurons corresponding to each input, an output layer with $m_{L+1} := m$ neurons corresponding to each output, and L hidden layers in between with $m_\ell, \ell = 1, \ldots, L$ neurons each. An example of a neural network with 2 hidden layers is depicted in Figure 1.

The output layer ($\ell = L + 1$) and each of the hidden layers is associated with a function $\mathcal{H}_\ell$:

$\mathbb{R}^{m_{\ell-1}} \to \mathbb{R}^{m_\ell}, \ell = 1, \dots, L+1$ such that the approximation $\widetilde{\boldsymbol{f}}$ is a composition of these functions, i.e.

$$\widetilde{\boldsymbol{f}}(\boldsymbol{x}, t) = \mathcal{H}_{L+1} \circ \cdots \circ \mathcal{H}_1(\boldsymbol{x}, t).$$

In particular, $\mathcal{H}_\ell$ is a possibly nonlinear transformation of an affine function expressed as $\mathcal{H}_\ell(\boldsymbol{z}) = \sigma_\ell(\boldsymbol{W}^\ell \boldsymbol{z} + \boldsymbol{b}^\ell)$ in which $\boldsymbol{z} \in \mathbb{R}^{m_{\ell-1}}$, σ_ℓ is an activation function that is applied to each component of its input argument, $\boldsymbol{W}^\ell \in \mathbb{R}^{m_\ell \times m_{\ell-1}}$ is the weight matrix, and $\boldsymbol{b}^\ell \in \mathbb{R}^{m_\ell}$ is a vector of biases. If the (i, j)-entry of $\boldsymbol{W}^\ell$ is 0, this signifies that there is no edge connecting the j-th neuron of the $(\ell - 1)$-th layer to the i-th neuron of the ℓ-th layer.

In the above formulation, the number of hidden layers, the number of neurons m_ℓ per hidden layer, and the activation function σ_ℓ have to be specified beforehand. Commonly used activation functions include the sigmoid $\sigma_\ell(z) = \frac{1}{1+e^{-z}}$, the ReLU $\sigma_\ell(z) = \max(z, 0)$, and the hyperbolic tangent $\sigma_\ell(z) = \tanh z$, the latter being used in our simulations. The weight matrices $\boldsymbol{W}^\ell$ and the bias vectors $\boldsymbol{b}^\ell$ are then estimated by minimizing a loss function $\mathcal{L}$ which measures the discrepancy between the available data and the prediction via $\widetilde{\boldsymbol{f}}(\boldsymbol{x}, t)$. One such loss function is the mean squared error (MSE) given by $\mathcal{L} = \sum_{i=1}^N \|\widetilde{\boldsymbol{f}}(\boldsymbol{x}_i, t_i) - \boldsymbol{f}(\boldsymbol{x}_i, t_i)\|_2^2$. The loss is minimized via gradient descent wherein the gradients of $\mathcal{L}$ with respect to $\boldsymbol{W}^\ell, \boldsymbol{b}^\ell$ are efficiently calculated through backpropagation.

We refer the reader to [10] for further details on choosing the activation and the loss functions, the various optimization algorithms for minimizing the loss, approaches to initializing the parameters, etc.

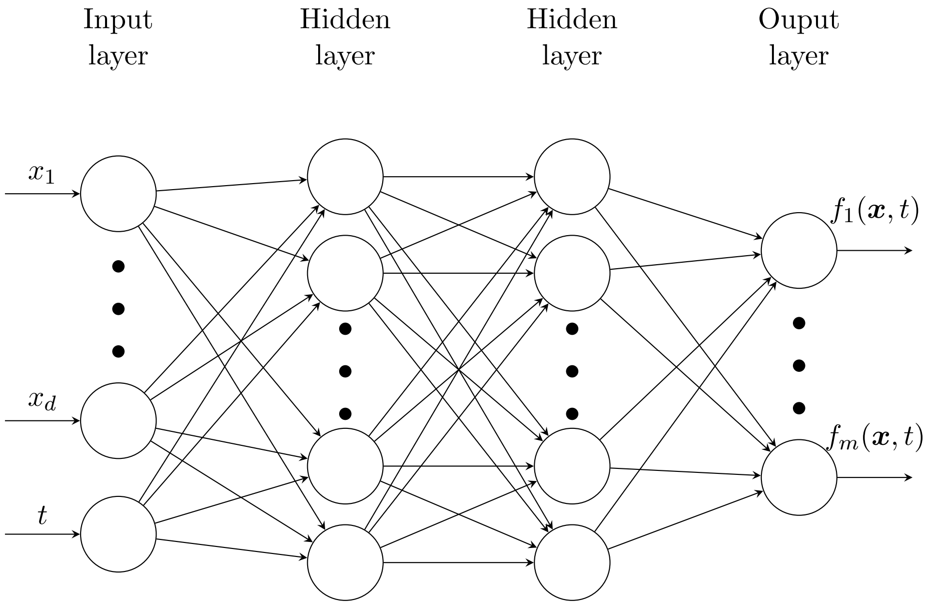

Figure 1: An example of a feedforward neural network with 2 hidden layers.

2.2 Solving PDEs using neural networks

Feedforward neural networks are universal approximators, i.e. they can sufficiently approximate any measurable function [8, 14] such as solutions to PDEs. Denote by $\boldsymbol{f}(\boldsymbol{x}, t)$ the solution to the

PDE given by

$$\begin{aligned} \mathcal{N}[\boldsymbol{f}(\boldsymbol{x},t)] &= \mathbf{0}, && (\boldsymbol{x},t) \in D \times [0,T] \\ \boldsymbol{f}(\boldsymbol{x},0) &= \boldsymbol{g}(\boldsymbol{x}), && \boldsymbol{x} \in D \\ \boldsymbol{f}(\boldsymbol{x},t) &= \boldsymbol{h}(\boldsymbol{x},t), && (\boldsymbol{x},t) \in \partial D \times [0,T] \end{aligned} \tag{2.1}$$

for the operator $\mathcal{N}$, spatial domain D and its boundary ∂D, end time T, and initial and boundary conditions $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x},t)$, with $\mathbf{0}$ denoting a vector of zeros. Unlike the setup traditionally utilized in machine learning wherein we desire the neural network to match available data on the PDE solution $\{(\boldsymbol{x}_i, t_i, \boldsymbol{f}(\boldsymbol{x}_i, t_i))\}_{i=1}^{N}$, [18] pursues an approach based on physics-informed neural networks. Instead, collocation points in the domain $D \times [0,T]$ are selected to enforce the governing equation and the initial and boundary equations in (2.1) which are known. Let $\{(\boldsymbol{x}_i^{Op}, t_i^{Op})\}_{i=1}^{N_{Op}}$, $\{(\boldsymbol{x}_i^{IC}, 0)\}_{i=1}^{N_{IC}}$, and $\{(\boldsymbol{x}_i^{BC}, t_i^{BC})\}_{i=1}^{N_{BC}}$ be 3 sets of collocation points corresponding to each equation in (2.1). For a specified architecture for the neural network approximation $\widetilde{\boldsymbol{f}}(\boldsymbol{x},t)$ of $\boldsymbol{f}(\boldsymbol{x},t)$, we seek the weight matrices and the bias vectors that minimize the loss function

$$\begin{aligned} \mathcal{L} = \frac{1}{N_{Op}} \sum_{i=1}^{N_{Op}} \|\mathcal{N}[\widetilde{\boldsymbol{f}}(\boldsymbol{x}_i^{Op}, t_i^{Op})]\|_2^2 &+ \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} \|\widetilde{\boldsymbol{f}}(\boldsymbol{x}_i^{IC}, 0) - \boldsymbol{g}(\boldsymbol{x}_i^{IC})\|_2^2 \\ &+ \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \|\widetilde{\boldsymbol{f}}(\boldsymbol{x}_i^{BC}, t_i^{BC}) - \boldsymbol{h}(\boldsymbol{x}_i^{BC}, t_i^{BC})\|_2^2. \end{aligned} \tag{2.2}$$

Computing $\mathcal{L}$ requires calculating gradients of $\widetilde{\boldsymbol{f}}$ that are present in the operator $\mathcal{N}$ which is efficiently carried out through automatic differentiation in TensorFlow [1].

If $\boldsymbol{x}$ is high-dimensional, a large number of collocation points in D would be required to ensure that $\widetilde{\boldsymbol{f}}$ satisfies the constraints in (2.1). In this case, [20] proposes a meshfree method in which randomly chosen batches of collocation points in $D \times [0,T]$ are selected to enforce (2.1) in the process of training the neural network.

We emphasize that our objective is to investigate the feasibility of neural networks in representing the probability density function $f(\boldsymbol{x},t)$ of the state $\boldsymbol{X}(t)$ that arises as the solution of a PDE. Consequently, we adopt the physics-informed neural network approach in [18] together with most of the architecture specifications they have used in their simulations. Our focus is not on finding the most effective choice of activation functions, number of hidden layers and hidden neurons, optimization algorithm, etc.

3 Differential equations for the state pdf

We detail how the pdf of the state can be represented as the solution of some differential equation. Stochastic analysis is performed in Section 3.1 to derive a differential equation for the chf of the state subject to commonly used random forcings. The Fokker-Planck equation is then reviewed in Section 3.2 which is a consequence of the differential equation for the chf.

3.1 Differential equation for the characteristic function of the state

We derive the differential equation for the characteristic function of dynamical systems subject to Gaussian and Poisson white noise defined as formal derivatives of the Brownian motion and compound Poisson processes. Examples are then presented to illustrate the application of the derived equation.

Consider the $\mathbb{R}^d$-valued diffusion process defined by the stochastic differential equation

$$d\boldsymbol{X}(t) = \boldsymbol{a}\big(\boldsymbol{X}(t-)\big)\,dt + \boldsymbol{b}\big(\boldsymbol{X}(t-)\big)\,d\boldsymbol{B}(t) + \boldsymbol{c}\big(\boldsymbol{X}(t-)\big)\,d\boldsymbol{C}(t), \quad t \geq 0, \tag{3.1}$$

where the drift $\boldsymbol{a}$ is a $(d,1)$-matrix, the diffusions $\boldsymbol{b}$ and $\boldsymbol{c}$ are (d, m_B) and (d, m_C)-matrices, the Brownian motion $\boldsymbol{B}$ is a vector of m_B independent standard Brownian motions, $\boldsymbol{C}$ is a vector of m_C independent compound Poison processes $C_r(t) = \sum_{\nu=1}^{N_r(t)} Y_{r,\nu}$, $r = 1, \ldots, m_C$, which depend on the homogeneous Poisson processes $\{N_r\}$ of intensities $\{\lambda_r\}$ and jump sizes $\{Y_{r,1}, Y_{r,2}, \ldots\}$ that are independent copies of the random variables $\{Y_r\}$ with $E[Y_r] = 0$, and $\boldsymbol{X}(t-) = \lim_{s\uparrow t} \boldsymbol{X}(s)$. It is assumed that the drift and diffusion coefficients are such that Eq. (3.1) admits a unique strong solution [11, Sect. 4.7.1.1 and 4.7.2].

Theorem 1 *The characteristic function of $\boldsymbol{X}(t)$, $\varphi(\boldsymbol{u}, t) = E\big[\exp\big(i\,\boldsymbol{u}'\,\boldsymbol{X}(t)\big)\big]$ for $\boldsymbol{u} \in \mathbb{R}^d$, satisfies*

$$\begin{aligned}
\frac{\partial\varphi(\boldsymbol{u},t)}{\partial t} &= i \sum_{k=1}^{d} u_k\, E\Big[e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(t-)}\, a_k\big(\boldsymbol{X}(t-)\big)\Big] \\
&\quad - \frac{1}{2}\sum_{k,l=1}^{d} u_k\, u_l\, E\Big[\exp\big(i\,\boldsymbol{u}'\,\boldsymbol{X}(t-)\big) \sum_{w=1}^{m_B} b_{kw}\big(\boldsymbol{X}(t-)\big)\, b_{lw}\big(\boldsymbol{X}(t-)\big)\Big] \\
&\quad + \sum_{r=1}^{m_c} \lambda_r\, E\Big[\int_{\mathbb{R}} e^{i\,\boldsymbol{u}'\,\left(\boldsymbol{X}(t-)+c^{(r)}(\boldsymbol{X}(t-))\,y\right)}\, dF_r(y) - e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(t-)}\Big].
\end{aligned} \tag{3.2}$$

Proof. We use the Itô formula to develop a differential equation for $\varphi(\boldsymbol{u}, t)$. The integral version of this formula for a mapping $\boldsymbol{X}(t) \mapsto g\big(\boldsymbol{X}(t)\big)$ which has continuous second order partial derivatives has the form

$$\begin{aligned}
g\big(\boldsymbol{X}(t)\big) - g\big(\boldsymbol{X}(0)\big) &= \underbrace{\sum_{k=1}^{d}\int_{0+}^{t} \frac{\partial g\big(\boldsymbol{X}(s-)\big)}{\partial x_k}\, dX_k(s)}_{I} + \underbrace{\frac{1}{2}\sum_{k,l=1}^{d}\int_{0+}^{t} \frac{\partial^2 g\big(\boldsymbol{X}(s-)\big)}{\partial x_k\,\partial x_l}\, d\big[X_k, X_l\big]^c(s)}_{II} \\
&\quad + \underbrace{\sum_{0<s\leq t}\Big[g\big(\boldsymbol{X}(s)\big) - g\big(\boldsymbol{X}(s-)\big) - \sum_{k=1}^{d} \frac{\partial g\big(\boldsymbol{X}(s-)\big)}{\partial x_k}\,\Delta X_k(s)\Big]}_{III},
\end{aligned} \tag{3.3}$$

where $[X_k, X_l]^c(s)$ denotes the continuous part of the quadratic covariation of the components X_k and X_l of $\boldsymbol{X}$ and $\Delta X_k(s) = X_k(s) - X_k(s-)$ is the jump of component X_k at time s [11, Sect. 4.6.2]. The Itô formula holds for semimartingales $\boldsymbol{X}(t)$ and shows that $g\big(\boldsymbol{X}(t)\big)$ is also a semimartingale.

The above formula can be applied for the real and imaginary parts of the mapping $\boldsymbol{X}(t) \mapsto \exp\big(i\,\boldsymbol{u}'\,\boldsymbol{X}(t)\big)$ since $\boldsymbol{X}(t)$ in Eq. (3.1) is a semimartingale and $\exp\big(i\,\boldsymbol{u}'\,\boldsymbol{X}(t)\big)$ has continuous partial derivatives. Since Itô's formula is linear in g and its derivative, it can be applied directly to the complex-valued mapping $g : \boldsymbol{X}(t) \mapsto \exp\big(i\,\boldsymbol{u}'\,\boldsymbol{X}(t)\big)$.

An overview of the derivation is as follows. We find the terms I, II, III on the right side of Eq. (3.3) for $g(\boldsymbol{X}(t)) = \exp\big(i\,\boldsymbol{u}'\,\boldsymbol{X}(t)\big)$, calculate their expectations, and find the output of the Itô formula. The differential equation for the characteristic function then results by differentiating the expectation of (3.3) with respect to time.

The first term I of the right side of Eq. (3.3) is

$$\begin{aligned} i\sum_{k=1}^{d} u_k \int_{0+}^{t} e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)}\, dX_k(s) = i\sum_{k=1}^{d} u_k \int_{0+}^{t} e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)} \Big(a_k\big(\boldsymbol{X}(s-)\big)\, ds + \sum_{w=1}^{m_B} b_{kw}\big(\boldsymbol{X}(s-)\big)\, dB_w(s) \\ + \sum_{v=1}^{m_C} c_{kv}\big(\boldsymbol{X}(s-)\big)\, dC_v(s)\Big) \end{aligned}$$

so that its expectation is

$$i\sum_{k=1}^{d} u_k\, E\Big[\int_{0+}^{t} e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)}\, a_k\big(\boldsymbol{X}(s-)\big)\, ds\Big] \tag{3.4}$$

since the stochastic integrals with respect to the Brownian motion and compound Poisson processes are martingales starting at zero so that they have zero expectations.

For the second term II of the right side of Eq. (3.3), we note that the Poisson white noise does not contribute to the processes $[X_k, X_l]^c(s)$ so that

$$\begin{aligned} & d[X_k, X_l]^c(s) \\ &= \Big[a_k\big(\boldsymbol{X}(s-)\big)\, ds + \sum_{w=1}^{m_B} b_{kw}\big(\boldsymbol{X}(s-)\big)\, dB_w(s), a_l\big(\boldsymbol{X}(s-)\big)\, ds + \sum_{v=1}^{m_B} b_{lv}\big(\boldsymbol{X}(s-)\big)\, dB_v(s)\Big]^c \\ &= \Big[\sum_{w=1}^{m_B} b_{kw}\big(\boldsymbol{X}(s-)\big)\, dB_w(s), \sum_{v=1}^{m_B} b_{lv}\big(\boldsymbol{X}(s-)\big)\, dB_v(s)\Big]^c \\ &= \sum_{w,v=1}^{m_B} \Big[b_{kw}\big(\boldsymbol{X}(s-)\big)\, dB_w(s), b_{lv}\big(\boldsymbol{X}(s-)\big)\, dB_v(s)\Big]^c \\ &= \sum_{w,v=1}^{m_B} b_{kw}\big(\boldsymbol{X}(s-)\big)\, b_{lv}\big(\boldsymbol{X}(s-)\big)\, \delta_{wv}\, ds = \sum_{w=1}^{m_B} b_{kw}\big(\boldsymbol{X}(s-)\big)\, b_{lw}\big(\boldsymbol{X}(s-)\big)\, ds \end{aligned}$$

by the definition and linearity of the quadratic covariation process, the postulated independence of the components of $\boldsymbol{B}(t)$, and properties of the Brownian motion, e.g., $[dB_w(t), dB_w(t)] = dt$. The integrand of the second term of (3.3) is $i^2\, u_k\, u_l\, \exp\big(i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)\big)$ so that the expectation of the term is

$$-\frac{1}{2}\sum_{k,l=1}^{d} u_k\, u_l\, E\Big[\int_{0+}^{t} \exp\big(i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)\big) \sum_{w=1}^{m_B} b_{kw}\big(\boldsymbol{X}(s-)\big)\, b_{lw}\big(\boldsymbol{X}(s-)\big)\, ds\Big]. \tag{3.5}$$

The continuous part of $\boldsymbol{X}(t)$ does not contribute to the last term III of the right side of Eq. (3.3). This term has three entries. The last entry has zero mean since the jumps $\Delta X_k(s) = X_k(s) - X_k(s-)$ are scaled versions of the jumps of $\boldsymbol{C}(s)$ which have zero expectations by assumption and so $E\big[\frac{\partial g(\boldsymbol{X}(s-))}{\partial x_k}\Delta X_k(s)\big] = E\big[\frac{\partial g(\boldsymbol{X}(s-))}{\partial x_k}\big]E[\Delta X_k(s)] = 0$. It therefore remains to examine the first two entries, i.e. $h(t) = \sum_{0<s\leq t}[g(\boldsymbol{X}(s)) - g(\boldsymbol{X}(s-))] = \sum_{0<s\leq t}[\exp\big(i\,\boldsymbol{u}'\,\boldsymbol{X}(s)\big) - \exp\big(i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)\big)]$. Instead of computing $E[h(t)]$ explicitly to find $\frac{d}{dt}E[h(t)]$, we consider $E[h(t+\Delta t)] - E[h(t)] =$

$E\Big[\sum_{t<s\le t+\Delta t}\big(e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s)}-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)}\big)\Big]$ and calculate

$$\lim_{\Delta t\to 0}\frac{E\Big[\displaystyle\sum_{t<s\le t+\Delta t}\big(e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s)}-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)}\big)\Big]}{\Delta t} \tag{3.6}$$

to find the contribution of this term to the differential equation for the characteristic function of $\boldsymbol{X}(t)$.

