Computer Methods in Applied Mechanics and Engineering 429 (2024) 117147

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

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

Bayesian reduced-order deep learning surrogate model for dynamic systems described by partial differential equations

Yuanzhe Wang [a], Yifei Zong [a], James L. McCreight [b,c], Joseph D. Hughes [b], Alexandre M. Tartakovsky [a,d,*]

[a] Department of Civil and Environmental Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801, United States of America
[b] Cooperative Programs for the Advancement of Earth System Science, University Corporation for Atmospheric Research, Boulder, CO 80307, United States of America
[c] Integrated Modeling and Prediction Division, U.S. Geological Survey, Chicago, IL 60613, United States of America
[d] Pacific Northwest National Laboratory, Richland, WA 99352, United States of America

ARTICLE INFO

Keywords:
Machine learning
Surrogate models
Dimensional reduction
Bayesian uncertainty quantification

ABSTRACT

We propose a reduced-order deep-learning surrogate model for dynamic systems described by time-dependent partial differential equations. This method employs space–time Karhunen–Loève expansions (KLEs) of the state variables and space-dependent KLEs of space-varying parameters to identify the reduced (latent) dimensions. Subsequently, a deep neural network (DNN) is used to map the parameter latent space to the state variable latent space.

An approximate Bayesian method is developed for uncertainty quantification (UQ) in the proposed KL-DNN surrogate model. The KL-DNN method is tested for the linear advection–diffusion and nonlinear diffusion equations, and the Bayesian approach for UQ is compared with the deep ensembling (DE) approach, commonly used for quantifying uncertainty in DNN models. It was found that the approximate Bayesian method provides a more informative distribution of the PDE solutions in terms of the coverage of the reference PDE solutions (the percentage of nodes where the reference solution is within the confidence interval predicted by the UQ methods) and log predictive probability. The DE method is found to underestimate uncertainty and introduce bias.

For the nonlinear diffusion equation, we compare the KL-DNN method with the Fourier Neural Operator (FNO) method and find that KL-DNN is 10% more accurate and needs less training time than the FNO method.

1. Introduction

We propose a novel method for constructing Bayesian surrogate models for time-dependent partial differential equation (PDE) problems. Several machine learning (ML) methods were recently developed for constructing surrogate models, including neural operators and finite space operators. The neural operator models include Fourier neural operator (FNO) [1], deep operator networks (DeepONets) [2], graph neural operator (GNO) [3], and the principal component analysis deep neural network (PCA-Net) method [4]. These operators learn the relationships between input functions (e.g., parameter fields) and output functions (e.g., state variables), i.e., these models can predict the PDE solution at any point. The finite-space operators (e.g., convolutional neural network

* Corresponding author at: Department of Civil and Environmental Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801, United States of America.
E-mail address: amt1998@illinois.edu (A.M. Tartakovsky).

https://doi.org/10.1016/j.cma.2024.117147
Received 16 April 2024; Received in revised form 6 June 2024; Accepted 7 June 2024
Available online 19 June 2024

image-to-image regression models [5]) learn the relationship between input parameters and PDE solutions discretized on a mesh and make predictions on the same mesh. For complex problems, the (parameter field) input space can be very large, which might lead to ML models with a relatively large number of parameters. Training such models requires large training datasets and significant computational resources. Also, estimating uncertainty in such ML models becomes a formidable challenge because of the curse of dimensionality (CoD) in many Bayesian methods, including Markov Chain Monte Carlo (MCMC) methods [6–8].

One possible solution to address CoD is to use dimension-reduction methods for finding the reduced latent space of parameters and state variables. For example, in the PCA-Net method, the principal component analysis (PCA) was used to find the latent spaces of the space-dependent diffusion coefficient and state variable [4]. Then, a deep neural network (DNN) was employed to map the latent space of the parameter field to the latent space of the PDE state variable.

There are multiple sources of uncertainty in ML surrogate models related to the non-convexity of the minimization problems in the training of DNNs and training and approximation errors. The common methods for quantifying uncertainty are deep ensembling (DE) and Bayesian methods [9]. The DE methods only account for uncertainty due to the initialization of the iterative solution of the minimization problem involved in the DNN training and tend to underestimate the total uncertainty in the surrogate models. The methods for direct sampling of posterior distributions such as MCMC suffer from the CoD, which limits their application to uncertainty quantification (UQ) in surrogate models. Several approximate Bayesian methods were recently proposed to address challenges in MCMC methods, including transport maps [10] and ABC methods [11].

The novelty of our work is two-fold. Firstly, we propose a reduced-order surrogate model for time-dependent PDEs, where the dimension reduction is performed in both space and time. This is achieved by the space–time Karhunen–Loéve expansion (KLE) [12,13] of the state variable. The space–time KLE is an extension of the KLEs, which are used to model spatial or temporal fields [14] and are closely related to the function PCA expansion in the PCA-Net method. We name this approach the KL-DNN method. A potential bottleneck in applying the KLE-DNN and PCA-Net methods for multi-dimensional time-dependent PDEs is the large size of the PCA decomposition, singular value decomposition, or the eigenvalue problems that are used in these methods to compute the basis functions in PCA or KLE representations. However, the KLE of a multidimensional function can be represented as a combination of one-dimensional KLEs [15], which significantly reduces this challenge.

The second contribution of this paper is a novel method for quantifying uncertainty in surrogate models. In this approach, we randomize the objective function in the KL-DNN method such that the samples obtained by solving the resulting minimization problem for the different realizations of the random noise terms approximate the posterior distribution of the parameters in the KL-DNN model. This method builds on randomization methods developed in the context of Bayesian inverse solutions, including randomized MAP [16], maximum likelihood [17], the randomize-then-optimize method [18], and the randomized physics-informed conditional KLE (rPICKLE) method [19].

The stability of solutions obtained with some surrogate ML models can become an issue, especially when the surrogate models are combined with PDE numerical models (e.g., when surrogate models are used as closure models for viscosity in Navier–Stokes numerical models [20]). We demonstrate that the KL-DNN method is stable given that the training dataset is sufficiently large.

This work is organized as follows. In Section 2, we present the KL-DNN method for general time-dependent PDEs. Section 3 describes the approximate Bayesian and DE methods for quantifying uncertainty in the KL-DNN surrogate model. Sections 4 and 5 present applications of the KL-DNN method for solving advection–diffusion and nonlinear diffusion equations, respectively. In Section 5, we also provide a comparison with the FNO method. Conclusions are given in Section 6.

2. Dynamic KL-DNN method

Consider a PDE problem

$$\mathcal{L}(u(\boldsymbol{x},t);\boldsymbol{p},y(\boldsymbol{x}))=0,\quad (\boldsymbol{x},t)\in(D,T) \tag{1}$$

subject to appropriate initial and boundary conditions. Here, $\mathcal{L}$ is the differential operator, $u(\boldsymbol{x},t)$ is the state variable, $y(\boldsymbol{x})$ is the space-dependent parameter field, $\boldsymbol{p}=[p_1,\dots,p_n]^{\mathrm{T}}$ is the vector of n scalar parameters, and D and T are the space and time domains, respectively.

Our objective is to develop a method for a rapid estimation of $u(\boldsymbol{x},t)$ for any $y(\boldsymbol{x})$ and $\boldsymbol{p}$. Recently, a dynamic physics-informed conditional Karhunen–Loéve expansion (dPICKLE) model for the PDE was proposed [13]. In this approach, a space–time KL expansion is used to represent $u(\boldsymbol{x},t)$:

$$u(\boldsymbol{x},t)\approx\hat{u}(\boldsymbol{x},t,\boldsymbol{\eta})=\bar{u}(\boldsymbol{x},t)+\sum_{i=1}^{N_\eta}\phi_i(\boldsymbol{x},t)\sqrt{\lambda_i}\eta_i, \tag{2}$$

where $\hat{u}$ is the KLE of u, $\boldsymbol{\eta}=(\eta_1,\dots,\eta_{N_\eta})^T$ is the vector of unknown parameters, and $\bar{u}(\boldsymbol{x},t)$, $\phi_i(\boldsymbol{x},t)$, and λ_i are estimated as properties of a random process that can provide an accurate statistical representation of $u(\boldsymbol{x},t)$. Specifically, $\bar{u}(\boldsymbol{x},t)$ is the mean and $\phi_i(\boldsymbol{x},t)$ and λ_i are the eigenfunctions and eigenvalues of the covariance $C_u(\boldsymbol{x},\boldsymbol{x}',t,t')$ of this random process, respectively. We compute the mean and covariance by sampling $y(\boldsymbol{x})$ and $\boldsymbol{p}$ from their prior distributions, solving the PDE (2) for each sample of $y(\boldsymbol{x})$ and $\boldsymbol{p}$, and computing $\bar{u}(\boldsymbol{x},t)$ and $C_u(\boldsymbol{x},\boldsymbol{x}',t,t')$ as the sample mean and covariance, respectively [13].

The eigenvalues λ_i are organized in descending order and truncated according to the desired tolerance, rtol:

$$\frac{\sum_{i=N_\eta+1}^{\infty}\lambda_i}{\sum_{i=1}^{\infty}\lambda_i}\le\mathrm{rtol}. \tag{3}$$

The space-dependent parameters are represented with the standard (space-dependent) KL expansion:

$$y(\boldsymbol{x}) \approx \hat{y}(\boldsymbol{x}, \boldsymbol{\xi}) = \bar{y}(\boldsymbol{x}) + \sum_{i=1}^{N_\xi} \chi_i(\boldsymbol{x}) \sqrt{\beta_i} \xi_i, \tag{4}$$

where $\hat{y}$ is the KLE of y, $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_{N_\xi})^T$ is the vector of parameters, $\bar{y}(\boldsymbol{x})$ is the prior mean, and $\chi_i(\boldsymbol{x})$ and β_i are the eigenfunctions and eigenvalues of $C_y(\boldsymbol{x}, \boldsymbol{x}')$, the prior covariance of $y(\boldsymbol{x})$.

Assuming that the Gaussian process model of $y(\boldsymbol{x})$ (including $\bar{y}(\boldsymbol{x})$ and $C_y(\boldsymbol{x}, \boldsymbol{x}')$) is known, the dPICKLE method computes the solution of Eq. (1) for any realization of $y(\boldsymbol{x})$ and $\boldsymbol{p}$ according to the following algorithm:

1. Generate N_{train} samples of $y(\boldsymbol{x})$ and $\boldsymbol{p}$ and solve Eq. (1) for each sample yielding the ensemble of solutions $\{u^{(i)}(\boldsymbol{x}, t)\}_{i=1}^{N_{\text{train}}}$.
2. Use the ensemble $\{u^{(i)}(\boldsymbol{x}, t)\}_{i=1}^{N_{\text{train}}}$ to compute the sample mean and covariance of $u(\boldsymbol{x}, t)$ as

$$\bar{u}(\boldsymbol{x}, t) = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} u^{(i)}(\boldsymbol{x}, t), \tag{5}$$

$$C_u(\boldsymbol{x}, \boldsymbol{x}', t, t') = \frac{1}{N_{\text{train}} - 1} \sum_{i=1}^{N_{\text{train}}} [u^{(i)}(\boldsymbol{x}, t) - \bar{u}(\boldsymbol{x}, t)][u^{(i)}(\boldsymbol{x}', t') - \bar{u}(\boldsymbol{x}', t')]. \tag{6}$$

3. Compute the eigenvalues and eigenfunctions in the KLEs of $u(\boldsymbol{x}, t)$ and $y(\boldsymbol{x})$ by solving the eigenvalue problems:

$$\int_D \int_T C_u(\boldsymbol{x}, \boldsymbol{x}', t, t') \phi_i(\boldsymbol{x}', t') \, \mathrm{d}\boldsymbol{x}' \mathrm{d}t' = \eta_i \phi_i(\boldsymbol{x}, t) \quad i = 1, \ldots, N_\eta \tag{7}$$

and

$$\int_D C_y(\boldsymbol{x}, \boldsymbol{x}') \chi_i(\boldsymbol{x}') \, \mathrm{d}\boldsymbol{x}' = \xi_i \chi_i(\boldsymbol{x}) \quad i = 1, \ldots, N_\xi. \tag{8}$$

We assume that the numerical solutions $\{u(\boldsymbol{x}, t)^{(i)}\}_{i=1}^{N_{\text{train}}}$ are available on a $N_x \times N_t$ space–time mesh (N_x is the number of spatial nodes and N_t is the number of time steps) and $\{y(\boldsymbol{x})^{(i)}\}_{i=1}^{N_{\text{train}}}$ are available at N_x points in space. Then, the numerical solution of Eqs. (7) and (8) will yield the values of eigenfunctions $\{\phi_i(\boldsymbol{x}, t)\}_{i=1}^{N_\eta}$ on the $N_x \times N_t$ mesh and $\{\chi_i(\boldsymbol{x})\}_{i=1}^{N_\xi}$ on the N_x spatial mesh.

