IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 December 2022, accepted 11 December 2022, date of publication 15 December 2022,
date of current version 21 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3229696

RESEARCH ARTICLE

MGDGAN: Multiple Generator and Discriminator Generative Adversarial Networks for Solving Stochastic Partial Differential Equations

SUNGHA CHO AND MINSEOK CHOI
Department of Mathematics, Pohang University of Science and Technology (POSTECH), Pohang, Gyeongbuk 37673, Republic of Korea
Corresponding author: Minseok Choi (mchoi@postech.ac.kr)

This work was supported by the National Research Foundation of Korea under Grant NRF-2021R1C1C1007875.

ABSTRACT We propose novel structures of generator and discriminator in physics-informed generative adversarial networks called multiple-generator-and-discriminator generative adversarial networks (MGDGANs), that are designed to solve stochastic partial differential equations (SPDEs). MGDGANs for SPDEs consist of three steps: a generator that samples a solution to the SPDEs, a physics-informed operator that enforces the governing equation, and a discriminator that distinguishes between samples from the generator and training samples. Inspired by the polynomial chaos, we represent the solution by the inner product of functions in spatial and random variables, and model each function by a separate generator. We show that the proposed multiple generator structure offers huge computational savings in training and prediction. If multiple stochastic processes exist in the system, then a distinct discriminator is used for each of them. We show that the loss function obtained by these distinct discriminators provides an equivalent metric to the Wasserstein distance loss by a single discriminator, and provide numerical examples to demonstrate that these multiple discriminators enhance the training accuracy. Numerical examples are demonstrated to verify that the proposed model is efficient in computation and memory; the model reduces computing time by more than a factor of 10 and relative l2 error by about one-third in the SPDE example.

INDEX TERMS Computational modeling, deep learning, generative adversarial networks, physics-informed deep generative models, uncertainty quantification.

I. INTRODUCTION

Stochastic partial differential equations (SPDEs) are used in a large range of fields, for example, weather forecasting [1], [2], chemical reaction flow [3], oceanographic modeling [4], and drone communications [5], [6]. Given data from a problem, SPDEs capture uncertainties of missing data of the solutions. However, solving SPDEs requires significant computational resources because the parameters are stochastic processes that necessitate numerous random variables to approximate them. In general, the solutions require a number of polynomial basis terms that increases exponentially as the stochastic dimension increases; this ''curse of dimensionality'' in uncertainty quantification is a long-standing problem that greatly increases the computation time. Numerous methods have been devised to alleviate this problem, including sparse grids and ANOVA decompositions [7], [8], but these methods only reduce the order of the number of basis terms and do not solve the problem.

Neural networks (NNs) have shown the potential to overcome the curse of dimensionality [9]. NN models have made breakthroughs in many applications such as object detection [10], Covid-19 diagnosis [11], speech recognition [12], natural language processing [13], image generation [14], recommendation systems [15], cosmological simulation [16], robot systems [17], and anomaly detection [18]. Many attempts have been made to solve deterministic partial differential equations (PDEs) by using deep neural networks (DNNs) [19], [20], [21], [22], [23], [24], [25], [26], [27], [28]. For this purpose, [20] and [21]

The associate editor coordinating the review of this manuscript and approving it for publication was Sudipta Roy.

used convolutional neural network (CNN) architectures, with [20] presenting Legendre polynomial expansion to train coefficients, and [21] using an autoregressive model to design time-dependent PDE solvers. To learn the differential equation solvers, [23] used Fourier neural operators, and [24] used deep operator networks. [29] and [28] discretized spatial derivatives using DNNs and learned PDEs by using the structures of finite difference and finite volume methods. [25] assigned differential operators to the NNs by using automatic differentiation; this model is called a physics-informed neural network (PINN).

Since these DNN models cannot capture the uncertainty of the solutions, probabilistic models have been developed. Under the assumption that we know the samples of input-output pair $((x, \xi), k(x, \xi))$ of a stochastic process k, where x is a spatial variable, and ξ is a random variable, [30] and [31] used NN surrogate $N(x, \xi)$ to find the solution of SPDEs. In contrast, many generative models have been applied to solve data-driven SPDEs [32], [33], [34], [35], [36], [37], [38]. The models learn PDE solutions from data samples of stochastic processes and are widely applicable in both forward and inverse problems. [33] and [35] used Bayesian NNs to learn the solutions (forward problem) and parameters (inverse problem) of PDEs. [37] applied dynamically orthogonal and bi-orthogonal conditions to NNs to calculate the quantity of interests of the solutions. [32] and [38] used normalizing flow models to solve SPDE problems. [34] and [36] used GANs to solve SPDEs; [34] predicted future distribution of solutions from the present distribution, and [36] performed automatic differentiation to solve high-dimensional SPDEs and the method is called physics-informed GAN (PI-GAN).

In this work, we propose multiple-generator-and-discriminator generative adversarial networks (MGDGANs) that effectively solve SPDE problems. The method uses GANs as in [36] and consists of a generating step, a physics-informed step, and a discriminating step. The generating step generates solutions by using generator networks, the physics-informed step computes the governing equation by automatic differentiation, and the discriminating step compares the computed values with training data by using discriminator networks. Inspired by the polynomial chaos, we represent the solution by the inner product of the functions in spatial and random variables, and each of these functions is modeled by separate generators. If multiple stochastic processes exist in the system, each is discriminated by distinct discriminators. The proposed multiple generator structure offers huge computational savings in the training and prediction, and the loss function obtained by these multiple discriminators provides an equivalent metric to the Wasserstein distance loss by a single discriminator. Numerical examples will be demonstrated to verify that the proposed model is efficient in computation and memory, and increases the training accuracy.

Main contributions: The main contributions of this work are summarized as follows.

- **Uncertainty Quantification:** The proposed model based on GANs is able to capture the uncertainty of the solution to SPDEs.
- **Fast computing time:** The proposed multiple generator structure offers huge computational savings in training and prediction.
- **Memory efficiency:** The memory usage is dramatically reduced by eliminating unnecessary repetition of spatial and random variables via multiple generator structures.
- **Better accuracy:** The proposed multiple discriminator structure decreases the generalization error.

The rest of the paper is organized as follows. Section II reviews GANs and physics-informed GANs. Section III proposes the multiple generator (MG) and multiple discriminator (MD) models, elucidates why the MG has a lower computational cost than the single generator (SG) model, and proves the equivalence of the MD with the single discriminator (SD) model. Section IV provides numerical examples of learning stochastic processes and solving SPDEs. Section V summarizes the paper and presents some remaining challenges.

II. REVIEW OF GANs AND PI-GANs

A. GENERATIVE ADVERSARIAL NETWORKS

1) VANILLA GAN

GANs are generative models that aim to approximate an unknown distribution $\mathbb{P}_r$ of given data points [39]. A GAN model is composed of two NNs: a generator G_θ and a discriminator D_ρ. The generator takes a noise random variable z from a known distribution $\mathbb{P}_z$ such as Gaussian and returns a sample that follows the pushforward measure $\mathbb{P}_g = \mathbb{P}_z(G_\theta^{-1})$. The discriminator takes a sample x and returns the probability that x comes from $\mathbb{P}_r$. The goal of the discriminator is to perfectly distinguish the samples of given data (real samples) from the samples generated by the generator (fake samples). In contrast, the generator is trained to deceive the discriminator to induce it to recognize $G_\theta(z)$ as a real sample. The following two-player game summarizes this process:

$$\min_\theta \max_\rho \mathbb{E}_{x\sim\mathbb{P}_r}[\log D_\rho(x)] + \mathbb{E}_{z\sim\mathbb{P}_z}[\log(1 - D_\rho(G_\theta(z)))]. \tag{1}$$

At the Nash equilibrium of the two-player game, the distributions of real and fake samples coincide, i.e., $\mathbb{P}_r = \mathbb{P}_g$. In practice, we start with N real samples $\{x_i\}_{i=1}^N$ of the data and use G_θ to generate the corresponding N fake samples $\{G_\theta(z_i)\}_{i=1}^N$, and perform Monte Carlo approximation to construct a loss function

$$\frac{1}{N}\sum_{i=1}^{N}[\log D_\rho(x_i) + \log(1 - D_\rho(G_\theta(z_i)))]. \tag{2}$$

The discriminator and generator are trained by gradient ascent and gradient descent, respectively.

2) WASSERSTEIN GANs

GANs can suffer from pathological problems such as mode collapse and gradient vanishing, which severely degrade the generator quality; several enhanced models have been developed to address this problem. This section briefly reviews one variant, the Wasserstein generative adversarial network (WGAN) [40]. To this end, we first introduce the Wasserstein distance

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Gamma(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|], \tag{3}$$

where $\Gamma(\mathbb{P}_r, \mathbb{P}_g)$ is a set of all joint distributions with marginal distributions $\mathbb{P}_r$ and $\mathbb{P}_g$. $W(\mathbb{P}_r, \mathbb{P}_g)$ is also known as the earth mover's distance because it represents the 'minimum change' from $\mathbb{P}_r$ to $\mathbb{P}_g$. Direct computation of the Wasserstein distance is intractable in general, and the Kantorovich-Rubinstein duality is used instead:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\|f\|_L \le 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)], \tag{4}$$

where the supremum is attained on the set of Lipschitz functions with the Lipschitz constant $\|f\|_L$ bounded by 1. Substituting f with an NN $\mathcal{D}_\rho$ and maximizing the equation with respect to the parameter ρ, we consider the following problem:

$$\max_\rho L(\rho, \theta) = \mathbb{E}_{x\sim\mathbb{P}_r}[D_\rho(x)] - \mathbb{E}_{z\sim\mathbb{P}_z}[D_\rho(G_\theta(x))]. \tag{5}$$

The optimal value $L(\rho^*, \theta)$ is an approximated Wasserstein distance, i.e., $L(\rho^*, \theta) \approx W(\mathbb{P}_r, \mathbb{P}_g)$. We solve $\min_\theta L(\rho^*, \theta)$ to obtain the optimal generator G_θ. A two-player game summarizes this procedure:

$$\begin{aligned}\min_\theta \max_\rho L(\rho, \theta) &= \mathbb{E}_{x\sim\mathbb{P}_r}[D_\rho(x)] - \mathbb{E}_{z\sim\mathbb{P}_z}[D_\rho(G_\theta(z))] \\ &\approx \frac{1}{N}\sum_{i=1}^{N}[D_\rho(x_i) - D_\rho(G_\theta(z_i)],\end{aligned} \tag{6}$$

where the Monte Carlo estimator is used for the last approximation. We call D_ρ the discriminator and G_θ the generator.