Consider a small time interval interval $(t,t+\Delta t]$, $0<\Delta t\ll 1$. We compute $E\Big[\displaystyle\sum_{t<s\le t+\Delta t}\big(e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s)}-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)}\big)\Big]$ by conditioning on the possible number of jumps of the Poisson processes $\{N_r\}_{r=1}^{m_C}$ in this interval. The probability of the event $\{N_r(\Delta t)=1, N_s(\Delta t)=0, s\neq r\}$ that component r of $\boldsymbol{C}(s)$ has a jump in $(t,t+\Delta t]$ and that the other components do not jump in this time interval is

$$P(N_r(\Delta t)=1,N_s(\Delta t)=0,s\neq r)=\lambda_r\Delta t e^{-\lambda_r\Delta t}\prod_{s\neq r}\exp\big(-\lambda_s\,\Delta t\big)\simeq\lambda_r\,\Delta t \tag{3.7}$$

provided that $\lambda_s\,\Delta t\ll 1$, $s=1,\ldots,m_C$. Note also that the probabilities of two or more jumps of the same or of different components of $\boldsymbol{C}(s)$ in $(t,t+\Delta t]$ are of order $(\Delta t)^2$ so that conditioning on these events will not contribute to the differential equation for the characteristic function of $\boldsymbol{X}(t)$ following (3.6). This means that we only need to consider single component jumps and add their contributions.

Suppose that the component $C_r(s)$ has a jump of size Y_r at T_r in $(t,t+\Delta t]$. Then

$$\begin{aligned}\sum_{t<s\le t+\Delta t}\Big[e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s)}-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)}\Big]&=e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(T_r)}-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(T_r-)}\\&=e^{i\,\boldsymbol{u}'\,\big(\boldsymbol{X}(T_r-)+c^{(r)}(\boldsymbol{X}(T_r-))\,Y_r\big)}-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(T_r-)},\end{aligned}$$

where $c^{(r)}$ is the rth column of the (d,m_C) diffusion matrix $\boldsymbol{c}$. It follows that

$$\begin{aligned}&E\Big[\sum_{t<s\le t+\Delta t}\big(e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s)}-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)}\big)\,\Big|\,N_r(\Delta t)=1,N_s(\Delta t)=0,s\neq r\Big]\\&=E\Big[e^{i\,\boldsymbol{u}'\,\big(\boldsymbol{X}(T_r-)+c^{(r)}(\boldsymbol{X}(T_r-))\,Y_r\big)}-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(T_r-)}\Big]\\&=E\Big[\int_{\mathbb{R}}e^{i\,\boldsymbol{u}'\,\big(\boldsymbol{X}(T_r-)+c^{(r)}(\boldsymbol{X}(T_r-))\,y\big)}\,dF_r(y)-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(T_r-)}\Big]\end{aligned} \tag{3.8}$$

where the last equality holds since the jump Y_r of C_r at time T_r is independent of $\boldsymbol{X}(T_r-)$. In the third line of (3.8), F_r denotes the distribution of Y_r and the expectation refers to $\boldsymbol{X}(T_r-)$. Since $P(T_r\le s\,|\,N_r(\Delta t)=1)=s/\Delta t$, i.e. the jump times of $C_r(t)$ in $(t,t+\Delta t]$ are uniformly distributed, we can replace T_r with a random number in $(t,t+\Delta t]$ so that (3.8) becomes

$$\begin{aligned}&E\Big[\int_t^{t+\Delta t}\frac{ds}{\Delta t}\int_{\mathbb{R}}e^{i\,\boldsymbol{u}'\,\big(\boldsymbol{X}(s-)+c^{(r)}(\boldsymbol{X}(s-))\,y\big)}\,dF_r(y)-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)}\Big]\\&\simeq E\Big[\int_{\mathbb{R}}e^{i\,\boldsymbol{u}'\,\big(\boldsymbol{X}(t-)+c^{(r)}(\boldsymbol{X}(t-))\,y\big)}\,dF_r(y)-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(t-)}\Big].\end{aligned} \tag{3.9}$$

We therefore have that

$$\begin{aligned}
&E\Big[\sum_{t<s\le t+\Delta t}\big(e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s)}-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)}\big)\Big]\\
&\simeq\sum_{r=1}^{m_c}E\Big[\sum_{t<s\le t+\Delta t}\big(e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s)}-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(s-)}\big)\,\Big|\,N_r(\Delta t)=1,N_s(\Delta t)=0,s\neq r\Big]\Delta t\lambda_r+O((\Delta t)^2)\\
&=\Delta t\sum_{r=1}^{m_c}\lambda_r\,E\Big[\int_{\mathbb{R}}e^{i\,\boldsymbol{u}'\,\big(\boldsymbol{X}(t-)+c^{(r)}(\boldsymbol{X}(t-))\,y\big)}\,dF_r(y)-e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(t-)}\Big]+O((\Delta t)^2) \qquad (3.10)
\end{aligned}$$

following (3.7) and (3.9).

To conclude the derivation, we now apply the expectation operator to (3.3). We subsequently differentiate with respect to time the left side of (3.3) and the first two terms I, II on the right side of this equation given by (3.4) and (3.5). The time derivative of the third term III is then computed by simplifying (3.6) using (3.10). This yields (3.2) as a result. ■

We remark that (3.2) is not a differential equation for the characteristic function of $\boldsymbol{X}(t)$ since the drift and diffusion coefficients are arbitrary functions. It becomes a differential equation for $\boldsymbol{X}(t)$ if the drift and diffusion coefficients are polynomials of $\boldsymbol{X}(t)$ and the diffusion matrix $\boldsymbol{c}$ has a particular structure. For example, if $\boldsymbol{c}$ in (3.1) does not depend on $\boldsymbol{X}(t)$ so that the Poisson white noise is additive, (3.2) results in a PDE. If $\boldsymbol{c}$ is linear in $\boldsymbol{X}(t)$ so that the Poisson white noise is multiplicative, (3.2) becomes an integro-differential equation as the subsquent examples demonstrate.

As $\varphi(\boldsymbol{u},t)=\int_{\mathbb{R}^d}e^{i\boldsymbol{u}'\boldsymbol{x}}f(\boldsymbol{x},t)\,d\boldsymbol{x}$, the existence of an integrable function $g(\boldsymbol{x})$ such that $|\frac{\partial}{\partial t}f(\boldsymbol{x},t)|\le g(\boldsymbol{x})$ for $(\boldsymbol{x},t)\in D\times[0,T]$ guarantees the existence of $\frac{\partial}{\partial t}\varphi(\boldsymbol{u},t)$ by the dominated convergence theorem. In addition, a solution to (3.2) exists as the characteristic function of a random vector can always be computed. Using facts from probability theory, and assuming that $\boldsymbol{X}(t)$ has a density and finite moments of order q, $\varphi(\boldsymbol{u},t)$ satisfies the following conditions $\forall t$, cf. [11, p. 480]:

- $\varphi(\mathbf{0},t)=1$ where $\mathbf{0}\in\mathbb{R}^d$,
- $|\varphi(\boldsymbol{u},t)|\le 1$,
- $\varphi(\boldsymbol{u},t)\to 0$ and $\frac{\partial^q\varphi(\boldsymbol{u},t)}{\partial u_1^{q_1}\cdots\partial u_d^{q_d}}\to 0$ as $\|\boldsymbol{u}\|\to\infty$, where $\boldsymbol{u}=(u_1,\ldots,u_d)$ and $q_i\in\{0\}\cup\mathbb{Z}^+$ such that $\sum_{i=1}^d q_i=q$.

Since the pdf and the chf of $\boldsymbol{X}(t)$ are Fourier pairs, the pdf $f(\boldsymbol{x},t)$ can be obtained via

$$f(\boldsymbol{x},t)=\frac{1}{(2\pi)^d}\int_{\mathbb{R}^d}e^{-i\boldsymbol{u}'\boldsymbol{x}}\,\varphi(\boldsymbol{u},t)\,d\boldsymbol{u}.$$

We now demonstrate the application of (3.2) to commonly studied diffusion processes. In the following calculations, we use the fact that $E\big[\exp\big(i\,\boldsymbol{u}'\,\boldsymbol{X}(t-)\big)\big]=E\big[\exp\big(i\,\boldsymbol{u}'\,\boldsymbol{X}(t)\big)\big]$. To see this, observe that $\boldsymbol{X}(t)$ and $\boldsymbol{X}(t-)$ differ on the event $\{N_r(\Delta t)\ge 1\}$ with $P(N_r(\Delta t)\ge 1)\to 0$ as $\Delta t\to 0$.

Example 1 (Verhulst model) *Suppose that $X(t)$ is a real-valued diffusion process with $d=1$, $m_B=m_C=1$, $a(x)=\rho\,x-x^2$, $b(x)=x$, and $c(x)=x$ and let the jumps of $C(t)$ be distributed according to F.*

The three terms on the right side of (3.2) are

$$\begin{aligned}
& i\,u\,E\big[e^{i\,u\,X(t-)}\,\big(\rho\,X(t-)-X(t-)^2)\big)\big]=\rho\,u\frac{\partial\varphi(u,t)}{\partial u}+i\,u\,\frac{\partial^2\varphi(u,t)}{\partial u^2},\\
& -\frac{u^2}{2}\,E\big[e^{i\,u\,X(t-)}\,X(t-)^2\big]=\frac{u^2}{2}\,\frac{\partial^2\varphi(u,t)}{\partial u^2},\quad \text{and}\\
& \lambda\left(E\bigg[\int_{\mathbb{R}}e^{i\,u\,X(t-)\,(1+y)}\,dF(y)\bigg]-E\big[e^{i\,u\,X(t-)}\big]\right)=\lambda\bigg(\int\varphi\big(u\,(1+y),t\big)\,dF(y)-\varphi(u,t)\bigg)
\end{aligned}$$

since $\varphi(u,t)=E\big[\exp\big(i\,u\,X(t)\big)\big]$ and $\partial^r\varphi(u,t)/\partial u^r=E\big[\big(i\,u\,X(t)\big)^r\,\exp\big(i\,u\,X(t)\big)\big]$.

The expressions of these terms and (3.2) give the following integro-differential equation for the characteristic function of the state $X(t)$ of the Verhulst model

$$\frac{\partial\varphi(u,t)}{\partial t}=\rho\,u\frac{\partial\varphi(u,t)}{\partial u}+i\,u\,\frac{\partial^2\varphi(u,t)}{\partial u^2}+\frac{u^2}{2}\,\frac{\partial^2\varphi(u,t)}{\partial u^2}+\lambda\bigg(\int\varphi\big(u\,(1+y)\big)\,dF(y)-\varphi(u,t)\bigg). \quad (3.11)$$

Example 2 (Duffing model) *Let $Y(t)$ be the displacement of a Duffing oscillator subjected to Gaussian and Poisson white noise processes whose jumps are distributed according to F so that the bivariate process $\boldsymbol{X}(t)$ with components $X_1(t)=Y(t)$ and $X_2(t)=\dot{Y}(t)$ satisfies the Itô differential equation*

$$\begin{cases} dX_1(t) &= X_2(t-)\,dt,\\ dX_2(t) &= -\nu^2\,\big(X_1(t-)+\alpha\,X_1(t-)^3\big)\,dt-2\,\zeta\,\nu\,X_2(t-)\,dt+b\,dB(t)+c\,dC(t),\end{cases} \quad (3.12)$$

where $\zeta\in(0,1)$ is the damping ratio, ν denotes the initial frequency, and α,b,c are real constants.

The differential equation of (3.2) for $d=2$, $u=(u_1,u_2)$, $m_B=m_C=1$, $\boldsymbol{a}(x)=\begin{bmatrix}a_1(x)\\a_2(x)\end{bmatrix}=\begin{bmatrix}x_2\\-\nu^2\,(x_1+\alpha\,x_1^3)-2\,\zeta\,\nu\,x_2\end{bmatrix}$, $\boldsymbol{b}=\begin{bmatrix}0\\b\end{bmatrix}$, $\boldsymbol{c}=\begin{bmatrix}0\\c\end{bmatrix}$ gives

$$\begin{aligned}
\frac{\partial\varphi(\boldsymbol{u},t)}{\partial t}=i\,\Bigg(&u_1\,E\big[e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(t-)}\,X_2(t-)\big]\\
&+u_2\,E\big[e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(t-)}\,\big(-\nu^2\,(X_1(t-)+\alpha\,X_1(t-)^3)-2\,\zeta\,\nu\,X_2(t-)\big)\big]\\
&-\frac{1}{2}\,u_2^2\,b^2\,E\big[e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(t-)}\big]+\lambda\,\bigg(\int_{\mathbb{R}}E\big[e^{i\,\big(u_1\,X_1(t-)+u_2\,X_2(t-)+u_2\,c\,y\big)}\big]\,dF(y)-E\big[e^{i\,\boldsymbol{u}'\,\boldsymbol{X}(t-)}\big]\bigg).
\end{aligned}$$

Since

$$\begin{aligned}
\int_{\mathbb{R}}E\big[e^{i\,\big(u_1\,X_1(t-)+u_2\,X_2(t-)+u_2\,c\,y\big)}\big]\,dF(y)&=E\Big[e^{i\,\big(u_1\,X_1(t-)+u_2\,X_2(t-)\big)}\Big]\int_{\mathbb{R}}e^{iu_2cy}\,dF(y)\\
&=\varphi(\boldsymbol{u},t)\phi(cu_2)
\end{aligned}$$

where ϕ is the characteristic function of the jumps, the above simplifies to

$$\begin{aligned}
\frac{\partial\varphi(\boldsymbol{u},t)}{\partial t}&=u_1\,\frac{\partial\varphi(\boldsymbol{u},t)}{\partial u_2}-\nu^2\,u_2\,\frac{\partial\varphi(\boldsymbol{u},t)}{\partial u_1}+\alpha\,\nu^2\,u_2\,\frac{\partial^3\varphi(\boldsymbol{u},t)}{\partial u_1^3}-2\,\zeta\,\nu\,u_2\,\frac{\partial\varphi(\boldsymbol{u},t)}{\partial u_2}\\
&\quad-\frac{b^2\,u_2^2}{2}\,\varphi(\boldsymbol{u},t)+\lambda\varphi(\boldsymbol{u},t)\,[\phi(cu_2)-1]. \quad (3.13)
\end{aligned}$$

3.2 Fokker-Planck equation

If $\boldsymbol{c} = \mathbf{0}_{d\times m_C}$ (a $d \times m_C$ matrix of zeros) in (3.1), i.e. the Poisson white noise is absent, the Fokker-Planck equation can be recovered [11, p. 482] by applying the Fourier transform to (3.2). The pdf of $\boldsymbol{X}(t)$ satisfies the PDE

$$\frac{\partial f(\boldsymbol{x},t)}{\partial t} = -\sum_{i=1}^{d} \frac{\partial}{\partial x_i}[a_i(\boldsymbol{x})\, f(\boldsymbol{x},t)] + \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j}\left[(\boldsymbol{b}(\boldsymbol{x})\boldsymbol{b}(\boldsymbol{x})')_{ij}\, f(\boldsymbol{x},t)\right] \tag{3.14}$$

for $(\boldsymbol{x},t) \in \mathbb{R}^d \times [0,T]$ subject to the constraint

$$\int_{\mathbb{R}^d} f(\boldsymbol{x},t)\, d\boldsymbol{x} = 1, \;\; t \in [0,T] \tag{3.15}$$

and the boundary conditions

$$\begin{aligned} &\lim_{|x_i|\to\infty} a_i(\boldsymbol{x}) f(\boldsymbol{x},t) = 0, \\ &\lim_{|x_i|\to\infty} (\boldsymbol{b}(\boldsymbol{x})\boldsymbol{b}(\boldsymbol{x})')_{ij}\, f(\boldsymbol{x},t) = 0, \\ &\lim_{|x_i|\to\infty} \frac{\partial}{\partial x_i}[(\boldsymbol{b}(\boldsymbol{x})\boldsymbol{b}(\boldsymbol{x})')_{ij}\, f(\boldsymbol{x},t)] = 0, \end{aligned}$$

for $i, j = 1, \ldots, d$ where $a_i(x)$ is the i-th row of $\boldsymbol{a}(x)$ while $(\boldsymbol{b}(\boldsymbol{x})\boldsymbol{b}(\boldsymbol{x})')_{ij}$ is the (i,j)-th component of this $d \times d$ matrix. The Fokker-Planck equation is a parabolic PDE whose maximum order of partial derivatives is at most 2 unlike the PDE for the chf, e.g. (3.11), (3.13).

If the SDE is driven by Poisson white noise, an extended Fokker-Planck equation can be derived under special cases [11,12]. To illustrate, if $X(t) \in \mathbb{R}$ satisfies the SDE

$$dX(t) = -\rho X(t-)\, dt + dC(t), \;\; t \geq 0$$

where $\rho > 0$, $C(t) = \sum_{w=1}^{N(t)} Y_w$ is a compound Poisson process that depends on the Poisson process $N(t)$ with intensity λ and $\{Y_w\}$ being independent copies of Y, applying (3.2), the PDE for the characteristic function of $X(t)$ is

$$\frac{\partial \varphi(u,t)}{\partial t} = -\rho u \frac{\partial \varphi(u,t)}{\partial u} + \lambda \left(E[e^{iuY}] - 1\right) \varphi(u,t).$$

Following [11, p. 484], the Fourier transform of this PDE yields

$$\frac{\partial f(x,t)}{\partial t} = \rho \frac{\partial}{\partial x}(x f(x,t)) + \lambda \sum_{k=1}^{\infty} \frac{(-1)^k E[Y^k]}{k!} \frac{\partial^k f(x,t)}{\partial x^k} \tag{3.16}$$

under some conditions which include that Y has finite moments of any order.

