A transfer learning method to solve Fokker–Planck equation based on the equivalent linearization

Gege Wang, Xiaolong Wang, Qi Liu, Jürgen Kurths, Yong Xu

Originally published as
Wang, G., Wang, X., Liu, Q., Kurths, J., Xu, Y. (2025): A transfer learning method to solve Fokker–Planck equation based on the equivalent linearization . - Chaos, 35, 8, 083119.
https://doi.org/10.1063/5.0260624

RESEARCH ARTICLE | AUGUST 08 2025

A transfer learning method to solve Fokker–Planck equation based on the equivalent linearization

Gege Wang ; Xiaolong Wang ; Qi Liu ; Jürgen Kurths ; Yong Xu

Chaos 35, 083119 (2025)
https://doi.org/10.1063/5.0260624

A transfer learning method to solve Fokker–Planck equation based on the equivalent linearization

Cite as: Chaos 35, 083119 (2025); doi: 10.1063/5.0260624
Submitted: 25 January 2025 · Accepted: 12 May 2025 ·
Published Online: 8 August 2025

Gege Wang,[1] Xiaolong Wang,[1,2] Qi Liu,[3] Jürgen Kurths,[4,5] and Yong Xu[1,6,a)]

AFFILIATIONS

[1]School of Mathematics and Statistics, Northwestern Polytechnical University, Xi'an 710072, China
[2]School of Mathematics and Statistics, Shaanxi Normal University, Xi'an 710119, China
[3]Department of Systems and Control Engineering, Institute of Science Tokyo, Tokyo 152-8552, Japan
[4]Potsdam Institute for Climate Impact Research, Potsdam 14412, Germany
[5]Department of Physics, Humboldt University Berlin, Berlin 12489, Germany
[6]MOE Key Laboratory for Complexity Science in Aerospace, Northwestern Polytechnical University, Xi'an 710072, China

[a)]**Author to whom correspondence should be addressed:** hsux3@nwpu.edu.cn

ABSTRACT

Efficient methods for solving the Fokker–Planck (FP) equation are crucial for studying stochastic systems. This paper proposes a transfer learning method to solve the FP equation, enabling the training process to proceed without starting from the beginning. The equivalent linearization is first applied to unify a class of stochastic differential equations into a single simplified form. Subsequently, a pre-trained neural network framework, inspired by transfer learning, is designed based on the FP equation of the simple system. By leveraging the pre-trained neural network, the solving process is accelerated by starting from a more advanced state. Finally, numerical experiments are conducted to verify the proposed approach, including one- and two-dimensional stochastic systems as well as a system driven by both Gaussian and Lévy noise. Results show that the contours of the FP equations can be learned by the network very expeditiously, greatly reducing training time while maintaining accuracy. The proposed method not only improves computational efficiency but also demonstrates strong generalization capabilities.

The Fokker–Planck (FP) equation describes the probability density function of the system state evolving over time. It elucidates the influence of random disturbances on system behaviors, playing a vital role in nonlinear stochastic dynamics. As neural network technology continues to progress, the application of these models to solve the FP equation has emerged as a research hotspot. However, existing methods require a significant amount of time for solving the FP equation, even when there are only minor changes in the system. This leads to a waste of computational resources. To address the challenges of long training times and effectively solving a class of FP equations, this paper proposes a transfer learning method to solve the FP equation by the equivalent linearization. Rather than training from scratch, the method employs the equivalent linearization to simplify the complex stochastic system, providing a well-suited initial value for the solving process. Numerical examples are given to demonstrate the effectiveness of the method. The results indicate that the proposed algorithm can acceleratedly solve the FP equation.

I. INTRODUCTION

Understanding of the natural world and social phenomena has evolved from simple causal relationships to complex dynamic systems. In these systems, nonlinearity and stochasticity have become the two main elements describing their intrinsic laws. Nonlinear stochastic dynamics is the study of how the state of a system evolves in a stochastic manner under nonlinear effects. The Fokker–Planck (FP) equation is a key topic of study in stochastic dynamics. It describes the evolution of the probability density of a stochastic system over time and transforms a stochastic problem into a deterministic one. It is the basis for our analyses of reliability, stability, etc.[1–4] and provides us with the possibility of revealing the mysteries of stochastic systems. The FP equation provides a method for analyzing and predicting the dynamic response of a system under stochastic excitation, which holds significant importance for the study of stochastic systems.[5–9] In finance, the FP equation is used to model the stochastic evolution of asset prices, providing a basis for option pricing and risk management.[6] In aerospace, the FP equation is used

to analyze the behaviors of aircraft in stochastic environments, especially the impacts of aerodynamic noise on flight performance.[1,9,10] The existing methods for solving the FP equation are computationally expensive and time-consuming. Therefore, developing efficient and accurate methods for solving the FP equation is the focus of our research.

In recent years, many new research results have emerged for solving the FP equation. Typically, the FP equation can be derived as analytical solutions only in special cases, such as one-dimensional or linear systems, and the related results are discussed in detail in the monographs.[11–14] Traditional numerical methods, such as the finite element,[15] finite difference,[16] path integral,[17] and Monte Carlo (MC) simulations[18] have been widely used. However, most of these methods rely on meshing, which is very effective for low-dimensional systems, but for high-dimensional systems, they consume many computational resources and suffer from the curse of dimensionality. With the rise of artificial intelligence, neural networks have been applied to the study of dynamical systems,[19] due to their mesh independence and their outstanding performance. Tang *et al.*[20] investigated the KRnet method based on the flow-based generative model. The flow-based solution method can solve high-dimensional systems, but this model has strict conditions that the network needs to satisfy as an invertible mapping. Moreover, the method does not perform well for complex structures. Xu *et al.*[21–24] proposed a deep learning-based approach to solve the FP equation. It improves the physics-informed neural network (PINN)[25] by introducing a normalization condition such that zero solution is avoided. Deep KD-tree[23] and low-rank separation representation[21] have been introduced to solve high-dimensional problems efficiently, and the amount of required data was significantly reduced. Since any probability density function (PDF) can be well approximated by a set of Gaussian functions,[26,27] some scholars chose to use the Gaussian mixture model to approximate the solution of the FP equation.[28–31] Chen and co-workers[30] proposed the radial basis function neural network (RBFNN), which used a series of Gaussian functions to fit the solution of the FP equation. When the sum of the weights of the Gaussian kernel functions equals one, the numerical solution must satisfy the normalization condition, which transforms the normalization computational constraint into the sum of several weights.

Although the above methods have significant advantages over traditional methods in solving an FP equation, each training takes much time. In particular, parameter variations or similar systems still need to be trained. To accelerate the optimization process, an idea is to use an optimized neural network as the initial value for training, thereby improving the optimization speed when solving new equations. To this end, we choose a basic equation to pre-train the neural network and then use this foundation to accelerate the solving of the FP equation. Transfer learning is a machine learning idea that aims to improve the learning task in the target domain by utilizing knowledge from the source domain.[32,33] Its main goal is to solve the problem of insufficient or incomplete data in the target domain and to improve the learning performance of the target domain by exploiting the knowledge of the source domain. Chen *et al.*[34] used the method of transfer learning for solving partial differential equations (PDEs). They analyzed the transfer performance of PDEs with different source terms and Reynolds numbers and proved that the transferability gap increases with the distance between the basic and the target task. Goswami *et al.*[35] used transfer learning with DeepONet, a deep operator network, to solve PDEs. They proved the advantage of this method in various transfer learning scenarios under different conditions. Zhai *et al.*[36] added the results of rough MC numerical simulations as *a priori* knowledge so that the constraints of boundary conditions and normalization can be removed, and the method also reduces the amount of data input. Transfer learning is essentially pre-training a neural network to solve new equations.

To apply the transfer learning to accelerate the solving of an FP equation, a key question must be settled. Finding a suitable initial system for pre-training is essential when solving an equation. Therefore, inspired by the idea of transfer learning, this paper proposes a method based on the equivalent linearization to find a suitable pre-trained system to speed up the solving of the original systems. Before we are solving a large number of FP equations, we can pre-train a special neural network on a equivalently linearized system first. As an equivalent linearization method provides an approximation for the FP equation, we can continue the following learning from the pre-trained neural network. A series of numerical experiments are provided to demonstrate the effectiveness of the algorithm.

The rest of the paper is organized as follows. In Sec. II, a comprehensive description of the transfer learning idea for solving the FP equation by equivalent linearization, as well as the algorithmic framework of the equivalent linearization neural network (ELNN), is presented. Section III illustrates the effectiveness of the algorithm with one and two-dimensional systems under different noise excitations. In Sec. IV, the generalization ability and the impact of the penalty factor are discussed. In addition, we discuss the algorithm's capability for extension. Finally, conclusions are given in Sec. V.

II. METHODOLOGY

A. The FP equation

Consider an n-dimensional stochastic differential equation of the following form:

$$\dot{X} = f(X,t) + g(X,t)\,\xi(t), \tag{1}$$

where $X = [x_1(t), x_2(t), \ldots, x_n(t)]^T$ is an n-dimensional variable, which represents the state of the system at time t. The vector function $f(X,t) = [f_1, f_2, \ldots, f_n]^T$ is n-dimensional with respect to X, and it is generally nonlinear. The function $g(X,t) \in \mathbb{R}^{n\times s}$ is the diffusion term. $\xi(t) = [\xi_1, \xi_2, \ldots, \xi_s]^T$ are stochastic excitations. For simplicity of explanation, we take the stochastic system driven by Gaussian noise as an example. In this case, $\xi(t)$ represents a vector of a uncorrelated standard Gaussian white noise process. Under Markov's assumptions, we can obtain the following steady-state FP equation:

$$-\sum_{i=1}^{n}\frac{\partial}{\partial x_i}\left[f_i p(x)\right] + \sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial^2}{\partial x_j \partial x_k}\left[\frac{b_{jk}(x)}{2}p(x)\right] = 0, \tag{2}$$

where $b_{jk}, j, k = 1, 2, \ldots, n$ are the elements of the matrix $g(x,t)g^T(x,t)$. $p(x)$ denotes the transition PDF of the system (1).