4. For the selected $y^*(\boldsymbol{x})$ and $\boldsymbol{p}^*$, find the solution of Eq. (1) by minimizing the PDE residuals on the $N_x \times N_t$ mesh as

$$\boldsymbol{\eta}^* = \min_{\boldsymbol{\eta}} \Big[\sum_{i=1}^{N_x} \sum_{j=1}^{N_t} \mathcal{L}(\hat{u}(\boldsymbol{x}_i, t_j; \boldsymbol{\eta}), \boldsymbol{p}^*, \hat{y}(\boldsymbol{x}_i; \boldsymbol{\xi}^*))^2 \Big]. \tag{9}$$

As in other ROM models [21], the dPICKLE minimization problem (9) must be solved for any new $y(\boldsymbol{x})$ and $\boldsymbol{p}$, which is faster than solving the original PDE (1) with standard numerical methods but may still take considerable time. Also, solving the minimization problem (9) requires numerically computing the space and time derivatives of the eigenfunctions, which makes this method *intrusive*.

In the KL-DNN method, we replace the minimization problem (9) in the dPICKLE model with a DNN model to map $\boldsymbol{\xi}$ and $\boldsymbol{p}$ to $\boldsymbol{\eta}$:

$$\boldsymbol{\eta}(\boldsymbol{p}, \boldsymbol{\xi}) \approx \mathcal{NN}(\boldsymbol{p}, \boldsymbol{\xi}, \boldsymbol{\theta}), \tag{10}$$

where $\mathcal{NN}(\boldsymbol{p}, \boldsymbol{\xi}, \boldsymbol{\theta})$ is a fully connected feed-forward DNN with the input layer containing $\boldsymbol{\xi}$ and $\boldsymbol{p}$, output layer composed of $\boldsymbol{\eta}$, and parameters (weights and biases) $\boldsymbol{\theta}$. The KL-DNN method takes full advantage of the set of N_{train} solutions obtained in Step 1 of the dPICKLE method by using it to both compute the mean and eigenpairs (as in the dPICKLE method) and train the DNN in Eq. (10).

The training set for estimating $\boldsymbol{\theta}$ is generated from the ensemble $\{\boldsymbol{p}^{(i)}, y(x)^{(i)}, u(\boldsymbol{x}, t)^{(i)}\}_{i=1}^{N_{\text{train}}}$ as follows:

1. For the solution $u(\boldsymbol{x}, t)^{(i)}$ with the parameters $\boldsymbol{p}^{(i)}$ and $y(x)^{(i)}$ ($i = 1, 2, \ldots, N_{\text{train}}$), compute $\boldsymbol{\xi}^{(i)}$ and $\boldsymbol{\eta}^{(i)}$ by least-square fitting the KLEs $\hat{y}(\boldsymbol{x}, \boldsymbol{\xi})$ and $\hat{u}(\boldsymbol{x}, t, \boldsymbol{\eta})$ to $y(\boldsymbol{x})^{(i)}$ and $u(\boldsymbol{x}, t)^{(i)}$, respectively, as

$$\boldsymbol{\xi}^{(i)} = \min_{\boldsymbol{\xi}} \sum_{j=1}^{N_x} [y(\boldsymbol{x}_j)^{(i)} - \hat{y}(\boldsymbol{x}_j, \boldsymbol{\xi})]^2 \tag{11}$$

and

$$\boldsymbol{\eta}^{(i)} = \min_{\boldsymbol{\eta}} \sum_{j=1}^{N_x} \sum_{k=1}^{N_t} [u(\boldsymbol{x}_j, t_k)^{(i)} - \hat{u}(\boldsymbol{x}_j, t_k, \boldsymbol{\eta})]^2. \tag{12}$$

2. Train the DNN by minimizing the loss function $\mathcal{L}_\eta(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} \mathcal{L}_\eta(\boldsymbol{\theta}) \tag{13}$$

where

$$\mathcal{L}_\eta(\boldsymbol{\theta}) = \lambda_\eta \sum_{i=1}^{N_{\text{train}}} \|\mathcal{NN}(\boldsymbol{p}^{(i)}, \boldsymbol{\xi}^{(i)}; \boldsymbol{\theta}) - \boldsymbol{\eta}^{(i)}\|_2^2 + \lambda_\theta \|\boldsymbol{\theta}\|_2^2, \tag{14}$$

$\|\boldsymbol{\theta}\|_2^2$ is the ℓ_2 regularization term, and λ_η and λ_θ are the weights.

3. Once the DNN is trained, the solution for any $y^*(\boldsymbol{x})$ and $\boldsymbol{p}^*$ is given as

$$u(\boldsymbol{x},t) \approx \hat{u}\left(\boldsymbol{x},t;\mathcal{NN}(\boldsymbol{p}^*,\boldsymbol{\xi}^*;\boldsymbol{\theta})\right). \tag{15}$$

In the KL-DNN method, the total error ϵ_t is a combination of the approximation error ϵ_{ap} (error due to approximating the PDE solution with a KLE) and the *operator* error (the DNN operator that maps the parameters to the PDE solution), ϵ_{op}. The relative ϵ_t, ϵ_{ap}, and ϵ_{op} errors can be defined as

$$\epsilon_t = \frac{\sqrt{\sum_{j=1}^{N_x}\sum_{k=1}^{N_t}[u(\boldsymbol{x}_j,t_k)-\hat{u}\left(\boldsymbol{x}_j,t_k,\mathcal{NN}(\boldsymbol{p},\boldsymbol{\xi};\boldsymbol{\theta}^*)\right)]^2}.}{\sqrt{\sum_{j=1}^{N_x}\sum_{k=1}^{N_t}u(\boldsymbol{x}_j,t_k)^2}}, \tag{16}$$

$$\epsilon_{op} = \frac{\|\boldsymbol{\eta}^* - \mathcal{NN}(\boldsymbol{\xi},\boldsymbol{p},\boldsymbol{\theta}^*)\|_2}{\|\boldsymbol{\eta}^*\|_2} \tag{17}$$

(where $\boldsymbol{\xi}$ and $\boldsymbol{p}$ are the parameters in the exact PDE solution $u(x,t)$), and

$$\epsilon_{ap} = \frac{\sqrt{\sum_{j=1}^{N_x}\sum_{k=1}^{N_t}[u(\boldsymbol{x}_j,t_k)-\hat{u}(\boldsymbol{x}_j,t_k,\boldsymbol{\eta}^*)]^2}}{\sqrt{\sum_{j=1}^{N_x}\sum_{k=1}^{N_t}u(\boldsymbol{x}_j,t_k)^2}}, \tag{18}$$

where $\boldsymbol{\eta}^*$ is the solution of the least square fitting of $\hat{u}(\boldsymbol{x},t;\boldsymbol{\eta})$ to $u(\boldsymbol{x},t)$:

$$\boldsymbol{\eta}^* = \min_{\boldsymbol{\eta}} \sum_{j=1}^{N_x}\sum_{k=1}^{N_t}[u(\boldsymbol{x}_j,t_k)-\hat{u}(\boldsymbol{x}_j,t_k,\boldsymbol{\eta})]^2. \tag{19}$$

The error ϵ_{ap} decreases as N_η increases. Also, ϵ_{ap} might depend on the accuracy of the mean and covariance estimates, which improve with increasing N_{train}. Therefore, ϵ_{ap} decreases with increasing N_{train}. The ϵ_{op} error is expected to decrease as the DNN size increases. However, a larger DNN requires more data for training, i.e., a larger N_{train}.

In the PICKLE method, the total error is a combination of the approximation error (which is defined similarly to ϵ_{ap} in the KL-DNN method) and the error in the residual least-square approximation of the PDE solution. The latter error decreases with increasing N_x and N_t, i.e., with increasing mesh resolution.

It must be noted that the formulation of the KL-DNN method is similar to that of the PCA-Net method, where PCA expansions are used in Eqs. (2) and (4) instead of KLEs and $\{u^{(i)}(\boldsymbol{x},t)\}_{i=1}^{N_{train}}$ and C_u are treated as the data and data covariance, respectively. KLEs allow treating u, $\boldsymbol{p}$, and y as random variables and the governing PDEs as stochastic. The stochastic interpretation has advantages for computing C_u and $(\eta_i,\phi_i)_{i=1}^{N_\eta}$, which is especially important when obtaining enough u_i samples becomes too expensive. Equally important, the size of the C_u matrix for multi-dimensional time-dependent PDEs can become so large that solving the eigenvalue problem or performing SVD would be unfeasible.

In the stochastic interpretation, Eqs. (5) and (6) constitute the Monte Carlo (MC) method, which has a relatively low convergence rate. The stochastic treatment of the governing equations allows using other methods (e.g., Polynomial Chaos [22,23] and the moment equation method [24,25]) for computing the mean and covariance of the state variables that are more efficient than the MC method. Also, treating u as a random field can enable the solution of very large eigenvalue problems by rewriting the KLE of u as [15]:

$$u(\boldsymbol{x},t,\omega) \approx \bar{u}(\boldsymbol{x},t) + \sum_{i=1}^{N_\eta}\phi_i(\boldsymbol{x},t)\sqrt{\lambda_i(t)}\eta_i(t,\omega), \tag{20}$$

where ω is the coordinate in the outcome space, $\eta_i(t,\omega)$ are the random variables, and $\lambda_i(t)$ and $\phi_i(\boldsymbol{x},t)$ are the eigenvalues and eigenfunctions of the covariance $C_u(t,\boldsymbol{x},\boldsymbol{x}')$, respectively, given by the solution of the eigenvalue problem:

$$\int_D C_u(\boldsymbol{x},\boldsymbol{x}',t)\phi_i(\boldsymbol{x}',t)\,\mathrm{d}\boldsymbol{x}' = \eta_i(t,\omega)\phi_i(\boldsymbol{x},t) \quad i=1,\ldots,N_\eta. \tag{21}$$

By construction, $\eta_i(t,\omega)$ are zero mean independent random variables, which can be expressed with one-dimensional KLEs as

$$\eta_i(t,\omega) \approx \sum_{j=1}^{N_\gamma} g_{ij}(t)\sqrt{\mu_{ij}}\gamma_{ij}, \tag{22}$$

where γ_{ij} are independent and identically distributed (i.i.d.) zero-mean random variables and $g_{ij}(t)$ and μ_{ij} are, respectively, the eigenfunctions and eigenvalues of $H_i(t,t')$, the covariance of $\eta_i(t,\omega)$, which is defined in [15]. These eigenpairs can be found by solving the eigenvalue problem

$$\int_T H_i(t,t')g_{ij}(t')\,\mathrm{d}t' = \mu_{ij}g_{ij}(t) \quad j=1,\ldots,N_\gamma. \tag{23}$$

The dimensionality of each of the eigenvalue problems in Eqs. (21) and (23) is smaller than the dimensionality of the eigenvalue problem (7). The surrogate model can be constructed by mapping $\boldsymbol{p}$ and $\boldsymbol{\xi}$ to the KLE parameters γ_{ij}.

