Journal of Computational Physics 428 (2021) 109984

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Deep coregionalization for the emulation of simulation-based spatial-temporal fields

Wei W. Xing [a,b], Robert M. Kirby [b,c], Shandian Zhe [c,*]

[a] School of Integrated Circuit Science and Engineering, Beihang University, China
[b] Scientific Computing and Imaging Institute, University of Utah, United States of America
[c] School of Computing, University of Utah, United States of America

ARTICLE INFO

Article history:
Received 18 February 2020
Received in revised form 3 August 2020
Accepted 1 November 2020
Available online 13 November 2020

Keywords:
Surrogate model
Gaussian process
Emulation
Spatial-temporal field
Multifidelity model

ABSTRACT

Data-driven surrogate models are widely used for applications such as design optimization and uncertainty quantification, where repeated evaluations of an expensive simulator are required. For most partial differential equation (PDE) simulations, the outputs of interest are often spatial or spatial-temporal fields, leading to very high-dimensional outputs. Despite the success of existing data-driven surrogates for high-dimensional outputs, most methods require a significant number of samples to cover the response surface in order to achieve a reasonable degree of accuracy. This demand makes the idea of surrogate models less attractive considering the high-computational cost to generate the data. To address this issue, we exploit the multifidelity nature of a PDE simulation and introduce *deep coregionalization*, a Bayesian nonparametric autoregressive framework for efficient emulation of spatial-temporal fields. To effectively extract the output correlations in the context of multifidelity data, we develop a novel dimension reduction technique, residual principal component analysis. Our model can simultaneously capture the rich output correlations and the fidelity correlations and make high-fidelity predictions with only a small number of expensive, high-fidelity simulation samples. We show the advantages of our model in three canonical PDE models and a fluid dynamics problem. The results show that the proposed method can not only approximate simulation results with significantly less cost (by bout 10%-25%) but also further improve model accuracy.

1. Introduction

Applications such as uncertainty quantification, design optimization, and inverse parameter estimation demand repeated simulations of partial differential equations (PDEs) in an input parameter space [1,2]. Due to the high-computational cost of a simulation, a data-driven surrogate model, also known as an emulator, is often employed in place of the simulation [3]. In practice, simulations of PDEs generally produce large spatial or spatial-temporal fields (e.g., velocity, temperature or electric fields). Due to their large size, modeling the field results poses a huge challenge in terms of model capacity and scalability for the surrogate model [4–6].

* Corresponding author at: Room 3466, MEB building, University of Utah, Salt Lake City, UT 84112, United States of America.
E-mail address: zhe@cs.utah.edu (S. Zhe).

https://doi.org/10.1016/j.jcp.2020.109984
0021-9991/

Gaussian process (GP) modeling is one of the most commonly used data-driven surrogates due to its capability to (1) quantify model uncertainty, (2) adopt prior knowledge, and (3) avoid overfitting for small datasets [3,7–11]. Unfortunately, GPs do not naturally extend to multiple output scenarios, especially the high-dimensional simulation-based spatial-temporal fields. Modeling the high-dimensional outputs entails efficiently learning the output correlations. A naive approach is to simply treat the output indexes (indicating the output value at a particular location of the spatial-temporal domain) as additional input parameters [12]. This approach is infeasible for high-dimensional outputs since the computational complexity for a general GP is $\mathcal{O}(N^3d^3)$, where N is the number of training samples and d is the output dimensionality [13]. If we assume a separable structure between the input correlations and the output correlations, the high-computational issues can be efficiently circumvented. Such a relaxation gives rise to the linear model of coregionalization (LMC), the classical framework for multioutput regression [14,15]. LMC linearly combines base functions with latent processes to model the high-dimensional output. Many modern multioutput GP models can be considered as special instances or variants of LMC [16–19]. Conti and O'Hagan [20] proposed a simplified LMC model, namely, the intrinsic coregionalization model (ICM) [21], which assumes a single latent process with a correlation matrix to govern the output correlations. To enhance the flexibility of ICM, Higdon et al. [4] found a set of bases from singular value decomposition (SVD) on the training outputs to encode the complicated spatial-temporal correlations for PDE simulation problems. This approach is further extended by using nonlinear dimension reduction techniques, such as kernel PCA [6] and Isomap [5] for nonlinear correlations. Recent developments of GP further introduce latent coordinate features and tensor algebra to improve the model flexibility and scalability [19,22]. Other methods that relax the separable assumption include process convolution [23–26] and deep learning hybrid GPs [27,28]. These methods, however, either are nonscalable to very high-dimensional outputs or require massive tuning. They are thus not suitable for emulations of simulation-based spatial-temporal fields.

Despite the success of LMC-based methods for emulations of spatial-temporal fields, in order to achieve a reasonable degree of accuracy, they require a large number of samples (corresponding to different inputs) that adequately cover the response surface. We will further show that the LMC model expressiveness is indeed limited by the number of training samples. As the number of required training samples grows drastically for a highly nonlinear response surface, the idea of data-driven surrogate modeling becomes less practical.

From the data-driven surrogate perspective, an efficient way to relax the high demand of training data is to inject prior knowledge of the PDE models into surrogate models, e.g., physics informed neural networks [29], conservation kernels [30], and hybrid physics-based data-driven surrogates [31]. Despite their success, these approaches cannot generalize effectively to the emulation of simulation-based spatial-temporal fields. Neural-networks-based methods require massive model tunning and are prone to overfitting; conservation kernels apply only to specific types of PDEs; hybrid physics methods require modification of original simulation codes as an intrusive method. Another more common solution for the emulation is to take advantage of the multifidelity nature of the simulation [32–35] and fuse information from different fidelities. More specifically, when generating training data for the data-driven surrogate models, we can easily reduce the computational cost by using a low-fidelity simulator (e.g., a simulator with a sparse discretization mesh) at the price of getting less accurate samples. The low-fidelity samples, despite being noisy and biased, normally show a strong correlation with the high-fidelity samples. Thus, the low-fidelity samples can be used in numerous ways to accelerate many science and engineering processes [32]. In this paper, we focus on how to harness the fidelity correlations such that we can approximate the simulation output with many affordable low-fidelity samples and transfer the knowledge to predict highly accurate results without full reliance on expensive high-fidelity data. To this end, we propose deep coregionalization, an efficient data-driven surrogate for emulations of very high-dimensional spatial-temporal fields. Our model simultaneously captures the correlation in the spatial-temporal fields and those between different fidelity observations to provide accurate high-fidelity predictions. Our contributions are as follows:

- We prove that under a mild assumption of the kernel structure, the propagation of fidelity knowledge of spatial-temporal fields can be done efficiently through a univariate GP autoregressive model of latent processes.
- To improve the model capacity for complex problems, we introduce deep coregionalization, which generalizes the GP autoregressive model for spatial-temporal fields by integrating the classic linear model of coregionalization (LMC).
- To efficiently capture the output correlations of different fidelities without introducing an overly-complicated model with excessive parameters, we propose residual principal component analysis (ResPCA), a dimension reduction technique that can effectively encode the rich correlations of high-dimensional fields, efficiently preserve a compact representation for multifidelity fields, and easily scale to high-dimensional field outputs (e.g., 1 million d.o.f.s).
- We validate the proposed method on three canonical PDEs (Poisson's equation, Burger's equation, and the heat equation) and a fluid dynamics problem. The results show a significant improvement not only in reducing the simulation budget (for generating the training data) but also in generating accurate, high-fidelity results, compared with models based on single-fidelity data.

