Journal of Computational Physics 457 (2022) 111080

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Adaptive deep density approximation for Fokker-Planck equations

Kejun Tang [a,b], Xiaoliang Wan [c], Qifeng Liao [a,*]

[a] School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China
[b] Peng Cheng Laboratory, Shenzhen 518055, China
[c] Department of Mathematics and Center for Computation and Technology, Louisiana State University, Baton Rouge 70803, USA

ARTICLE INFO

Article history:
Received 19 March 2021
Received in revised form 14 December 2021
Accepted 12 February 2022
Available online 22 February 2022

Keywords:
Density estimation
Flow-based generative models
Fokker-Planck equations
Deep learning

ABSTRACT

In this paper we present an adaptive deep density approximation strategy based on KRnet (ADDA-KR) for solving the steady-state Fokker-Planck (F-P) equations. F-P equations are usually high-dimensional and defined on an unbounded domain, which limits the application of traditional grid based numerical methods. With the Knothe-Rosenblatt rearrangement, our newly proposed flow-based generative model, called KRnet, provides a family of probability density functions to serve as effective solution candidates for the Fokker-Planck equations, which has a weaker dependence on dimensionality than traditional computational approaches and can efficiently estimate general high-dimensional density functions. To obtain effective stochastic collocation points for the approximation of the F-P equation, we develop an adaptive sampling procedure, where samples are generated iteratively using the approximate density function at each iteration. We present a general framework of ADDA-KR, validate its accuracy and demonstrate its efficiency with numerical experiments.

1. Introduction

During the past few decades there has been a rapid development in numerical methods for Fokker-Planck equations. This explosion in interest has been driven by the need of assessing time evolution of probability density functions in randomly perturbed dynamical systems, which are widely used in physical and biological modeling [1–3]. It is known that there exist two main challenges for efficiently solving the Fokker-Planck equations: the spatial variable can be high-dimensional, which causes difficulties in applying grid based numerical methods, e.g. finite element methods [4,5]; the original spatial domain is typically unbounded, and it is challenging to derive a well-posed boundary condition for a bounded computational domain. To alleviate these difficulties, new numerical methods based on deep learning currently gain a lot of attention [6,7], and this paper is devoted to deep learning for the Fokker-Planck equations.

Deep learning methods for partial differential equations (PDEs) are under active development. In [8,9], a deep Ritz method is proposed based on variational methods. In [10–13], physics-informed neural networks are developed through infusing PDEs into networks as a constraint. A deep Galerkin method is proposed in [14]. Bayesian deep convolutional encoder-decoder networks for PDEs with high-dimensional random inputs are developed in [15,16]. Deep learning strategies are also introduced to discover physical laws [17,18]. In addition, efficient deep learning methods based on domain

* Corresponding author.
E-mail addresses: tangkj@shanghaitech.edu.cn (K. Tang), xlwan@lsu.edu (X. Wan), liaoqf@shanghaitech.edu.cn (Q. Liao).

https://doi.org/10.1016/j.jcp.2022.111080
0021-9991/

decomposition are studied in [19–24], and deep neural network methods for complex geometries and irregular domains are proposed in [25,26]. The main idea of deep learning methods for PDEs is to reformulate a PDE problem as an optimization problem and train deep neural networks through minimizing the corresponding loss functional. In these methods, stochastic collocation points are required to estimate the loss functional. We note that the stochastic collocation points herein are for the spatial variable, while stochastic collocation for PDEs with random inputs (especially for parameters) are discussed in detail in [27–33]. To result in an efficient deep learning strategy for PDEs, properly choosing the collocation points is crucial. Intuitively, the distribution of the collocation points should be consistent with the properties of the PDE solution in a certain sense. In our recent work [19], a hierarchical sampling procedure is proposed based on domain decomposition iterations, while it focuses on low-dimensional problems. As the spatial variable of the Fokker-Planck equation can be high-dimensional, it remains an open challenging problem to generate effective collocation points. We develop an effective adaptive sampling procedure to alleviate this issue in this work. Adaptivity is widely used in machine learning techniques to make the training process more effective by exploring the relation between the model and the data, e.g., active learning selects the most helpful samples to increase efficiency [34,35] and meta-learning tries to match learning algorithms with task properties [36]. In our problem, we will update the training set partially or completely according to the learned model, i.e., the approximate solution of the F-P equation, and the updated training set will yield a better approximate solution.

As the solution of the Fokker-Planck equation is a probability density function, solving this problem can also be considered as a density estimation problem. It is known that density estimation is a central topic in unsupervised learning, and it still remains an open challenge for high-dimensional density estimation [37]. Recently, two kinds of deep learning models have shown great promise for estimating high-dimensional probability density functions (PDFs), which include the flow-based generative model [38,39] and the neural ordinary differential equation model [40,41]. In this work, we focus on the flow-based generative model, which is to construct invertible mappings from a prescribed prior distribution to the empirical distribution given by data and build explicit probability density functions using the change of variables. The Knothe-Rosenblatt (KR) rearrangement [42] shows that such an invertible mapping can be achieved with a triangular structure. Incorporating with the KR rearrangement, we propose an invertible block-triangular mapping, called KRnet, which generalizes the flow-based generative model given by real NVP [38]. We note that there are a lot of generative models which can efficiently generate samples of the distributions under consideration but do not explicitly give the corresponding density functions, e.g., generative adversarial networks (GANs) [43] and the variational autoencoder (VAE) [44]. In addition, coupling flow-based generative models and reduced-order models into an importance sampling estimator is studied in [45].

In this work, we propose an adaptive deep density approximation method based on KRnet (ADDA-KR) for solving Fokker-Planck equations. We first provide additional details and results for KRnet that was outlined in the letter [46]. After that, we use KRnet to construct solutions of the Fokker-Planck equation. Since KRnet can induce a family of probability density functions, normality and vanishing boundary conditions are satisfied naturally. Like other deep learning algorithms for solving PDEs, our method is also meshfree. The PDE problem is converted into an optimization problem and it can be solved through stochastic gradient descent on a set of collocation points, while traditional grid-based numerical methods (e.g. finite element methods) rapidly become computationally infeasible since the number of grid points grows exponentially with the dimensionality. The choice of the collocation points plays a crucial role in a meshless method. The distribution of the collocation points should be consistent with the regularity of the solution for both accuracy and efficiency. Since the solution of the F-P equation is a probability density function, one way to achieve this is to use the samples of the solution PDF as the collocation points. Based on such an idea, we propose an adaptive approach ADDA-KR that has two main steps: training a KRnet to approximate the solution of the Fokker-Planck equation, and using the trained KRnet to generate collocation points for the next iteration. After each iteration, the distribution of the collocation points is more consistent with the solution PDF.

The rest of the paper is organized as follows. In the next section, the Fokker-Planck equations and the problem setting are introduced. Our KRnet is presented in section 3. In section 4, our novel adaptive deep density approximation approach for the Fokker-Planck equation is presented. In section 5, we demonstrate the efficiency of our adaptive sampling approach with numerical experiments. Finally section 6 concludes the paper.

2. Problem setup

Consider the state X_t modeled by the following stochastic differential equation

$$dX_t = \boldsymbol{\mu}(X_t, t)dt + \boldsymbol{G}(X_t, t)d\boldsymbol{w}_t, \tag{1}$$

where $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_d]^{\mathsf{T}}$ is a vector field, $\boldsymbol{G}(X_t, t) \in \mathbb{R}^{d\times d}$ is a matrix-valued function and $\boldsymbol{w}_t$ is a d-dimensional standard Wiener process. The Fokker-Planck equation, which describes the probability density function of X_t, is

$$\begin{aligned}\frac{\partial p(\boldsymbol{x}, t)}{\partial t} = \mathcal{L}p(\boldsymbol{x}, t) := \nabla \cdot [p(\boldsymbol{x}, t)\nabla V(\boldsymbol{x}, t)] + \nabla \cdot [\nabla \cdot (p(\boldsymbol{x}, t)\boldsymbol{D}(\boldsymbol{x}, t))], \qquad \forall (\boldsymbol{x}, t) \in \mathbb{R}^d \times \mathbb{R}^+,\\ \int_{\mathbb{R}^d} p(\boldsymbol{x}, t)d\boldsymbol{x} = 1,\ p(\boldsymbol{x}, t) \geq 0, \qquad \forall (\boldsymbol{x}, t) \in \mathbb{R}^d \times \mathbb{R}^+,\\ p(\boldsymbol{x}, 0) = p_0(\boldsymbol{x}),\end{aligned} \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ denotes a random vector, $V(\boldsymbol{x}, t)$ is a potential function, $\boldsymbol{D}(\boldsymbol{x}, t)$ is a diffusion matrix, $p(\boldsymbol{x}, t)$ is the unknown probability density function (PDF) of $\boldsymbol{x}$ with the initial PDF $p_0(\boldsymbol{x})$, and $\mathcal{L}$ denotes the partial differential operator. Following [1], the potential function $V(\boldsymbol{x}, t)$ and the diffusion matrix $\boldsymbol{D}(\boldsymbol{x}, t)$ can be expressed as

$$\nabla V(\boldsymbol{x}, t) = -\boldsymbol{\mu}(\boldsymbol{x}, t),$$
$$\boldsymbol{D}(\boldsymbol{x}, t)) = \frac{1}{2}\boldsymbol{G}(\boldsymbol{x}, t)\boldsymbol{G}(\boldsymbol{x}, t)^{\mathsf{T}}.$$

In this work, we focus on the stationary solution of Eq. (2), i.e., the invariant measure independent of time,

$$\mathcal{L}p(\boldsymbol{x}) = \nabla \cdot [p(\boldsymbol{x})\nabla V(\boldsymbol{x})] + \nabla \cdot [\nabla \cdot (p(\boldsymbol{x})\boldsymbol{D}(\boldsymbol{x}))] = 0, \tag{3}$$

with the boundary condition

$$p(\boldsymbol{x}) \to 0 \quad \text{as} \quad \|\boldsymbol{x}\|_2 \to \infty, \tag{4}$$

and some extra constraints on $p(\boldsymbol{x})$

$$\int_{\mathbb{R}^d} p(\boldsymbol{x})d\boldsymbol{x} = 1, \quad \text{and} \quad p(\boldsymbol{x}) \geq 0, \tag{5}$$

where $\|\boldsymbol{x}\|_2$ indicates the ℓ_2 norm of $\boldsymbol{x}$.

There are several difficulties for the approximation of equation (3). First, the boundary condition and the constraints of $p(\boldsymbol{x})$ may not be easily satisfied when we employ the traditional approaches such as the finite element method. Since the support of $p(\boldsymbol{x})$ is $\mathbb{R}^d$, the computation domain has to be truncated, implying that the boundary condition must be approximated, e.g., a homogeneous boundary condition. To preserve the nonnegativity of $p(\boldsymbol{x})$, a projection step is needed for the box constraint. Second, it requires a fine mesh to capture the whole information when the target density is multimodal, i.e., the potential function $V(\boldsymbol{x})$ has many local minima [47], which is computationally infeasible when the dimension d is even moderately large. We also note that a homogeneous boundary condition usually requires a large computational domain, which makes a uniform refinement even more challenging, if no prior information can be used for certain adaptivity on mesh generation. To address these issues, we will propose an adaptive deep density approximation method to solve the Fokker-Planck equation (3) using a deep generative model for $p(\boldsymbol{x})$. The flow-based generative model not only provides an explicit density function that satisfies naturally all constraints on $p(\boldsymbol{x})$, but also suggests a simple but effective adaptive strategy for the approximation of equation (3) through sampling the current approximation of $p(\boldsymbol{x})$.

3. KRnet

KRnet is a flow-based generative model for density estimation or approximation. In this section we briefly overview KRnet that has been outlined in our recently published letter [46] and present more details that were not included in [46] due to the page limit. Let $X \in \mathbb{R}^d$ be a random vector associated with a given data set, and its probability density function (PDF) is denoted by $p_X(\boldsymbol{x})$. The target is to estimate $p_X(\boldsymbol{x})$ using available data. Let $Z \in \mathbb{R}^d$ be a random vector associated with a PDF $p_Z(\boldsymbol{z})$, where $p_Z(\boldsymbol{z})$ is a prior distribution (e.g., Gaussian distribution). The flow-based generative modeling is to seek an invertible mapping $\boldsymbol{z} = f(\boldsymbol{x})$ where $f(\cdot)$ is a bijection: $f : \boldsymbol{x} \mapsto \boldsymbol{z}$ [38]. By the change of variables, we have the PDF of $X = f^{-1}(Z)$ as

$$p_X(\boldsymbol{x}) = p_Z(f(\boldsymbol{x}))\left|\det \nabla_{\boldsymbol{x}} f\right|. \tag{6}$$

Once the prior distribution $p_Z(\boldsymbol{z})$ is specified, equation (6) provides an explicit PDF of X. Given a set of training data, the invertible mapping $f(\cdot)$ can be learned by maximizing the likelihood or minimizing the cross entropy. The inverse of $f(\cdot)$ provides a convenient way to sample X as $X = f^{-1}(Z)$.