3. Uncertainty quantification: Deep ensembling and randomized KL-DNN method

There are several sources of uncertainty in the KL-DNN model. One is due to the nonlinearity of the least-square problem in Eq. (13). Despite adding the ℓ_2 regularization term, we find that the solution of Eq. (13) depends on the initial guess for the DNN parameters $\boldsymbol{\theta}$. Here, we use the iterative L-BFGS algorithm to solve this minimization problem. We find that stochastic gradient descent algorithms such as Adam produce similar (initialization-dependent) results. We call this source of uncertainty the DNN training uncertainty. The DNN training uncertainty can be quantified using the DE method, where the minimization problem (14) is randomly initialized and solved N_{ens} times for different values of the seed, yielding the ensemble of DNN parameters $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^{N_{\text{ens}}}$. Then, the DE prediction of $u(\boldsymbol{x},t)$ for the parameters $\boldsymbol{p}$ and $\boldsymbol{\xi}$ is given by the sample mean $\bar{u}_{\text{DE}}(\boldsymbol{x},t)$, and uncertainty is described by the variance $\sigma^2_{\text{DE}}(\boldsymbol{x},t)$ defined as:

$$\bar{u}_{\text{DE}}(\boldsymbol{x},t;\boldsymbol{p},\boldsymbol{\xi}) = \frac{1}{N_{\text{ens}}}\sum_{i=1}^{N_{\text{ens}}} \hat{u}\left(\boldsymbol{x},t,\mathcal{NN}(\boldsymbol{p},\boldsymbol{\xi};\boldsymbol{\theta}^{(i)})\right),$$

$$\sigma^2_{\text{DE}}(\boldsymbol{x},t;\boldsymbol{p},\boldsymbol{\xi}) = \frac{1}{N_{\text{ens}}-1}\sum_{i=1}^{N_{\text{ens}}} [\hat{u}\left(\boldsymbol{x},t,\mathcal{NN}(\boldsymbol{p},\boldsymbol{\xi};\boldsymbol{\theta}^{(i)})\right) - \bar{u}_{\text{DE}}(\boldsymbol{x},t;\boldsymbol{p},\boldsymbol{\xi})]^2.$$

There is also uncertainty due to the DNN regression model error and the prior assumptions about the distribution of $\boldsymbol{\theta}$ expressed in the regularization term in Eq. (14). Traditionally, this uncertainty is treated probabilistically using the Bayesian framework that treats the DNN parameters $\boldsymbol{\theta}$ as random variables with the posterior distribution $P(\boldsymbol{\theta}|\boldsymbol{d})$ given the measurements $\boldsymbol{d}$ defined by the Bayes' rule:

$$P(\boldsymbol{\theta}|\boldsymbol{d}) = \frac{P(\boldsymbol{d}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{\int P(\boldsymbol{d}|\boldsymbol{\theta})P(\boldsymbol{\theta})d\boldsymbol{\theta}}, \tag{24}$$

where $\boldsymbol{d} = \{\boldsymbol{\eta}^{(i)}\}_{i=1}^{N_{\text{train}}}$ is the training data, $P(\boldsymbol{\theta})$ is the prior distribution of $\boldsymbol{\theta}$, and $P(\boldsymbol{d}|\boldsymbol{\theta})$ is the likelihood function. The integral in the denominator of Eq. (24) is the normalization coefficient, which makes the posterior distribution integrate to one.

The likelihood corresponding to the first term in the KL-DNN loss function is

$$P(\boldsymbol{d}|\boldsymbol{\theta}) = \left(\frac{1}{\sqrt{2\pi}\sigma_\eta}\right)^{N_{\text{train}}+N_\eta}\prod_{i=1}^{N_{\text{train}}}\prod_{j=1}^{N_\eta}\exp\left[-\frac{(\eta_j^{(i)} - \hat{\eta}_j(\boldsymbol{p}^{(i)},\boldsymbol{\xi}^{(i)};\boldsymbol{\theta}))^2}{2\sigma_\eta^2}\right], \tag{25}$$

where $\sigma_\eta^2 = \frac{1}{\lambda_\eta}$. The prior corresponding to the ℓ_2 regularization term in the loss function is

$$P(\boldsymbol{\theta}) = \left(\frac{1}{\sqrt{2\pi}\sigma_\theta}\right)^{N_\theta}\prod_{i=1}^{N_\theta}\exp\left(-\frac{\theta_i^2}{2\sigma_\theta^2}\right), \tag{26}$$

where $\sigma_\theta^2 = \frac{1}{\lambda_\theta}$. With these choices of the prior and likelihood, the DE solutions $\boldsymbol{\theta}^{(i)}$ of the minimization problem (13) provide the modes of the posterior distribution.

The standard approach for sampling $P(\boldsymbol{\theta}|\boldsymbol{d})$ is the Hamiltonian Monte Carlo (HMC) method. However, HMC suffers from the curse of dimensionality, i.e., the computational cost of HMC rapidly increases with the increasing dimensionality of $\boldsymbol{\theta}$. As an alternative, here we propose a randomized sampling algorithm, which is based on the randomization of the loss function (14) as

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{\sigma_\eta^2}\sum_{i=1}^{N_{\text{train}}}\|\mathcal{NN}(\boldsymbol{\xi}_i,\boldsymbol{\theta}) - \boldsymbol{\eta}_i^{data} - \boldsymbol{\alpha}_i\|_2^2 + \frac{1}{\sigma_\theta^2}\|\boldsymbol{\theta}-\boldsymbol{\beta}\|_2^2, \tag{27}$$

where $\boldsymbol{\alpha}_i$ $(i = 1,\dots,N_{\text{train}})$ and $\boldsymbol{\beta}$ are vectors of i.i.d. Gaussian random variables with zero mean and variances $\sigma_\eta^2 = \frac{1}{\lambda_\eta}$ and $\sigma_\theta^2 = \frac{1}{\lambda_\theta}$, respectively.

Minimizing the randomized loss function for different realizations of $\{\boldsymbol{\alpha}_i\}_{i=1}^{N_{\text{train}}}$, $\boldsymbol{\beta}$, and the random initial guesses of $\boldsymbol{\theta}$ produces samples $\{\boldsymbol{\theta}_{\text{r}}^{(i)}\}_{i=1}^{N_{\text{ens}}}$ of the posterior distribution of $\boldsymbol{\theta}$. We name this the rKL-DNN method. The subscript "r" is used to distinguish $\boldsymbol{\theta}_{\text{r}}^{(i)}$ from $\boldsymbol{\theta}^{(i)}$ obtained by minimizing the loss function (14) using DE.

For a linear (in $\boldsymbol{\theta}$) model of $\boldsymbol{\eta}$, the rKL-DNN-sampled distribution converges to the posterior distribution as the number of samples approaches infinity (see, for example, the convergence proof for the rPICKLE model in [19]). For nonlinear models such as the DNN model in our approach, the proposed sampling scheme can introduce bias, which can be removed using Metropolis rejection schemes. However, it was found that the rejection rates in similar randomized schemes (e.g., rPICKLE and randomized MAP) are very small [16,19]. Therefore, here we accept all samples $\boldsymbol{\theta}_{\text{r}}^{(i)}$.

An important question in the Bayesian estimation of uncertainty is selecting the variances σ_η^2 and σ_θ^2. We note that the solution $\boldsymbol{\theta}^*$ of the minimization problem with the *deterministic* loss function (14) only depends on the ratio $\lambda_{reg} = \lambda_\theta/\lambda_\eta = \sigma_\eta^2/\sigma_\theta^2$, which is the regularization coefficient in the loss (14). This regularization coefficient can be determined to minimize the error in the surrogate model prediction for a testing data set.

On the other hand, the posterior distribution depends on the values of σ_η^2 and σ_θ^2 and not just their ratio. Here, we select the values of σ_η^2 and σ_θ^2 to maximize the log predictive probability (LPP) with respect to *testing* data that is defined as the sum of the

point-wise log probabilities of the reference being observed given the statistical prediction [26]:

$$\mathrm{LPP} = -\sum_{i=1}^{N_x}\sum_{j=1}^{N_t}\Bigg\{\frac{[\bar{u}_{\mathrm{rKL\text{-}DNN}}(\boldsymbol{x}_i,t_j;\boldsymbol{\xi}_{\mathrm{test}},\boldsymbol{p}_{\mathrm{test}})-u_{\mathrm{test}}(\boldsymbol{x}_i,t_j)]^2}{2\sigma^2_{\mathrm{rKL\text{-}DNN}}(\boldsymbol{x}_i,t_j;\boldsymbol{\xi}_{\mathrm{test}},\boldsymbol{p}_{\mathrm{test}})} + \frac{1}{2}\log[2\pi\sigma^2_{\mathrm{rKL\text{-}DNN}}(\boldsymbol{x}_i,t_j;\boldsymbol{\xi}_{\mathrm{test}},\boldsymbol{p}_{\mathrm{test}})]\Bigg\}, \tag{28}$$

where $\bar{u}(\boldsymbol{x}_i,t_j;\boldsymbol{\xi}_{\mathrm{test}},\boldsymbol{p}_{\mathrm{test}})$ and $\sigma^2_{\mathrm{rKL\text{-}DNN}}(\boldsymbol{x}_i,t_j;\boldsymbol{\xi}_{\mathrm{test}},\boldsymbol{p}_{\mathrm{test}})$ are the sample mean and variance of $u(\boldsymbol{x},t)$ given the parameters $(\boldsymbol{\xi}_{\mathrm{test}},\boldsymbol{p}_{\mathrm{test}})$ computed from rKL-DNN method as

$$\bar{u}_{\mathrm{rKL\text{-}DNN}}(\boldsymbol{x},t;\boldsymbol{p},\boldsymbol{\xi}) = \frac{1}{N_{\mathrm{ens}}}\sum_{i=1}^{N_{\mathrm{ens}}}\hat{u}\left(\boldsymbol{x},t,\mathcal{NN}(\boldsymbol{p},\boldsymbol{\xi};\boldsymbol{\theta}_{\mathrm{r}}^{(i)})\right) \tag{29}$$

and

$$\sigma^2_{\mathrm{rKL\text{-}DNN}}(\boldsymbol{x},t;\boldsymbol{p},\boldsymbol{\xi}) = \frac{1}{N_{\mathrm{ens}}-1}\sum_{i=1}^{N_{\mathrm{ens}}}[\hat{u}\left(\boldsymbol{x},t,\mathcal{NN}(\boldsymbol{p},\boldsymbol{\xi};\boldsymbol{\theta}_{\mathrm{r}}^{(i)})\right)-\bar{u}_{rKL-DNN}(\boldsymbol{x},t;\boldsymbol{p},\boldsymbol{\xi})]^2. \tag{30}$$

4. One-dimensional advection–diffusion equation

In this section, we consider the one-dimensional advection–diffusion equation (ADE)

$$\frac{\partial u}{\partial t}+v\frac{\partial u}{\partial x}=D\frac{\partial^2 u}{\partial x^2},\quad (x,t)\in(0,L)\times(0,T) \tag{31}$$

subject to the initial and boundary conditions:

$$u(0,t)=C_0, \tag{32}$$

$$u(\infty,t)=0, \tag{33}$$

$$u(x,0)=0, \tag{34}$$

where v and D are the (constant) velocity and the diffusion coefficient. Our aim is to train a surrogate model $\hat{u}(x,t,v,D;\boldsymbol{\theta})$, which predicts $u(x,t)$ for any values of v and D in the ranges $v\in[v_{min},v_{max}]$ and $D\in[D_{min},D_{max}]$.

This ADE allows an approximate analytical solution known as the Ogata-Banks solution [27]. We use this analytical solution to generate a training dataset for constructing the KLE $\hat{u}(x,t;\boldsymbol{\eta})$ and training the DNN map $\mathcal{NN}(\boldsymbol{p};\boldsymbol{\theta})$, where $\boldsymbol{p}=[v,D]^T$. In Step 1 of the dPICKLE algorithm, we sample v and D from the independent uniform distributions with the ranges $v\in[v_{min},v_{max}]$ and $D\in[D_{min},D_{max}]$, respectively, and evaluate $u(x,t)$ for each sample of v and D on a structured mesh with the grid size Δx and time step Δt. Then, we compute the KLE $\hat{u}(x,t;\boldsymbol{\eta})$ using Steps 2 and 3 of the dPICKLE algorithm. Finally, we train the surrogate model and evaluate its accuracy for several combinations of v and D. We compare the accuracy of the surrogate and dPICKLE models as a function of N_{train} and Δx and Δt. The dependence of errors on N_{train} is important because generating samples (when the analytical solution is not available) is a significant part of the total computational costs of both the KL-DNN and dPICKLE algorithms. The dependence of errors on Δx and Δt is also important because the Δx and Δt determine the size of the discrete problem in the numerical solution of the PDE and the size of the eigenvalue problem.

We solve the ADE for $L=86$ and $T=600$, and train the PICKLE and KL-DNN models for the parameters in the range $D_{min}=0.036$, $D_{max}=0.085$, $v_{min}=0.128$, and $v_{max}=0.298$. These parameters correspond to the range of Peclet numbers $Pe=\frac{vL}{D}$ of $[128,700]$.

First, we compute the ADE solutions for $Pe=175$ and 600 using the DE-KL-DNN method, where the prediction is given by the mean $\bar{u}_{DE}(x,t;)$ and the uncertainty is quantified by the variance $\sigma^2_{\mathrm{DE}}(\boldsymbol{x},t)$. We set the regularization coefficient to $\lambda_{\mathrm{reg}}=10^{-6}$ and, unless mentioned otherwise, $N_{\mathrm{ens}}=100$.

Fig. 1 presents the analytical solution for $Pe=175$ and 600, and Fig. 2 shows point errors in the KL-DNN and PICKLE solutions for $Pe=175$ ($v=0.15$ and $D=0.07$) obtained with $N_{\mathrm{train}}=20$, 100, and 1000. The errors are computed with respect to the analytical solutions shown in Fig. 1. We find that the KL-DNN solutions have smaller point errors than the PICKLE solutions for all considered N_{train}. The point errors in KL-DNN decrease with increasing N_{train}, while the PICKLE point errors do not change significantly.