The rest of this paper is organized as follows: Section 2 makes clear the problem of the emulation of spatial-temporal fields. Section 3 firstly introduces the background, univariate GP, and multifidelity univariate nonlinear autoregression. It then generalizes the nonlinear autoregression by assuming a separable kernel function to introduce a intrinsic deep coregionalization model. Finally, it proposes the general deep coregionalization framework and introduces a novel ResPCA algorithm to fulfill an efficient model training implementation. Section 4 demonstrates the superiority of the proposed

method compared with the state-of-the-art high-dimensional GP emulators on three canonical PDEs—Poisson's equation, Burger's equation, and the heat equation—and a fluid dynamics problem, the lid-driven cavity. Section 5 concludes this paper with discussions on deep coregionalization's connections to the existing models and further improvements.

2. Statement of the problem

Consider a general parameterized nonlinear system of steady-state or transient PDEs of arbitrary order for the dependent scalar variable $u(\boldsymbol{x}, t, \boldsymbol{\xi})$,

$$\begin{cases} \frac{\partial}{\partial t} u(\boldsymbol{x}, t, \boldsymbol{\xi}) + \mathcal{F}(u(\boldsymbol{x}, t, \boldsymbol{\xi})) = \mathcal{S}(\boldsymbol{x}, t, \boldsymbol{\xi}), & \Omega \times (0, T] \times \mathcal{X}, \\ \mathcal{B}(u, \boldsymbol{\xi}) = 0, & \partial\Omega \times (0, T] \times \mathcal{X}, \\ u(\boldsymbol{x}, 0, \boldsymbol{\xi}) = u_0(\boldsymbol{x}, \boldsymbol{\xi}), & \Omega \times \{t = 0\} \times \mathcal{X}, \end{cases} \tag{1}$$

where $\mathcal{F}$ is a nonlinear operator that may contain parameters indicated by $\boldsymbol{\xi} \in \mathcal{X} \subset \mathbb{R}^l$. $\mathcal{S}$ is the source term/function, Ω and T are the spatial and temporal domain of interest, $\mathcal{B}$ is the boundary condition operator also parameterized via $\boldsymbol{\xi}$, and $u_0(\boldsymbol{x}, \boldsymbol{\xi})$ parameterizes the initial condition. The variable $\boldsymbol{x} \in \Omega \subset \mathbb{R}^p$ is the spatial coordinate of an p-dimensional space and $t \in [0, T]$ parameterizes the temporal domain. The PDEs can be fully nonlinear and parameterized in an arbitrary fashion (including the initial and boundary conditions); they are also assumed to be well posed (i.e., solutions always exist and are unique) for Ω, T, and $\mathcal{X}$ being considered. For a given parameter $\boldsymbol{\xi}$, the system is solved for $u(\boldsymbol{x}, t) \in \Omega \times (0, T]$ using a computationally expensive numerical solver. For applications such as uncertainty quantification (where $\boldsymbol{\xi}$ encodes the randomness) and design optimization ($\boldsymbol{\xi}$ encodes the design parameters), a large number of computations are required; the direct implementation of the numerical solver can be computationally prohibitive. To resolve this issue, we propose a data-driven surrogate to efficiently generate $u(\boldsymbol{x}, t)$ given an unseen parameter $\boldsymbol{\xi}$ in the range of interest for the spatial-temporal domain of $\Omega \times (0, T]$. A direct approximation of this function is difficult due to the number of samples we need to cover the response surface for such a high-dimensional input space problem [4]. Following the pioneer work of Higdon et al. [4], we can record values at specified (fixed) spatial locations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_x}$ and temporal locations $t_1, \ldots, t_{N_t}$ to provide a discrete approximation of $u(\boldsymbol{x}, t)$. With the recording coordinates, our quantity of interest becomes a vectorial function of the PDE parameters

$$\boldsymbol{y}(\boldsymbol{\xi}) = \left(u(\boldsymbol{x}_1, t_1, \boldsymbol{\xi}), \ldots, u(\boldsymbol{x}_{N_x}, t_1, \boldsymbol{\xi}), u(\boldsymbol{x}_1, t_2, \boldsymbol{\xi}), \ldots, u(\boldsymbol{x}_{N_x}, t_{N_t}, \boldsymbol{\xi})\right)^T \in \mathbb{R}^d,$$

where $\boldsymbol{x}_i$ is a spatial coordinate of a regular/irregular grid, and $d = N_x \times N_t$ is the total number of spatial-temporal grid points. Since the number of spatial and temporal coordinates N_x and N_t needs to be large enough to provide an accurate approximation, the dimension of vector $\boldsymbol{y}(\boldsymbol{\xi})$ becomes extremely large (e.g., 1 million d.o.f.s for a modest-size 2d simulation with a 100×100 spatial grid and 100 time steps). Our challenge becomes how to approximate the mapping $\boldsymbol{y}(\boldsymbol{\xi}) : \mathcal{X} \to \mathbb{R}^d$ with a limited computational budget.

A fine-fidelity model is a numerical solver that solves for $u(\boldsymbol{x}, t, \boldsymbol{\xi})$ such that the solution $u^{(F)}(\boldsymbol{x}, t, \boldsymbol{\xi})$ is an accurate approximation of $u(\boldsymbol{x}, t, \boldsymbol{\xi})$. Depending on the particular problems, different numerical solvers, e.g., finite element and finite volume, can be applied. We do not discuss the particular choice of numerical solver as it is irrelevant to our model. For almost all numerical solvers, the accuracy of the results can be controlled using a different parameter setting for the solvers. For instance, we can apply coarse spatial-temporal meshes, inaccurate time-stepping methods, and simplified physics/computation to generate less accurate, yet computationally cheap, results. On the other hand, we can increase the spatial-temporal mesh coarseness and use a high-order approximation technique to produce accurate but computationally expensive results. Without loss of generality, we use a superscript (f) to denote the fidelity level. Note that the relation between the solver fidelity and its simulation error is not required for our model. The solution vector generated by the fidelity-f solver is then denoted as

$$\boldsymbol{y}^{(f)}(\boldsymbol{\xi}) = \left(u^{(f)}(\boldsymbol{x}_1, t_1, \boldsymbol{\xi}), \ldots, u^{(f)}(\boldsymbol{x}_{N_x}, t_1, \boldsymbol{\xi}), u^{(f)}(\boldsymbol{x}_1, t_2, \boldsymbol{\xi}) \ldots, u^{(f)}(\boldsymbol{x}_{N_x}, t_{N_t}, \boldsymbol{\xi})\right)^T, \tag{2}$$