3.1. A new affine coupling layer

In flow-based generative models, the invertible mapping $f(\cdot)$ is constructed by stacking a sequence of simple bijections, each of which is a shallow neural network, and thus the overall mapping is a deep net. The mapping $f(\cdot)$ can be written in a composite form:

$$\boldsymbol{z} = f(\boldsymbol{x}) = f_{[L]} \circ \ldots \circ f_{[1]}(\boldsymbol{x}) \quad \text{and} \quad \boldsymbol{x} = f^{-1}(\boldsymbol{z}) = f_{[1]}^{-1} \circ \ldots \circ f_{[L]}^{-1}(\boldsymbol{z}), \tag{7}$$

where $f_{[i]}$ is called an affine coupling layer at stage i. The Jacobian matrix can be obtained by the chain rule

$$\left|\det \nabla_{\boldsymbol{x}} f\right| = \prod_{i=1}^{L} \left|\det \nabla_{\boldsymbol{x}_{[i-1]}} f_{[i]}\right|, \tag{8}$$

where $\boldsymbol{x}_{[i-1]}$ indicate the intermediate variables with $\boldsymbol{x}_{[0]} = \boldsymbol{x}$ and $\boldsymbol{x}_{[L]} = \boldsymbol{z}$. Let $\boldsymbol{x}_{[i]} = [\boldsymbol{x}_{[i],1}, \boldsymbol{x}_{[i],2}]^{\mathsf{T}}$ be a partition of $\boldsymbol{x}_{[i]}$ with $\boldsymbol{x}_{[i],1} \in \mathbb{R}^m$ and $\boldsymbol{x}_{[i],2} \in \mathbb{R}^{d-m}$ for $i = 0, \ldots, L-1$. One technique to define the affine coupling layer is the real NVP [38]:

$$\begin{aligned} \boldsymbol{x}_{[i],1} &= \boldsymbol{x}_{[i-1],1} \\ \boldsymbol{x}_{[i],2} &= \boldsymbol{x}_{[i-1],2} \odot \exp\left(\log \boldsymbol{s}_i(\boldsymbol{x}_{[i-1],1})\right) + \boldsymbol{t}_i(\boldsymbol{x}_{[i-1],1}), \end{aligned} \tag{9}$$

where $\boldsymbol{s}_i : \mathbb{R}^m \mapsto \mathbb{R}^{d-m}$ and $\boldsymbol{t}_i : \mathbb{R}^m \mapsto \mathbb{R}^{d-m}$ are the scaling and the translation depending on $\boldsymbol{x}_{[i-1],1}$, and $\odot$ is the Hadamard product or element-wise product. Note that $\boldsymbol{x}_{[i-1],1}$ remains fixed and the modification of $\boldsymbol{x}_{[i-1],2}$ is linear with respect to $\boldsymbol{x}_{[i-1],2}$ and nonlinear in terms of $\boldsymbol{x}_{[i-1],1}$. This way, the Jacobian matrix $\nabla_{\boldsymbol{x}_{[i-1]}} f_{[i]}$ is lower-triangular whose determinant can be evaluated efficiently. Furthermore, $(\boldsymbol{s}_i, \boldsymbol{t}_i)$ is usually modeled by a neural network $\mathsf{NN}_{[i]}$

$$(\boldsymbol{s}_i, \boldsymbol{t}_i) = \mathsf{NN}_{[i]}(\boldsymbol{x}_{[i-1],1}). \tag{10}$$

We proposed a new affine coupling layer $f_{[i]}$ as follows [46]

$$\begin{aligned} \boldsymbol{x}_{[i],1} &= \boldsymbol{x}_{[i-1],1} \\ \boldsymbol{x}_{[i],2} &= \boldsymbol{x}_{[i-1],2} \odot \left(1 + \alpha \tanh(\boldsymbol{s}_i(\boldsymbol{x}_{[i-1],1}))\right) + e^{\boldsymbol{\beta}_i} \odot \tanh(\boldsymbol{t}_i(\boldsymbol{x}_{[i-1],1})), \end{aligned} \tag{11}$$

where $0 < \alpha < 1$ is a hyperparameter and the parameter $\boldsymbol{\beta}_i \in \mathbb{R}^{d-m}$ is trainable. Our affine coupling layer keeps the mechanism of the real NVP when updating the data, and it has the following advantages. First, the second equation in Eq. (11) adapts the trick of ResNet [48], where an identity mapping is added to improve the training process. Second, the constant $\alpha \in (0, 1)$ is introduced to improve numerical stability. It is seen that the range of $\det \nabla_{\boldsymbol{x}_{[i-1]}} f_{[i]}$ is $[(1-\alpha)^{d-m}, (1+\alpha)^{d-m}]$ for our affine coupling layer and $(0, +\infty)$ for the original real NVP. Our formulation can alleviate the illnesses when the determinant of the Jacobian in the original real NVP occasionally becomes too large or too small. Third, the trainable factor $e^{\boldsymbol{\beta}_i}$ depends on the whole training set, which helps avoid possible large oscillation in $\boldsymbol{t}_i(\boldsymbol{x}_{[i-1],1})$ such that the number of outliers can be reduced for sample generation [46]. In our numerical experiments, we set $\alpha = 0.6$ and it works well.

Since the affine coupling layer $f_{[i]}$ only updates a part of $\boldsymbol{x}_{[i-1]}$, another affine coupling layer is needed for a complete update. In other words, the next affine coupling layer $f_{[i+1]}$ can be defined as

$$\begin{aligned} \boldsymbol{x}_{[i+1],1} &= \boldsymbol{x}_{[i],1} \odot \left(1 + \alpha \tanh(\boldsymbol{s}_{i+1}(\boldsymbol{x}_{[i],2}))\right) + e^{\boldsymbol{\beta}_{i+1}} \odot \tanh\left(\boldsymbol{t}_{i+1}(\boldsymbol{x}_{[i],2})\right) \\ \boldsymbol{x}_{[i+1],2} &= \boldsymbol{x}_{[i],2}, \end{aligned}$$

where the components $\boldsymbol{x}_{[i],1}$ are updated and $\boldsymbol{x}_{[i],2}$ remains unchanged. From the dynamical point of view, a long chain of affine coupling layers may result in a highly nonlinear transformation of the input. To enhance the performance and efficiency of the mapping $f(\boldsymbol{x})$, we proposed KRnet to address the following questions: 1) How should we partition the vector? 2) How can we increase the modeling capability except for increasing the depth L? 3) Can we provide a robust nonlinear bijection at least in a component-wise way?

3.2. The overall structure of KRnet

The basic idea of KRnet is to define the structure of $f(\boldsymbol{x})$ in terms of the Knothe-Rosenblatt rearrangement. Let μ_Z and μ_X be the probability measures of two random variables $X, Z \in \mathbb{R}^d$ respectively. A mapping $\mathcal{T}$: $Z \mapsto X$ is called a transport map such that $\mathcal{T}_{\#}\mu_Z = \mu_X$, where $\mathcal{T}_{\#}\mu_Z$ is the push-forward of μ_Z such that $\mu_X(B) = \mu_Z(\mathcal{T}^{-1}(B))$ for every Borel set B [42]. The Knothe-Rosenblatt rearrangement tells us that the transport map $\mathcal{T}$ may have a lower-triangular structure

$$\boldsymbol{z} = \mathcal{T}^{-1}(\boldsymbol{x}) = \begin{bmatrix} \mathcal{T}_1(x_1) \\ \mathcal{T}_2(x_1, x_2) \\ \vdots \\ \mathcal{T}_d(x_1, \ldots, x_d) \end{bmatrix}. \tag{12}$$

This mapping can be regarded as a limit of sequence of optimal transport maps when the quadratic cost degenerates [42]. Noticing that the invertible mapping $f(\boldsymbol{x})$ also defines a transport map, we then incorporate the triangular structure of the Knothe-Rosenblatt rearrangement into the definition of $f(\boldsymbol{x})$ which results in KRnet as a generalization of real NVP [38]. Let $\boldsymbol{x} = [\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(K)}]^{\mathsf{T}}$ be a partition of $\boldsymbol{x}$, where $\boldsymbol{x}^{(i)} = [x_1^{(i)}, \ldots, x_m^{(i)}]^{\mathsf{T}}$ with $1 \le K \le d$, $1 \le m \le d$, and $\sum_{i=1}^{K} \dim(\boldsymbol{x}^{(i)}) = d$. Our KRnet takes an overall form

$$\boldsymbol{z} = f_{\mathsf{KR}}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}^{(1)}) \\ f_2(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) \\ \vdots \\ f_K(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(K)}) \end{bmatrix}, \tag{13}$$

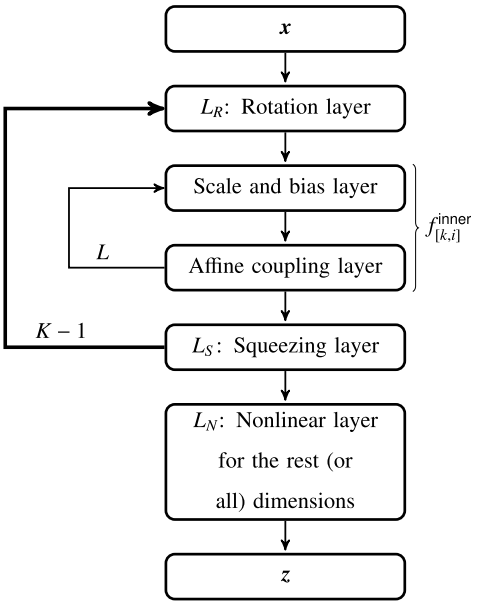


Fig. 1. The flow chart of KRnet.

where each f_i is an invertible mapping defined as in equation (7) for $i = 2, \ldots, K$. Note that f_1 is not included if $K = d$ because we need to partition a vector to two parts to define the affine coupling layer. KRnet consists of one outer loop and $K-1$ inner loops. The outer loop has $K-1$ stages, corresponding to the $K-1$ mappings f_i in equation (13) with $i = 2, \ldots, K$, and for each stage, an inner loop of L affine coupling layers is defined. More specifically, we have

$$\boldsymbol{z} = f_{\mathsf{KR}}(\boldsymbol{x}) = L_N \circ f^{\text{outer}}_{[K-1]} \circ \cdots \circ f^{\text{outer}}_{[1]}(\boldsymbol{x}), \tag{14}$$

where $f^{\text{outer}}_{[i]}$ is defined as

$$f^{\text{outer}}_{[k]} = L_S \circ f^{\text{inner}}_{[k,L]} \circ \cdots \circ f^{\text{inner}}_{[k,1]} \circ L_R. \tag{15}$$

Here $f^{\text{inner}}_{[k,i]}$ indicates a combination of one affine coupling layer and one scale and bias layer, and L_N, L_S and L_R indicate the nonlinear layer, the squeezing layer and the rotation layer, respectively, which will be briefly overviewed in the next section.

The flow chart of KRnet is illustrated in Fig. 1. Let us look at how the information flows in the KRnet. Each $\boldsymbol{x}_{[k]} = [\boldsymbol{x}^{(1)}_{[k]}, \ldots, \boldsymbol{x}^{(K)}_{[k]}]^{\mathsf{T}}$ has the same partition with $\boldsymbol{x}_{[k]} = f^{\text{outer}}_{[k]}(\boldsymbol{x}_{[k-1]})$ with $\boldsymbol{x}_{[0]} = \boldsymbol{x}$, $k = 1, \ldots, K-1$. At the beginning, a sequence of affine coupling layers in $f^{\text{outer}}_{[1]}$ is applied to the partition $\boldsymbol{x}_{[0]} = [\boldsymbol{x}^{(1:K-1)}_{[0]}, \boldsymbol{x}^{(K)}_{[0]}]^{\mathsf{T}}$, where $\boldsymbol{x}^{(1:K-1)}_{[0]}$ includes $\boldsymbol{x}^{(i)}_{[0]}$, $i = 1, \ldots, K-1$. From then on, the last partition $\boldsymbol{x}^{(K)}_{[k]}$ will remain fixed for $k > 1$. For the next iteration $f^{\text{outer}}_{[2]}$, the partition $[\boldsymbol{x}^{(1:K-2)}_{[1]}, \boldsymbol{x}^{(K-1)}_{[1]}]^{\mathsf{T}}$ will be used with $\boldsymbol{x}^{(K)}_{[1]}$ being deactivated. In general, after the stage $K-i+1$ of the outer loop, the i-th partition of $\boldsymbol{x}^{(i)}_{[k]}$ will become deactivated, in addition to the dimensions that are deactivated in the previous stages.

