Probabilistic Engineering Mechanics 76 (2024) 103594

Contents lists available at ScienceDirect

Probabilistic Engineering Mechanics

journal homepage: www.elsevier.com/locate/probengmech

Separable Gaussian neural networks for high-dimensional nonlinear stochastic systems

Xi Wang [a], Siyuan Xing [b], Jun Jiang [a], Ling Hong [a], Jian-Qiao Sun [c,*]

[a] State Key Laboratory for Strength and Vibration, Xi'an Jiaotong University, Xi'an 710049, PR China
[b] Department of Mechanical Engineering, California Polytechnic State University, San Luis Obispo, CA 93047, USA
[c] Department of Mechanical Engineering School of Engineering, University of California, Merced, CA 95343, USA

ARTICLE INFO

Keywords:
Design of radial basis function neural networks
Nonlinear stochastic system
Stationary probability density function
Fokker–Planck–Kolmogorov equation

ABSTRACT

This paper extends the recently developed method of separable Gaussian neural networks (SGNN) to obtain solutions of the Fokker–Planck–Kolmogorov (FPK) equation in high-dimensional state space. Several challenges when extending SGNN to high-dimensional state space are addressed including proper definition of domain for placing Gaussian neurons and region for data sampling, and numerical integration issue of evaluating marginal probability density functions. Three benchmark nonlinear dynamic systems with increasing complexity and dimension are examined with the SGNN method. In particular, the steady-state probability density of the response is obtained with the SGNN method and compared with the results of extensive Monte Carlo simulations. It should be pointed out that some solutions of high-dimensional FPK equations for nonlinear dynamic systems would be very difficult to obtain without SGNN.

1. Introduction

Engineering structures and mechanical systems often experience random excitations including sea waves, airflow, and earthquakes [1]. The most important response to obtain is the probability density function (PDF), which plays a crucial role in stochastic analysis and reliability assessment. The response PDF of the stochastic nonlinear system is determined by the Fokker–Planck–Kolmogorov (FPK) equation [2,3]. However, analytical solutions of the FPK equation have been found only for linear systems and a few strictly conditioned nonlinear systems [4,5]. Numerical approximate solutions are often pursued in engineering applications.

Other approximate and numerical approaches for solving the FPK equation, including equivalent linearization [6–9], equivalent nonlinearization [10–12], stochastic averaging method [13,14], closure method [15], finite element method [16], finite difference method [17], path integral method [18–20], the exponential polynomial closure method for the stationary PDF of nonlinear systems [21], the state–space-split exponential polynomial closure method for high dimensional nonlinear systems [22], the generalized cell mapping method [23] and Monte Carlo simulation [24]. All these methods have various limitations when applied to study transient and stationary PDF of high-dimensional nonlinear stochastic systems due to the excessive demand on memory and CPU time needed to find the global solution of the PDF in the high-dimensional state space. The brutal force Monte Carlo simulation is the most general method for strongly nonlinear stochastic systems, and becomes computationally prohibitive for high-dimensional nonlinear stochastic systems, particularly when the small probability distribution in the tail region is needed for reliability assessment. This paper presents the separable Gaussian neural networks (SGNN) method which is promising for obtaining the PDF of high-dimensional nonlinear stochastic systems.

Since the 1990s, there has been a growing interest in adopting machine learning methods for solving partial differential equations including the FPK equation. Artificial neural networks (ANNs) have been employed as a universal approach for solving partial differential equations (PDEs) [25–29]. Cybenko demonstrated that an ANN employing continuous sigmoidal activation functions has the capacity to approximate continuous functions with arbitrary precision [30]. In recent years, driven by the rapid advancement of parallel computing powered by GPUs and automatic differentiation, the application of multi-layer ANNs, known as the deep learning, has gained popularity in obtaining solutions of high-dimensional problems [31–35].

Besides multi-layer ANNs, the single-layer radial-basis-function neural networks equipped with Gaussian neurons (GRBFNN) have also been validated as effective for analyzing both steady-state and transient responses [36,37], as well as for studying first passage problems [38]. The localized property of Gaussian functions gives the GRBFNN method an advantage in capturing highly complex distributions of stochastic

* Corresponding author.
E-mail address: jqsun@ucmerced.edu (J.-Q. Sun).

https://doi.org/10.1016/j.probengmech.2024.103594
Received 30 November 2023; Received in revised form 6 March 2024; Accepted 6 March 2024
Available online 11 March 2024

responses of strongly nonlinear systems. Unfortunately, this also results in the significant disadvantage of an exponential rise in the number of neurons as the system dimension increases. In [39], an iterative method is presented to optimally select the number and location of Gaussian neurons for GRBFNN. As the dimension of the system grows, the selection process itself can become time-consuming, while in the meantime, the number of optimal neurons also rises rapidly.

To improve the computational efficiency of GRBFNN while maintaining accuracy, Xing and Sun created a feedforward network called separable Gaussian neural networks (SGNN) [40]. SGNN treats the neurons of a uni-variate Gaussian function as if it were a hidden layer. The multi-layer forward propagation structure of SGNN, as compared to GRBFNN, requires fewer training parameters. It has been found that SGNN exhibits enhanced trainability and is more adaptable to tuning compared to multi-layer ANNs with ReLU and Sigmoid functions, and demonstrates superior function approximation capabilities through extensive numerical examples in high-dimensional space.

When SGNN is applied to high-dimensional systems, it also faces a number of challenges. This paper aims to develop algorithms to deal with these challenges. In particular, we develop an approach to treat the high-dimensional numerical integration when applying SGNN. Furthermore, to ensure that the solution generated by SGNN remains within the predefined boundaries, we impose constraints on the mean and standard deviation of Gaussian neurons. The loss function therefore consists of the sum of the squared residual of the FPK equation, the normalization condition, and the boundary condition.

The structure of the paper is as follows. In Section 2, a brief overview of the FPK equation is provided. Section 3 provides a detailed description of the SGNN method. The loss function and training process crucial for solving the FPK equation are also introduced. In Section 4, three examples of 2D, 4D and 6D systems are studied with the SGNN method. Monte Carlo simulations (MCS) will be used to assess the accuracy of the SGNN solution. The paper is concluded in Section 6.

2. Problem statement

Consider the following d-dimensional stochastic differential equation (SDE):

$$\frac{dX_j}{dt} = g_j(\mathbf{X}) + \sum_{k=1}^{m} h_{jk}(\mathbf{X})W_k(t), \quad 1 \le j \le d, \tag{1}$$

where $g_j(\mathbf{X})$ and $h_{jk}(\mathbf{X})$ are linear or nonlinear functions of $\mathbf{X}$. $W_j(t)$ are independent Gaussian white noises with zero mean and the correlation matrix $\mathrm{E}[W_j(t)W_j(t-\tau)] = 2D_j\delta(\tau)$ $(j = 1, 2, \ldots, m)$. The PDF $p(\mathbf{x})$ of the steady-state response satisfies the reduced FPK equation as follows,

$$\mathcal{L}_{FPK}[p(\mathbf{x})] \equiv -\sum_{i=1}^{d} \frac{\partial[m_i(\mathbf{x})p(\mathbf{x})]}{\partial x_i} + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial^2\left[b_{ij}(\mathbf{x})p(\mathbf{x})\right]}{\partial x_i \partial x_j} = 0, \tag{2}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T \in \mathbb{R}^d$ is a d-dimensional state vector. $\mathcal{L}_{FPK}[\cdot]$ denotes the FPK operator. $m_i(\mathbf{x})$ and $b_{ij}(\mathbf{x})$ are the drift and diffusion terms defined as follows,

$$m_i(\mathbf{x}) = g_i(\mathbf{x}) + \sum_{j=1}^{d}\sum_{k=1}^{m}\left[D_k h_{jk}(\mathbf{x})\frac{\partial h_{ik}(\mathbf{x})}{\partial x_j}\right], \tag{3}$$

$$b_{ij}(\mathbf{x}) = \sum_{k=1}^{m} 2D_k h_{ik}(\mathbf{x})h_{jk}(\mathbf{x}). \tag{4}$$

The PDF satisfies the following normalization condition,

$$\int_{\mathbb{R}^d} p(\mathbf{x})d\mathbf{x} = 1. \tag{5}$$

Since Eq. (2) is a homogeneous equation, the normalization offers a mechanism to avoid the trivial solution in numerical studies.

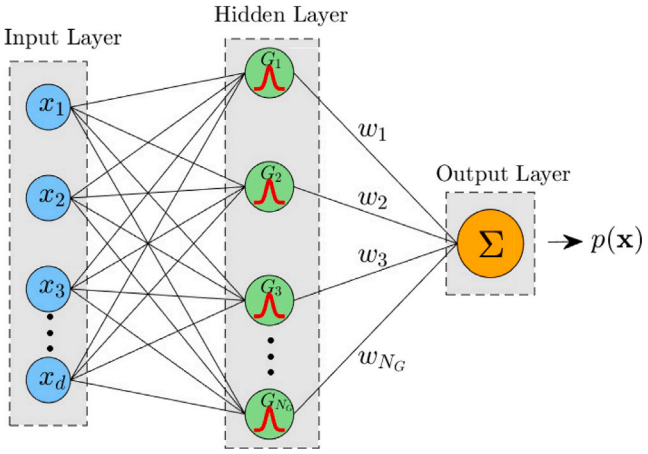

Fig. 1. The structure of RBFNN.

3. The SGNN method

In this section, we first review the RBFNN method for solving the reduced FPK equation of stochastic nonlinear systems. Next, a detailed description of the architecture of SGNN will be presented. Subsequently, the loss function for solving the reduced FPK equation and the training process will be provided. Finally, we will demonstrate how to obtain the marginal PDFs in the framework of SGNN by projecting of the PDF solution into an arbitrary subspace.