3) SPECTRAL NORMALIZATION

One of the difficulties with WGAN is that the discriminator network D_ρ must satisfy the Lipschitz condition $\|D_\rho\|_L \le 1$. We accomplish this goal by spectral normalization, which directly normalizes the Lipschitz constant $\|D_\rho\|_L$ by using the spectral norm $\|A\|_2 = \max_{x\neq 0}(\|Ax\|_2/\|x\|_2)$ of a matrix A [41]. Specifically, an NN consists of sequential evaluations of affine functions $a_i(x) = W_i x$ and a nonlinear activation function $\sigma(z)$, where the bias is omitted for simplicity. As a result, the Lipschitz norm of an NN $D_\rho(x) = (a_L \circ \sigma \circ a_{L-1} \circ \cdots \circ \sigma \circ a_1)(x)$ is bounded by the product of Lipschitz norms of the components:

$$\|D_\rho\|_L \le \|a_L\|_L \|\sigma\|_L \cdots \|\sigma\|_L \|a_1\|_L. \tag{7}$$

Use of a Lipschitz activation function ($\|\sigma\|_L \le 1$) such as $\sigma(x) = tanh(x)$ or $ReLU(x)$ yields

$$\|D_\rho\|_L \le \prod_{i=1}^{L} \|a_i\|_L. \tag{8}$$

Moreover, the Lipschitz norm of the affine function $a_i(x) = W_i x$ is equal to the spectral norm $\|W_i\|_2$ of the matrix W_i. Therefore, normalizing the discriminator by the product of spectral norms of all the matrices yields a Lipschitz discriminator:

$$\frac{D_\rho}{\prod_{i=1}^{L} \|W_i\|_2}, \tag{9}$$

this procedure is called spectral normalization. From now on, we use the WGAN with spectral normalization for the losses and denote the spectral normalized discriminator as D_ρ.

B. PHYSICS-INFORMED GANs

In this section, we introduce PI-GANs for use in solving SPDEs.

1) PROBLEM DESCRIPTION

Consider stochastic processes $k(x;\omega), f(x;\omega)$, and $b(x;\omega)$ with a spatial variable $x \in \mathbb{R}^{d_x}$ in a domain $\mathcal{D}$ and a random event ω from a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We solve the following form of SPDEs that include the stochastic processes:

$$\begin{aligned}\mathcal{L}(k(x;\omega), u(x;\omega), x) &= f(x;\omega), x \in \mathcal{D}, \omega \in \Omega, \\ \mathcal{B}(u(x;\omega), x) &= b(x;\omega), x \in \partial\mathcal{D}, \omega \in \Omega,\end{aligned} \tag{10}$$

where $\mathcal{L}$ is a differential operator, and $\mathcal{B}$ is a boundary operator.

In this paper, we consider forward problems: solve $u(x;\omega)$ given data snapshots of stochastic processes k, f, and b:

$$\begin{aligned}\{k^{(i)} &= (k(x^{k,1};\omega_i), \cdots, k(x^{k,m_k};\omega_i))\}_{i=1}^N, \\ \{f^{(i)} &= (f(x^{f,1};\omega_i), \cdots, f(x^{f,m_f};\omega_i))\}_{i=1}^N, \\ \{b^{(i)} &= (b(x^{b,1};\omega_i), \cdots, b(x^{b,m_b};\omega_i))\}_{i=1}^N,\end{aligned} \tag{11}$$

and we quantify the accuracy of the model by using the mean and standard deviation of the approximated solution $G^u(x,z) \approx u(x;\omega)$ obtained by the Monte Carlo method:

$$\begin{aligned}\mu(x) &\approx \frac{1}{N}\sum_{i=1}^{N} G^u(x, z_i), \\ \sigma(x) &\approx \sqrt{\frac{1}{N}\sum_{i=1}^{N} G^u(x, z_i)^2 - \mu(x)^2}.\end{aligned} \tag{12}$$

2) APPLY GANs TO SPDEs

We briefly introduce PI-GANs to solve SPDE problems and refer to [36] for details. To learn a stochastic process $k(x,\omega)$ with data snapshot samples $(k(x^1,\omega_i), \cdots, k(x^m,\omega_i))$, we use a generator $G^k_\theta(x,z)$ that takes both the spatial variable

x and the noise random variable z as inputs, and we have the training data and the generated samples:

$$\begin{aligned} k^{(i)} &= (k(x^1, \omega_i), \cdots, k(x^m, \omega_i)) \\ \tilde{k}_\theta^{(i)} &= (\tilde{k}(x^1, z_i), \cdots, \tilde{k}(x^m, z_i)) \\ &= (G_\theta^k(x^1, z_i), \cdots, G_\theta^k(x^m, z_i)). \end{aligned} \tag{13}$$

We train the discriminator and the generator by solving the following WGAN problem:

$$\min_\theta \max_\rho \frac{1}{N} \sum_{i=1}^{N} [D_\rho(k^{(i)}) - D_\rho(\tilde{k}_\theta^{(i)})]. \tag{14}$$

This process is equivalent to learning a distribution of a random vector $(k(x^1, \omega), \cdots, k(x^m, \omega))$ in $\mathbb{R}^m$, but the spatial variable x enables generalization of $G_\theta^k(x, z)$ for any x in the domain. Generating N snapshots with m sensors $\{(G_\theta^k(x^1, z_i), \cdots, G_\theta^k(x^m, z_i)\}_{i=1}^N$ requires mN NN evaluations; hence the computational cost is $O(mN)$. We call this a single generator (SG) model.

Bearing this in mind, we now demonstrate how to solve the SPDE problems of the form (10), given training samples in (11). First, we generate samples of k by using generator $\tilde{k}(x, z) = G_{\theta_k}^k(x, z)$ and samples of u by using generator $\tilde{u}(x, z) = G_{\theta_u}^u(x, z)$. The differential operator $\mathcal{L}$ and the boundary operator $\mathcal{B}$ are applied to G^k and G^u in aid of the automatic differentiation:

$$\begin{aligned} \hat{\mathcal{L}}_{\theta_u}(x, z) &= \mathcal{L}(\tilde{k}(x, z), \tilde{u}(x, z), x), \\ \hat{\mathcal{B}}_{\theta_u}(x, z) &= \mathcal{B}(\tilde{u}(x, z), x). \end{aligned} \tag{15}$$

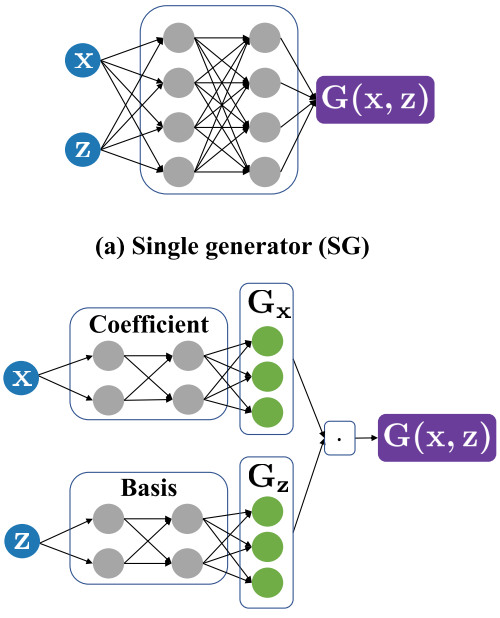
According to the governing equations, the distribution of $\hat{\mathcal{L}}_{\theta_u}(x, z)$ should be identical to that of f. Therefore, we compare the generated samples $\{\tilde{f}_{\theta_u}^{(i)} = (\hat{\mathcal{L}}_{\theta_u}(x^{f,1}, z_i), \cdots, \hat{\mathcal{L}}_{\theta_u}(x^{f,m_f}, z_i))\}_{i=1}^N$ to the training samples $\{f^{(i)} = (f(x^{f,1}, \omega_i), \cdots, (x^{f,m_f}, \omega_i))\}_{i=1}^N$, and perform a similar procedure for the boundary operator $\mathcal{B}$. In summary, the resulting samples are generated as follows:

$$\begin{aligned} \tilde{k}_{\theta_k}^{(i)} &= (G_{\theta_k}^k(x^{k,1}, z_i), \cdots, G_{\theta_k}^k(x^{k,m_k}, z_i)) \in \mathbb{R}^{m_k}, \\ \tilde{f}_{\theta_u}^{(i)} &= (\hat{\mathcal{L}}_{\theta_u}(x^{f,1}, z_i), \cdots, \hat{\mathcal{L}}_{\theta_u}(x^{f,m_f}, z_i)) \in \mathbb{R}^{m_f}, \\ \tilde{b}_{\theta_u}^{(i)} &= (\hat{\mathcal{B}}_{\theta_u}(x^{b,1}, z_i), \cdots, \hat{\mathcal{B}}_{\theta_u}(x^{b,m_b}, z_i)) \in \mathbb{R}^{m_b}, \end{aligned} \tag{16}$$

and the following WGAN problem trains the generators G^k and G^u:

$$\min_{\theta_k, \theta_u} \max_\rho \frac{1}{N} \sum_{i=1}^{N} [D_\rho(k^{(i)}, f^{(i)}, b^{(i)}) - D_\rho(\tilde{k}_{\theta_k}^{(i)}, \tilde{f}_{\theta_u}^{(i)}, \tilde{b}_{\theta_u}^{(i)})]. \tag{17}$$

After the training, G^u satisfies $\mathcal{L}(G^k(x, z), G^u(x, z), x) \approx f(x, \omega)$ and $\mathcal{B}(G^u(x, z), x) \approx b(x, \omega)$, and hence G^u is an approximated solution. In this case, we match the whole joint distribution of (k, f, b) by using a single discriminator D_ρ. We call this a single discriminator (SD) model. Note that the PI-GAN [36] employed a single generator and discriminator.