3.3. Other types of layer used in KRnet

Except for the affine coupling layers, several other types of layers are needed for the definition of KRnet. We briefly overview these layers in this section and provide some details excluded in the letter [46]. Since each $\boldsymbol{x}_{[k]}$ has the same partition, we will drop the subscript for simplicity.

Squeezing layer L_S is used to deactivate some dimensions using a mask

$$\boldsymbol{q} = [\underbrace{1, \ldots, 1}_{n}, \underbrace{0, \ldots, 0}_{d-n}]^{\mathsf{T}}, \tag{16}$$

where the components $\boldsymbol{q} \odot \boldsymbol{x}$ will keep being updated and the rest components $(1 - \boldsymbol{q}) \odot \boldsymbol{x}$ will be fixed from then on.

Scale and bias layer provides a simplification of the batch normalization [49], which is defined as

$$\hat{\boldsymbol{x}} = \boldsymbol{a} \odot \boldsymbol{x} + \boldsymbol{b}, \tag{17}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are trainable and initialized by the mean and standard deviation of the data. After the initialization, $\boldsymbol{a}$ and $\boldsymbol{b}$ will be treated as regular trainable parameters that are independent of the data. Numerical experiments show that the scale and bias layer is simple but effective, which provides a comparable performance to the batch normalization layer in our problem setting.

Rotation layer L_R defines a linear mapping of the input $\boldsymbol{x}$

$$\hat{\boldsymbol{x}} = \hat{\boldsymbol{W}}\boldsymbol{x}.$$

through a trainable matrix

$$\hat{\boldsymbol{W}} = \begin{bmatrix} \boldsymbol{W} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{U} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix},$$

where $\boldsymbol{W} \in \mathbb{R}^{n\times n}$, n is defined in the mask $\boldsymbol{q}$, $\boldsymbol{I} \in \mathbb{R}^{(d-n)\times(d-n)}$ is an identity matrix, and $\boldsymbol{W} = \boldsymbol{L}\boldsymbol{U}$ is the LU factorization of $\boldsymbol{W}$. We expect $\hat{\boldsymbol{W}}$ to provide a rotation such that the less important dimensions will be put at the end and deactivated by the next squeezing layer. Entries below the main diagonal of $\boldsymbol{L}$ and entries in the upper triangle of $\boldsymbol{U}$ are trainable. In practice, we simply optimize the trainable entries of $\boldsymbol{L}$ and $\boldsymbol{U}$ without enforcing the orthonormality of $\hat{\boldsymbol{W}}$ and such a simplification works well.

Nonlinear layer L_N provides a component-wise nonlinear transformation. For simplicity, we only consider one component x of the data. We start with a nonlinear mapping $F(s) : [0, 1] \mapsto [0, 1]$:

$$F(s) = \int_0^s p(t)dt, \tag{18}$$

where $p(s)$ is a probability density function. Let $0 = s_0 < s_1 < \ldots < s_{\hat{m}+1} = 1$ be a mesh of the interval $[0, 1]$ with element size $h_i = s_{i+1} - s_i$. Define $p(s)$ as a piece-wise linear polynomial

$$p(s) = \frac{w_{i+1} - w_i}{h_i}(s - s_i) + w_i, \quad \forall s \in [s_i, s_{i+1}], \tag{19}$$

where

$$p(s_i) = w_i.$$

Then $F(s)$, corresponding to a cumulative density function, is a quadratic function

$$F(s) = \frac{w_{i+1} - w_i}{2h_i}(s - s_i)^2 + w_i(s - s_i) + \sum_{k=0}^{i-1} \frac{w_k + w_{k+1}}{2} h_i, \quad \forall s \in [s_i, s_{i+1}], \tag{20}$$

whose inverse and derivative can be explicitly computed.

As the support of each dimension of $\boldsymbol{x}$ is $(-\infty, \infty)$, a question is how to apply $F(s)$ to the data. A straightforward strategy is to map $(-\infty, \infty)$ to $(0, 1)$ before $F(s)$ is applied. However, when the inverse is considered, the singularity of mapping a finite interval to an infinite one may introduce issues on robustness. To alleviate this problem, we decompose $(-\infty, \infty) = (-\infty, -a) \cup [-a, a] \cup (a, \infty)$ with $a > 0$, and define the following nonlinear mapping

$$\hat{F}(x) = \begin{cases} \beta_s(x + a) - a, & x \in (-\infty, -a) \\ 2aF\left(\frac{x+a}{2a}\right) - a, & x \in [-a, a] \\ \beta_s(x - a) + a, & x \in (a, \infty), \end{cases} \tag{21}$$

where $\beta_s > 0$ is a scaling factor. It is seen that we only consider a nonlinear mapping for the data located in $[-a, a]$ and $\hat{F}(x)$ maps $[-a, a]$ to itself. On $(-\infty, -a) \cup (a, \infty)$, $\hat{F}(x)$ is simply a linear mapping. The reasoning of such a strategy is that the range of data in the training set is always finite, and after being well scaled and shifted the data will be roughly centered at the origin, implying that a nonlinear mapping on $[-a, a]$ is sufficient as long as a is large enough. To maintain the invertibility, we require some regularity at $x = \pm a$. More specifically, $\hat{F}'(x)$ should exist at $x = \pm a$. Since $\hat{F}'(x) = \beta_s$ on $(-\infty, -a) \cup (a, \infty)$, we have, on $[-a, a]$, $\hat{F}'(x)|_{x=\pm a} = F'(s)|_{s=0,1} = p(s)|_{s=0,1} = \beta_s$. So the trainable parameters include $p(s_i) = w_i$, $i = 1, \ldots, \hat{m}$, subject to the constraint $\int_0^1 p(s)ds = 1$.

Remark 1. The nonlinear layer is only employed before the final output (see Fig. 1), which can be applied to all dimensions or simply the dimensions that have not been deactivated by the squeezing layer. In both cases, the nonlinear layer enlarges the prescribed prior distribution by a nonlinear component-wise transformation. The parameter β_s acts as an estimate of the density $p(s)$ at $s = 0, 1$. If a is sufficiently large, β_s can be small accordingly. The prior distribution is often chosen as the standard Gaussian, which means that the density is larger around the origin when the data pass the nonlinear layer. This suggests we may consider an adaptive mesh for more effectiveness, in other words, the mesh is finer around $s = 1/2$ and coarser around $s = 0, 1$.

3.4. The complexity of KRnet

We count the number of trainable parameters in KRnet. For simplicity, we assume that each $f_{[k]}^{\text{outer}}$ has L general coupling layers $f_{[k,i]}^{\text{inner}}$. Let d_k be the number of effective dimensions for $f_{[k]}^{\text{outer}}$ and $N_{\text{NN},k}$ the number of model parameters for the neural network Eq. (10) used in $f_{[k,i]}^{\text{inner}}$. We note that the main characteristic of KRnet is that a portion of dimensions will be deactivated as k increases. As d_k decreases with k, we expect that the neural network Eq. (10) in $f_{[k,i]}^{\text{inner}}$ should become simpler for a larger k. In other words, $N_{\text{NN},k}$ may decrease as k increases. For simplicity, we let $N_{\text{NN},k} = rN_{\text{NN},k-1}$, where $0 < r < 1$, without worrying about the detailed configuration of the neural network. The number of trainable parameters is d_k^2 for L_R, and $\hat{m}d$ for L_N, and $2d_k$ for the scale and bias layer. Assume that $d = mK$. We have $d_k = d - (k-1)m$, $k = 1, \ldots, K$. According to the flow chart in Fig. 1, we have the total number of model parameters as

$$N_{\text{dof}} = \hat{m}d + \sum_{k=1}^{K-1} (N_{\text{NN},1} r^{k-1} L + (K-k+1)^2 m^2 + 2(K-k+1)mL). \tag{22}$$

The model complexity is mainly determined by the depth L and the number K for the partition of data.

3.5. KRnet for density estimation

We study the performance of KRnet for density estimation in this part and provide more results on the comparison between the real NVP and the KRnet that were not included in [46]. Once the KRnet is constructed, we train the model $p_X(\boldsymbol{x}; \Theta)$ by maximizing the likelihood of the data or minimizing the cross entropy between the data distribution and the density model, where Θ includes all the trainable model parameters. Let $\mathcal{S} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N_t}$ be the training set and $p_{X,\text{data}}(\boldsymbol{x})$ the underlying data distribution. The Kullback-Leibler (KL) divergence between $p_{X,\text{data}}(\boldsymbol{x})$ and $p_X(\boldsymbol{x}; \Theta)$ is

$$\min_{\Theta} D_{KL}(p_{X,\text{data}}(\boldsymbol{x}) || p_X(\boldsymbol{x}; \Theta)) = \mathbb{E}_{\boldsymbol{x} \sim p_{X,\text{data}}(\boldsymbol{x})} \left[\log \frac{p_{X,\text{data}}(\boldsymbol{x})}{p_X(\boldsymbol{x}; \Theta)} \right] = H\left(p_{X,\text{data}}(\boldsymbol{x}), p_X(\boldsymbol{x}; \Theta)\right) - H\left(p_{X,\text{data}}(\boldsymbol{x})\right) \tag{23}$$

where $H\left(p_{X,\text{data}}(\boldsymbol{x})\right)$ is the entropy of $p_{X,\text{data}}(\boldsymbol{x})$ and $H\left(p_{X,\text{data}}(\boldsymbol{x}), p_X(\boldsymbol{x}; \Theta)\right)$ is the cross entropy of $p_{X,\text{data}}(\boldsymbol{x})$ and $p_X(\boldsymbol{x}; \Theta)$. Since $p_{X,\text{data}}(\boldsymbol{x})$ is independent of Θ, minimizing the KL divergence is equivalent to minimizing the cross entropy. Note that

$$H(p_{X,\text{data}}(\boldsymbol{x}), p_X(\boldsymbol{x}; \Theta)) \approx -\frac{1}{N_t} \sum_{i=1}^{N_t} \log p_X(\boldsymbol{x}^{(i)}; \Theta), \tag{24}$$

which corresponds to the negation of the log-likelihood.

To measure the quality of KRnet, we compute the KL divergence Eq. (23) on a validation set between a reference PDF and the trained density model. The training data sets $\mathcal{S}$ is generated as follows. Assume that X has i.i.d. components and each component $X_i \sim \text{Logistic}(0, s)$ has a PDF $\rho(x_i; 0, s)$. We generate a sample $\boldsymbol{x}^{(i)}$ of X, and then check if it satisfies the following constraint:

$$\left\| \boldsymbol{R}_{\gamma,\theta_j} [x_j^{(i)}, x_{j+1}^{(i)}]^{\mathsf{T}} \right\|_2 \geq C, \quad j = 1, \ldots, d-1, \tag{25}$$

where C is a specified constant, and

$$\boldsymbol{R}_{\gamma,\theta_j} = \begin{bmatrix} \gamma & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_j & -\sin\theta_j \\ \sin\theta_j & \cos\theta_j \end{bmatrix}, \quad \theta_j = \begin{cases} \frac{\pi}{4}, & \text{if } j \text{ is even} \\ \frac{3\pi}{4}, & \text{otherwise} \end{cases}.$$

The sample $\boldsymbol{x}^{(i)}$ will be accepted if the constraint Eq. (25) is satisfied and rejected otherwise. This way, an elliptic hole is generated for any two adjacent dimensions of data points. The reference PDF is then defined as

$$p_{X,\text{ref}}(\boldsymbol{x}) = \frac{I_B(\boldsymbol{x}) \prod_{i=1}^{d} \rho(x_i; 0, s)}{\mathbb{E}[I_B(X)]}, \tag{26}$$

where B is the set defined by equation (25) and $I_B(\cdot)$ is an indicator function with $I_B(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in B$; 0, otherwise. For this test problem, we set $d = 8$, $\gamma = 3$ and $C = 7.6$. This case has been studied in [46], where the rotation layers and nonlinear layers are turned off. In [46] an algebraic convergence has been observed numerically for both the real NVP and the KRnet, where the convergence rate of KRnet is about twice as large as that of the real NVP. We here only demonstrate the effectiveness of the rotation layer and the nonlinear layer.

