SIAM J. Sci. Comput.
Vol. 42, No. 2, pp. A639–A665

LEARNING IN MODAL SPACE: SOLVING TIME-DEPENDENT STOCHASTIC PDEs USING PHYSICS-INFORMED NEURAL NETWORKS[*]

DONGKUN ZHANG[†], LING GUO[‡], AND GEORGE EM KARNIADAKIS[†§]

Abstract. One of the open problems in scientific computing is the long-time integration of nonlinear stochastic partial differential equations (SPDEs), especially with arbitrary initial data. We address this problem by taking advantage of recent advances in scientific machine learning and the spectral dynamically orthogonal (DO) and borthogonal (BO) methods for representing stochastic processes. The recently introduced DO/BO methods reduce the SPDE to solving a system of deterministic PDEs and a system of stochastic ordinary differential equations. Specifically, we propose two new physics-informed neural networks (PINNs) for solving time-dependent SPDEs, namely the neural network (NN)-DO/BO methods. The proposed methods incorporate the DO/BO constraints into the loss function (along with the modal decomposition of the SPDE) with an implicit form instead of generating explicit expressions for the temporal derivatives of the DO/BO modes. Hence, the NN-DO/BO methods can overcome some of the drawbacks of the original DO/BO methods. For example, we do not need the assumption that the covariance matrix of the random coefficients is invertible as in the original DO method, and we can remove the assumption of no eigenvalue crossing as in the original BO method. Moreover, the NN-DO/BO methods can be used to solve time-dependent stochastic inverse problems with the same formulation and same computational complexity as for forward problems. We demonstrate the capability of the proposed methods via several numerical examples, namely: (1) A linear stochastic advection equation with deterministic initial condition: we obtain good results with the proposed methods, while the original DO/BO methods cannot be applied directly in this case. (2) Long-time integration of the stochastic Burgers' equation: we show the good performance of NN-DO/BO methods, especially the effectiveness of the NN-BO approach for such problems with many eigenvalue crossings during the whole time evolution, while the original BO method fails. (3) Nonlinear reaction diffusion equation: we consider both the forward problem and the inverse problems, including very noisy initial point values, to investigate the flexibility of the NN-DO/BO methods in handling inverse and mixed type problems. Taken together, these simulation results demonstrate that the NN-DO/BO methods can be employed to effectively quantify uncertainty propagation in a wide range of physical problems, but future work should address the efficiency issue of PINNs for forward problems.

Key words. scientific machine learning, data-driven modeling, uncertainty quantification, dynamical orthogonality, biorthogonality, inverse problems

AMS subject classifications. 60H35, 34F05, 62M45

DOI. 10.1137/19M1260141

1. Introduction.

Physics-informed neural networks (PINNs) [17] are a special class of PDE-induced networks that encode the physics (expressed by the PDE) into a deep neural network (DNN) that shares parameters with a standard DNN that approximates the quantity of interest (QoI), e.g., the solution of the PDE. In practice,

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section May 7, 2019; accepted for publication (in revised form) December 23, 2019; published electronically March 16, 2020.
https://doi.org/10.1137/19M1260141

Funding: This work was supported by the DOE PhILMs project (DE-SC0019453), ARL-Utah 100028801-BROWN-APP (a subaward of Cooperative Agreement W911NF-12-2-0023), and the NSF of China (11671265).

[†]Division of Applied Mathematics, Brown University, Providence, RI 02912 (dongkun_zhang@brown.edu, george_karniadakis@brown.edu).

[‡]Department of Mathematics, Shanghai Normal University, Shanghai, China 200234 (lguo@shnu.edu.cn).

[§]Pacific Northwest National Laboratory, Richland, WA 99354.

this implies that the loss function that expresses a mismatch in the labelled data is augmented by the residual of the PDE, which is represented efficiently by automatic differentiation and is evaluated at random points in the time-space domain. This approximation of the nonlinear operators by the DNN is justified theoretically based on the pioneering work of [3, 4], which goes well beyond the universal function approximation theorem of [10]. This simple and easy-to-program algorithm has been shown to be successful for diverse problems in physics and fluid mechanics [16, 19, 20], especially for inverse problems and even for discovering hidden physics [29]. The advantages of encoding the PDE itself into a DNN are multiple: (1) we require much less data to train the DNN since we are searching for the minima on the manifold-solution of the PDE; (2) we respect the conservation laws of mass, momentum, and energy; and most importantly, (3) we can truly predict the state of the system, unlike the DNNs driven solely by data that can interpolate accurately only within the training domain. While there is still a lot of work to be done to make PINNs efficient simulation machines, one of the main open issues is *uncertainty quantification* in predicting the QoI, which will reflect the various sources of uncertainty, i.e., from the approximation of the DNN to the data and physical model uncertainties.

In [30] we addressed the issue of *total uncertainty* for first time and combined *dropout* and *arbitrary polynomial chaos* to model stochasticity in steady SPDEs. Here, we consider the more difficult case of time-dependent nonlinear SPDEs, and hence we need to introduce a more effective way of dealing with the complexity of long-time integration of stochastic systems. To this end, we employ a generalized form of time-dependent Karhunen–Loève (KL) decomposition, first introduced in [21], appropriate for second-order random fields, which has the form

$$u(x,t;\omega) \approx \sum_{i=1}^{\infty} u_i(x,t)Y_i(t;\omega), \quad \omega \in \Omega. \tag{1.1}$$

This approach can evolve the time-dependent basis of modes $u_i(x,t)$ and stochastic coefficients $Y_i(t;\omega)$ simultaneously, and it is different from the standard polynomial chaos methods [28, 27, 26, 15]. To remove the redundancy in this representation, we need some constraints. For example, this can be achieved by imposing *dynamical constraints* on the spatial basis, the so-called dynamically orthogonal (DO) methodology first proposed in [22, 23]. Alternatively, by imposing static constraints on both the spatial and stochastic basis, the biorthogonal (BO) methodology was developed in [5, 6]. For both DO and BO we need to derive explicitly the evolution equations for all of the components involved, i.e., the mean, spatial basis, and stochastic basis. The DO and BO formulations are mathematically equivalent [9], but they exhibit computationally complementary properties. Specifically, the BO formulation may fail due to crossing of the eigenvalues of the covariance matrix [9], while both BO and DO become unstable when there is a high condition number of the covariance matrix or zero eigenvalues. A rigorous and sharp error bounds of the DO method was first given by Musharbash, Nobile, and Zhou in [14], where it was shown that the DO modes can capture the effective directions. For more applications and improvements of DO/BO methods, we refer to [25, 9, 24, 2] and references therein.

The purpose of this paper is to combine PINNs and the DO/BO methodologies to obtain new effective methods for solving time-dependent SPDEs—we will refer to them as NN-DO/BO methods. Concretely, we first build a surrogate neural net for the solution of the time-dependent SPDEs based on the generalized KL expansion (1.1). Then the DO/BO constraints are included in the loss function, and we train the neural network by minimizing this loss function to obtain the solution. Compared with the original DO/BO method, the merits of the proposed methods are the following:

- We do not need the assumption in our NN-DO approach that the covariance matrix of the random coefficients is invertible, even for SPDEs with deterministic initial conditions.
- We can deal with eigenvalue crossing in the given time domain when applying our NN-BO approach.
- The same NN-DO/BO formulation and computer code can be applied for solving time-dependent stochastic *inverse* problems or problems driven by sparse noisy data, with the same computational complexity.

The organization of this paper is as follows. In section 2, we set up the time-dependent stochastic problems. In section 3, we give a brief review of the DO and BO methodologies. In section 4, we formulate our NN-DO/BO framework after the introduction of the PINNs for solving deterministic differential equations. In section 5, we provide a detailed study of the accuracy and performance of the NN-DO/BO approach with numerical examples. We include two benchmark cases that are specifically designed to have exact solution for the DO and BO representations, followed by a nonlinear stochastic forward problem with high input stochastic dimensionality and noisy data as the initial condition, and a nonlinear inverse problem where we try to identify the model parameters. Finally, we conclude with a brief discussion in section 6.

2. Problem setup.

Let $(\Omega, \mathcal{F}, P)$ be a probability space where Ω is the sample space, $\mathcal{F}$ is the σ-algebra of subsets of Ω, and P is a probability measure. Let D be a bounded domain in $\mathbb{R}^d$ ($d = 1, 2,$ or 3) whose boundary is denoted by ∂D, and let $[0, T]$ be the time domain of interest. We consider the following time-dependent SPDE:

$$\frac{\partial u}{\partial t} = \mathcal{N}_x[u(x,t;\omega)], \quad x \in D,\, t \in [0,T],\, \omega \in \Omega, \tag{2.1}$$

with initial and boundary conditions

$$u\,(x,t;\omega) = u_0(x;\omega), \qquad t = t_0, \tag{2.2}$$

$$\mathcal{B}_x\,[u(x,t;\omega)] = h(x,t;\omega), \qquad x \in \partial D, \tag{2.3}$$

where $\mathcal{N}_x$ is a differential operator, and $\mathcal{B}_x$ is a linear differential operator acting on the domain boundary. Assume that our quantity of interest, $u(x,t;\omega)$, is a second-order random field. The initial and boundary conditions for (2.1) are denoted by $u_0(x;\omega)$ and $h(t,x;\omega)$. Our aim is to solve (2.1), and specifically, to evaluate the mean and standard deviation of the solution $u(x,t;\omega)$.

3. An overview of the DO and BO decomposition methods.

For a random field $u(x,t;\omega)$ that evolves in time, the generalized KL expansion at a given time t is

$$u(x,t;\omega) = \bar{u}(x,t) + \sum_{i=1}^{\infty} \sqrt{\lambda_i}\phi_i(x,t)\xi_i(t;\omega), \quad \omega \in \Omega, \tag{3.1}$$

where $\bar{u}$ is the mean, $\xi_i(t;\omega)$ $(i = 1,2,3,\ldots)$ are zero-mean independent random variables, and λ_i and ϕ_i are the ith largest eigenvalue and the corresponding eigenfunction of the covariance kernel, i.e., they solve the following eigenproblem:

$$\int_D C_{u(x_1,t)u(x_2,t)}\phi_i(x_2,t)\mathrm{d}x_2 = \lambda_i\phi_i(x_1,t). \tag{3.2}$$

Here $C_{u(x_1,t)u(x_2,t)} = \mathbb{E}\left[(u(x_1,t;\omega) - \bar{u}(x_1,t))(u(x_2,t;\omega) - \bar{u}(x_2,t))\right]$ is the covariance kernel of u.

Next, we consider a generalized expansion first proposed in [22]:

$$u(x,t;\omega) = \bar{u}(x,t) + \sum_{i=1}^{\infty} u_i(x,t)Y_i(t;\omega), \quad \omega \in \Omega. \tag{3.3}$$

Similar to the KL expansion, the random field $u(x,t;\omega)$ is decomposed into two parts: (i) the deterministic mean field function $\bar{u}(x,t)$ and (ii) the random fluctuation part, consisting of an infinite summation of deterministic orthogonal fields $u_i(x,t)$ with 0-mean stochastic coefficients $Y_i(t;\omega)$. Formally, we have

$$\bar{u}(x,t) = \mathbb{E}[u(x,t;\omega)] = \int_\Omega u(x,t;\omega)\, \mathrm{d}P(\omega)\,, \tag{3.4}$$

$$\langle u_i, u_j\rangle = 0 \quad \text{for } i \neq j \text{ and } i,j = 1,2,\ldots, \tag{3.5}$$

and

$$\mathbb{E}[Y_i] = 0 \quad \text{for } i = 1,2,\ldots. \tag{3.6}$$

We define the linear subspace $V_S = \operatorname{span}\{u_i(x,t)\}_{i=1}^N$ as the linear space spanned by the first N deterministic basis functions (bases). For now let us assume that $Y_i(t;\omega)$ are linearly independent and $\Omega_S = \operatorname{span}\{Y_i(t;\omega)\}_{i=1}^N$ is the linear subspace in $L^2(\Omega)$ spanned by the first N stochastic coefficients. The truncated expansion $u_N(x,t;\omega)$, defined by

$$u_N(x,t;\omega) = \bar{u}(x,t) + \sum_{i=1}^{N} u_i(x,t)Y_i(t;\omega), \tag{3.7}$$

is the projection of $u(x,t;\omega)$ onto the subspace $V_S \times \Omega_S$. Without making any assumptions on their form, the governing equations (2.1) and (3.4)–(3.6) represent the only information that can be utilized to derive the evolution equations of u_i and Y_i. Note that both the stochastic coefficients $Y_i(t;\omega)$ and the orthogonal bases $u_i(x,t)$ are time-dependent (and they are evolving according to the system dynamics), unlike the standard polynomial chaos where the stochastic coefficients are time-independent. There exists some redundancy in (3.7), and therefore, additional constraints need to be imposed in order to formulate a well-posed problem for the unknown quantities. Here we review the DO and BO approaches, which have different assumptions on the constraints.