We define

$$\mathscr{L}_{FP} = -\sum_{i=1}^{n} \frac{\partial}{\partial x_i} f_i + \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^2}{\partial x_j \partial x_k} \frac{b_{jk}(x)}{2}$$

as the FP operator. To obtain the solution of Eq. (2), the following conditions must be satisfied:

$$\mathscr{L}_{FP} p(x) = 0, \tag{3a}$$

$$\lim_{x \to \infty} p(x) = 0, \tag{3b}$$

$$\int_{R^n} p(x)\, dx = 1, \tag{3c}$$

$$p(x) \geq 0. \tag{3d}$$

Equations (1) and (2) describe the motion of a particle from different perspectives. The solution to Eq. (1) is the trajectory of a particle, while Eq. (2) gives the statistical properties of stochastic systems. The FP equation (2) transforms stochastic problems into deterministic ones. This provides a macroscopic perspective and is a central problem in nonlinear stochastic dynamics.

However, the solution of Eqs. (1) and (2) is not in one-to-one correspondence; i.e., two different stochastic systems may obtain the same PDF. Consider the system (1) and the following system:

$$\dot{Y} = f'(Y,t) + g'(Y,t)\,\xi'(t), \tag{4}$$

where each component represents the same meaning as the system (1). If the systems (1) and (4) share the same PDF, i.e., the steady and transient PDFs are identical, then we say that the systems are stochastically equivalent in the strict sense.[13] In this case, their corresponding FP equations also have the same solutions. If we begin with the FP equation, two systems that are stochastically equivalent in the strict sense need to satisfy two conditions as follows:

$$g'(Y,t) = g(X,t), \tag{5a}$$

$$f'(Y,t) = F\left(f(X,t)\right). \tag{5b}$$

The expressions for Eqs. (5a) and (5b) can be derived based on the detailed balance and the generalized stationary potential. Here, we only provide the final result. A detailed derivation can be found in Ref. 13. In stochastic analysis, if two systems are stochastically equivalent in the strict sense, they share the same statistical solutions. This property is crucial for the analysis of stochastic dynamical systems, as it allows one to obtain statistical information by studying a simpler one.

B. The equivalent linearization

Next, we will introduce the equivalent linearization method. This method was first introduced to different application areas by Booton[37] and Cauyhey,[38] respectively. Equivalent linearization is a method of converting a nonlinear system into an equivalent linear system, which makes it easier to analyze and solve nonlinear problems. It aims to create an equivalent linear system that is able to approximately represent the behaviors of the original nonlinear system in some aspects (e.g., dynamic response).

The classical equivalent linearization method determines a linear system by minimizing the mean square error between systems, which requires solving for the second-order moments of the stochastic process. However, since the Lévy process lacks finite variance and higher-order moments, the classical methods for determining equivalent linear systems cannot be used. To enhance the algorithm's applicability, we adopt a modified approach for solving the linear system. Grigoriu extends the equivalent linearization method to nonlinear systems driven by Lévy white noise,[39] introducing a new error formulation based on the characteristic function to derive the linearized system. Using this approach, the following linear system is obtained:

$$\dot{X} = \rho X + g(X,t)\,\xi(t), \tag{6}$$

where ρ is a constant $n \times n$ matrix. This method can achieve approximation accuracy similar to that of the classical equivalent linearization method.

Then, the stochastic responses of the original system (1) can be well approximated by solving the FP equation corresponding to Eq. (6). The term "equivalent" refers to the equivalence of the stochastic responses of the systems, which is yielded by approximating the nonlinear term with a linear one.

C. The ELNN algorithm

In this subsection, we will elaborate on the presented method based on a transfer learning idea, called the ELNN algorithm, for solving the FP equation (2).

A natural idea is that if we have solved the system (4) before solving the probabilistic solution of the system (1), the PDF of the system (1) can be obtained directly. However, finding this equivalent equation requires additional computational effort, and solving either the original FP equation or its equivalent form still requires a full neural network optimization process. Therefore, studying the equivalent equation does not speed up the process. To accelerate the optimization process, we need to identify a single simplified equation that performs well across FP equations with different forms. By using the neural network solution of this simplified equation as the initial value, the optimization algorithms for solving FP equations could converge faster. Inspired by the idea of transfer learning, we consider systems that are a little weaker than stochastically equivalent systems in the strict sense: only Eq. (5a) is satisfied, and we call the systems "weakly" stochastically equivalent. In this case, the PDFs of both systems are different, but the corresponding FP equation necessarily contains some of the same information. Then, when we use neural networks to solve the FP equation, there will be similarities in neural network parameters.

When we solve an FP equation with an unknown solution, we can construct "weakly" stochastically equivalent systems by the method of equivalent linearization. It should be noted that the value of ρ in Eq. (6) is a constant, and the specific values of these constants will not significantly affect the training time or accuracy. Ideally, the closer ρ is to the equivalent linearized system, the better the result, as it more accurately reflects the system's response. We want to find

a simple initial system, which is not unique. To enhance the generalization of the algorithm and simplify the process, we directly set Eq. (6) in a simple form for the system (1), which is given in detail in Sec. III.

After the previous derivation, we get an approximation of the system (1), corresponding to the FP equation,

$$-\sum_{i=1}^{n}\frac{\partial}{\partial x_i}\left[\sum_{j=1}^{n}\rho_{ij}Xp\right]+\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial^2}{\partial x_j\partial x_k}\left[\frac{b_{jk}(x)}{2}p\right]=0, \quad (7)$$

where ρ_{ij} are the elements of the matrix ρ.

Then, the framework of transfer learning has been completed. In this framework, the PDFs described by Eqs. (2) and (7) represent the target domain and the source domain, respectively. These domains share similar information, allowing knowledge transfer from the source domain to improve learning in the target domain. The system (6) for the source domain is derived using the method of equivalent linearization, which simplifies the complex dynamics of the original system (1). Finally, Eq. (5a) ensures that the transfer effect is positive, meaning that the knowledge transfer from the source domain to the target domain leads to an improvement in the target domain's performance, rather than degrading it.

We choose the simple artificial neural network with L layers and n_l nodes in each hidden layer to represent the solution of the FP equation. The tunable parameters of the neural network are denoted as Θ. The output p is the corresponding PDF. The training set and the boundary set are $T_D=\{X_i^D\}_{i=1}^{N_D}$, $T_B=\{X_i^B\}_{i=1}^{N_B}$, where T_B represents the boundary of the training set T_D. N_D and N_B indicate the number of data in the training and boundary sets, respectively. In the one-dimensional case, We assume that the computational domain is $[a,b]$, which is discretized uniformly with a step size of Δ_t. Thus, we obtain $N_D=(b-a)/\Delta_t$. The activation functions of the layers in the three examples given below are tanh function, which can be adjusted appropriately according to the network effects. The nonnegativity of the solution, i.e., the condition (3d), is ensured by the SoftPlus function in the output layer.[22] The Adam algorithm is performed to minimize the loss function.

When solving the steady-state FP equation (2), the four conditions in Eq. (3) must be satisfied. The constraints in Eqs. (3a)–(3c) correspond to the three parts on the right-hand side of Eq. (8), and the loss function is as follows:

$$\mathscr{L}_{Loss}=a_1\cdot\frac{1}{N_D}\sum_{i=1}^{N_D}\left(\mathscr{L}_{FP}p(X^D)\right)^2+a_2\cdot\frac{1}{N_B}\sum_{i=1}^{N_B}\left(p(X^B)\right)^2+a_3\cdot\left|\sum_{i=1}^{N_D}{\Delta_t}^{Dim}p(X^D)-1\right|^2, \quad (8)$$

where $a_i(i=1,2,3)$ are penalty factors that help the loss function converge faster, and Dim represents the dimension of the variable X.

The above network setup is applicable to both pre-train (step i) and fine-tune (step ii) the network. The detailed flow of the proposed ELNN algorithm is as follows

(i) The first step of transfer learning is to pre-train the neural network. When we get the FP equation (2), we first need to see if we have solved Eq. (7) before. If so, go directly to the second step. Otherwise, we first linearize the corresponding stochastic differential equation equivalently. Refer to Sec. II A for the detailed steps. Then, we use the DL-FP algorithm to solve Eq. (7). Save the parameters of the neural network when the pre-training is finished. The neural network trained in this step contains the information of the system (1) when Eq. (5a) is satisfied.

(ii) The second step is to fine-tune the neural network for solving new FP equations. Compared to the DL-FP, the most important feature of the ELNN algorithm is the setting of the initial parameters of the neural network. In the DL-FP algorithm, it is randomly initialized, but the ELNN algorithm makes full use of *a priori* knowledge.

ALGORITHM 1. The proposed ELNN algorithm.

Pre-training: Solving the FP equation (7) after the equivalent linearization
Input: The training set and the boundary set of Eq. (7).
Output: The network parameters Θ.
1. **Equivalent linearization:** Perform a simple equivalent linearization for the system (1).
2. Solve the simplified FP equation (7) with a neural network.
3. Save the network parameters Θ.