We now compare the performance of KRnet and real NVP numerically. In KRnet, we deactivate the dimensions by one, i.e., $K = 7$. We let $N_{\text{NN},k} = 0.9N_{\text{NN},k-1}$ by adjusting the width of the neural network $\text{NN}_{[i]}$, $i = 1, 2, 3$, which consists of two fully connected hidden layers of the same width. Other configurations of $\text{NN}_{[i]}$ can also be considered. One example is given

Table 1
The effects of rotation and nonlinear layers in KRnet. δ_I, δ_{II} and δ_{III} are relative errors of KRnet, respectively, for the aforementioned three stages. δ is the relative error of real NVP, whose depth is chosen to roughly match the DOFs of the KRnets from the same column. For the nonlinear layers, we use 32 nonuniform elements to decompose $[-30, 30]$, i.e., $a = 30$. Note that the rotation layers and the nonlinear layers do not introduce a significant increase in the total number of DOFs. The percentages in parentheses indicate the degree of drop in terms of δ_I.

KRnet	$L = 2$	$L = 4$	$L = 6$	$L = 8$
δ_I	7.54e-2	2.45e-2	1.44e-2	9.50e-3
δ_{II}	6.53e-2 (↓13%)	2.24e-2 (↓9%)	1.39e-2 (↓3%)	9.11e-3 (↓4%)
δ_{III}	4.93e-2 (↓35%)	1.95e-2 (↓20%)	1.26e-2 (↓13%)	8.34e-3 (↓12%)
Real NVP	$L = 10$	$L = 20$	$L = 32$	$L = 42$
δ	2.17e-2	1.98e-2	2.11e-2	2.05e-2

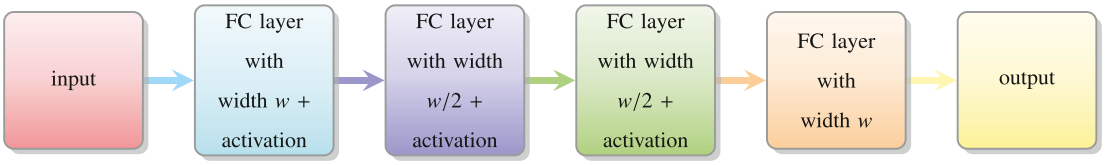


Fig. 2. The architecture of $\mathsf{NN}_{[i]}$ for affine coupling layers, for $i = 0, \ldots, L-1$ (FC layers refer to fully connected layers).

in Fig. 2, which is used in section 5. The neural network $\mathsf{NN}_{[i]}$ (for $i = 0, \ldots, L-1$) consists of three hidden layers and one linear layer, where the first hidden layer and the linear layer have w neurons, and the middle two layers have $w/2$ neurons. In this experiment, we combine the two middle hidden layers to one hidden layer with w neurons. We set $w = 24$ and use the rectified linear unit function (ReLU) as the activation function [50]. The depth of the real NVP will be determined by N_{dof} of the KRnet, since we split the dimensions into two halves in real NVP. The KRnet will be implemented as follows. We train KRnet with three stages and record the errors of each stage. In the first stage, we switch off both the rotation layers and the nonlinear layers and train the model for 8000 epochs; in the second stage, we switch on the rotation layers and restart the training process for another 2000 epochs; finally, we switch on both the rotation layers and the nonlinear layers and continue the training process for another 2000 epochs. For the real NVP, we simply run 8000 epochs. For each epoch, we compute the relative error

$$\delta = \frac{D_{KL}(p_{X,\text{ref}}(\boldsymbol{x}) || p_X(\boldsymbol{x}; \Theta))}{H(p_{X,\text{ref}}(\boldsymbol{x}))} \tag{27}$$

using the validation set, since the cross entropy should converge to the differential entropy of the reference PDF. We record the minimum relative error of all epoches. Furthermore, to reduce the bias of δ, we will sample 10 independent training sets and repeat the training process ten times to obtain an averaged relative error δ. The relative errors corresponding to the above three stages of training KRnet are denoted as δ_I, δ_{II} and δ_{III}. We will sample 3.2×10^5 data points for both the training set and the validation set. We employ the Adam optimizer [51] with learning rate 0.001 and batch size 80000.

The results of numerical experiments have been summarized in Table 1. First of all, both δ_{II} and δ_{III} are smaller than δ_I, indicating that the rotation layers and nonlinear layers are able to improve the model performance. Such an improvement is more noticeable for a smaller L. Second, for the specific setup of the numerical experiments, the errors δ_i, $i = I, II, III$, of the KRnet decay consistently as L increases while the errors of the real NVP do not show consistent decay. Since we compute the errors after 8000 epochs for all L, this shows that for a comparable model complexity the KRnet needs less epoches to obtain a substantial decrease in error than the real NVP. Third, as also shown in [46], the real NVP performance better than KRnet for a small L. The real NVP can be regarded as a KRnet with a half-half partition, i.e., $K = 2$ and $m = \frac{d}{2} = 4$. For a fixed complexity, the performance of KRnet depends on both K and L. In Fig. 3, we compare the approximated distributions given by the real NVP with $L = 42$ and the KRnet with $L = 8$, where both the rotation layers and the nonlinear layers are switched on.

4. Adaptive deep density approximation for the stationary Fokker-Planck equation

We intend to use KRnet as a PDF model to approximate the Fokker-Planck equation to alleviate the difficulties from the curse of dimensionality. In particular, we will develop an adaptive deep density approximation (ADDA) approach, which consists of two components: 1) solving the Fokker-Planck equation on a certain set of collocation points by a machine learning technique; 2) choosing a new set of collocation points to refine the current approximate solution. These two components are implemented alternately to achieve adaptivity such that both the accuracy and the efficiency will be improved.

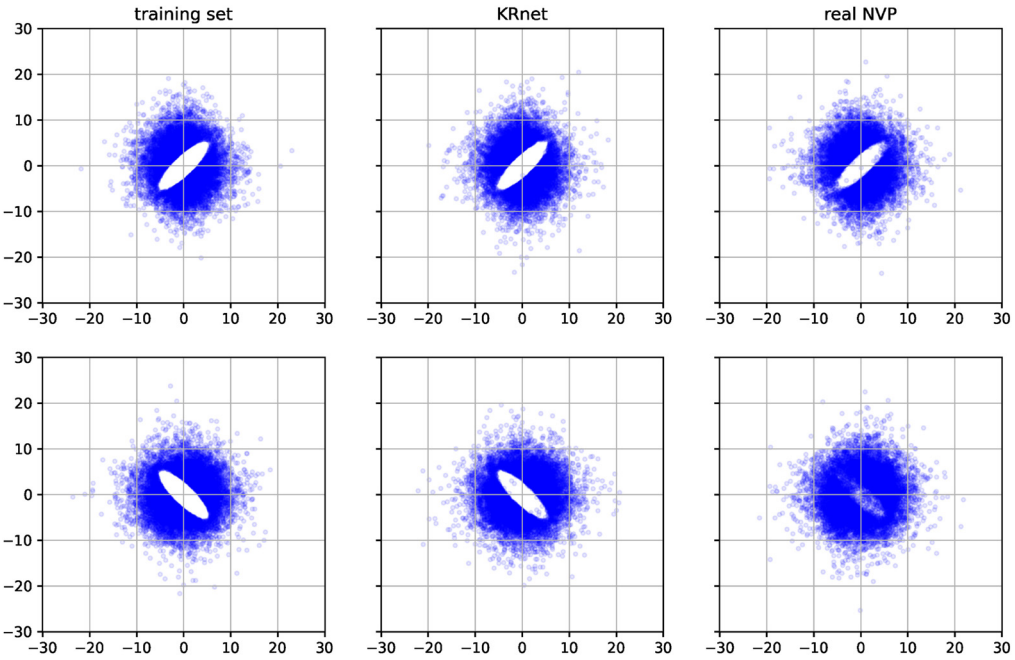


Fig. 3. Training data, and data sampled from KRnet and real NVP. The first row shows the components x_1 and x_2, and the second row shows the components x_4 and x_5. We pick the two pairs of adjacent dimensions, where the real NVP performs the best and the worst, respectively.

4.1. Stochastic gradient descent based on stochastic collocation points

Let $p_X(\boldsymbol{x};\Theta)$ be a probability density function associated with the random vector X, which is based on the KRnet. All the constraints in Eq. (4) and Eq. (5) are naturally satisfied since $p_X(\boldsymbol{x};\Theta)$ is a family of probability density functions, implying that the difficulties caused by the boundary conditions and the nonnegativity of PDF have disappeared. We seek to approximate the solution $p(\boldsymbol{x})$ of the Fokker-Planck equation by $p_X(\boldsymbol{x};\Theta)$ to take advantage of the weaker dependence of deep neural networks on dimensionality than traditional computational approaches such as the finite element methods [52–54].

The main idea of a machine learning approach to solve PDEs is to consider an optimization problem defined on a set of collocation points where the equation is constrained. Let $p_{\text{data}}(\boldsymbol{x})$ be a probability density function, based on which we define a loss functional

$$J\left(p_X(\boldsymbol{x};\Theta)\right) = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left(r^2(\boldsymbol{x};\Theta)\right) = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left(\mathcal{L}^2(p_X(\boldsymbol{x};\Theta))\right) \tag{28}$$

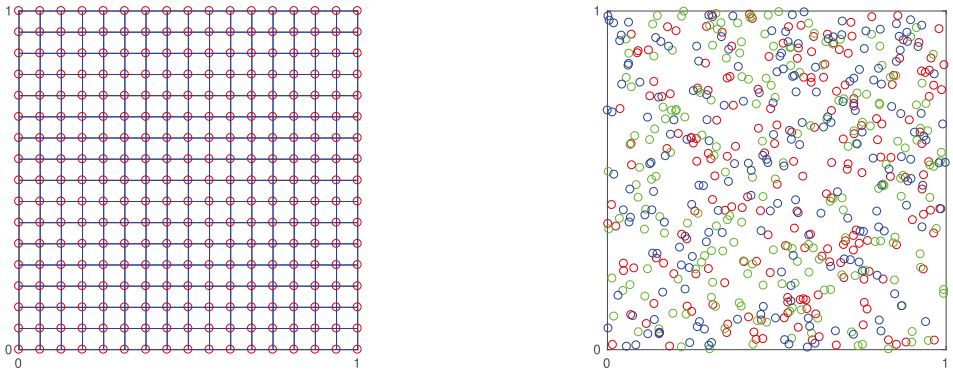
where $\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}$ denotes the expectation with respect to the training set, and r is the residual loss. The solution $p(\boldsymbol{x})$ of Eq. (3) can be approximated by $p_X(\boldsymbol{x};\Theta)$ through minimizing the loss functional $J(p_X(\boldsymbol{x};\Theta))$. In reality, we usually do not have much prior understanding about the residual, and simply assign $p_{\text{data}}(\boldsymbol{x})$ a simple distribution, e.g., a uniform distribution defined on a finite computational domain. We then use $p_{\text{data}}(\boldsymbol{x})$ to sample a set $\mathcal{C} = \{\boldsymbol{x}^{(i)}\}_{i=1}^N$ of collocation points to approximate the loss functional, i.e.,

$$\hat{J}\left(p_X(\boldsymbol{x};\Theta)\right) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}^2\left(p_X(\boldsymbol{x}^{(i)};\Theta)\right) \approx J\left(p_X(\boldsymbol{x};\Theta)\right), \tag{29}$$

based on which we choose the optimal parameter Θ^*:

$$\Theta^* = \arg\min_{\Theta} \hat{J}(p_X(\boldsymbol{x};\Theta)). \tag{30}$$

The optimization problem Eq. (30) will be solved by stochastic gradient-based optimization [55,51], which is summarized as follows. The set of collocation points can be divided into n_b mini-batches $\{\mathcal{C}_{i_b}\}_{i_b=1}^{n_b}$, where every mini-batch $\mathcal{C}_{i_b}$ contains m samples such that $N = mn_b$. Denoting the parameters at i_b-th iteration of a certain epoch j as $\Theta_{i_b}^{(j)}$, for every mini-batch $\mathcal{C}_{i_b}$ and $\boldsymbol{x}^{(l)} \in \mathcal{C}_{i_b}, l = 1,\ldots,m$, one can apply the mini-batch to estimate the expectation of the residual loss and the stochastic gradient, and then update the parameters Θ based on the following scheme



(a) FEM mesh (b) random samples

Fig. 4. An example of linear finite element meshes and stochastic collocation points in $[0,1]^2$.

$$\Theta_{i_b}^{(j)} = \Theta_{i_b-1}^{(j)} - \eta \nabla_\Theta \left[\frac{1}{m} \sum_{l=1}^{m} \left(r(\boldsymbol{x}^{(l)}; \Theta_{i_b-1}^{(j)}) \right)^2 \right] \quad \text{for } i_b = 1, \ldots, n_b,\ j = 1, 2, \ldots \tag{31}$$

where η is a given learning rate. Compared with the gradient descent method, the stochastic gradient descent method only requires computing the gradient on the mini-batch $\mathcal{C}_{i_b}$. In this work we employ the Adam optimizer, which is widely used to accelerate the training process for deep neural networks, as this method adopts adaptive learning rates for different components of parameters through estimates of first and second moments of the gradients [51].