3.1. Dynamically orthogonal (DO) representation. As first proposed in [22], a natural constraint to overcome redundancy is that the evolution of the bases $\{u_i(x,t)\}_{i=1}^N$ be orthogonal to the space V_S; this can be expressed through the following DO condition:

$$\frac{dV_S}{dt} \perp V_S \Longleftrightarrow \left\langle \frac{\partial u_i(x,t)}{\partial t}, u_j(x,t)\right\rangle = 0, \quad i,j = 1,\ldots,N. \tag{3.8}$$

Here $\langle u(x,t), v(x,t)\rangle$ is defined as the spatial inner product $\int_D u(x,t)v(x,t)\,\mathrm{d}x$. Comparing (3.7) with the standard KL expansion (3.1), in the DO representation we set

u_i to have unit length, and Y_i carries the scaling coefficient as the result of the eigenvalues. Note that the DO condition preserves the orthonormality and the length of the bases $\{u_i(x,t)\}_{i=1}^N$ since

(3.9)
$$\frac{\partial}{\partial t}\langle u_i(\cdot,t),u_j(\cdot,t)\rangle=\left\langle\frac{\partial u_i(\cdot,t)}{\partial t},u_j(\cdot,t)\right\rangle+\left\langle u_i(\cdot,t),\frac{\partial u_j(\cdot,t)}{\partial t}\right\rangle=0,\quad i,j=1,\dots,N.$$

It is proved in [22] that the DO condition leads to a set of independent and explicit evolution equations for all of the unknown quantities. Here we state the DO evolution equations without proof.

THEOREM 1 (see [22]). *Under the assumptions of the DO representation, the original SPDE* (2.1) *is reduced to the following system of equations:*

$$\begin{aligned}\frac{\partial\bar{u}(t,x)}{\partial t}&=\mathbb{E}[\mathcal{N}_x[u(\cdot,t;\omega)]],\\ \frac{\mathrm{d}Y_i(t;\omega)}{\mathrm{d}t}&=\langle\mathcal{N}_x[u(\cdot,t;\omega)]-\mathbb{E}[\mathcal{N}_x[u(\cdot,t;\omega)]],u_i(\cdot,t)\rangle,\quad i=1,\dots,N,\\ \sum_{i=1}^N C_{Y_i(t)Y_j(t)}\frac{\partial u_i(t,x)}{\partial t}&=\prod_{V_s^\perp}\mathbb{E}\left[\mathcal{N}_x\left[u(\cdot,t;\omega)\right]Y_j\right],\qquad j=1,\dots,N.\end{aligned}\tag{3.10}$$

The projection in the orthogonal complement of the linear subspace V_S is defined as $\prod_{V_S^\perp}F(x)=F(x)-\prod_{V_s}F(x)=F(x)-\sum_{k=1}^N\langle F(\cdot),u_k(\cdot,t)\rangle u_k(\cdot,t)$, and the covariance of the stochastic coefficients is $C_{Y_i(t)Y_j(t)}=E\left[Y_i(t;\omega)Y_j(t;\omega)\right]$. The associated boundary conditions are determined by

$$\begin{aligned}\mathcal{B}_x\left[\bar{u}(x,t;\omega)\right]|_{x\in\partial D}&=\mathbb{E}[h(x,t;\omega)],\\ \mathcal{B}_x\left[u_i(x,t)\right]|_{x\in\partial D}&=\mathbb{E}\left[Y_j(t;\omega)h(x,t;\omega)\right]C^{-1}_{Y_i(t)Y_j(t)}\quad\text{for } i=1,2,\dots,N,\end{aligned}\tag{3.11}$$

and the initial conditions at $t=t_0$ for the DO components are given by

$$\begin{aligned}&\bar{u}\left(x,t_0\right)=\mathbb{E}\left[u_0(x;\omega)\right],\\ &Y_i\left(t_0;\omega\right)=\langle u_0(\cdot,\omega)-\bar{u}\left(x,t_0\right),v_i(\cdot)\rangle,\\ &u_i\left(x,t_0\right)=v_i(x)\end{aligned}\tag{3.12}$$

for all $i=1,\dots,N$, where $v_i(x)$ is the eigenfield of the standard KL expansion of $u_0(x;\omega)$.

It is shown in [22] that by imposing suitable restrictions on the DO representation, the equations for methods such as polynomial chaos (PC) or proper orthogonal decomposition (POD) can be recovered from the DO evolution equations. For example, PC can be recovered by setting $Y_i(t;\omega)=\Psi_i(\xi(\omega))$, where $\Psi_i(\xi)$ is an orthogonal polynomial in terms of ξ. Moreover, it is shown in [7] that there exists a one-to-one correspondence between the eigenvalues of the KL expansion for $u(x,t;\omega)$ and the eigenvalues of the covariance matrix $C_{Y_i(t)Y_j(t)}$ in the DO representation given any fixed time t. Thus, the stochastic coefficients Y_i together with the modes u_i provide the necessary information to describe both the shape and the magnitude of the uncertainty that characterizes a stochastic field, but also the principle directions in which the stochasticity is distributed.

The moments of $u(x,t;\omega)$ can be readily computed from the DO representation. For example, the first moment, i.e., the mean, can be trivially obtained from the first term $\bar{u}(x,t)$, and the variance can be calculated as follows:

$$\mathrm{Var}[u] = \mathbb{E}\left[(u-\bar{u})^2\right] = \mathbb{E}\left[\left(\sum_{i=1}^{N} u_i Y_i\right)^2\right] = \sum_{i,j=1}^{N} u_i \mathbb{E}[Y_i Y_j] u_j. \tag{3.13}$$

3.2. BO representation. An alternative way to overcome the aforementioned redundancy in (3.7) is the BO condition, which imposes the static constraint that both the spatial basis functions and stochastic coefficients are orthogonal in time [6]. In other words, we have the following conditions:

$$\langle u_i(\cdot,t), u_j(\cdot,t)\rangle = \lambda_i(t)\delta_{ij}, \quad \mathbb{E}[Y_i Y_j] = \delta_{ij}, \quad i,j = 1,\ldots,N, \tag{3.14}$$

where the λ_is are the eigenvalues of the covariance kernel and δ_{ij} is the Dirac's delta function. There is a slight difference between the DO and BO representations: the stochastic coefficients carry the eigenvalues of the covariance operator in the DO representation, while the spatial bases carry the eigenvalues of the covariance operator in the BO representation.

Next, we define the matrices S and M whose entries are

$$S_{ij} = \left\langle u_i, \frac{\partial u_j}{\partial t}\right\rangle, \quad M_{ij} = \mathbb{E}\left[Y_i \frac{\mathrm{d}Y_j}{\mathrm{d}t}\right]. \tag{3.15}$$

Then by taking the time derivative of (3.14), we have

$$\begin{aligned} S_{ij} + S_{ji} &= \left\langle u_i, \frac{\partial u_j}{\partial t}\right\rangle + \left\langle \frac{\partial u_i}{\partial t}, u_j\right\rangle = 0 && \text{for } i \neq j,\\ S_{ij} &= \frac{1}{2}\frac{\mathrm{d}\lambda_i(t)}{\mathrm{d}t} && \text{for } i = j,\\ M_{ij} + M_{ji} &= \mathbb{E}\left[Y_i \frac{\mathrm{d}Y_j}{\mathrm{d}t}\right] + \mathbb{E}\left[\frac{\mathrm{d}Y_i}{\mathrm{d}t} Y_j\right] = 0. \end{aligned} \tag{3.16}$$

Here we state the BO evolution equations without proof.

THEOREM 2 (see [5]). *We assume that the bases and stochastic coefficients satisfy the BO condition. Then, the original SPDE* (2.1) *is reduced to the following system of equations:*

$$\begin{aligned} \frac{\partial \bar{u}(t,x)}{\partial t} &= \mathbb{E}[\mathcal{N}_x[u(\cdot,t;\omega)]],\\ \lambda_i \frac{\mathrm{d}Y_i(t;\omega)}{\mathrm{d}t} &= -\sum_{j=1}^{N} S_{ij} Y_j + \langle \mathcal{N}_x[u] - \mathbb{E}[\mathcal{N}_x[u]], u_i(\cdot,t)\rangle, \quad i = 1,\ldots,N,\\ \frac{\partial u_i(t,x)}{\partial t} &= -\sum_{j=1}^{N} M_{ij} u_j + \mathbb{E}[\mathcal{N}_x[u] Y_i], \quad i = 1,\ldots,N. \end{aligned} \tag{3.17}$$

Moreover, if $\lambda_i \neq \lambda_j$ for $i \neq j$, $i,j,= 1,2,\ldots,N$, the N-by-N matrices S and M have closed form expression:

$$\begin{aligned} M_{ij} &= \begin{cases} \frac{G_{ij}+G_{ji}}{-\lambda_i+\lambda_j} & \text{if } i \neq j,\\ 0 & \text{if } i = j, \end{cases}\\ S_{ij} &= \begin{cases} G_{ij} + \lambda_i M_{ij} & \text{if } i \neq j,\\ G_{ii} & \text{if } i = j, \end{cases} \end{aligned} \tag{3.18}$$

where the matrix G_{ij} is defined as $\langle \mathbb{E}\left[\mathcal{N}_x[u]Y_j\right], u_i\rangle$.

Similar to the DO method, the boundary condition is given by

$$\begin{aligned}\mathcal{B}_x\left[\bar{u}(x,t;\omega)\right]|_{x\in\partial D} &= \mathbb{E}[h(x,t;\omega)],\\ \mathcal{B}_x\left[u_i(x,t)\right]|_{x\in\partial D} &= \mathbb{E}\left[Y_i(t;\omega)h(x,t;\omega)\right] \quad \text{for } i=1,2,\dots,N,\end{aligned} \tag{3.19}$$

and the initial condition is generated from the KL expansion of $u_0(x;\omega)$.

3.3. A brief summary of DO/BO methods. Let us note the following difference between the DO and BO conditions: the spatial bases, u_i, under the DO condition evolve toward the direction which is normal to the space V_S into which they expand (orthogonality is automatically maintained), while under the BO condition, only the mutual orthogonality within u_i and within Y_i are required and there is no restriction upon the direction of their evolution. As a compensation for the lack of constraints in spatial bases, the BO condition puts an additional orthogonality restriction on the random coefficients Y_i. However, Choi, Sapsis, and Kerniadakis [9] have proved theoretically the equivalence between the DO and BO methods, in a sense that one method is an exact reformulation of the other via a differential transformation.

Each method can be applied to a limited range of problems since the evolution equations are valid only if certain assumptions are satisfied. For the DO method, it is assumed that the covariance matrix of random coefficients, $C_{Y_iY_j}$, is invertible. Therefore, it fails when applied to some benchmark problems such as a stochastic PDE with deterministic initial condition. This is because all coefficients Y_i are equal to 0 at the initial state and their covariance matrix is singular. For the BO method, it is assumed that there is no eigenvalue crossing in the given time domain in order to calculate the explicit expression of M_{ij} and S_{ij}. However, some strategies have been proposed to get around those issues, e.g., the hybrid gPC-DO method [8] and the pseudo-inverse hybrid BO-DO method [2] were developed to address the above limitations.

Inspired by the DO and BO methods, we introduce a new procedure for solving time-dependent stochastic PDEs within the framework of physics-informed neural networks (PINNs). The proposed methods inherit the similarities between the DO and BO methods and can be implemented with either the DO or the BO version, which are free from the aforementioned restrictions.

4. Methodology.

4.1. PINN. In this part, we briefly review using DNNs to solve the deterministic differential equations in [11, 12, 18] and their generalization for solving deterministic inverse problems in [19]. Suppose that we have a parameterized deterministic differential equation:

$$\begin{aligned}\mathcal{N}_x[u;\eta] = 0, &\quad x\in D,\\ \text{B.C.:}\qquad \mathcal{B}_x[u]=0, &\quad x\in\Gamma,\end{aligned} \tag{4.1}$$

where $u(x)$ is the solution, and η denotes the parameters.