3.1. Review of the RBFNN method

The RBFNN method has been proven to be effective in the analysis of stochastic responses, according to a number of recent studies [36–38]. Assume that the trial solution for the stationary PDF takes the form of a weighted sum of Gaussian PDFs

$$\bar{p}(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{N_G} w_j G_j(\mathbf{x}), \tag{6}$$

where N_G is the number of neurons. $G_j(\mathbf{x}) \equiv G(\mathbf{x}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ is a Gaussian neuron with the mean $\boldsymbol{\mu}_j$ and covariance matrix $\boldsymbol{\Sigma}_j$ defined as

$$G(\mathbf{x}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma}_j)}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right]. \tag{7}$$

The RBFNN solution (6) can be regarded as a neural network with a single hidden layer using the Gaussian activation function, whose structure is shown in Fig. 1. This shallow neural network has been proven to possess universal approximation capabilities [41]. As we shall use the radial form of the Gaussian function, the covariance matrix of each neuron is assumed to be diagonal such that $\boldsymbol{\Sigma}_j = \mathrm{diag}[\sigma_{j,k}^2]$.

The Gaussian functions are chosen such that

$$\int_{\mathbb{R}^d} G_j(\mathbf{x})d\mathbf{x} = 1, \quad \text{for all } j, \tag{8}$$

the normalization condition for the RBFNN solution $\bar{p}(\mathbf{x}, \mathbf{w})$ is reduced to a constraint on the weights w_j as,

$$\sum_{j=1}^{N_G} w_j = 1. \tag{9}$$

We substitute the solution in Eqs. (6) to (2), and obtain the error of the equation as

$$e(\mathbf{x}, \mathbf{w}) = -\sum_{i=1}^{d} \frac{\partial}{\partial x_i}[m_i(\mathbf{x})\bar{p}(\mathbf{x}, \mathbf{w})] \tag{10}$$

$$+\sum_{i=1}^{d}\sum_{j=1}^{d}\frac{\partial^2}{\partial x_i\partial x_j}\left[\frac{b_{ij}(\mathbf{x})}{2}\bar{p}(\mathbf{x},\mathbf{w})\right]\equiv\sum_{k=1}^{N_G}s_k(\mathbf{x})w_k$$

where $\mathbf{w}=[w_1,w_2,\ldots,w_{N_G}]^T$, and

$$s_k(\mathbf{x})=-\sum_{i=1}^{d}\frac{\partial}{\partial x_i}[m_i(\mathbf{x})G_k(\mathbf{x})]+\sum_{i=1}^{d}\sum_{j=1}^{d}\frac{\partial^2}{\partial x_i\partial x_j}\left[\frac{b_{ij}(\mathbf{x})}{2}G_k(\mathbf{x})\right]. \tag{11}$$

To obtain the solution to the FPK equation that satisfies the normalization condition (9), we introduce a Lagrange multiplier λ and the total loss function is defined as,

$$J(\mathbf{w},\lambda)=\frac{1}{2}\sum_{i=1}^{N_s}e^2(\mathbf{x}_i,\mathbf{w})+\lambda\left(\sum_{j=1}^{N_G}w_j-1\right), \tag{12}$$

where N_s is the number of sampled points $\mathbf{x}_i$. The expression (12) serves as the loss function, which can be minimized with respect to the training parameters for the RBFNN.

In some previous studies on the RBFNN method for solving FPK equations, the means and standard deviations are kept as constants [36–38]. Usually, the domain of interest is discretized into uniform grids. The grids are the mean locations of Gaussian neurons while the standard deviation is set equal to the grid size. When the means $\boldsymbol{\mu}$ and standard deviations $\boldsymbol{\Sigma}$ are fixed, the minimization of the loss function (12) leads to a set of linear algebraic equations to determine the optimal weight coefficients $\mathbf{w}$.

The means and standard deviations can be considered as trainable parameters. Discretization of the computational domain gives initial values of means and standard deviations. They are updated in the gradient descent search for minimal loss function (12). Since the RBFNN method works with the discretized computational domain, it becomes computationally intensive for higher-dimensional systems.

3.2. Architecture of SGNN

The d-variate Gaussian radial basis function is separable and can be expressed as a product of multiple uni-variate Gaussian functions as,

$$G(\mathbf{x},\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)=\prod_{k=1}^{d}g(x_k,\mu_{j,k},\sigma_{j,k}), \tag{13}$$

$$g(x_k,\mu_{j,k},\sigma_{j,k})=\frac{1}{\sqrt{2\pi}\sigma_{j,k}}\exp\left[-\frac{1}{2\sigma_{j,k}^2}(x_k-\mu_{j,k})^2\right], \tag{14}$$

where $\mu_{j,k}$ is the kth component of the vector $\boldsymbol{\mu}_j$ and $\sigma_{j,k}$ is the kth component of the diagonal matrix $\boldsymbol{\Sigma}_j$. This chain of multiplications in Eq. (13) is viewed as the forward propagation network with d layers, each consisting of a single uni-variate Gaussian neuron of one variable x_k. The structure of the separable Gaussian neural network (SGNN) is shown in Fig. 2.

Let $\bar{p}(\mathbf{x},\mathbf{w})$ denote the output of SGNN, i.e. the PDF solution of the FPK equation. $H^k=[h_1^k,h_2^k,\ldots,h_{m_k}^k]$ represents the output of the kth hidden layer and m_k is the number of neurons in the kth hidden layer. $\mathbf{W}^k=[w_{ij}^k]$ where w_{ij}^k represents the weight from the ith neuron of kth layer to the jth neuron of the $(k+1)$th layer. Then, the output can be written in the feedforward manner as,

$$h_j^1=g(x_1,\mu_{j,1},\sigma_{j,1}) \tag{15}$$

$$h_j^k=\left[\sum_{i=1}^{m_{k-1}}w_{ij}^{k-1}h_i^{k-1}\right]g(x_k,\mu_{j,k},\sigma_{j,k}),\quad k=2,\ldots,d \tag{16}$$

$$\bar{p}(\mathbf{x},\mathbf{w})=\sum_{i=1}^{m_d}w_{i1}^d h_i^{k-1}=\left[\sum_{i_1=1}^{m_1}\sum_{i_2=1}^{m_2}\cdots\sum_{i_d=1}^{m_d}w_{i_d1}^d\left(\prod_{k=1}^{d-1}w_{i_ki_{k+1}}^k\right)\right]\prod_{k=1}^{d}g(x_k,\mu_{i_k,k},\sigma_{i_k,k})$$

$$=\sum_{j=1}^{N_G}\hat{w}_jG(\mathbf{x},\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j), \tag{17}$$

where $N_G=\prod_{k=1}^{d}m_k$, $\boldsymbol{\mu}_j=[\mu_{i_k,k}]$, $\boldsymbol{\Sigma}_j=[\sigma_{i_k,k}^2]$

$$j=\sum_{k=1}^{d-1}(i_k-1)\prod_{l=k+1}^{d}m_l+i_d,\quad 1\le j\le N_G, \tag{18}$$

$$G(\mathbf{x},\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)=\prod_{k=1}^{d}g(x_k,\mu_{i_k,k},\sigma_{i_k,k}), \tag{19}$$

$$\hat{w}_j=\sum_{i_1=1}^{m_1}\sum_{i_2=1}^{m_2}\cdots\sum_{i_d=1}^{m_d}w_{i_d1}^d\left(\prod_{k=1}^{d-1}w_{i_ki_{k+1}}^k\right). \tag{20}$$

The output of SGNN shares the same form as that of GRBFNN in Eq. (6), i.e. a weighted sum of Gaussian neurons. In fact, any solution of SGNN can be found to be an equivalent solution of GRBFNN. However, the reverse is not true. For the details of the mathematical properties of SGNN, the reader is referred to [40].

3.3. The SGNN solution of FPK equation

Recall that the normalization condition (5) of PDFs is defined in $\mathbf{R}^d$. In computations, the domain of interest D is chosen to be finite and large enough such that the normalization condition is replaced with the integration in D,

$$\int_D\bar{p}(\mathbf{x},\mathbf{w})d\mathbf{x}=1. \tag{21}$$

In addition, an artificial boundary condition can be imposed to avoid the loss of probability out of the finite domain D, which means

$$\bar{p}(\mathbf{x},\mathbf{w})=0,\forall\mathbf{x}\notin D \tag{22}$$

Consider a finite rectangular domain $D=[a_1,b_1]\times[a_2,b_2]\cdots[a_d,b_d]$. Impose the artificial boundary condition to the SGNN solution. We have

$$\sum_{j=1}^{N_G}\hat{w}_jG(\mathbf{x},\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)=0,\forall\mathbf{x}\notin D. \tag{23}$$

A more strict sufficient condition for the artificial boundary condition reads

$$G(\mathbf{x},\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)=0,\forall\mathbf{x}\notin D,1\le j\le N_G. \tag{24}$$

Consider a d dimensional problem and define the so-called $3-\sigma$ domain such that over 99.7% probability of the Gaussian function is contained in the domain denoted as D_j.

$$D_j=\{\mathbf{x}\mid\|x_k-\mu_{j,k}\|\le3\sigma_{j,k},1\le k\le d\}=[\mu_{j,1}-3\sigma_{j,1},\mu_{j,1}+3\sigma_{j,1}]\times\cdots\times[\mu_{j,d}-3\sigma_{j,d},\mu_{j,d}+3\sigma_{j,d}]. \tag{25}$$

Let $D_G=\cup_jD_j$. We make sure that $D_j\subseteq D$ for all j and $D_G\subseteq D$. Hence, we have

$$\left\|\mu_{j,k}-\frac{a_k+b_k}{2}\right\|+3\sigma_{j,k}\le\frac{b_k-a_k}{2},\ 1\le j\le m_k,\ 1\le k\le d. \tag{26}$$

These conditions make the boundary condition (22) automatically satisfied. When the condition (26) holds, we have

$$\int_D G(\mathbf{x},\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)d\mathbf{x}=1,\ 1\le j\le N_G. \tag{27}$$

Hence, the normalization in terms of the coefficients reads

$$\sum_{j=1}^{N_G}\hat{w}_j=\sum_{i_1=1}^{m_1}\sum_{i_2=1}^{m_2}\cdots\sum_{i_d=1}^{m_d}w_{i_d1}^d\left(\prod_{k=1}^{d-1}w_{i_ki_{k+1}}^k\right)=1. \tag{28}$$

This is to compare with Eq. (9).