4.2. Adaptive sampling procedure

Compared with the standard finite element methods (FEM) [5], the deep learning approach does not require mesh generation to solve PDEs, which shares more similarities to meshless methods, and the approximation of Eq. (28) fits naturally with stochastic gradient-based optimization. Fig. 4 shows a linear finite element mesh in $[0,1]^2$ and the collocation points that are generated with a uniform distribution in $[0,1]^2$.

Adaptivity plays an important role in classical numerical methods for the approximation of PDEs. Considering a finite element method subject to a certain mesh of the computation domain, we expect that the element-wise approximation errors are distributed in a nearly uniform way. This means that the most effective mesh should be non-uniform since the regularity of the solution varies in the computation domain. In our problem, the distribution $p_{\text{data}}(\boldsymbol{x})$ of the collocation points will affect the approximation of $J(p_X(\boldsymbol{x};\Theta))$ and the optimal parameter Θ^* as well. Apparently a uniform distribution is not an optimal choice for $p_{\text{data}}(\boldsymbol{x})$ especially for high-dimensional problems. For a certain amount of collocation points, the curse of dimensionality will weaken the contribution of each collocation point to our learning problem, which will be worsen for the approximation of PDF if the exact solution $p(\boldsymbol{x})$ is far away from being uniform. We then expect to use samples from a nonuniform distribution $p_{\text{data}}(\boldsymbol{x})$ for the approximation $J(p_X(\boldsymbol{x};\Theta))$, where a simple criterion is that $p_{\text{data}}(\boldsymbol{x})$ should be consistent with the true solution $p(\boldsymbol{x})$ to some extent. This will result in adaptive deep density approximation (ADDA) for the approximation of the Fokker-Planck equation.

It is, in general, difficult to generate samples that are adaptive to the true solution $p(\boldsymbol{x})$. Fortunately, flow-based deep generative models provide an opportunity for us to do this thanks to the invertible mapping. Our strategy is as follows. Starting with an initial set of collocation points $\mathcal{C}_0 = \left\{\boldsymbol{x}_{(0)}^{(i)}\right\}_{i=1}^N$ drawn from a uniform distribution, we train and obtain the KRnet $Z = f_{\text{KR},(0)}(X;\Theta^{*,(0)})$, which corresponds to the PDF $p_X^{(0)}(\boldsymbol{x};\Theta^{*,(0)})$. We then generate a new set $\mathcal{C}_1 = \left\{\boldsymbol{x}_{(1)}^{(i)}\right\}_{i=1}^N$ of collocation points by $X = f_{\text{KR},(0)}^{-1}(Z)$ using N samples from the prior distribution of Z. Then $\mathcal{C}_1$ is a set of samples from $p_X^{(0)}(\boldsymbol{x};\Theta^{*,(0)})$. We continue to update the KRnet using $\Theta^{*,(0)}$ as the initial parameters and $\mathcal{C}_1$ as the training set, which yields $f_{\text{KR},(1)}(X;\Theta^{*,(1)})$. Then another iteration starts. In general, we sample the current optimal PDF model $p_X^{(k)}(\boldsymbol{x};\Theta^{*,(k)})$ to generate a new training set $\mathcal{C}_{k+1} = \left\{\boldsymbol{x}_{(k+1)}^{(i)}\right\}_{i=1}^N$ and update the KRnet to $f_{\text{KR},(k+1)}(\boldsymbol{x};\Theta^{*,(k+1)})$. This way, the samples for the training process become more and more consistent with the true solution, if $p_X^{(k)}(\boldsymbol{x};\Theta^{*,(k)})$ approaches $p(\boldsymbol{x})$ as k increases. In other words, more collocation points will be chosen in the region of high density while less collocation points in the region of low density. Our adaptive training process has been summarized in Algorithm 1, where $N_{\text{adaptive}} \in \mathbb{N}$ is a given number of maximum adaptivity iterations, and this strategy is called the adaptive deep density approximation based on KRnet (ADDA-KR) from now on. The final KRnet-induced PDF is the ADDA-KR approximation for the steady state Fokker-Planck problem (3)–(5).

We note that the adaptivity in Algorithm 1 can be further tuned. One possible strategy is to update the training set gradually for each training stage, e.g., up to a certain percentage. In this work, we replace the whole training set from the previous stage just for simplicity.

4.3. Implementation issues

When minimizing the loss functional Eq. (29), numerical underflow issues can be encountered, especially when $\boldsymbol{x}$ is relatively high-dimensional. That is, the loss functional can be too small to provide an effective gradient descent direction. To alleviate this issue, we develop the following scaling strategy in our implementation. Multiplying both sides of equation (3) by a constant $C_s > 0$ gives

$$\mathcal{L}(C_s p(\boldsymbol{x})) = \nabla \cdot [C_s p(\boldsymbol{x}) \nabla V(\boldsymbol{x})] + \nabla \cdot [\nabla \cdot (C_s p(\boldsymbol{x}) \boldsymbol{D}(\boldsymbol{x}))] = 0. \tag{32}$$

The solution of the above equation is the same as the solution of the original stationary Fokker-Planck equation (3). However, if C_s is large enough, Eq. (32) is numerically more stable than Eq. (3), and the loss functional Eq. (29) associated with Eq. (32) can typically provide effective gradient descent directions to optimize the parameters Θ. In our practical implementation, we usually set $C_s = 100$.

Algorithm 1 Adaptive deep density approximation based on KRnet (ADDA-KR) for the Fokker-Planck equation.

Input: Initial KRnet $p_X^{(0)}(\boldsymbol{x}; \Theta_0^{(0)})$, maximum epoch number N_e, maximum iteration number N_{adaptive}, learning rate η, batch size m, and initial training set $\mathcal{C}_0 = \left\{\boldsymbol{x}_{(0)}^{(i)}\right\}_{i=1}^{N}$.

Divide $\mathcal{C}_0 = \left\{\boldsymbol{x}_{(0)}^{(i)}\right\}_{i=1}^{N}$ into n_b mini-batch $\{\mathcal{C}_{i_b}\}_{i_b=1}^{n_b}$.
for $k = 1 : N_{\text{adaptive}}$ **do**
  for $j = 0 : N_e - 1$ **do**
    for $i_b = 1 : n_b$ **do**
      Compute the values of the residual loss $r(\boldsymbol{x}_{(k-1)}^{(l)}; \Theta_{i_b-1}^{(j)})$ (see Eq. (29)) for $l = 1, \ldots, m$, on the mini-batch $\mathcal{C}_{i_b}$.
      Update the parameters $\Theta_{i_b}^{(j)}$ using the Adam optimizer with learning rate η.
    end for
    if $j = N_e - 1$ **then**
      Let $\Theta^{*,(k)} := \Theta_{n_b}^{(N_e-1)}$.
    else
      Let $\Theta_0^{(j+1)} := \Theta_{n_b}^{(j)}$.
    end if
    Shuffle the set of collocation points $\mathcal{C}_{k-1} = \left\{\boldsymbol{x}_{(k-1)}^{(i)}\right\}_{i=1}^{N}$.
    Divide $\mathcal{C}_{k-1} = \left\{\boldsymbol{x}_{(k-1)}^{(i)}\right\}_{i=1}^{N}$ into n_b mini-batch $\{\mathcal{C}_{i_b}\}_{i_b=1}^{n_b}$.
  end for
  if $k = N_{\text{adaptive}}$ **then**
    Let $\Theta := \Theta^{*,(k)}$.
  else
    Generate $\mathcal{C}_{k+1} = \left\{\boldsymbol{x}_{(k+1)}^{(i)}\right\}_{i=1}^{N}$ by $p_X^{(k)}(\boldsymbol{x}; \Theta^{*,(k)})$.
    Let $\Theta_0^{(0)} := \Theta^{*,(k)}$.
  end if
end for
Obtain the ADDA-KR solution $p_X(\boldsymbol{x}; \Theta) := p_X^{(N_{\text{adaptive}})}(\boldsymbol{x}; \Theta)$.

Output: The ADDA-KR solution $p_X(\boldsymbol{x}; \Theta)$.

5. Numerical study

In this section, numerical experiments are conducted to illustrate the effectiveness of our ADDA-KR (adaptive deep density approximation based on KRnet) approach presented in Algorithm 1. Five test problems for the Fokker-Planck equation are studied—one one-dimensional test problem, two two-dimensional test problems (one is a single modal distribution, and the other is a bimodal distribution), one four-dimensional test problem, and one eight-dimensional test problem. The activation function of $\text{NN}_{[i]}$ (see Eq. (10)) is set to the hyperbolic tangent function for all test problems. For comparison, we also test the performance of a direct adaptive version of classic real NVP, and as the real NVP utilizes a half-half partition (see section 3.5), we refer to it as ADDA-HH. The implementation of ADDA-HH is to replace the KRnet in ADDA-KR (Algorithm 1) by the classical real NVP, and we set the same input parameters for both ADDA-KR and ADDA-HH in all our test problems. In addition, results of non-adaptive versions of KRnet and real NVP are included for high-dimensional test problems (the four-dimensional and the eight-dimensional test problems), which are referred to as Uniform-KR and Uniform-HH. In Uniform-KR and Uniform-HH, collocation points are generated through uniform distributions, and other settings of KRnet and real NVP are the same as the settings for ADDA-KR in these test problems.

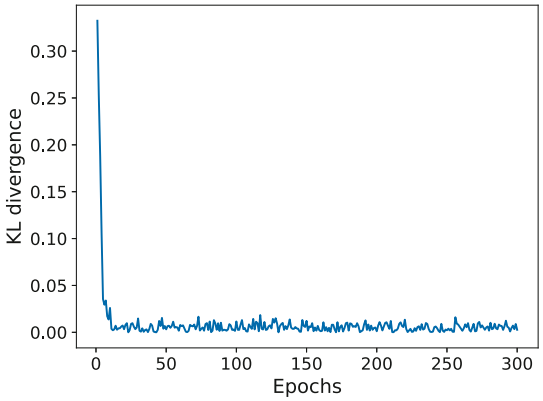


Fig. 5. The KL divergence with respect to epochs, one-dimensional test problem.

5.1. A one-dimensional test problem

We start with this one-dimensional case, where the governing equation is

$$\begin{aligned} &\frac{\partial(xp(x))}{\partial x}+\frac{1}{2}\frac{\partial^2(p(x))}{\partial x^2}=0,\\ &\int_{\mathbb{R}} p(x)dx=1,\ p(x)\geq 0, \end{aligned} \tag{33}$$

and the exact solution is

$$p(x)=\frac{\exp(-x^2)}{\sqrt{\pi}}. \tag{34}$$

For this one-dimensional problem, KRnet is the same as the classical real NVP, meaning that only the affine coupling layers are needed. As the affine coupling layers (see section 3.1) need at least two-dimensions, we use $[x, x]$ as an input in our implementation of KRnet. We generate the initial parameters $\Theta_0^{(0)}$ for the inputs of Algorithm 1, using Glorot Gaussian initialization [56], and then construct the initial KRnet $p_X^{(0)}(\boldsymbol{x};\Theta_0^{(0)})$. The number of epochs is set to $N_e=300$, and only one adaptivity iteration is conducted for this one-dimensional problem, i.e., $N_{\text{adaptive}}=1$. The learning rate for Adam optimizer is set to $\eta=0.0002$, and the batch size is set to $m=500$. The initial training set $\mathcal{C}_0$ is generated through the uniform distribution with range $[-5,5]$, and the sample size is set to $|\mathcal{C}_k|=3000$ for each iteration step k for $k=0,\ldots,N_{\text{adaptive}}$. In addition, we take $L=8$ affine coupling layers, and two fully connected layers with $w=48$ neurons for $\mathsf{NN}_{[i]}$ (see Eq. (10)).