Applying: Solving the FP equation (2)
Input: The training set and the boundary set of Eq. (2), the network parameters Θ.
Output: The solution of the FP equation (2).
1. **Transfer learning:** Load the parameters Θ to initialize the new neural network.
2. Input the training set of Eq. (2) and then fine-tune the new neural network.
3. **Return:** results.

If we solve only a single FP equation, the overall training time is not necessarily reduced. However, when we need to solve many FP equations, the training time can be greatly reduced by the transfer learning method.

III. EXAMPLES

A. Example 1

Consider a one-dimensional nonlinear stochastic system as follows:

$$\dot{x}=\gamma x-\beta x^3+\sqrt{2D_1}W(t),\ (\gamma>0,\beta>0), \quad (9)$$

where $W(t)$ is a standard Gaussian white noise, and D_1 represents the noise intensity. The corresponding steady-state FP equation is

$$-\frac{\partial}{\partial x}\left[\left(\gamma x-\beta x^3\right)p(x)\right]+D_1\frac{\partial^2}{\partial x^2}p(x)=0. \quad (10)$$

The analytical solution of Eq. (10) is

$$p(x)=C_1\cdot\exp\left[\frac{1}{4D_1}\left(2\gamma x^2-\beta x^4\right)\right],$$

where C_1 is a normalization constant.

Next, we apply the ELNN algorithm given in Sec. II to solve Eq. (10). We set $D_1 = 0.125$. As outlined before, the exact determination of ρ is not essential for the linearization. Rather, selecting suitably simple values that ensure the pre-trained system achieves steady-state behavior is considered sufficient. To illustrate this, we consider the following simplified system:

$$\dot{x} = -x + \sqrt{2D_1}W(t). \tag{11}$$

We directly present the simplified system (11) in its simplest form, avoiding complex derivations. The FP equation corresponding to system (11) is

$$\frac{\partial}{\partial x}\left[xp\left(x\right)\right] + D_1\frac{\partial^2}{\partial x^2}p\left(x\right) = 0. \tag{12}$$

First, a neural network is used to solve Eq. (12). The computational domain is $[-2.5, 2.5]$, and the step size is $\Delta_t = 0.01$. In the neural network, $L = 4, n_l = 20$. Here, we choose to iterate 50 000 times. We set the penalty factors to $a_1 = 1, a_2 = 1$, and $a_3 = 1$. It is important to note that pre-training is primarily aimed at finding a neural network that is closer to the system to be solved. Therefore, we do not have to pursue the accuracy of the solution too much when training the neural network. Then, save the parameters of the trained network.

Next, we load the trained neural network parameters as the initial values and then solve Eq. (10) for 100 FP equations with different system parameters. We choose $\gamma = [0.1, 0.2, \ldots, 1.0]$ and $\beta = [0.1, 0.2, \ldots, 1.0]$ to verify the advantages of the proposed ELNN method in solving the FP equation (10). Figure 1(a) represents the difference of the loss function of the DL-FP and the ELNN methods. We denote the iteration number at which the loss function first reaches 10^{-3} using the DL-FP method as t_1 and that for the ELNN method as t_2. Figure 1(a) represents the difference of $\Delta_{iter} = t_1 - t_2$. The numbers in the grid cells represent the values of Δ_{iter} . If $\Delta_{iter} > 0$, it represents a faster decrease of the loss function when using the ELNN algorithm. As shown in the figure, the ELNN algorithm decreases much faster than the DL-FP. Sometimes, relying only on the loss function does not indicate that the solution is accurate. Therefore, we give the following accuracy formula:

$$R = 1 - \frac{\|\hat{P} - P\|_2}{\|P\|_2},$$

where P represents the exact solution and $\hat{P}$ represents the numerical solution. We denote R_{ELNN}, R_{DL-FP} as the accuracy when using the ELNN algorithm and the DL-FP algorithm, respectively, with the same iterations. Figure 1(b) gives a comparison of the accuracy of the two algorithms under different γ and β. We can see from Fig. 1 that the ELNN algorithm outperforms the DL-FP algorithm in a statistical sense.

With a specific parameter selected $\gamma = 0.3, \beta = 0.5$, Fig. 2 visualizes the characteristics of transfer learning in more detail. We see that the ELNN method learns the contours of the solution faster than the DL-FP method. To clearly illustrate the decline of the loss function, a minimum value filter was applied to smooth the curve. It can be seen that the ELNN method always outperforms the DL-FP method in most cases (Fig. 3).

B. Example 2

Consider a second-order nonlinear stochastic system as follows:

$$\ddot{x} + \eta\dot{x} + \lambda x + \kappa x^3 = \sqrt{2D_2}W(t), \tag{13}$$

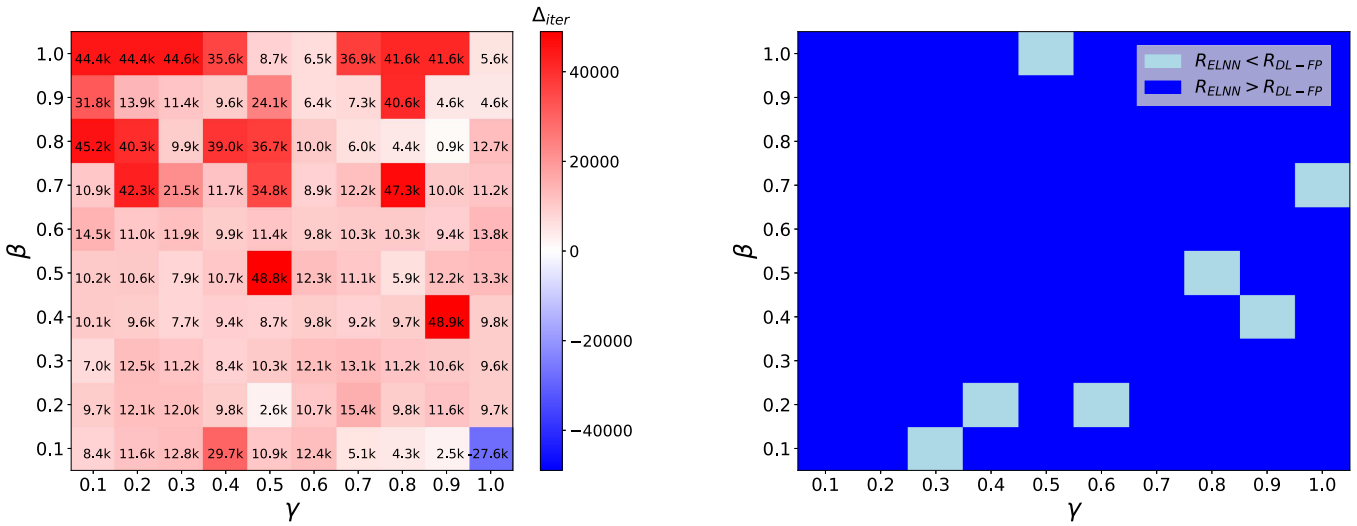


FIG. 1. Comparisons between the ELNN and DL-FP algorithms under different γ, β for the system (9). (a) The difference of iterations for which the loss function first reaches 10^{-3}. (b) The accuracy of the two algorithms with the 50 000 iterations.

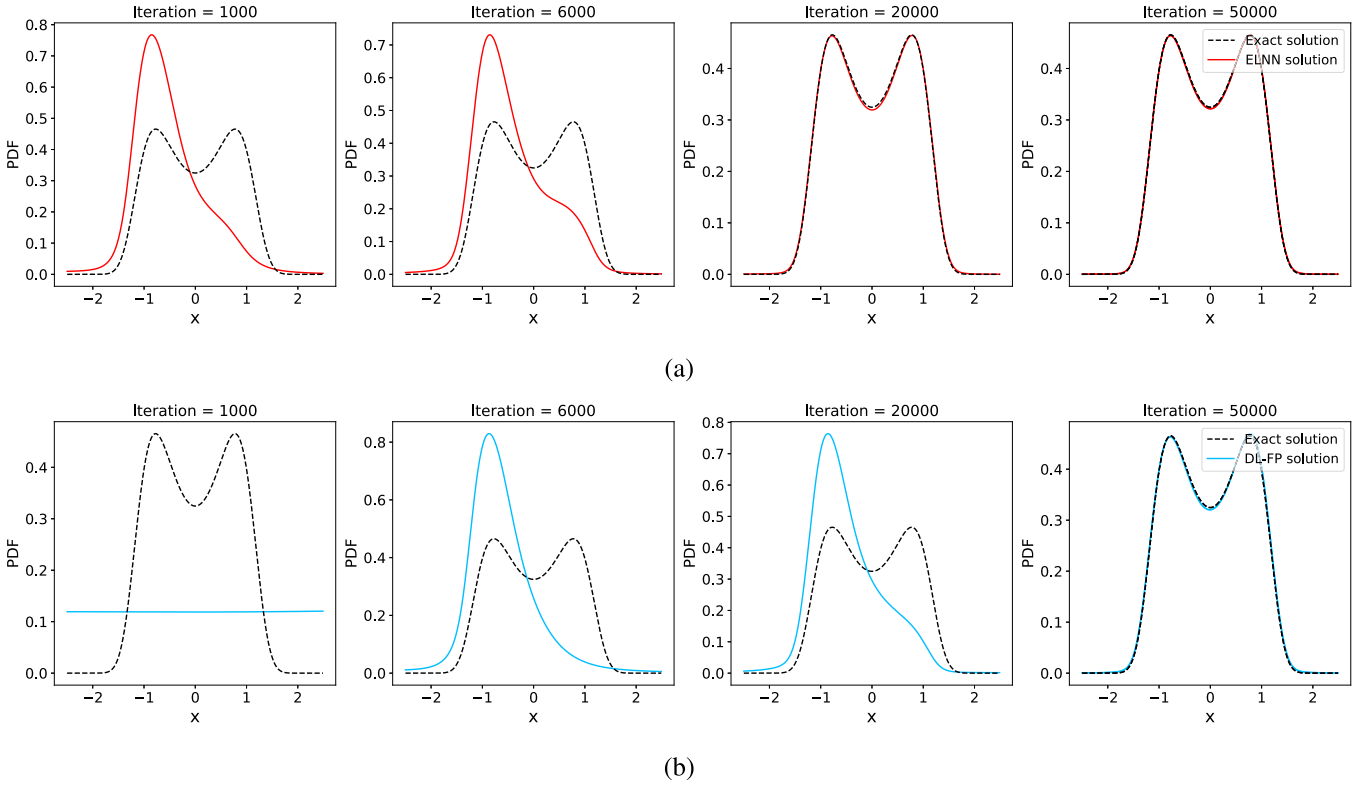


FIG. 2. Comparisons of the ELNN and DL-FP algorithms under different iterations for the system (9). (a) Solutions using the ELNN algorithm. (b) Solutions using the DL-FP algorithm.