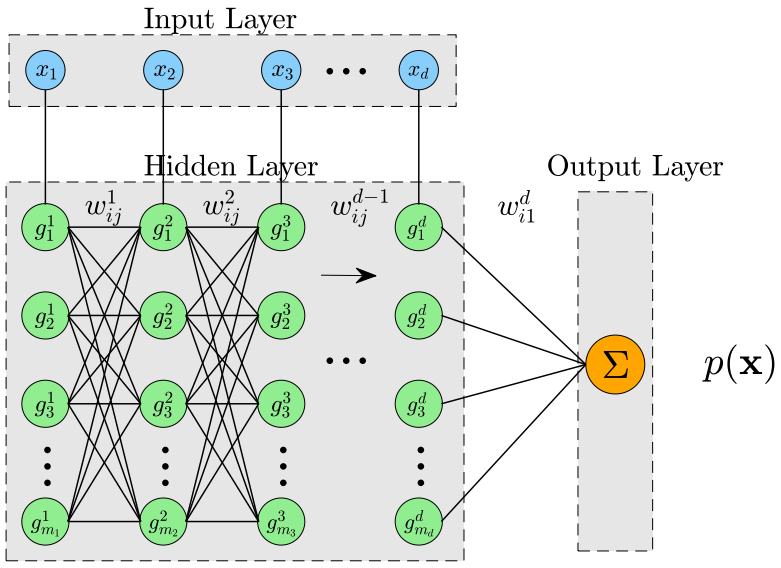

Fig. 2. The structure of an SGNN with d hidden layers. $\mathbf{x} = [x_1, x_2, \dots, x_d]^T$ is the network input. m_k $(k = 1, 2, \dots, d)$ represent the number of neurons in the kth hidden layer. w_{ij}^k $(k = 1, 2, \dots, d-1)$ denote the weights between the kth and $(k+1)$th hidden layer. w_{ij}^d denote the weights between the last hidden layer and the output layer. $g_i^k(x_k) = g(x_k, \mu_{i,k}, \sigma_{i,k})$ where $k = 1, 2, \dots, d$ and $i = 1, 2, \dots, m_k$, which represent the Gaussian activation function of the ith neuron in the kth hidden layer.

3.4. Loss function

The loss function for solving the FPK equation by the SGNN is defined as,

$$\mathcal{L}^S(\boldsymbol{\theta}) = \lambda_1 E_1^S + \lambda_2 E_2^S + \lambda_3 E_3^S, \tag{29}$$

where

$$E_1^S = \frac{1}{N_s} \sum_{i=1}^{N_s} \left| \mathcal{L}_{FPK}\left(\tilde{p}(\mathbf{x}_i, \mathbf{w})\right) \right|^2, \quad \mathbf{x}_i \in D, \tag{30}$$

$$E_2^S = \left| \sum_{\forall j} \hat{w}_j - 1 \right|^2, \tag{31}$$

$$E_3^S = \sum_{k=1}^{d} \sum_{j=1}^{m_k} \text{ReLU}\left(\left\| \mu_{j,k} - \frac{a_k + b_k}{2} \right\| + 3\sigma_{j,k} - \frac{b_k - a_k}{2} \right)^2, \tag{32}$$

where E_1^S, E_2^S, and E_3^S represent the losses of the SGNN solution $\tilde{p}(\mathbf{x}, \mathbf{w})$ with respect to the FPK equation, the normalization condition and the boundary condition, respectively. λ_i $(i = 1, 2, 3)$ are the weights to be determined. $\boldsymbol{\theta} = [w_{ij}^k; \mu_{i,k}; \sigma_{i,k}]$ denote all the trainable parameters.

Later in the paper, we somewhat arbitrarily choose $\lambda_1 = 10$, $\lambda_2 = 1$, and $\lambda_3 \in [1, 20]$. These weights are certainly not optimal. To find optimal values for the weights is an optimization problem, which is a topic for another paper.

3.5. Computational issues

We apply the stochastic gradient descent algorithm Adam with a learning rate $\alpha = 10^{-3}$ to train SGNN with mini-batches. This selection of the learning rate for the Adam algorithm is common in machine learning. It strikes a balance between computational accuracy and speed of convergence, and has been validated in diverse applications. It works because the Adam algorithm adaptively adjusts the learning rate in iterations based on momentum and second raw moment, making the learning less sensitive to the initial value of the learning rate.

We have found that the mini-batch approach is quite effective with SGNN for the FPK equation. The initial weights w_{ij}^k are randomly generated following a standard Gaussian distribution with zero mean and unit variance. This is a common choice and has been proven helpful for searching global optimal solutions in many numerical experiments.

To ensure good convergence of SGNN, the initial means and standard deviations of Gaussian neurons should satisfy the boundary condition (26). The initial means are uniformly distributed along each dimension of the domain $D_G \subset D$. The initial standard deviations are set as the distance between two adjacent centers of Gaussian neurons. The Latin hypercube sampling (LHS) method is employed to perform sampling over the domain $D_s = D$ [42].

Algorithm 1 shows the process of the proposed SGNN method for solving the FPK equation.

Algorithm 1 The SGNN method for the FPK equation.

Require: The FPK equation $\mathcal{L}_{FPK}[p(\mathbf{x}, t|\mathbf{x}_0, t_0)]$, the domain of interest D, the number of neurons m_k in each layer, the number of sampling points N_s, the size of mini-batch N_b, maximum number of iterations $\mathcal{M}$,

Ensure: The probability density function $p(\mathbf{x})$

```
Initialize SGNN undetermined parameters θ
Sample N_s points in D
Divide the set of sampling points into N_s/N_b mini-batches
for iteration i = 1 to M do
    for iteration j = 1 to N_s/N_b do
        Take the jth mini-batch to calculate the loss function L^S
        and the gradient ∂L^S/∂θ
        Update θ by the Adam algorithm
    end for
end for
```

3.6. Marginal probability density from the SGNN solution

We introduce a systematic way to compute marginal probability density functions for arbitrary sub-set of state variables from the SGNN solution.

Let $\mathbf{k} = \{k_1, k_2, \dots, k_n\}$ denote the set of indices such that $n < d$ and $k_1 < k_2 \dots < k_n$. We project the PDF solution to any n-dimensional subspace $\mathbf{x}_n = [x_{k_1}, x_{k_2}, \cdots, x_{k_n}]$. Let $\mathbf{r} = [x_i], i \notin \mathbf{k}$ be a $(d-n)$-dimensional vector. Rewrite the d-dimensional PDF solution as $p(\mathbf{x}) = p(\mathbf{x}_n, \mathbf{r})$.

The projection $p^{\mathbf{k}}(\mathbf{x}_n)$ is calculated as

$$p^{\mathbf{k}}(\mathbf{x}_n) = \int_{\mathbb{R}^{d-n}} p(\mathbf{x}_n, \mathbf{r}) d\mathbf{r} \tag{33}$$

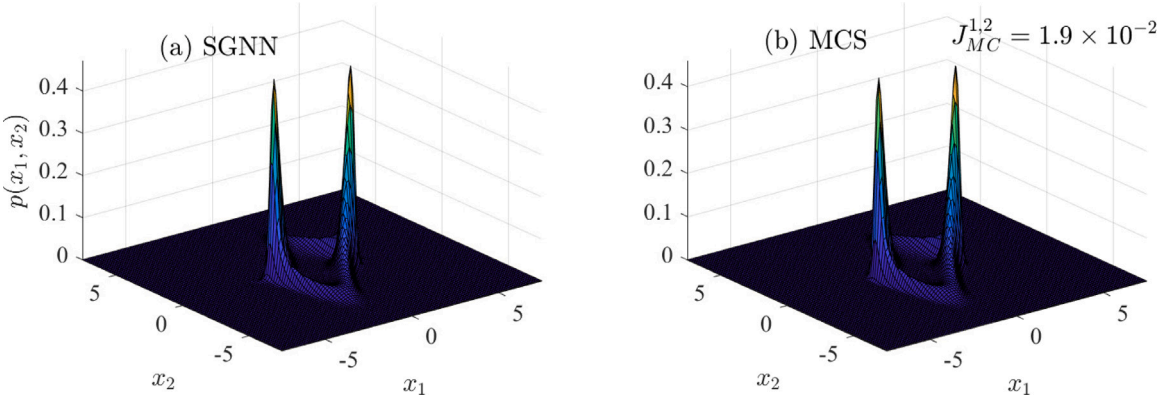

Fig. 3. The comparison of the PDF solution obtained by the SGNN method and MCS of the 2D Van der Pol system. The error as defined in Eq. (40) is $J_{MC}^{1,2} = 1.9 \times 10^{-2}$.

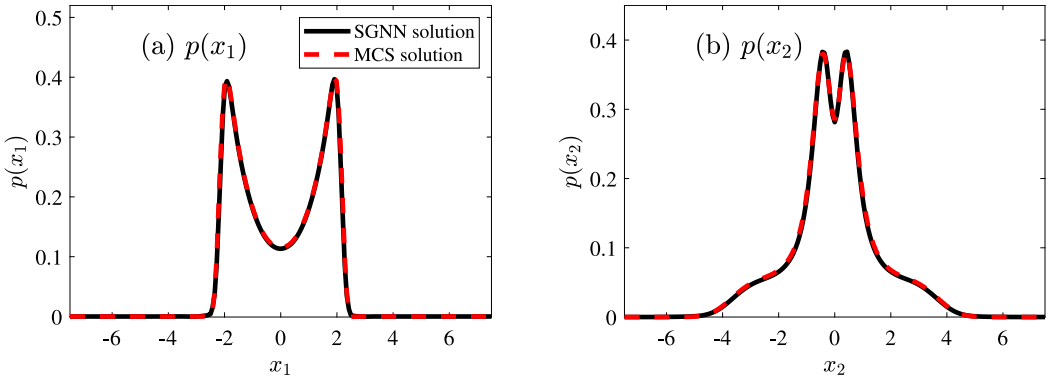

Fig. 4. The marginal PDFs of $p(x_1)$ and $p(x_2)$ of the 2D Van der Pol system.