We finally remark that for some systems driven by Lévy white noise with α-stable random variable increments, a PDE for the state chf may be formulated while it may not be possible to derive a Fokker-Planck type PDE for the pdf if $\alpha \in (0,2)$ [11, Ex. 7.34].

4 Neural network-based representation for the state pdf

We illustrate how a neural network can be trained to approximate solutions to the differential equations introduced in Section 3. Section 4.1 discusses the formulation of the loss function of the neural network to incorporate the constraints of these 2 types of differential equations. The physics-informed neural network approach is then applied to compute the pdf and the chf of the Brownian motion for which analytical solutions are available. Finally, a comparison is made in Section 4.2 which elaborates on the advantages and disadvantages of solving each differential equation using neural networks from an analytical and numerical perspective.

4.1 Training neural networks to approximate the pdf and the chf

We discuss how neural networks can be trained to solve the differential equations introduced in Sections 3.1 and 3.2 and comment on their approximation quality.

An important constraint for the Fokker-Planck equation (3.14) is that the pdf must integrate to 1 at all times. It was observed in [2,3] that failure to enforce this constraint in the neural network training resulted in a pdf that did not match the analytical solution. It was therefore proposed to represent $f(x,t)$ by

$$f(\boldsymbol{x},t)=\frac{e^{-v(\boldsymbol{x},t)}}{\int_{\mathbb{R}^d} e^{-v(\boldsymbol{x},t)}\,d\boldsymbol{x}} \tag{4.1}$$

for some function $v(\boldsymbol{x},t)$ so that $f(x,t)$ is positive and integrates to 1. Hence, if $\mathcal{N}$ represents the operator of the Fokker-Planck PDE, instead of solving for $f(\boldsymbol{x},t)$ such that $\mathcal{N}[f(\boldsymbol{x},t)]=0$ and (3.15) holds, we solve for $v(\boldsymbol{x},t)$ such that $\mathcal{M}[v(\boldsymbol{x},t)]=0$ for some operator $\mathcal{M}$. Our objective is therefore to find a neural network approximation $\widetilde{v}(\boldsymbol{x},t)$ for which $\mathcal{M}[\widetilde{v}(\boldsymbol{x},t)]=0$ is satisfied so that an approximation $\widetilde{f}(\boldsymbol{x},t)$ to the state pdf $f(\boldsymbol{x},t)$ can be obtained via $\widetilde{f}(\boldsymbol{x},t)=\frac{e^{-\widetilde{v}(\boldsymbol{x},t)}}{\int_{\mathbb{R}^d} e^{-\widetilde{v}(\boldsymbol{x},t)}\,d\boldsymbol{x}}$.

Suppose that the pdf $f(\boldsymbol{x},0)=\frac{e^{-v(\boldsymbol{x},0)}}{\int_{\mathbb{R}^d} e^{-v(\boldsymbol{x},0)}\,d\boldsymbol{x}}$ of $\boldsymbol{X}(0)$ is available. For a specified neural network architecture (number of hidden layers, number of neurons per hidden layer, and type of activation function), Algorithm 1 summarizes how a neural network approximation $\widetilde{v}(\boldsymbol{x},t)$ for $v(\boldsymbol{x},t)$ on $(\boldsymbol{x},t)\in\mathbb{R}^{d+1}\times[0,T]$ can be obtained. The input layer in this case consists of $d+1$ neurons while there is only 1 neuron in the output layer.

Algorithm 1 Training neural networks to solve the Fokker-Planck equation

```
1: Truncate the spatial domain by choosing a compact $D \subset \mathbb{R}^d$ sufficiently large so that most of the probability mass of $\boldsymbol{X}(t)$ is contained in $D$
2: Select $N_{Op}$ collocation points $\{(\boldsymbol{x}_i^{Op}, t_i^{Op})\}_{i=1}^{N_{Op}} \subset D \times [0,T]$ to enforce the governing equations
3: Select $N_{IC}$ collocation points $\{(\boldsymbol{x}_i^{IC}, 0)\}_{i=1}^{N_{IC}} \subset D \times 0$ to enforce the initial condition
4: Solve for the neural network parameters to minimize the loss
```

$$\mathcal{L}=\frac{1}{N_{Op}}\sum_{i=1}^{N_{Op}}(\mathcal{M}[\widetilde{v}(\boldsymbol{x}_i^{Op},t_i^{Op})])^2+\frac{1}{N_{IC}}\sum_{i=1}^{N_{IC}}(\widetilde{v}(\boldsymbol{x}_i^{IC},0)-v(\boldsymbol{x}_i^{IC},0))^2 \tag{4.2}$$

Since the Fokker-Planck equation is generally defined on an unbounded spatial domain, enforcing the boundary conditions is numerically challenging regardless of the numerical scheme employed.

However, once the optimal neural network parameters are found, $\widetilde{v}(\boldsymbol{x}, t)$ and its derivatives can be queried for any point $(\boldsymbol{x}, t) \in D \times [0, T]$ via automatic differentiation in TensorFlow. This can be used to verify that the boundary conditions are met.

We remark that [24] also pursued a neural network solution to the steady-state Fokker Planck equation ($\frac{\partial f(\boldsymbol{x},t)}{\partial t} = 0$) but did not utilize the transformation (4.1) to incorporate the normalization constraint. Instead, if $\widetilde{f}(\boldsymbol{x})$ is the neural network approximation to the steady-state pdf, their loss function included a discretized version of $(\int_{\mathbb{R}^d} f(\boldsymbol{x})\, d\boldsymbol{x} - 1)^2$ in addition to enforcing the governing PDE and the boundary condition. This approach does not guarantee that $\widetilde{f}(x) \geq 0$ and moreover, it is not clear how many time points are needed to incorporate the constraint (3.15) to the loss function for solving the time-varying Fokker Planck equation. In the remainder of this work, we adopt the approach in [2,3].

Unlike the probability density function, the characteristic function of a random vector is generally complex-valued. This means that the output layer of a neural network representation $\widetilde{\varphi}(\boldsymbol{u}, t)$ of $\varphi(\boldsymbol{u}, t)$ has 2 neurons, each corresponding to the real and imaginary parts of $\varphi(\boldsymbol{u}, t)$. In some cases, it will be demonstrated in Section 5 that probabilistic arguments on $\boldsymbol{X}(t)$ can be made to show that $f(\boldsymbol{x}, t)$ is symmetric about $\boldsymbol{x} = \boldsymbol{0}$ thereby showing that $\varphi(\boldsymbol{u}, t)$ is real-valued. Assuming that a differential equation for the chf can be derived, denote its governing equation by $\mathcal{Q}[\varphi(\boldsymbol{u}, t)] = 0$ for some operator $\mathcal{Q}$. For a specified architecture, Algorithm 2 elaborates how a neural network approximation can be obtained for $\varphi(\boldsymbol{u}, t)$ on $(\boldsymbol{u}, t) \in \mathbb{R}^{d+1} \times [0, T]$ assuming that the chf $\varphi(\boldsymbol{u}, 0)$ of $\boldsymbol{X}(0)$ is available. As with the Fokker-Planck equation, the input layer has $d + 1$ neurons.

Algorithm 2 Training neural networks to solve the diff. eq. for the characteristic function

```
Truncate the frequency domain by choosing a compact $D \subset \mathbb{R}^d$ sufficiently large so that $D$ contains most of the frequencies of the chf of $\boldsymbol{X}(t)$
Select $N_{Op}$ collocation points $\{(\boldsymbol{u}_i^{Op}, t_i^{Op})\}_{i=1}^{N_{Op}} \subset D \times [0, T]$ to enforce the governing equations
Select $N_{IC}$ collocation points $\{(\boldsymbol{u}_i^{IC}, 0)\}_{i=1}^{N_{IC}} \subset D \times 0$ to enforce the initial condition
Select $N_0$ collocation points $\{(\boldsymbol{0}, t_i^0)\}_{i=1}^{N_0} \subset \boldsymbol{0} \times [0, T]$ to enforce the condition at the origin
Solve for the neural network parameters to minimize the loss
```

$$\mathcal{L} = \frac{1}{N_{Op}} \sum_{i=1}^{N_{Op}} \left| \mathcal{Q}[\widetilde{\varphi}(\boldsymbol{u}_i^{Op}, t_i^{Op})] \right|^2 + \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} \left| \widetilde{\varphi}(\boldsymbol{u}_i^{IC}, 0) - \varphi(\boldsymbol{u}_i^{IC}, 0) \right|^2 + \frac{1}{N_0} \sum_{i=1}^{N_0} \left| \widetilde{\varphi}(\boldsymbol{0}, t_i^0) - 1 \right|^2 \tag{4.3}$$

In the loss function (4.3), $|\cdot|$ refers to the magnitude of a complex number since $\widetilde{\varphi}(\boldsymbol{u}, t)$ is complex-valued. As in the Fokker-Planck equation, it is numerically challenging to incorporate the boundary conditions of the differential equation for the chf however, once the neural network approximation is attained, it can be queried to ensure that such conditions are met. As for the constraint that $|\varphi(\boldsymbol{u}, t)| \leq 1$, it may be possible to apply a transformation to $\varphi(\boldsymbol{u}, t)$ to impose this condition. Our numerical experiments revealed that the resulting neural network solution does not violate this constraint.

In order to assess the quality of the neural network approximation to the PDE solution, it was demonstrated in [20, Theorem 7.1] that if a neural network with a single hidden layer is used to approximate the solution to a class of quasilinear parabolic PDEs, the loss function approaches zero as the number of neurons increases. This holds if the terms in the PDE satisfy a Lipschitz condition. The proof of this result can be adapted and extended to other types of PDEs provided that similar Lipschitz-type conditions are met. Furthermore, [20, Theorem 7.2] shows that the neural network approximation converges to the unique PDE solution under additional assumptions on the parabolic PDE. While this result can be applied to the Fokker-Planck equation, it is not directly applicable to the chf PDE since the order of partial derivatives of the latter can exceed 2.

Following the above discussion on training neural networks to solve the Fokker-Planck equation or the differential equation for the chf, we apply this methodology to a simple example for which the analytical solution of both the pdf (Example 3) and the chf (Example 4) are available for all time.

Example 3 *Let $X(t)$ satisfy $dX(t) = \sigma dB(t), t \in [0,1]$, with $X(0) \sim N(0,\nu)$ where $\sigma, \nu = 1$ and $B(t)$ is the Brownian motion. We use physics-informed neural networks to solve the Fokker-Planck equation and reconcile the approximation with the analytical solution.*

The Fokker-Planck equation for $X(t)$ is given by

$$\frac{\partial f(x,t)}{\partial t} - \frac{1}{2}\sigma^2 \frac{\partial^2 f(x,t)}{\partial x^2} = 0$$

whose analytical solution can be readily verified as $f(x,t) = \frac{1}{\sqrt{2\pi(\nu+\sigma^2 t)}} e^{-\frac{x^2}{2(\nu+\sigma^2 t)}}$. Applying the transformation $f(x,t) = \frac{e^{-v(x,t)}}{\int_{-\infty}^{\infty} e^{-v(x,t)}\,dx}$ yields the following PDE for $v(x,t)$:

$$\mathcal{M}[v(x,t)] = \frac{\partial v(x,t)}{\partial t} - \frac{\sigma^2}{2}\left(\frac{\partial^2 v(x,t)}{\partial x^2} - \left(\frac{\partial v(x,t)}{\partial x}\right)^2\right) - \frac{\int_{-\infty}^{\infty} e^{-v(x,t)} \frac{\partial v(x,t)}{\partial t}\,dx}{\int_{-\infty}^{\infty} e^{-v(x,t)}\,dx} = 0, \tag{4.4}$$

for which we seek a neural network approximation. Observe that (4.4) does not have a unique solution because if $v(x,t)$ satisfies (4.4) then so does $v(x,t) + g(t)$ for some differentiable function g with $g(0) = 0$. Analytically, this does not pose an issue since we still recover $f(x,t)$ but as will be noted in the applications in Section 5, it may be a source of numerical issues which was not explored in [2, 3].

The neural network architecture we used to solve (4.4) on the truncated spatial domain $D = [-7,7]$ is comprised of an input layer with 2 neurons, 4 hidden layers with 100 neurons each, and an output layer with a single neuron. Hyperbolic tangent activation functions were utilized in all simulations in this work. To construct the loss function (4.2) for Example 3, we generated a regular grid of $N_{Op} = 151 \times 101$ points in the input domain $[-7,7] \times [0,1]$ to enforce the governing equation while $N_{IC} = 100$ points were randomly chosen among the 151 equally spaced points in D to impose the initial condition. The same mesh on D was used to numerically approximate the integral terms that appear in the operator (4.4). The value of the loss function (4.2) of the trained neural network is 7.994686×10^{-7}. Figure 2 examines the performance of the neural network approximation compared to the analytical solution. The left panel displays plots of $v(x,t) = \frac{x^2}{2(1+t)}$ (solid) and $\widetilde{v}(x,t)$ (dashed) for $t = 1$. As can be seen, $\widetilde{v}(x,t)$ is a shifted version of $v(x,t)$ because (4.4) does not have a unique solution. This behavior was also observed for other values of t. However, the middle panel indicates that $f(x,t)$ (solid) and $\widetilde{f}(x,t)$ (dashed) at $t = 1$ coincide once the solution is normalized because (4.1) is unaffected by the vertical shift in the left panel. The right panel plots the error $\max_x |f(x,t) - \widetilde{f}(x,t)|$ thereby confirming that the neural network is able to recover the analytical solution.

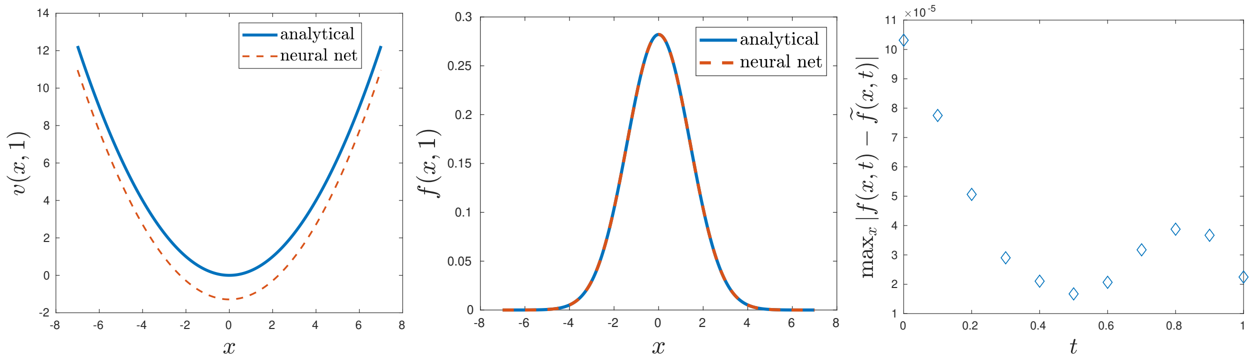

Figure 2: Comparison between the neural network approximation and the analytical solution of the Fokker-Planck equation for Example 3. Left: $v(x,1)$ (solid) vs $\widetilde{v}(x,1)$ (dashed). Middle: $f(x,1)$ (solid) vs $\widetilde{f}(x,1)$ (dashed). Right: $\max_x |f(x,t) - \widetilde{f}(x,t)|$ for $t = [0,1]$ with increments of 0.1.

Example 4 *Let $X(t)$ satisfy the SDE and initial conditions in Example 3. A physics-informed neural network is trained to solve the PDE for the chf of $X(t)$ which is then reconciled with the analytical solution.*

The PDE for the chf of $X(t)$ has the form (see (3.2))

$$\mathcal{Q}[\varphi(u,t)] = \frac{\partial \varphi(u,t)}{\partial t} + \frac{1}{2}u^2\sigma^2\varphi(u,t) = 0 \tag{4.5}$$

that admits the analytical solution $\varphi(u,t) = e^{-\frac{1}{2}(\nu+\sigma^2 t)u^2}$ which is the characteristic function of a Gaussian random variable with mean 0 and variance $\nu + \sigma^2 t$. Without knowledge of this analytical solution, we anticipate that the chf is real-valued since $X(t)$ is a scaled Brownian motion which is a Gaussian process with mean 0. The pdf of $X(t)$ is therefore symmetric with respect to the spatial origin.

The same neural network architecture in Example 3 was implemented for this example with $D = [-7,7]$ as the truncated frequency domain for $\varphi(x,t)$. To formulate the loss function (4.3) for Example 4, we used $N_{Op} = 15000$ latin hypercube samples [22] in $D \times [0,1]$, $N_{IC} = 100$ random points in D, and $N_0 = 100$ equally spaced points in $[0,1]$ so that the governing equation, initial condition, and condition at the origin hold, respectively. The trained neural network attained a loss function value of 3.4449415×10^{-6}. Figure 3 exhibits the comparison between the neural network approximation and the known analytical soltion. The left panel highlights that $\widetilde{\varphi}(u,t)$ (dashed) matches $\varphi(u,t)$ (solid) for $t = 1$ while the right panel affirms that the neural network solution is comparable to the analytical solution based on the plotted values of $\max_u |\varphi(u,t) - \widetilde{\varphi}(u,t)|$ for various time points.

4.2 Comparison between the differential equations for the chf and the pdf

Our objective in this work is to obtain the pdf of the state $\boldsymbol{X}(t)$ that satisfies a stochastic differential equation. From the subsections above, two approaches have been presented to accomplish this, namely, solving the Fokker-Planck equation and solving the differential equation for the characteristic function of $\boldsymbol{X}(t)$ and computing its Fourier transform. In the following, the advantages and disadvantages of each approach are outlined, see Table I for a summary.