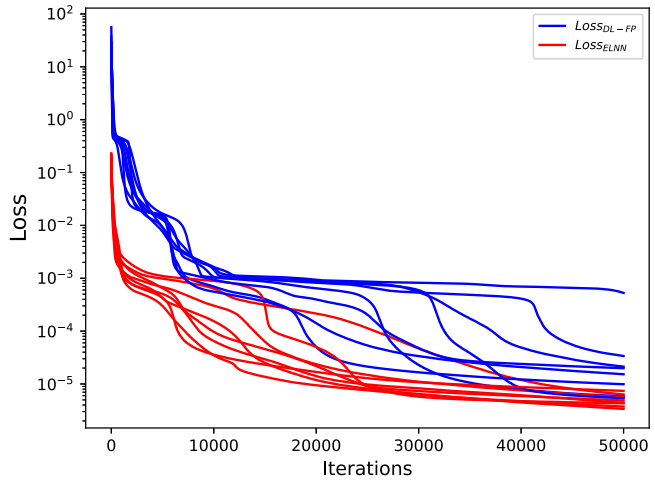


FIG. 3. Comparison of the loss function with iterations under different algorithms for the system (9).

where $W(t)$ is a standard Gaussian white noise. $\eta > 0$ is the damping coefficient. λ and $\kappa > 0$ represent the linear and nonlinear stiffness coefficients, respectively. The noise intensity is D_2. Equation (13) can be rewritten as

$$\begin{cases} \dot{x} = y, \\ \dot{y} = -\eta y - \lambda x - \kappa x^3 + \sqrt{2D_2}W(t). \end{cases} \tag{14}$$

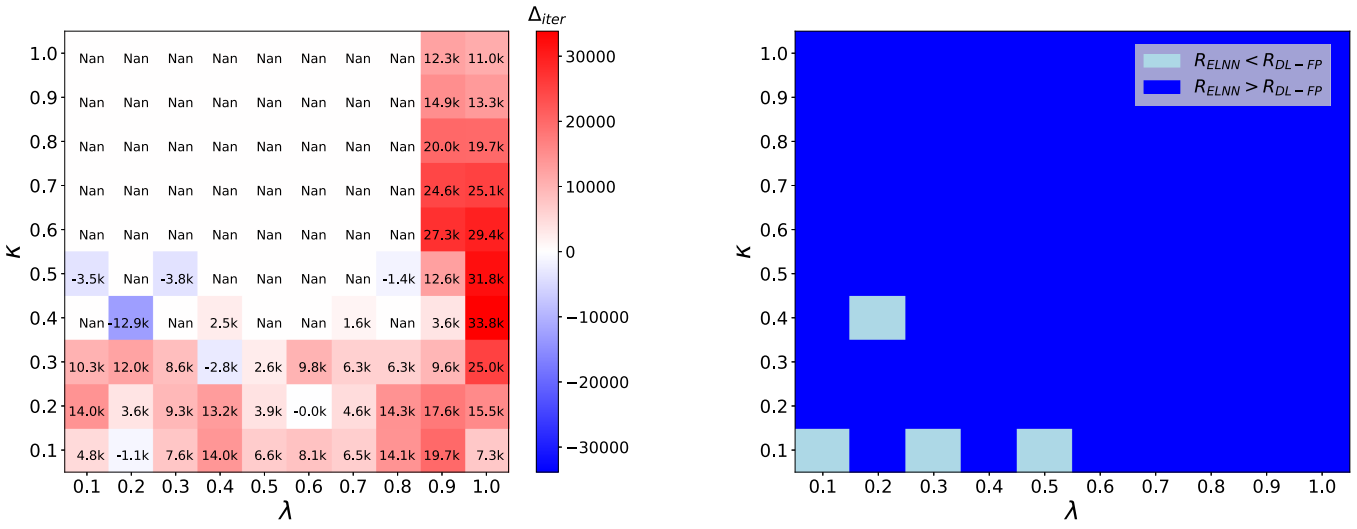
The corresponding steady-state FP equation for the system (14) is

$$-\frac{\partial}{\partial x}[yp(x,y)] - \frac{\partial}{\partial y}\left[\left(-\eta y - \lambda x - \kappa x^3\right)p(x,y)\right] + D_2\frac{\partial^2}{\partial y^2}p(x,y) = 0. \tag{15}$$

The exact solution to Eq. (15) is

$$p(x,y) = C_2 \cdot \exp\left[-\frac{\eta}{2D_2}\left(y^2 + \lambda x^2 + \frac{\kappa}{2}x^4\right)\right],$$

where C_2 is a normalization constant. In a similar manner, we focus on a stochastic system that is capable of reaching a steady state and



(a) (b)

FIG. 4. Comparisons between the ELNN and DL-FP algorithms under different λ, κ for the system (13). (a) The difference of iterations for which the loss function first reaches 10^{-3}. (b) The accuracy of the two algorithms with the 50 000 iterations.

adopt its simplest admissible form,

$$\ddot{x} = -\dot{x} - x + \sqrt{2D_2}W(t). \tag{16}$$

Then, we have

$$\begin{cases} \dot{x} = y, \\ \dot{y} = -y - x + \sqrt{2D_2}W(t). \end{cases}$$

The steady-state FP equation for the system (16) is

$$-\frac{\partial}{\partial x}[yp(x,y)] - \frac{\partial}{\partial y}\left[(-y-x)\,p(x,y)\right] + D_2\frac{\partial^2}{\partial y^2}p(x,y) = 0. \tag{17}$$

The system parameters are set to $\eta = 0.2$ and $D_2 = 0.05$. First, the DL-FP algorithm is used to solve Eq. (17). The computational domain is $[-2.0, 2.0] \times [-2.0, 2.0]$, and the time step $\Delta_t = 0.01$. The network is parameterized by $L = 5, n_l = 20$, and the number of iterations is 50 000. The penalty factors in the network are $a_1 = 400, a_2 = 1$, and $a_3 = 0.01$. Then, save the parameters of the network and perform transfer learning. For different λ and κ, new networks are created to solve Eq. (15) without the need for pre-training for $\lambda = [0.1, 0.2, \ldots, 1.0]$ and $\kappa = [0.1, 0.2, \ldots, 1.0]$. Figure 4 has the same meaning as Fig. 1. The "Nan" in Fig. 4(a) refers to the fact that the loss function does not reach 10^{-3} with neither of the two algorithms. However, the solution accuracy is always above 75% when using the ELNN algorithm; when using the DL-FP algorithm, the solution accuracy is mostly below 30%. Furthermore, Fig. 4(b) offers a more precise illustration of the distinctions between the two algorithms. With the above parameter settings, the training of the solution of system (13) takes much time for the DL-FP method when compared to the ELNN method. The training time for the two-dimensional system (13) is roughly ten times longer compared to the one-dimensional system (9) on the same equipment. We have also solved other systems and found that the training time increases with the increase in dimensions. This shows that the advantage of the ELNN algorithm becomes more obvious as the dimensionality increases.

Let $\lambda = 1.0$ and $\kappa = 0.2$, the two algorithms are used to solve Eq. (15). Figure 5 demonstrates the error plots at different iterations. The ELNN algorithm is on the left and the DL-FP algorithm is on the right. It is clear that the ELNN algorithm has a smaller error for the same number of iterations; i.e., the network learns the shape of the solution faster than the DL-FP. Figure 6 gives the decrease of the loss function under different algorithms. It can be seen that the loss function of the ELNN algorithm decreases faster than the DL-FP.