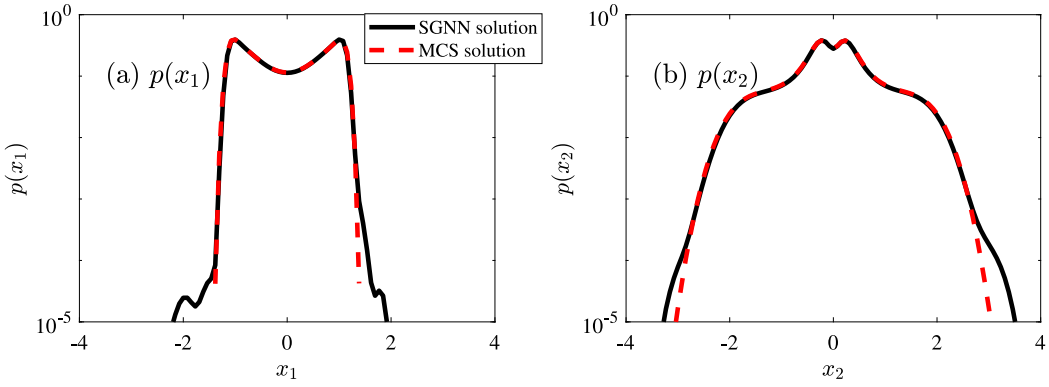

Fig. 5. The semi-logarithmic plots of marginal PDFs $p(x_1)$ and $p(x_2)$ of the 2D Van der Pol system.

This numerical integration in high-dimensional state space is time-consuming. However, in the proposed method, it can be computed analytically by taking advantage of the radial form of the Gaussian functions in SGNN. We can show that

$$p^{\mathbf{k}}(\mathbf{x}_n) = \sum_{i_1=1}^{m_1}\sum_{i_2=1}^{m_2}\cdots\sum_{i_d=1}^{m_d} w_{i_d 1}^{d}\left(\prod_{k=1}^{d-1} w_{i_k i_{k+1}}^{k}\right)\prod_{l=1}^{n} g(x_{k_l}, \mu_{i_{k_l},k_l}, \sigma_{i_{k_l},k_l}) = \sum \bar{\mathbf{W}}_{\mathbf{k}} \cdot \bar{\mathbf{G}}_{\mathbf{k}} \tag{34}$$

where $\cdot$ denotes the inner product, $\bar{\mathbf{W}}_{\mathbf{k}}$ and $\bar{\mathbf{G}}_{\mathbf{k}}$ are nth order tensors with the indices $[m_{k_1}, m_{k_2}, \ldots, m_{k_n}]$ defined as,

$$\bar{\mathbf{W}}_{\mathbf{k}}(i_{k_1}, \ldots, i_{k_n}) = \sum_{i_1=1}^{m_1}\cdots\sum_{i_{k_1-1}=1}^{m_{k_1-1}}\sum_{i_{k_1+1}=1}^{m_{k_1+1}}\cdots\sum_{i_d=1}^{m_d} w_{i_d 1}^{d}\left(\prod_{k=1}^{d-1} w_{i_k i_{k+1}}^{k}\right) \tag{35}$$

$$\bar{\mathbf{G}}_{\mathbf{k}}(i_{k_1}, \ldots, i_{k_n}) = \left[\prod_{l=1}^{n} g(x_{k_l}, \mu_{i_{k_l},k_l}, \sigma_{i_{k_l},k_l})\right] \tag{36}$$

Introduce a matrix notation $\mathbf{W}^{i,j} \in \mathbb{R}^{m_i \times m_j}$ as the product of a series of the weight matrices of the connecting layers,

$$\mathbf{W}^{i,j} = \begin{cases} \mathbf{W}^i \mathbf{W}^{i+1} \ldots \mathbf{W}^j & \text{if } i \le j \\ 1 & \text{if } i > j \end{cases} \tag{37}$$

Thus, the tensor $\bar{\mathbf{W}}_{\mathbf{k}}$ can be expressed in terms of $\mathbf{W}^{i,j}$,

$$\bar{\mathbf{W}}_{\mathbf{k}}(i_{k_1}, \ldots, i_{k_n}) = \left[\sum_{i_1=1}^{m_1} w_{i_1,i_{k_1}}^{1,k_1}\right]\prod_{l=2}^{n} w_{i_{k_l},i_{k_{l+1}}}^{k_l,k_{l+1}}\left[\sum_{i_d=1}^{m_d} w_{i_{k_n},i_d}^{k_n,d}\right]. \tag{38}$$

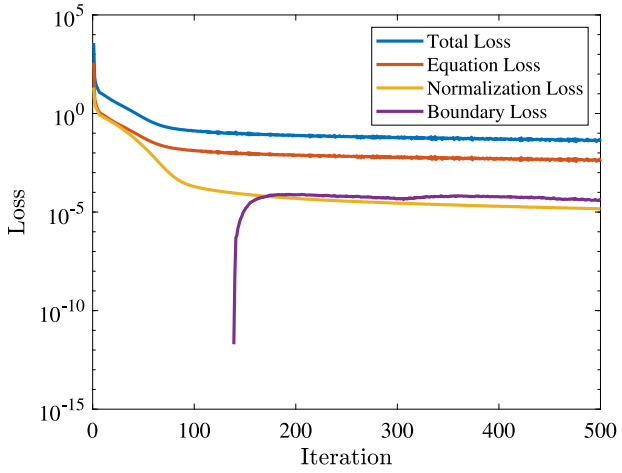

Fig. 6. The loss function of the 2D Van der Pol system.

4. Examples

In this section, three examples with distinct features are presented to demonstrate the versatility and efficiency of the SGNN method in solving the FPK equation of nonlinear stochastic systems, especially for high-dimensional systems. The examples include systems in 2, 4 and 6 dimensional state space.

Two distinct types of errors are defined as measures to evaluate the accuracy of the SGNN solution. The first is the root mean square error of the FPK equation given by:

$$J_{FPK} = \sqrt{\int_{\mathbb{R}^d} |\mathcal{L}_{FPK}(\bar{p}(\mathbf{x}))|^2 d\mathbf{x}}. \tag{39}$$

Note that we have dropped the dependence of the PDF on θ for brevity from now on.

The joint PDFs obtained by SGNN are compared with the results from MCS. For the sake of easy illustration, we consider second order joint PDFs $p(x_{k_1}, x_{k_2})$. The difference of the joint PDFs obtained by SGNN and the MCS results is taken as the second error.

$$J_{MC}^{k_1,k_2} = \sqrt{\int_{\mathbb{R}^2} |\bar{p}(x_{k_1}, x_{k_2}) - p_{MC}(x_{k_1}, x_{k_2})|^2 dx_{k_1} dx_{k_2}} \tag{40}$$

where $p_{MC}(x_{k_1}, x_{k_2})$ denotes the PDF computed from MCS. This error is more appropriate for low-dimensional problems because the needed sample size by MCS is manageable.

4.1. Van der Pol system

The Van der Pol system, which arises in various fields of physics and engineering, resides in a 2D state space. The governing equation of a strongly nonlinear SDOF Van der Pol system excited by Gaussian white noise reads,

$$\begin{aligned} \frac{dX_1}{dt} &= X_2, \\ \frac{dX_2}{dt} &= -\beta(X_1^2 - 1)X_2 - X_1 + W_1(t), \end{aligned} \tag{41}$$

where $W_1(t)$ represents the Gaussian white noise excitation with intensity $2D_1$. The drift and diffusion terms of the reduced FPK equation are

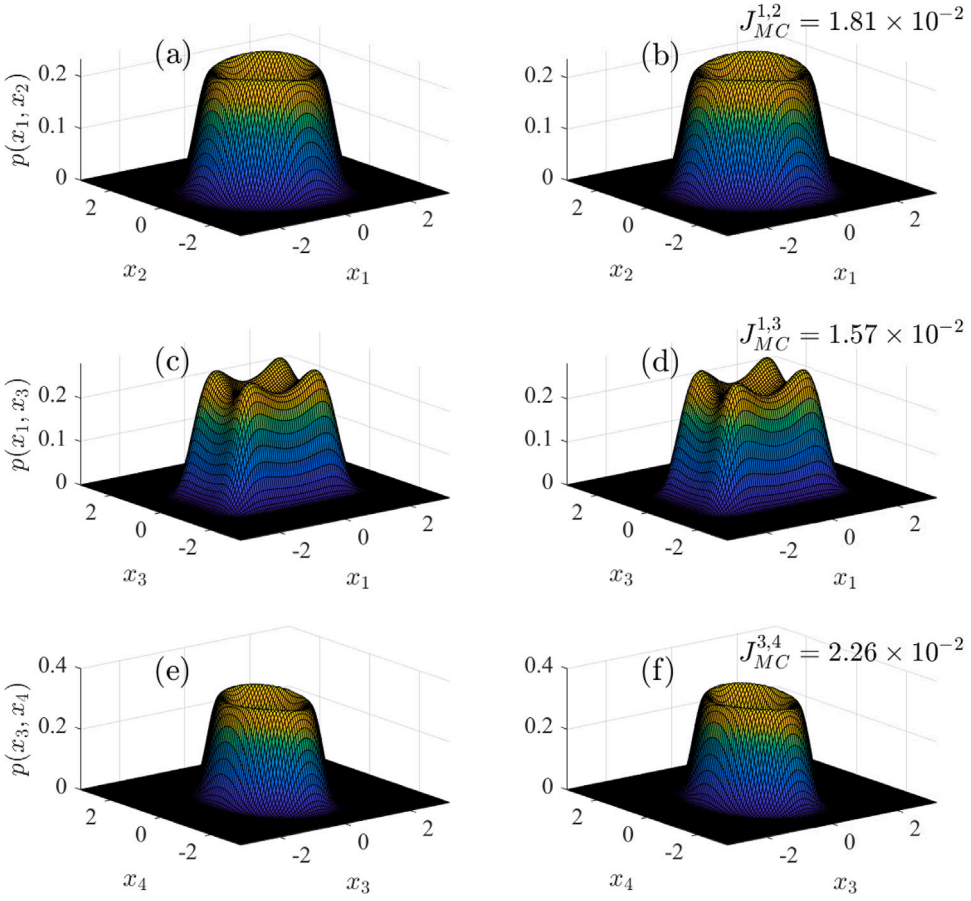

Fig. 7. The marginal steady-state joint PDFs of system (43). First row: $x_1 - x_2$ sub-space. Second row: $x_1 - x_3$. Third row: $x_3 - x_4$. Left column: The SGNN results. Right column: MCS results.