(a) Single generator (SG)

(b) Multiple generator (MG)

FIGURE 1. Comparison of the SG model and the proposed MG model. x denotes the spatial variable, and z is the noise random variable.

III. METHODS

In this section, we propose our methods: the multiple generator (MG) and the multiple discriminator (MG) models. The MG is a structure of generator for deterministic and random inputs, whereas the MD is a structure of discriminator for multiple stochastic processes.

A. MULTIPLE GENERATOR MODEL

The MG model is motivated by the polynomial chaos [42] and is composed of two NNs of output dimension p that separate the spatial variable x and the noise random variable z:

$$\begin{aligned} G_x(x) &= (G_{1,x}(x), \cdots, G_{p,x}(x)), \\ G_z(z) &= (G_{1,z}(z), \cdots, G_{p,z}(z)). \end{aligned} \tag{18}$$

The model output is an inner product of G_x and G_z:

$$\begin{aligned} G(x, z) &= G_x(x) \cdot G_z(z) \\ &= \sum_{i=1}^{p} G_{i,x}(x) G_{i,z}(z). \end{aligned} \tag{19}$$

We call $G_z(z)$ the 'basis network' and $G_x(x)$ the 'coefficient network'. The schematic of the SG and MG models is in Fig. 1. The MG model also has been used in [43]; however, the paper focused primarily on meta-learning of historical data by using GANs and deep operator networks and not on the efficiency of the MG model.

We consider the computational complexity and memory efficiency of the MG model. To generate N data snapshots with m sensors, the SG calculates $\{G(x_j, z_i) : i = 1, \cdots, N, j = 1, \cdots, m\}$, and hence requires mN network evaluations. In contrast, the MG evaluates m networks $\{G_x(x_j) : j = 1 \cdots, m\}$ and N networks

TABLE 1. Per-layer complexity of the single generator (SG) and the multiple generator (MG) models.

Model	Complexity per Layer	Complexity of Inner Product
SG	$O(mNd^2)$	-
MG	$O((m+N)d^2)$	$O(mNp)$

$\{G_z(z_i) : i = 1 \cdots, N\}$ separately, and performs an inner product $G_x(x_j) \cdot G_z(z_i)$ for each (i, j). The computational complexity of the two models is summarized in Table 1. The computational complexity of each hidden layer is larger than that of the inner products, provided that the hidden layer width d is bigger than the output dimension p of the MG model; this constraint is true for general neural network structures. The inner products are performed simultaneously by one matrix product, whereas hidden layer calculation is performed several times. Therefore, the total complexity of the SG is larger than that of the MG.

The computational memory can also be analyzed in the same manner: the SG saves all mN network outputs $G(x_j, z_i), i = 1, \cdots, N, j = 1, \cdots, m$, whereas the MG saves $m+N$ values $G_x(x_1), \cdots, G_x(x_m), G_z(z_1), \cdots, G_z(z_N)$ and their inner products. Each hidden layer contains mNd elements for the SG and $(m+N)d$ elements for the MG, and the inner product output of the MG consists of mN elements. Therefore, the SG requires much more memory than the MG.

B. MULTIPLE DISCRIMINATOR MODEL

Instead of using a loss function with a single discriminator as in (17), we propose the MD model that applies distinct discriminators D_k, D_f, and D_b to different stochastic processes k, f, and b, respectively. The MD model sums the discriminator values to obtain the loss function:

$$\frac{1}{N}\sum_{i=1}^{N}[D_k(k^{(i)}) - D_k(\tilde{k}^{(i)}) + D_f(f^{(i)}) - D_f(\tilde{f}^{(i)}) + D_b(b^{(i)}) - D_b(\tilde{b}^{(i)})]. \tag{20}$$

The structures of the SD and MD models are presented in Fig. 2.

The loss function (20) also induces a metric between (k, f, b) and $(\tilde{k}, \tilde{f}, \tilde{b})$, and if k, f, and b are independent, the metric is equivalent to the Wasserstein distance induced by the SD model loss (17).

Theorem 1: The loss function (17) corresponds to the Wasserstein distance $\mathfrak{L}_{SD} = \mathcal{W}((k, f, b), (\tilde{k}, \tilde{f}, \tilde{b}))$, *and the loss function (20) corresponds to* $\mathfrak{L}_{MD} = \mathcal{W}(k, \tilde{k}) + \mathcal{W}(f, \tilde{f}) + \mathcal{W}(b, \tilde{b})$. *If* k, f, *and* b *are independent, then the topologies induced by* $\mathfrak{L}_{SD}$ *and* $\mathfrak{L}_{MD}$ *are equivalent.*

Proof: See Appendix V. □

The SD model minimizes the Wasserstein distance $\mathcal{W}((k, f, b), (\tilde{k}, \tilde{f}, \tilde{b}))$ to train the joint distribution of (k, f, b), whereas the MD model minimizes the distance $\mathcal{W}(k, \tilde{k}) + \mathcal{W}(f, \tilde{f}) + \mathcal{W}(b, \tilde{b})$ to train each distribution separately. If k, f, and b are independent, the distance $\mathcal{W}(k, \tilde{k}) + \mathcal{W}(f, \tilde{f}) + \mathcal{W}(b, \tilde{b})$ removes the redundancy of learning the independence in $(\tilde{k}, \tilde{f}, \tilde{b})$ and effectively trains each process such that the resulting model becomes more accurate than the SD model. These assertions will be verified numerically in Section IV.

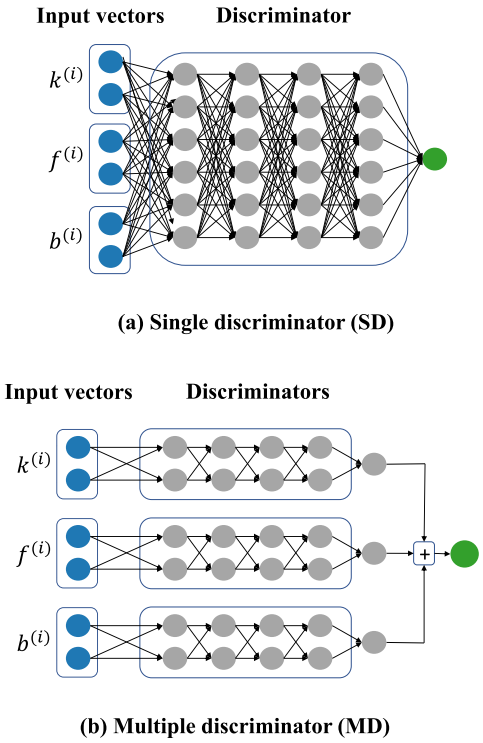


FIGURE 2. Comparison of the SD and the MD model.

C. ALGORITHM

Algorithm 1 MGDGAN to Solve the Forward Problem (10)

Require: Multiple discriminators with parameter ρ, multiple generators with parameter θ, learning rate l_D, l_G, training data k, f, and b.

1: Spectral normalizes the discriminators as in (9).
2: Generate the generator samples $\tilde{k}$ and $\tilde{u}$ by using (19) and compute $\tilde{f}$ and $\tilde{b}$ in (15) by automatic differentiations.
3: Compute the loss function $\mathfrak{L}$ with the multiple discriminators in (20).
4: Update the discriminators $\rho \leftarrow \text{RMSprop}(\rho - l_D \nabla \mathfrak{L})$
5: Spectral normalizes the discriminators.
6: Compute the loss function $\mathfrak{L}$ following 2-3.
7: Update generators, $\theta \leftarrow \text{RMSprop}(\theta - l_G \nabla \mathfrak{L})$.
8: Repeat **2-7** until convergence.