C. Example 3

The proposed method also works well for solving the FP equation corresponding to a system under more complex noise excitations. In this subsection, the ELNN algorithm will be used to solve a one-dimensional FP equation for the joint excitation of Gaussian and Lévy noise. Consider a nonlinear stochastic system as follows:[40]

$$\dot{x} = -x^5 + \sqrt{2D_3}W(t) + \xi_L(t), \tag{18}$$

where $W(t)$ is a standard Gaussian white noise, with the noise intensity D_3. In addition, $\xi_L(t)$ is a symmetric α-stable Lévy noise,[41] characterized by the noise intensity D_L and the stability index α. The steady-state FP equation for the system (18) is

$$\frac{\partial}{\partial x}\left[x^5 p\,(x)\right] + D_3\frac{\partial^2}{\partial x^2}p\,(x) + D_L\frac{\partial^\alpha}{\partial|x|^\alpha}p(x) = 0, \tag{19}$$

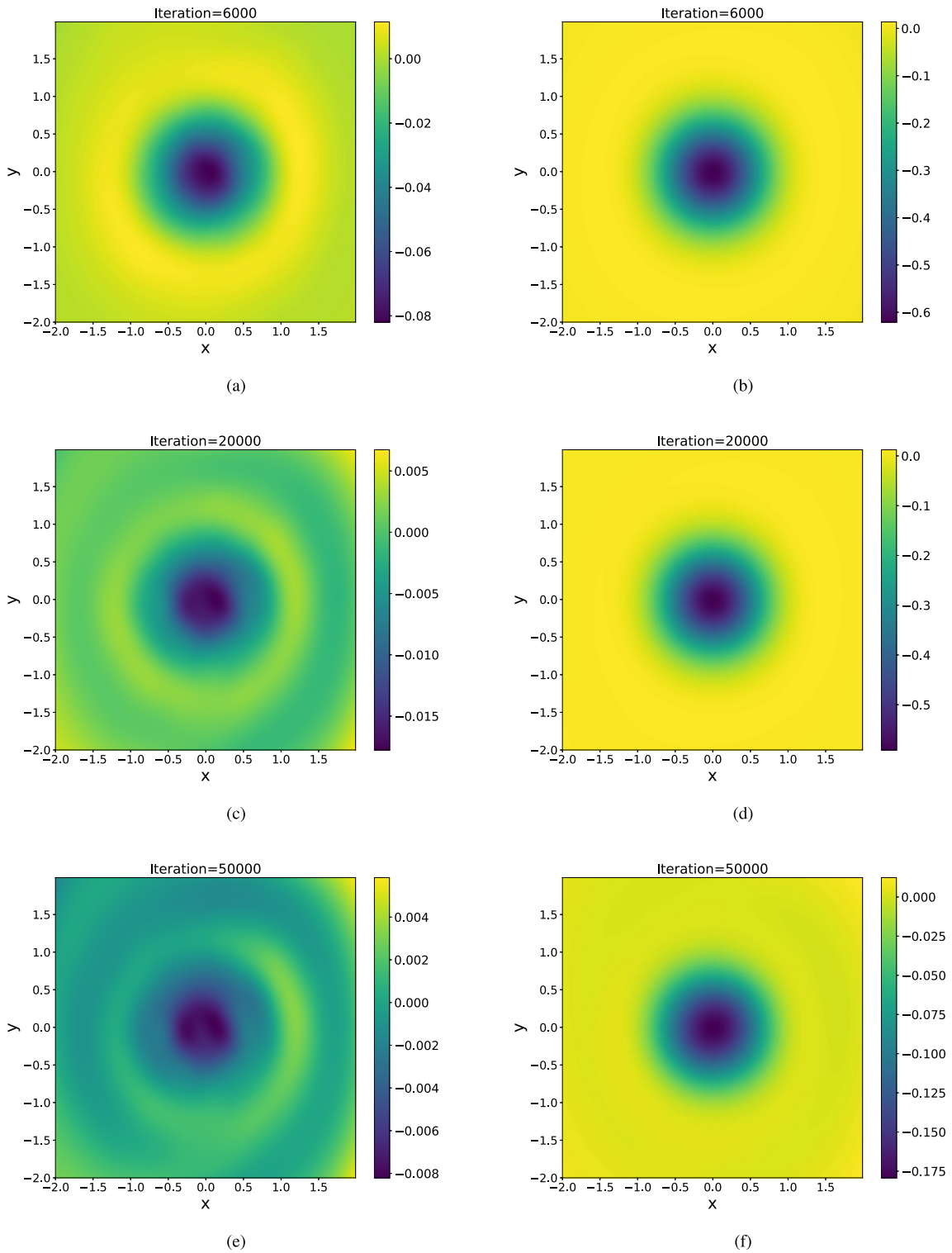


FIG. 5. Comparisons of the ELNN and DL-FP algorithm for the system (13). (a), (c), and (e) Error plots of the system against the exact solution when using the ELNN algorithm for 6000, 20 000, and 50 000 iterations, respectively. (b), (d), and (f) Error plots using the DL-FP algorithm for the same iterations.

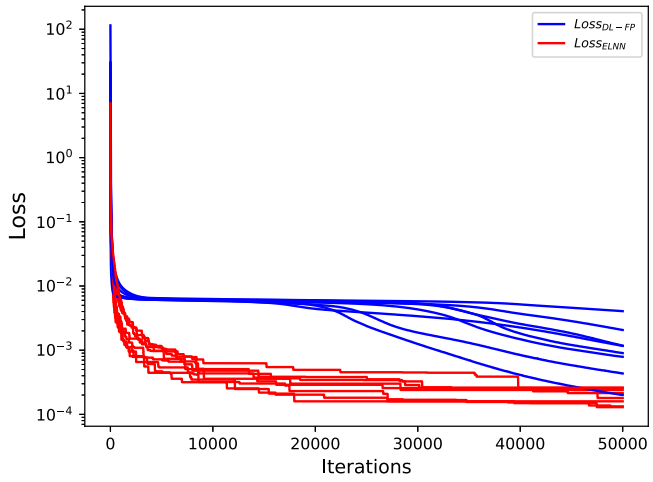


FIG. 6. Comparison of the loss function with iterations under different algorithms for the system (13).

where $\frac{\partial^\alpha}{\partial|x|^\alpha}p(x)$ is the Riesz fractional derivative. To proceed, we adopt the following simplified linear stochastic system:

$$\dot{x} = -x + \sqrt{2D_3}W(t) + \xi_L(t). \tag{20}$$

The steady-state FP equation for the system (20) is

$$\frac{\partial}{\partial x}\left[xp\,(x)\right] + D_3\frac{\partial^2}{\partial x^2}p\,(x) + D_L\frac{\partial^\alpha}{\partial|x|^\alpha}p(x) = 0.$$

The FP equation for stochastic systems under Lévy noise excitation is fractional, which cannot be solved using automatic differentiation techniques. We employ a "fractional central derivative" approach with second-order accuracy to approximate the Riesz fractional derivative.[24]

First, the DL-FP algorithm is used to solve Eq. (20). The computational domain is $[-3.0, 3.0]$, and $\Delta_t = 0.05$. The network is set up as $L = 4, n_l = 40$, and the iteration is = 100 000. The values of the penalty factors are $a_1 = 200, a_2 = 1$, and $a_3 = 1$. After the training is completed, we save the network parameters.

Since the system (18) has no unknown parameters and no analytical solution, the comparison, similar to that shown in Fig. 1, is

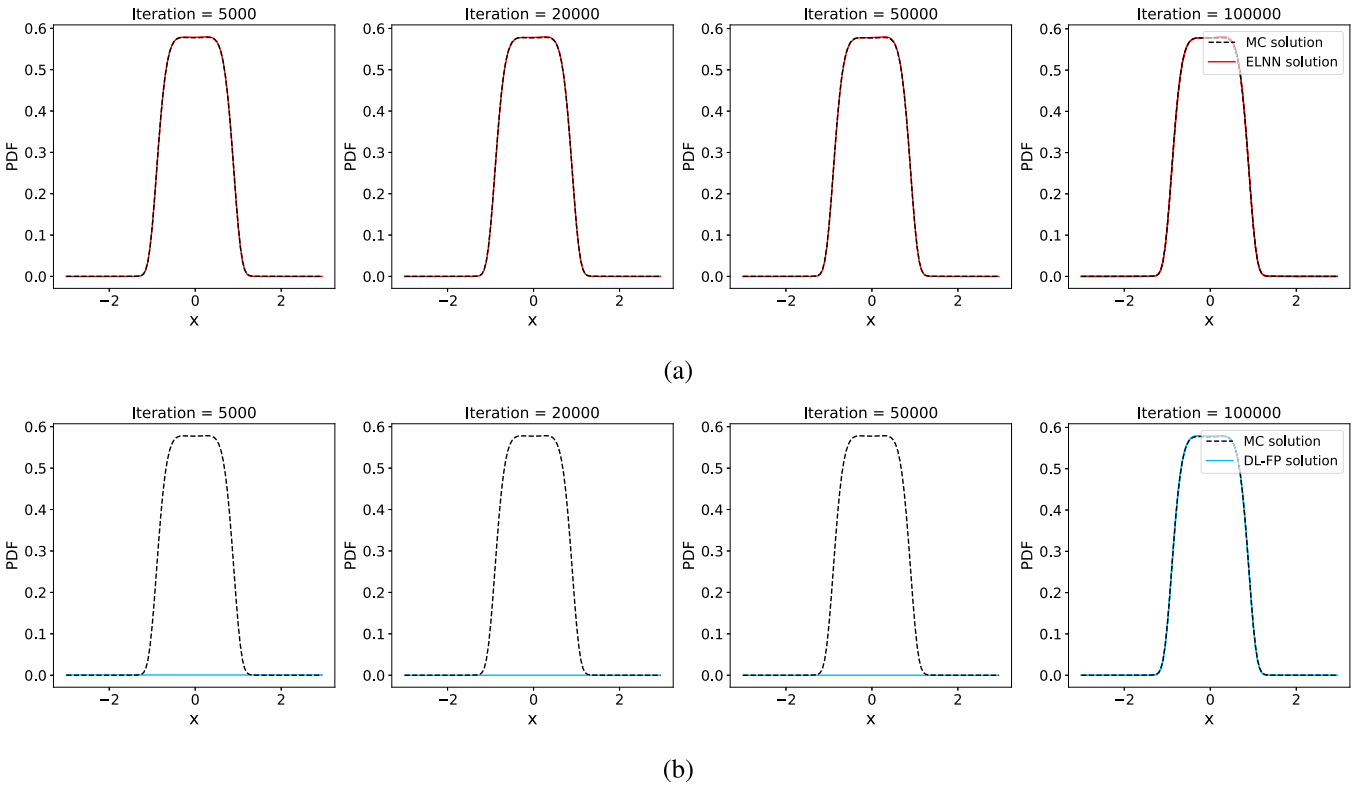


FIG. 7. Comparisons of the ELNN and DL-FP algorithms under different iterations for the system (18). (a) The solutions from the ELNN algorithm. (b) The solutions from the DL-FP algorithm.