where $u^{(f)}(\boldsymbol{x}_i, t_j, \boldsymbol{\xi})$ is the detailed solution based on the solver with the fidelity indicated by f. In general, we can run a simulation at different fidelity settings to obtain a multifidelity dataset $\{\boldsymbol{\Xi}^{(f)}, \mathbf{Y}^{(f)}\}_{f=1}^F$, where $\mathbf{Y}^{(f)}$ is a $N_f \times d$ matrix of solutions at fidelity-f, with $\boldsymbol{\Xi}^{(f)}$ being an $N_f \times l$ matrix for the corresponding inputs. We use $f = F$ to denote the highest fidelity and $f = 1$ the lowest. Following the common setting for multifidelity emulation in [34,35], we also assume that the training set of fidelity f is a subset of the previous fidelity $f - 1$, i.e., $\boldsymbol{\Xi}^{(f)} \subset \boldsymbol{\Xi}^{(f-1)}$. For convenience, we introduce the index notation $\mathbf{e}$ to denote the subset index. Particularly, $\boldsymbol{\Xi}^{(f)} = \boldsymbol{\Xi}_{\mathbf{e}_f,:}$, where $\mathbf{e}_f$ indicates the subset indexes for fidelity-f data, $\boldsymbol{\Xi}$ contains all possible candidates and the subscript $\mathbf{e}_{f,:}$ extracts rows indicated by $\mathbf{e}_f$. Our goal in designing a high-fidelity emulator is as follows: given multifidelity data $\{\boldsymbol{\Xi}^{(f)}, \mathbf{Y}^{(f)}\}_{f=1}^F$, how do we efficiently approximate the mapping $\boldsymbol{y}(\boldsymbol{\xi}) : \mathcal{X} \to \mathbb{R}^d$?

3. Model formulation

3.1. Gaussian process emulator

Traditional data-driven surrogates attacked this problem by using data generated from only a few accurate high-fidelity simulations $\mathcal{D}^{(F)} = \{\Xi, \mathbf{Y}^{(F)}\}$ [3,7–9,23,4–6]. For simplicity, consider the quantity of interest being univariate, i.e., $d = 1$. We use $y^{(F)}(\boldsymbol{\xi})$ to distinguish a univariate function from a multivariate one $\boldsymbol{y}^{(F)}(\boldsymbol{\xi})$. A GP assumes that the finite set of function values on $\Xi = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{N_F})]^T$, namely $\mathbf{Y}^{(F)} = [y^{(F)}(\boldsymbol{\xi}_1), \ldots, y^{(F)}(\boldsymbol{\xi}_{N_F})]^T$, follows a multivariate Gaussian distribution, $p(\mathbf{Y}^{(F)} \mid \Xi) = \mathcal{N}\left(\mathbf{Y}^{(F)} \mid \mathbf{m}, \mathbf{K} + \tau^{-1}\mathbf{I}\right)$. Here $\mathbf{m} = [m(\boldsymbol{\xi}_1), \ldots, m(\boldsymbol{\xi}_{N_F})]^T$ is the mean function that is usually set to $\mathbf{0}$ after centering the data collection, $\mathbf{Y}^{(F)}$, and τ is the inverse noise variance that accounts for model inadequacies and numerical error [12,36]. $\mathbf{K}$ is the covariance matrix with elements $[\mathbf{K}]_{ij} = k(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j)$, where $k(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j)$ is a kernel function for the corresponding inputs. Choosing the right kernel function for a specific application is nontrivial. When there is no prior knowledge to guide the choice, the automatic relevance deterrence (ARD) kernel [36],

$$k(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j) = \theta_0 \exp\left(-(\boldsymbol{\xi}_i - \boldsymbol{\xi}_j)\mathrm{diag}(\theta_1, \ldots, \theta_l)(\boldsymbol{\xi}_i - \boldsymbol{\xi}_j)^T\right), \tag{3}$$

is often utilized. The ARD kernel can freely capture the individual influence of each input parameter on the output results. The hyperparameters $\{\tau, \theta_0, \ldots, \theta_l\}$ can be estimated by maximizing the marginal likelihood of the GP regression,

$$\mathcal{L} = \frac{1}{2}\ln|\mathbf{K} + \tau^{-1}\mathbf{I}| - \frac{1}{2}\mathbf{Y}^{(F)^T}(\mathbf{K} + \tau^{-1}\mathbf{I})^{-1}\mathbf{Y}^{(F)} - \frac{N}{2}\ln(2\pi). \tag{4}$$

The main computational cost is the inversion of $\mathbf{K}$, which is $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ for time and space complexity, respectively. We can derive the posterior as a function of $\boldsymbol{\xi}$ using a conditional Gaussian distribution [36],

$$\begin{aligned} p\left(y^{(F)}(\boldsymbol{\xi}) \,\middle|\, \Xi, \mathbf{Y}\right) &= \mathcal{N}\left(y^{(F)}(\boldsymbol{\xi}) \,\middle|\, \mu(\boldsymbol{\xi}), v(\boldsymbol{\xi})\right) \\ \mu(\boldsymbol{\xi}) &= \boldsymbol{k}(\boldsymbol{\xi})^T(\mathbf{K} + \tau^{-1}\mathbf{I})^{-1}\mathbf{Y}^{(F)} \\ v(\boldsymbol{\xi}) &= k(\boldsymbol{\xi}, \boldsymbol{\xi}) - \boldsymbol{k}(\boldsymbol{\xi})^T(\mathbf{K} + \tau^{-1}\mathbf{I})^{-1}\boldsymbol{k}(\boldsymbol{\xi}), \end{aligned} \tag{5}$$

where $\boldsymbol{k}(\boldsymbol{\xi}) = [k(\boldsymbol{\xi}, \boldsymbol{\xi}_1), \ldots, k(\boldsymbol{\xi}, \boldsymbol{\xi}_{N_F})]^\top$ is the vector of covariance between $\boldsymbol{\xi}$ and existing (training) data input Ξ. Note that for simulation data without stochastic factors, the data can be considered noise free and the noise precision can be set to infinite. In this case, it is normal to still include a large τ as a regularization term such that the inversion $(\mathbf{K} + \tau^{-1}\mathbf{I})^{-1}$ is not ill conditioned.