The overall procedure for solving SPDEs by MGDGAN is summarized in Algorithm 1. We repeat the discriminator update (steps 2-5) and the generator update (steps 6-7) until convergence, and physics-informed operation is implemented when computing $\tilde{f}$ and $\tilde{b}$ by automatic differentiation in step 2. We used the RMSprop optimizer to update parameters: the WGAN training becomes unstable when using a momentum

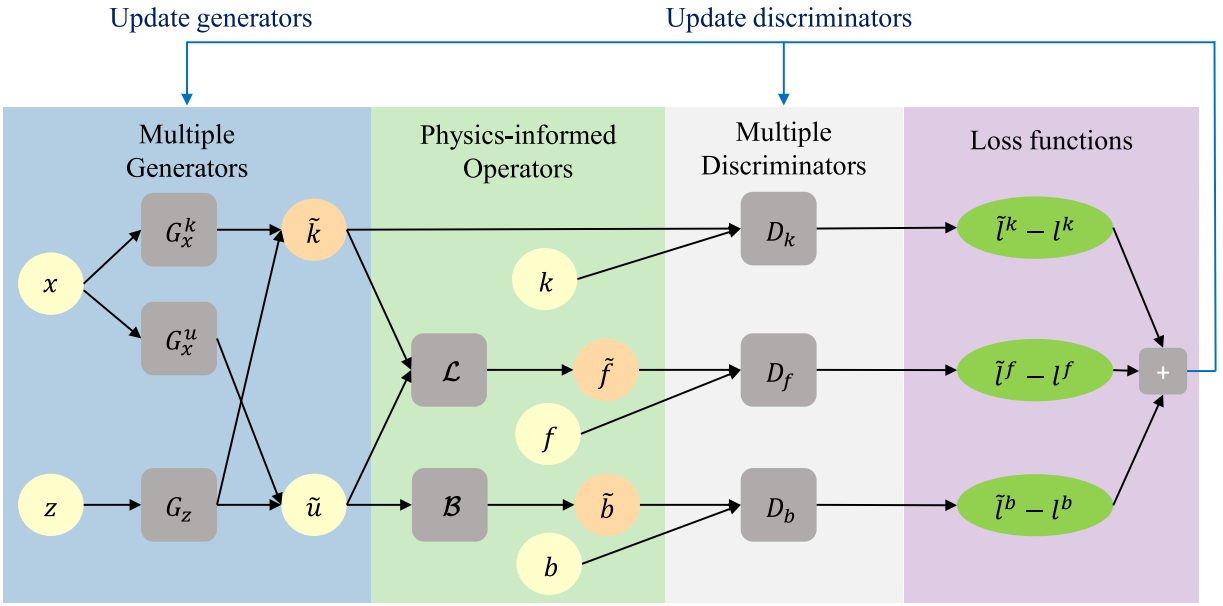


FIGURE 3. Schematic structure of Algorithm 1. The first part uses multiple generators to generate $\tilde{k}$ and $\tilde{u}$, the second part applies physics-informed operators to generate $\tilde{f}$ and $\tilde{b}$, and the last part uses the multiple discriminators to compute loss values.

based optimizer such as Adam, whereas the RMSprop optimizer performs well in nonstationary problems [40]. The schematic structure corresponding to the algorithm is drawn in Fig. 3; we use the same basis network for $\tilde{k}$ and $\tilde{u}$ to reflect that the stochastic processes k and u belong to the same probability space.

D. IMPROVING NN TRAINING

1) MASKING

To increase the generator quality, we apply a mask to the basis network to construct the 'stacked neural network' in the DeepOnet framework [44]. To this end, we perform elementwise multiplication of a block diagonal matrix M to the weight parameters $W \in \mathbb{R}^{m\times n}$

$$M = \begin{bmatrix} M_1 & 0 & 0 & \cdots & 0 \\ 0 & M_2 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \cdots & 0 \\ & & \vdots & & \\ 0 & 0 & 0 & \cdots & M_r \end{bmatrix}, \tag{21}$$

where all the elements of the block matrix M_i are 1. More precisely, consider $\tilde{a}(x) = (M \odot W)x$, where $\odot$ is elementwise multiplication and, x is an input of the affine layer, then $M \odot W$ represents the block diagonal elements of W, and the output becomes

$$\tilde{a}(x) = \begin{bmatrix} W_1 & 0 & 0 & \cdots & 0 \\ 0 & W_2 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \cdots & 0 \\ & & \vdots & & \\ 0 & 0 & 0 & \cdots & W_r \end{bmatrix} x, \tag{22}$$

where W_i is the same size as M_i. Let $W_i \in \mathbb{R}^{m_i\times n_i}$ and $x = [\vec{x}_1, \cdots, \vec{x}_r]^T$, where $\vec{x}_i \in \mathbb{R}^{n_i}$, then we have the stacked neural network outputs

$$\tilde{a}(x) = \begin{bmatrix} W_1\vec{x}_1 \\ \vdots \\ W_r\vec{x}_r \end{bmatrix}. \tag{23}$$

2) FOURIER EMBEDDINGS

In our context, realizations of the stochastic processes are high-frequency functions, and according to the frequency principle [45] and the neural tangent kernel theory [46], the high-frequency functions are difficult to train if the eigenvalues of the neural tangent kernel operator dramatically decrease; this phenomenon is known as spectral bias.

To solve this problem, we use Fourier embeddings to learn the Gaussian process with high-frequency samples; the method adds a non-trainable layer to the input of the NNs [47]:

$$[\sin(Bx), \cos(Bx)], \tag{24}$$

where B is a weight sampled from $\mathcal{N}(0, h_b I)$ with hyperparameter h_b, and the sin and cos functions are performed elementwisely. The non-trainable matrix B is used to relieve the effect of the spectral bias. The choice of hyperparameter h_b is crucial because it determines the overall frequency of the NNs. The effect of h_b will be demonstrated in the results section.

IV. NUMERICAL RESULTS

This section demonstrates examples of stochastic processes and elliptic SPDE problems. To verify the computational efficiency of the MG model over the SG model, we measure

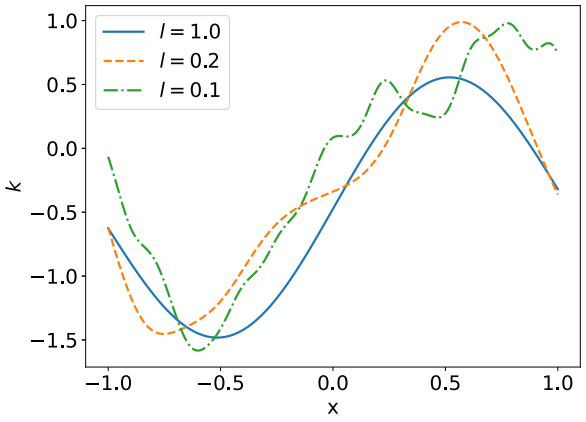


FIGURE 4. Reference sample paths for the Gaussian processes with various correlation lengths.

computation times T_{SG} for the SG model and T_{MG} for the MG model with various numbers m of sensors and N of snapshots. To measure accuracy, we compare the means and standard deviations of the model to those of the reference solution. Hyperparameters of the examples are summarized in Appendix V-B.

A. GAUSSIAN PROCESS

Consider a Gaussian process $k(x;\omega) \sim GP(\mu, C)$ with

$$\mu(x) = \sin(\pi x),$$
$$C(x,y) = \sigma^2 \exp\left(-\frac{|x-y|^2}{2l^2}\right), x, y \in [-1,1], \quad (25)$$

where $\sigma = 0.3$, and $l \in \{1, 0.2, 0.1\}$. The frequency of the sample paths increases as l decreases because the correlation among random samples $k(x;\omega)$ decreases as the value of l decreases (Fig. 4). We apply the Fourier embedding for the cases of $l = 0.2$ and $l = 0.1$ to mitigate the frequency principle in III-D2.

1) $l = 1.0$

We compare the SG with the MG for approximating the Gaussian process (25) with $l = 1.0$ in Fig. 5. As presented in Fig. 5(a), (b), the MG is at least ten times as fast as the SG, and the computational savings of the MG over the SG become more significant as m or N increases. T_{MG} is almost constant, whereas T_{SG} is linearly increasing. In Fig. 5(c), the accuracies for the MG and the SG are comparable. Both models successfully reconstructed the reference solution in Fig. 5(d), (e), and (f), which are the mean and standard deviation, covariance, and eigenvalues of the Gaussian process, respectively.

2) $l = 0.2$

We apply Fourier embeddings to the generators to deal with high-frequency components that occur due to low correlation length. For the SG model, the method is used for the spatial variable x: $[\sin(Bx), \cos(Bx)]$, $B \sim \mathcal{N}(0, I)$, whereas for the MG model, the method is applied to both the spatial and random variables x and z: $[\sin(B_x x), \cos(B_x x)]$, $[\sin(B_z z), \cos(B_z z)]$, $B_x, B_z \sim \mathcal{N}(0, I)$.

The results are presented in Fig. 6. In Fig. 6(a) and (b), the MG has a significantly smaller computational cost than the SG. As m or N increases, T_{MG} remains flat, whereas T_{SG} grows linearly with respect to m or N. Fig. 6(d-f) demonstrate that both models successfully reconstruct the reference solutions, with comparable errors in Fig. 6(c).

3) $l = 0.1$

For $l = 0.1$, the frequency of the sample paths is higher than for $l = 0.2$, and the choice of hyperparameter h_b in the Fourier embedding (24) is more important than the case of $l = 0.2$. We first compare the SG with the MG for $h_b = 3$, and then further explore the effect of the Fourier embedding hyperparameter $h_b \in \{1, 3, 10, 20, 30\}$ for the MG model.

The comparison of the MG with the SG for $h_b = 3$ is demonstrated in Fig. 7. In Fig. 7(a) and (b), T_{MG} is at least 30 times smaller than T_{SG}, and the difference becomes more significant as m or N increases. Both models have similar accuracy in Fig. 7(c) and reconstructed the reference solution well in Fig. 7(d-f). The errors of the MG for various Fourier embedding hyperparameters $h_b \in \{1, 3, 4, 20, 30\}$ are presented in Fig. 8. The accuracy increases as h_b increases but the model is degenerated for h_b greater than 20, i.e., when the frequency of the model exceeds the frequency of the target Gaussian process. Therefore, h_b must be chosen carefully.