The main drawback of the Fokker-Planck equation is that it may not be possible to derive a PDE for the pdf of $\boldsymbol{X}(t)$ if the forcing term is not Gaussian white noise (GWN). Even if an extended

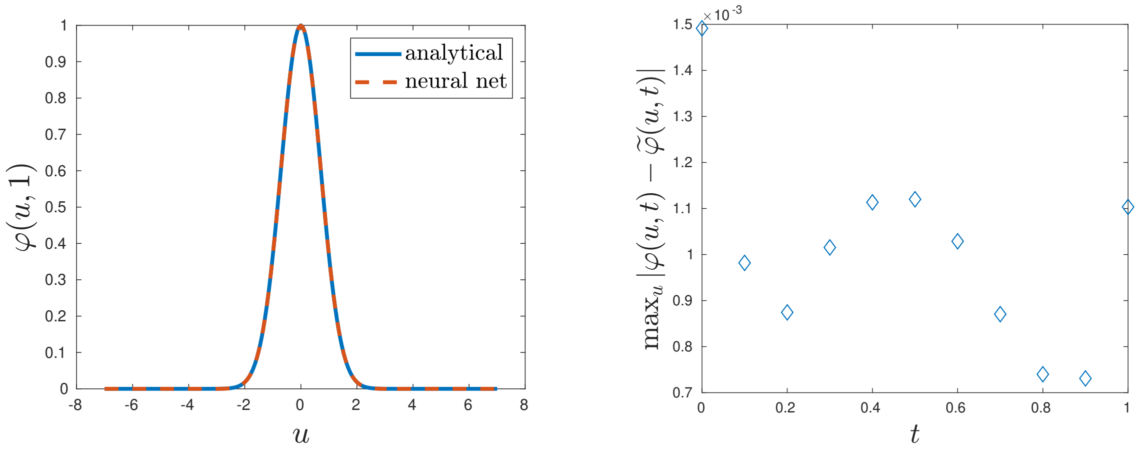

Figure 3: Comparison between the neural network approximation and the analytical solution for the chf PDE in Example 4. Left: $\varphi(u,1)$ (solid) vs $\widetilde{\varphi}(u,1)$ (dashed). Right: $\max_u |\varphi(u,t) - \widetilde{\varphi}(u,t)|$ for $t = [0,1]$ with increments of 0.1.

	Fokker-Planck PDE	Chf diff. eq.
Can be derived?	available for GWN forcing but may be unavailable otherwise	polynomial drift and diffusion and special structure for jump coefficient
Dimension	1 (real-valued)	2 (complex-valued)
Transformation	needed to incorporate normalization constraint	none
Diff. eq. type	integro-differential equation if transformation applied	generally an integro-differential equation
Derivative order	at most 2 for GWN forcing	max degree of polynomial coefficient
Post-processing	none	Fourier transform

Table I: Comparison between the Fokker-Planck equation and the differential equation for the chf.

Fokker-Planck equation as in (3.16) can be deduced, it may not be favorable to solve this PDE using any numerical scheme because truncating the series expression may introduce numerical errors if the jumps have non-zero moments of any order. From a numerical perspective, enforcing the constraint (3.15) can be challenging. While a transformation such as (4.1) resolves this issue, our numerical experiments in Section 5 reveal that this may not be suitable in high dimensions. In particular, the non-uniqueness of the solution to the PDE for $v(\boldsymbol{x},t)$ and the absence of constraints on $v(\boldsymbol{x},t)$ imply that it is likely for $v(\boldsymbol{x},t)$ to be very negative rendering $\int_{\mathbb{R}^d} e^{-v(\boldsymbol{x},t)}\, d\boldsymbol{x}$ impossible to be approximated numerically. The loss function $\mathcal{L}$ becomes nan as a result and the optimization algorithm is unable to continue searching for a local minimum.

However, if the Fokker-Planck equation can be derived for $\boldsymbol{X}(t)$ and if the constraint can be seamlessly incorporated, solving for the pdf in this manner is convenient since the solution to the Fokker-Planck PDE is already the quantity of interest, $f(\boldsymbol{x},t)$, which does not need to be post-processed. For systems with Gaussian white noise forcing, the maximum number of derivatives in the PDE is at most 2.

In contrast, the differential equation for the characteristic function can only be derived if the drift and diffusion coefficients are polynomials and that the diffusion matrix for the Poisson white noise forcing has a special structure. Despite this, an advantage of this approach is that there are scenarios [11] for which a differential equation for the chf can be derived while a PDE for the pdf is not available, especially when the forcing term has jumps such as the Poisson and Lévy white noise. In addition, the constraints of this equation are more convenient to implement as they do not involve any normalization.

The disadvantages of solving the differential equation for the chf include the fact that the chf is generally complex-valued. This implies that a system of differential equations needs to be solved. From (3.2), it can be observed that the chf differential equation is an integro-differential equation in which the maximum order of the derivatives is equivalent to the highest polynomial degree of the drift and diffusion coefficient. Finally, the solution to the differential equation has to be post-processed via the Fourier transform to derive the pdf of $\boldsymbol{X}(t)$.

5 Applications

We investigate the capabilities of physics-informed neural networks to solve the Fokker-Planck equation or the differential equation for the characteristic function that arise from various diffusion processes. Since the analytical solution to these equations is unavailable, the target solution is approximated via Monte Carlo simulation. The applications below serve to highlight the advantages and disadvantages outlined in Section 4.2 and also demonstrate how the neural network solution can be utilized to study probabilistic phenomenon. They also result in differential equations which are different from the ones tackled in [2, 3, 18, 20, 24].

Section 5.1 and 5.2 consider the 1-dimensional Verhulst model subject to Gaussian and Poisson white noise, respectively. In the former, the Fokker-Planck equation is solved to show that the neural network solution can recover the known analytical stationary density. In the latter, the differential equation for the chf is solved which represents an example of a system of integro-differential equations. Section 5.3 is concerned with reconciling the pdf obtained from the Fokker-Planck equation and the one obtained from solving the chf PDE for the Duffing oscillator subject to Gaussian white noise. Section 5.4 revisits the Duffing oscillator and illustrates how a Poisson white noise forcing with sufficiently large jump intensity yields a chf that is similar to what would be obtained under Gaussian white noise. Finally, Section 5.5 deals with an example of solving the chf PDE in a 3-dimensional frequency domain.

The Python scripts written for all simulations presented is readily available upon request. We also reiterate that our objective is not to seek the optimal neural network architecture nor identify the ideal number of collocation points. Rather, we examine the feasibility of a neural network approximation for the state pdf. In practice, constructing a neural network involves a training, validation, and testing phase [10] with each phase relying on distinct sets of collocation points. In the simulations below, the number of collocation points for the training phase can be increased to see if the loss from the training phase decreases. The testing phase can serve to prevent the occurrence of overfitting. We do not undertake testing and validation in this work since we compare the neural network approximation with the estimate produced by Monte Carlo simulation.

5.1 Verhulst model with Gaussian white noise

Suppose that $X(t)$ satisfies the SDE given by the Verhulst model

$$dX(t) = (\rho X(t) - X(t)^2)\,dt + \sigma X(t)\,dB(t), \quad t \geq 0,$$

where $B(t)$ is the Brownian motion. It will be shown that the neural network representation of the pdf of $X(t)$ coincides with the analytical stationary pdf.

By applying (3.14), the Fokker-Planck equation for this SDE is

$$\frac{\partial f(x,t)}{\partial t} = -\frac{\partial}{\partial x}((\rho x - x^2) f(x,t)) + \frac{\sigma^2}{2}\frac{\partial^2}{\partial x^2}(x^2 f(x,t)). \tag{5.1}$$

Following the calculations in [11, p. 72], the analytical stationary density is $f_s(x) = kx^{2(\rho/\sigma^2-1)}e^{-2x/\sigma^2}$ for $x > 0$ where k is the normalizing constant. For the neural network implementation, we apply the transformation (4.1) so that we solve for $v(x,t)$ satisfying

$$\begin{aligned}\mathcal{M}[v(x,t)] = v_t(x,t) + (-\rho + 2x + \sigma^2) - (x^2 - \rho x + 2\sigma^2 x) v_x(x,t) + \\ \frac{\sigma^2 x^2}{2}(v_x(x,t)^2 - v_{xx}(x,t)) + \frac{c'(t)}{c(t)} = 0\end{aligned} \tag{5.2}$$

in which $c(t) = \int_{\mathbb{R}} e^{-v(x,t)}\,dx$. In our simulations, we chose $\rho = 2, \sigma = 1$ and $X(0) \sim \Gamma(k = 1, \theta = 1.5)$, i.e. a gamma distribution with shape $k = 1$ and scale $\theta = 1.5$, which implies that $f(x,0) = \frac{1}{\Gamma(k)\theta^k}e^{-\left(\frac{x}{\theta} - \ln x^{k-1}\right)}$ and $v(x,0) = \frac{x}{\theta} - (k-1)\ln x$. With these parameters, the stationary distribution can then be represented as $f_s(x) = 4x^2e^{-2x} = 4e^{-v_s(x)}$, $v_s(x) = 2x - 2\ln x$.

We pursued a neural network solution to (5.2) on the truncated spatial domain $x \in [0,9]$ for $t \in [0,4]$. The network architecture is composed of 2 neurons for the input layer, 1 neuron for the output layer, and 4 hidden layers with 100 neurons each. A mesh of 151 equally spaced points in the spatial domain and 301 equally spaced points in the time domain was constructed to obtain $N_{Op} = 151 \times 301$ collocation points to enforce the governing equation. This mesh was also used to numerically compute the integral terms in (5.2). Out of the 151 points in $x \in [0,9]$, $N_{IC} = 101$ points were randomly chosen to impose the initial condition constraints.

Figures 4 and 5 exhibit the results of the trained neural network which attained a loss value of 0.000163. Figure 4 compares the analytical solution with the neural network representation; in particular, the left panel displays $v_s(x)$ (solid) and $\widetilde{v}(x,4)$ (dashed) which differ by a constant while the right panel displays $f_s(x)$ (solid) and $\widetilde{f}(x,4)$ (dashed) which coincide. These plots show that the neural network solution is able to recover the analytical stationary density at $t = 4$ and further confirms that (5.2) has no unique solution as remarked above. Figure 5 plots the neural network solution $\widetilde{f}(x,t)$ at $t = 0$ (left panel) and $t = 0.5$ (right panel) to illustrate that it is consistent with histograms of $X(0)$ and $X(0.5)$ obtained from 100000 Monte Carlo samples.

5.2 Verhulst model with Poisson white noise

Let $X(t)$ satisfy the SDE given by the Verhulst model

$$dX(t) = (\rho X(t-) - X(t-)^2)\,dt + X(t-)\,dC(t), \quad t \geq 0,$$

where $C(t) = \sum_{k=1}^{N(t)} Y_k$ is a compound Poisson process that depends on a Poisson process $N(t)$ with intensity λ and iid random variables Y_k with distribution F. It will be demonstrated that the neural

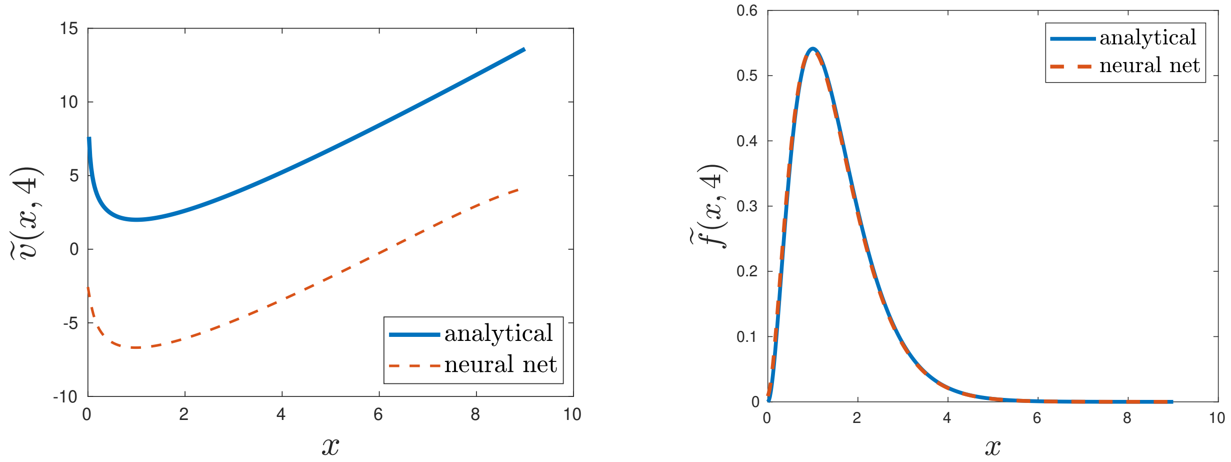

Figure 4: Comparison between the neural network approximation and the analytical stationary pdf for Section 5.1. Left: $v_s(x)$ (solid) vs $\widetilde{v}(x,4)$ (dashed). Right: $f_s(x)$ (solid) vs $\widetilde{f}(x,4)$ (dashed).

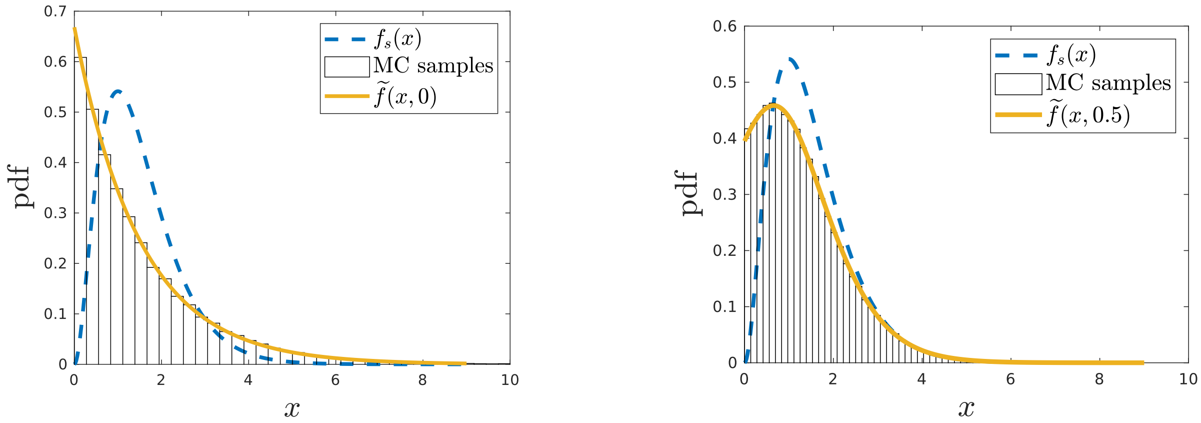

Figure 5: Comparison between the neural network approximation and Monte Carlo samples of $X(t)$ for Section 5.1. Left: histogram of $X(0)$ and $\widetilde{f}(x,0)$ (solid). Right: histogram of $X(0.5)$ and $\widetilde{f}(x,0.5)$ (solid). The analytical stationary density $f_s(x)$ is present in both plots (dashed).

network representation of the real and imaginary parts of the chf of $X(t)$ closely approximates the estimate provided by Monte Carlo simulation.

According to (3.2) and (3.11), the chf of $X(t)$ satisfies

$$\mathcal{Q}[\varphi(u,t)] = \frac{\partial \varphi(u,t)}{\partial t} - \rho u \frac{\partial \varphi(u,t)}{\partial u} - iu \frac{\partial^2 \varphi(u,t)}{\partial u^2} - \lambda \left[\int_{\mathbb{R}} \varphi(u(1+y),t)\, dF(y) - \varphi(u,t) \right] = 0 \quad (5.3)$$

which is complex-valued and is consequently a system of partial integro-differential equations. For demonstration, the following parameters were utilized: $\rho = 2, \lambda = 12$, Y_k is a discrete random variable taking on values $\{z_k\}_{k=1}^{7} = \left\{-\frac{1}{2} + \frac{k-1}{6}\right\}_{k=1}^{7}$ with equal probability so that

$$\int_{\mathbb{R}} \varphi(u(1+y),t)\, dF(y) = \frac{1}{7} \sum_{k=1}^{7} \varphi(u(1+z_k),t)$$

in (5.3), and $X(0) \sim U(0.5, 5.5)$ with $\varphi(u,0) = \left(\frac{\sin(5.5u)-\sin(0.5u)}{5u}\right) - i\left(\frac{\cos(5.5u)-\cos(0.5u)}{5u}\right)$. We seek a neural network approximation $\widetilde{\varphi}(u,t) \in \mathbb{C}$ such that $\mathcal{Q}[\widetilde{\varphi}(u,t)] = 0$ on the truncated domain $(u,t) \in [0,10] \times [0,1]$. It is furthermore assumed that $\widetilde{\varphi}(u,t) = 0$ whenever $u > 10$ to avoid extrapolating the neural network solution when evaluating the expression $\varphi(u(1+y),t)$ in (5.3). Note that this assumption is not restricted to the approach pursued here in solving differential equations; such assumption would have to be made for other numerical schemes.

The neural network utilized is composed of an input and output layer with 2 neurons each and 4 hidden layers with 100 neurons each. To construct the loss function (4.3), $N_{IC} = 400$ equally spaced points in $(0,10]$, $N_0 = 100$ equally spaced points in $[0,1]$, and a regular grid of $N_{Op} = 20000$ points consisting of 400 points in $[0,10]$ and 50 points in $[0,1]$ were generated which yielded a loss value of 4.9294514×10^{-4}. The neural network solution is then compared to the chf $\varphi^{MC}(u,t)$ resulting from 50000 Monte Carlo samples of $X(t)$ simulated through forward Euler.

Figures 6 and 7 display the comparison between $\widetilde{\varphi}(u,t)$ and $\varphi^{MC}(u,t)$, $u \geq 0$, for $t = 0.25$ and $t = 1$, respectively. In each figure, the left subplot shows the real part of the characteristic function while the right subplot shows the imaginary part. As the plots indicate, both approximations to the actual chf $\varphi(u,t)$ are similar. The discrepancy between the two approximations for u close to 10 is due to the assumption imposed that $\widetilde{\varphi}(u,t) = 0$ for values of u exceeding the truncated domain.