3.2. Deep Gaussian process autoregression

Let us now consider the multifidelity univariate data, i.e., $\{\Xi^{(f)}, \mathbf{Y}^{(f)}\}_{f=1}^{F}$, where $\mathbf{Y}^{(f)} = [y^{(f)}(\boldsymbol{\xi}_1), \ldots, y^{(f)}(\boldsymbol{\xi}_{N_f})]^T$ is a collection of samples at fidelity f. The general autoregressive formulation for multifidelity data fusion is

$$y^{(f)}(\boldsymbol{\xi}) = g^{(f)}\left(y^{(f-1)}(\boldsymbol{\xi})\right) + \epsilon^{(f)}(\boldsymbol{\xi}), \tag{6}$$

where $g^{(f)}\left(y^{(f-1)}(\boldsymbol{\xi})\right)$ is an arbitrary function that maps the low-fidelity results to the high-fidelity ones, and $\epsilon^{(f)}(\boldsymbol{\xi})$ captures the model inadequacies and numerical errors between different fidelities, which is assumed Gaussian distributed. If we assume a simple linear form for the mapping, i.e., $g^{(f)}(y^{(f-1)}(\boldsymbol{\xi})) = c \cdot y^{(f-1)}(\boldsymbol{\xi})$, we recover the classic autoregressive model put forth by Kennedy and O'Hagan [33]. This method was further improved by Le Gratiet [37], who used a deterministic parametric form of $g^{(f)}$ and an efficient numerical scheme to reduce the computational cost. Despite successfully dealing with some demonstrated problems, this parametric approach does not generalize well due to the difficulty of model selection and the demand for large training datasets [34]. To resolve this issue, a Bayesian nonparametric treatment can be implemented by placing a GP prior over the function $g^{(f)}$. This approach is also known as the *deep GP* [38]. Although being capable of modeling complex problems, deep GP is infamous for its intractability. It requires computationally expensive variational approximation for model training and thus quickly becomes infeasible for multifidelity simulation problems. Perdikaris et al. [34] put forward a GP-based nonlinear autoregressive scheme with an additive structure,

$$y^{(f)}(\boldsymbol{\xi}) = g^{(f)}\left(\boldsymbol{\xi}, y_*^{(f-1)}(\boldsymbol{\xi})\right), \tag{7}$$

where the error term $\epsilon^{(f)}(\boldsymbol{\xi})$ is absorbed into $g^{(f)}(\boldsymbol{\xi})$, and $y_*^{(f-1)}(\boldsymbol{\xi})$ is the exact function value for $\boldsymbol{\xi}$ at fidelity $(f-1)$. This formulation specifies that $y^{(f)}(\boldsymbol{\xi})$ is a GP given the previous fidelity observations and the model inputs. To better reflect the autoregressive nature, Perdikaris et al. [34] suggested a covariance function that decomposes as

$$k^{(f)}\left([\boldsymbol{\xi}, y_*^{(f-1)}(\boldsymbol{\xi})], [\boldsymbol{\xi}', y_*^{(f-1)}(\boldsymbol{\xi}')]\right) = k_{\xi}^{(f)}(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}_{f_\xi}) \cdot k_y^{(f)}(y_*^{(f-1)}(\boldsymbol{\xi}), y_*^{(f-1)}(\boldsymbol{\xi}'); \boldsymbol{\theta}_{f_y}), \tag{8}$$

where $k_{\xi}^{(f)}$ and $k_y^{(f)}$ are valid covariance functions, with $\{\boldsymbol{\theta}_{f_\xi}, \boldsymbol{\theta}_{f_y}\}$ denoting the corresponding hyperparameters. These hyperparameters can be learned via maximum-likelihood estimation as in Eq. (4) using input data $\{\boldsymbol{\Xi}^{(f)}, \mathbf{Y}^{(f-1)}_{\mathbf{e}^{(f)}\cap\mathbf{e}^{(f-1)},:}\}$ and output data $\mathbf{Y}^{(f)}$, where $\mathbf{e}^{(f)} \cap \mathbf{e}^{(f-1)}$ indicates the intersection of index for fidelity-f and -$(f-1)$. Thus, unlike the standard deep GP [38], this autoregressive model does not have the intractable issues for estimating the unknown latent variables and hyperparameters. This method requires estimation of $(l+3)$ hyperparameters with an ARD kernel for each fidelity model.

3.3. Deep coregionalization

Despite being much more efficient than single-fidelity models, the GP autoregressive model does not naturally extend to multivariate cases, especially the high-dimensional ones. First, when using a common anisotropic kernel such as the ARD kernel, the deep kernel function $k_y^{(f)}$ in Eq. (8) requires d hyperparameters to be optimized, which is practically impossible for our problem where d can scale up to 1 million. Second, it is unclear how to harness the strong output correlations. If we simply treat each output independently (given the inputs), we risk severe overfitting when learning with a small set of training examples, a common situation for the high-fidelity data.

3.3.1. Deep intrinsic coregionalization

To model the output correlations, we can directly modify the model of Perdikaris et al. [34] by equipping Eq. (8) with a separable output correlation kernel structure as

$$\mathrm{Cov}\left(y_i^{(f)}(\boldsymbol{\xi}), y_j^{(f)}(\boldsymbol{\xi}')\right) = k_d^{(f)}(i,j) \cdot k_{\xi}^{(f)}(\boldsymbol{\xi}, \boldsymbol{\xi}') \cdot k_y^{(f)}(\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi}), \boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi}')), \tag{9}$$

where $k_d^{(f)}(i,j)$ is an arbitrary valid kernel function encoding the output correlation between i-th output $y_i^{(f)}$ and j-th output $y_j^{(f)}$. Because a correlation matrix must be symmetric, the output correlation matrix has a maximum of $d(d+1)/2$ hyperparameters, which poses challenges to directly working with Eq. (9). Fortunately, Eq. (9) has an equivalent form.

Lemma 1. *If a multivariate GP's covariance function can be decomposed as Eq. (9), it must have an equivalent form of*

$$\boldsymbol{y}^{(f)}(\boldsymbol{\xi}) = \boldsymbol{\psi}^{(f)} z^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})\right), \tag{10}$$

where $z^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})\right)$ *is a GP with covariance function* $k_{\xi}^{(f)}(\boldsymbol{\xi}, \boldsymbol{\xi}') \cdot k_y^{(f)}(\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi}), \boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi}'))$ *and* $\boldsymbol{\psi}^{(f)} \in \mathbb{R}^{d\times 1}$ *is a base vector corresponding to the sum of the eigenvectors times the square roots of the eigenvalues of the kernel matrix* $\mathbf{K}_d$*, which has components* $[K]_{ij} = k_d^{(f)}(i,j)$.

We leave the proof in Appendix A for the sake of clarity. Substituting Eq. (10) into Eq. (9), we have

$$\mathrm{Cov}\left(y_i^{(f)}(\boldsymbol{\xi}), y_j^{(f)}(\boldsymbol{\xi}')\right) = k_d^{(f)}(i,j) \cdot k_{\xi}^{(f)}(\boldsymbol{\xi}, \boldsymbol{\xi}') \cdot k_y^{(f)}\left(\boldsymbol{\psi}^{(f-1)} z_*^{(f-1)}(\boldsymbol{\xi}), \boldsymbol{\psi}^{(f-1)} z_*^{(f-1)}(\boldsymbol{\xi}')\right). \tag{11}$$