B. STOCHASTIC ELLIPTIC EQUATION WITH TWO SPATIAL DIMENSION

We consider a two-dimensional stochastic elliptic equation. The PDE has various applications, such as modeling electrical potential in conductive materials and flow of a fluid in porous media in the exploitation of oil and gas [48]:

$$-\nabla \cdot [k(x,y;\omega)\nabla u(x,y;\omega)] = f(x,y;\omega), (x,y) \in \mathcal{D},$$
$$u(x,y) = 0, (x,y) \in \partial\mathcal{D}. \quad (26)$$

where $\mathcal{D} = (-1,1) \times (-1,1)$ and

$$k(x,y;\omega) = \exp(\hat{k}(x,y;\omega)) + \frac{1}{2},$$
$$\hat{k}(x;\omega) \sim GP(\mu, C),$$
$$\mu(x,y) = \sin(\pi x)\sin(\pi y),$$
$$C((x,y),(x',y')) = \sigma^2 \exp(-\frac{\|(x,y)-(x',y')\|_2^2}{2l^2}),$$
$$f(x,y) = 20\sin(\pi x)\sin(\pi y). \quad (27)$$

The diffusivity coefficient k is assumed to be a Gaussian process, with the exponential function applied to ensure the positiveness of k; the forcing term f is a deterministic function.

We split the second-order elliptic equation into two first-order SPDEs:

$$k(x,y;\omega)\nabla u(x,y;\omega) = -\tau(x,y;\omega), (x,y) \in \mathcal{D},$$
$$\nabla \cdot \tau(x,y;\omega) = f(x,y;\omega), (x,y) \in \partial\mathcal{D}. \quad (28)$$

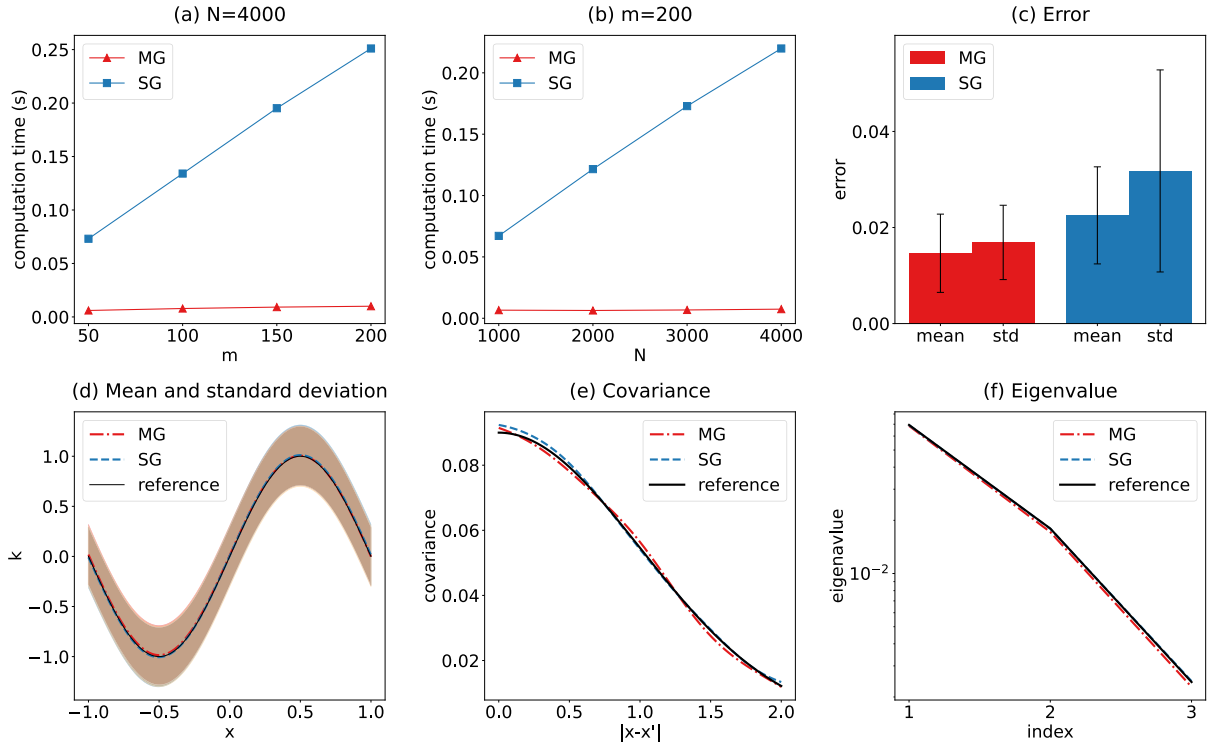


FIGURE 5. Learning Gaussian process with $l = 1.0$. Computational time T_{SG} and T_{MG} with (a) $N = 4000$, $m \in \{50, 100, 150, 200\}$ and (b) $m = 200$, $N \in \{1000, 2000, 3000, 4000\}$. (c) Errors of the mean and standard deviation. (d) Mean and one-standard-deviation band (e) Covariance (f) Eigenvalue of the covariance function.

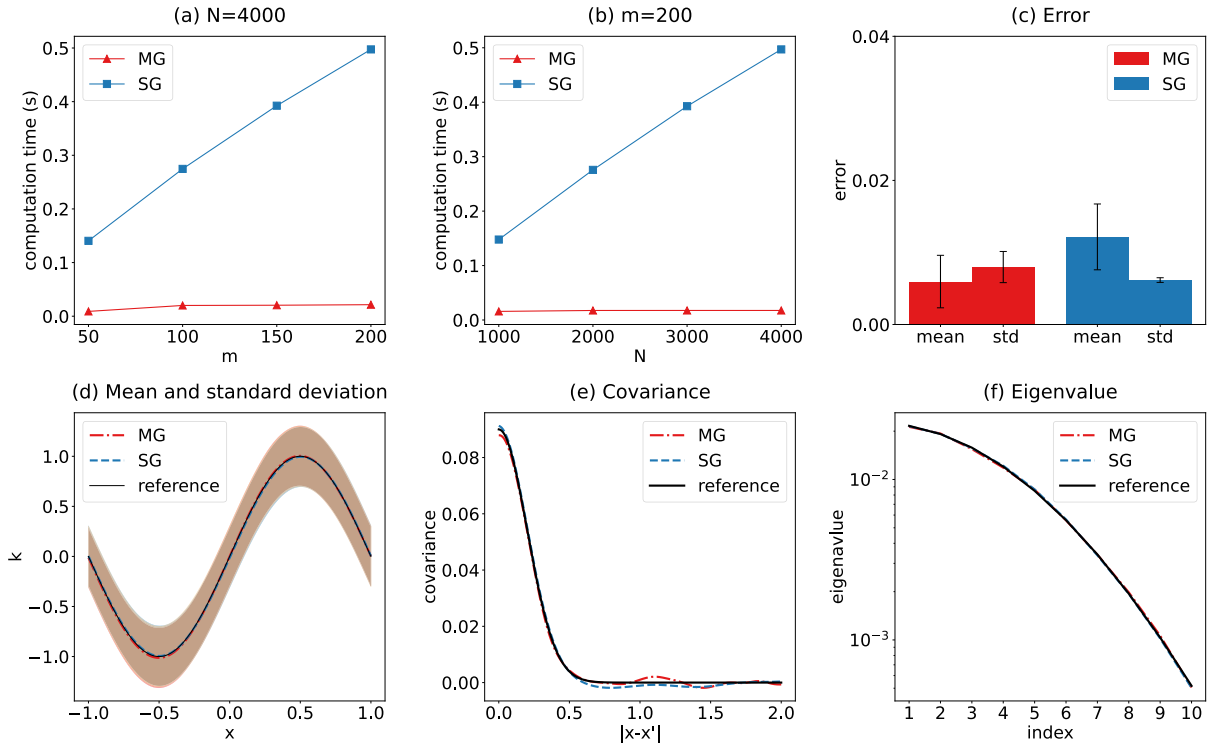


FIGURE 6. Learning Gaussian process with $l = 0.2$.

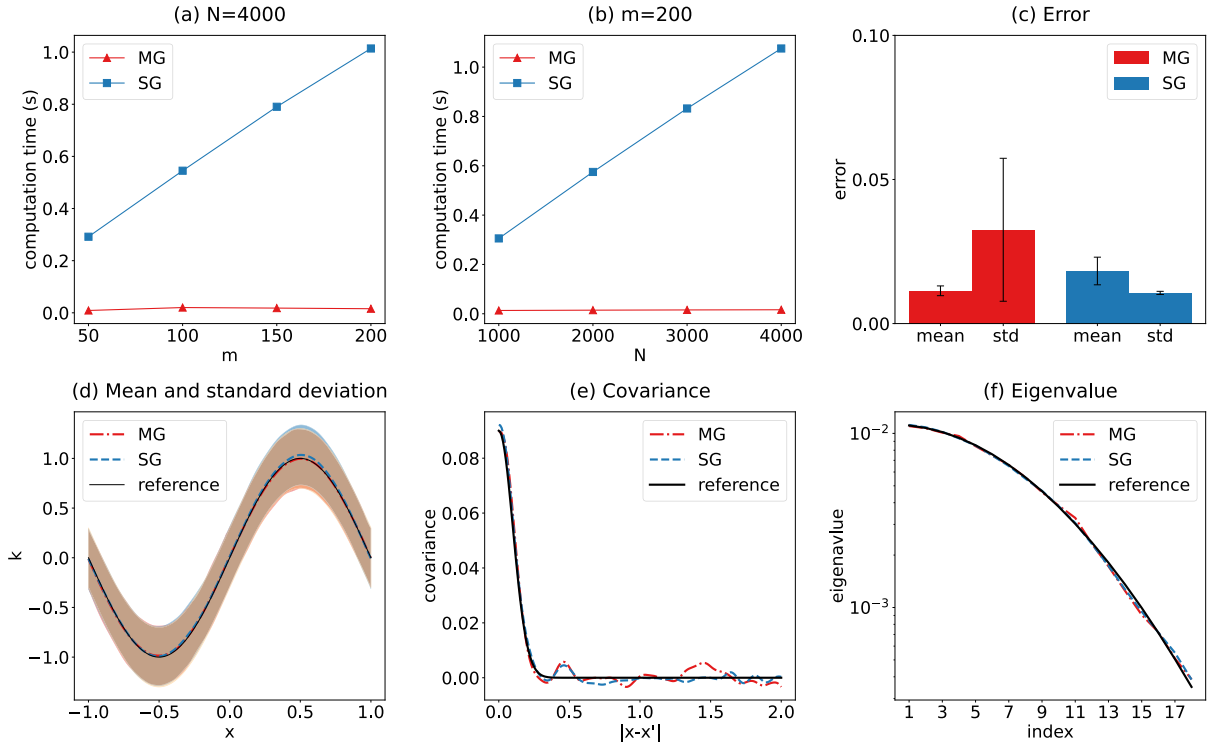


FIGURE 7. Learning Gaussian process with $l = 0.1$.