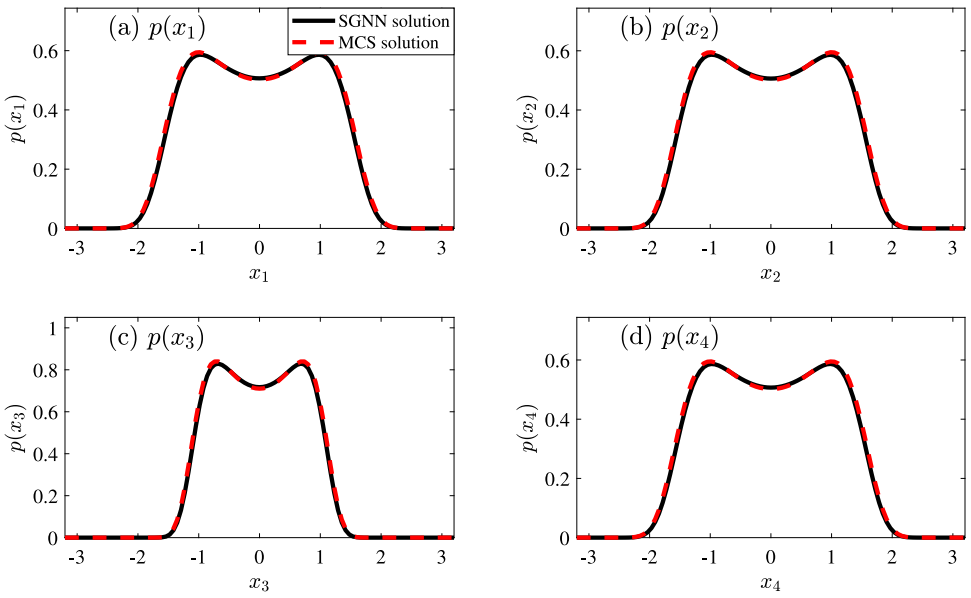

Fig. 8. The comparison of the single-variant marginal PDF solutions obtained by the SGNN method and MCS of system (43). Black solid line: SGNN. Red dashed line: MCS. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

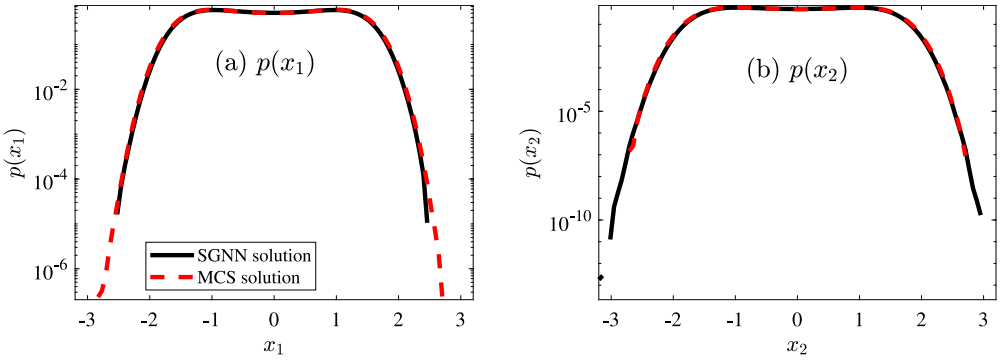

Fig. 9. The semi-logarithmic plots of marginal PDFs $p(x_1)$ and $p(x_2)$ of the 2-DOF nonlinear system.

shown as follows,

$$A_1 = x_2, \quad B_{22} = 2D_1,$$
$$A_2 = -\beta(x_1^2 - 1)x_2 - x_1. \tag{42}$$

This strongly nonlinear Van der Pol system is often used as a benchmark to examine the effectiveness of solution methods due to its typical difficulty in the study of stochastic analysis. The analytical solution of the reduced FPK equation of the Van der Pol system (41) has not been reported. The RBFNN method has been demonstrated to be effective in obtaining the steady-state PDF of this strongly nonlinear Van der Pol system [36]. The number of neurons required by the RBFNN method to compute the accurate solution is reported to be 10 201 in the early study [36], and is reduced to 2812 with an improved algorithm [39].

Let $\beta = 1.5$ and $D_1 = 0.25$. The region of initial centers is selected to be $D_G = [-6, 6] \times [-6, 6]$. The region of sampling points is $D_s = [-7.5, 7.5] \times [-7.5, 7.5]$. We set the weights in the loss function as $\lambda_1 = 10$, $\lambda_2 = 1$ and $\lambda_3 = 1$. For this 2D system, the neural networks have two hidden layers and 61 neurons in each hidden layer. The initial centers are evenly distributed in each dimension and initial widths are set as the distance between two adjacent centers, resulting in a total of $2 \times 61 = 122$ neurons, significantly fewer than the 2812 neurons reported before. The number of data points of the training set is $N_s = 8192 \times 64$. The mini-batch size for training is $N_b = 1024$.

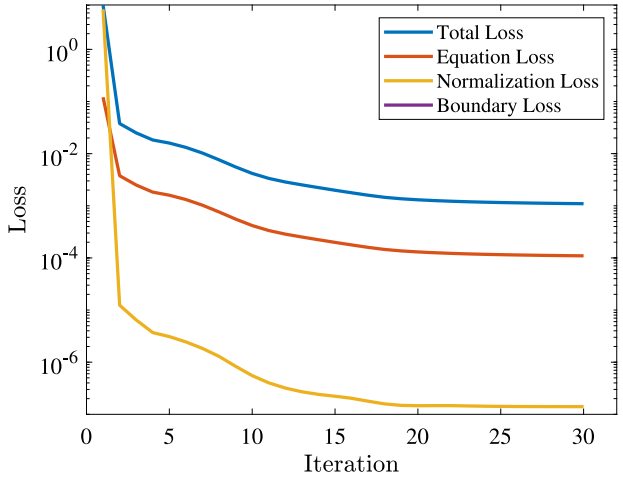

Fig. 10. The variation of the loss function of the 4D system (43) during training. In the training process, the boundary condition is always satisfied, resulting in the boundary loss being zero, which is not shown in the figure.

Fig. 3 shows the comparison between the SGNN solution obtained after 500 iteration with the MCS solution with 10^8 samples. Errors for