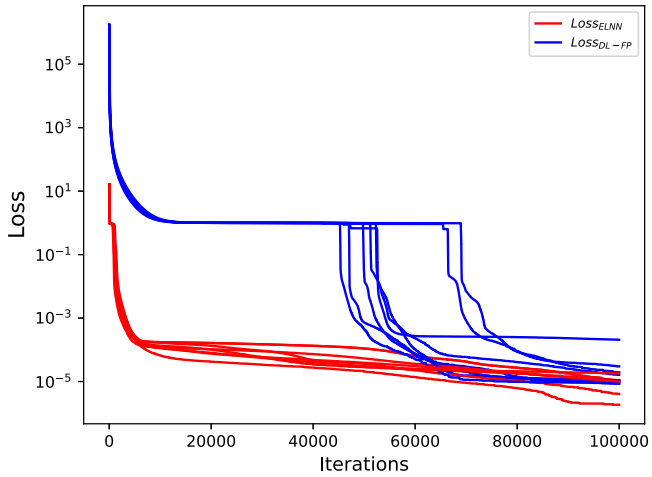


FIG. 8. Comparison of the loss function with iterations under different algorithms for the system (18).

omitted. We employ the ELNN algorithm and the DL-FP algorithm to obtain the results, respectively. Figure 7 shows the variation of the loss function, and it can be seen that the red curve decreases very quickly. Because Eq. (19) does not have an analytical solution, we use Monte Carlo simulation as a comparison. Figure 8 presents the solution of both methods. We find that the ELNN algorithm can learn the solution of the equation faster than the DL-FP because it has been pre-trained and the network contains similar information.

IV. DISCUSSION

The results in Sec. III clearly demonstrate that the proposed ELNN algorithm is highly effective in solving FP equations. Once the model is obtained through pretraining, it can be applied to solve a class of FP equations, demonstrating strong generalization capabilities. In this section, we will address the following key questions. (1) What about the generalization ability of the ELNN algorithm? (2) How does the penalty factor influence the efficiency of the ELNN algorithm? (3) What is the feasibility of the ELNN algorithm when the conditions, Eq. (5a), are not satisfied?

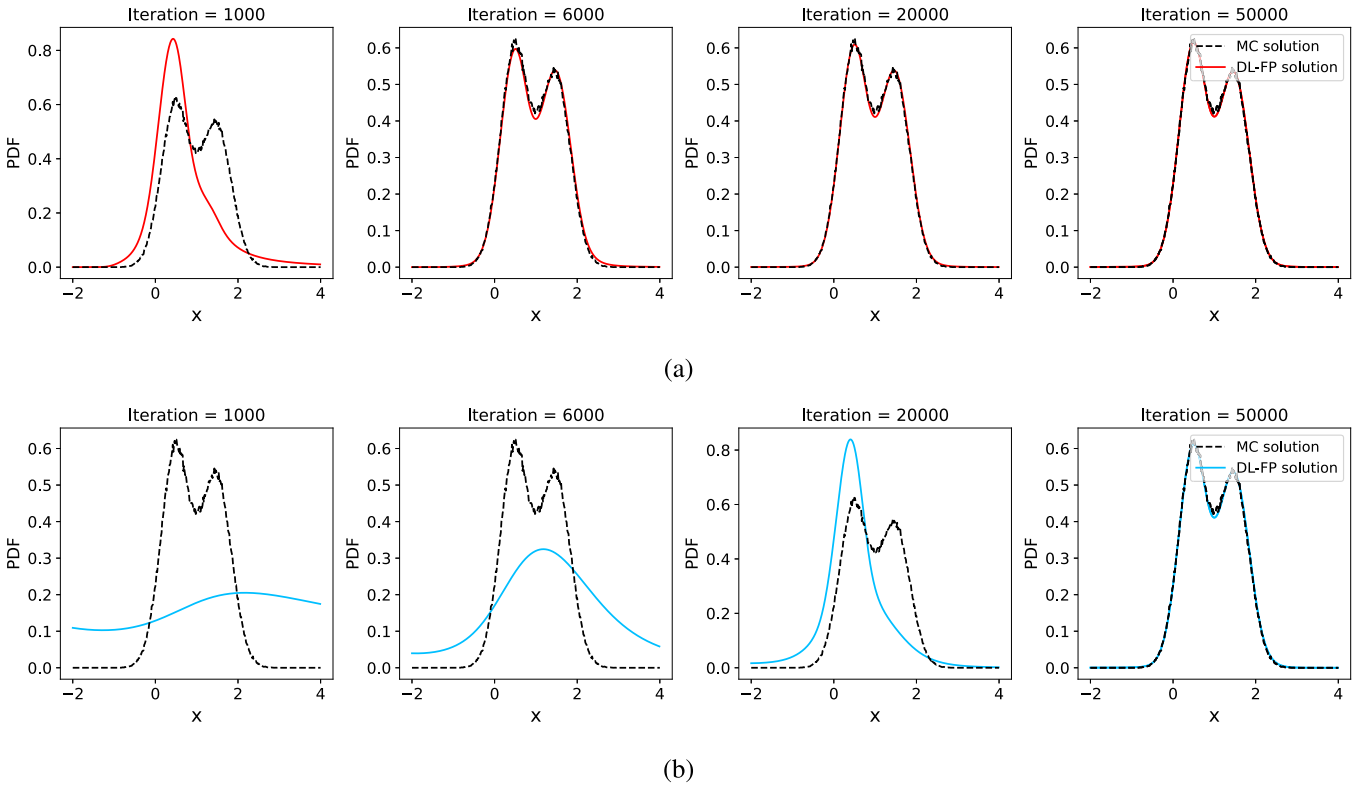


FIG. 9. Comparisons of the ELNN and DL-FP algorithms under different iterations for the system (21). (a) The solutions from the ELNN algorithm. (b) The solutions from the DL-FP algorithm.

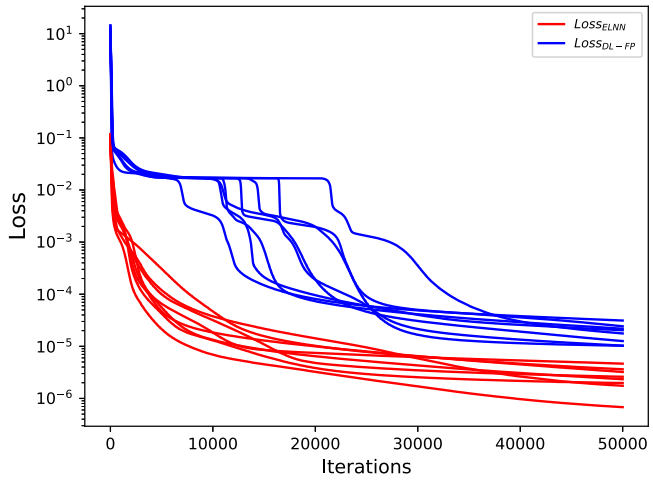


FIG. 10. Comparison of the loss function with iterations under different algorithms for the system (21) with the same simplified system (11).

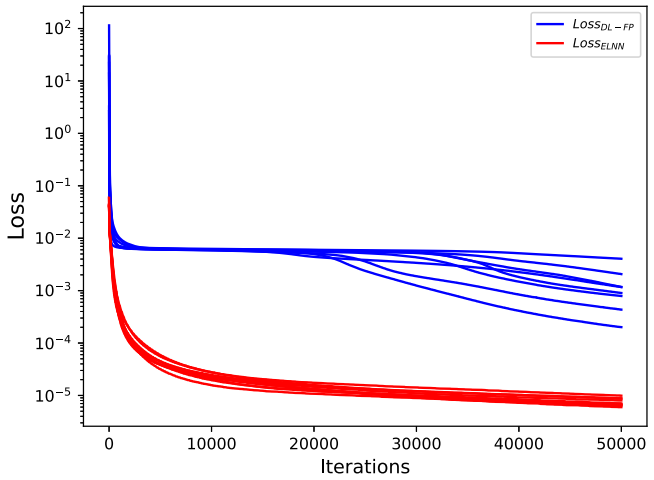


FIG. 11. Comparison of the loss function with iterations under different algorithms for the system (13) under the ELNN without penalty factors ($a_i = 1$) and the DL-FP with penalty factors ($a_1 = 400, a_2 = 1, a_3 = 0.01$).