A DNN, denoted by $\hat{u}(x;\theta)$, is constructed as a surrogate of the solution $u(x)$, and it takes the coordinate x as the input and outputs a vector that has the same dimension as u. Here we use θ to denote the DNN parameters that will be tuned at the training stage, namely, θ contains all of the weights $\boldsymbol{w}$ and biases $\boldsymbol{b}$ in $\hat{u}(x;\theta)$. For this surrogate network $\hat{u}$, we can take its derivatives with respect to its input by applying the chain rule for differentiating compositions of functions using the automatic differentiation, which is conveniently integrated in many machine learning packages such as Tensorflow [1]. The restrictions on $\hat{u}$ is two-fold: first, given the set of scattered data of the $u(x)$ observations, the network should be able to reproduce the observed value when taking the associated x as input; second, $\hat{u}$ should comply with the physics imposed by (4.1). The second part is achieved by defining a residual network:

$$\hat{f}(x;\theta,\eta) := \mathcal{N}_x[\hat{u}(x;\theta);\eta], \tag{4.2}$$

which is computed from $\hat{u}$ straightforwardly with automatic differentiation. This residual network $\hat{f}$, also named the physics-informed neural network (PINN), shares the same parameters θ with network $\hat{u}$ and should output the constant 0 for any input $x \in D$. Figure 1 shows a sketch of the PINN. At the training stage, the shared parameters θ (and also η, if it is also to be inferred) are fine-tuned to minimize a loss function that reflects the above two constraints.

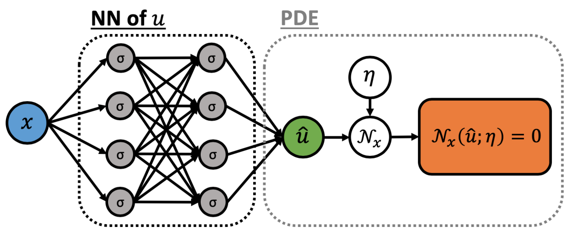


FIG. 1. *Schematic of the PINN for solving steady state differential equations.*

Suppose we have a total number of N_u observations on u, collected at location $\{x_u^{(i)}\}_{i=1}^{N_u}$, and N_c is the number of training points $\{x_f^{(i)}\}_{i=1}^{N_c}$, where we evaluate the residual $\hat{f}(x_f^{(i)};\theta,\eta)$. We shall use (x^*, y^*) to represent a single instance of training data, where the first entry x^* denotes the input and the second entry y^* denotes the anticipated output (also called label). The workflow of solving a differential equation with PINN can be summarized in Algorithm 1.

4.2. A weak formulation interpretation of the DO and BO methods. The derivation of equations for both the DO and BO methods can be summarized in four steps as follows:

1. Apply operator $\mathbb{E}[\cdot]$ on both sides of the SPDE and replace u by the finite expansion in (3.7). Notice that $\mathbb{E}[Y_i] = 0$. This leads to the first equation in (3.10) and (3.16):

$$\mathbb{E}\left[\frac{\partial u}{\partial t}\right] = \frac{\partial \bar{u}}{\partial t} = \mathbb{E}\left[\mathcal{N}_x[u(x,t;\omega)]\right]. \tag{4.3}$$

2. Apply operator $\langle \cdot, u_i \rangle$ on both sides of the SPDE:

$$\left\langle \frac{\partial u}{\partial t}, u_i \right\rangle = \langle \mathcal{N}_x[u(x,t;\omega)], u_i \rangle. \tag{4.4}$$

3. Apply operator $\mathbb{E}[\cdot Y_i]$ on both sides of the SPDE:

$$\mathbb{E}\left[\frac{\partial u}{\partial t} Y_i\right] = \mathbb{E}\left[\mathcal{N}_x[u(x,t;\omega)] Y_i\right]. \tag{4.5}$$

4. Substitute u by the truncated expansion in (3.7), and use the DO and BO constraints, (3.8) and (3.14), to simplify (4.4) and (4.5).

Due to the orthogonality of $u_i(x,t)$, they form a valid set of bases in the physical space D. The random coefficients $Y_i(t;\omega)$ are also linearly independent as they are orthogonal under the BO representation, and the DO representation is equivalent to the BO representation so Y_i will not degenerate in the DO expansion either. Therefore, the random coefficients $Y_i(t;\omega)$ form a valid set of basis in the probability space $L^2(\Omega)$. Consequently, (4.3)–(4.5) are the weak formulation of the original SPDE in the physical space and the probability space (note that (4.3) is the inner product of both sides of the original SPDE on constant 1, which can be regarded as the 0th basis in the probability space), and they provide all of the necessary information to find the solution u_N in $V_S \times \Omega_S$.

Algorithm 1. PINN for solving deterministic PDEs.

Step 1: Specify the training set:

$$\hat{u} \text{ network: } \{(x_u^{(i)}, u(x_u^{(i)}))\}_{i=1}^{N_u}, \quad \hat{f} \text{ network: } \{(x_f^{(i)}, 0)\}_{i=1}^{N_f}.$$

Step 2: Construct a DNN $\hat{u}(x;\theta)$ with random initialized parameters θ.

Step 3: Construct the residual network $\hat{f}(x;\theta,\eta)$ by substituting the surrogate $\hat{u}$ into the governing equation (4.2) via automatic differentiation and arithmetic operations.

Step 4: Specify a loss function by summing the mean squared error of both the u observations and the residual:

$$\mathcal{LOSS}(\theta,\eta) = \frac{1}{N_u}\sum_{i=1}^{N_u}[\hat{u}(x_u^{(i)};\theta) - u(x_u^{(i)})]^2 + \frac{1}{N_c}\sum_{i=1}^{N_c}\hat{f}(x_f^{(i)};\theta,\eta)^2. \tag{4.6}$$

Step 5: Train the DNN to find the best parameters θ and η by minimizing the loss function:

$$\theta = \arg\min \mathcal{LOSS}(\theta,\eta). \tag{4.7}$$

4.3. NN-DO/BO methods.

In this section we formalize the algorithm of solving time-dependent stochastic PDEs using PINNs. First, we rewrite (3.7) as

$$u_N(x,t;\omega) = \bar{u}(x,t) + \sum_{i=1}^{N} a_i(t)u_i(x,t)Y_i(t;\omega), \tag{4.8}$$

while enforcing $\langle u_i, u_i\rangle = 1$ and $\mathbb{E}[Y_i^2] = 1$. The time-dependent coefficients $a_i(t)$ are scaling factors and play the role of $\sqrt{\lambda_i}$ when we compare (4.8) with the standard KL expansion (3.1). Suppose that the original SPDE is parameterized into a PDE that involves a finite set of random variables $\xi(\omega)$; then $u(x,t;\omega)$ can be written as $u(x,t;\xi)$, and $Y_i(t;\omega)$ can be written as $Y_i(t;\xi)$. Four separate neural networks are constructed:

1. The neural net $\bar{u}_{nn}(x,t)$ that takes x and t as the input and outputs $\mathbb{E}[u(x,t;\omega)]$.
2. The neural net $A_{nn}(t)$ that takes t as the input and outputs an N-dimensional vector representing $a_i(t)$ for $i = 1, 2, \ldots, N$.

3. The neural net $U_{nn}(x, t)$ that takes x and t as the input and outputs an N-dimensional vector representing $u_i(x, t)$ for $i = 1, 2, \dots, N$.
4. The neural net $Y_{nn}(\xi, t)$ that takes ξ and t as the input and outputs an N-dimensional vector representing $Y_i(t; \xi)$ for $i = 1, 2, \dots, N$.

A surrogate neural net for the solution $u_N(x, t; \omega)$ can be constructed from those four neural nets by substituting them into (4.8), yielding

$$u_{nn}(x, t; \xi) = \bar{u}_{nn} + \sum_{i=1}^{N} A_{nn,i} U_{nn,i} Y_{nn,i}. \tag{4.9}$$

Since the weak formulation of SPDE involves integration in both the physical and probability spaces, the neural nets are evaluated at the physical training points $\{x_c^k\}_{k=1}^{n_x}$ and the probabilistic training points $\{\xi_c^l\}_{l=1}^{n_\xi}$, where n_x and n_ξ are the numbers of training points. In the time domain $[0, T]$ we uniformly sample n_t random points $\{t_c^s\}_{s=1}^{n_t}$. Once we have constructed the computation graph, the derivatives of the quantity of interest with respect to time t and space coordinate x can be easily obtained via the auto-differentiation algorithm, and the integration terms can be evaluated by using a numerical quadrature rule.

The loss function is a weighted summation of four components: the weak formulation of SPDE, initial/boundary conditions, constraints on U_{nn} and Y_{nn}, and the additional regularization terms. The loss function in each part consists of mean squared errors (MSEs) associated with the prescribed constraints calculated from the sampled training points. Next, we will illustrate each of these four components of loss function and write out their explicit expressions.

4.3.1. Loss function for the weak formulation of SPDE. The weak form of the SPDE, i.e., (4.3)–(4.5), can be rewritten as

$$\epsilon_1^{ks} := \mathbb{E}\left[\frac{\partial u_{nn}}{\partial t} - \mathcal{N}_x[u_{nn}(x_c^k, t_c^s; \xi)]\right] = 0, \tag{4.10}$$

$$\epsilon_2^{sl} := \left\langle \frac{\partial u_{nn}}{\partial t} - \mathcal{N}_x[u_{nn}(x, t_c^s; \xi_c^l)], U_{nn,i}(x, t_c^s) \right\rangle = 0, \tag{4.11}$$

$$\epsilon_3^{ks} := \mathbb{E}\left[\left(\frac{\partial u_{nn}}{\partial t} - \mathcal{N}_x[u_{nn}(x_c^k, t_c^s; \xi)]\right) Y_{nn,i}(t_c^s; \xi)\right] = 0, \tag{4.12}$$

where the integration in the physical space and the probability space shall be evaluated by using a numerical quadrature rule. The first part of the loss function is calculated by

$$\mathrm{MSE_w} = \frac{1}{n_x n_t} \sum_{k,s} \left(\epsilon_1^{ks}\right)^2 + \frac{1}{n_t n_\xi} \sum_{s,l} \left(\epsilon_2^{sl}\right)^2 + \frac{1}{n_x n_t} \sum_{k,s} \left(\epsilon_3^{ks}\right)^2. \tag{4.13}$$

4.3.2. Loss function for initial and boundary conditions. Let t_0 be the initial time of computation. The initial condition for the representation in (4.8) is similar to (3.12). The only difference is that Y_i are normalized to have unit variance, and the standard deviation of $\langle u_0(x; \xi) - \bar{u}(x, t_0), v_i(x) \rangle$ is assigned to be the initial value for a_i. Here $v_i(x)$ are the normalized KL modes for $u(x, t_0; \omega)$, and they are the initial value for u_i. That is,

$$\begin{aligned} &\bar{u}(x, t_0) = \mathbb{E}\left[u(x, t_0; \xi)\right], \\ &u_i(x, t_0) = v_i(x), \\ &a_i(t_0) = \sqrt{\mathbb{E}\left[\langle u(x, t_0; \xi) - \mathbb{E}[u(x, t_0; \xi)], v_i\rangle^2\right]}, \\ &Y_i(t_0; \xi) = \frac{1}{a_i(t_0)} \left\langle u(x, t_0; \xi) - \mathbb{E}[u(x, t_0; \xi)], v_i\right\rangle. \end{aligned} \tag{4.14}$$

For the deterministic initial condition, $u_i(x, t_0)$ are set to be orthonormal bases satisfying the boundary condition, $Y_i(t_0; \xi)$ are set to be the generalized polynomial chaos (gPC) bases of ξ with unit variance, and $a_i(t_0)$ is set to be 0. The initial condition shall be imposed upon the neural network by adding an extra penalty term $\mathrm{MSE}_{\mathrm{IC}}$, and it is calculated as follows:

$$\begin{aligned} \mathrm{MSE}_{\mathrm{IC}} =& \frac{1}{n_x} \sum_{k=1}^{n_x} \left(\bar{u}_{nn}(x_c^k, t_0) - \bar{u}(x_c^k, t_0)\right)^2 + \frac{1}{N n_x} \sum_{i=1}^{N} \sum_{k=1}^{n_x} \left(U_{nn,i}(x_c^k, t_0) - u_i(x_c^k, t_0)\right)^2 \\ &+ \frac{1}{N} \sum_{i=1}^{N} \left(A_{nn,i}(t_0) - a_i(t_0)\right)^2 + \frac{1}{N n_\xi} \sum_{i=1}^{N} \sum_{l=1}^{n_\xi} \left(Y_{nn,i}(t_0; \xi_c^l) - Y_i(t_0; \xi_c^l)\right)^2. \end{aligned} \tag{4.15}$$

The boundary condition is imposed by taking the weak formulation of (2.3) in the random space, i.e.,

$$\begin{aligned} \mathbb{E}\left[\mathcal{B}_x[u(x_b, t; \xi)]\right] = \mathbb{E}[h(x_b, t; \xi)] &\Rightarrow \mathcal{B}_x\left[\bar{u}(x_b, t)\right] = \mathbb{E}[h(x_b, t; \xi)], \\ \mathbb{E}\left[\mathcal{B}_x[u(x_b, t; \xi)] Y_i(t; \xi)\right] &= \mathbb{E}[h(x_b, t; \xi) Y_i(t; \xi)] \\ &\Rightarrow \sum_{j=1}^{N} C_{Y_i Y_j} a_j(t) \mathcal{B}_x\left[u_j(x_b, t)\right] = \mathbb{E}[h(x_b, t; \xi) Y_i(t; \xi)]. \end{aligned} \tag{4.16}$$

Thus, the loss associated with the boundary condition is

$$\begin{aligned} \mathrm{MSE}_{\mathrm{BC}} = &\frac{1}{n_t} \sum_{s=1}^{n_t} \left(\mathcal{B}_x[\bar{u}_{nn}(x_b, t_c^s)] - \mathbb{E}\left[h(x_b, t; \xi)\right]\right)^2 \\ &+ \frac{1}{N n_t} \sum_{i=1}^{N} \sum_{s=1}^{n_t} \left(\sum_{j=1}^{N} C_{Y_i Y_j}(t_c^s) A_{nn,j}(t_c^s) \mathcal{B}_x\left[U_{nn,j}(x_b, t_c^s)\right] \right. \\ &\left. - \mathbb{E}\left[h(x_b, t_c^s; \xi) Y_{nn,i}(t_c^s; \xi)\right] \right)^2, \end{aligned} \tag{4.17}$$

where the expectations and covariance matrix shall be evaluated by using a numerical quadrature rule.