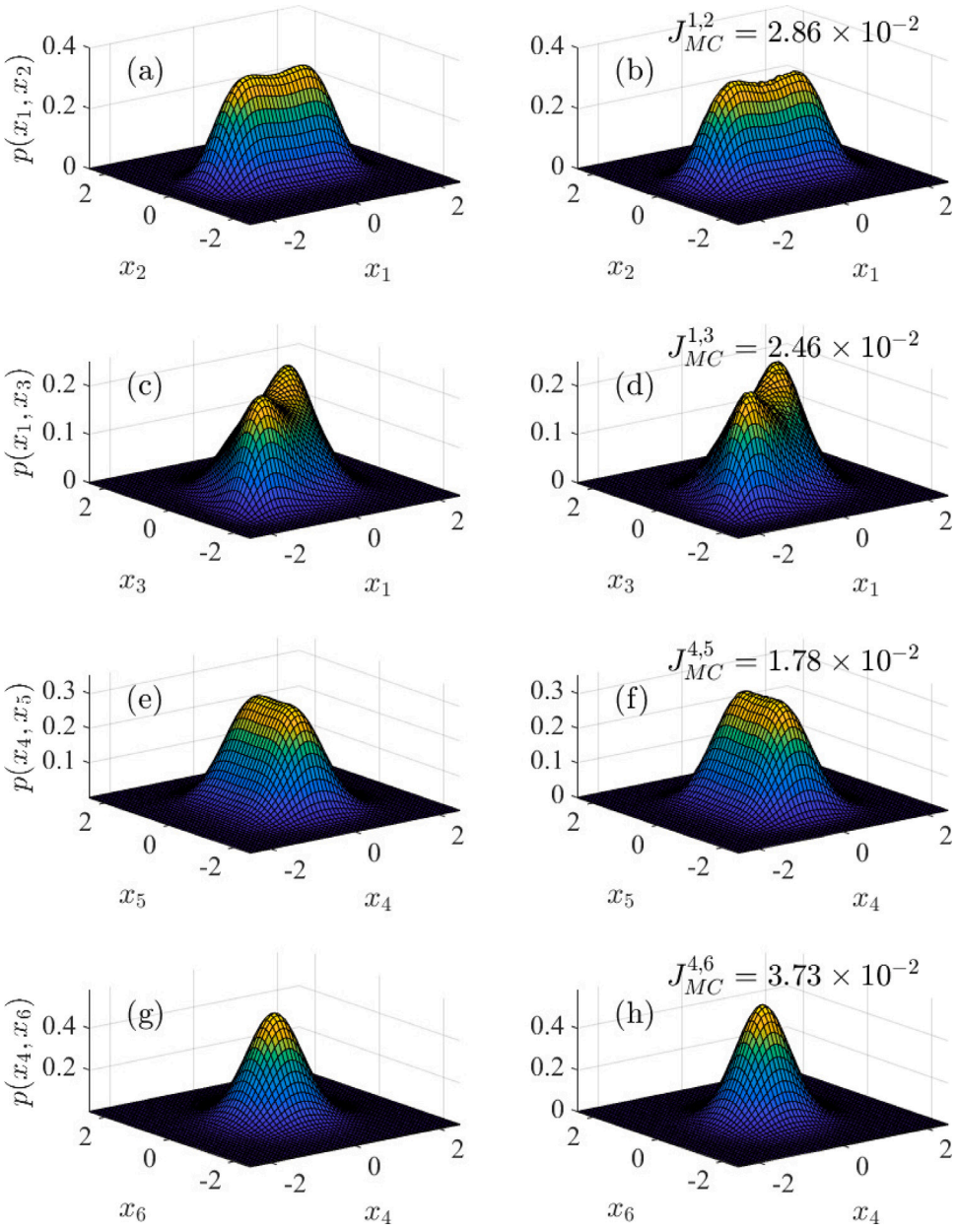

Fig. 11. The marginal steady-state joint PDFs of system (45). First row: $x_1 - x_2$ sub-space. Second row: $x_1 - x_3$. Third row: $x_4 - x_5$. Fourth row: $x_4 - x_6$. Left column: The SGNN results. Right column: MCS results.

the FPK equation, normalization condition, and boundary condition are 3.7233×10^{-3}, 1.4702×10^{-5}, and 3.9916×10^{-5}, respectively. The RMS error between joint PDF $p(x_1, x_2)$ obtained by SGNN and MCS are 1.9×10^{-2}. The good agreement between SGNN results and MCS results speaks highly the effectiveness of this method.

Fig. 4 shows the marginal probability densities $p(x_1)$ and $p(x_2)$ of the 2D Van der Pol system. Fig. 5 shows the same results in semi-logarithmic scale. The SGNN solution and simulation results agree well in the small probability region. We should point out that the current settings of the SGNN method for this and other examples are focused on the global solution of the PDF in the entire state space. As a result, the SGNN solution in the high probability region tends to agree with the simulation results better than the solution in the small probability region where both the SGNN solution and the simulation results can have higher variance error. Should the focus is on the small probability in the tail region, different sampling strategies for SGNN and simulation can be explored to improve the accuracy. This is a topic for another study.

Fig. 6 shows the loss during the training process. Throughout the training process, the boundary conditions remain consistently satisfied, while the normalization condition and the FPK equation gradually converge. The total training time for 500 iterations is 6020 s. However, it is significant to note that the loss function converges after 100 iterations within 1223 s. In contrast, the MCS solution with 10^8 samples requires 1400 s to complete.

4.2. A 4D nonlinear system

As the second example, we consider a two-DOF nonlinear system under Gaussian white noises.

$$\begin{aligned} \frac{dX_1}{dt} &= X_2, \\ \frac{dX_2}{dt} &= -(-\alpha_{11} + \alpha_{12} X_2^2) X_2 - \omega_1^2 X_1 - \beta_1 X_4 + W_1(t), \\ \frac{dX_3}{dt} &= X_4, \end{aligned} \tag{43}$$

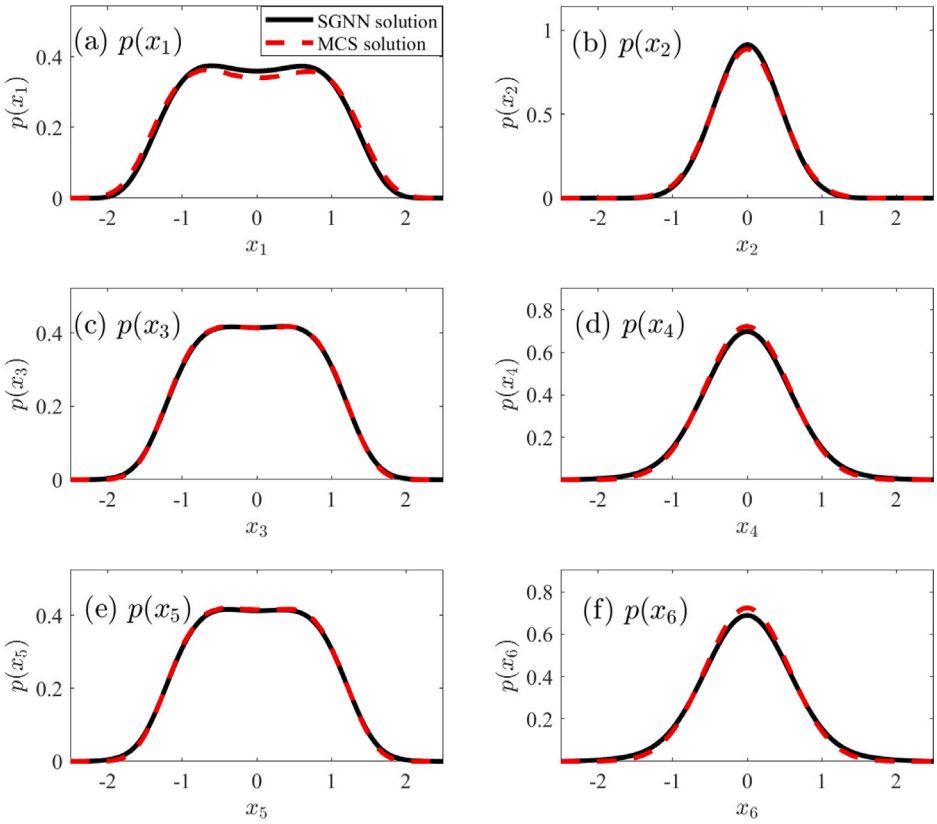

Fig. 12. The comparison of the single-variant marginal PDF solutions obtained by the SGNN method and MCS of system (45). Black solid line: SGNN. Red dashed line: MCS. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

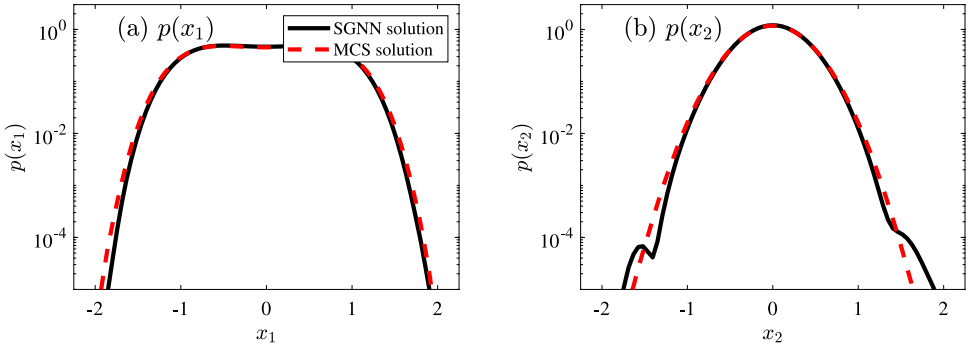

Fig. 13. The semi-logarithmic plots of marginal PDFs $p(x_1)$ and $p(x_2)$ of system (45).

$$\frac{dX_4}{dt} = -(-\alpha_{21} + \alpha_{22}X_4^2)X_4 - \omega_2^2 X_3 + \beta_2 X_2 + W_2(t).$$

where $\omega_1 = 1.0$, $\omega_2 = \sqrt{2}$, $\alpha_{11} = \alpha_{21} = -0.06$, $\alpha_{12} = \alpha_{22} = 0.08$ and $\beta_1 = \beta_2 = 0.01$. $W_1(t)$ and $W_2(t)$ are independent Gaussian white noise excitations with intensity $D_1 = D_2 = 0.1$. The drift and diffusion terms of the reduced FPK equation are given by,

$$\begin{aligned} &A_1 = x_2, \quad A_3 = x_4, \\ &A_2 = -(-\alpha_{11} + \alpha_{12}x_2^2)x_2 - \omega_1^2 x_1 - \beta_1 x_4, \\ &A_4 = -(-\alpha_{21} + \alpha_{22}x_4^2)x_4 - \omega_2^2 x_3 + \beta_2 x_2, \\ &B_{22} = 2D_1, \quad B_{44} = 2D_2. \end{aligned} \tag{44}$$

The domain where the Gaussian neurons reside is $D_G = [-1.6, 1.6]^4$. The region for sampling points is $D_s = [-2.4, 2.4]^4$. The weights in the loss function are chosen as $\lambda_1 = 10$, $\lambda_2 = 1$ and $\lambda_3 = 10$. The SGNN has 4 hidden layers and 41 neurons in each hidden layer, leading to a total of $4 \times 41 = 164$ neurons. The number of trainable parameters is $2 \times 4 \times 41 + (4-1) \times 41^2 = 5371$. The total number of training data is $N_s = 8192 \times 1024$. The mini-batch size is $N_b = 1024 \times 8$. Thus, there are 1024 updates of training parameters in each iterations.

The steady-state PDF solutions projected into different sub-spaces of the 4D stochastic system (43) obtained by the SGNN method after 30 iterations are shown in Fig. 7 together with the MCS results obtained with 10^9 samples. The RMS error of the FPK equation is 1.04×10^{-4}. The RMS errors defined in Eq. (40) of joint PDFs $p(x_1, x_2)$, $p(x_1, x_3)$ and $p(x_3, x_4)$ are 1.81×10^{-2}, 1.57×10^{-2} and 2.26×10^{-2}, respectively.

Fig. 8 shows the marginal probability densities of $p(x_1)$, $p(x_2)$, $p(x_3)$ and $p(x_4)$. Fig. 9 shows the same results of $p(x_1)$ and $p(x_2)$ in semi-logarithmic scale. The results by the SGNN and MCS agree well in the region of small probabilities.