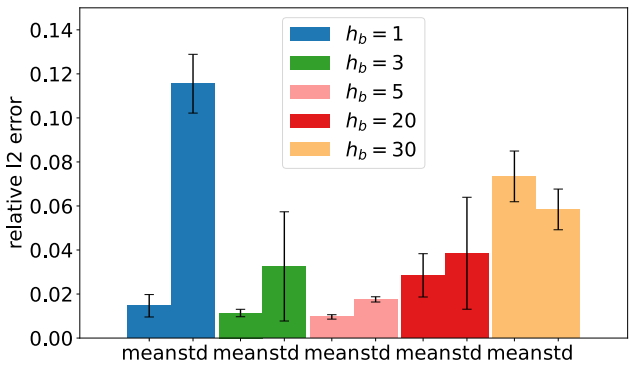


FIGURE 8. Errors of the MG model for different Fourier embedding hyperparameters $h_b \in \{1, 3, 5, 20, 30\}$ when learning Gaussian process l=0.1.

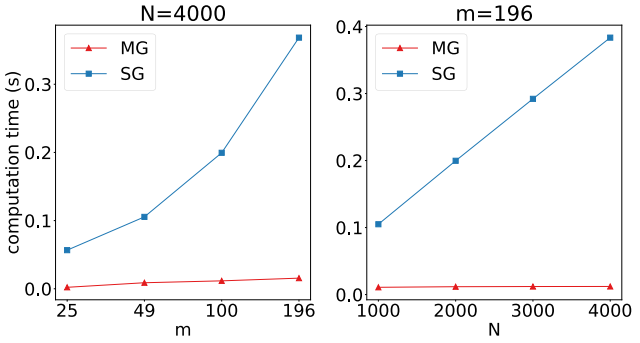


FIGURE 9. Computation times T_{SG} and T_{MG} for (left) $N = 4000$, $m \in \{25, 49, 100, 196\}$ and (right) $m = 196$, $N \in \{1000, 2000, 3000, 4000\}$.

An additional variable $\tau = (\tau_1, \tau_2) \in \mathbb{R}^2$ is introduced to reduce the order of derivatives computed in the loss function, and hence we have four generators for k, u, τ_1, and τ_2:

$$\begin{aligned} G^k(x, y, z) &= G_x^k(x, y) \cdot G_z(z), \\ G^u(x, y, z) &= (1 - x^2)(1 - y^2) G_x^u(x, y) \cdot G_z(z), \\ G^{\tau_1}(x, y, z) &= G_x^{\tau_1}(x, y) \cdot G_z(z), \\ G^{\tau_2}(x, y, z) &= G_x^{\tau_2}(x, y) \cdot G_z(z), \end{aligned} \tag{29}$$

where G_x^k, G_x^u, $G_x^{\tau_1}$, and $G_x^{\tau_2}$ are coefficient networks and G_z is a basis network (Fig. 3). All generators share the same basis network $G_z(z)$, and the solution generator G^u has a function $(1 - x^2)(1 - y^2)$ to enforce the boundary condition.

1) RESULTS

We compare SG-SD, MG-SD, and the proposed MGD models for the correlation length $l = 1.0$. The first two models use a single discriminator $D(k, \tau, f)$, and the third model uses multiple discriminators $D_k(k)$, $D_\tau(\tau)$, and $D_f(f)$.

Computation times T_{SG} and T_{MG} are plotted in Fig. 9. T_{MG} is at least ten times as fast as T_{SG}, and the difference increases when m or N increases. Note that T_{MG} is almost constant, whereas T_{SG} grows rapidly with respect to m or N. The mean of the solution and its relative l_2 error are shown in the top rows of Fig. 10 and Table 2, respectively, where all three models agree well with the reference solution. The standard deviation of the solution and its relative l_2 error are presented in the bottom rows of Fig. 10 and Table 2, respectively, where

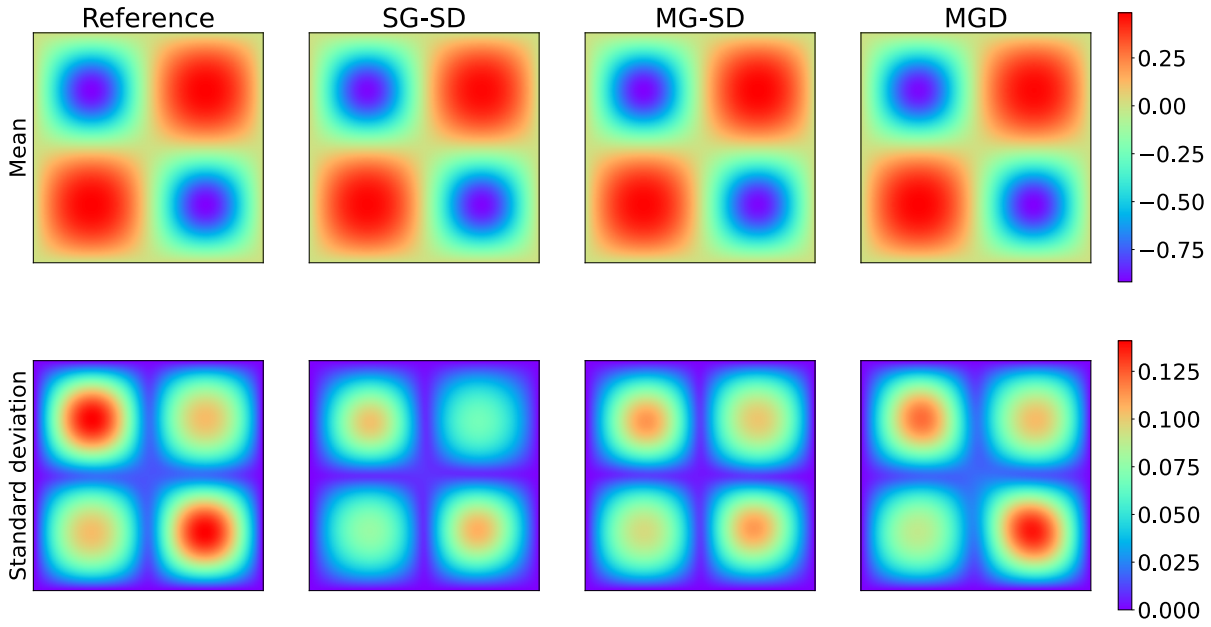


FIGURE 10. Mean (top) and standard deviation (bottom) of the solution for the elliptic SPDE. Columns from left to right: reference solution, SG-SD, MG-SD, and MGD (proposed).

TABLE 2. Errors of the means and standard deviations of the elliptic SPDE.

mean error	SG-SD	0.015
	MG-SD	**0.013**
	MGD	0.015
std error	SG-SD	0.312
	MG-SD	0.201
	MGD	**0.110**

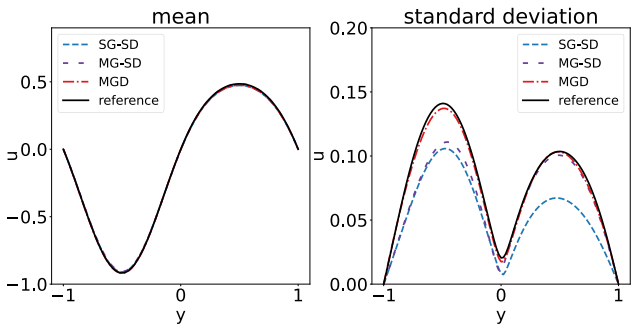


FIGURE 11. The mean and standard deviation plots at $x = 0.5$ for solutions to the elliptic SPDE problem.

the proposed MGD model reduces the error by one-third compared to the SG-SD model. In particular, we plot the mean and standard deviation of the solution at $x = 0.5$ in Fig. 11. It clearly shows that the proposed model MGD achieves the best standard deviation while the means of all three models agree well with that of the reference solution.

V. CONCLUSION

We have proposed multiple generators and discriminators in GANs to solve stochastic partial differential equations. We construct two neural networks as generators to encode the functions in the spatial and random variables separately, and then merge them together to compute the solution, which offers huge computational savings in the graining and prediction over the SG model for comparable accuracy. The model is also memory-efficient and has strong scalability. We show that the multiple discriminator model provides an equivalent metric to the single discriminator model. We test MGDGANs on stochastic processes and stochastic partial differential equations and show that MGDGANs can achieve small generalization errors by employing multiple discriminators and can reduce computing time by more than a factor of 10 by employing multiple generators.