The same trend can be seen in Fig. 3 for the ϵ_{t} error in the KL-DNN and PICKLE solutions as functions of N_{train} for $Pe=175$ and 600. The PICKLE ϵ_{t} error is practically independent of N_{train}, while the KL-DNN ϵ_{t} error decreases by one order of magnitude as N_{train} increases from 20 to 1000. For comparison, we also show the relative approximation error ϵ_{ap}. Both ϵ_{ap} and ϵ_{t} errors in the PICKLE and KL-DNN solutions increase with Pe.

For $Pe=175$, ϵ_{ap} is significantly smaller than ϵ_{t} in the PICKLE and KL-DNN solutions, demonstrating that the main source of error in the KL-DNN and PICKLE methods comes from DNN mapping and residual least-square formulation, respectively, rather than from the KL representation of $u(x,t)$. Also, we observe that ϵ_{ap} is practically independent of N_{train}. For $Pe=600$, the ϵ_{ap} is only slightly smaller than ϵ_{t} in the KL-DNN solution. This indicates that KL-DNN as well as PICKLE solutions can be further improved by increasing N_η, which is expected to decrease ϵ_{ap} and (therefore) ϵ_{t}.

Fig. 3 (left panel) shows that (for fixed N_η) ϵ_{ap} is practically independent of N_{train}, which is the reason for ϵ_{t} in PICKLE to be independent of N_{train}. On the other hand, in KL-DNN, ϵ_{t} decreases with increasing N_{train}.

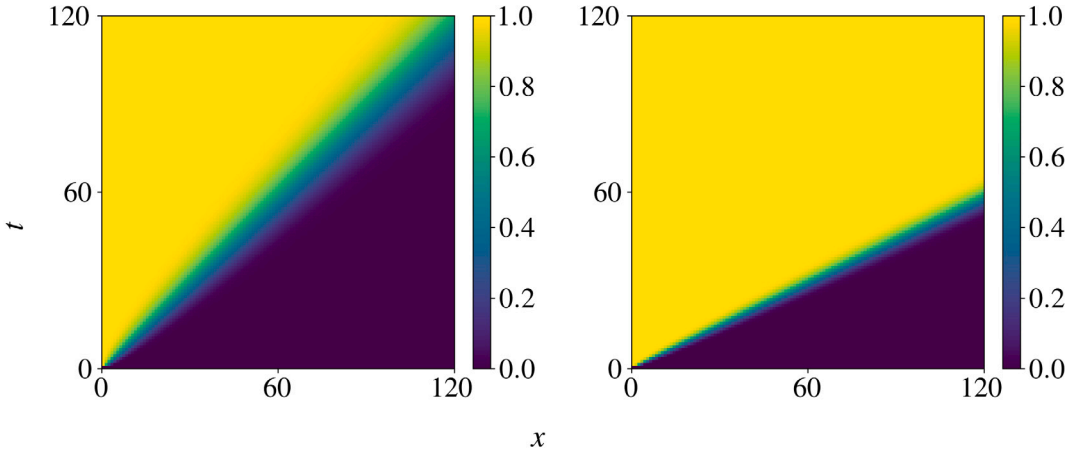

Fig. 1. Analytical solutions of ADE for $Pe = 175$ (left) and $Pe = 600$ (right).

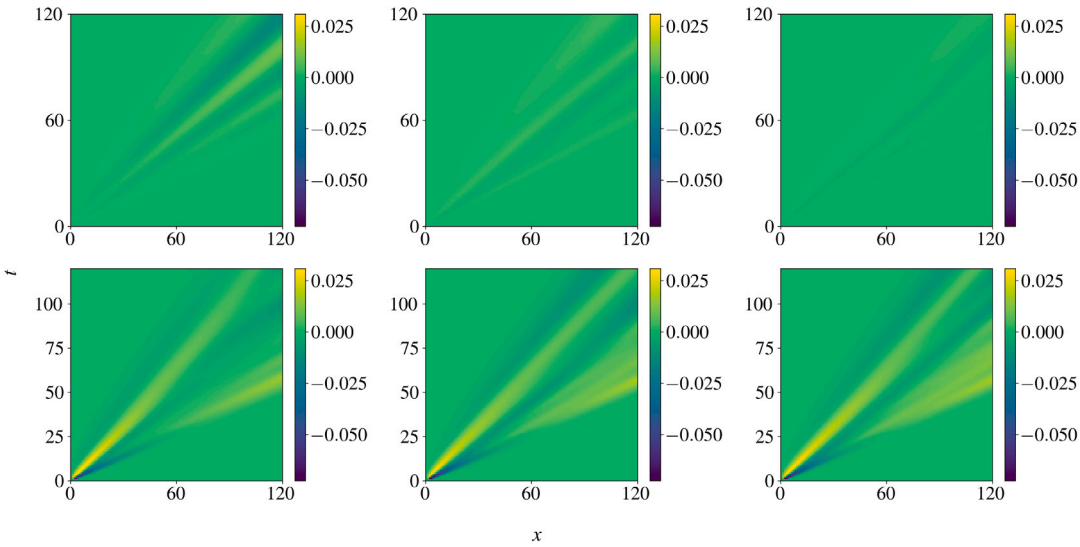

Fig. 2. Point errors in the DE-KL-DNN (top row) and PICKLE (bottom row) solutions of ADE obtained with $N_{\text{train}} = 20$, 100, and 1000. The solutions are obtained for $v = 0.15$ and $D = 0.07$ corresponding to $Pe = 175$. The space–time mesh resolution is 120×120. In the DE-KL-DNN method, the regularization parameter is $\lambda_{reg} = 10^{-4}$ for $N_{\text{train}} = 20$ and 100 and $\lambda_{reg} = 10^{-6}$ for $N_{\text{train}} = 1000$. The number of terms in the KLE is $N_\eta = 20$.

In Fig. 3 (right panel), we find that ϵ_{ap} decreases with N_{train} and reaches an asymptotic value at $N_{\text{train}} = 200$. The KL-DNN ϵ_{t} error also decreases with N_{train} until it reaches an asymptotic value at $N_{\text{train}} = 500$. The PICKLE ϵ_{t} error is practically independent of N_{train} and, for the smallest tested $N_{\text{train}} = 20$, the PICKLE solution is more accurate than the KL-DNN solution.

These results show that the KL-DNN method is an efficient alternative to PICKLE when a sufficient number of samples is available for training the DNN map. On the other hand, the physics constraints in PICKLE allow obtaining an accurate solution even with relatively few samples—in the case with $Pe = 600$, $N_{\text{train}} = N_\eta = 20$ is sufficient to obtain an accurate PICKLE solution.

The error in the PICKLE solution can be further decreased by decreasing Δx and Δt as seen in Fig. 4, which displays ϵ_{t} errors in the KL-DNN and PICKLE solutions as functions of $N_x = N_t$ for $Pe = 175$ and 600 and $N_{\text{train}} = 20$. The PICKLE ϵ_{t} error decreases with increasing N_x, while the KL-DNN ϵ_{t} error is practically independent of the grid resolution. The KL-DNN error is smaller than the PICKLE error for smaller Pe and larger than the PICKLE error for larger Pe for all considered N_x and N_t. We reiterate that for $Pe = 600$, the KL-DNN method becomes more accurate than PICKLE for larger N_{train}.

Fig. 4 also shows that for a fixed N_η, the approximation error does not significantly depend on N_x and N_t. The reason for this can be seen in Fig. 5, which shows the decay of eigenvalues of the covariance matrices for $Pe = 175$ computed on meshes $N_x = N_t = 60$ and 180. The first 40 eigenvalues of the two covariance matrices have similar values. In the presented above ADE solutions, we use the first 20 eigenvalues in the KLE, and the resulting truncation errors (rtol) computed from Eq. (3) are 2.012×10^{-6} and 2.008×10^{-6} for $N_x = N_t = 60$ and 180, respectively.

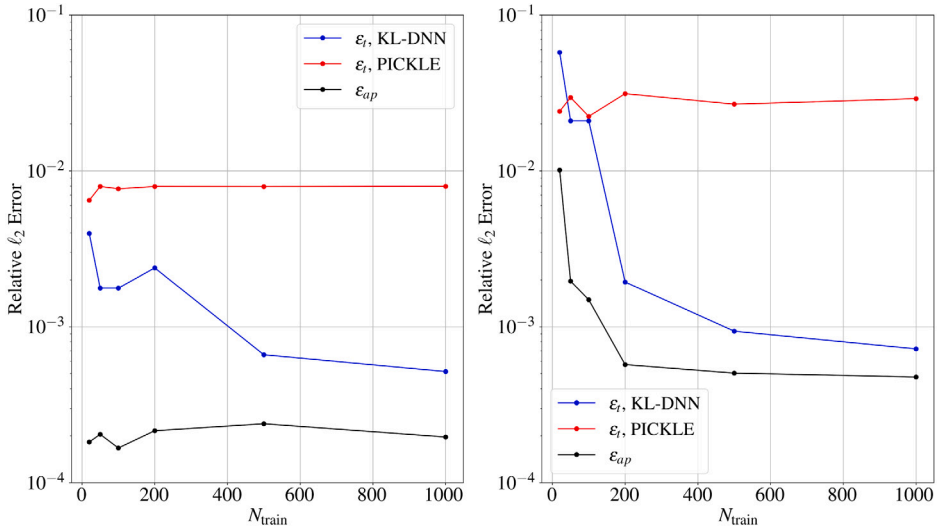

Fig. 3. The total ϵ_{t} errors in the DE-KL-DNN and PICKLE solutions and the approximation error ϵ_{op} as functions of N_{train} for (left) $V = 0.7V^*$ and $D = 1.2D^*$ (Pe = 175) and (right) $V = 1.4V^*$ and $D = 0.7D^*$ (Pe = 600).

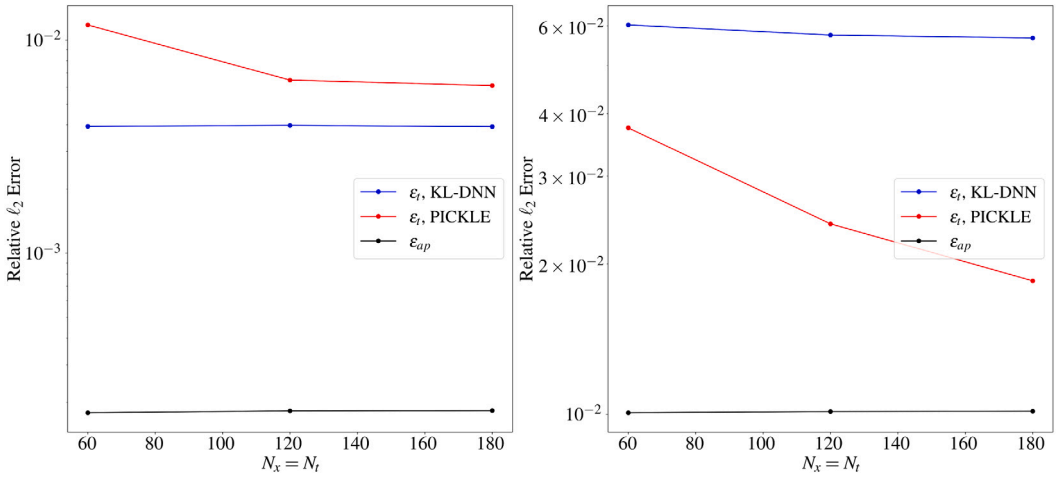

Fig. 4. Relative ℓ_2 errors in the KL-DNN and PICKLE ADE solutions as functions of $N_x = N_t$ and (left) $Pe = 175$ ($V = 0.7V^*$ and $D = 1.2D^*$) and (right) $Pe = 600$ ($V = 1.4V^*$ and $D = 0.7D^*$). $N_{\text{train}} = 20$. For comparison, the relative approximation error ϵ_{ap} is also provided.

Fig. 6 shows the standard deviation in the KL-DNN solution computed with DE for $N_{\text{train}} = 20$, 100, and 1000, and $Pe = 175$. The uncertainty in the KL-DNN prediction decreases with increasing N_{train}, as do the point and ϵ_{t} errors in Figs. 2 and 3. It is important to note that the point and ϵ_{t} error estimates require the reference solution (which, in general, is not available). The DE standard deviation estimate does not require the reference solution and can be used as an a priori measure of the solution accuracy.

Based on the results presented so far in this section, we conclude that the KL-DNN method can outperform the reduced-order methods such as PICKLE for problems where a sufficiently large number of samples is available, i.e., N_{train} is sufficiently large. Also, these KL-DNN results were obtained using the DE-KL-DNN method. Next, we compare the rKL-DNN and DE-KL-DNN methods. We set $\sigma_{\alpha}^2 = 10^{-8}$ and $\sigma_{\beta}^2 = 10^{-2}$ in the rKL-DNN model and $\lambda_{\text{reg}} = 10^{-6}$ in the DE-KL-DNN model.