A. The generalization ability

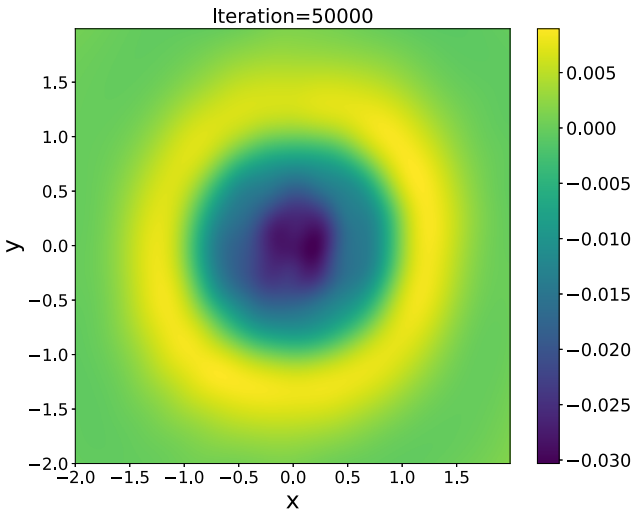
As mentioned earlier, the model obtained through pre-training can speed up the solution of a class of FP equations. However, in the example provided in Sec. III, only the solution for a single equation is presented. Therefore, we will first discuss the generalization ability of the ELNN algorithm, i.e., whether the pre-trained model can solve a class of FP equations. Here, we just take Example 1 as an illustrated example.

Consider the following system:[22]

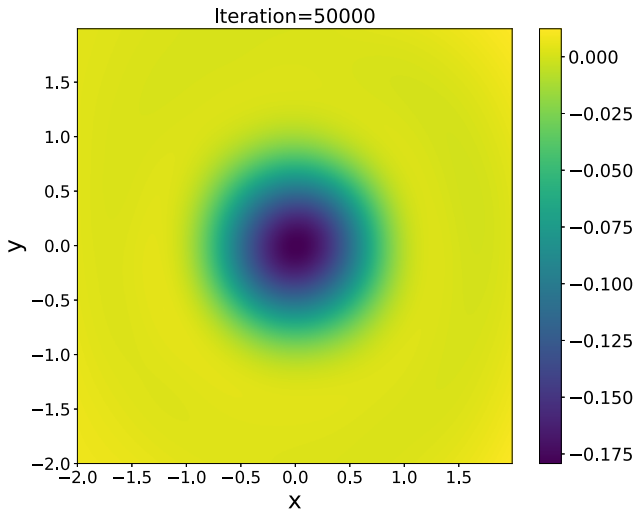
$$\dot{x} = a - x + \frac{x^8}{1+x^8} + \sqrt{2D_4}W(t), \tag{21}$$

where $W(t)$ and D_4 represent the same meanings as in the system (9), and $a = 0.5$. The corresponding steady-state FP equation is

$$-\frac{\partial}{\partial x}\left[\left(a - x + \frac{x^8}{1+x^8}\right)p(x)\right] + D_4\frac{\partial^2}{\partial x^2}p(x) = 0. \tag{22}$$

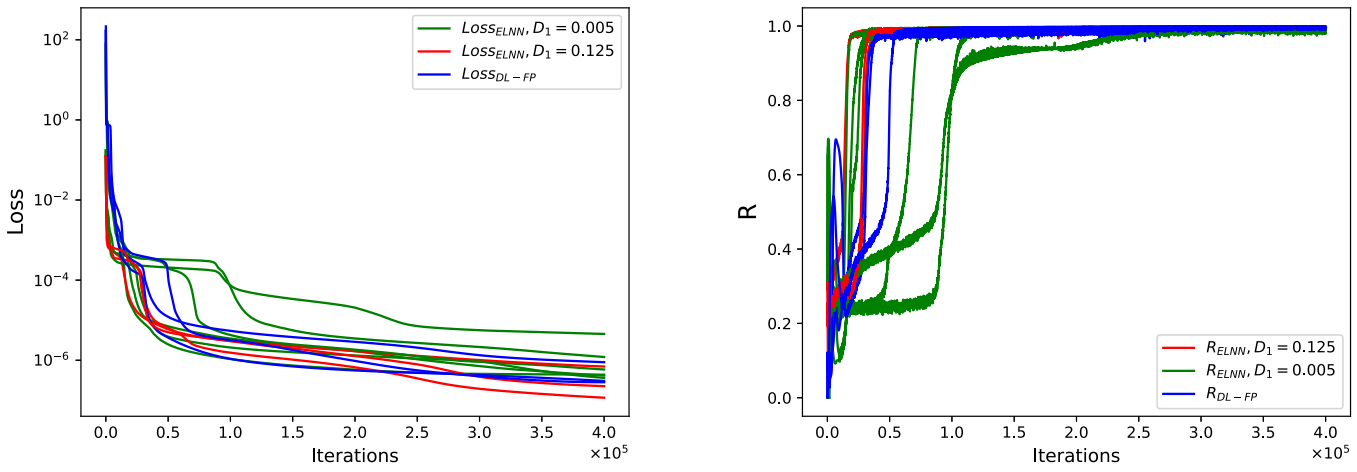


(a)



(b)

FIG. 12. Comparisons of the ELNN and DL-FP algorithms for the system (13). (a) Error plots with ELNN ($a_i = 1$) after 50 000 iterations. (b) Error plots with DL-FP ($a_1 = 400, a_2 = 1, a_3 = 0.01$) for the same iterations.



(a) (b)

FIG. 13. Comparisons between the ELNN and DL-FP algorithms for the system (9) when the conditions of Algorithm 1 are not satisfied. (a) Comparison of the loss function with iterations under different algorithms. (b) The accuracy with the same iterations.

After equivalent linearization of the system (21), we get the same simplified system (11). Thus, the pre-trained model from Example 1 can be used to solve Eq. (22) in an accelerated way. The results are illustrated in Figs. 9 and 10. We can intuitively observe that the pre-trained model from Example 1 is able to learn the solution of Eq. (22) effectively. This reveals the model's exceptional generalization ability, highlighting its ability to maintain high performance and effectively adapt to the FP equations corresponding to systems with the same simplified systems.

B. The penalty factor

It is well known that appropriate penalty factors can significantly accelerate the convergence of the loss function. However, the selection of the penalty factors lacks theoretical guidance, often leading to a waste of time in choosing the appropriate ones.[22] To ensure that each component in Eq. (8) converges at approximately the same rate, a feasible approach is to plot the variation of each part of the loss function with respect to the number of iterations. By increasing the penalty factors for the components that converge more slowly, uniform convergence rates for all components can be achieved. In this subsection, taking the system (13) as an example, we discuss the performance of the ELNN algorithm in the absence of penalty factors.

Let $\lambda = 1.0, \kappa = 0.2$, and keep the other parameters unchanged. The DL-FP algorithm with a penalty factor and the ELNN algorithm without a penalty factor ($a_i = 1$) are used separately to solve Eq. (14). The results obtained are shown in Figs. 11 and 12. The loss function decreases more rapidly under the ELNN algorithm than the DL-FP, as shown in Fig. 11. It can reach the order of 10^{-5}, smaller than the order of the loss function in Fig. 6. This difference is due to the parameter $a_1 = 400$ of the loss function in Fig. 6, which amplifies the loss value. Therefore, the variation of the loss function does not fully explain the issue. To better illustrate this, we provide the error plot in Fig. 12. With the same number of iterations, the error of the ELNN algorithm is significantly smaller than the DL-FP. Thus, even without an appropriate penalty factor, the ELNN algorithm demonstrates excellent performance.

C. The extension of the ELNN algorithm

The transfer learning fully utilizes the prior knowledge and improves the efficiency of solving the FP equation by neural

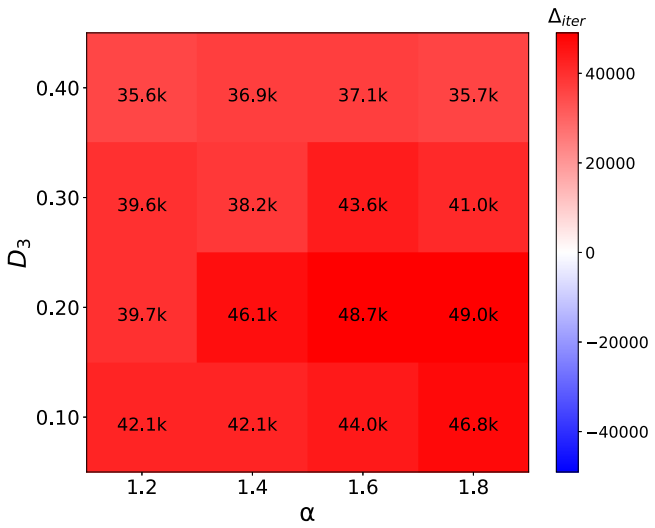


FIG. 14. The difference of iterations for the loss function to reach 10^{-3} for the first time when using the two algorithms at different α, D_3 for the system (18) under partially satisfied Algorithm 1.

networks. In this subsection, we will discuss the performances of the ELNN method when Eq. (5a) is not satisfied.

In this case, we take Example 1 as a representative for analysis. We fix the parameters $\gamma = 0.3$ and $\beta = 0.5$. We choose different noise intensities $D_1 = 0.005$ and $D_1 = 0.125$, respectively. According to the first step of the ELNN algorithm, we get two models and then fine-tune the network. The results are shown in Fig. 13. It can be seen that when the noise intensities of systems (9) and (11) are not the same, the impact varies. Compared to the DL-FP algorithm, the performance is sometimes as outstanding as in Example 1, while at other times, it has little impact on the solution. And sometimes it takes a long time (the number of training iteration) to "erase" the network's previous memory and continue to learn the properties of the new system.