The SGNN solution closely matches the MCS results demonstrating the effectiveness of this method. The variation of the loss function during network training is shown in Fig. 10. The total training time for 30 iterations is 7626 s. In contrast, the MCS solution with 10^9 samples requires 14 605 s to complete.

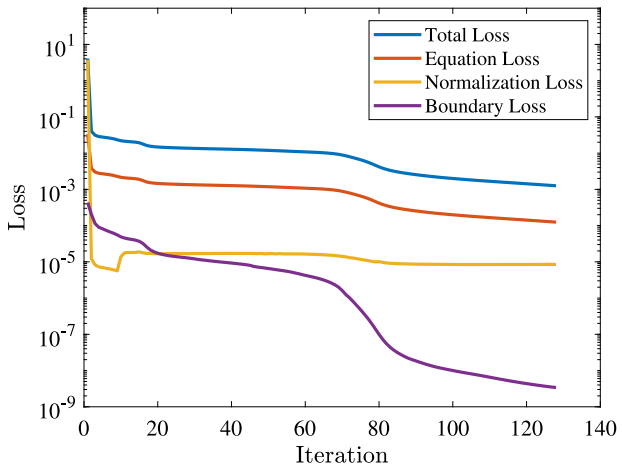

Fig. 14. The variation of the loss function of the 6D system (45) during training.

4.3. A 6D nonlinear system

Consider a three-DOF coupled Duffing system governed by

$$\begin{aligned}
\frac{dX_1}{dt} &= X_2,\\
\frac{dX_2}{dt} &= -\omega_1^2 X_1 - \alpha_{11}X_1^3 - \alpha_{12}X_1^2X_3 - \alpha_{13}X_1^2X_5 + \beta_{11}X_2 + \beta_{12}X_3\\
&\quad + \beta_{13}X_5 + W_1(t),\\
\frac{dX_3}{dt} &= X_4,\\
\frac{dX_4}{dt} &= -\omega_2^2 X_3 + \alpha_{21}X_1^3 - \alpha_{22}X_3^3 + \beta_{21}X_1 - \beta_{22}X_4 + W_2(t),\\
\frac{dX_5}{dt} &= X_6,\\
\frac{dX_6}{dt} &= -\omega_3^2 X_5 + \alpha_{31}X_1^3 - \alpha_{32}X_5^3 + \beta_{31}X_1 - \beta_{32}X_6 + W_3(t),
\end{aligned} \tag{45}$$

where $\omega_1 = 0.2$, $\omega_2 = 0.4, \omega_3 = 0.4$, $\alpha_{11} = 0.3$, $\alpha_{12} = -0.06$, $\alpha_{13} = -0.06$, $\alpha_{21} = 0.04$, $\alpha_{22} = 0.5$, $\alpha_{31} = 0.04$, $\alpha_{32} = 0.5$, $\beta_{11} = -0.2$, $\beta_{12} = 0.15$, $\beta_{13} = 0.15$, $\beta_{21} = 0.3$, $\beta_{22} = 0.2$, $\beta_{31} = 0.3$ and $\beta_{32} = 0.2$. $W_i(t)$ represent independent Gaussian white noise excitations with intensity $D_1 = D_2 = 0.05$. The drift and diffusion terms of the reduced FPK equation are given by,

$$\begin{aligned}
&A_1 = x_2, \quad A_3 = x_4, \quad A_5 = x_6,\\
&A_2 = -\omega_1^2 x_1 - \alpha_{11}x_1^3 - \alpha_{12}x_1^2x_3 - \alpha_{13}x_1^2x_5\\
&\quad + \beta_{11}x_2 + \beta_{12}x_3 + \beta_{13}x_5,\\
&A_4 = -\omega_2^2 x_3 + \alpha_{21}x_1^3 - \alpha_{22}x_3^3 + \beta_{21}x_1 - \beta_{22}x_4,\\
&A_6 = -\omega_3^2 x_5 + \alpha_{31}x_1^3 - \alpha_{32}x_5^3 + \beta_{31}x_1 - \beta_{32}x_6,\\
&B_{22} = 2D_1, \quad B_{44} = 2D_2, \quad B_{66} = 2D_3.
\end{aligned} \tag{46}$$

The domain to place the Gaussian neurons is $D_G = [-2, 2]^6$. The region of sampling points is $D_s = [-2.5, 2.5]^6$. The weights for the loss are $\lambda_1 = 10$, $\lambda_2 = 1$ and $\lambda_3 = 10$. The SGNN has 6 hidden layers with 41 neurons in each hidden layer. A total number of neurons is $6\times41 = 246$. The number of trainable parameters is $2\times6\times41 + (6-1)\times41^2 = 8897$. The total training data is $N_s = 8192\times1024$. We choose the mini-batch size as $N_b = 1024\times8$.

The marginal steady-state PDF solutions of the 4D stochastic system (45) obtained by the SGNN method after 130 iterations and by MCS with 10^9 samples are shown in Figs. 11 and 12. The RMS error of the FPK equation is 1.25×10^{-4}. The RMS errors defined in (40) of joint PDFs $p(x_1, x_2)$, $p(x_1, x_3)$, $p(x_4, x_5)$ and $p(x_4, x_6)$ are 2.86×10^{-2}, 2.46×10^{-2}, 1.78×10^{-2} and 3.73×10^{-2}, respectively. To show the details of the solution at the tail end, we show two marginal PDFs $p(x_1)$ and $p(x_2)$ in logarithmic scale in Fig. 13. The variation of the loss function during the training process is shown in Fig. 14.

The total training time for 130 iterations is 75 184 s. In contrast, the MCS solution with 10^9 samples requires 19 743 s to complete. The CPU time of MCS is less than that of the SGNN method, because the simulation is done in the subspace of the marginal probability density. 10^9 samples used by MCS are only sufficient to yield projections of the high-dimensional PDF in the low-dimensional subspace and are insufficient to obtain accurate PDF in the high-dimensional state space. By contrast, the SGNN delivers a semi-analytical solution of the PDF in the entire high-dimensional state space.

To illustrate this point further, we present Fig. 15 to show a comparison of a marginal PDF $p(x_1, x_2)$ between the SGNN and MCS results with 10^9 samples. The simulation is carried out in the $x_1 - x_2$ subspace with $x_3 = x_4 = x_5 = x_6 = 0$. It is apparent that much more sample points are needed for the MCS to deliver an accurate PDF.

Remark

A remark is in order regarding the comparison of SGNN with GRBFNN and other deep neural networks. Extensive numerical experiments have been done in [40] to compare the prediction, modeling performance and computational efficiency of SGNN with GRBFNN and deep neural networks with ReLu activation functions. It is found that for various complex functions in low and high-dimensional space, SGNN consistently outperforms GRBFNN and other deep neural networks in terms of the combined measure of accuracy and efficiency. For this reason, we shall not present comparison of FPK solutions by the SGNN method with those by GRBFNN and other deep neural networks.

5. Discussions

5.1. Comparison of SGNN with GRBFNN

The solutions obtained from SGNN and GRBFNN share identical forms. However, the most significant distinction between them lies in the substantial difference in the number of neurons and trainable parameters under the same spatial partition. For a d-dimensional system, assuming that $m_1 = m_2 \cdots = m_d = N$, the number of neurons and trainable parameters for the GRBFNN solution are N^d and $3N^d$, respectively. In contrast, for the SGNN solution, these numbers are dN and $(d-1)N^2 + 2dN$, respectively. Compared to GRBFNN, SGNN employs a number of neurons and trainable parameters that increase linearly with dimensionality, making it more suitable for high-dimensional problems.

5.2. Comparison of SGNN with ANN

In solving the FPK equation, traditional neural networks face a major challenge because the normalization condition can only be computed through numerical integration in the high-dimensional state space. Consequently, this precludes the utilization of mini-batch methods to accelerate training. For instance, Zhang et al. computed a linear 3D system using a network with 7 hidden layers, each containing 20 neurons, resulting in a computational time of 11 298.2 s [43]. In contrast, the SGNN, employing 3 hidden layers with 20 neurons each, achieved accurate results in merely 220 s.

6. Conclusions

This paper introduces a novel approach of SGNN to obtain the steady-state PDF of the reduced FPK equation of high-dimensional nonlinear stochastic systems. SGNN lets all unit-variant Gaussian neurons distributed in a region of a single state variable form a hidden layer, such that the number of hidden layers is equal to the dimension of the state space. The output of SGNN is a subset of that of a corresponding GRBFNN and is close to the true output of GRBFNN when the Hessian

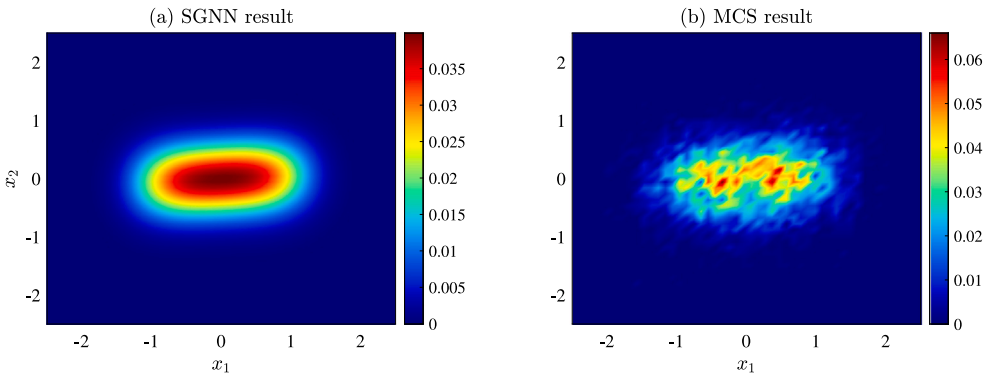

Fig. 15. The steady-state PDF of system (45) on the $x_1 - x_2$ with $x_3 = x_4 = x_5 = x_6 = 0$. Left: The SGNN result. Right: MCS results.

matrix of SGNN captures all the dominant eigenvalues of the Hessian matrix of GRBFNN [40]. The loss function used to train the neural networks consists of the residual of the FPK equation, the normalization condition and the boundary condition. Because the normalization condition and boundary condition directly impose constraints on the trainable parameters, the corresponding losses converge quickly in the training process. The choice of the normalized Gaussian functions as neurons removes the need for numerical integration when the marginal PDFs in arbitrary subspace are computed. We have developed the general expression for easy creation of marginal PDFs of any order. Three examples of complex nonlinear dynamic systems have been studied to illustrate the effectiveness of the proposed SGNN method in solving high-dimensional FPK equations. Extensive Monte Carlo simulations have been carried out to check the accuracy of the proposed method.