To assess the accuracy of our ADDA-KR approach (Algorithm 1), we compute the KL divergence between the exact solution $p(x)$ and our ADDA-KR solution $p_X(x;\Theta)$:

$$\begin{aligned} D_{KL}(p(x)||p_X(x;\Theta)) &= \int_{-\infty}^{\infty} p(x)\log p(x)dx-\int_{-\infty}^{\infty} p(x)\log p_X(x;\Theta)dx\\ &= -\frac{1}{2}(1+\log\pi)-\int_{-\infty}^{\infty} p(x)\log p_X(x;\Theta)dx \end{aligned}$$

where the last term of the above equation is approximated by Monte Carlo integration with 10^4 samples. Fig. 5 shows the KL divergence decreases to zero quickly. Fig. 6 shows the exact solution $p(x)$ and our ADDA-KR solution $p_X(x;\Theta)$, where it can be seen that they are visually indistinguishable.

5.2. Two-dimensional test problems

In this part, two-dimensional Fokker-Planck equations are considered, where the solution of the first one is a single modal distribution and the solution of the second one is a bimodal distribution.

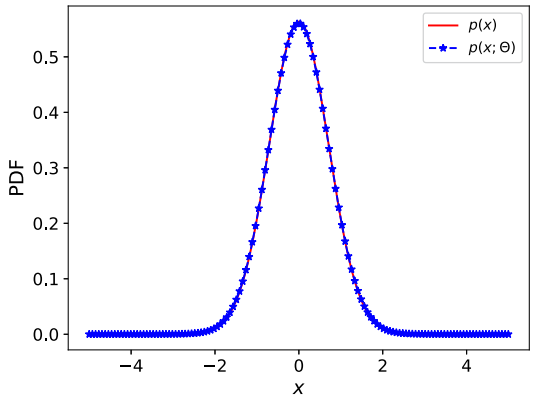


Fig. 6. The exact solution and the ADDA-KR solution, one-dimensional test problem.

5.2.1. Two-dimensional single modal distribution

The stationary Fokker-Planck equation for this test problem is

$$\begin{aligned} &\nabla \cdot [p(\boldsymbol{x})\boldsymbol{A}\boldsymbol{x}] + \nabla \cdot [\nabla \cdot (p(\boldsymbol{x})\boldsymbol{D}] = 0, \\ &\int_{\mathbb{R}^d} p(\boldsymbol{x})d\boldsymbol{x} = 1, \ p(\boldsymbol{x}) \geq 0, \end{aligned} \tag{35}$$

where $\boldsymbol{A}$ and $\boldsymbol{D}$ are two constant matrices. This equation is corresponding to the following Ornstein-Uhlenbeck process

$$dX_t = -\boldsymbol{A}X_t dt + \boldsymbol{G}d\boldsymbol{w}_t, \tag{36}$$

where $\boldsymbol{D} = \boldsymbol{G}\boldsymbol{G}^{\mathsf{T}}/2$.

The solution of Eq. (35) exists if the real parts of the eigenvalues of $\boldsymbol{A}$ are larger than zero [1], and it can be written as

$$p(\boldsymbol{x}) = (2\pi)^{-1}(\det\boldsymbol{\Sigma})^{-\frac{1}{2}}\exp(-\frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}), \tag{37}$$

where the covariance matrix $\boldsymbol{\Sigma}$ is determined by the following Lyapunov equation

$$\boldsymbol{A}\boldsymbol{\Sigma} + \boldsymbol{\Sigma}\boldsymbol{A}^{\mathsf{T}} = 2\boldsymbol{D}. \tag{38}$$

The above Lyapunov equation has a unique solution if and only if the eigenvalues λ_i of $\boldsymbol{A}$ satisfy $\lambda_i \neq -\lambda_j$ for all $i, j = 1, 2$. In this test problem, the constant matrix $\boldsymbol{A}$ for the drift term and the diffusion matrix $\boldsymbol{D}$ are set to

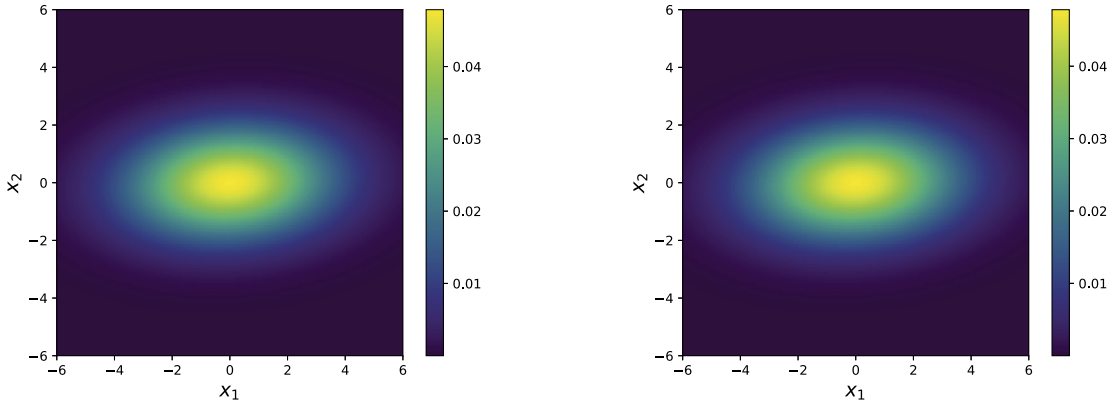
$$\boldsymbol{A} = \begin{bmatrix} 1.37096037 & -0.48306187 \\ -0.48306187 & 1.62903963 \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} 22.52429192 & -6.55821381 \\ -6.55821381 & 12.68972 \end{bmatrix},$$

which implies that the covariance matrix $\boldsymbol{\Sigma}$ is

$$\boldsymbol{\Sigma} = \begin{bmatrix} 8.12186142 & -0.26372569 \\ -0.26372569 & 3.81664391 \end{bmatrix}.$$

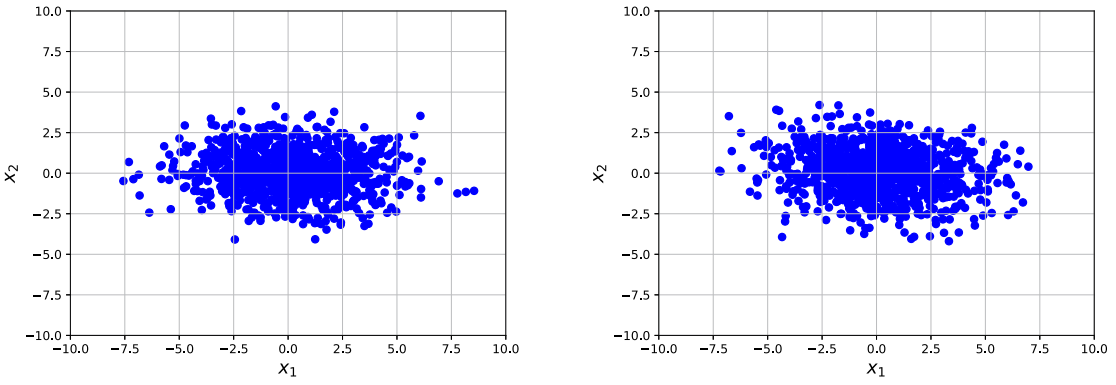
We generate the initial parameters $\Theta_0^{(0)}$ with Glorot Gaussian initialization [56], and then construct the initial KRnet $p_X^{(0)}(\boldsymbol{x}; \Theta_0^{(0)})$ for Algorithm 1. The number of epochs is set to $N_e = 300$, and two adaptivity iterations are conducted for this problem, i.e., $N_{\text{adaptive}} = 2$. The learning rate for Adam optimizer is set to $\eta = 0.0002$, and the batch size is set to $m = 1000$. The initial training set $\mathcal{C}_0$ is generated through the uniform distribution with range $[-6, 6]^2$, and the sample size is set to $|\mathcal{C}_k| = 6 \times 10^4$ for each iteration step k for $k = 0, \ldots, N_{\text{adaptive}}$. In addition, we take $L = 8$ affine coupling layers, and two fully connected layers with $w = 48$ neurons for $\mathsf{NN}_{[i]}$ (see Eq. (10)).

Fig. 7 shows the exact solution $p(\boldsymbol{x})$ and our ADDA-KR solution $p_X(\boldsymbol{x}; \Theta)$, where it can be seen that they are visually indistinguishable. For this test problem, there is no significant difference between the ADDA-KR solution and the ADDA-HH solution, and we then only show the exact solution and our ADDA-KR solution. Fig. 8 shows samples drawn from the exact solution of Eq. (35) and our ADDA-KR solution, which confirms that the corresponding distributions ($p(x)$ and $p_X(\boldsymbol{x}; \Theta)$) are very close.



(a) Exact solution $p(\boldsymbol{x})$ (b) ADDA-KR solution $p_X(\boldsymbol{x};\Theta)$

Fig. 7. Solutions, two-dimensional single modal test problem.



(a) Samples of the exact solution $p(\boldsymbol{x})$. (b) Samples of the ADDA-KR solution $p_X(\boldsymbol{x};\Theta)$.

Fig. 8. Samples, two-dimensional single modal test problem.

5.2.2. Two-dimensional bimodal distribution

In this test problem, the Fokker-Planck equation considered is

$$\begin{aligned} -\nabla\cdot[p(\boldsymbol{x})\nabla\log(\beta_1 p_1(\boldsymbol{x})+\beta_2 p_2(\boldsymbol{x}))]+\nabla^2 p(\boldsymbol{x})=0,\\ \int_{\mathbb{R}^d} p(\boldsymbol{x})d\boldsymbol{x}=1,\ \ p(\boldsymbol{x})\geq 0, \end{aligned} \tag{39}$$

where for $k=1,2$, each $p_k(\boldsymbol{x})$ is the probability density function of the normal distribution with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$, and $\beta_1+\beta_2=1$. The solution of Eq. (39) is the following Gaussian mixture distribution [47, p. 123],

$$p(\boldsymbol{x})=\beta_1 p_1(\boldsymbol{x})+\beta_2 p_2(\boldsymbol{x}). \tag{40}$$

Here, we set $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and β_k for $k=1,2$ as

$$\begin{aligned} &\beta_1=0.55,\ \beta_2=0.45,\quad \boldsymbol{\mu}_1=[-1,-1]^{\mathsf{T}},\ \boldsymbol{\mu}_2=[2,2]^{\mathsf{T}}\\ &\boldsymbol{\Sigma}_1=\begin{bmatrix} 6.12186142 & -0.26372569\\ -0.26372569 & 1.81664391\end{bmatrix},\quad \boldsymbol{\Sigma}_2=\begin{bmatrix} 2.8828528 & -0.70234742\\ -0.70234742 & 2.69199911\end{bmatrix}. \end{aligned} \tag{41}$$

The matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are positive definite, and their entries are randomly constructed.

We again generate the initial parameters $\Theta_0^{(0)}$ with Glorot Gaussian initialization, and then construct the initial KRnet $p_X^{(0)}(\boldsymbol{x};\Theta_0^{(0)})$. The number of epochs is set to $N_e=200$, and the maximum number of adaptivity iterations conducted for this problem is set to $N_{\text{adaptive}}=5$. The learning rate for Adam optimizer is set to $\eta=0.0001$, and the batch size is set to $m=1000$. The initial training set $\mathcal{C}_0$ is generated through the uniform distribution with range $[-5,5]^2$, and the sample size

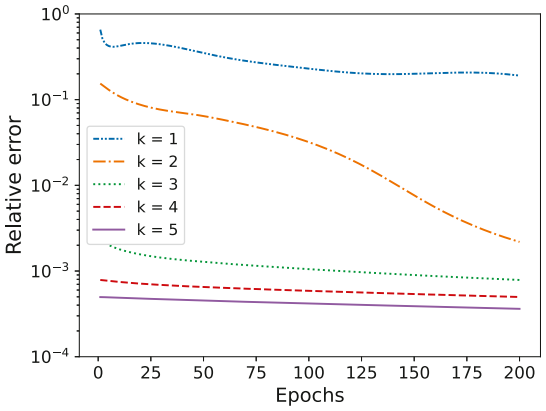


Fig. 9. The relative error for ADDA-KR, two-dimensional bimodal test problem.

is set to $|\mathcal{C}_k| = 6 \times 10^4$ for each iteration step k for $k = 0, \ldots, N_{\text{adaptive}}$. In addition, we take $L = 8$ affine coupling layers for both KRnet and real NVP, and two fully connected layers with $w = 48$ neurons for $\text{NN}_{[i]}$ (see Eq. (10)). For KRnet, we set $K = 2$ to focus on the effectiveness of the rotation layer and the nonlinear layer for this test problem. To assess the effectiveness of our ADDA-KR approach, we generate a validation data set $\mathcal{C}_v = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N_v}$, and compute the relative error defined by Eq. (27). The KL divergence is approximated by Monte Carlo integration

$$D_{KL}(p(\boldsymbol{x})||p_X(\boldsymbol{x};\Theta)) \approx \frac{1}{N_v}\sum_{i=1}^{N_v}\Big(\log p(\boldsymbol{x}^{(i)}) - \log p(\boldsymbol{x}^{(i)};\Theta)\Big), \tag{42}$$

where $\boldsymbol{x}^{(i)}$ is drawn from the exact solution $p(\boldsymbol{x})$, and the size of the validation data set is set to 3.2×10^5 such that the KL-divergence can be approximated well.