This model still is difficult to work with because each $\boldsymbol{\psi}^{(f)}$ contains d model parameters to be optimized regardless of the hyperparamter of the kernel function. Note that the mapping $\boldsymbol{\psi}^{(f-1)} z_*^{(f-1)}(\boldsymbol{\xi})$ is a function of $z_*^{(f-1)}(\boldsymbol{\xi})$ provided that $\boldsymbol{\psi}^{(f-1)}$ is fixed, which suggests that $\boldsymbol{\psi}^{(f-1)} z_*^{(f-1)}(\boldsymbol{\xi})$ is linearly embedded in a (low-dimensional) subspace where $z_*^{(f-1)}(\boldsymbol{\xi})$ lives. This particular affine projection structure allows us to find a compact representation for the kernel function $k_y^{(f)}\left(\boldsymbol{\psi}^{(f-1)} z_*^{(f-1)}(\boldsymbol{\xi}), \boldsymbol{\psi}^{(f-1)} z_*^{(f-1)}(\boldsymbol{\xi}')\right)$:

Lemma 2. *If a kernel function* $k_y^{(f)}\left(\boldsymbol{\psi}^{(f-1)} z_*^{(f-1)}(\boldsymbol{\xi}), \boldsymbol{\psi}^{(f-1)} z_*^{(f-1)}(\boldsymbol{\xi}')\right)$ *is stationary, e.g., the ARD kernel, it must have a compact representation* $k_z^{(f)}\left(z_*^{(f-1)}(\boldsymbol{\xi}), \quad z_*^{(f-1)}(\boldsymbol{\xi}')\right)$*, which is also a stationary kernel by absorbing the base vector* $\boldsymbol{\psi}^{(f-1)}$.

Lemma 2, whose proof is given in Appendix B, suggests that the model complexity can be significantly reduced by absorbing the base vector $\boldsymbol{\psi}^{(f)}$ into the kernel function. This simplification makes modeling of very high-dimensional fields feasible. Applying this conclusion to Eq. (11) gives a compact representation for the full kernel function:

$$\mathrm{Cov}\left(y_i^{(f)}(\boldsymbol{\xi}), y_j^{(f)}(\boldsymbol{\xi}')\right) = k_d^{(f)}(i,j)\cdot \hat{k}_z^{(f)}\left([\boldsymbol{\xi}, z_*^{(f-1)}(\boldsymbol{\xi})], [\boldsymbol{\xi}', z_*^{(f-1)}(\boldsymbol{\xi}')]\right), \tag{12}$$

where $\hat{k}_z^{(f)}\left([\boldsymbol{\xi}, z_*^{(f-1)}(\boldsymbol{\xi})], [\boldsymbol{\xi}', z_*^{(f-1)}(\boldsymbol{\xi}')]\right) = k_\xi^{(f)}(\boldsymbol{\xi},\boldsymbol{\xi}')\cdot k_z^{(f)}\left(z_*^{(f-1)}(\boldsymbol{\xi}), z_*^{(f-1)}(\boldsymbol{\xi}')\right)$ is a composite kernel of $k_x^{(f)}$ and $k_z^{(f)}$. Rewriting Eq. (12) based on Lemma 1, we get

$$\boldsymbol{y}^{(f)}(\boldsymbol{\xi}) = \boldsymbol{\psi}^{(f)}\hat{z}^{(f)}(\boldsymbol{\xi}, z_*^{(f-1)}(\boldsymbol{\xi})), \tag{13}$$

where $\hat{z}_*^{(f-1)}(\boldsymbol{\xi})$ is a GP with kernel function $\hat{k}_z^{(f)}\left([\boldsymbol{\xi}, z_*^{(f-1)}(\boldsymbol{\xi})], [\boldsymbol{\xi}', z_*^{(f-1)}(\boldsymbol{\xi}')]\right)$. Recursively using Eq. (13) and Lemma 2 for $f = 1, \cdots, F$, we can see clearly that the lower fidelity knowledge propagates to the higher fidelity model only through the univariate latent process $\hat{z}_*^{(f-1)}(\boldsymbol{\xi})$, which can be solved efficiently using the univariate autoregression model of Perdikaris et al. [34].

Remark 1. If each multifidelity high-dimensional autoregressive model has a covariance structure of Eq. (9), the high-fidelity function $\boldsymbol{y}^{(F)}(\boldsymbol{\xi})$ depends only on an univariate autoregressive model and a base vector. Because such a covariance structure corresponds to a simplified intrinsic model of coregionalization [39], we name this model *intrinsic deep coregionalization*.

3.3.2. Deep coregionalization

Remark 1 implies that the propagation of the fidelity correlations in a high-dimensional model can be achieved equivalently throughout the propagation of the latent processes, which are independent of the output correlations. Up to this point, we have proposed an efficient way to extend the classic autoregressive models and their variations to resolve the high-dimensional challenge. Although similar assumptions, e.g., stationary kernel and separable structures, are also made in many low-dimensional multifidelity works [33,37,40,41], the model suggested by Remark 1 is restrictive and cannot generalize well to a wide variety of high-dimensional problems because 1) the kernel function is assumed to be stationary, which greatly limits the capacity of a GP model, and 2) Eq. (9) assumes an oversimplified separable kernel structure [14,15,4].

Indeed, Lemma 1 assumes the simplest case of the classic linear model of coregionalization (LMC) where there is only one latent process involved. This special case can be directly generalized by introducing multiple (potentially infinite) independent latent processes:

$$\boldsymbol{y}^{(f)}(\boldsymbol{\xi}) = \sum_{r=1}^{R_f} \boldsymbol{\psi}_r^{(f)} z_r^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})\right) = \boldsymbol{\Psi}^{(f)}\boldsymbol{z}^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})\right), \tag{14}$$

where $\boldsymbol{\Psi}^{(F)} = [\boldsymbol{\psi}_1^{(F)}, \ldots, \boldsymbol{\psi}_{R_f}^{(F)}]$ is the collection of orthogonal bases $\boldsymbol{\psi}_r^{(F)} \in \mathbb{R}^d$, and $\boldsymbol{z}^{(F)}(\boldsymbol{\xi}) = [z_1^{(F)}(\boldsymbol{\xi}), \ldots, z_{R_F}^{(F)}(\boldsymbol{\xi})]^T$ is the collection of corresponding latent processes, characterized by GPs. This formulation is also a common fundamental decomposition assumption successfully made for many physical processes and random processes [4,42,5,6,31]. This formulation is equivalent to a particular case of LMC [15,4], where each output-correlation matrix is parameterized by a vector of d length. Although the basis $\boldsymbol{\psi}_r^{(f)}$ is independent of ξ and thus the model still assumes a separable structure, the large number of latent process R_f can greatly improve the model capacity. When R_f tends to be infinity, the underlying covariance can approximate a nonseparable kernel [31]. Also, we can now further improve model capacity by introducing different kernel functions, e.g., nonstationary and period kernel functions, for different latent processes. The formulation of Eq. (14) implicitly represents a generalization of covariance structure suggested by Eq. (9) using an additive structure.