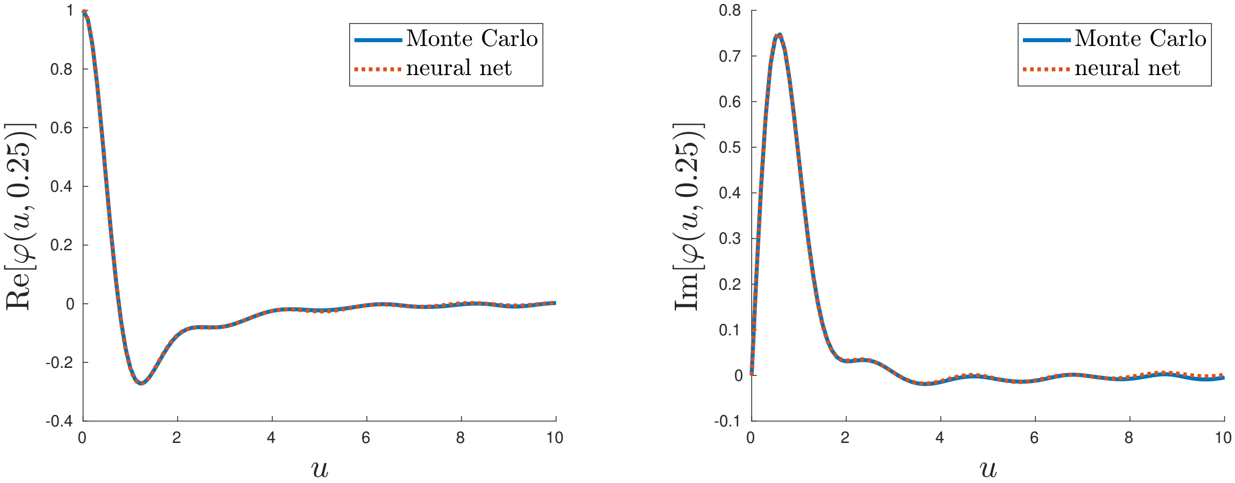

Figure 6: Comparison between the neural network approximation and the chf obtained via Monte Carlo for Section 5.2 at $t = 0.25$. Left: $\mathrm{Re}[\varphi^{MC}(u,0.25)]$ (solid) vs $\mathrm{Re}[\widetilde{\varphi}(u,0.25)]$ (dashed). Right: $\mathrm{Im}[\varphi^{MC}(u,0.25)]$ (solid) vs $\mathrm{Im}[\widetilde{\varphi}(u,0.25)]$ (dashed).

5.3 Duffing oscillator with Gaussian white noise

Let $X(t)$, $t \in [0,1]$, be the displacement of an oscillator with cubic stiffness under the Duffing model subject to Gaussian white noise external forcing. The displacement satisfies the SDE

$$\ddot{X}(t) + 2\zeta\nu\dot{X}(t) + \nu^2(X(t) + \alpha X(t)^3) = W(t) \tag{5.4}$$

where $\zeta \in (0,1)$ is the damping ratio, ν is the initial frequency, α is a constant, and $W(t)$ is Gaussian white noise with zero mean and one-sided spectral density of intensity $g_0 > 0$. In system

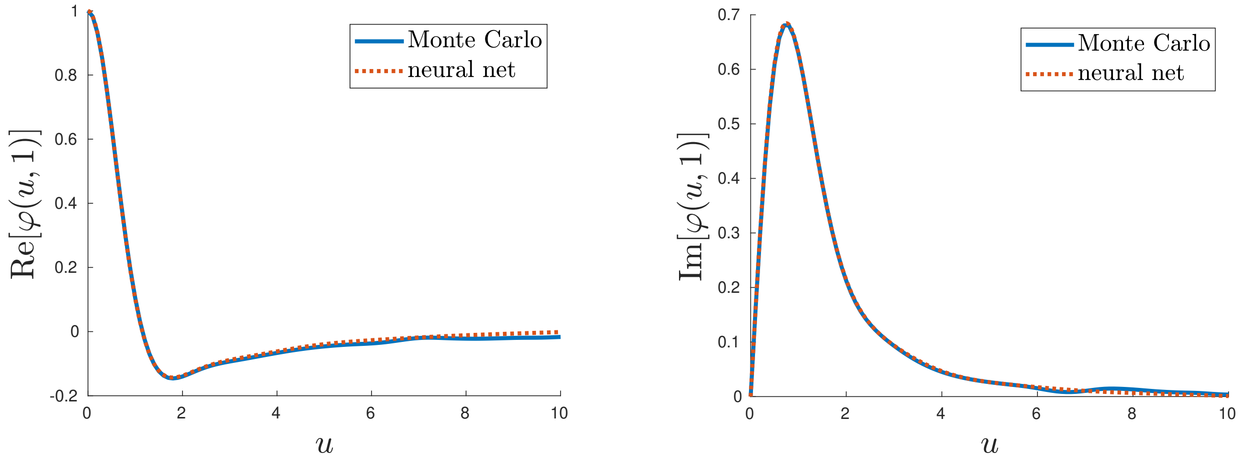

Figure 7: Comparison between the neural network approximation and the chf obtained via Monte Carlo for Section 5.2 at $t = 1$. Left: $\mathrm{Re}[\varphi^{MC}(u,1)]$ (solid) vs $\mathrm{Re}[\widetilde{\varphi}(u,1)]$ (dashed). Right: $\mathrm{Im}[\varphi^{MC}(u,1)]$ (solid) vs $\mathrm{Im}[\widetilde{\varphi}(u,1)]$ (dashed).

form, this can be expressed as

$$d\begin{bmatrix} X_1(t) \\ X_2(t) \end{bmatrix} = \begin{bmatrix} X_2(t) \\ -\nu^2(X_1(t)+\alpha X_1(t)^3) - 2\zeta\nu X_2(t) \end{bmatrix} dt + \begin{bmatrix} 0 \\ \sqrt{\pi g_0} \end{bmatrix} dB(t) \tag{5.5}$$

where $X_1(t) = X(t), X_2(t) = \dot{X}(t)$ and $B(t)$ is Brownian motion [11, p. 480]. The objective is to numerically verify that the pdfs of $X(t)$ resulting from a neural network approximation of the Fokker-Planck equation and that of the PDE for the characteristic function coincide.

In the simulations that follow, the parameters values we selected were $\zeta = 0.25, \nu = 1, \alpha = 1, g_0 = 1$ and $X_1(0) \sim N(0,1), X_2(0) \sim N(0,1)$ with $\rho = \mathrm{Corr}(X_1(0), X_2(0)) = 0.8$. The sections below detail the neural network approximation we pursued under each approach. To examine the accuracy of each approximation, 250000 Monte Carlo samples of $X_1(t), X_2(t)$ are generated by simulating (5.5) via Runge-Kutta scheme with time step size of 0.005.

5.3.1 Fokker-Planck equation

According to (3.14), the Fokker-Planck equation for (5.5) is

$$\frac{\partial f(\boldsymbol{x},t)}{\partial t} = -x_2\frac{\partial f(\boldsymbol{x},t)}{\partial x_1} + 2\zeta\nu f(\boldsymbol{x},t) + (\nu^2(x_1+\alpha x_1^3) + 2\zeta\nu x_2)\frac{\partial f(\boldsymbol{x},t)}{\partial x_2} + \frac{\pi g_0}{2}\frac{\partial^2 f(\boldsymbol{x},t)}{\partial x_2^2}$$

which is transformed to

$$\begin{aligned} \mathcal{M}[v(\boldsymbol{x},t)] = v_t(\boldsymbol{x},t) + x_2 v_{x_1}(\boldsymbol{x},t) + 2\zeta\nu - (\nu^2(x_1+\alpha x_1^3) + 2\zeta\nu x_2)v_{x_2}(\boldsymbol{x},t) \\ + \frac{\pi g_0}{2}((v_{x_2}(\boldsymbol{x},t))^2 - v_{x_2x_2}(\boldsymbol{x},t)) + \frac{c'(t)}{c(t)} = 0 \end{aligned} \tag{5.6}$$

upon applying (4.1) with $c(t) = \int_{\mathbb{R}}\int_{\mathbb{R}} e^{-v(x_1,x_2,t)}\,dx_1\,dx_2$. From the initial conditions specified above, we have $f(\boldsymbol{x},0) = \frac{1}{2\pi\sqrt{1-\rho^2}}\exp\left(-\frac{1}{2(1-\rho^2)}\left[x_1^2+x_2^2-2\rho x_1x_2\right]\right)$ or $v(\boldsymbol{x},0) = \frac{1}{2(1-\rho^2)}\left[x_1^2+x_2^2-2\rho x_1x_2\right]$ for the transformed variable.

We seek a neural network approximation $\widetilde{v}(\boldsymbol{x},t)$ such that $\mathcal{M}[\widetilde{v}(\boldsymbol{x},t)]=0$ over the truncated domain $(x_1,x_2)\in[-4,4]\times[-8,8]$. The architecture of the network consists of an input layer with 3 neurons, an output layer with 1 neuron, and 6 hidden layers with 50 neurons each. The network was trained using a regular grid of $N_{IC}=1025$ points (25 points in $x_1\in[-4,4]$ and 41 points in $x_2\in[-8,8]$) and $N_{Op}=50000$ points formed by taking the tensor product of 50 latin hypercube samples in $t\in[0,1]$ and 1000 latin hypercube samples in $(x_1,x_2)\in[-4,4]\times[-8,8]$. These N_{Op} collocation points were also used to estimate the terms $c'(t)$ and $c(t)$ in the operator $\mathcal{M}[v(\boldsymbol{x},t)]$ (5.6) via Monte Carlo integration.

Figures 8, 9, 10, and 11 summarize the performance of the neural network approximation $\widetilde{f}(\boldsymbol{x},t)$ to the solution of the Fokker-Planck equation which is then compared to the pdf $f^{MC}(\boldsymbol{x},t)$ representing the kernel density estimate from the Monte Carlo samples of $X_1(t),X_2(t)$. The panels of Figure 8 display $\widetilde{v}(\boldsymbol{x},t)$ at $t=0.25,0.75$. Figure 9 compares $f^{MC}(\boldsymbol{x},t)$ (left) with $\widetilde{f}(\boldsymbol{x},t)$ (right) for $t=0.25$; the same comparison is made in Figure 10 but for $t=0.75$. Difference plots which reflect the absolute discrepancy between $f^{MC}(\boldsymbol{x},t)$ and $\widetilde{f}(\boldsymbol{x},t)$ at both times are shown in Figure 11. Finally, we also compute $\frac{\max_{\boldsymbol{x}}|f^{MC}(\boldsymbol{x},t)-\widetilde{f}(\boldsymbol{x},t)|}{\max_{\boldsymbol{x}}|f^{MC}(\boldsymbol{x},t)|}$ which is 0.0621, 0.0770, 0.0601, and 0.0552 for $t=0.25,0.5,0.75,1$. As these plots and calculations reveal, the neural network and Monte Carlo approximations are similar in behavior. Furthermore, denote by $\widetilde{f}_1(x_1,t)$ the estimate of the pdf of $X(t)$ which results by marginalizing $\widetilde{f}(\boldsymbol{x},t)$ through $\widetilde{f}_1(x_1,t)=\int_{[-8,8]}\widetilde{f}(x_1,x_2,t)\,dx_2$. As Figure 12 demonstrates, $\widetilde{f}_1(x_1,t)$ is able to match the histograms of Monte Carlo samples of $X_1(t)$ for $t=0.25$ (left) and $t=0.75$ (right).

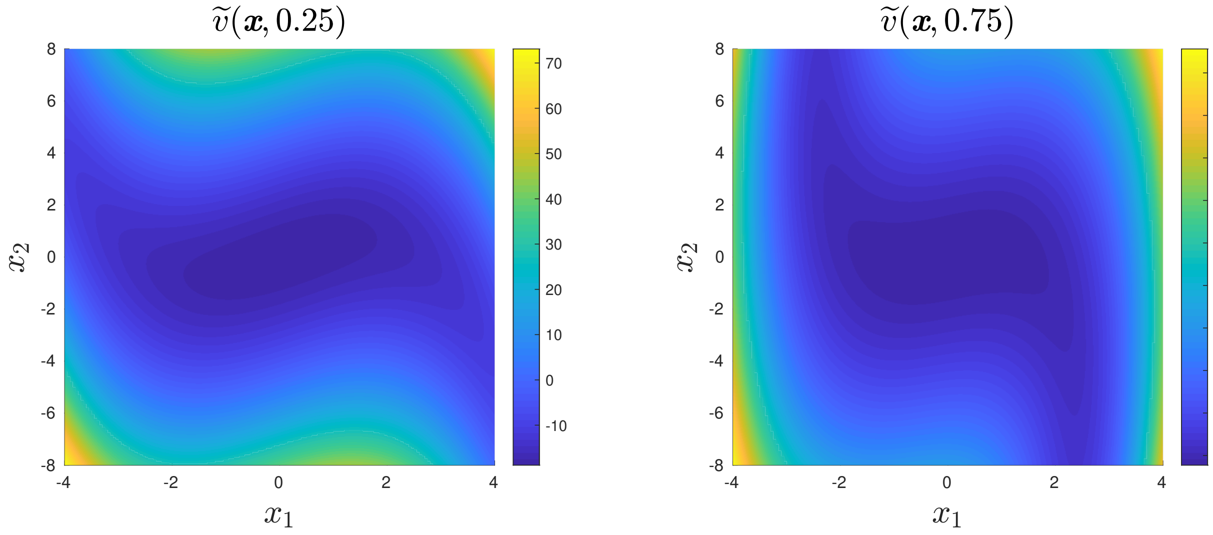

Figure 8: Neural network approximation $\widetilde{v}(\boldsymbol{x},t)$ for Section 5.3.1 at $t=0.25$ (left) and $t=0.75$ (right).

While the above plots suggest that $\widetilde{f}(\boldsymbol{x},t)$ adequately solves the Fokker-Planck equation, the loss value corresponding to $\widetilde{v}(\boldsymbol{x},t)$ is 0.0131580755 which is relatively higher than those from the previous examples. To understand why $\widetilde{f}(\boldsymbol{x},t)$ offers a sufficient approximation despite having a large loss, we construct a binned scatterplot in Figure 13 between $\|\boldsymbol{x}\|$ and $|\mathcal{M}[\widetilde{v}(\boldsymbol{x},t)]|$ using the $N_{Op}=50000$ collocation points we have generated. Figure 13 underscores that the loss value is large because the error in the governing equation is large for collocation points far from the origin, i.e. close to the boundary of $[-4,4]\times[-8,8]$. However, at these collocation points, the magnitude of the actual pdf $f(\boldsymbol{x},t)$ is considerably small due to the boundary conditions of the Fokker-Planck equation.

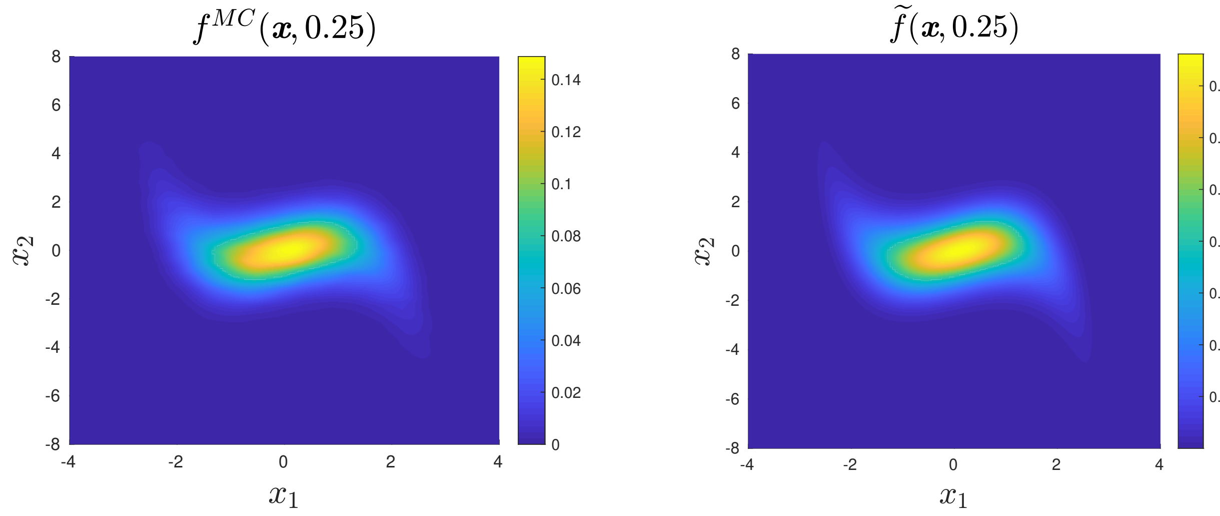

Figure 9: Comparison between the pdf $f^{MC}(\boldsymbol{x}, t)$ obtained via kernel density estimation on Monte Carlo samples (left) and the neural network approximation $\widetilde{f}(\boldsymbol{x}, t)$ (right) for Section 5.3.1 at $t = 0.25$.

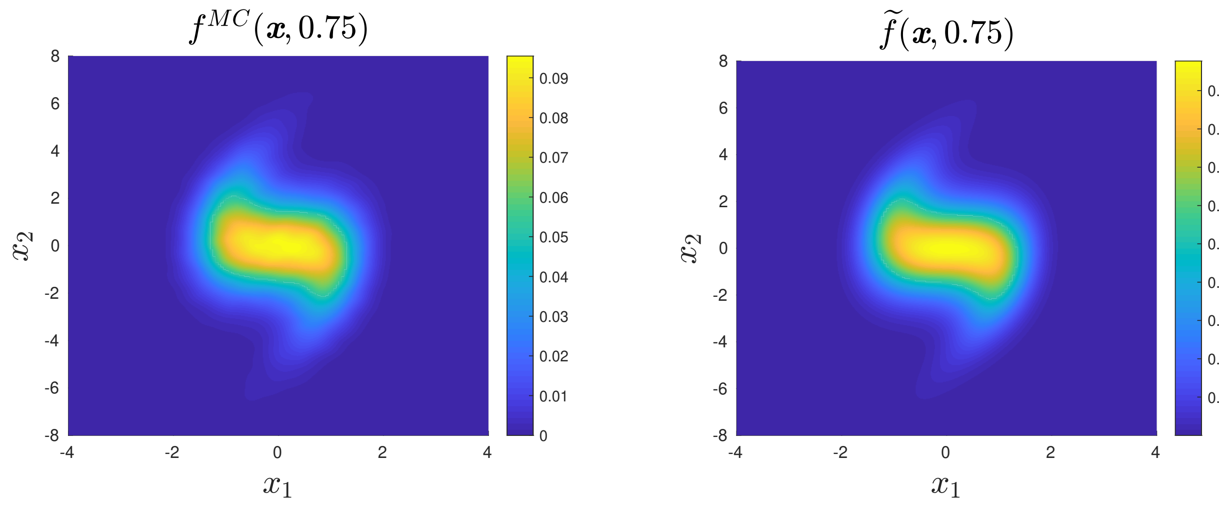

Figure 10: Comparison between the pdf $f^{MC}(\boldsymbol{x}, t)$ obtained via kernel density estimation on Monte Carlo samples (left) and the neural network approximation $\widetilde{f}(\boldsymbol{x}, t)$ (right) for Section 5.3.1 at $t = 0.75$.

Thus, when the transformation (4.1) is applied to normalize the solution $\widetilde{v}(\boldsymbol{x}, t)$, the large error at these collocation points is nullified for the approximation $\widetilde{f}(\boldsymbol{x}, t)$. The same behavior persists for other neural network architectures we have investigated. This presents an example as to why it may be disadvantageous to solve the Fokker-Planck equation using neural networks – it may not be always possible to diagnose why the loss value for $\widetilde{v}(\boldsymbol{x}, t)$ is large. Finally, we noticed in our numerical experiments that for some architectures, $\widetilde{v}(\boldsymbol{x}, t)$ has a tendency of being very negative which renders $\frac{c'(t)}{c(t)}$ and hence $\mathcal{M}[\widetilde{v}(\boldsymbol{x}, t)]$ (5.6) nan. The optimization algorithm for minimizing the loss is unable to proceed in such cases. This scenario is due to lack of a unique solution to (5.6) as discussed in Section 4.2.