Fig. 9 shows the relative error between the exact solution $p(\boldsymbol{x})$ and our ADDA-KR solution $p_X(\boldsymbol{x};\Theta)$ at each adaptivity iteration step k. It is clear that, as the adaptivity iteration step increases, the relative error decreases quickly. In addition, it can be seen that as the number of epochs increases, the relative error decreases. Fig. 10 shows the comparison between our ADDA-KR and ADDA-HH. From Fig. 10(a), it can be seen that the relative error of ADDA-KR is smaller than that of ADDA-HH at each adaptivity iteration step. Fig. 10(b), Fig. 10(c) and Fig. 10(d) show the relative error decreases as the number of epochs increases, at adaptivity iteration steps $k = 1, 3, 5$ respectively. It can be seen that the relative error of ADDA-KR is clearly smaller than that of ADDA-HH for each value of epochs, except for the situations that the epoch number is smaller than 125 at the first adaptivity iteration in 10(b). Fig. 11 shows the exact solution $p(\boldsymbol{x})$ and the ADDA-KR solution $p_X(\boldsymbol{x};\Theta)$, where it can be seen that this bimodal distribution is well approximated by our ADDA-KR solution.

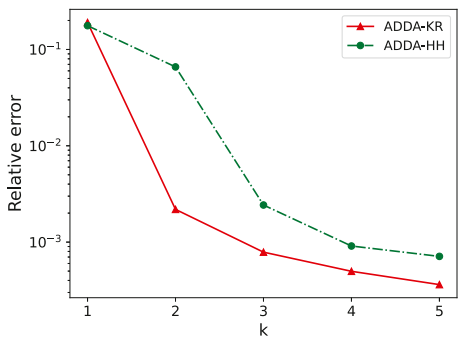
5.3. High-dimensional bimodal distributions (four-dimensional and eight-dimensional test problems)

In this part, we again consider the Fokker-Planck equation with two peaks Eq. (39) and set $\beta_1 = 0.55, \beta_2 = 0.45$. However, the dimensionality of the problem considered in this part is different from section 5.2. We here consider a four-dimensional ($d = 4$) problem and an eight-dimensional ($d = 8$) problem. The exact solution of Eq. (39) is a Gaussian mixture distribution Eq. (40). For $d = 4$, we set

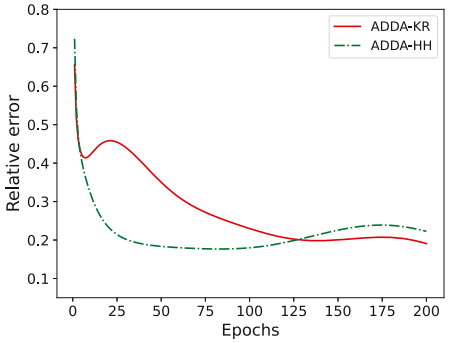
$$\begin{aligned} &\boldsymbol{\mu}_1 = [-1, -1, -0.3, -0.3]^{\mathsf{T}}, \boldsymbol{\mu}_2 = [2, 2, 0.6, 0.6]^{\mathsf{T}} \\ &\boldsymbol{\Sigma}_1' = \begin{bmatrix} \boldsymbol{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & 0.6\boldsymbol{\Sigma}_1 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2' = \begin{bmatrix} \boldsymbol{\Sigma}_2 & \mathbf{0} \\ \mathbf{0} & 0.6\boldsymbol{\Sigma}_2 \end{bmatrix}, \end{aligned} \tag{43}$$

where $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are given in Eq. (41), and $\boldsymbol{\Sigma}_1'$ and $\boldsymbol{\Sigma}_2'$ are the covariance matrices of p_1 and p_2 for this test problem.

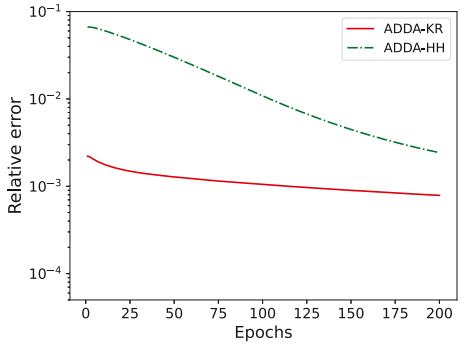
Similarly to the previous settings, we generate the initial parameters $\Theta_0^{(0)}$ with Glorot Gaussian initialization, and then construct the initial KRnet $p_X^{(0)}(\boldsymbol{x};\Theta_0^{(0)})$. The number of epochs is set to $N_e = 1$, and the number of adaptivity iterations conducted for this problem is set to $N_{\text{adaptive}} = 16$. Here, KRnet is trained and sampled in an interleaved manner. That is for both the four-dimensional and the eight-dimensional test problems, samples $\mathcal{C}_k$ drawn at the k-th adaptivity iteration are immediately used for training KRnet at the $(k+1)$-th iteration, while $p_X^{(k+1)}(\boldsymbol{x};\Theta)$ is immediately used for sampling. The learning rate for Adam optimizer is set to $\eta = 0.0001$, and the batch size is set to $m = 500$. The initial training set $\mathcal{C}_0$ is generated through the uniform distribution with range $[-6, 6]^4$, and two cases of the collocation sample size are considered: one is 10^5 and the other is 2×10^5. In addition, we take $L = 8$ affine coupling layers for KRnet, and $L = 16$ for real NVP. The architecture of $\text{NN}_{[i]}$ is the same as that shown in Fig. 2 with $w = 120$. For KRnet, we set $K = 3$. The



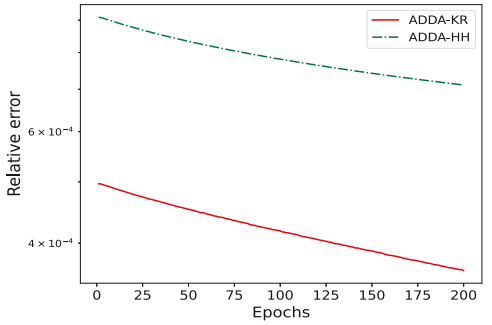
(a) Relative error w.r.t. adaptivity iteration step k.



(b) Relative error at $k = 1$.

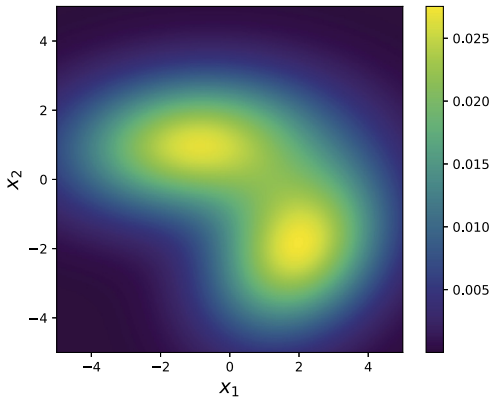


(c) Relative error at $k = 3$.

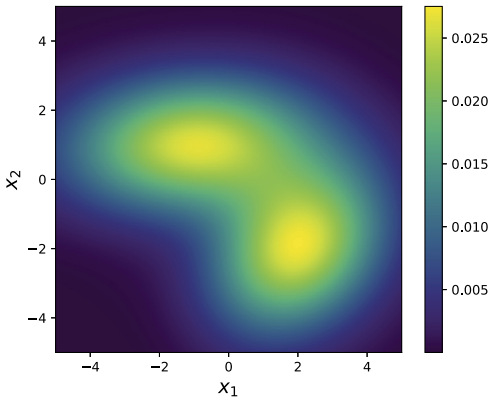


(d) Relative error at $k = 5$.

Fig. 10. The relative error for ADDA-KR and ADDA-HH, two-dimensional bimodal test problem.



(a) Exact solution $p(\boldsymbol{x})$

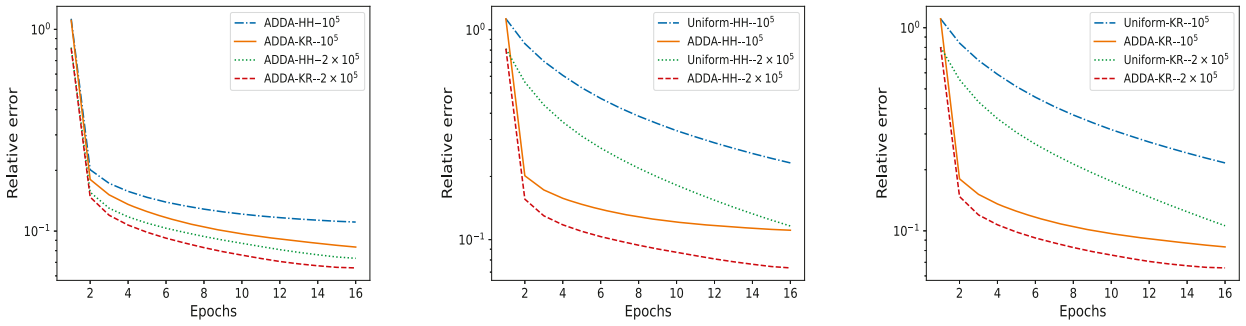


(b) ADDA-KR solution $p_X(\boldsymbol{x};\Theta)$

Fig. 11. Solutions, two-dimensional bimodal test problem.

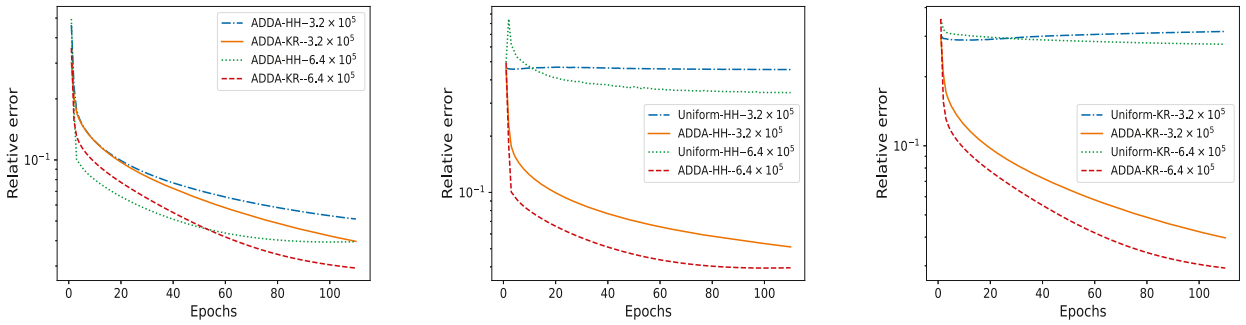
rotation layer and the nonlinear layer are turned on. To assess the accuracy of ADDA-KR, we again compute the relative error Eq. (27) between $p(\boldsymbol{x})$ and $p_X(\boldsymbol{x};\Theta)$ using 3.2×10^5 validation samples drawn from the exact solution.

Fig. 12 shows the relative error between $p(\boldsymbol{x})$ and $p_X(\boldsymbol{x};\Theta)$ for ADDA-KR and ADDA-HH, where different numbers of collocation points are considered. From Fig. 12(a), it can be seen that the relative error of ADDA-KR is smaller than that of ADDA-HH. From Fig. 12(b) and Fig. 12(c), as the number of epochs increases, the relative errors of ADDA-KR and ADDA-HH decrease quickly, while the relative errors of the uniform sampling strategies (Uniform-KR and Uniform-HH) decrease slowly. In addition, it can be seen that the relative error decreases as the number of training points increases from 10^5 to 2×10^5 for ADDA-KR, ADDA-HH, Uniform-KR and Uniform-HH.