$$\mathrm{Cov}\left(y_i^{(f)}(\boldsymbol{\xi}), y_j^{(f)}(\boldsymbol{\xi}')\right) = \sum_{r=1}^{R_f} k_{dr}^{(f)}(i,j)\cdot k_{\xi r}^{(f)}(\boldsymbol{\xi},\boldsymbol{\xi}')\cdot k_{yr}^{(f)}(\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi}), \boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi}')), \tag{15}$$

where $k_{dr}^{(f)}, k_{\xi r}^{(f)}$ and $k_{yr}^{(f)}$ correspond to the r-th latent processes of between-output correlations, between-input correlations, and cross-fidelity correlation, respectively. $k_{dr}^{(f)}$ is simplified to have a maximum of d hyperparameters. To relax the stationary assumption made by Lemma 2, we now construct the deep coregionalization model on the general autoregressive model of Eq. (7). Substituting Eq. (14) into Eq. (7), we have

$$\begin{aligned}
\boldsymbol{\Psi}^{(f)}\boldsymbol{z}^{(f)}(\boldsymbol{\xi}) &= \mathbf{g}^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{\Psi}^{(f-1)}\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right)\\
\boldsymbol{z}^{(f)}(\boldsymbol{\xi}) &= {\boldsymbol{\Psi}^{(f)}}^T\mathbf{g}^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{\Psi}^{(f-1)}\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right)\\
\boldsymbol{z}^{(f)}(\boldsymbol{\xi}) &= \hat{\mathbf{g}}^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right),
\end{aligned} \tag{16}$$

where $\hat{\mathbf{g}}^{(f)}$ is the composite function absorbing $\boldsymbol{\Psi}^{(f)}$ and $\boldsymbol{\Psi}^{(f-1)}$ into the multivariate mapping $\mathbf{g}^{(f)}$.

Theorem 3.1. *If each fidelity output admits an LMC formulation (of Eq. (14)), $\hat{\mathbf{g}}_r^{(f)}$ must exist and can be decomposed into several independent univariate GPs, $\{\hat{g}_r^{(f)}(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}))\}_{r=1}^{R_f}$.*

We provide the proof in Appendix C. With this conclusion, we can see that the high-fidelity field depends only on the latent process of its previous fidelity and the inputs once the bases are found. The output correlations are captured independently, similar to the LMC for each fidelity, whereas the fidelity correlations are propagated through independent univariate deep GPs. We thus call the model *deep coregionalization*.

3.4. Model training and efficient implementation

So far, we have introduced the skeleton of our deep coregionalization model. The challenge is the optimization of model parameters, model hyperparameters, and the number of latent processes. We first rewrite Eq. (14) as a probabilistic formulation for each fidelity:

$$\begin{aligned}\boldsymbol{y}^{(f)}(\boldsymbol{\xi}) &= \sum_{r=1}^{R_f} \boldsymbol{\psi}_r^{(f)} \cdot \mathcal{GP}_r^{(f)}\left(0, k_r^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}), \boldsymbol{\xi}', \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}')\right)\right) \\ &= \mathcal{GP}^{(f)}\left(0, \sum_{r=1}^{R_f} \boldsymbol{\psi}_r^{(f)} \boldsymbol{\psi}_r^{(f)^T} \otimes k_r^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}), \boldsymbol{\xi}', \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}')\right)\right)\end{aligned} \tag{17}$$

where $\boldsymbol{\psi}_r^{(f)} \boldsymbol{\psi}_r^{(f)^T}$ is known as a coregionalization matrix for the r-th latent process, $\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})$ is the projections of $\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})$ on the space supported by $\boldsymbol{\Psi}^{(f-1)}$, $k_r^{(f)}$ is the kernel function for the r-th latent function of fidelity-f, and $\mathcal{GP}$ means a Gaussian process. The joint likelihood for the multifidelity observations $\{\boldsymbol{\Xi}^{(f)}, \mathbf{Y}^{(f)}\}_{f=1}^{F}$ is

$$\mathcal{L} = \sum_{f=1}^{F}\left(\frac{1}{2}\ln|\mathbf{K}_f + \tau_f^{-1}\mathbf{I}| - \frac{1}{2}\mathrm{vec}(\mathbf{Y}^{(f)})^T(\mathbf{K}_f + \tau_f^{-1}\mathbf{I})^{-1}\mathrm{vec}(\mathbf{Y}^{(f)}) - c_f\right), \tag{18}$$

where $c_f = \frac{dN_f}{2}\ln(2\pi)$, $\mathbf{K}_f = \sum_{r=1}^{R_f} \mathbf{K}_r^{(f)} \otimes \boldsymbol{\psi}_r^{(f)} \boldsymbol{\psi}_r^{(f)^T}$ is the full covariance matrix in fidelity-f, and $\mathbf{K}_r^{(f)} = k_r^{(f)}(\boldsymbol{\Xi}^{(f)}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\Xi}^{(f)}), \boldsymbol{\Xi}^{(f)}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\Xi}^{(f)}))$ is the covariance matrix for the fidelity-f and the r-th latent process for all available training data. Since $\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\Xi}^{(f)})$ depends on the posterior of the previous fidelity predictions for $f > 1$, the joint likelihood (18) does not have a tractable form and thus requires sampling approaches or variational approaches, which are either difficult to implement or inaccurate. Taking advantage of the nested training data structure (i.e., $\boldsymbol{\Xi}^{(f)} \subset \boldsymbol{\Xi}^{(f-1)}$), we treat the likelihood at each fidelity independently since the posterior $\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\Xi}^{(f)})$ is provided by the training data treated as deterministic values [34]. We can now separately optimize the joint likelihood w.r.t. model parameters and hyperparameters as implemented by Perdikaris et al. [34].

3.4.1. Efficient implementations via PCA

Even though optimizations of the likelihood function can be conducted separately without further referring to approximation methods such as Monte-Carlo sampling or variational Bayes, the direct implementation is still challenging in our case despite successful previous works [15,43,19]. First, the high-dimensionality d introduces $\sum_{f=1}^{F} d \times R_f$ model parameters, which will make the optimization extremely difficult for even a modest size of d with limited training data, and may cause overfitting issues [44,26]. Second, the number of latent processes is unclear. We can certainly place a sparse prior, e.g., Laplace prior, for the bases. However, the initialization normally requires a large number of latent processes to ensure a model with the capacity to allow for a shrinkage effect, which in turn greatly increases the computational burden (proportional to d). Lastly, optimization of Eq. (18) requires inversion of matrix $\mathbf{K}_f$, which is a $dN_f \times dN_f$ matrix, and a direct inversion is practically infeasible.

The choice of low-rank R_f introduces a trade-off between model capacity and training complexity. Shah et al. [31] explored the connection between the continuous function space (i.e., $d \to \infty$) and the finite case and proved that the maximum number is $R_f = d$, whereas a much smaller number $R_f \ll$ can be found via eigenanalysis of the covariance matrix of the functional space. The eigenanalysis approach is consistent with the work of Ramsey and Silverman [45] and Higdon et al. [4], in which a physical process is characterized using fundamental bases provided by principal component analysis (PCA), described in Algorithm 1. Another benefit of this approach is a heuristic method to choose the number of bases by choosing R_f, such that at least 90% of the variance of the output is covered by the bases. Higdon et al. [4] reported that the components that explain the minor trend of the variation do not add to the predictive ability of the GP model.