More work is required to fully appreciate the proposed method, especially for the time-dependent stochastic problems, which we aim to carry out an extensive investigation in our future works. We can also employ other network architectures for the generators and discriminators, such as convolutional neural networks, which may improve further the accuracy of the proposed method.

The limitations of the proposed method are as follows: first, we do not understand the training dynamics of GANs theoretically yet. The training process involves simultaneous learning of several objective functions taking into account the governing equation and the boundary conditions. The problem of multi-object optimization may bring pathological problems such as a discrepancy in the convergence speed of the different objectives. There have been few results on the training dynamics for PINNs based on neural tangent kernel but the training dynamics of GANs is yet to be developed. Second, it is difficult to find the optimal output dimension of the basis and coefficient network. A large dimension may increase computation cost whereas a small dimension may

reduce the accuracy of the training results. Third, there have not been any results on the convergence theory of GANs. We believe that addressing these challenges will improve our understanding of physics-informed GANs for uncertainty quantification and help to develop state-of-the-art models in computational science and engineering.

APPENDIX A EQUIVALENCE OF THE SD LOSS AND THE MD LOSS

A. NOTATIONS

Given probability measures p and q, we define the following set related to the Wasserstein distances

$$\Gamma(p,q) = \{\gamma(x,y) : \int \gamma(x,y)dy = p, \int \gamma(x,y)dx = q\}. \tag{30}$$

Let X_1, X_2, Y_1, and Y_2 be random vectors that follow the distributions p_1, p_2, q_1, and q_2, respectively, and $\eta(p_1, q_1)$ and $\eta(p_2, q_2)$ be the joint distributions of (p_1, q_1) and (p_2, q_2), respectively. We define

$$\begin{aligned}\Gamma_0 &= \Gamma(\eta(p_1,q_1), \eta(p_2,q_2)),\\ \Gamma_X &= \Gamma(p_1,p_2),\\ \Gamma_Y &= \Gamma(q_1,q_2),\end{aligned} \tag{31}$$

then the Wasserstein distances are

$$\begin{aligned}\mathcal{L}_0 &= \inf_{\gamma_0\in\Gamma_0} \mathbb{E}_{\gamma_0}[\|(X_1,Y_1)-(X_2,Y_2)\|],\\ \mathcal{L}_x &= \inf_{\gamma_x\in\Gamma_X} \mathbb{E}_{\gamma_x}[\|X_1-X_2\|],\\ \mathcal{L}_y &= \inf_{\gamma_y\in\Gamma_Y} \mathbb{E}_{\gamma_y}[\|Y_1-Y_2\|].\end{aligned} \tag{32}$$

Remark 1: WGAN loss $D(x_1,y_1) - D(x_2,y_2)$ minimizes $\mathcal{L}_0$ and WGAN loss with MD $D(x_1) - D(x_2) + D(y_1) - D(y_2)$ minimizes $\mathcal{L}_x + \mathcal{L}_y$.

B. MAIN THEOREM

Theorem 2: The function $\mathcal{L}_x + \mathcal{L}_y$ from the (32) is indeed a metric. Under the assumption that X_1 is independent of Y_1, and X_2 is independent of Y_2, the topology induced by $\mathcal{L}_x + \mathcal{L}_y$ is equivalent to that of the Wasserstein distance $\mathcal{L}_0$. More precisely, the following inequalities hold:

$$\mathcal{L}_x + \mathcal{L}_y \geq \mathcal{L}_0 \geq \frac{1}{2}(\mathcal{L}_x + \mathcal{L}_y). \tag{33}$$

Proof: Since Wasserstein distance is a metric, $\mathcal{L}_x$ and $\mathcal{L}_y$ are metrics and a sum of two metrics is also a metric.

- $\mathcal{L}_0 \geq \frac{1}{2}(\mathcal{L}_x + \mathcal{L}_y)$

Let $\gamma_0 \in \Gamma_0$, define

$$\begin{aligned}\gamma_x(x_1,x_2) &= \int \gamma_0(x_1,y_1,x_2,y_2)dy_1dy_2,\\ \gamma_y(y_1,y_2) &= \int \gamma_0(x_1,y_1,x_2,y_2)dx_1dx_2.\end{aligned} \tag{34}$$

By the definition of Γ_0, we have

$$\int \gamma_x dx_1 = \int \gamma_0 dx_1dy_1dy_2 = \int \eta(p_2,q_2)dy_2 = p_2,$$

TABLE 3. Sample size.

case	train sensor	train snapshot	test sensor	test snapshot
GP($l = 1.0$)	41	10000	201	100000
GP($l = 0.2$)	41	20000	201	100000
GP($l = 0.1$)	41	20000	201	100000
SPDE	11×11	8000	101×101	100000

TABLE 4. Generator size, FE represents Fourier embedding.

case	architecture	depth	width	p	d_z	FE size
GP($l = 0.1$)	SG	2	80	-	3	-
	MG	2	32	64	3	-
GP($l = 0.2$)	SG	3	64	-	10	64
	MG	2	40	64	10	64
GP($l = 0.1$)	SG	3	128	-	18	128
	MG	2	75	128	18	128
SPDE	SG	2	128	-	3	-
	MG	2	64	64	3	-

$$\int \gamma_x dx_2 = \int \gamma_0 dx_2dy_2dy_1 = \int \eta(p_1,q_1)dy_1 = p_1, \tag{35}$$

and $\gamma_x \in \Gamma_X$. Similarly $\gamma_y \in \Gamma_Y$. Now

$$\begin{aligned}&\mathbb{E}_{\gamma_0}\|(X_1,Y_1)-(X_2,Y_2)\|\\ &= \int \|(x_1,y_1)-(x_2,y_2)\|\gamma_0 dx_1dy_1dx_2dy_2\\ &\geq \int \frac{1}{2}(\|x_1-x_2\|+\|y_1-y_2\|)\gamma_0 dx_1dy_1dx_2dy_2\\ &= \frac{1}{2}\Big(\int \|x_1-x_2\|\gamma_0 dy_1dy_2dx_1dx_2\\ &\quad + \int \|y_1-y_2\|\gamma_0 dx_1dx_2dy_1dy_2\Big)\\ &= \frac{1}{2}\Big(\int \|x_1-x_2\|\gamma_x dx_1dx_2 + \int \|y_1-y_2\|\gamma_y dy_1dy_2\Big)\\ &= \frac{1}{2}(\mathbb{E}_{\gamma_x}\|X_1-X_2\| + \mathbb{E}_{\gamma_y}\|Y_1-Y_2\|)\\ &\geq \frac{1}{2}(\mathcal{L}_x+\mathcal{L}_y).\end{aligned} \tag{36}$$

Since γ_0 is arbitrary, we have $\mathcal{L}_0 \geq (\mathcal{L}_x + \mathcal{L}_y)/2$.

Remark 2: The result can be extended to the pairs of n random vectors $(Z_1^1, \cdots, Z_1^n)$ and $(Z_2^1, \cdots, Z_2^n)$, with the inequality constant $1/n$ instead of $1/2$. Note that this part does not require the independence of Z_i^1 and Z_i^2.

- $\mathcal{L}_x + \mathcal{L}_y \geq \mathcal{L}_0$

Let $\gamma_x \in \Gamma_X$, $\gamma_y \in \Gamma_Y$, then from the independence condition of X_i and Y_i, we get

$$\begin{aligned}\int \gamma_x(x_1,x_2)\gamma_y(y_1,y_2)dx_2dy_2 &= \int \gamma_x dx_2 \int \gamma_y dy_2\\ &= p_1q_1\\ &= \eta(p_1,q_1),\\ \int \gamma_x(x_1,x_2)\gamma_y(y_1,y_2)dx_1dy_1 &= \int \gamma_x dx_1 \int \gamma_y dy_1\\ &= p_2q_2\\ &= \eta(p_2,q_2).\end{aligned} \tag{37}$$

TABLE 5. Discriminator size.

case	architecture	depth	width
GP($l = 0.1$)	-	4	64
GP($l = 0.2$)	-	4	64
GP($l = 0.1$)	-	4	64
SPDE	SD	4	128
	MD	4	64

Therefore, $\gamma_0 := \gamma_x \gamma_y \in \Gamma_0$. Now

$$\begin{aligned}
&\mathbb{E}_{\gamma_x}[\|X_1 - X_2\|] + \mathbb{E}_{\gamma_y}[\|Y_1 - Y_2\|] \\
&= \int \|x_1 - x_2\| \gamma_x dx_1 dx_2 + \int \|y_1 - y_2\| \gamma_y dy_1 dy_2 \\
&= \int \|x_1 - x_2\| \gamma_x \gamma_y dx_1 dx_2 dy_1 dy_2 \\
&\quad + \int \|y_1 - y_2\| \gamma_x \gamma_y dx_1 dx_2 dy_1 dy_2 \\
&= \int (\|x_1 - x_2\| + \|y_1 - y_2\|) \gamma_0 dx_1 dx_2 dy_1 dy_2 \\
&\geq \int \|(x_1, y_1) - (x_2, y_2)\| \gamma_0 dx_1 dx_2 dy_1 dy_2 \\
&= \mathbb{E}_{\gamma_0} \|(X_1, Y_1) - (X_2, Y_2)\| \\
&\geq \mathcal{L}_0. \qquad (38)
\end{aligned}$$

Since γ_x and γ_y are arbitrary, $\mathcal{L}_x + \mathcal{L}_y \geq \mathcal{L}_0$. □

APPENDIX B HYPERPARAMETER SETTINGS

We used the RMSprop optimizer with $(\alpha, \beta) = (2 \cdot 10^{-4}, 0.99)$ for the discriminators and $(\alpha, \beta) = (5 \cdot 10^{-5}, 0.99)$ for the generators, following the two time-scale update rule (TTUR) [49]. For examples considering the Gaussian process, tanh activation is used for the generators, and rectified linear unit (ReLU) activation is used for the discriminators: tanh is one of the most common activation functions in the PINN framework because we expect a smooth approximation for the solutions. Smoothness is not necessary for discriminators; thus, we use the ReLU because of its fast evaluation speed. For examples considering the SPDE, we used the sine activation function for the generators due to its well-behaved derivative representations [50] and ReLU activation for the discriminators.