Fig. 7 shows the ϵ_{t} errors in the ADE solutions given by $\bar{u}_{\text{rKL-DNN}}(x,t)$ and $\bar{u}_{\text{DE}}(x,t)$ with respect to the reference solution as functions of N_{ens}. We find that the errors in both methods are very similar (with the DE-KL-DNN errors being slightly smaller than the rKL-DNN errors) and practically independent of N_{ens}.

Fig. 8 depicts the standard deviations as functions of x and t estimated from the rKL-DNN and DE-KL-DNN methods. We find that rKL-DNN predicts larger standard deviations than the DE-KL-DNN. Fig. 8 also presents the *coverage* of the reference solution in the two methods where the yellow color denotes areas where the reference solution is within the confidence interval (mean plus/minus two standard deviations) predicted by each method. It can be seen that rKL-DNN provides better coverage of the reference solution

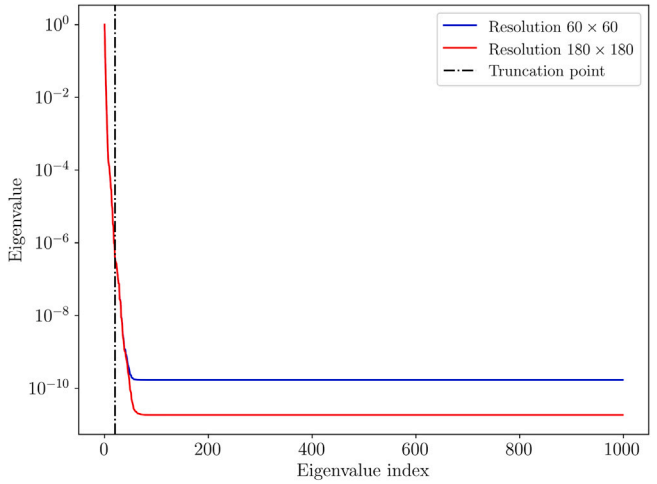

Fig. 5. Normalized eigenvalues λ_i (organized in the decaying order) of the u covariance matrixes computed on the 60×60 and 180×180 as functions of the index i. In the KL-DNN models, the KLE of u is truncated at $N_\eta = 20$.

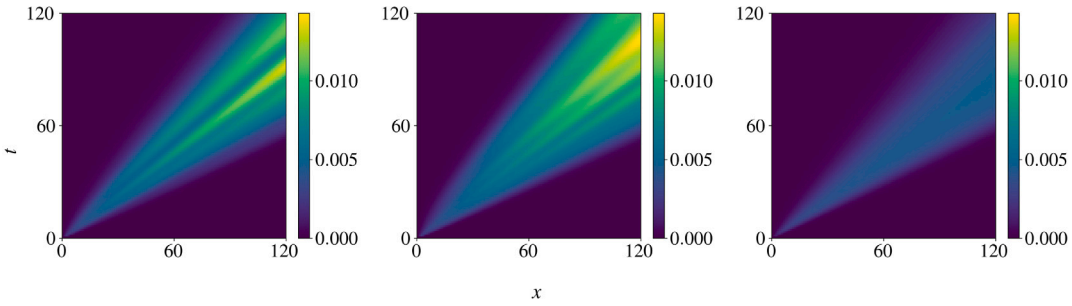

Fig. 6. Standard deviation $\sigma_{\mathrm{DE}}(x,t)$ in the ADE KL-DNN solution obtained from the DE-KL-DNN method for $Pe = 175$ and $N_{\mathrm{train}} = 20$ (left), 100 (middle), and 1000 (right).

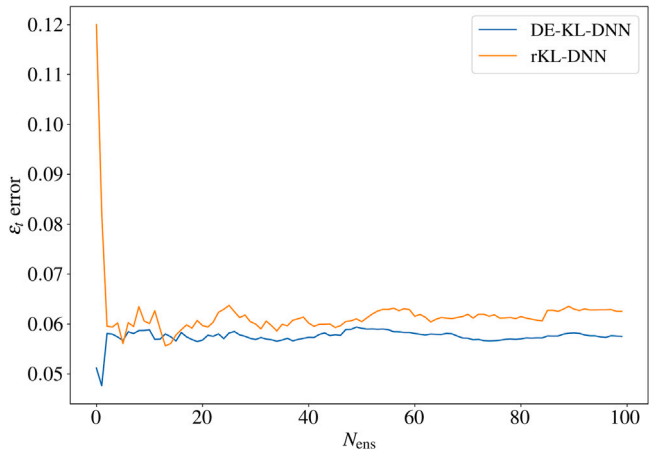

Fig. 7. DE-KL-DNN and rKL-DNN ϵ_{t} errors in the ADE solution with $Pe = 175$ as functions of N_{ens}. $N_{\mathrm{train}} = N_\eta = 20$.

(96.9%) than the DE-KL-DNN method (90.8%). Also, rKL-DNN has a higher LPP than DE-KL-DNN (14744.7 versus 13303.1). These results show that the rKL-DNN method provides a more accurate estimate of uncertainty than DE-KL-DNN.

Finally, in Fig. 9 we show the marginal and bivariate distributions of $\eta_1, \eta_2, \ldots, \eta_5$, which are used to train the rKL-DNN and DE-KL-DNN models. These distributions are highly non-Gaussian, suggesting that the distribution of $u^{(i)}$ samples in the dataset is

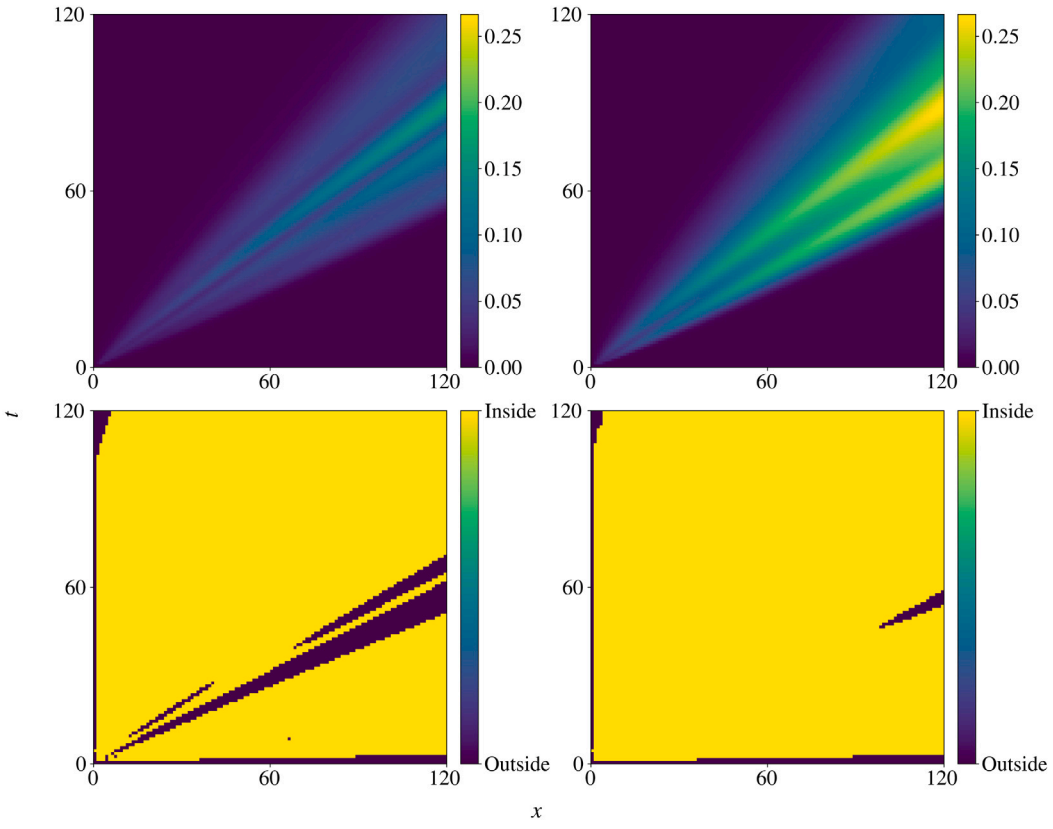

Fig. 8. Standard deviation (first row) and coverage (second row) of the ADE solution computed from the DE-KL-DNN (left column) and rKL-DNN (right column) using $N_{\text{train}} = N_\eta = 20$ and $N_{\text{ens}} = 100$. In the coverage plot, the yellow and blue colors denote areas where the reference solution is inside and outside the confidence interval, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

also non-Gaussiam. From the results in this section, we conclude that KL-DNN surrogate models can be well-trained using samples with a non-Gaussian distribution.

5. Nonlinear two-dimensional diffusion equation

In this section, we test the DE-KL-DNN and rKL-DNN methods for a synthetic groundwater flow problem known as the Freyberg problem [28,29]. The governing equation for this problem, describing two-dimensional depth-averaged groundwater flow in an unconfined aquifer, has the form:

$$S_y u(\boldsymbol{x},t)\frac{\partial u(\boldsymbol{x},t)}{\partial t} = \nabla\cdot(K(\boldsymbol{x})u(\boldsymbol{x},t)\nabla u(\boldsymbol{x},t)) + u(\boldsymbol{x},t)[f(t)+g(x,t)], \tag{35}$$

where S_y is the specific yield (assumed to be constant and known), $K(\boldsymbol{x})$ is the hydraulic conductivity, $f(t)$ is the time-dependent recharge, and $g(\boldsymbol{x},t)$ is the source term due to pumping wells and the interaction with a river. Our objective is to develop a surrogate model of $u(\boldsymbol{x},t)$ as a function of $y(\boldsymbol{x}) = \ln K(\boldsymbol{x})$ (no dependence on scalar parameters $\boldsymbol{p}$ is considered in this case). The $\ln K(\boldsymbol{x})$ transform is not necessary but is often used when the surrogate model is employed for solving the inverse problem of estimating $K(x)$ given some measurements of u. If a KLE is used to model K, then the solution of the inverse problem is reduced to estimating $\boldsymbol{\xi}$ in the KLE model. One of the constraints on $K(\boldsymbol{x})$ is that it must be positive, and representing $\ln K(\boldsymbol{x})$ with KLE ensures that the inverse solution always satisfies this constraint.

The reference log-transmissivity field $y_{\text{ref}}(\boldsymbol{x}) = \ln K_{\text{ref}}(\boldsymbol{x})$ and the computational domain are shown in Fig. 10. There are 40 rows and 20 columns in the domain. Each cell is a square with a side length of 250 m. The total domain extent is 5000 by 10 000 m. Cells colored black are *inactive*, i.e., the governing equation is not solved in these cells, and there are $N_m = 706$ active cells. The domain is conceptualized as a headwater catchment with no-flow boundaries on the north, east, and west sides. The boundary separating the active and inactive cells is also treated as a no-flow boundary. The southern boundary (shown by blue cells) has the Robin (generalized head) boundary condition. The cyan color indicates a river running from North to South across the model domain, with a specified inflow on the Northern end. The locations of the six pumping wells are indicated by red cells. The initial condition is computed as the solution of the steady-state version of Eq. (35) with the recharge $f_0 = f(t=0)$.