Since the simulation data is normally considered noise free, we can further omit the noise term τ_f in Eq. (18) [34,6,4]. We can then substitute the coregionalization matrix, provided by Algorithm 1, into Eq. (18) to derive a likelihood function depending on only a small amount of hyperparameters:

Algorithm 1 Coregionalization via PCA.

Input: Multifidelity high-dimensional simulation data $\{\mathbf{Y}^{(f)}\}_{f=1}^{F}$, number of bases for each fidelity $\{R^{(f)}\}_{f=1}^{F}$,

Output: Coregionalization bases $\{\mathbf{\Psi}^{(f)}\}_{f=1}^{F}$ and

1: **for** each $f \in [1, F]$ **do**

2: Centering output data, $\overline{\mathbf{Y}}^{(f)} = \mathbf{H}\mathbf{Y}^{(f)}$, where $\mathbf{H} = \mathbf{I} - \frac{1}{N_f}\mathbf{1}\mathbf{1}^T$ is the centering matrix.

3: Compute the covariance matrix $\mathbf{C}^{(f)} = \frac{1}{N_f}\overline{\mathbf{Y}}^{(f)^T}\overline{\mathbf{Y}}^{(f)}$

4: Solve the eigenvalue problem $\lambda_r^{(f)}\mathbf{u}_r^{(f)} = \mathbf{C}^{(f)}\mathbf{u}_r^{(f)}$; $\lambda_1 > \cdots > \lambda_{R^{(f)}}$ for $r = 1, \ldots, R^{(f)}$

5: Compute the coregionalization bases $\boldsymbol{\psi}_r^{(f)} = \sqrt{\lambda_r^{(f)}}\mathbf{u}_r^{(f)}$

6: **end for**

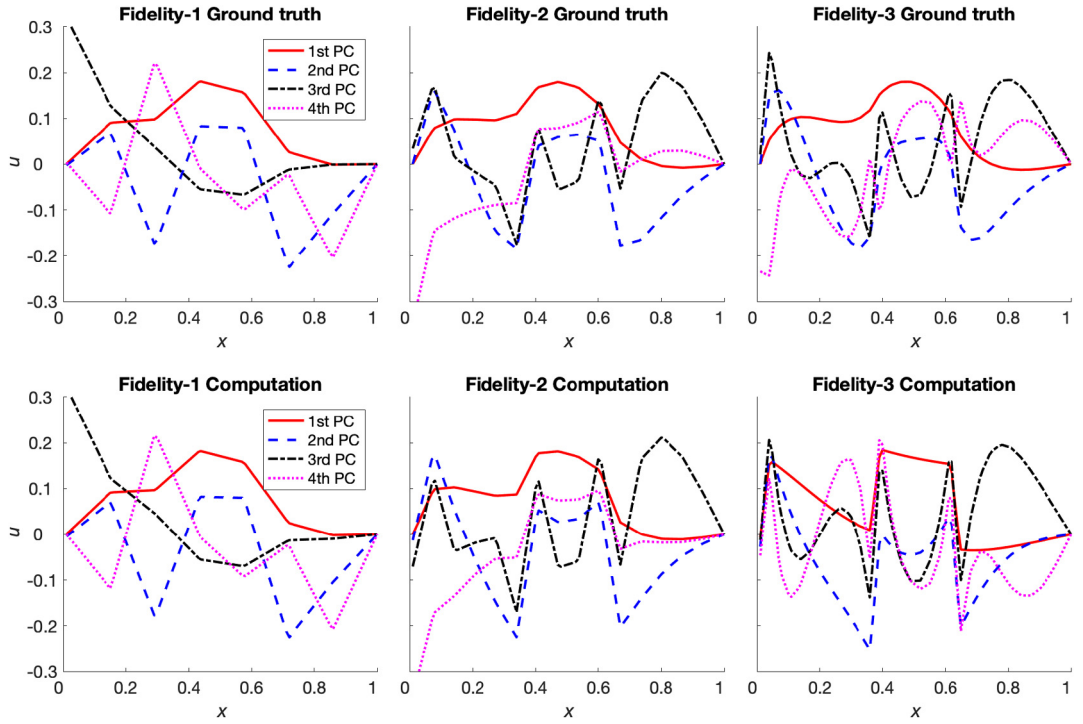

Fig. 1. Ground truth (top row) and computed (bottom row) normalized spatial principal components (PCs) for Burger's equation considered in Experiment 4. The number of data points for computation are {64,14,4} for Fidelity-{1,2,3}. It is clear that Fidelity-3 computed principal components are misleading due to the lack of training data; the ground-truth principal components are getting more and similar as fidelity increases.

$$\mathcal{L} \propto \sum_{f=1}^{F}\sum_{r=1}^{R_f}\frac{1}{2}\ln|\mathbf{K}_r^{(f)}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{Z}_r^{(f)^T}(\mathbf{K}_r^{(f)})^{-1}\mathbf{Z}_r^{(f)}\right), \tag{19}$$

where $\mathbf{Z}_r^{(f)} = \mathbf{Y}^{(f)}\boldsymbol{\psi}_r^{(f)}$ is the projection of $\mathbf{Y}^{(f)}$ onto the base $\boldsymbol{\psi}_r^{(f)}$, $\mathbf{K}_r^{(f)}$ is as defined in Eq. (8), and $[K_r^{(f)}]_{ij} = k_{fr}\left(\boldsymbol{\xi}_i, \boldsymbol{z}^{(f-1)}(\boldsymbol{\xi}_i), \boldsymbol{\xi}_j, \boldsymbol{z}^{(f-1)}(\boldsymbol{\xi}_j)\right)$ is the covariance matrix for the latent process $\hat{\mathbf{g}}_r^{(f)}(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}))$ based on its own kernel function k_{fr} and inputs $\boldsymbol{\Xi}^{(f)}$ and $\mathbf{Z}_{\mathbf{e}_f,:}^{(f-1)}$

3.4.2. Residual principal component analysis

The PCA approach of Algorithm 1 works well for many single-fidelity applications [4,19], but a direct implementation for our model here is inappropriate. First, the principal components (the bases) are essentially based on the empirical covariance matrix given data $\mathbf{Y}^{(f)}$. Since we can afford to have only a limited number of high-fidelity samples, the empirical covariance matrix is a very rough approximation of the real one, and the resulting principal components are consequently inaccurate (see Fig. 1). Second, even if we are always given enough samples to explore the output correlation at each fidelity, direct implementation of Higdon et al. [4] is of low efficiency. According to our experiments, when directly implementing PCA, the dominant components and their corresponding coefficients for each fidelity are always similar (see Fig. 1). The similarity implies that a using a complicated nonlinear model, e.g., GP, to propagate such a simple correlation in a deep structure is of low efficiency and may lead to inferior model accuracy. Instead, we can limit the information passing down the model and only model the updated (added) information using a GP model. Specifically, we specify

Algorithm 2 ResPCA.