Note that the periodic boundary condition can be strictly imposed by modifying the neural nets $\bar{u}_{nn}$ and U_{nn} by replacing the input x with the combination of $\sin(2\pi x/L)$ and $\cos(2\pi x/L)$, where L is the length of domain D. This is because any continuous 2π-periodic function can be written as a nonlinear function of $\sin(x)$ and $\cos(x)$. This modification simplifies the loss function by removing the loss due to the periodic boundary condition.

4.3.3. Loss function for the constraints on $\boldsymbol{U_{nn}}$ and $\boldsymbol{Y_{nn}}$. Here we can have different implementations that favor either the DO or the BO method. Both DO and BO representations require that $\mathbb{E}[Y_i] = 0$, and thus the loss functions in both implementations should involve the term $\frac{1}{n_t}\sum_{s=1}^{n_t}\left(\mathbb{E}[Y_i(t_c^s;\xi)]\right)^2$. For the DO constraint, (3.8) should be satisfied. In addition, we require that

$$\mathbb{E}\left[Y_i(t;\xi)\frac{\mathrm{d}Y_i(t;\xi)}{\mathrm{d}t}\right] = 0 \quad \forall t \text{ and } i = 1, 2, \dots, N,$$

so that Y_i stay normalized with unit variance. The loss function for DO is

$$\begin{aligned}\mathrm{MSE}_{\mathrm{DO}} =& \frac{1}{Nn_t}\sum_{i=1}^{N}\sum_{s=1}^{n_t}\left(\mathbb{E}[Y_i(t_c^s;\xi)]\right)^2\\ &+ \frac{1}{N^2 n_t}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{s=1}^{n_t}\left\langle\frac{\mathrm{d}U_{nn,i}(x,t_c^s)}{\mathrm{d}t}, U_{nn,j}(x,t_c^s)\right\rangle^2\\ &+ \frac{1}{Nn_t}\sum_{i=1}^{N}\sum_{s=1}^{n_t}\mathbb{E}\left[Y_{nn,i}(t_c^s;\xi)\frac{\mathrm{d}Y_{nn,i}(t_c^s;\xi)}{\mathrm{d}t}\right]^2.\end{aligned} \tag{4.18}$$

For the BO constraints, (3.16) generates the following loss function:

$$\begin{aligned}\mathrm{MSE}_{\mathrm{BO}} =& \frac{1}{Nn_t}\sum_{i=1}^{N}\sum_{s=1}^{n_t}\left(\mathbb{E}[Y_i(t_c^s;\xi)]\right)^2\\ &+ \frac{1}{N^2 n_t}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{s=1}^{n_t}\left(\left\langle\frac{\mathrm{d}U_{nn,i}(x,t_c^s)}{\mathrm{d}t}, U_{nn,j}(x,t_c^s)\right\rangle\right.\\ &\left.+ \left\langle\frac{\mathrm{d}U_{nn,j}(x,t_c^s)}{\mathrm{d}t}, U_{nn,i}(x,t_c^s)\right\rangle\right)^2\\ &+ \frac{1}{Nn_t}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{s=1}^{n_t}\left(\mathbb{E}\left[Y_{nn,i}(t_c^s;\xi)\frac{\mathrm{d}Y_{nn,j}(t_c^s;\xi)}{\mathrm{d}t}\right]\right.\\ &\left.+ \mathbb{E}\left[Y_{nn,j}(t_c^s;\xi)\frac{\mathrm{d}Y_{nn,i}(t_c^s;\xi)}{\mathrm{d}t}\right]\right)^2.\end{aligned} \tag{4.19}$$

Since we put all of the scaling factors to $a(t)$ and keep $u_i(x,t)$ normalized, $S_{ij}+S_{ji} = 0$ in (3.16) still holds true when i is equal to j.

4.3.4. Loss function for additional regularization. Additional regularization terms shall be added to the loss function to reduce the risk of overfitting. Here we remark that it is helpful to add a penalty term from the original equation (2.1) to speed up the training. The loss from the original equation is

$$\mathrm{MSE}_0 = \frac{1}{n_x n_t n_\xi}\sum_{k=1}^{n_x}\sum_{s=1}^{n_t}\sum_{l=1}^{n_\xi}\left(\frac{\partial u_{nn}}{\partial t} - \mathcal{N}_x\left[u_{nn}(x_c^k, t_c^s; \xi_c^l)\right]\right)^2. \tag{4.20}$$

4.3.5. Putting the loss functions together. A sketch of the computation graph for the loss functions $\mathrm{MSE}_{\mathrm{w}}$ and $\mathrm{MSE}_{\mathrm{DO/BO}}$ is shown in Figure 2. The loss function used for training the PINNs is the weighted summation of the aforementioned

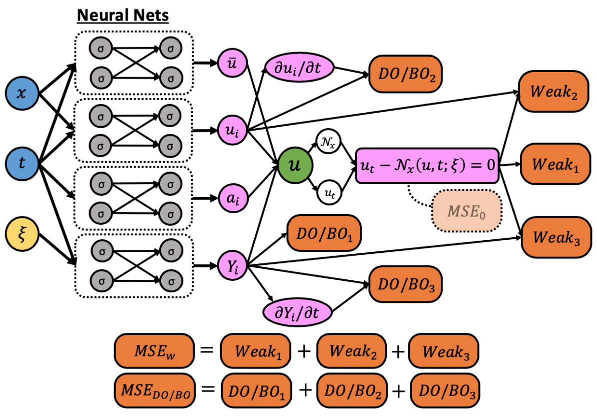

Neural Nets
x
t
ξ
$\bar{u}$
u_i
a_i
Y_i
$\partial u_i/\partial t$
$\partial Y_i/\partial t$
u
$\mathcal{N}_x$
u_t
$u_t - \mathcal{N}_x(u,t;\xi) = 0$
MSE_0
DO/BO_1
DO/BO_2
DO/BO_3
$Weak_1$
$Weak_2$
$Weak_3$
$MSE_w = Weak_1 + Weak_2 + Weak_3$
$MSE_{DO/BO} = DO/BO_1 + DO/BO_2 + DO/BO_3$

FIG. 2. *Schematic of the NN-DO/BO for solving time-dependent stochastic differential equations, where the blocks $Weak_1$, $Weak_2$, and $Weak_3$ correspond to the three right-hand side terms in* (4.13)*, and blocks DO/BO_1, DO/BO_2, and DO/BO_3 correspond to the three right-hand side terms in* (4.18) *and* (4.19)*, respectively.*

MSEs. Intuitively, we want to train the networks to gradually satisfy the weak formulation of the SPDE, while not violating the physical restrictions and the DO/BO constraints, as these are the cornerstones of the weak formulation. Therefore, we put a relatively large weight in front of $\mathrm{MSE}_{\mathrm{IC}}$, $\mathrm{MSE}_{\mathrm{BC}}$, and $\mathrm{MSE}_{\mathrm{DO/BO}}$, making their scale the same as that of $\mathrm{MSE}_{\mathrm{w}}$, if not slightly larger. The idea behind this is to remove the redundancy of (4.8) in the first place. We put a small weight for the regularization term since it is only used to help speedup the training process and is not essential. Nevertheless, the distribution of weights is still an open question for future research. In the numerical tests we train our neural nets by minimizing the following loss function:

$$\mathcal{LOSS} = \mathrm{MSE}_{\mathrm{w}} + 100 \times (\mathrm{MSE}_{\mathrm{IC}} + \mathrm{MSE}_{\mathrm{BC}} + \mathrm{MSE}_{\mathrm{DO/BO}}) + 0.1 \times \mathrm{MSE}_0. \tag{4.21}$$

This proposed algorithm can be implemented with the DO or BO constraints, which we name the NN-DO or NN-BO method, respectively. The proposed algorithm is summarized as follows.

Algorithm 2. NN-DO/BO for solving time-dependent stochastic PDEs.

Step 1: Build the neural networks for $\bar{u}_{nn}(x,t)$, $A_{nn}(t)$, $U_{nn}(x,t)$ and $Y_{nn}(t;\xi)$.
Step 2: Select n_x training points in the physical domain D, and n_ξ training points in the stochastic space Ω. Randomly pick n_t points in the time domain $[0,T]$ from a uniform distribution.
Step 3: Specify the method to use (DO or BO) and calculate the loss function in (4.21).
Step 4: Train the neural networks by minimizing the loss function.
Step 5: Reconstruct the SPDE solution using (4.9).

We remark that the bottleneck of the original DO/BO method occurs when generating an explicit expression for the temporal derivatives of the bases (step 4 in subsection 4.2). For the standard DO method, this involves calculating the inverse of a covariance matrix which could be singular, and for the standard BO method, to obtain the explicit expressions for matrices S and M in (3.15), one has to assume no eigenvalue crossing. In the proposed NN-DO/BO algorithm, there is no need to derive explicit expressions from constraints; instead we need only write the constraints into the loss functions as they are.

5. Simulation results.

We first test our NN-DO/BO methods with two benchmark cases that are especially designed to have exact solutions for the DO and BO representations. To demonstrate the advantage of the NN-DO/BO methods over the standard methods, we then solve a nonlinear diffusion-reaction equation with a 19-dimensional random input, where the problem is solved with very rough initial conditions given as discrete point values. Finally, an inverse problem is also considered to demonstrate the new capability of the proposed NN-DO/BO methods. For all test cases we use deep feed-forward neural networks for $\bar{u}_{nn}(x,t)$, $A_{nn}(t)$, $U_{nn}(x,t)$, and $Y_{nn}(t;\xi)$. The loss functions are defined in (4.21), and the Adam optimizer with learning rate 0.001 is used to train the networks.

5.1. Application to a linear stochastic problem.

In this section we present a pedagogical example by solving the linear stochastic advection equation using the NN-DO/BO methods. The stochastic advection equation with a random advection coefficient has the form

$$\begin{aligned} &\frac{\partial u(x,t;\xi)}{\partial t} + \xi\frac{\partial u(x,t;\xi)}{\partial x} = 0 \quad \forall (x,t) \in D\times[0,T], \\ &u(x,0;\xi) = -\sin(x) \quad \forall x\in D, \end{aligned} \tag{5.1}$$

where the physical domain D is $[-\pi,\pi]$, and we obtain the solution until final time $T=\pi$. Periodic boundary conditions are considered such that $u(-\pi,t)=u(\pi,t)$ $\forall t\in[0,T]$. The randomness comes from the advection velocity, which is modeled as a Gaussian random variable $\xi\sim N(0,\sigma^2)$, where we set σ to be 0.8. The exact DO/BO solutions are included in section SM1 and were derived from [7].

We set n_t, n_x, and n_ξ all to be 50. The data points in the time domain $\{t_c^s\}_{s=1}^{n_t}$ are sampled from a uniform distribution. The training points $\{x_c^k\}_{k=1}^{n_x}$ are equidistantly distributed in $[-\pi,\pi]$. For the training points in the stochastic space, instead of using the Gauss–Hermite quadrature rule, we generate $\{\xi_c^l\}_{l=1}^{n_\xi}$ by applying the inverse cumulative distribution function of the standard normal distribution to the Gauss–Legendre quadrature points in $[0,1]$, because the generated ξ will be more concentrated near the origin, making it easier to train the neural networks. The neural networks are trained with an Adam optimizer (learning rate 0.001) for 300000 epochs.