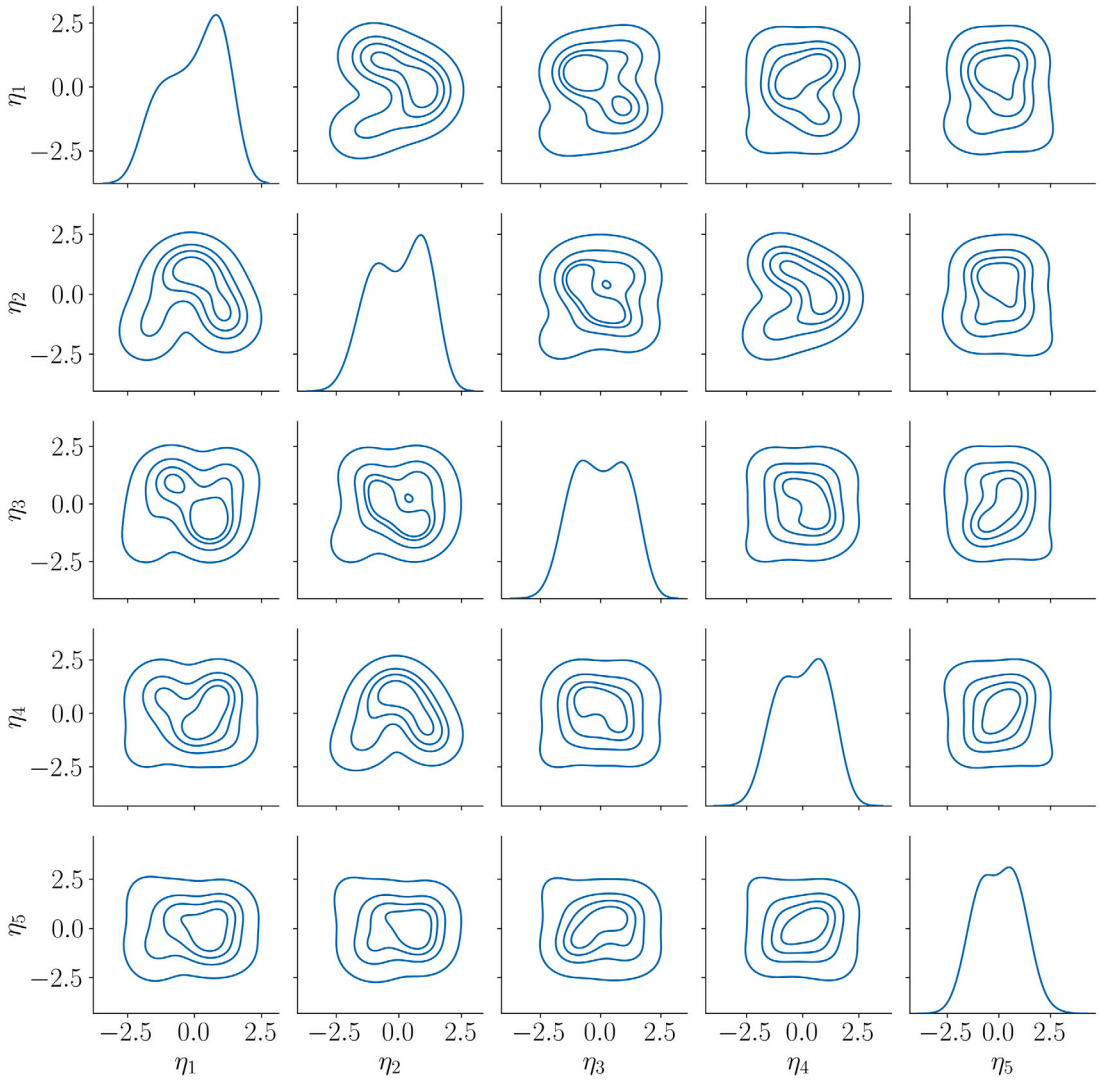

Fig. 9. The marginal and bivariate joint distributions $\eta_1, \eta_2, \ldots, \eta_5$ estimated from the ADE training dataset with $N_{\text{train}} = 1000$.

We solve Eq. (35) using the MODFLOW 6 (MF6) software [30,31] with the Robin BC implemented using its general head boundary stress package, and the river is modeled with the streamflow routing stress package. Recharge and the pumping wells are modeled using the MF6 recharge and well stress packages, respectively. The Freyberg problem is run for a total of 12 years using 25 time steps of varying length. The first time step has a length of 10 years, and each of the remaining 24 time steps has a length of 1 month. Recharge rates are varied monthly in the last two years of the simulation. The well pumping rates are varied annually in the last two years of the simulation.

The reference hydraulic conductivity field $K_{\text{ref}}(\boldsymbol{x})$ and $\{K^{(i)}(\boldsymbol{x})\}_{i=1}^{N_{\text{train}}}$ for the training dataset are generated according to the algorithm described in [32,33], where an a priori $K(\boldsymbol{x})$ field is modified with multipliers, first at the grid scale (each cell) and then at the domain scale (a single scalar). We start from a spatially uniform *prior* $K(\boldsymbol{x}) = 11.1$ m/day. The grid-scale multipliers are realizations of a normal distribution with mean one and standard deviation 1.2 m/day sampled in log space (mean zero and standard deviation 0.1823). The standard deviation is determined by a range of values for the multiplier, in our case $[0.2, 5.0]$. This range is also used to truncate the resulting samples. The grid-scale multipliers are given a spatial correlation structure using an exponential variogram with a correlation length of 1000 m and a variance of one. Next, at the domain scale, a scalar multiplier is applied to adjust the mean of the a priori K. This scalar multiplier is sampled from the same normal distribution described above for the grid-scale parameters. Once the product of the multipliers is calculated, a second truncation is made, this time to the bounds of $[0.01, 100]$ m/day.

Next, we use MF6 and $K_{\text{ref}}(\boldsymbol{x})$ and $\{K^{(i)}(\boldsymbol{x})\}_{i=1}^{N_{\text{train}}}$ as input parameter fields to compute the solutions of the PDE (35), $u_{\text{ref}}(\boldsymbol{x}, t)$ and $\{u^{(i)}(\boldsymbol{x}, t)\}_{i=1}^{N_{\text{train}}}$. The samples $\{y^{(i)} = \ln K^{(i)}(\boldsymbol{x})\}_{i=1}^{N_{\text{train}}}$ and $\{u^{(i)}(\boldsymbol{x}, t)\}_{i=1}^{N_{\text{train}}}$ are used to construct the KLEs of $y(\boldsymbol{x}) = \ln K(\boldsymbol{x})$ and $u(\boldsymbol{x}, t)$

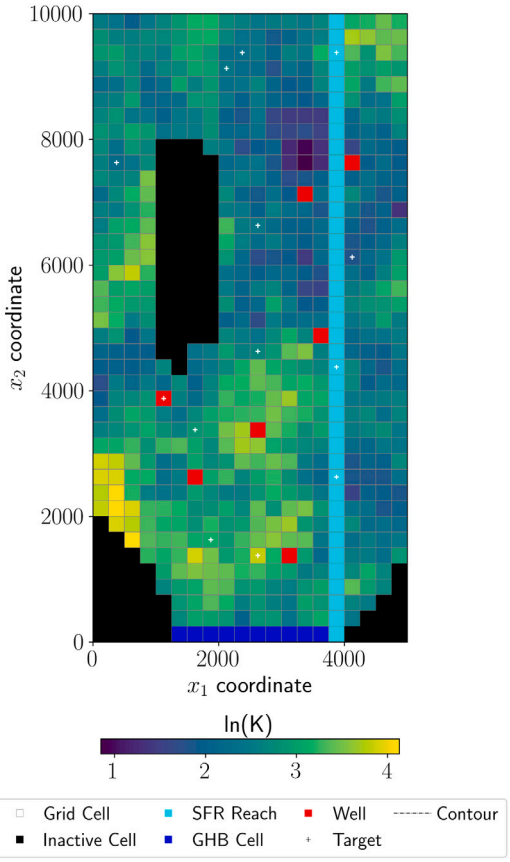

Fig. 10. The computational domain of the hypothetical "Freyberg" aquifer as implemented in MODFLOW 6. The color map shows the reference hydraulic conductivity field $K(x)$ overlaid by static domain features: inactive or no-flow cells, the stream reach, general head boundary (GHB) cells, pumping wells, and target or observation wells. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

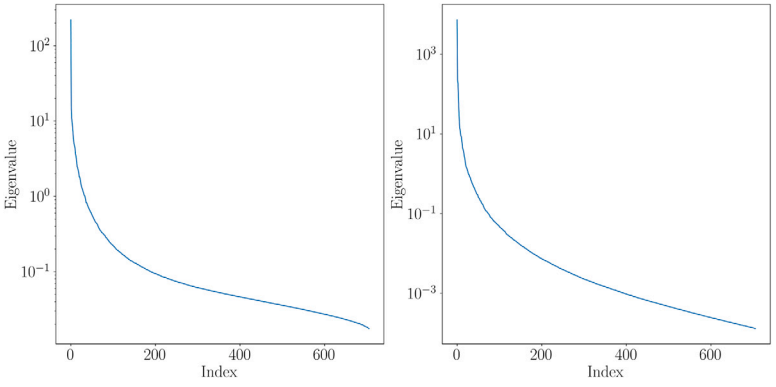

Fig. 11. The decay of the $y(\boldsymbol{x})$ (left) and $u(\boldsymbol{x},t)$ (right) eigenvalues. In the simulations, the KLE of y is truncated at $N_\xi = 150$ (rtol = 0.069) and the KLE of u is truncated at $N_\eta = 90$ (rtol = 0.00045).

and to train the DNNs $\{\mathcal{NN}_\eta(\xi;\theta_{\mathrm{r}}^{(i)})\}_{i=1}^{N_{\mathrm{ens}}}$ and $\{\mathcal{NN}_\eta(\xi;\theta^{(i)})\}_{i=1}^{N_{\mathrm{ens}}}$ using the rKL-DNN and DE-KL-DNN methods, respectively. We use $N_{\mathrm{train}} = 5000$ samples in the training dataset. In the DE-KL-DNN and rKL-DNN methods, we set $N_\xi = 150$, $N_\eta = 90$, $N_{\mathrm{ens}} = 100$, $\lambda_{\mathrm{reg}} = 10^{-5}$, $\sigma_{\boldsymbol{\alpha}}^2 = 10^{-8}$, and $\sigma_{\boldsymbol{\beta}}^2 = 10^{-3}$. *blue* Fig. 11 shows the decay of the y and u eigenvalues. For both fields, we observe the *fast* and *slow* decay regimes. We select N_ξ and N_η to coincide with the transition points from the slow to fast regimes. The corresponding rtol values are 0.069 for the y field and 0.00045 for the u field.

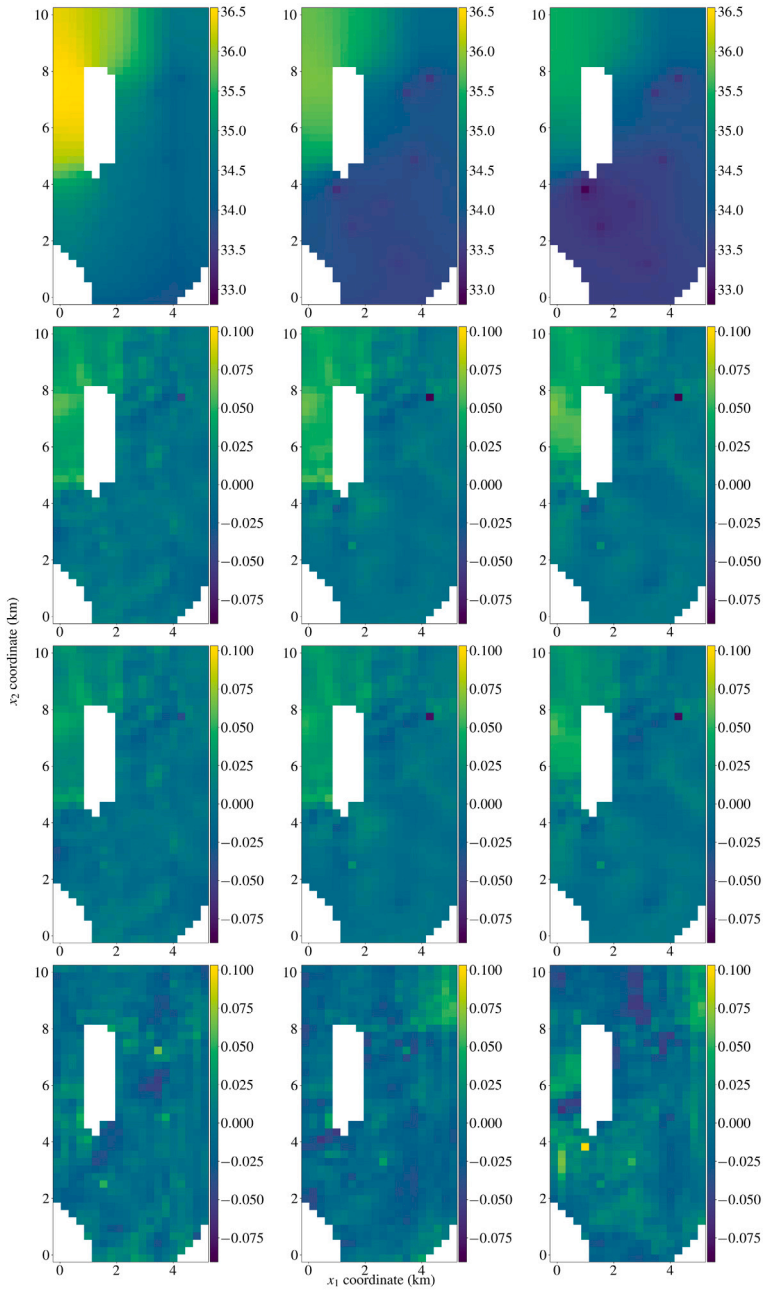

Fig. 12. The reference hydraulic head field u at times $t_1 = 10$, $t_2 = 11$, and $t_3 = 12$ years (first row). The error in the mean u prediction obtained from DE-KL-DNN (second row) and rKL-DNN (third row) and u estimated from FNO (fourth row) with respect to the reference conductivity field.