CRediT authorship contribution statement

Xi Wang: Writing – original draft, Software, Conceptualization. **Siyuan Xing:** Writing – review & editing, Software, Methodology, Conceptualization. **Jun Jiang:** Supervision, Methodology, Conceptualization. **Ling Hong:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization. **Jian-Qiao Sun:** Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors of the manuscript have no conflict of interest with each other or with any institution.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work is supported by Natural Science Foundation of China (Grant Nos. 11972070, 11972274 and 12172267).

References

[1] J.-Q. Sun, Stochastic Dynamics and Control, Elsevier, 2006.
[2] Y.-K. Lin, Probabilistic Theory of Structural Dynamics, McGraw-Hill, 1967.
[3] H. Risken, The Fokker-Planck Equation – Methods of Solution and Applications, Springer-Verlag, New York, 1984.
[4] T.K. Caughey, Nonlinear theory of random vibrations, in: Advances in Applied Mechanics, vol. 11, Elsevier, 1971, pp. 209–253, http://dx.doi.org/10.1016/S0065-2156(08)70343-0.
[5] Y.K. Lin, G.Q. Cai, Probabilistic Structural Dynamics - Advanced Theory and Applications, McGraw-Hill, New York, 1995.
[6] T.S. Atalik, S. Utku, Stochastic linearization of multi-degree-of-freedom nonlinear systems, Earthq. Eng. Struct. Dyn. 4 (4) (1976) 411–420, http://dx.doi.org/10.1002/eqe.4290040408.
[7] G. Falsone, Stochastic linearization of MDOF systems under parametric excitations, Int. J. Non-Linear Mech. 27 (6) (1992) 1025–1037, http://dx.doi.org/10.1016/0020-7462(92)90053-A.
[8] W.D. Iwan, R.G. Whirley, Nonstationary equivalent linearization of nonlinear continuous systems, Probab. Eng. Mech. 8 (3) (1993) 273–280, http://dx.doi.org/10.1016/0266-8920(93)90021-M.
[9] T.K. Caughey, Equivalent linearization techniques, J. Acoust. Soc. Am. 35 (11) (2005) 1706–1711, http://dx.doi.org/10.1121/1.1918794.
[10] T. Caughey, On the response of non-linear oscillators to stochastic excitation, Probab. Eng. Mech. 1 (1) (1986) 2–4.
[11] D.C. Polidori, J.L. Beck, Approximate solutions for non-linear random vibration problems, Probab. Eng. Mech. 11 (3) (1996) 179–185.
[12] W. Zhu, T. Soong, Y. Lei, Equivalent nonlinear system method for stochastically excited Hamiltonian systems, J. Appl. Mech. 61 (3) (1994) 618–623.
[13] W.Q. Zhu, Recent developments and applications of the stochastic averaging method in random vibration, Appl. Mech. Rev. 49 (10S) (1996) S72–S80, http://dx.doi.org/10.1115/1.3101980.
[14] Z. Huang, W. Zhu, Exact stationary solutions of averaged equations of stochastically and harmonically excited MDOF quasi-linear systems with internal and/or external resonances, J. Sound Vib. 204 (2) (1997) 249–258.
[15] J.-Q. Sun, C.S. Hsu, Cumulant-neglect closure method for nonlinear systems under random excitations, J. Appl. Mech. 54 (3) (1987) 649–655, http://dx.doi.org/10.1115/1.3173083.
[16] J. Náprstek, R. Král, Finite element method analysis of Fokker-Plank equation in stationary and evolutionary versions, Adv. Eng. Softw. 72 (2014) 28–38, http://dx.doi.org/10.1016/j.advengsoft.2013.06.016.
[17] M.P. Zorzano, H. Mais, L. Vazquez, Numerical solution of two dimensional Fokker—Planck equations, Appl. Math. Comput. 98 (2–3) (1999) 109–117.
[18] M.F. Wehner, W. Wolfer, Numerical evaluation of path-integral solutions to Fokker-Planck equations, Phys. Rev. A 27 (5) (1983) 2663, http://dx.doi.org/10.1103/PhysRevA.27.2663.
[19] M. Wehner, W. Wolfer, Numerical evaluation of path-integral solutions to Fokker-Planck equations. II. Restricted stochastic processes, Phys. Rev. A 28 (5) (1983) 3003.
[20] M. Wehner, W. Wolfer, Numerical evaluation of path-integral solutions to Fokker-Planck equations. III. Time and functionally dependent coefficients, Phys. Rev. A 35 (4) (1987) 1795.
[21] G.-K. Er, Exponential closure method for some randomly excited non-linear systems, Int. J. Non-Linear Mech. 35 (1) (2000) 69–78, http://dx.doi.org/10.1016/S0020-7462(98)00088-2.
[22] G.-K. Er, Probabilistic solutions of some multi-degree-of-freedom nonlinear stochastic dynamical systems excited by filtered Gaussian white noise, Comput. Phys. Comm. 185 (4) (2014) 1217–1222.
[23] J.Q. Sun, F. Xiong, O. Schütze, C. Hernández Castellanos, Cell Mapping Methods, Algorithmic Approaches and Applications, Springer, Berlin, 2019, pp. 1–226, http://dx.doi.org/10.1007/978-981-13-0457-6.
[24] H. Pradlwarter, G. Schuëller, On advanced Monte Carlo simulation procedures in stochastic structural dynamics, Int. J. Non-Linear Mech. 32 (4) (1997) 735–744.
[25] B.P. van Milligen, V. Tribaldos, J.A. Jiménez, Neural network differential equation and plasma equilibrium solver, Phys. Rev. Lett. 75 (20) (1995) 3594–3597, http://dx.doi.org/10.1103/PhysRevLett.75.3594.
[26] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.
[27] T. Leephakpreeda, Novel determination of differential-equation solutions: Universal approximation method, J. Comput. Appl. Math. 146 (2) (2002) 443–457, http://dx.doi.org/10.1016/S0377-0427(02)00397-7.

[28] A. Malek, R. Shekari Beidokhti, Numerical solution for high order differential equations using a hybrid neural network – Optimization method, Appl. Math. Comput. 183 (1) (2006) 260–271, http://dx.doi.org/10.1016/j.amc.2006.05.068.
[29] R. Shekari Beidokhti, A. Malek, Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques, J. Franklin Inst. B 346 (9) (2009) 898–913, http://dx.doi.org/10.1016/j.jfranklin.2009.05.003.
[30] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (4) (1989) 303–314, http://dx.doi.org/10.1007/BF02551274.
[31] J. Berg, K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41, http://dx.doi.org/10.1016/j.neucom.2018.06.056.
[32] J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. 115 (34) (2018) 8505–8510, http://dx.doi.org/10.1073/pnas.1718942115.
[33] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364, http://dx.doi.org/10.1016/j.jcp.2018.08.029.
[34] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, http://dx.doi.org/10.1016/j.jcp.2018.10.045.
[35] X. Chen, L. Yang, J. Duan, G.E. Karniadakis, Solving inverse stochastic problems from discrete particle observations using the Fokker-Planck equation and physics-informed neural networks, SIAM J. Sci. Comput. 43 (3) (2021) B811–B830, http://dx.doi.org/10.1137/20M1360153.
[36] X. Wang, J. Jiang, L. Hong, J.-Q. Sun, Random vibration analysis with radial basis function neural networks, Int. J. Dyn. Control 10 (5) (2022) 1385–1394, http://dx.doi.org/10.1007/s40435-021-00893-2.
[37] J. Qian, L. Chen, J.-Q. Sun, Random vibration analysis of vibro-impact systems: RBF neural network method, Int. J. Non-Linear Mech. 148 (2023) 104261, http://dx.doi.org/10.1016/j.ijnonlinmec.2022.104261.
[38] X. Wang, J. Jiang, L. Hong, J.-Q. Sun, First-passage problem in random vibrations with radial basis function neural networks, J. Vib. Acoust. 144 (5) (2022) http://dx.doi.org/10.1115/1.4054437.
[39] X. Wang, J. Jiang, L. Hong, L. Chen, J.-Q. Sun, On the optimal design of radial basis function neural networks for the analysis of nonlinear stochastic systems, Probab. Eng. Mech. 73 (2023) 103470, http://dx.doi.org/10.1016/j.probengmech.2023.103470.
[40] S. Xing, J.-Q. Sun, Separable Gaussian neural networks: Structure, analysis, and function approximations, Algorithms 16 (2023) 453.
[41] J. Park, I.W. Sandberg, Universal approximation using radial-basis-function networks, Neural Comput. 3 (1991) 246–257.
[42] D. Huntington, C. Lyrintzis, Improvements to and limitations of Latin hypercube sampling, Probab. Eng. Mech. 13 (4) (1998) 245–253, http://dx.doi.org/10.1016/S0266-8920(97)00013-1.
[43] Y. Xu, H. Zhang, Y. Li, K. Zhou, Q. Liu, J. Kurths, Solving Fokker-Planck equation using deep learning, Chaos 30 (1) (2020) http://dx.doi.org/10.1063/1.5132840.