5.1.1. Case 1: NN-DO method.

The standard DO method cannot be directly applied to this SPDE with deterministic initial condition. However, by applying the NN-DO method we obtain good results. Considering (4.8), the initial conditions are

$$\begin{aligned} &\bar{u}(x,0)=u(x,0), && a_1(0)=a_2(0)=0, \\ &u_1(x,0)=-\frac{1}{\pi}\cos(x), && u_2(0)=-\frac{1}{\pi}\sin(x), \\ &Y_1(0;\xi)=-\xi, && Y_2(0;\xi)=-\frac{\sqrt{2}}{2}(\xi^2-1), \end{aligned} \tag{5.2}$$

where we use the periodic orthonormal bases in the $[-\pi,\pi]$ interval as the initial conditions for u_1 and u_2, and we use the normalized Hermite polynomials for the initial conditions of Y_1 and Y_2. The neural networks $\bar{u}_{nn}(x,t)$ and $U_{nn}(x,t)$ have three hidden layers with 32 neurons per hidden layer, the network $A_{nn}(t)$ has three hidden layers with 16 neurons per hidden layer, and the network $Y_{nn}(t;\xi)$ has four hidden layers with 64 neurons in each hidden layer. The reference solutions for the mean, variance, and the modes u_i are taken directly from the exact solutions. The reference values for the normalizing factors, a_i, are the standard deviations of the stochastic coefficients Y_i^{DO} in the exact DO expansion, and the reference values for Y_i are the normalized stochastic coefficients Y_i^{DO}/a_i.

We compare the results obtained from the NN-DO method with the exact solutions. Figure 3a shows the evolution of the scaling factors a_i $(i = 1, 2)$ with time; they start from zero due to the deterministic initial condition, and increase monotonically until the convergence at $T = \pi$, indicating that the randomness in the system grows from zero to a fully developed state during the time period $t \in [0, \pi]$. Figure 3b shows the comparison of the DO bases obtained from the NN-DO method and the exact bases at $T = \pi$. The DO bases generated by the neural networks agree well with the reference solutions. Figure 3c shows the comparison of the stochastic coefficients Y_i $(i = 1, 2)$ versus the normalized exact DO coefficients Y_i^{DO} at four different times $t = 0, \pi/3, 2\pi/3$, and π. The random coefficients as functions of the random variable ξ evolve with time and develop a subtle wavy structure while preserving the orthogonality. The NN-DO method uncovers the evolution behavior of Y_i. The mean and variance of the NN-DO solution at $t = 2\pi/3$ and $t = \pi$ versus the exact ones are shown in Figures 3d and 3e, respectively. Apparently, the scale of variance is large compared to the scale of mean, indicating that the random fluctuation dominates the averaged solution profile. Table 1 summarizes the L_2 error (defined by $\|f_{NN} - f_{exact}\|_2$ for any function f) and the relative L_2 error (defined by $\|f_{NN} - f_{exact}\|_2/\|f_{exact}\|_2$) of the NN-DO results versus the exact solutions at the final time $T = \pi$, indicating the good performance of the NN-DO method.

5.1.2. Case 2: NN-BO method.

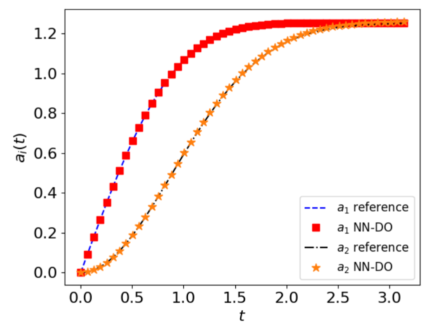
We solve the same problem (5.1) again, but this time we use the BO constraints by including (4.19) as part of the loss function. The initial conditions and reference solutions for the BO components, i.e., a_i, u_i, and Y_i, are the same as those of the previous case, and the neural networks used to approximate the BO components have the same structure that the networks used in the previous case. Table 2 summarizes the errors of the BO components at the final time $T = \pi$. The NN-BO method demonstrates very good performance similar to the NN-DO method.

To illustrate the effectiveness of the choice of weights in (4.21), we plot the value for each component of the loss function during the first 100000 epochs of training in Figure 4. The decay of the loss associated with the weak formulation $\mathrm{MSE_w}$ (from more than 0.1 to less than 0.001) is the main effect of the training process. The losses for the initial conditions ($\mathrm{MSE_{IC}}$) and the BO conditions ($\mathrm{MSE_{BO}}$) are kept small (around 10^{-5} to 10^{-4}), which shows that the whole training process is governed by the initial condition and the BO condition. The loss associated with the original equation ($\mathrm{MSE_0}$) is decaying, indicating that the result is getting closer to the desired solution, but due to the small weight, the contribution of this loss to the total loss is very limited.

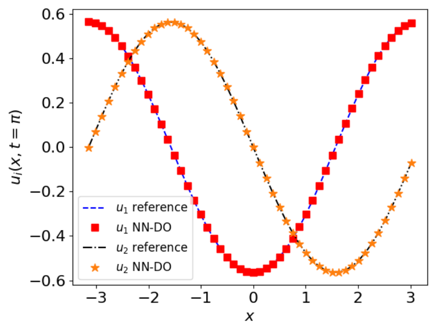
5.2. Application to nonlinear stochastic problem.

In this section, we apply the NN-DO/BO methods to solve nonlinear stochastic problems by considering the following stochastic Burgers' equation:

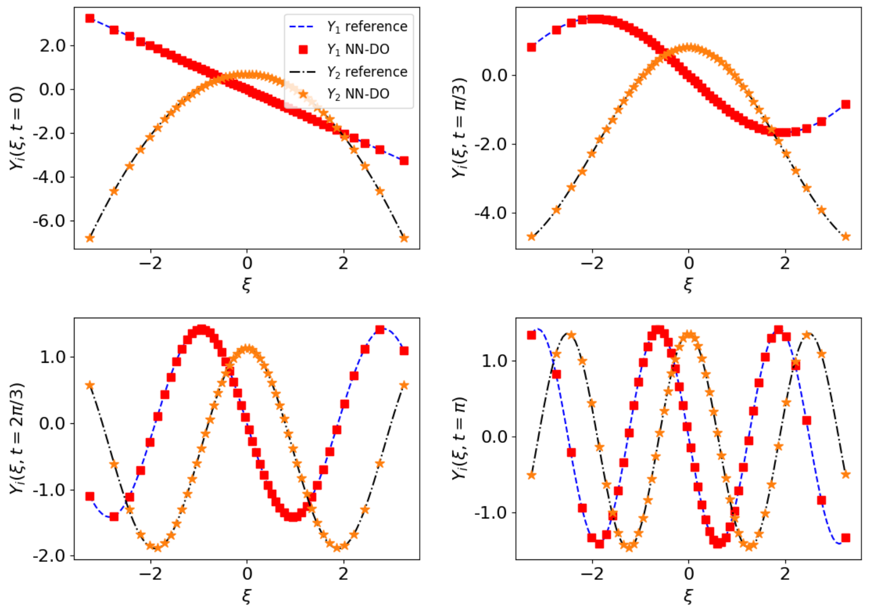
$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} + f(x, t; \omega) \quad \forall t \in [0, T] \quad \text{and} \quad x \in D, \tag{5.3}$$



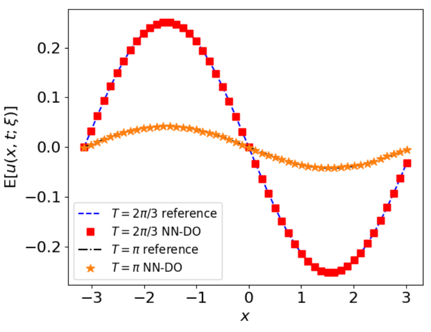
(a)



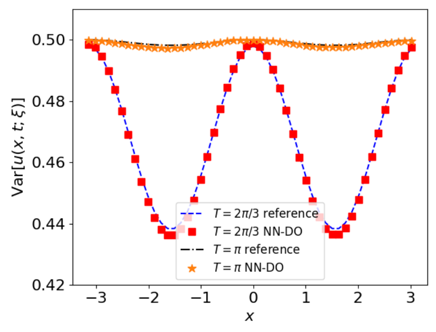
(b)



(c)



(d)



(e)

FIG. 3. *Stochastic advection equation (NN-DO).* (a) *The evolution of the scaling factors a_i.* (b) *The spatial bases u_i at the final time $t = \pi$.* (c) *The random coefficients Y_1 and Y_2 at $t = 0, \pi/3, 2\pi/3$, and π.* (d) *Solution mean at $t = 2\pi/3$ and π.* (e) *Solution variance at $t = 2\pi/3$ and π. We use the exact solutions as reference. The scattered points in* (b), (d), *and* (e) *denote the training points in the physical space, and the scatter points in* (c) *denote the training points in the probabilistic space.*

Table 1

Stochastic advection equation (NN-DO). The L_2 and relative L_2 errors of NN-DO solutions versus the exact solutions at the final time $T = \pi$.

	$\mathbb{E}[\boldsymbol{u}]$	$\textbf{Var}[\boldsymbol{u}]$	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$
L_2 error	0.0006	0.0006	0.0010	0.0051
Relative L_2 error	1.96%	0.11%	0.09%	0.55%
	$\boldsymbol{u_1}$	$\boldsymbol{u_2}$	$\boldsymbol{Y_1}$	$\boldsymbol{Y_2}$
L_2 error	0.0001	0.0002	0.0007	0.0013
Relative L_2 error	0.04%	0.04%	0.52%	0.93%

Table 2

Stochastic advection equation (NN-BO). The L_2 and relative L_2 errors of NN-BO solutions versus the exact solutions at the final time $T = \pi$.

	$\mathbb{E}[\boldsymbol{u}]$	$\textbf{Var}[\boldsymbol{u}]$	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$
L_2 error	0.0006	0.0006	0.0009	0.0054
Relative L_2 error	1.98%	0.13%	0.08%	0.59%
	$\boldsymbol{u_1}$	$\boldsymbol{u_2}$	$\boldsymbol{Y_1}$	$\boldsymbol{Y_2}$
L_2 error	0.0051	0.0047	0.0019	0.0019
Relative L_2 error	1.27%	1.18%	1.33%	1.36%

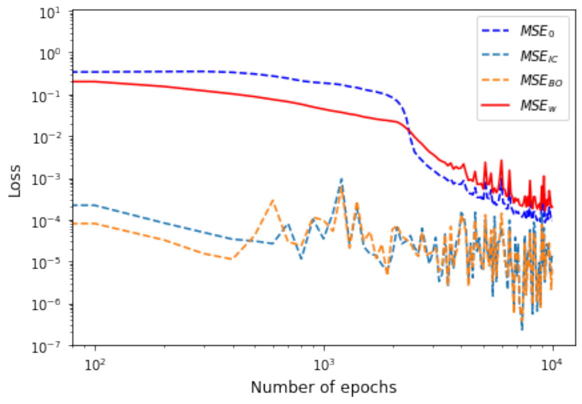


Fig. 4. *Stochastic advection equation (NN-BO). Various components of the loss function during the training process.*

where the physical domain D is $[-\pi, \pi]$, and $\nu = 0.1$ is the viscosity coefficient. Suppose that the random forcing term $f(x, t; \omega)$ is parameterized by two identically independent uniformly distributed random variables in $[0, 1]$, denoted by $\xi_1(\omega)$ and $\xi_2(\omega)$. Then, the stochastic behavior of solution $u(x, t; \omega)$ can be fully described by ξ_1 and ξ_2, too. In this example, we create a manufactured solution $u(x, t; \xi_1, \xi_2)$ such that the exact DO and BO components can be calculated explicitly. The manufactured solution is

$$\begin{aligned} u(x, t; \xi_1, \xi_2) = &- \sin(x - t) - \sqrt{3}(1.5 + \sin(t)) \cos(x - t)(2\xi_1 - 1) \\ &+ \sqrt{3}(1.5 + \cos(3t)) \cos(2x - 3t)(2\xi_2 - 1). \end{aligned} \tag{5.4}$$

The random forcing term $f(x, t; \omega)$ can be calculated given the manufactured solution. Due to its lengthy expression, here we omit writing the explicit formula for $f(x, t; \omega)$. The explicit expressions of the DO and BO expansions are included in section SM2. By normalizing the bases and the random coefficients, both the DO expansion and