Fig. 12 shows the reference hydraulic head field u_{ref} at times $t_1 = 10$, $t_2 = 11$, and $t_3 = 12$ years and the point errors in the $u(\boldsymbol{x}, t)$ prediction obtained with the DE-KL-DNN and rKL-DNN methods. The $u(\boldsymbol{x}, t)$ predictions in these methods are given by $\bar{u}_{\text{DE}}(\boldsymbol{x}, t)$ and $\bar{u}_{\text{rKL-DNN}}(\boldsymbol{x}, t)$, respectively.

For comparison, we use the same dataset $\{y^{(i)}(\boldsymbol{x}), u^{(i)}(\boldsymbol{x}, t)\}_{i=1}^{N_{\text{train}}}$ to train a FNO surrogate model [1]. The point errors in the FNO-predicted hydraulic head with respect to the reference solution are also shown in Fig. 12. In the FNO model, we use a fully connected neural network to lift the input tensor with four channels to a desired high-dimensional channel space. Here, the four channels of the input tensor contain the $y(x_1, x_2, t)$ values at N_m points in the space–time domain and the x_1, x_2, and t coordinates where u is predicted. We set the width (the dimensionality) of the channel space to 32 (we find that the higher dimensionality significantly

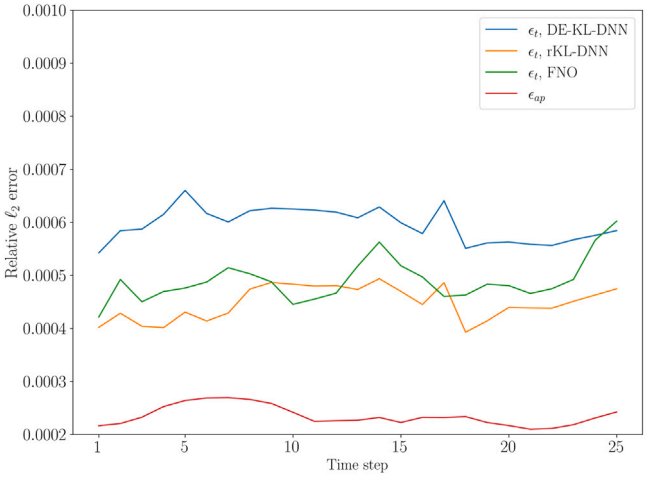

Fig. 13. The ϵ_t errors in $u(\boldsymbol{x}, t)$ predictions obtained from the DE-KL-DNN, rKL-DNN, and FNO methods at different time steps. The KLE approximation error $\epsilon_{ap}(t)$ is also shown.

increases the training time without reducing the testing error). In each Fourier layer, we filter out Fourier modes greater than 8 (the largest number of nodes available in the x_1 direction). We set the training epoch to 1000 with a batch size of 32 and employ the Adam optimizer with a step learning rate (LR) scheduler that decays the initial learning rate of 1×10^{-3} by half every 100 epochs. In the training dataset, the values of y and u in the inactive cells of the domain are set to -1. During the training phase, we create a mask to filter out these cells so they do not contribute to the loss computation.

The rKL-DNN, DE-KL-DNN, and FNO methods produce point errors in approximately the same range, even though the distribution of the point errors is different. The ϵ_t errors in the DE-KL-DNN, rKL-DNN, and FNO methods are 5.968×10^{-4}, 4.486×10^{-4}, and 4.914×10^{-4}, respectively.

Fig. 13 shows the $\epsilon_t(t)$ errors in these three methods and the KLE approximation error as functions of time. DE-KL-DNN has the highest ϵ_t error except for the late time when the FNO error becomes largest. The rKL-DNN error is the smallest for most times. We also observe that ϵ_{ap} is 2–3 times smaller than the ϵ_t errors in the three methods, indicating that the KLE representation of the solution is not the main source of the DE-KL-DNN and rKL-DNN errors. These results indicate that for the considered application, rKL-DNN slightly outperforms FNO (e.g., ϵ_t in rKL-DNN is approximately 10% smaller than in FNO). An additional advantage of the rKL-DNN method is that it has much fewer input parameters than the FNO method. As a result, the computational time for training $N_{ens} = 100$ DNNs in rKL-DNN (time for training one DNN is 40 s, the total time is 1.1 h) is significantly smaller than the time for training the FNO model (16 h for training a single FNO network).

Fig. 14 shows the standard deviations in the DE-KL-DNN and rKL-DNN estimates of $u_{ref}(\boldsymbol{x}, t)$. We find a similar pattern in the distribution of the standard deviations in both methods. For a given time, larger values of the standard deviations correspond to larger values of $u_{ref}(\boldsymbol{x}, t)$. However, the standard deviation in rKL-DNN is about 100% larger than in DE-KL-DNN.

Fig. 15 shows the coverage of $u_{ref}(\boldsymbol{x}, t)$ by the confidence intervals predicted by the DE-KL-DNN and rKL-DNN methods. The percentage of coverage in DE-KL-DNN is 80% at time t_1, 83% at time t_2, and 74% at time t_3. In rKL-DNN, the percent of coverage is higher—93%, 95%, and 91% at times t_1, t_2, and t_3, respectively. LPP defined in Eq. (28) is another measure of the informativeness of the parameter distribution (a higher LPP corresponds to a more informative distribution). We find the LPP of the DE-KL-DNN solution is 37 753 versus 48 079 in the rKL-DNN solution, indicating that rKL-DNN provides a more informative posterior than the DE-KL-DNN method.

Fig. 16 depicts the marginal and bivariate distributions of $\eta_1, \eta_2, \ldots, \eta_5$, which are used to train the rKL-DNN and DEns models. As in the ADE problem, these distributions are non-Gaussian, further demonstrating that the Gaussianity assumption is not required for constricting the KL-DNN surrogate models.

6. Discussion and conclusions

We proposed a reduced-order DNN surrogate model for learning the solution $u(\boldsymbol{x}, t)$ of a time-dependent PDE as a function of the parameter vector $\boldsymbol{p}$ and/or parameter field $y(\boldsymbol{x})$. The dimension reduction is achieved by using a space–time KLE of $u(\boldsymbol{x}, t)$ (with the latent parameter vector $\boldsymbol{\eta}$) and a spatial KLE of $y(\boldsymbol{x})$ (with the latent parameter vector $\boldsymbol{\xi}$). Then, the PDE solution is approximated using a DNN map from the space $(\boldsymbol{p}, \boldsymbol{\xi})$ to $\boldsymbol{\eta}$. We termed this method the KL-DNN method.

There are multiple sources of uncertainty in the KL-DNN model of $u(\boldsymbol{x}, t)$. We propose two methods for quantifying uncertainty in the KL-DNN surrogate model. The first approach is based on DE, which is often used to quantify uncertainty in DNN methods. In

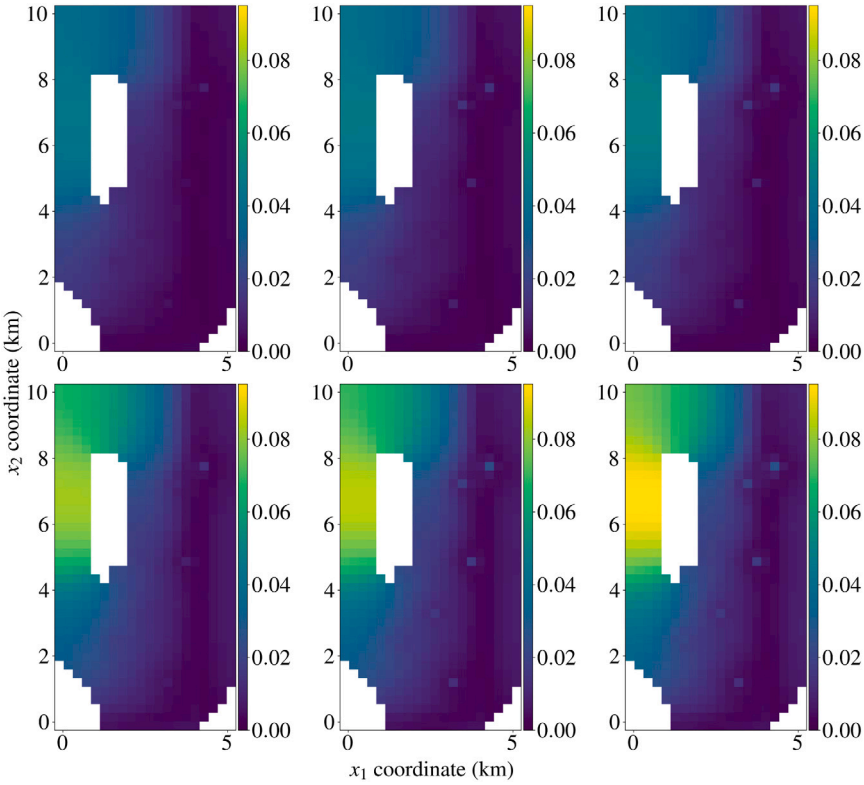

Fig. 14. The standard deviations in the $u_{\text{ref}}(\boldsymbol{x}, t)$ estimates as functions of $\boldsymbol{x}$ and t obtained from DE-KL-DNN and rKL-DNN at times $t_1 = 10$, $t_2 = 11$, and $t_3 = 12$ years.

the Bayesian framework, DE samples provide coordinates of the local modes in the posterior distribution of the DNN coefficients. In the resulting DE-KL-DNN model, the model prediction and associated uncertainty are given by the mean and variance of the DE samples.

In the second approach, which we term the randomized KL-DNN or rKL-DNN method, we approximately sample the posterior Bayesian distribution of the DNN parameters from the randomized loss function. In this approach, the prediction and the associated uncertainty are given by the mean and variance of the posterior distribution. This is the advantage of the rKL-DNN approach over the DE-KL-DNN approach because the mean and variance of the modes' coordinates do not have a clear statistical meaning because they are different from the posterior distribution's mean and variance.

We applied the rKL-DNN and DE-KL-DNN methods for building the surrogate models of the one-dimensional linear ADE and two-dimensional nonlinear diffusion equation with a space-dependent diffusion coefficient. The diffusion PDE model describes time-varying horizontal flow in an unconfined aquifer. For the linear ADE model, we found that the DE-KL-DNN and rKL-DNN predictions have similar errors, but the rKL-DNN method provides better coverage of the reference solution than the DE-KL-DNN method. Our comparison with the physics-informed ML (dPICKLE) method revealed that the KL-DNN surrogate model has higher accuracy if the training dataset is sufficiently large.

For the nonlinear diffusion equation, our results showed that the rKL-DNN method has both a smaller prediction error and better coverage than the DE-KL-DNN method. For both problems, we found that DE-KL-DNN predicts a smaller variance than the rKL-DNN model, which in combination with the poorer coverage indicates that the DE-KL-DNN model underestimates uncertainty.

We found the proposed methods are stable for the considered application because the errors and variances do not increase significantly with time if the training datasets are significantly large.

Finally, for the nonlinear diffusion equation problem, we compared the rKL-DNN method with FNO, a state-of-the-art ML method for constructing surrogate models. We found that the solution given by the posterior mean of the rKL-DNN method is approximately 10 percent more accurate than the FNO solution. Other advantages of the rKL-DNN method over FNO include the uncertainty bounds and a significantly smaller training time.