Hyperparameters of samples are listed in Table 3. To check the model's generalizability, the resolutions of the test samples are higher than those of the training samples.

Hyperparameters of generators are listed in Table 4. We choose layer width and depth such that the SG and the MG models have comparable numbers of parameters and noise dimension d_z to capture 99% of the energy of the reference covariance function. The coefficient and the basis networks have the same depth and width.

Hyperparameters of discriminators are listed in Table 5. Each GP example only contains one stochastic process and does not require the MD structure. In the MD, all discriminators have the same layer depth and width.

REFERENCES

[1] D. Crommelin and B. Khouider, "Stochastic and statistical methods in climate, atmosphere, and ocean science," in *Proc. BAMS*, 2015, pp. 1377–1386.
[2] H. Mena and L. Pfurtscheller, "An efficient SPDE approach for el Niño," *Appl. Math. Comput.*, vol. 352, pp. 146–156, Jul. 2019.
[3] M. Reagan, H. Najm, B. Debusschere, O. L. Maître, O. Knio, and R. Ghanem, "Spectral stochastic uncertainty quantification in chemical systems," *Combust. Theor. Moel.*, vol. 8, no. 3, p. 607, 2004.
[4] A. M. Stuart, "Inverse problems: A Bayesian perspective," *Acta Numer.*, vol. 19, pp. 451–559, May 2010.
[5] E. A. Affum, S. A. Ajagbe, K. A. Boateng, M. O. Adigun, and E. Addo, "Response analysis of varied Q-power values of cosine distribution in spatial correlation," in *Proc. IEEE Int. Symp. Antennas Propag. USNC-URSI Radio Sci. Meeting (AP-S/URSI)*, Jul. 2022, pp. 2070–2071.
[6] E. A. Affum, M. O. Adigun, K. A. Boateng, S. A. Ajagbe, and E. Addo, "Enhancing UAV communication performance: Analysis using interference based geometry stochastic model and successive interference cancellation," in *Proc. ICCSA*. Cham, Switzerland: Springer, 2022, pp. 232–245.
[7] H.-J. Bungartz and M. Griebel, "Sparse grids," *Acta Numer.*, vol. 13, pp. 147–269, May 2004.
[8] H. Rabitz, Ö. F. Aliş, J. Shorter, and K. Shim, "Efficient input—Output model representations," *Comput. Phys. Commun.*, vol. 117, nos. 1–2, pp. 11–20, Mar. 1999.
[9] J. Han, E. JentzenWeinan, and A. Jentzen, "Solving high-dimensional partial differential equations using deep learning," *Proc. Nat. Acad. Sci. USA*, vol. 115, no. 34, pp. 8505–8510, 2018.
[10] S. A. Ajagbe, O. A. Oki, M. A. Oladipupo, and A. Nwanakwaugwum, "Investigating the efficiency of deep learning models in bioinspired object detection," in *Proc. ICECET*, 2022, pp. 1–6.
[11] Y. H. Bhosale and K. S. Patnaik, "Application of deep learning techniques in diagnosis of Covid-19 (coronavirus): A systematic review," *Neural Process. Lett.*, vol. 54, pp. 1–53, Sep. 2022.
[12] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 8599–8603.
[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[15] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web (WWW)*, 2017, pp. 173–182.
[16] M. Mustafa, D. Bard, W. Bhimji, Z. Lukić, R. Al-Rfou, and J. M. Kratochvil, "CosmoGAN: Creating high-fidelity weak lensing convergence maps using generative adversarial networks," *Comput. Astrophys. Cosmol.*, vol. 6, no. 1, pp. 1–13, Dec. 2019.
[17] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A platform for embodied AI research," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9338–9346.
[18] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *Proc. ICIPMI*. Cham, Switzerland: Springer, 2017, pp. 146–157.
[19] J. Berg and K. Nyström, "A unified deep artificial neural network approach to partial differential equations in complex geometries," *Neurocomputing*, vol. 317, pp. 28–41, Nov. 2018.
[20] B. Chudomelka, Y. Hong, H. Kim, and J. Park, "Deep neural network for solving differential equations motivated by Legendre-Galerkin approximation," 2020, *arXiv:2010.12975*.
[21] N. Geneva and N. Zabaras, "Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks," *J. Comput. Phys.*, vol. 403, Feb. 2020, Art. no. 109056.
[22] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 987–1000, Sep. 1998.
[23] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," 2020, *arXiv:2010.08895*.

[24] L. Lu, P. Jin, and G. E. Karniadakis, "DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators," 2019, *arXiv:1910.03193*.
[25] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019.
[26] J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," *J. Comput. Phys.*, vol. 375, pp. 1339–1364, Dec. 2018.
[27] E. Weinan and B. Yu, "The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems," *Commun. Math. Statist.*, vol. 6, no. 1, pp. 1–12, 2018.
[28] J. Zhuang, D. Kochkov, Y. Bar-Sinai, M. P. Brenner, and S. Hoyer, "Learned discretizations for passive scalar advection in a two-dimensional turbulent flow," *Phys. Rev. Fluids*, vol. 6, no. 6, Jun. 2021, Art. no. 064605.
[29] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner, "Learning data-driven discretizations for partial differential equations," *Proc. Nat. Acad. Sci. USA*, vol. 116, no. 31, pp. 15344–15349, 2019.
[30] M. Amin Nabian and H. Meidani, "A deep neural network surrogate for high-dimensional random partial differential equations," 2018, *arXiv:1806.02957*.
[31] R. K. Tripathy and I. Bilionis, "Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification," *J. Comput. Phys.*, vol. 375, pp. 565–588, Dec. 2018.
[32] L. Guo, H. Wu, and T. Zhou, "Normalizing field flows: Solving forward and inverse stochastic differential equations using physics-informed flow models," *J. Comput. Phys.*, vol. 461, Jul. 2022, Art. no. 111202.
[33] J. Jung and M. Choi, "Bayesian deep learning framework for uncertainty quantification in high dimensions," 2022, *arXiv:2210.11737*.
[34] L. Yang, C. Daskalakis, and G. E. Karniadakis, "Generative ensemble regression: Learning particle dynamics from observations of ensembles with physics-informed deep generative models," *SIAM J. Sci. Comput.*, vol. 44, no. 1, pp. B80–B99, Feb. 2022.
[35] L. Yang, X. Meng, and G. E. Karniadakis, "B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data," *J. Comput. Phys.*, vol. 425, Jan. 2021, Art. no. 109913.
[36] L. Yang, D. Zhang, and G. E. Karniadakis, "Physics-informed generative adversarial networks for stochastic differential equations," *SIAM J. Sci. Comput.*, vol. 42, no. 1, pp. A292–A317, Jan. 2020.
[37] D. Zhang, L. Guo, and G. E. Karniadakis, "Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks," *SIAM J. Sci. Comput.*, vol. 42, no. 2, pp. A639–A665, Jan. 2020.
[38] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data," *J. Comput. Phys.*, vol. 394, pp. 56–81, Oct. 2019.
[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 139–144.
[40] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. ICML*, 2017, pp. 214–223.
[41] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," 2018, *arXiv:1802.05957*.
[42] D. Xiu, "Generalized polynomial chaos," in *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton, NJ, USA: Princeton Univ. Press, 2010, doi: 10.1515/9781400835348.
[43] X. Meng, L. Yang, Z. Mao, J. del Águila Ferrandis, and G. E. Karniadakis, "Learning functional priors and posteriors from data and physics," *J. Comput. Phys.*, vol. 457, May 2022, Art. no. 111073.
[44] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators," *Nature Mach. Intell.*, vol. 3, no. 3, pp. 218–229, Mar. 2021.
[45] Z.-Q. John Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma, "Frequency principle: Fourier analysis sheds light on deep neural networks," 2019, *arXiv:1901.06523*.
[46] S. Wang, X. Yu, and P. Perdikaris, "When and why PINNs fail to train: A neural tangent kernel perspective," *J. Comput. Phys.*, vol. 449, Jan. 2022, Art. no. 110768.
[47] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 7537–7547.
[48] M. K. Hubbert, "Darcy's law and the field equations of the flow of underground fluids," *Trans. AIME*, vol. 207, no. 1, pp. 222–239, Dec. 1956.
[49] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Nov. 2017, pp. 1–12.
[50] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 7462–7473.

SUNGHA CHO received the B.S. degree in mathematics from the Pohang University of Science and Technology, Pohang, South Korea, in 2015, where he is currently pursuing the Ph.D. degree in mathematics. His research interests include uncertainty quantification of the nonlinear systems and deep learning.

MINSEOK CHOI received the B.S. degree in mechanical engineering and mathematics and the M.S. degree in mechanical engineering from Seoul National University, Seoul, South Korea, in 2002 and 2007, respectively, and the Ph.D. degree in applied mathematics from Brown University, Providence, USA, in 2014. He was a Postdoctoral Researcher with Princeton University, USA, until 2017. He is currently an Assistant Professor in mathematics at the Pohang University of Science and Technology (POSTECH), South Korea. His research interests include physics-informed machine learning, uncertainty quantification, and related areas of applied mathematics.