BO expansion yield the same expression in the form of (4.8):

$$\begin{aligned}
&u_1(x,t) = -\frac{1}{\sqrt{\pi}}\cos(x-t), && u_2(x,t) = \frac{1}{\sqrt{\pi}}\cos(2x-3t),\\
&a_1(t) = \sqrt{\pi}(1.5+\sin(t)), && a_2(t) = \sqrt{\pi}(1.5+\cos(3t)),\\
&Y_1(t;\xi_1,\xi_2) = 2\xi_1 - 1, && Y_2(t;\xi_1,\xi_2) = 2\xi_2 - 1.
\end{aligned} \tag{5.5}$$

We obtain the solution until $T = 10\pi$ to demonstrate the long-term performance of the NN-DO/BO method. In practice, we divide the time domain into ten non-overlapping subdomains of equal length, each of which has the length π. In each subdomain the expansion components are approximated by an independent set of feed-forward neural networks. We train the time domains one after another and use the results from the previous interval at the end time as the initial conditions for the next subdomain. This domain decomposition strategy circumvents the difficulty of approximating functions of massive fluctuations with a single neural network, and thus will make the training process easier. We use an equal number of training points for all time subdomains, and set $n_t = 30$ and $n_x = 50$. Again, the samples of $\{t_c^s\}_{s=1}^{n_t}$ are drawn from a uniform distribution, and the spatial training points $\{x_c^k\}_{k=1}^{n_x}$ are equidistantly distributed in $[-\pi,\pi]$. For the training points in the stochastic space, we use the eighth-order Gauss–Legendre quadrature rule for both ξ_1 and ξ_2, generating 64 points in the probabilistic space. The same neural network setups are implemented for the following two test cases: the $\bar{u}_{nn}$, A_{nn}, and Y_{nn} networks all have three hidden layers, each of which has 32 neurons, and the U_{nn} network is constructed with three hidden layers and 64 neurons per hidden layer. We only change the loss function in favor of either the DO or BO condition. The neural networks are trained with an Adam optimizer (learning rate 0.001) for 50000 epochs.

5.2.1. Case 1: NN-DO method. First, we test the NN-DO method. The initial conditions are taken directly from (5.5). Figure 5 shows the relative L_2 errors of the solution mean and variance when compared to the exact solution, and in Table 3 we report both errors for all of the DO components at the final time $T = 10\pi$. All relative L_2 errors are around or less than 1%, indicating the good performance of the proposed NN-DO method.

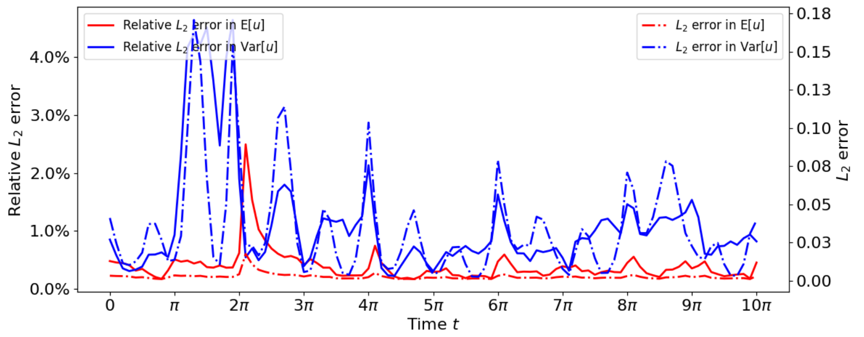


FIG. 5. *Stochastic Burgers' equation (NN-DO). The L_2 and relative L_2 errors in the mean and variance obtained by the NN-DO method. We use the exact solutions for reference.*

5.2.2. Case 2: NN-BO method. Now, we use the BO constraints to train the neural networks. Figure 6a shows the evolution of a_i $(i = 1, 2)$ as time grows, where

TABLE 3
Stochastic Burgers' equation (NN-DO). The L_2 and relative L_2 errors of NN-DO solutions versus the exact solutions at the final time $T = 10\pi$.

	$\mathbb{E}[\boldsymbol{u}]$	**Var**$[\boldsymbol{u}]$	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$
$\boldsymbol{L_2}$ **error**	0.0029	0.0278	0.0104	0.0084
Relative $\boldsymbol{L_2}$ **error**	0.40%	0.57%	0.35%	0.28%
	$\boldsymbol{u_1}$	$\boldsymbol{u_2}$	$\boldsymbol{Y_1}$	$\boldsymbol{Y_2}$
$\boldsymbol{L_2}$ **error**	0.0042	0.0021	0.0008	0.0004
Relative $\boldsymbol{L_2}$ **error**	1.04%	0.53%	0.62%	0.34%

the low frequency component, a_1, and the high frequency component, a_2, co-exist at the same amplitude. Note that the scaling factors a_i correspond to the eigenvalues in the standard BO method, and they cross frequently during the whole time evolution, and also within each time subdomain. In this situation, the standard BO method would fail due to the lack of explicit formulas for matrices M and S in (3.15). The proposed NN-BO method does not suffer from this issue. In Figure 6b, we compare the bases u_i $(i = 1, 2)$ at $t = 10\pi$ obtained from the NN-BO method to the exact solutions, and we plot the NN-BO solution mean (Figure 6c) and solution variance (Figure 6d) versus the exact values at time $t = 5\pi$ and 10π. It is evident that the NN-BO solutions agree with the exact reference solutions very well. From Figure 6d we can observe that the solution variance evolves from $t = 5\pi$ to $t = 10\pi$ to develop a greater magnitude range and a more complex shape, and the NN-BO method precisely captures this progress. Figure 6e shows the relative L_2 errors of the solution mean and variance, and in Table 4 we report both errors for all of the BO components at the final time $T = 10\pi$. All relative L_2 errors are less than 1%, indicating the good performance of the proposed NN-BO method for this nonlinear example where a significant amount of eigenvalue crossings happen.

5.3. Application to nonlinear diffusion-reaction equation. Consider the following reaction diffusion equation with a nonlinear source term:

$$\frac{\partial u}{\partial t} = au_{xx} + bu^2 + f(x;\omega) \quad \forall t \in [0,1] \text{ and } x \in [-1,1], \tag{5.6}$$

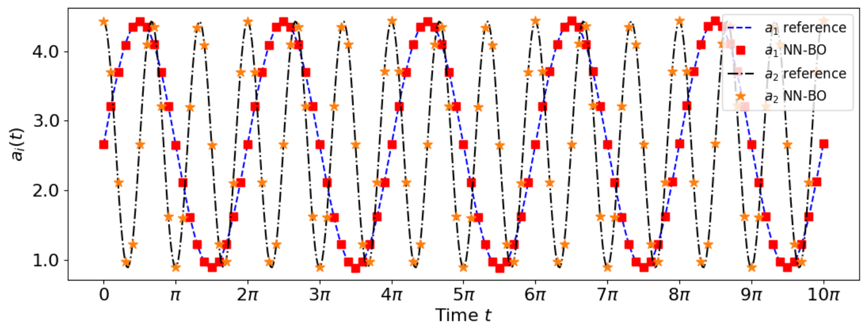
where the random force $f(x;\omega) = (1 - x^2)g(x;\omega)$ is the source of randomness, while a and b are time-independent diffusion and reaction coefficients, respectively. The random process $g(x;\omega)$ is modeled as a Gaussian random field, i.e., $g(x;\omega) \sim \mathcal{GP}(1, C(x_1, x_2))$, where $C(x_1, x_2)$ is a squared exponential kernel with standard deviation σ_g and correlation length l_c:

$$C(x_1, x_2) = \sigma_g^2 \exp\left(-\frac{(x_1 - x_2)^2}{l_c^2}\right). \tag{5.7}$$

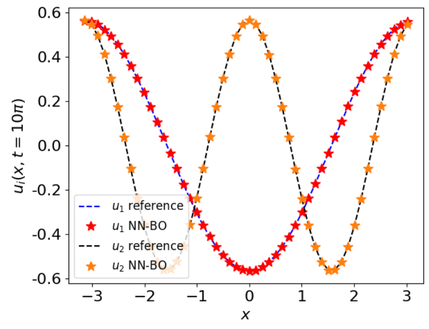
The solution satisfies the Dirichlet boundary conditions $u(-1, t;\omega) = u(1, t;\omega) = 0$, and the deterministic initial condition $u(x, 0;\omega) = -\sin(\pi x)$. We consider two different scenarios here:

- Forward problem: the coefficients a and b are given, and we solve for $u(x, t;\omega)$.
- Inverse problem: the coefficients a and b are unknown but additional information for $u(x, t;\omega)$ is given; we solve for $u(x, t;\omega)$, while we also aim to identify a and b.

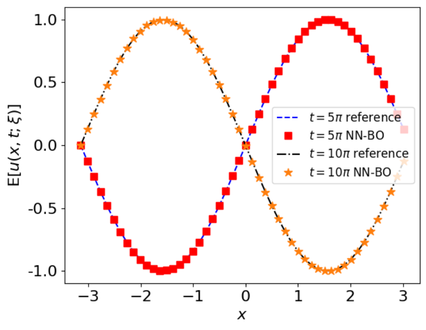
For brevity, here we only show the results obtained from the NN-BO method, as the NN-DO method exhibits a similar performance.



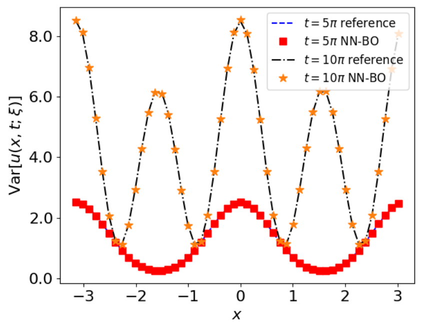
(a)



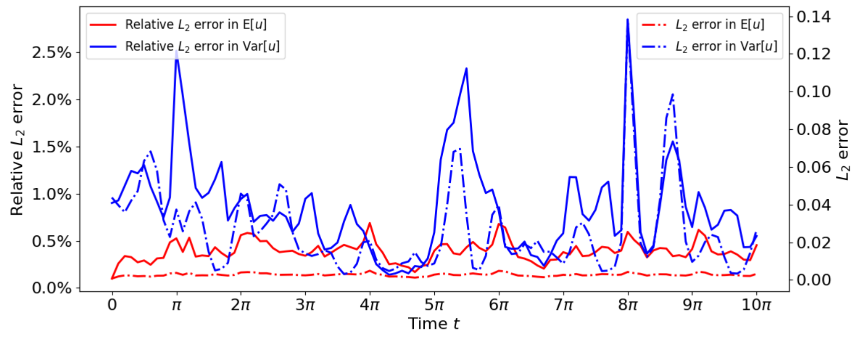
(b)



(c)



(d)



(e)

FIG. 6. *Stochastic Burgers' equation (NN-BO).* (a) *The evolution of the scaling factors a_i; they correspond to the eigenvalues in the standard BO method. There are many eigenvalue crossings during the whole time and within each time subdomain. Therefore, the standard BO method cannot be directly applied to this case.* (b) *The spatial bases u_i at the final time $t = 10\pi$.* (c) *Solution mean at $t = 5\pi$ and 10π.* (d) *Solution variance at $t = 5\pi$ and 10π.* (e) *The L_2 and relative L_2 error in the mean and variance calculated by the NN-BO method. We use the exact solutions as reference. The scattered points in* (b), (c), *and* (d) *denote the training points in the physical space.*

TABLE 4
Stochastic Burgers' equation (NN-BO). The L_2 and relative L_2 errors of NN-BO solutions versus the exact solutions at the final time $T = 10\pi$.

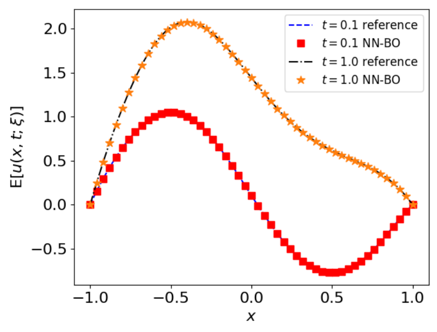
	$\mathbb{E}[\boldsymbol{u}]$	$\mathbf{Var}[\boldsymbol{u}]$	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$
L_2 error	0.0032	0.0267	0.0055	0.0073
Relative L_2 error	0.45%	0.55%	0.19%	0.25%
	$\boldsymbol{u_1}$	$\boldsymbol{u_2}$	$\boldsymbol{Y_1}$	$\boldsymbol{Y_2}$
L_2 error	0.0018	0.0020	0.0007	0.0005
Relative L_2 error	0.45%	0.49%	0.59%	0.39%

5.3.1. Forward problem. We set the diffusion coefficient $a = 0.1$ and the reaction coefficient $b = 0.5$. For the random force $f(x;\omega)$, we set $\sigma_g = 1$ and $l_c = 0.1$, thus requiring 19 KL modes to capture at least 98% of the fluctuation energy of $f(x;\omega)$. The neural networks used in the NN-BO method are built as follows: $\bar{u}_{nn}$ has three hidden layers with 32 neurons per layer, U_{nn} and Y_{nn} have three hidden layers with 64 neurons per layer, and A_{nn} is composed of N independent neural networks (N is the number of BO expansion terms), each of which has three hidden layers with four neurons per layer, approximating one single scaling factor a_i. This is because we expect that a_i may oscillate greatly in vastly different scales during the time evolution. We use $n_x = 51$ equidistantly distributed training points $\{x_c^k\}_{k=1}^{n_x}$ in space, $n_t = 50$ uniformly distributed training points $\{t_c^s\}_{k=1}^{n_t}$ in the time domain, and $n_l = 1000$ random samples $\{\xi_c^l\}_{l=1}^{n_\xi}$ in the 19-dimensional random space. The neural networks are trained with an Adam optimizer (learning rate 0.001) for 300000 epochs.