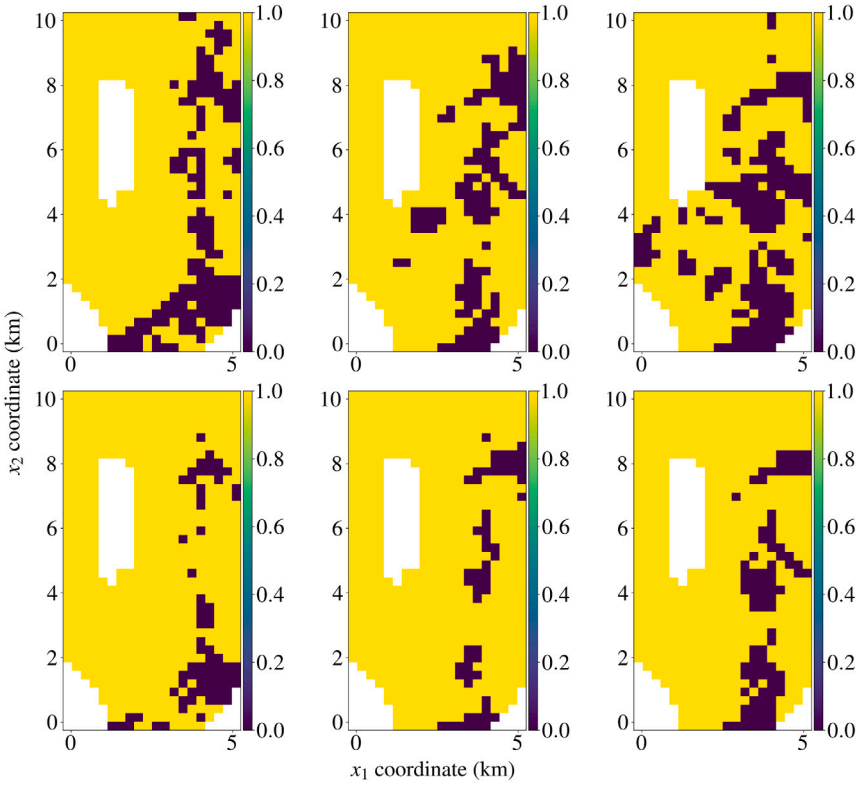

Fig. 15. Coverage of $u_{\text{ref}}(\boldsymbol{x}, t)$ by the confidence intervals (mean plus/minus two standard deviations) computed from DE-KL-DNN (top row) and rKL-DNN (bottom row) at times $t_1 = 10$, $t_2 = 11$, and $t_3 = 12$ years. In the coverage plot, the yellow and blue colors denote areas where the reference solution is inside and outside the confidence interval, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

CRediT authorship contribution statement

Yuanzhe Wang: Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Data curation. **Yifei Zong:** Writing – original draft, Validation, Software. **James L. McCreight:** Writing – original draft, Supervision, Software, Methodology, Investigation, Conceptualization. **Joseph D. Hughes:** Writing – original draft, Software, Methodology, Funding acquisition. **Alexandre M. Tartakovsky:** Writing – original draft, Validation, Supervision, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgments

This material is based, in part, upon work supported by the U.S. Geological Survey, United States under Grant No. G22AP00361. Wang was partially supported by the U.S. Geological Survey, United States. Y. Zong was supported by the U.S. Department of Energy (DOE) Advanced Scientific Computing Research program. A.M. Tartakovsky was partially supported by the DOE project "Science-Informed Machine Learning to Accelerate Real-time (SMART) Decisions in Subsurface Applications Phase 2 – Development and Field Validation," and the United States National Science Foundation. Pacific Northwest National Laboratory is operated by Battelle for the DOE under Contract DE-AC05-76RL01830. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the opinions or policies of the U.S. Geological Survey. Mention of trade names or commercial products does not constitute their endorsement by the U.S. Geological Survey.

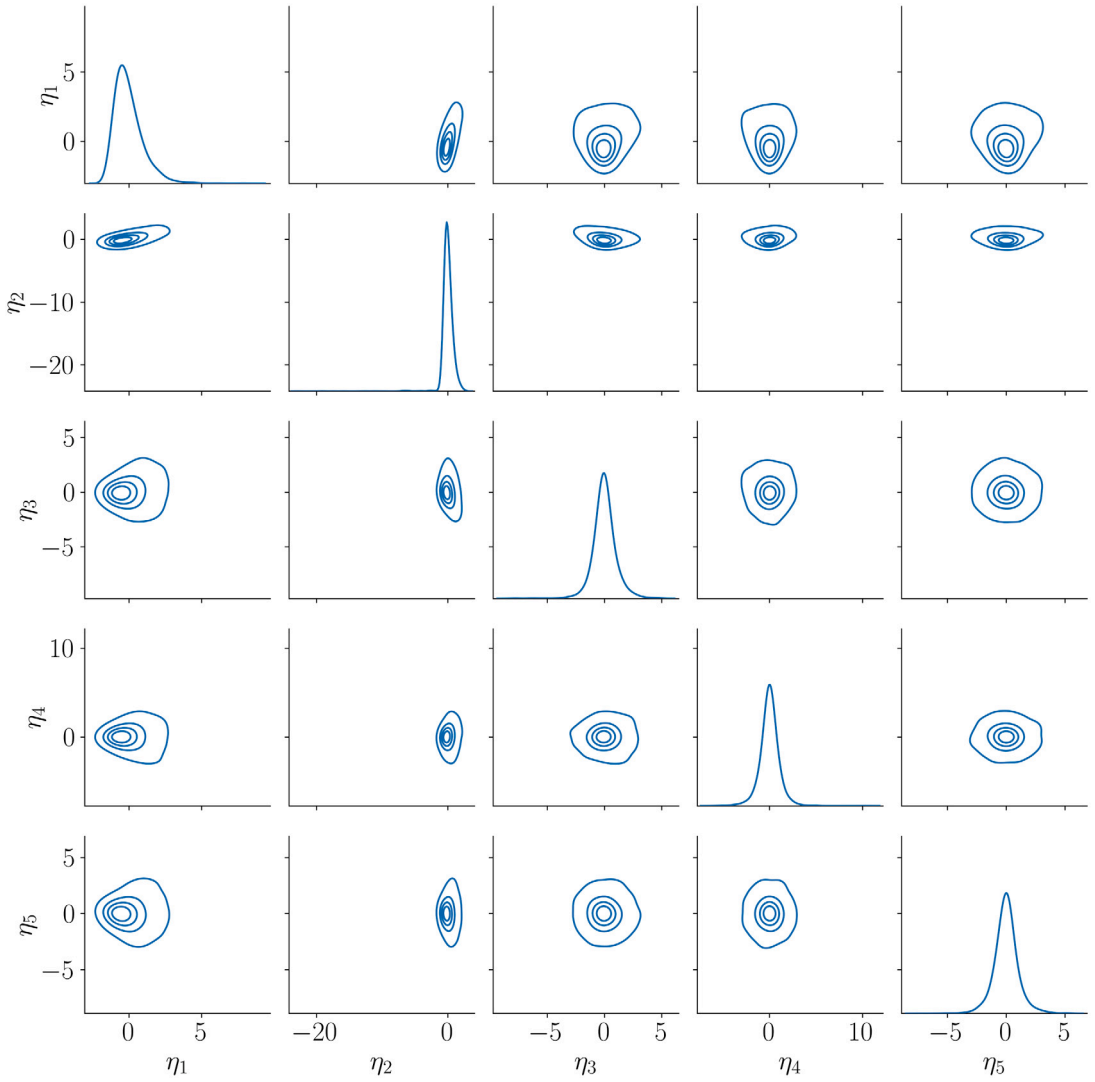

Fig. 16. The marginal and bivariate joint distributions $\eta_1, \eta_2, \ldots, \eta_5$ estimated from the nonlinear diffusion equation training dataset with $N_{\text{train}} = 5000$.

References

[1] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, 2020, arXiv preprint arXiv:2010.08895.

[2] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, Nature machine intelligence 3 (3) (2021) 218–229.

[3] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Graph kernel network for partial differential equations, 2020, arXiv preprint arXiv:2003.03485.

[4] K. Bhattacharya, B. Hosseini, N.B. Kovachki, A.M. Stuart, Model reduction and neural networks for parametric pdes, SMAI j. Comput. Math. 7 (2021) 121–157.

[5] M. Tang, Y. Liu, L.J. Durlofsky, A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems, J. Comput. Phys. 413 (2020) 109456.

[6] I. Langmore, M. Dikovsky, S. Geraedts, P. Norgaard, R. Von Behren, Hamiltonian monte carlo in inverse problems ill-conditioning and multimodality, Int. J. Uncertain. Quantif. 13 (1) (2023).

[7] R.M. Neal, Mcmc using hamiltonian dynamics, Handbook of Markov Chain Monte Carlo 2 (11) (2011) 2.

[8] M. Betancourt, A conceptual introduction to hamiltonian monte carlo, 2017, arXiv preprint arXiv:1701.02434.

[9] A.F. Psaros, X. Meng, Z. Zou, L. Guo, G.E. Karniadakis, Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons, J. Comput. Phys. 477 (2023) 111902.

[10] M.D. Parno, Y.M. Marzouk, Transport map accelerated markov chain monte carlo, SIAM/ASA J. Uncertain. Quant. 6 (2) (2018) 645–682.

[11] E. Klinger, D. Rickert, J. Hasenauer, Pyabc: distributed, likelihood-free inference, Bioinformatics 34 (20) (2018) 3591–3593.

[12] M. Grigoriu, Models for space–time random functions, Probab. Eng. Mech. 43 (2016) 5–19, http://dx.doi.org/10.1016/j.probengmech.2015.11.004, https://www.sciencedirect.com/science/article/pii/S0266892015300618.

[13] A.M. Tartakovsky, Y. Zong, Physics-informed machine learning method with space–time karhunen-loève expansions for forward and inverse partial differential equations, J. Comput. Phys. (2023) 112723.
[14] M. Loève, Elementary Probability Theory, Springer, New York, NY, 1977, pp. 1–52.
[15] Z. Zheng, H. Dai, Simulation of multi-dimensional random fields by karhunen–loève expansion, Comput. Methods Appl. Mech. Engrg. 324 (2017) 221–247.
[16] K. Wang, T. Bui-Thanh, O. Ghattas, A randomized maximum a posteriori method for posterior sampling of high dimensional nonlinear bayesian inverse problems, SIAM J. Sci. Comput. 40 (1) (2018) A142–A171.
[17] Y. Chen, D.S. Oliver, Ensemble randomized maximum likelihood method as an iterative ensemble smoother, Math. Geosci. 44 (2012) 1–26.
[18] J.M. Bardsley, A. Solonen, H. Haario, M. Laine, Randomize-then-optimize: A method for sampling from posterior distributions in nonlinear inverse problems, SIAM J. Sci. Comput. 36 (4) (2014) A1895–A1910.
[19] Y. Zong, D. Barajas-Solano, A.M. Tartakovsky, Randomized physics-informed machine learning for uncertainty quantification in high-dimensional inverse problems, 2023, arXiv:2312.06177.
[20] A. Prakash, Y.J. Zhang, Projection-based reduced order modeling and data-driven artificial viscosity closures for incompressible fluid flows, Comput. Methods Appl. Mech. Engrg. 425 (2024) 116930, http://dx.doi.org/10.1016/j.cma.2024.116930, https://www.sciencedirect.com/science/article/pii/S0045782524001865.
[21] M. Rathinam, L.R. Petzold, A new look at proper orthogonal decomposition, SIAM J. Numer. Anal. 41 (5) (2003) 1893–1925.
[22] D. Xiu, G.E. Karniadakis, Modeling uncertainty in flow simulations via generalized polynomial chaos, J. Comput. Phys. 187 (1) (2003) 137–167.
[23] G. Lin, A.M. Tartakovsky, Numerical studies of three-dimensional stochastic darcy's equation and stochastic advection-diffusion-dispersion equation, J. Sci. Comput. 43 (1) (2010) 92–117.
[24] D.M. Tartakovsky, S.P. Neuman, Transient flow in bounded randomly heterogeneous domains: 1.exact conditional moment equations and recursive approximations, Water Resour. Res. 34 (1) (1998) 1–12.
[25] D.M. Tartakovsky, Z. Lu, A. Guadagnini, A.M. Tartakovsky, Unsaturated flow in heterogeneous soils with spatially distributed uncertain hydraulic parameters, J. Hydrol. 275 (3) (2003) 182–193, http://dx.doi.org/10.1016/S0022-1694(03)00042-8, studies on Water Movement and Solute Transport in Arid Regions https://www.sciencedirect.com/science/article/pii/S0022169403000428.
[26] C.E. Rasmussen, C.K. Williams, et al., Gaussian Processes for Machine Learning, vol. 1, Springer, 2006.
[27] A. Ogata, R.B. Banks, A Solution of the Differential Equation of Longitudinal Dispersion in Porous Media, US Government Printing Office, 1961.
[28] D.L. Freyberg, An exercise in ground-water model calibration and prediction, Groundwater 26 (3) (1988) 350–360.
[29] R.J. Hunt, M.N. Fienen, J.T. White, Revisiting an exercise in groundwater model calibration and prediction after 30 years: Insights and new directions, Groundwater 58 (2) (2020) 168–182.
[30] C.D. Langevin, J.D. Hughes, A.M. Provost, M.J. Russcher, S. Panday, MODFLOW as a configurable multi-model hydrologic simulator, Groundwater 62 (1) (2024) 111–123.
[31] C.D. Langevin, J.D. Hughes, E.R. Banta, R.G. Niswonger, S. Panday, A.M. Provost, Documentation for the MODFLOW 6 Groundwater Flow Model, 6, U.S. Geological Survey Techniques and Methods, 2017, chap. A55.
[32] S.A. McKenna, A. Akhriev, D. Echeverría Ciaurri, S. Zhuk, Efficient uncertainty quantification of reservoir properties for parameter estimation and production forecasting, Math. Geosci. 52 (2) (2020) 233–251.
[33] J.T. White, L.K. Foster, M.N. Fienen, M.J. Knowling, B. Hemmings, J.R. Winterle, Toward reproducible environmental modeling for decision support: A worked example, Front. Earth Sci. 8 (2020) 50.