Since the system (18) is subjected to two noise excitations, we discuss the case where only one noise intensity is consistent; that is, the conditions of Algorithm 1, Eq. (5a), are partially satisfied. Consider the system (18), where the stability index α and the Gaussian noise intensity D_3 change while fixing D_L. However, it can be seen from Fig. 14 that the fit remains good. This shows that in random cases when the conditions are appropriately relaxed, the ELNN algorithm can still achieve good results.

V. CONCLUSIONS

In this paper, a transfer learning algorithm called the ELNN is proposed to solve the FP equations. Three prototypical examples show in a statistical sense that the presented method can solve the FP equations efficiently and quickly. The ELNN method can achieve better performance than the DL-FP, illustrated by a series of comparisons. In addition, an example for joint noise excitation shows that the ELNN algorithm still works well when not all assumptions are fully satisfied. The ELNN algorithm makes full use of prior knowledge. Its core is transfer learning. Approximations are first made by equivalent linearization, and the model is highly generalizable. Although this paper only compares with the DL-FP, the idea of transfer learning has broad applicability and can be extended to other algorithms for solving the FP equations.

Additionally, we find that the ELNN algorithm can quickly solve the FP equations even without appropriate penalty factors. This reduces the dependence on experimental experience. We have also discussed the effectiveness of the ELNN algorithm in solving the FP equations when Eq. (5a) is not satisfied. It is sometimes better and sometimes worse. Therefore, how to quantify the transfer learning model containing information about the equations to be solved is also something that needs to be studied in depth in the future.

ACKNOWLEDGMENTS

This work was partly supported by the National Natural Science Foundation of China under Grant Nos. U2441204 and 12202255. Qi Liu would like to thank the support of the China Scholarship Council.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Gege Wang: Conceptualization (equal); Methodology (equal); Software (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Xiaolong Wang:** Methodology (equal); Supervision (equal); Visualization (equal); Writing – review & editing (equal). **Qi Liu:** Supervision (equal); Validation (equal); Writing – review & editing (equal). **Jürgen Kurths:** Conceptualization (equal); Supervision (equal); Writing – review & editing (equal). **Yong Xu:** Conceptualization (equal); Funding acquisition (equal); Methodology (equal); Supervision (equal); Validation (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] Q. Liu, Y. Xu, and Y. Li, "Complex dynamics of a conceptual airfoil structure with consideration of extreme flight conditions," Nonlinear Dyn. **111**, 14991–15010 (2023).
[2] Z. Huang and X. Jin, "Response and stability of a SDOF strongly nonlinear stochastic system with light damping modeled by a fractional derivative," J. Sound Vibr. **319**, 1121–1135 (2009).
[3] W. Zhu, Z. Huang, and Y. Suzuki, "Response and stability of strongly non-linear oscillators under wide-band random excitation," Int. J. Non-Linear Mech. **36**, 1235–1250 (2001).
[4] J.-B. Chen and J. Li, "Dynamic response and reliability analysis of non-linear stochastic structures," Probab. Eng. Mech. **20**, 33–44 (2005).
[5] J. Li and J. Chen, "Probability density evolution method for dynamic response analysis of structures with uncertain parameters," Comput. Mech. **34**, 400–409 (2004).
[6] P. Tankov, *Financial Modelling with Jump Processes* (Chapman & Hall/CRC, 2003).
[7] P. Hänggi, P. Talkner, and M. Borkovec, "Reaction-rate theory: Fifty years after Kramers," Rev. Mod. Phys. **62**, 251 (1990).
[8] C. Gardiner and P. Zoller, *Quantum Noise: A Handbook of Markovian and Non-Markovian Quantum Stochastic Methods with Applications to Quantum Optics* (Springer Science & Business Media, 2004).
[9] Q. Liu, H. Nakao, X. Wang, G. Li, X. Liu, and Y. Xu, "Reconstructing attractors of a conceptual airfoil system via next generation reservoir computing," AIAA J. **63**, 1349–1367 (2025).
[10] Q. Liu, Y. Xu, J. Kurths, and X. Liu, "Complex nonlinear dynamics and vibration suppression of conceptual airfoil models: A state-of-the-art overview," Chaos **32**, 062101 (2022).
[11] L. C. Evans, *An Introduction to Stochastic Differential Equations* (American Mathematical Society, 2012), Vol. 82.
[12] H. Risken and H. Risken, *Fokker–Planck Equation* (Springer, 1996).
[13] Y. K. Lin and G. Q. Cai, *Probabilistic Structural Dynamics: Advanced Theory and Applications* (McGraw-Hill, New York, 1995).
[14] J.-Q. Sun, *Stochastic Dynamics and Control* (Elsevier, 2006).
[15] J. Náprstek and R. Král, "Finite element method analysis of Fokker–Planck equation in stationary and evolutionary versions," Adv. Eng. Softw. **72**, 28–38 (2014).

[16]B. Sepehrian and M. K. Radpoor, "Numerical solution of non-linear Fokker–Planck equation using finite differences method and the cubic spline functions," Appl. Math. Comput. **262**, 187–190 (2015).
[17]Y. Xu, W. Zan, W. Jia, and J. Kurths, "Path integral solutions of the governing equation of sdes excited by Lévy white noise," J. Comput. Phys. **394**, 41–55 (2019).
[18]K. Kikuchi, M. Yoshida, T. Maekawa, and H. Watanabe, "Metropolis Monte Carlo method as a numerical technique to solve the Fokker–Planck equation," Chem. Phys. Lett. **185**, 335–338 (1991).
[19]X. Wang, J. Feng, Y. Xu, and J. Kurths, "Deep learning-based state prediction of the Lorenz system with control parameters," Chaos **34**, 033108 (2024).
[20]K. Tang, X. Wan, and Q. Liao, "Adaptive deep density approximation for Fokker–Planck equations," J. Comput. Phys. **457**, 111080 (2022).
[21]H. Zhang, Y. Xu, Q. Liu, and Y. Li, "Deep learning framework for solving Fokker–Planck equations with low-rank separation representation," Eng. Appl. Artif. Intell. **121**, 106036 (2023).
[22]Y. Xu, H. Zhang, Y. Li, K. Zhou, Q. Liu, and J. Kurths, "Solving Fokker–Planck equation using deep learning," Chaos **30**, 013133 (2020).
[23]H. Zhang, Y. Xu, Q. Liu, X. Wang, and Y. Li, "Solving Fokker–Planck equations using deep KD-tree with a small amount of data," Nonlinear Dyn. **108**, 4029–4043 (2022).
[24]H. Zhang, Y. Xu, Y. Li, and J. Kurths, "Statistical solution to SDEs with α-stable Lévy noise via deep neural network," Int. J. Dyn. Control **8**, 1129–1140 (2020).
[25]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
[26]H. W. Sorenson and D. L. Alspach, "Recursive Bayesian estimation using Gaussian sums," Automatica **7**, 465–479 (1971).
[27]D. Alspach and H. Sorenson, "Nonlinear Bayesian estimation using Gaussian sum approximations," IEEE Trans. Autom. Control **17**, 439–448 (1972).
[28]W. Sun, J. Feng, J. Su, and Y. Liang, "Data driven adaptive Gaussian mixture model for solving Fokker–Planck equation," Chaos **32**, 033131 (2022).
[29]W. Anderson and M. Farazmand, "Fisher information and shape-morphing modes for solving the Fokker–Planck equation in higher dimensions," Appl. Math. Comput. **467**, 128489 (2024).
[30]W. Ye, L. Chen, J. Qian, and J. Sun, "RBFNN for calculating the stationary response of SDOF nonlinear systems excited by Poisson white noise," Int. J. Struct. Stab. Dyn. **23**, 2350019 (2023).
[31]X. Wang, J. Jiang, L. Hong, L. Chen, and J.-Q. Sun, "On the optimal design of radial basis function neural networks for the analysis of nonlinear stochastic systems," Probab. Eng. Mech. **73**, 103470 (2023).
[32]K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," J. Big Data **3**, 9 (2016).
[33]S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Trans. Knowl. Data Eng. **22**, 1345–1359 (2010).
[34]X. Chen, C. Gong, Q. Wan, L. Deng, Y. Wan, Y. Liu, B. Chen, and J. Liu, "Transfer learning for deep neural network-based partial differential equations solving," Adv. Aerodyn. **3**, 36 (2021).
[35]S. Goswami, K. Kontolati, M. D. Shields, and G. E. Karniadakis, "Deep transfer operator learning for partial differential equations under conditional shift," Nat. Mach. Intell. **4**, 1155–1164 (2022).
[36]J. Zhai, M. Dobson, and Y. Li, "A deep learning method for solving Fokker–Planck equations," in *Mathematical and Scientific Machine Learning* (PMLR, 2022), pp. 568–597.
[37]R. C. Booton, "Nonlinear control systems with random inputs," IRE Trans. Circuit Theory **1**, 9–18 (1954).
[38]T. K. Caughey, "Equivalent linearization techniques," J. Acoust. Soc. Am. **35**, 1706–1711 (1963).
[39]M. Grigoriu, "Equivalent linearization for systems driven by Lévy white noise," Probab. Eng. Mech. **15**, 185–190 (2000).
[40]W. Zan, Y. Xu, J. Kurths, A. V. Chechkin, and R. Metzler, "Stochastic dynamics driven by combined Lévy–Gaussian noise: Fractional Fokker–Planck–Kolmogorov equation and solution," J. Phys. A **53**, 385001 (2020).
[41]Y. Xu, Y. Li, H. Zhang, X. Li, and J. Kurths, "The switch in a genetic toggle system with Lévy noise," Sci. Rep. **6**, 31505 (2016).