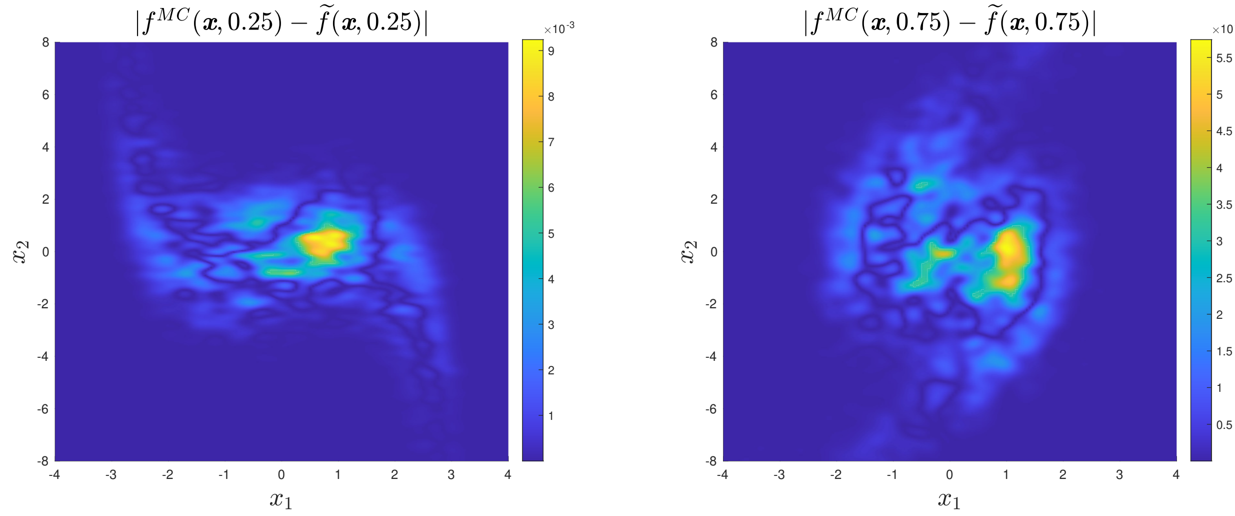

Figure 11: Absolute discrepancy $|f^{MC}(\boldsymbol{x}, t) - \widetilde{f}(\boldsymbol{x}, t)|$ at $t = 0.25$ (left) and $t = 0.75$ (right) for Section 5.3.1.

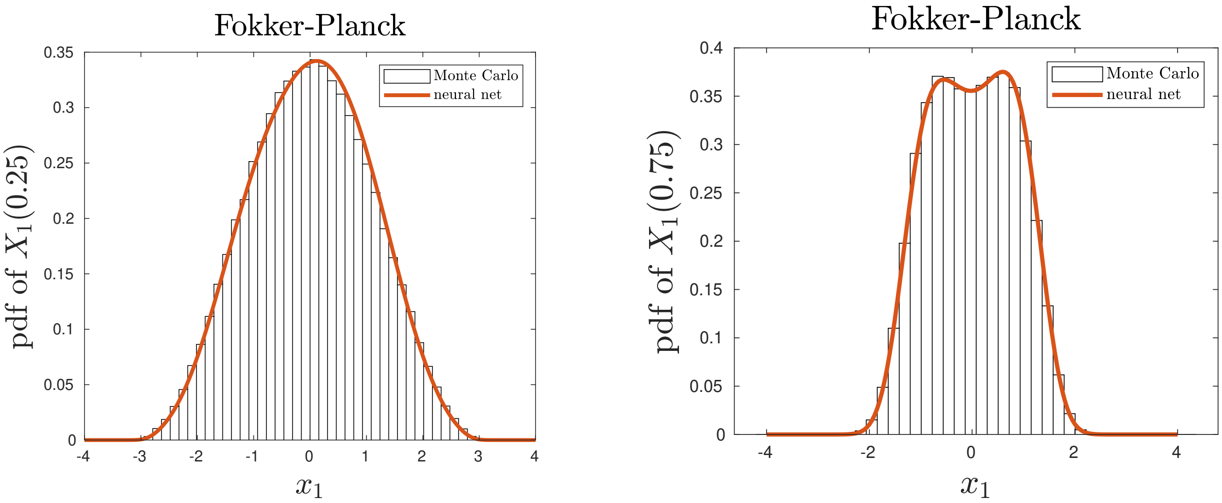

Figure 12: Comparison between the neural network approximation $\widetilde{f}_1(x_1, t)$ to the pdf of $X_1(t)$ and the histogram based on Monte Carlo samples of $X_1(t)$ for Section 5.3.1 at $t = 0.25$ (left) and $t = 0.75$ (right).

5.3.2 Characteristic function PDE

In contrast to the previous section, we estimate the pdf of $X_1(t) = X(t)$ by first solving the PDE of the characteristic function and subsequently applying the Fourier transform. From (3.2) and (3.13), the chf $\varphi(\boldsymbol{u}, t)$ satisfies

$$\mathcal{Q}[\varphi(\boldsymbol{u}, t)] = \frac{\partial \varphi(\boldsymbol{u}, t)}{\partial t} - (u_1 + 2\zeta\nu u_2)\frac{\partial \varphi(\boldsymbol{u}, t)}{\partial u_2} - \nu^2 u_2 \frac{\partial \varphi(\boldsymbol{u}, t)}{\partial u_1} + \nu^2 \alpha u_2 \frac{\partial^3 \varphi(\boldsymbol{u}, t)}{\partial u_1^3} - \frac{\pi g_0}{2} u_2^2 \varphi(\boldsymbol{u}, t) = 0 \tag{5.7}$$

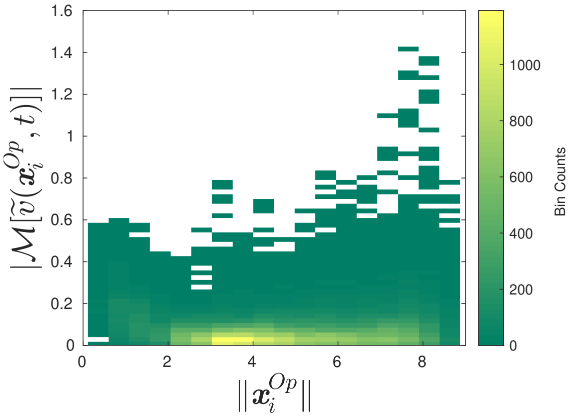

Figure 13: Binned scatterplot of $\|\boldsymbol{x}\|$ vs $|\mathcal{M}[\widetilde{v}(\boldsymbol{x},t)]|$ using N_{Op} collocation points to enforce (5.6) for Section 5.3.1.

with initial condition $\varphi(\boldsymbol{u},0) = \exp\left(-\frac{1}{2}\boldsymbol{u}'\begin{bmatrix}1 & \rho\\ \rho & 1\end{bmatrix}\boldsymbol{u}\right)$. This PDE was solved on the truncated domain $\boldsymbol{u} \in [-6,6]^2$.

The neural network approximation $\widetilde{\varphi}(\boldsymbol{u},t)$ we seek is equipped with an architecture that constitutes an input layer with 3 neurons, an output layer with 1 neuron, and 5 hidden layers with 50 neurons each. The output layer only has 1 neuron because we can leverage on prior probabilistic information on (5.5) to deduce that $\varphi(\boldsymbol{u},t)$ is real-valued. To see this, by using the fact that $B(t)$ and $-B(t)$ identically distributed, it follows that $(X_1(t),X_2(t))$ and $(-X_1(t),-X_2(t))$ are identically distributed since both sets of random vectors satisfy the SDE (5.5). This implies that $f(\boldsymbol{x},t)$ is symmetric with respect to the spatial origin and hence, $\varphi(\boldsymbol{u},t)$ has imaginary part 0. To compute the loss function, the collocation points we utilized were a regular grid of $N_{IC} = 33\times 33$ points in $\boldsymbol{u}\in[-6,6]^2$ as well as latin hypercube samples with $N_0 = 100$ points in $t\in[0,1]$ and $N_{Op} = 100000$ points in $(\boldsymbol{u},t)\in[-6,6]^2\times[0,1]$ which resulted in a loss value of 5.3324147×10^{-5}.

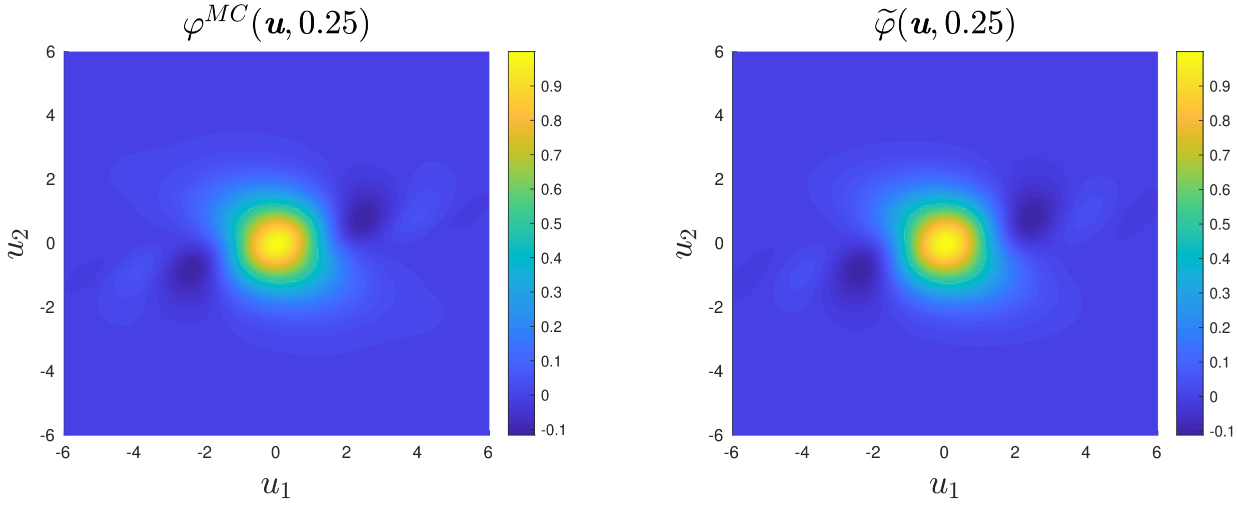

Figure 14: Comparison between the chf $\varphi^{MC}(\boldsymbol{u},t)$ obtained from Monte Carlo samples (left) and the neural network approximation $\widetilde{\varphi}(\boldsymbol{u},t)$ (right) for Section 5.3.2 at $t=0.25$.

Figures 14 and 15 illustrate the neural network approximation $\widetilde{\varphi}(\boldsymbol{u},t)$ at $t=0.25$ and $t=0.75$,

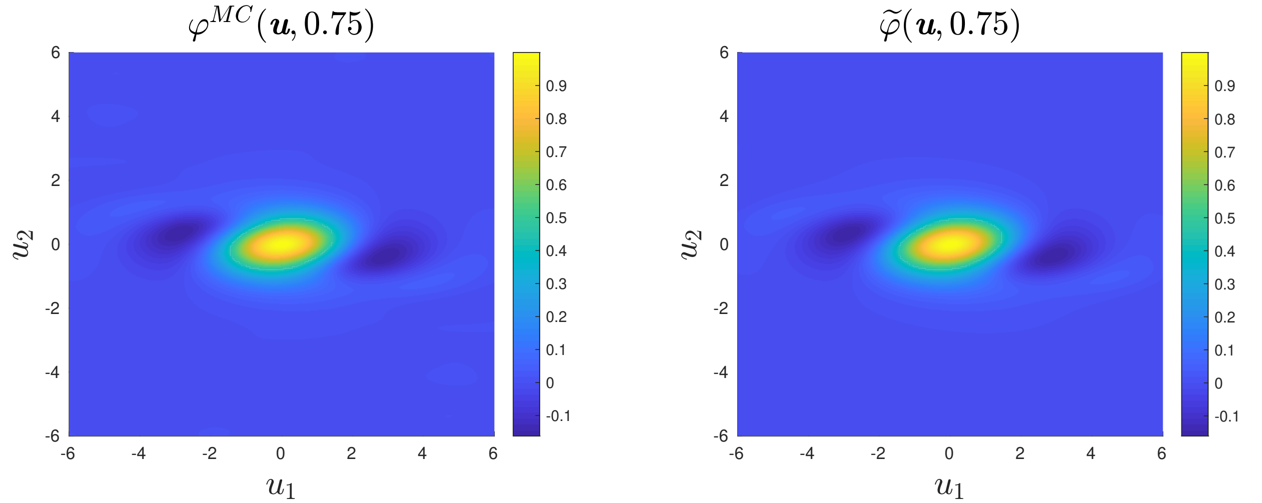

Figure 15: Comparison between the chf $\varphi^{MC}(\boldsymbol{u},t)$ obtained from Monte Carlo samples (left) and the neural network approximation $\widetilde{\varphi}(\boldsymbol{u},t)$ (right) for Section 5.3.2 at $t = 0.75$.

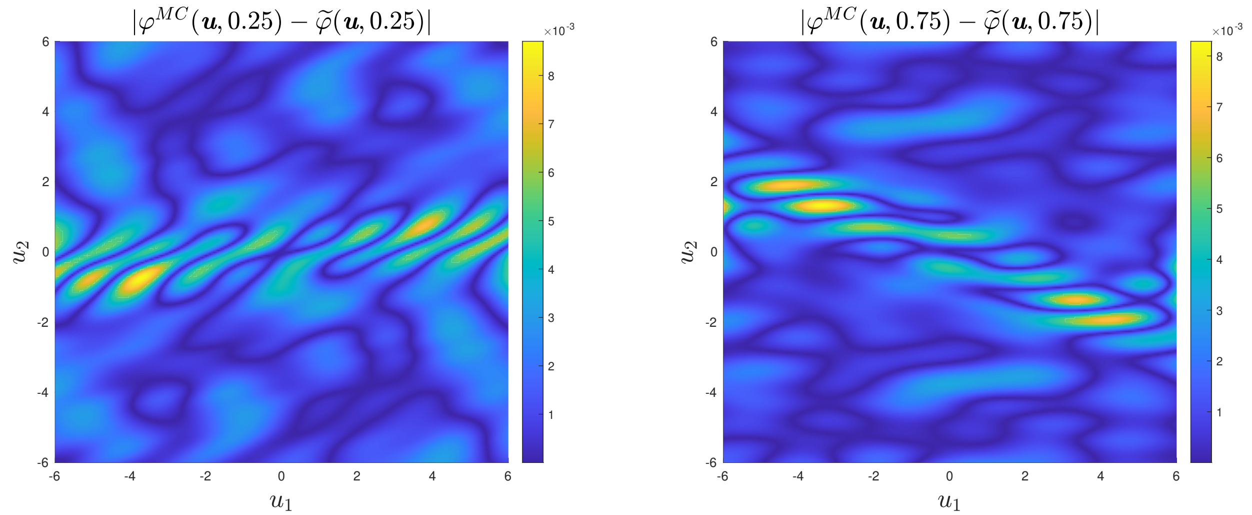

Figure 16: Absolute discrepancy $|\varphi^{MC}(\boldsymbol{u},t) - \widetilde{\varphi}(\boldsymbol{u},t)|$ at $t = 0.25$ (left) and $t = 0.75$ (right) for Section 5.3.2.

respectively. In each figure, $\widetilde{\varphi}(\boldsymbol{u},t)$ (right subplot) is compared to $\varphi^{MC}(\boldsymbol{u},t)$ (left subplot) which is an estimate of the chf of $(X_1(t), X_2(t))$ based on Monte Carlo simulation. Difference plots which reflect the absolute discrepancy between $\varphi^{MC}(\boldsymbol{u},t)$ and $\widetilde{\varphi}(\boldsymbol{u},t)$ at both times are shown in Figure 16. We also compute $\frac{\max_{\boldsymbol{u}} |\varphi^{MC}(\boldsymbol{u},t)-\widetilde{\varphi}(\boldsymbol{u},t)|}{\max_{\boldsymbol{u}} |\varphi^{MC}(\boldsymbol{u},t)|}$ which is 0.0087, 0.0112 , 0.0083, and 0.0151 for $t = 0.25, 0.5, 0.75, 1$. These calculations and the left and right panels of each figure support the observation that $\widetilde{\varphi}(\boldsymbol{u},t)$ adequately represents the target chf $\varphi(\boldsymbol{u},t)$.

To reconcile this approach with the approach based on the Fokker-Planck equation in Section 5.3.1, we apply the Fourier transform to $\widetilde{\varphi}(\boldsymbol{u},t)$ to determine an estimate of the pdf of $X_1(t)$, i.e. $\widetilde{f}_1(x_1,t) = \frac{1}{2\pi}\int_{-6}^{6} e^{-iu_1x_1}\widetilde{\varphi}(u_1,0,t)\,du_1$. Figure 17 presents plots of $\widetilde{f}_1(x_1,t)$ with a histogram of samples of $X_1(t)$ for $t = 0.25$ (left) and $t = 0.75$ (right). The plots in Figures 17 and 12 confirm that solving the PDE for the chf to obtain the pdf of $X(t)$ offers an alternative approach that is consistent with solving the Fokker-Planck equation as we expected.

In summary, Section 5.3 contrasted two approaches to construct a neural network representation for the pdf. The disadvantage of the approach elaborated in Section 5.3.1 is that the loss value for $\widetilde{v}(\boldsymbol{x},t)$ may not be indicative of the accuracy of $\widetilde{f}(\boldsymbol{x},t)$. Although the approach in Section 5.3.2 does not possess such challenges, automatic differentiation would have to be invoked for partial derivatives of order greater than 2 for the terms appearing in the chf PDE. In our experience, this meant a slower calculation of the loss function for the neural network during the training process.