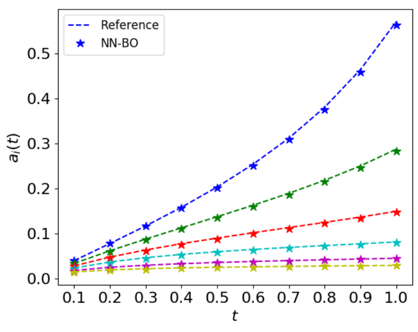
First, we investigate the performance of NN-BO method using six BO expansion terms. To obtain the reference solution for the BO decomposition, we numerically solved the original BO equations with the finite difference scheme in space and a 3rd-level Adams–Bashforth scheme in time. Due to the deterministic initial condition, in practice we start with a Monte Carlo method until $t = 0.01$, and then switch to solving the BO equations. To obtain the reference for the solution statistics we solve the SPDE using a Monte Carlo method with 1000 samples.

Figure 7a shows the NN-BO solution mean at $t = 0.1$ and $t = 1.0$, and Figure 7b shows the evolution of the scaling factors a_i, where the first four BO modes gradually pick up energy as the result of the nonlinear source term, while the energy in the fifth and sixth modes is relatively stable in time. This illustrates the efficiency of the BO representation, i.e., only a small number of modes is necessary to capture most of the stochasticity in this 19-dimensional SPDE. Figure 7c compares the modal functions learned from the NN-BO method with the reference, and Table 5 displays the root mean squared error of the random coefficients Y_i. The proposed NN-BO method generates accurate predictions at both the early stage of the solution ($t = 0.1$) and the end time ($t = 1.0$).

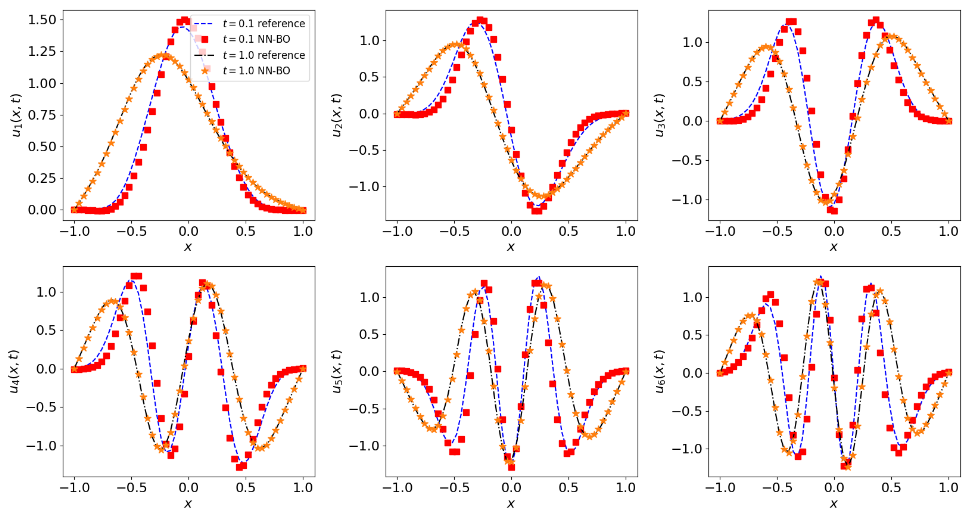
Next, we analyze the effect of the number of BO expansion modes by comparing the variances of solution calculated using five, six, and seven BO modes, and moreover, we solve the diffusion-reaction equation with noisy sensor data as the initial condition. Figure 8a shows the noisy sensor measurements of $u(x, t = 0)$, where the 30 sensors are uniformly placed in the domain, and the red dots are perturbed measurements generated by artificially adding independent Gaussian random noise of standard deviation 0.1 to the hidden true values. Figure 8b shows a comparison of the NN-BO solution mean and standard deviation, calculated based on noisy sensor data, and the reference mean and standard deviation, obtained with the Monte Carlo simulation. Figure 9a shows the predicted variance at time $t = 1.0$ versus the reference solution.



(a)



(b)



(c)

FIG. 7. *Stochastic diffusion-reaction equation (forward problem).* (a) *Solution mean at $t = 0.1$ and $t = 1.0$; the reference mean is calculated from the Monte Carlo simulation.* (b) *Scaling factors a_i at different time steps.* (c) *The BO modes u_i at $t = 0.1$ and 1.0. The reference solutions in* (b) *and* (c) *are calculated using the standard numerical BO method.*

TABLE 5
Stochastic diffusion-reaction equation (forward problem). Root mean squared error of the random coefficients Y_i calculated using the NN-BO method at $t = 0.1$ and $t = 1.0$; the reference Y_i is calculated using the standard numerical BO method.

RMSE	$\mathbf{Y_1}$	$\mathbf{Y_2}$	$\mathbf{Y_3}$	$\mathbf{Y_4}$	$\mathbf{Y_5}$	$\mathbf{Y_6}$
$t = 0.1$	0.098	0.175	0.225	0.292	0.237	0.275
$t = 1.0$	0.042	0.039	0.045	0.050	0.061	0.057

The NN-BO method slightly underestimates the variance due to the truncated expansion, and using noisy sensor measurements as the initial condition does not change the prediction at final time too much. Figure 9b compares the relative L_2 error of the solution variance at $t = 1.0$ obtained using three different methods: NN-BO, gPC, and the standard BO. The gPC method generates the largest error as it fails to capture the evolution of the system's stochastic structure due to the nonlinearity; therefore,

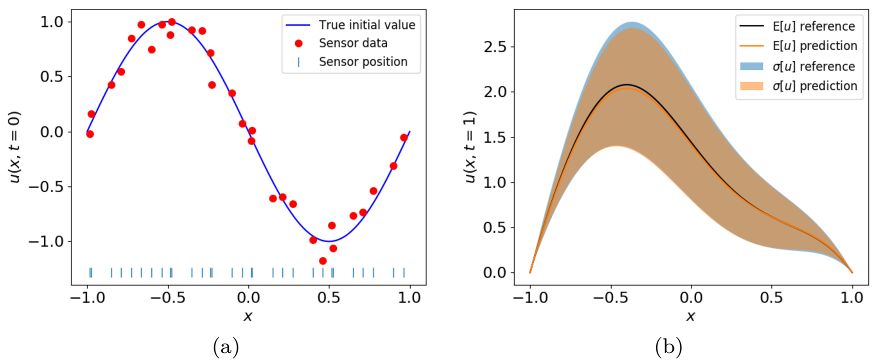


FIG. 8. *Stochastic diffusion-reaction equation with noisy data as initial condition.* (a) *Noisy sensor data as initial condition.* (b) *Mean and standard deviation of the predicted solution $u(x, t; \omega)$ versus the reference mean and standard deviation.*

to achieve the same accuracy, one has to include a larger number of modes using the gPC method than using the BO method. Again, we can observe that better accuracy can be achieved when more modes are included, and we obtain similar accuracy when using noisy sensor data as the initial condition. The NN-BO method is less accurate than the standard numerical BO method due to dominant optimization errors. However, it circumvents the need to generate artificial stochastic initial conditions and can make use of scattered, noisy sensor measurements as constraints, rather than explicit mathematical expressions. Another advantage of the NN-BO method over the standard BO method is that it can efficiently solve a time-dependent nonlinear *inverse* stochastic problem.

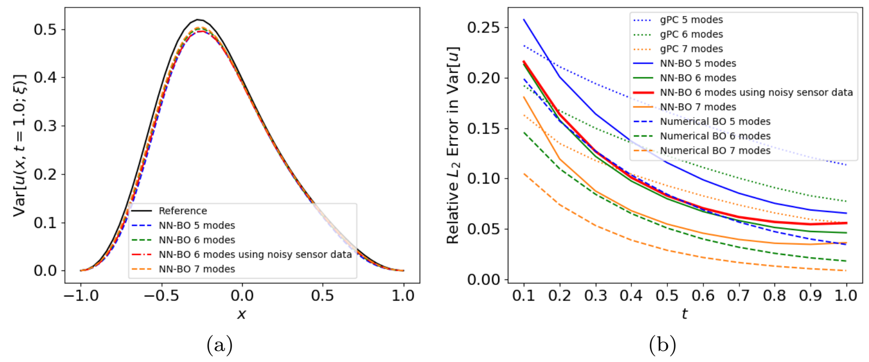


FIG. 9. *Stochastic diffusion-reaction equation (forward).* (a) *Variance of the NN-BO solution calculated using 5, 6, and 7 modes; the reference variance is calculated from the Monte Carlo simulation.* (b) *Comparison of the L_2 errors of the solution variance calculated by the NN-BO method, the standard numerical BO method, and the gPC method. The gPC method generates the largest error since it fails to capture the dynamic evolution of stochastic basis for nonlinear problems.*

5.3.2. Inverse problem. Here again we solve (5.6), but this time we assume that we do not know the exact diffusion and reaction coefficients a and b. Some additional information about $u(x, t; \omega)$ is provided to help us infer these two coeffi-

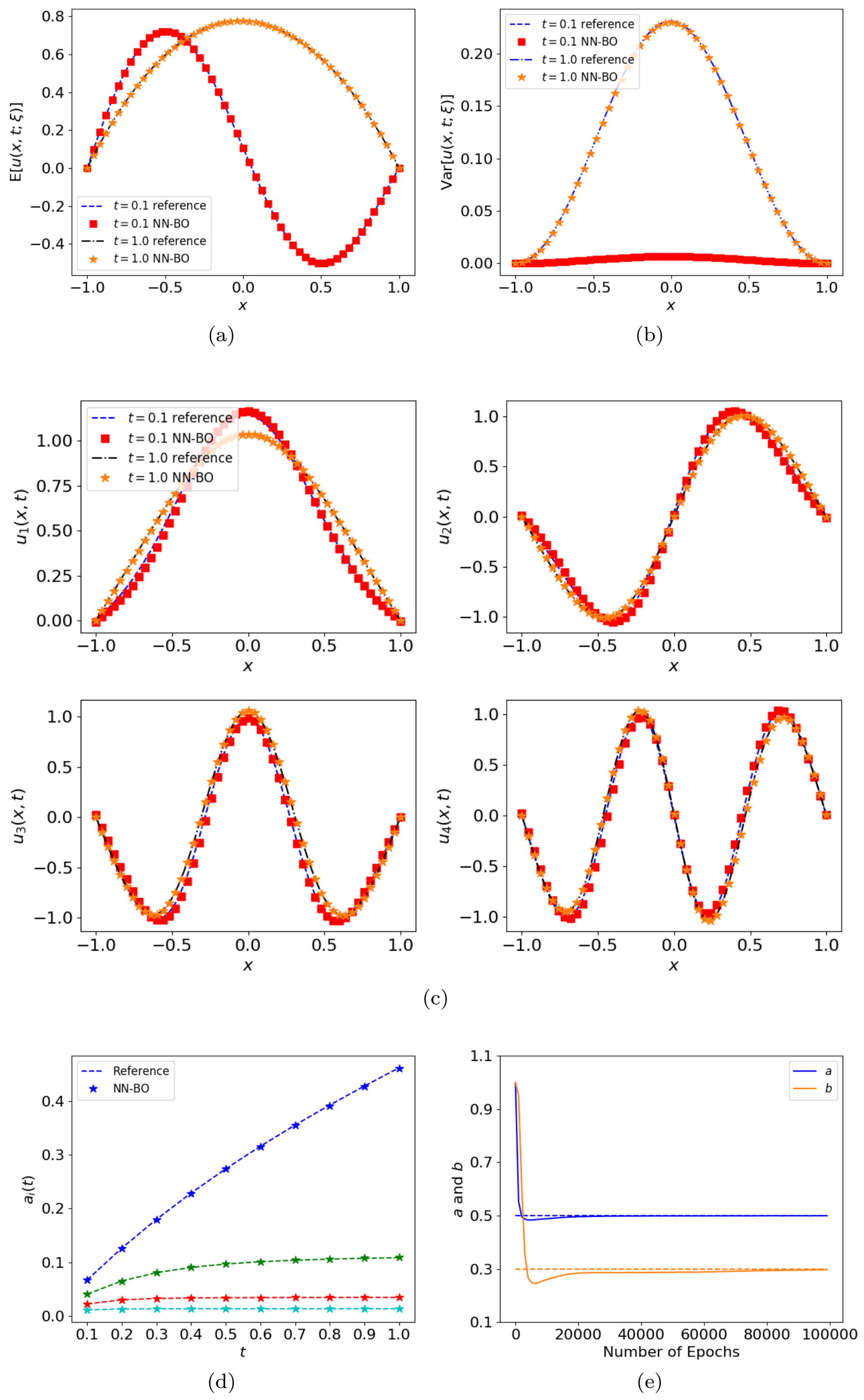


Fig. 10. *Stochastic diffusion-reaction equation (inverse problem).* (a) *Solution mean at* $t = 0.1$ *and* $t = 1.0$. (b) *Solution variance at* $t = 0.1$ *and* $t = 1.0$. (c) *The BO modes* u_i *at* $t = 0.1$ *and* $t = 1.0$. (d) *The evolution of the scaling factors* a_i. (e) *Convergence of predicted* a *and* b *to the true hidden values during the training process. The reference solutions in* (a) *and* (b) *are calculated by solving the forward problem using Monte Carlo simulation; the reference solutions in* (c) *and* (d) *are calculated by numerically solving the forward problem using the standard BO method.*