(a) ADDA-KR and ADDA-HH (b) ADDA-HH and Uniform-HH (c) ADDA-KR and Uniform-KR

Fig. 12. Relative errors, four-dimensional test problem.



(a) ADDA-KR and ADDA-HH (b) ADDA-HH and Uniform-HH (c) ADDA-KR and Uniform-KR

Fig. 13. Relative errors, eight-dimensional test problem.

Finally, we consider an eight-dimensional bimodal distribution. For this problem, we set

$$\boldsymbol{\mu}_1 = [-1,-1,-0.3,-0.3,-0.4,-0.4,-1.6,-1.6]^{\mathsf{T}},\ \boldsymbol{\mu}_2 = [2,2,0.6,0.6,0.8,0.8,2.3,2.3]^{\mathsf{T}}$$

$$\tilde{\boldsymbol{\Sigma}}_1 = \begin{bmatrix} \boldsymbol{\Sigma}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 0.6\boldsymbol{\Sigma}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 0.8\boldsymbol{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 1.2\boldsymbol{\Sigma}_1 \end{bmatrix},\quad \tilde{\boldsymbol{\Sigma}}_2 = \begin{bmatrix} \boldsymbol{\Sigma}_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 0.6\boldsymbol{\Sigma}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 0.8\boldsymbol{\Sigma}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 1.2\boldsymbol{\Sigma}_2 \end{bmatrix}, \tag{44}$$

where $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are given in Eq. (41), and $\tilde{\boldsymbol{\Sigma}}_1$ and $\tilde{\boldsymbol{\Sigma}}_2$ are the covariance matrices of p_1 and p_2 for this test problem.

Again, we generate the initial parameters $\Theta_0^{(0)}$ with Glorot Gaussian initialization, and then construct the initial KRnet $p_X^{(0)}(\boldsymbol{x};\Theta_0^{(0)})$. The number of epochs is set to $N_e = 1$, and the maximum number of adaptivity iterations conducted for this problem is set to $N_{\text{adaptive}} = 120$. The learning rate for Adam optimizer is set to $\eta = 0.0001$, and the batch size is set to $m = 4000$. The initial training set $\mathcal{C}_0$ is generated through the uniform distribution with range $[-6,6]^8$, and two cases of the collocation sample size are considered: one is 3.2×10^5 and the other is 6.4×10^5. In addition, we take $L = 10$ affine coupling layers for KR, and $L = 20$ for real NVP. The architecture of $\text{NN}_{[i]}$ is the same as that shown in Fig. 2 with $w = 160$. For KRnet, we set $K = 3$. The rotation layer and the nonlinear layer are turned on. We again compute the relative error Eq. (27) using 3.2×10^5 validation samples drawn from the exact solution.

Fig. 13 shows the relative error between $p(\boldsymbol{x})$ and $p_X(\boldsymbol{x};\Theta)$ for ADDA-KR and ADDA-HH. From Fig. 13(a), it can be seen that the relative error of ADDA-KR is smaller than that of ADDA-HH, when the number of epochs is larger than 60. From Fig. 13(b) and Fig. 13(c), as the number of epochs increases, the relative errors of ADDA-KR and ADDA-HH decrease quickly, while the relative errors of the uniform sampling strategies (Uniform-KR and Uniform-HH) decrease slowly. In addition, it can be seen that the relative error decreases as the number of collocation points increases from 3.2×10^5 to 6.4×10^5 for ADDA-KR, ADDA-HH, Uniform-KR and Uniform-HH.

6. Conclusions

Conducting adaptivity is of fundamental importance for the efficient approximation of high-dimensional Fokker-Planck equations. With a focus on deep learning methods, we have developed an adaptive deep density approximation strategy

based on KRnet (ADDA-KR) in this work. Our KRnet, which is built on a block-triangular structure inspired by the Knothe-Rosenblatt rearrangement, gives an explicit family of probability density functions, which can serve as solution candidates of the Fokker-Planck equation. We also showed that KRnet is effective for estimating high-dimensional density functions in general. The fact that KRnet can efficiently generate samples integrates the two main steps in our ADDA-KR strategy to achieve efficient iterations: train KRnet for the Fokker-Planck equation with current collocation points, and generate new collocation points using the KRnet for the next iteration. Compared to real NVP, which is a widely used generative model, numerical results show that our ADDA-KR gives much more accurate numerical solutions for the Fokker-Planck equation. ADDA-KR in general works very well for Fokker-Planck equations with dimension of $O(10)$. For higher-dimensional cases, the sparsity of high-dimensional data will induce more severe difficulties, where we may need to consider dimension reduction to adapt more problem properties into the algorithm.

CRediT authorship contribution statement

Kejun Tang: Programming, Methodology, Writing-Original draft preparation. **Xiaoliang Wan:** Conceptualization, Methodology, Programming, Writing. **Qifeng Liao:** Conceptualization, Methodology, Writing-Reviewing and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

K. Tang and Q. Liao are supported by the National Natural Science Foundation of China (No. 12071291) and the Science and Technology Commission of Shanghai Municipality (No. 20JC1414300), and X. Wan's work was supported by the National Science Foundation under grant DMS-1913163.

References

[1] H. Risken, Fokker-Planck-Kolmogorov Equation, Springer, 1984.
[2] S. Jin, B. Yan, A class of asymptotic-preserving schemes for the Fokker–Planck–Landau equation, J. Comput. Phys. 230 (2011) 6420–6437.
[3] Y. Li, A data-driven method for the steady state of randomly perturbed dynamics, Commun. Math. Sci. 17 (2019) 1045–1059.
[4] B. Spencer, L. Bergman, On the numerical solution of the Fokker-Planck equation for nonlinear stochastic systems, Nonlinear Dyn. 4 (4) (1993) 357–372.
[5] H.C. Elman, D.J. Silvester, A.J. Wathen, Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics, Oxford University Press, USA, 2014.
[6] M. Dobson, Y. Li, J. Zhai, An efficient data-driven solver for Fokker-Planck equations: algorithm and analysis, arXiv:1906.02600, 2019.
[7] 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.
[8] W. E, A proposal on machine learning via dynamical systems, Commun. Math. Stat. 5 (1) (2017) 1–11.
[9] W. E, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[10] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations, arXiv:1711.10561, 2017.
[11] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part II): data-driven discovery of nonlinear partial differential equations, arXiv:1711.10566, 2017.
[12] M. Raissi, P. Perdikaris, 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 (2019) 686–707.
[13] G. Pang, L. Lu, G.E. Karniadakis, fPINNs: fractional physics-informed neural networks, SIAM J. Sci. Comput. 41 (4) (2019) A2603–A2626.
[14] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[15] Y. Zhu, N. Zabaras, Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, J. Comput. Phys. 366 (2018) 415–447.
[16] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, J. Comput. Phys. 394 (2019) 56–81.
[17] K. Wu, D. Xiu, Numerical aspects for approximating governing equations using data, J. Comput. Phys. 384 (2019) 200–221.
[18] K. Wu, T. Qin, D. Xiu, Structure-preserving method for reconstructing unknown Hamiltonian systems from trajectory data, SIAM J. Sci. Comput. 42 (6) (2020) A3704–A3729.
[19] K. Li, K. Tang, T. Wu, Q. Liao, D3M: a deep domain decomposition method for partial differential equations, IEEE Access 8 (2020) 5283–5294.
[20] A.D. Jagtap, E. Kharazmi, G.E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems, Comput. Methods Appl. Mech. Eng. 365 (2020) 113028.
[21] S. Dong, Z. Li, Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations, Comput. Methods Appl. Mech. Eng. 387 (2021) 114129.
[22] W. Li, X. Xiang, Y. Xu, Deep domain decomposition method: elliptic problems, in: J. Lu, R. Ward (Eds.), Proceedings of the First Mathematical and Scientific Machine Learning Conference, in: Proceedings of Machine Learning Research, PMLR, vol. 107, Princeton University, Princeton, NJ, USA, 2020, pp. 269–286.
[23] A. Heinlein, A. Klawonn, M. Lanser, J. Weber, Combining machine learning and domain decomposition methods—a review, Technical report, Universität zu Köln, October 2020.
[24] E. Kharazmi, Z. Zhang, G.E. Karniadakis, hp-VPINNs: variational physics-informed neural networks with domain decomposition, Comput. Methods Appl. Mech. Eng. 374 (2021) 113547.

[25] H. Sheng, C. Yang, PFNN: a penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries, J. Comput. Phys. (2020) 110085.
[26] H. Gao, L. Sun, J.-X. Wang, Phygeonet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, J. Comput. Phys. 428 (2021) 110079.
[27] D. Xiu, Numerical Methods for Stochastic Computations: A Spectral Method Approach, Princeton University Press, 2010.
[28] D. Xiu, J.S. Hesthaven, High-order collocation methods for differential equations with random inputs, SIAM J. Sci. Comput. 27 (3) (2005) 1118–1139.
[29] I. Babuška, F. Nobile, R. Tempone, A stochastic collocation method for elliptic partial differential equations with random input data, SIAM J. Numer. Anal. 45 (3) (2007) 1005–1034.
[30] J. Foo, X. Wan, G.E. Karniadakis, The multi-element probabilistic collocation method (ME-PCM): error analysis and applications, J. Comput. Phys. 227 (22) (2008) 9572–9595.
[31] X. Ma, N. Zabaras, An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations, J. Comput. Phys. 228 (8) (2009) 3084–3113.
[32] A. Narayan, D. Xiu, Stochastic collocation methods on unstructured grids in high dimensions via interpolation, SIAM J. Sci. Comput. 34 (3) (2012) A1729–A1752.
[33] H. Lei, X. Yang, B. Zheng, G. Lin, N.A. Baker, Constructing surrogate models of complex systems with enhanced sparsity: quantifying the influence of conformational uncertainty in biomolecular solvation, Multiscale Model. Simul. 13 (4) (2015) 1327–1353.
[34] P. Ren, Y. Xiao, X. Cheang, P.-Y. Huang, Z. Li, X. Chen, X. Wang, A survey of deep active learning, arXiv:2009.00236v1, 2020.
[35] R. Cang, H. Yao, Y. Ren, One-shot generation of near-optimal topology through theory-driven machine learning, Comput. Aided Des. 109 (2019) 12–21.
[36] R. Vilalta, Y. Drissi, A perspective view and survey of meta-learning, Artif. Intell. Rev. 18 (2001) 77–95.
[37] D.W. Scott, Multivariate Density Estimation: Theory, Practice, and Visualization, John Wiley & Sons, 2015.
[38] L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using real NVP, arXiv:1605.08803, 2016.
[39] D.P. Kingma, P. Dhariwal Glow, Generative flow with invertible 1×1 convolutions, in: Advances in Neural Information Processing Systems, 2018, pp. 10215–10224.
[40] L. Zhang, W. E, L. Wang, Monge-Ampère flow for generative modeling, arXiv:1809.10188, 2018.
[41] T.Q. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, in: Advances in Neural Information Processing Systems, 2018, pp. 6571–6583.
[42] G. Carlier, A. Galichon, F. Santambrogio, From Knothe's transport to Brenier's map and a continuation method for optimal transport, SIAM J. Math. Anal. 41 (6) (2010) 2554–2576.
[43] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[44] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, Stat 1050 (2014) 1.
[45] X. Wan, S. Wei, Coupling the reduced-order model and the generative model for an importance sampling estimator, J. Comput. Phys. 408 (2020) 109281.
[46] K. Tang, X. Wan, Q. Liao, Deep density estimation via invertible block-triangular mapping, Theor. Appl. Mech. Lett. 10 (2020) 143.
[47] G.A. Pavliotis, Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations, vol. 60, Springer, 2014.
[48] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[49] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, arXiv:1502.03167, 2015.
[50] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.
[51] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, arXiv:1412.6980, 2014.
[52] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. 2 (4) (1989) 303–314.
[53] M. Leshno, V.Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Netw. 6 (6) (1993) 861–867.
[54] Z. Lu, H. Pu, F. Wang, Z. Hu, L. Wang, The expressive power of neural networks: a view from the width, in: Advances in Neural Information Processing Systems, 2017, pp. 6231–6239.
[55] L. Bottou, F.E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM Rev. 60 (2) (2018) 223–311.
[56] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.