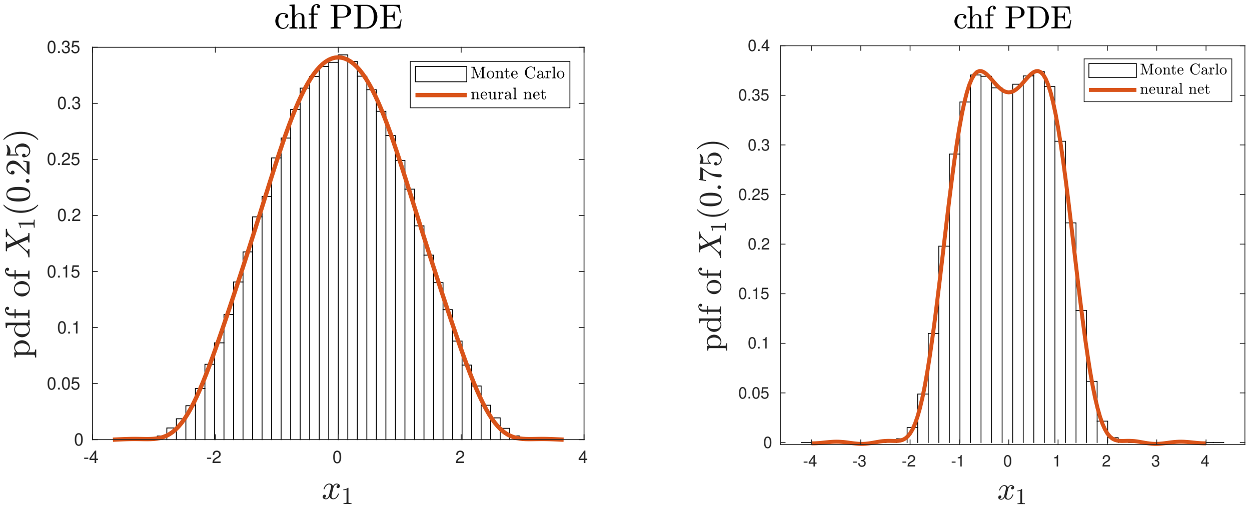

Figure 17: Comparison between the neural network approximation $\widetilde{f}_1(x_1,t)$ to the pdf of $X_1(t)$ and the histogram based on Monte Carlo samples of $X_1(t)$ for Section 5.3.2 at $t=0.25$ (left) and $t=0.75$ (right).

5.4 Duffing oscillator with Poisson white noise

We revisit the Duffing oscillator introduced in Section 5.3 where instead, $W(t)$ in (5.4) is Poisson white noise [11], i.e. $W(t)$ is the formal derivative $\frac{dC^\lambda(t)}{dt}$ where $C^\lambda(t)=\sum_{k=1}^{N(t)} Y_k$ is a compound Poisson process described by a Poisson process $N(t)$ with intensity λ and random variables Y_k which are independent copies of Y. For distinction, denote by $(X_1^\lambda(t),X_2^\lambda(t))$ the state variables of (5.4) if $W(t)$ is Poisson white noise while $(X_1(t),X_2(t))$ refers to the state due to Gaussian white noise as in Section 5.3. It will be numerically confirmed using neural network approximations for the chf of $(X_1^\lambda(t),X_2^\lambda(t))$ that as $\lambda\to\infty$, the distribution of $(X_1^\lambda(t),X_2^\lambda(t))$ approximates that of $(X_1(t),X_2(t))$.

We utilize the same parameters and initial condition as in Section 5.3 and set $Y\sim N(0,\sigma^2)$. The values for λ,σ^2 are chosen such that $\pi g_0=\lambda E[Y^2]$ to ensure that the second moment properties of the random forcing in Section 5.3, $\sqrt{\pi g_0}B(t)$, and in this section, $C^\lambda(t)$, are identical. It was demonstrated analytically and numerically in [13] that under some conditions on Y_k, $(X_1^\lambda(t),X_2^\lambda(t))\to(X_1(t),X_2(t))$ in probability as $\lambda\to\infty$ if the system (5.5) is linear with additive noise. As (5.5) is nonlinear, the following heuristic can be used to illustrate that $(X_1^\lambda(t),X_2^\lambda(t))$ still converges in probability to $(X_1(t),X_2(t))$. Consider the system

$$\begin{aligned} d\boldsymbol{X}(t) &= \boldsymbol{a}(\boldsymbol{X}(t))\,dt+\boldsymbol{b}\sqrt{\pi g_0}\,dB(t), \quad \text{and} \\ d\boldsymbol{X}^\lambda(t) &= \boldsymbol{a}(\boldsymbol{X}^\lambda(t))\,dt+\boldsymbol{b}\,dC^\lambda(t) \end{aligned}$$

where $\boldsymbol{X}(t) = (X_1(t), X_2(t))$, $\boldsymbol{X}^\lambda(t) = (X_1^\lambda(t), X_2^\lambda(t))$, $\boldsymbol{a}(x_1, x_2) = \begin{bmatrix} x_2 \\ -\nu^2(x_1 + \alpha x_1^3) - 2\zeta\nu x_2 \end{bmatrix}$ and $\boldsymbol{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. Since for every $t \geq 0$, $C^\lambda(t)$ converges in probability to $\sqrt{\pi g_0} B(t)$ as $\lambda \to \infty$ [21], so do the increments $dC^\lambda(t)$ and $\sqrt{\pi g_0}\, dB(t)$. The convergence of $\boldsymbol{X}^\lambda(t)$ to $\boldsymbol{X}(t)$ follows because they depend continuously on $dC^\lambda(t)$ and $\sqrt{\pi g_0}\, dB(t)$, respectively. We now verify this qualitatively by visually inspecting sample paths of the state variables in the following figures. Figure 18 corresponds to $(X_1(t), X_2(t))$ in Section 5.3 while Figures 19 and 20 correspond to $(X_1^\lambda(t), X_2^\lambda(t))$ for $E[Y^2] = 0.01$ and $E[Y^2] = 3$, respectively, with $\lambda = \frac{\pi g_0}{E[Y^2]}$. The figures certify that for larger values of λ, the sample paths of $(X_1(t), X_2(t))$ and $(X_1^\lambda(t), X_2^\lambda(t))$ are almost indistinguishable, especially in the second state variable.

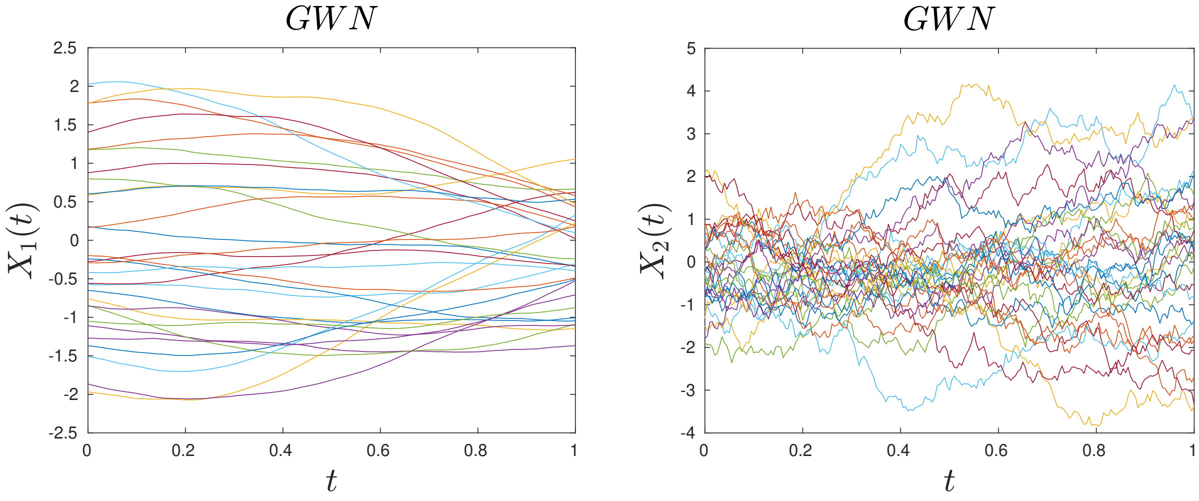

Figure 18: Sample paths of $X_1(t)$ (left) and $X_2(t)$ (right) for the Duffing oscillator subject to Gaussian white noise in Section 5.3.

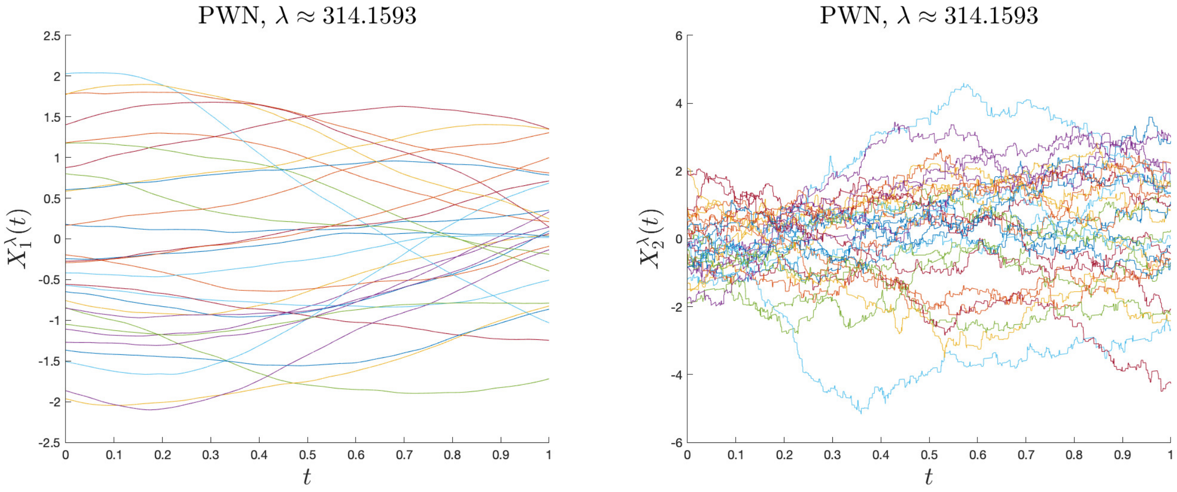

Figure 19: Sample paths of $X_1^\lambda(t)$ (left) and $X_2^\lambda(t)$ (right) for the Duffing oscillator subject to Poisson white noise in Section 5.4 with $E[Y^2] = 0.01$.

Our objective is then to reproduce the same conclusion by investigating the characteristic function

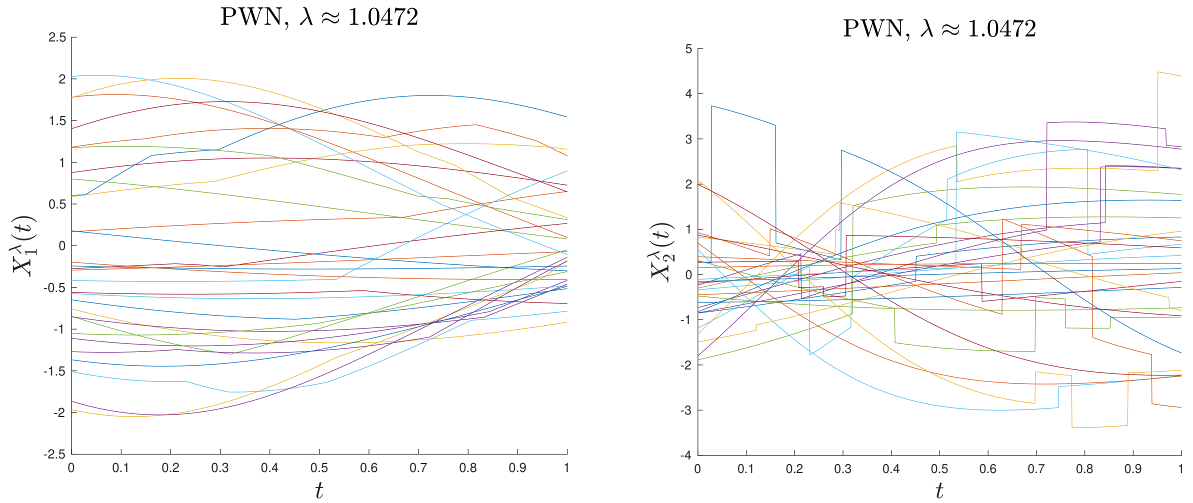

Figure 20: Sample paths of $X_1^\lambda(t)$ (left) and $X_2^\lambda(t)$ (right) for the Duffing oscillator subject to Poisson white noise in Section 5.4 with $E[Y^2] = 3$.

of $(X_1^\lambda(t), X_2^\lambda(t))$. From (3.13), $\varphi(\boldsymbol{u}, t)$ satisfies the PDE

$$\mathcal{Q}[\varphi(\boldsymbol{u},t)] = \frac{\partial \varphi(\boldsymbol{u},t)}{\partial t} - (u_1 + 2\zeta\nu u_2)\frac{\partial \varphi(\boldsymbol{u},t)}{\partial u_2} - \nu^2 u_2 \frac{\partial \varphi(\boldsymbol{u},t)}{\partial u_1} + \nu^2 \alpha u_2 \frac{\partial^3 \varphi(\boldsymbol{u},t)}{\partial u_1^3} - \lambda \varphi(\boldsymbol{u},t)\,[\phi(u_2) - 1] = 0 \quad (5.8)$$

where $\phi(\cdot)$ is the chf of Y. Similar calculations as above highlight that $\varphi(\boldsymbol{u}, t)$ is real-valued since Y and $-Y$ and consequently, $C^\lambda(t)$ and $-C^\lambda(t)$ have the same distribution.

We sought a neural network approximation $\widetilde{\varphi}(\boldsymbol{u}, t)$ on the truncated domain $\boldsymbol{u} \in [-6, 6]^2$ using an architecture comprised of an input layer with 3 neurons, an output layer with 1 neuron, and 5 hidden layers with 50 neurons each. We generated exactly the same set of collocation points as in Section 5.3.2 to evaluate the loss function. Figures 21 and 22 depict $\widetilde{\varphi}(\boldsymbol{u}, t)$ for the case in which $E[Y^2] = 0.01$ ($\lambda \approx 314.1593$) and $E[Y^2] = 3$ ($\lambda \approx 1.0472$), respectively, at times $t = 0.25, 0.75$. The loss value for the trained neural network associated with former scenario was 3.274475×10^{-5} while that of the latter was 1.2586042×10^{-5}. It is clear from scrutinizing the plots in Figure 21 and that of Figures 14 and 15 that the chf of $(X_1^\lambda(t), X_2^\lambda(t))$ for large λ appears identical to that of $(X_1(t), X_2(t))$ subject to Gaussian white noise. The maximum absolute discrepancy between the neural network approximations in Figures 14 and 15 compared to the neural network approximation in Figure 21 is approximately 0.0068 for $t = 0.25$ and 0.0065 for $t = 0.75$. In contrast, the plots in Figure 22 differ from the previously mentioned figures which was expected owing to the asymptotic results discussed above. The maximum absolute discrepancy between the neural network approximations in Figures 14 and 15 compared to the neural network approximation in Figure 22 is approximately 0.2677 for $t = 0.25$ and 0.2255 for $t = 0.75$.

5.5 Characteristic function for a 3-dimensional state vector

Suppose that $X(t)$, $t \in [0, 1]$, represents the displacement of a damped harmonic oscillator in which the external forcing is a non-Gaussian process with continuous samples. Let $X(t)$ satisfy the SDE

$$\begin{aligned} &\ddot{X}(t) + \beta \dot{X}(t) + \nu^2 X(t) = S(t)^3, \\ &dS(t) = -\alpha S(t)\,dt + \sigma\sqrt{2\alpha}\,dB(t) \end{aligned} \quad (5.9)$$

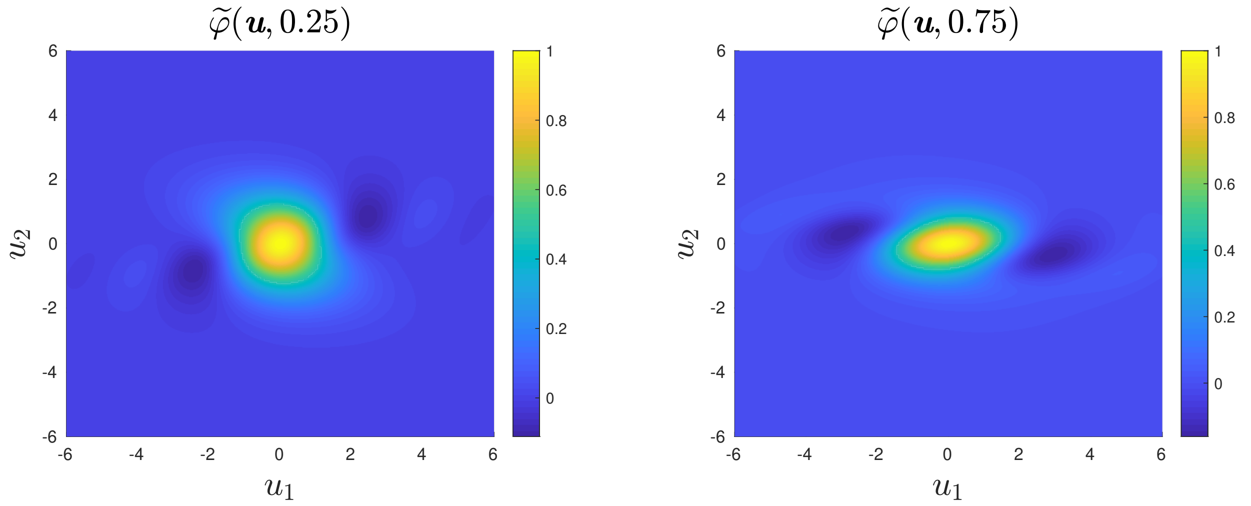

Figure 21: Plots of the neural network approximation $\widetilde{\varphi}(\boldsymbol{u}, t)$ for Section 5.4 with $E[Y_k^2] = 0.01$ at $t = 0.25$ (left) and $t = 0.75$ (right).