cients. In this example, the extra information is the mean value of $u(x, t; \omega)$ evaluated at three locations $x = -0.5, 0, 0.5$ and at two times $t = 0.1, 0.9$, i.e., a total of six measurements of $\mathbb{E}[u]$. We set $\sigma_g = 1$ and $l_c = 0.4$, and the "hidden" values of a and b are selected to be 0.5 and 0.3, respectively. To solve this inverse problem, we use a BO representation with four modes, and adopt the same setup of the neural networks and training points employed in the forward problem. When setting up the PINNs, a and b are coded as "variables" instead of as "constants" so that they will be tuned at the training stage. Meanwhile, we include an additional term in the loss function that calculates the MSE of the predicted $\bar{u}_{nn}(x, t)$ versus the measurement data, so that the loss function will make use of the extra information to infer the coefficients. Without loss of generality, we choose both the initial values of a and b to be 1.0, and in practice these values could be chosen based on reasonable guesses. The neural networks are trained with the Adam optimizer (learning rate 0.001) for 300000 epochs. As in the forward problem, the reference solution statistics are calculated with Monte Carlo simulation, and the reference BO components are generated by numerically solving the BO equations.

TABLE 6

Stochastic diffusion-reaction equation (inverse problem). Root mean squared error of the random coefficients Y_i calculated using the NN-BO method at $t = 0.1$ and $t = 1.0$; the reference Y_i is calculated from the forward problem using the standard numerical BO method.

RMSE	$\boldsymbol{Y_1}$	$\boldsymbol{Y_2}$	$\boldsymbol{Y_3}$	$\boldsymbol{Y_4}$
$t = 0.1$	0.061	0.088	0.075	0.084
$t = 1.0$	0.019	0.041	0.039	0.060

Figures 10a and 10b shows the predicted solution mean and variance, respectively. Figures 10c and 10d show the predicted BO modes u_i and the scaling factors a_i. Table 6 displays the root mean squared errors of the random coefficients Y_i. It is evident that when compared to the reference solutions, the NN-BO method is still accurate for solving the inverse problem. Finally, we display the convergence history of the predicted a and b in Figure 10e, and we can observe that the inferred values converge to the true values after less than 100000 training epochs.

6. Summary.

In this paper, we have presented two methods for solving time-dependent stochastic partial differential equations (SPDEs), i.e., the NN-DO method and the NN-BO method. They both make use of the expressiveness of physics-informed neural networks (PINNs). Similar to the standard dynamically orthogonal (DO) and bi-orthogonal (BO) methods, the proposed methods use either dynamical constraints on the spatial bases (NN-DO), or static constraints on both the spatial and the stochastic bases (NN-BO) to remove the time redundancy of the generalized Karhunen–Lòeve expansion. Since the loss functions of neural networks can be directly established from an implicit form of the DO/BO constraints, the proposed methods are free from the assumptions needed for deriving the standard DO and BO equations, and thus they can be applied to a broader range of UQ problems. We demonstrated the performance of the NN-DO/BO methods with two artificially designed benchmark cases where exact DO/BO solutions can be derived, and we applied the NN-BO method to solve a time-dependent nonlinear diffusion reaction equation. Our numerical results show that the proposed NN-DO/BO methods are accurate for SPDEs with deterministic initial conditions and frequent eigenvalue crossings, and are reliable for long-time integration and high-dimensional random input. Moreover, additional flexibility over the standard BO/DO methods was demonstrated by the proposed methods for solving SPDEs by making direct use of the noisy scattered measurement data. These methods seamlessly solve the time-dependent stochastic inverse problems by encoding the extra information into the loss function, while tuning the hidden parameters at the training stage. These advantages were demonstrated in the last numerical example, which exhibits the true potential of the NN-DO/BO method when applied to real physics/engineering applications.

The effect of the number of training points for the NN-DO/BO method is yet to be discovered and thus remains an important research topic in our future research. The limitations of the NN-DO/BO methods are two-fold. The first limitation is related to limited accuracy, i.e., the absolute errors cannot reach levels below about 10^{-5} due to the inherent inaccuracy of solving a nonconvex optimization problem with no theoretical guarantees of a global minimum. Another limitation is the excessive cost associated with training the NN-DO/BO methods, especially for long-time integration. To this end, a promising approach is the use of parallel algorithms in time, such as the parareal algorithm [13]. For example, in the Burgers' equation example we could train all of ten time-subdomains simultaneously and use the parareal algorithm iteration to obtain continuous-in-time solutions. This will be particularly effective if we use many time-subdomains that can be trained in parallel. In fact, our preliminary experiments suggest that PINN training can be greatly accelerated using this approach for time-dependent PDEs; this concept can also be extended to domain decomposition in space as well.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, D. Matthieu, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: A system for large-scale machine learning*, in 12th USENIX Symposium on Operating Systems Design and Implementation (2016), 2016, pp. 265–283, https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[2] H. Babaee, M. Choi, T. P. Sapsis, and G. E. Karniadakis, *A robust bi-orthogonal/dynamically-orthogonal method using the covariance pseudo-inverse with application to stochastic flow problems*, J. Comput. Phys., 344 (2017), pp. 303–319.

[3] T. Chen and H. Chen, *Approximations of continuous functionals by neural networks with application to dynamic systems*, IEEE Trans. Neural Netw., 4 (1993), pp. 910–918.

[4] T. Chen and H. Chen, *Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems*, IEEE Trans. Neural Netw., 6 (1995), pp. 911–917.

[5] M. Cheng, T. Y. Hou, and Z. Zhang, *A dynamically bi-orthogonal method for time-dependent stochastic partial differential equations* I: *Derivation and algorithms*, J. Comput. Phys., 242 (2013), pp. 843–868.

[6] M. Cheng, T. Y. Hou, and Z. Zhang, *A dynamically bi-orthogonal method for time-dependent stochastic partial differential equations* II: *Adaptivity and generalizations*, J. Comput. Phys., 242 (2013), pp. 753–776.

[7] M. Choi, *Time-dependent Karhunen-Loève Type Decomposition Methods for SPDEs*, Ph.D. thesis, Brown University, Providence, RI, 2014.

[8] M. Choi, T. P. Sapsis, and G. E. Karniadakis, *A convergence study for SPDEs using combined polynomial chaos and dynamically-orthogonal schemes*, J. Comput. Phys., 245 (2013), pp. 281–301, https://doi.org/10.1016/j.jcp.2013.02.047.

[9] M. Choi, T. P. Sapsis, and G. E. Karniadakis, *On the equivalence of dynamically orthogonal and bi-orthogonal methods: Theory and numerical simulations*, J. Comput. Phys., 270 (2014), pp. 1–20.

[10] G. Cybenko, *Approximation by superpositions of a sigamoidal function*, Math. Contr., Signals Syst., 2 (1989), pp. 303–314.

[11] I. E. Lagaris, A. C. Likas, and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE Trans. Neural Netw., 9 (1998), pp. 987–1000,

https://doi.org/10.1109/72.712178.

[12] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, *Neural-network methods for boundary value problems with irregular boundaries*, IEEE Trans. Neural Netw., 11 (2000), pp. 1041–1049, https://doi.org/10.1109/72.870037.

[13] J.-L. Lions, Y. Maday, and G. Turinici, *Résolution d'edp par un schéma en temps «pararéel»*, C. R. Acad. Sci. Paris Sér. I Math., 332 (2001), pp. 661–668.

[14] E. Musharbash, F. Nobile, and T. Zhou, *Error analysis of the dynamically orthogonal approximation of time dependent random PDEs*, SIAM J. Sci. Comput., 37 (2015), pp. A776–A810, https://doi.org/10.1137/140967787.

[15] A. Narayan and T. Zhou, *Stochastic collocation on unstructured multivariate meshes*, Commun. Comput. Phys., 18 (2015), pp. 1–36, https://doi.org/10.4208/cicp.020215.070515a.

[16] M. Raissi, *Forward-backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations*, preprint, https://arxiv.org/abs/1804.07010, 2018.

[17] M. Raissi and G. E. Karniadakis, *Hidden physics models: Machine learning of nonlinear partial differential equations*, J. Comput. Phys., 357 (2018), pp. 125–141, https://doi.org/10.1016/j.jcp.2017.11.039.

[18] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics Informed Deep Learning (Part* I*): Data-driven Solutions of Nonlinear Partial Differential Equations*, preprint, https://arxiv.org/abs/1711.10561, 2017.

[19] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics Informed Deep Learning (Part* II*): Data-driven Discovery of Nonlinear Partial Differential Equations*, preprint, https://arxiv.org/abs/1711.10566, 2017.

[20] M. Raissi, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, *Deep learning of vortex-induced vibrations*, J. Fluid Mech., 861 (2019), pp. 119–137.

[21] T. P. Sapsis, *Dynamically Orthogonal Field Equations for Stochastic Fluid Flows and Particle Dynamics*, Ph.D. thesis, MIT, Cambridge, MA, 2011.

[22] T. P. Sapsis and P. Lermusiaux, *Dynamically orthogonal field equations for continuous stochastic dynamical systems*, Phys. D, 238 (2009), pp. 2347–2360, https://doi.org/10.1016/j.physd.2009.09.017.

[23] T. P. Sapsis and P. F. J. Lermusiaux, *Dynamical criteria for the evolution of the stochastic dimensionality in flows with uncertainty*, Phys. D, 241 (2012), pp. 60–76.

[24] D. Subramani and P. F. J. Lermusiaux, *Energy-optimal path planning by stochastic dynamically orthogonal level-set optimization*, Ocean Modeling, 100 (2016), pp. 57–77.

[25] M. P. Ueckermann, P. F. J. Lermusiaux, and T. P. Sapsis, *Numerical schemes for dynamically orthogonal equations of stochastic fluid and ocean flows*, J. Comput. Phys., 233 (2013), pp. 272–294.

[26] D. Xiu, *Numerical Methods for Stochastic Computations: A Spectral Method Approach*, Princeton University Press, Princeton, NJ, 2010, https://doi.org/10.2307/j.ctv7h0skv.

[27] D. Xiu and J. S. Hesthaven, *High-order collocation methods for differential equations with random inputs*, SIAM J. Sci. Comput., 27 (2005), pp. 1118–1139, https://doi.org/10.1137/040615201.

[28] D. Xiu and G. E. Karniadakis, *The Wiener–Askey polynomial chaos for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2002), pp. 619–644, https://doi.org/10.1137/s1064827501387826.

[29] A. Yazdani, M. Raissi, and G. E. Karniadakis, *Hidden Fluid Mechanics: Navier-Stokes Informed Deep Learning from the Passive Scalar Transport*, preprint, https://arxiv.org/abs/1808.04327, 2018.

[30] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis, *Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems*, J. Comput. Phys., 397 (2019), 108850, https://doi.org/10.1016/j.jcp.2019.07.048.