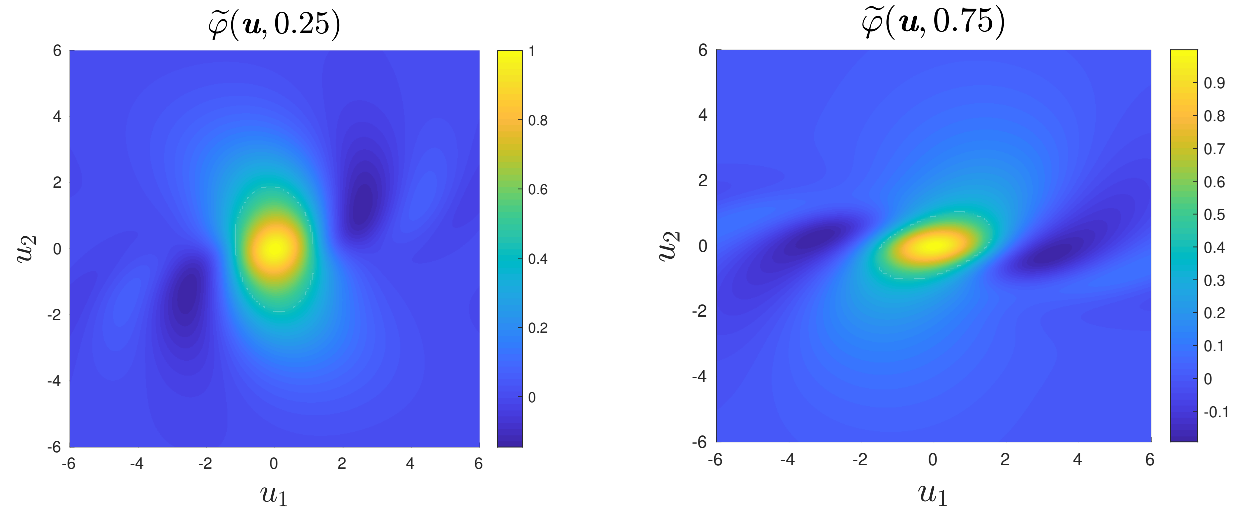

Figure 22: Plots of the neural network approximation $\widetilde{\varphi}(\boldsymbol{u}, t)$ for Section 5.4 with $E[Y_k^2] = 3$ at $t = 0.25$ (left) and $t = 0.75$ (right).

or in system form,

$$d \begin{bmatrix} X_1(t) \\ X_2(t) \\ X_3(t) \end{bmatrix} = \begin{bmatrix} X_2(t) \\ -(\nu^2 + \beta)X_1(t) + S(t)^3 \\ -\alpha S(t) \end{bmatrix} dt + \begin{bmatrix} 0 \\ 0 \\ \sigma\sqrt{2\alpha} \end{bmatrix} dB(t) \tag{5.10}$$

where $B(t)$ is the Brownian motion. It will be demonstrated that even when the frequency domain is 3-dimensional, the neural network approximation to the chf of $\boldsymbol{X}(t) = [X_1(t), X_2(t), X_3(t)]^T$ can adequately match the Monte Carlo solution.

For our simulation, we consider $\boldsymbol{X}(0) \sim N(\boldsymbol{0}_{3\times 1}, \boldsymbol{I}_{3\times 3})$ with $\boldsymbol{I}$ denoting the 3×3 identity matrix so that $\varphi(\boldsymbol{u}, 0) = \exp\left(-\frac{1}{2}\boldsymbol{u}'\boldsymbol{u}\right), \boldsymbol{u} = (u_1, u_2, u_3)$. The parameters we set are $\beta = 0.5, \nu = 3, \alpha = 0.12$. The target chf $\varphi^{MC}(\boldsymbol{u}, t)$ was estimated through 200000 Monte Carlo samples of $\boldsymbol{X}(t)$ produced using forward Euler with time step 0.005.

According to (3.2), the PDE satisfied by $\varphi(\boldsymbol{u},t)$ is

$$\mathcal{Q}[\varphi(\boldsymbol{u},t)] = \frac{\partial\varphi(\boldsymbol{u},t)}{\partial t} - (u_1 - \beta u_2)\frac{\partial\varphi(\boldsymbol{u},t)}{\partial u_2} + \nu^2 u_2\frac{\partial\varphi(\boldsymbol{u},t)}{\partial u_1} + \alpha u_3\frac{\partial\varphi(\boldsymbol{u},t)}{\partial u_3} \\ + u_2\frac{\partial^3\varphi(\boldsymbol{u},t)}{\partial u_3^3} + \sigma^2\alpha u_3^2\varphi(\boldsymbol{u},t) = 0 \tag{5.11}$$

whose solution we approximate on the truncated domain $(u_1, u_2, u_3) \in [-5.5, 5.5]^2 \times [-3, 3]$. The architecture we employed included an input layer with 4 neurons, an output layer with 1 neuron, and 5 hidden layers with 50 neurons each. Following similar arguments above, the chf of $\boldsymbol{X}(t)$ is real-valued since both $\boldsymbol{X}(t)$ and $-\boldsymbol{X}(t)$ satisfy the dynamics (5.10). To train the neural network, latin hypercube samples were simulated for the collocation points which constituted $N_{IC} = 2000$ points in the truncated domain of $\boldsymbol{u}$, $N_0 = 100$ points in $t \in [0, 1]$, and $N_{Op} = 100000$ points in $(\boldsymbol{u}, t) \in [-5.5, 5.5]^2 \times [-3, 3] \times [0, 1]$. The trained network yielded a loss value of 7.966772×10^{-6}.

We assess the performance of $\widetilde{\varphi}(\boldsymbol{u},t)$ with respect to the target Monte Carlo solution in Figures 23, 24, and 25. Each figure displays $\varphi^{MC}(\boldsymbol{u},t)$ (left panel) and $\widetilde{\varphi}(\boldsymbol{u},t)$ (right panel) evaluated at three pairs of values of (u_3, t). The plots suggest that $\widetilde{\varphi}(\boldsymbol{u},t)$ can sufficiently capture the target chf. The maximum absolute discrepancy $\max_{u_1,u_2} |\varphi^{MC} - \widetilde{\varphi}|$ for each figure is given by 0.0152, 0.0149, and 0.0160, respectively. Note the approximation quality despite the fact that the frequency domain in these figures is far from the origin where the characteristic function attains its maximum value of 1. Finally, we query $\widetilde{\varphi}(\boldsymbol{u},t)$ at $u_1 = u_2 = 0$ and various values of t to acertain that it can recover the analytical expression $\varphi(0, 0, u_3, t) = e^{-\frac{1}{2}u_3^2}$ which is due to the fact that $S(t) \sim N(0, 1)$. In Figure 26, we compile plots of $\varphi(0, 0, u_3, t)$ and $\widetilde{\varphi}(0, 0, u_3, t)$ which are almost indistinguishable. This further supports the approximation quality of $\widetilde{\varphi}(\boldsymbol{u},t)$.

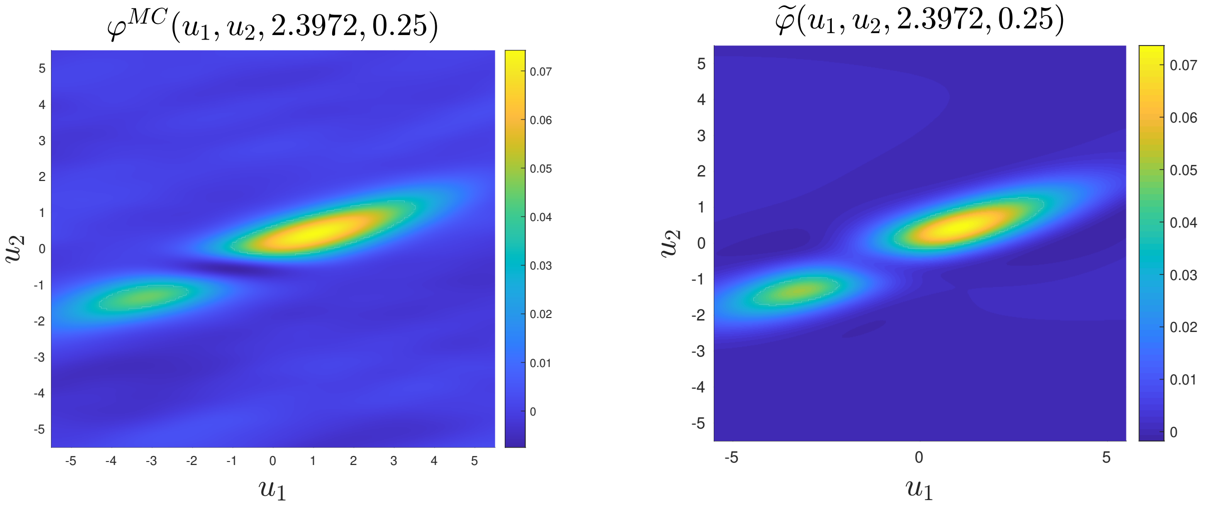

Figure 23: Comparison between the chf $\varphi^{MC}(\boldsymbol{u},t)$ obtained from Monte Carlo samples (left) and the neural network approximation $\widetilde{\varphi}(\boldsymbol{u},t)$ (right) for Section 5.5 at $u_3 = 2.3972, t = 0.25$.

6 Conclusion

This work is concerned with estimating the pdf of the state vector whose dynamics are described by a diffusion process. This is traditionally accomplished by solving the Fokker-Planck equation

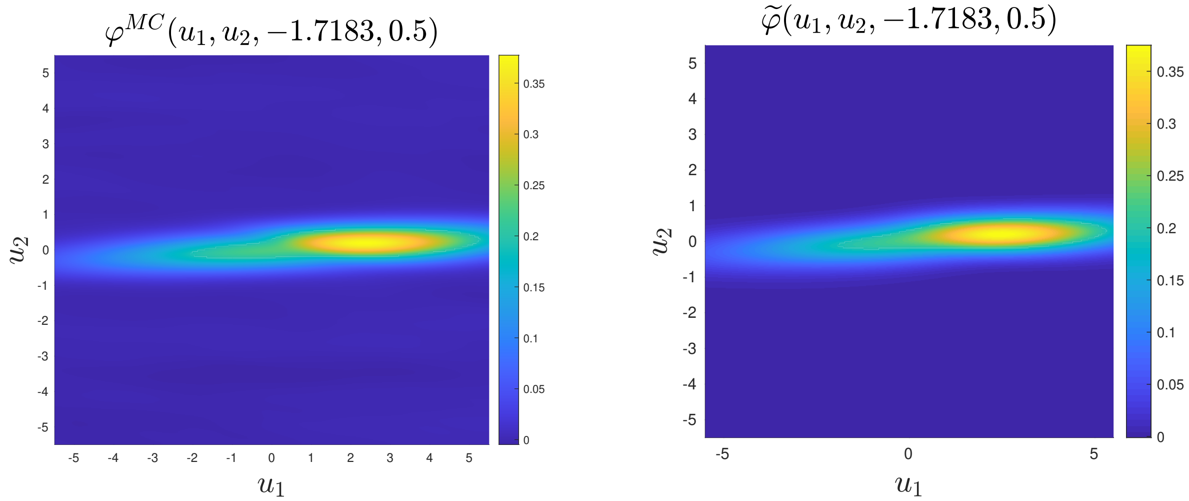

Figure 24: Comparison between the chf $\varphi^{MC}(\boldsymbol{u}, t)$ obtained from Monte Carlo samples (left) and the neural network approximation $\widetilde{\varphi}(\boldsymbol{u}, t)$ (right) for Section 5.5 at $u_3 = -1.7183, t = 0.5$.

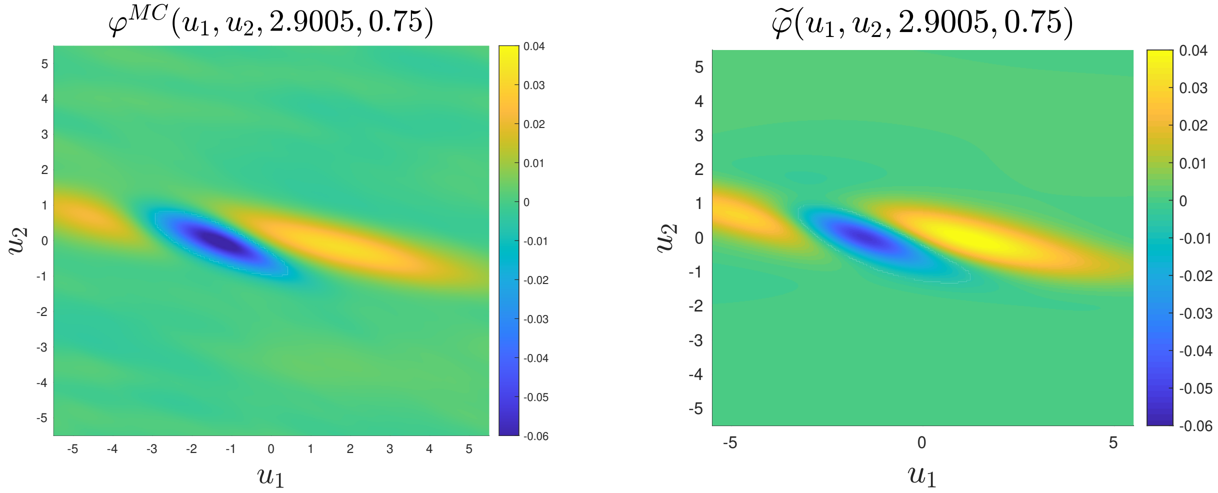

Figure 25: Comparison between the chf $\varphi^{MC}(\boldsymbol{u}, t)$ obtained from Monte Carlo samples (left) and the neural network approximation $\widetilde{\varphi}(\boldsymbol{u}, t)$ (right) for Section 5.5 at $u_3 = 2.9005, t = 0.75$.

which describes the time evolution of the pdf. Since solving this PDE through standard numerical methods may be infeasible for spatial dimensions larger than 3, the use of physics-informed neural networks to approximate the solution to this PDE was investigated. In addition, we sought a neural network solution to the differential equation for the characteristic of the state from which the pdf of the state can be deduced. By incorporating probabilistic constraints on the pdf and chf, we outlined strategies for designing the loss function to train a neural network representation for these quantities.

Through a wide variety of applications, it was demonstrated that the neural network approximation to the pdf or chf can match the analytical or Monte Carlo solution even for integro-differential equations and systems of PDEs. They also highlighted the advantages and disadvantages of solving one type of differential equation over another. For example, while the differential equation for the chf is complex-valued with high-order derivatives, there are instances for which a PDE for the pdf is

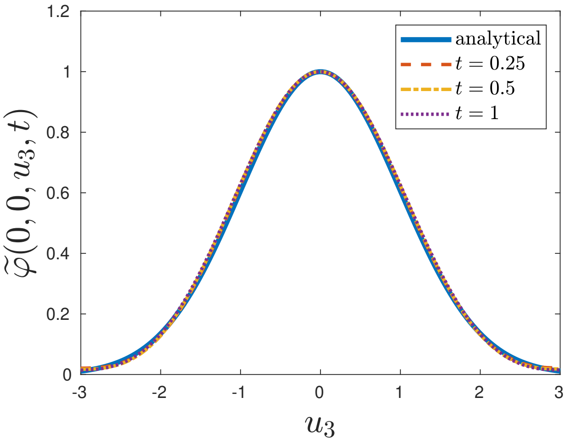

Figure 26: Comparison of the analytical expression for $\varphi(0, 0, u_3, t)$ (solid) and the neural network approximation $\widetilde{\varphi}(0, 0, u_3, t)$ at various times (different styles of dashed lines) for Section 5.5.

unavailable. Imposing the normalization constraint of the Fokker-Planck equation may also present numerical challenges in the neural network solution. Nevertheless, the applications underscore that solving either type of differential equation with neural networks offers consistent information on the pdf of the state and that the neural network representation of the pdf or the chf may be useful for a wide variety of applications. The ideas developed here can be readily extended to high-dimensional problems wherein training the neural network solution to the differential equation has to be performed in a gridless manner.

Acknowledgements

We are grateful to Prof. Maziar Raissi and Ali Al-Aradi for sharing their code from which we based our python scripts for the neural network simulations. The work reported in this paper has been partially supported by the National Science Foundation under grant CMMI-1639669. This support is gratefully acknowledged.

Data availability statement

The data that support the findings of this study are available from the corresponding author upon reasonable request. Please send an e-mail to wtu4@cornell.edu.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] A. Al-Aradi, A. Correia, D. de Frietas Naiff, G. Jardim, and Y. Saporito. Applications of the deep galerkin method to solving partial integro-differential and hamilton-jacobi-bellman equations, 2020. arXiv:1912.01455v2.

[3] A. Al-Aradi, A. Correia, D. Naiff, G. Jardim, and Y. Saporito. Solving nonlinear and high-dimensional partial differential equations via deep learning, 2018. arXiv:1811.08782.

[4] J. Chen and Z. Rui. Dimension-reduced FPK equation for additive white-noise excited nonlinear structures. *Probabilistic Engineering Mechanics*, 53:1–13, June 2018.

[5] N. Chen and A. J. Majda. Efficient statistically accurate algorithms for the fokker–planck equation in large dimensions. *Journal of Computational Physics*, 354:242–268, Feb. 2018.

[6] N. Chen, A. J. Majda, and X. T. Tong. Rigorous analysis for efficient statistically accurate algorithms for solving fokker–planck equations in large dimensions. *SIAM/ASA Journal on Uncertainty Quantification*, 6(3):1198–1223, Jan. 2018.

[7] H. Cho, D. Venturi, and G. Karniadakis. Numerical methods for high-dimensional probability density function equations. *Journal of Computational Physics*, 305:817–837, Jan. 2016.

[8] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, Dec. 1989.

[9] A. Dektor and D. Venturi. Dynamically orthogonal tensor methods for high-dimensional nonlinear pdes, 2019. arXiv:1907.05924.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[11] M. Grigoriu. *Stochastic calculus. Applications in science and engineering.* Birkhäuser, Boston, 2002.

[12] M. Grigoriu. Characteristic function equations for the state of dynamic systems with gaussian, poisson, and lévy white noise. *Probabilistic Engineering Mechanics*, 19(4):449–461, 2004.

[13] M. Grigoriu. Reliability of linear systems under poisson white noise. *Probabilistic Engineering Mechanics*, 24(3):397–406, July 2009.

[14] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, Jan. 1989.

[15] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations.* Springer Berlin Heidelberg, 1992.

[16] A. Masud and L. A. Bergman. Solution of the four dimensional fokker–planck equation: still a challenge. In G. Augusti, G. Schuëller, and M. Ciampoli, editors, *ICOSSAR*, Millpress, Rotterdam, 2005.

[17] L. Pichler, A. Masud, and L. A. Bergman. Numerical solution of the fokker–planck equation by finite difference and finite element methods—a comparative study. In *Computational Methods in Stochastic Dynamics*, pages 69–85. Springer Netherlands, 2013.

[18] M. Raissi, P. Perdikaris, and G. 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, Feb. 2019.

[19] H. Risken. *The Fokker-Planck Equation.* Springer Berlin Heidelberg, 1989.

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

[21] A. Skorohod. *Studies in the theory of random processes.* Dover Publications, Inc. New York, 1982.

[22] M. Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[23] S. Wojtkiewicz and L. Bergman. Numerical solution of high dimensional fokker-planck equations. In *8th ASCE Specialty Conference on Probabilistic Mechanics and Structural Reliability*, 2000.

[24] Y. Xu, H. Zhang, Y. Li, K. Zhou, Q. Liu, and J. Kurths. Solving fokker-planck equation using deep learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1):013133, Jan. 2020.