Input: Multifidelity multivariate data $\{\mathbf{Y}^{(f)}\}_{f=1}^{F}$, subset indexes $\{\mathbf{e}_f\}_{f=1}^{F}$, and number of bases for each fidelity $\{R^{(f)}\}_{f=1}^{F}$,
Output: Residual bases $\{\mathbf{\Psi}^{(f)}\}_{f=1}^{F}$ and residual coefficients $\{\mathbf{Z}^{(f)}\}_{f=1}^{F}$;
1: $\widetilde{\mathbf{Y}}^{(1)} = \mathbf{Y}^{(1)}$
2: **for** each $f \in [2, F]$ **do**
3: Achieve residual information $\widetilde{\mathbf{Y}}^{(f)} = \mathbf{Y}^{(f)} - \mathbf{Y}^{(f-1)}_{(\mathbf{e}_f \cap \mathbf{e}_{f-1}),:}$
4: **end for**
5: Call Algorithm 1 using residual data $\{\widetilde{\mathbf{Y}}^{(f)}\}_{f=1}^{F}$ and the number of bases for each fidelity $\{R^{(f)}\}_{f=1}^{F}$ to get the residual bases $\widetilde{\mathbf{\Psi}}^{(f)}$ (on the residual data).
6: **for** each $f \in [2, F]$ **do**
7: Achieve residual coefficients $\widetilde{\mathbf{Z}}^{(f)} = \widetilde{\mathbf{Y}}^{(f)}\widetilde{\mathbf{\Psi}}^{(f)}$
8: **end for**

$$\begin{cases} \boldsymbol{y}^{(f)}(\boldsymbol{\xi}) = \mathbf{\Psi}^{(f)}\boldsymbol{z}^{(f)}(\boldsymbol{\xi}) & \text{for } f = 1 \\ \boldsymbol{y}^{(f)}(\boldsymbol{\xi}) = \mathbf{\Psi}^{(f)}\boldsymbol{z}^{(f)}(\boldsymbol{\xi}) + \boldsymbol{y}^{(f-1)}(\boldsymbol{\xi}) & \text{for } f \geq 2. \end{cases} \tag{20}$$

In this formulation, each fidelity layer learns the residual (additional) information compared with the previous fidelity layer rather than the whole output information, and thus we call this a *residual deep structure* similar to the work of He et al. [46]. Except for efficiency, another advantage of this approach is that the predictive posterior for high-fidelity results naturally decomposes into an additive structure. The additive structure can help us identify the main sources of uncertainty and adjust our model at each fidelity accordingly to reduce model uncertainty and improve model accuracy.

For a practical implementation, we do not need to rederive the model likelihood function. The likelihood function (19) depends only on the projections $\mathbf{Z}_r^{(f)}$ and the kernel functions for the kernels, which indicates that we only need to provide the projections $\mathbf{Z}_r^{(f)}$ on the residual spaces and the alternative bases. To this end, we introduce the *residual principal component analysis* (ResPCA), a modification of PCA applied to the residual information instead of the original data as shown in Algorithm 2. We can then substitute the projections of residual coefficients $\widetilde{\mathbf{Z}}^{(f)}$ and the residual bases $\widetilde{\mathbf{\Psi}}^{(f)}$ into Eq. (19) to replace $\mathbf{Z}^{(f)}$ and $\mathbf{\Psi}^{(f)}$ for model training.

3.5. Prediction and uncertainty propagation

Due to the intractability of a deep GP structure, except for the latent posterior $\boldsymbol{z}^{(1)}(\boldsymbol{\xi})$ corresponding to the lowest fidelity $\boldsymbol{y}^{(1)}(\boldsymbol{\xi})$, the predictive latent posterior for each fidelity is

$$\boldsymbol{z}_*^{(f)}(\boldsymbol{\xi}) = \int \mathbf{g}^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right) p\left(\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right) d\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}). \tag{21}$$

This integral is intractable and the posterior is no longer a Gaussian, but we can approximate it numerically using sampling-based methods. We can further reduce the complexity due to the independent assumption of the LMC; Eq. (21) naturally decomposes as

$$\boldsymbol{z}_*^{(f)}(\boldsymbol{\xi}) = \prod_{k=1}^{R_f} \int g_k^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right) \prod_{r=1}^{R_{f-1}} p\left(z_{*r}^{(f-1)}(\boldsymbol{\xi})\right) dz_{*r}^{(f-1)}(\boldsymbol{\xi}). \tag{22}$$

We can now easily implement a Monte Carlo sampling method to calculate the latent process posteriors. Another way to solve the intractable integral is to apply a Gaussian approximation of each latent posterior as in Perdikaris et al. [34] and Girard et al. [47]. Once we obtain the approximated posterior, the first- and second-order statistics admit a tractable solution due to the summation structure. Given the posterior of the latent processes $\boldsymbol{z}^{(f)}(\boldsymbol{\xi})$ and the bases $\mathbf{\Psi}^{(f)}$, the mean and variance of the finest fidelity field $\boldsymbol{y}^{(F)}(\boldsymbol{\xi})$ can be calculated as

$$\begin{aligned} \mathbb{E}[\boldsymbol{y}^{(F)}(\boldsymbol{\xi})] &= \sum_{f=1}^{F} \mathbf{\Psi}^{(f)} \mathbb{E}[\boldsymbol{z}_*^{(f)}(\boldsymbol{\xi})], \\ \mathrm{Var}[\boldsymbol{y}^{(F)}(\boldsymbol{\xi})] &= \sum_{f=1}^{F} \left(\mathbf{\Psi}^{(f)} \mathrm{Var}[\boldsymbol{z}_*^{(f)}(\boldsymbol{\xi})] {\mathbf{\Psi}^{(f)}}^T\right) \\ &\quad - 2 \sum_{f \neq f'} \left(\mathbf{\Psi}^{(f)} \mathrm{Cov}[\boldsymbol{z}_*^{(f)}(\boldsymbol{\xi}), \boldsymbol{z}_*^{(f')}(\boldsymbol{\xi})] {\mathbf{\Psi}^{(f')}}^T\right), \end{aligned} \tag{23}$$

where $\mathrm{Var}[\boldsymbol{z}^{(f)}(\boldsymbol{\xi})]$ is the diagonal covariance matrix of the fidelity-f latent posterior, and $\mathrm{Cov}[\boldsymbol{z}^{(f)}(\boldsymbol{\xi}), \boldsymbol{z}^{(f')}(\boldsymbol{\xi})$ is the covariance matrix of $\boldsymbol{z}^{(f)}(\boldsymbol{\xi})$ and $\boldsymbol{z}^{(f')}(\boldsymbol{\xi})$. This covariance matrix does not have an analytical form and can be calculated empirically using sampling methods.

3.6. Model complexity

Before we describe our experiments, we review our model and its implementation details for the emulation of spatial-temporal fields. 1) To build a surrogate model, we need to collect different fidelity data from simulations at design inputs $\mathbf{\Xi}$. To better explore the response surface, we use a Sobol sequence [48] to generate the design points. For fidelity-f, we choose the first N_f design points of $\mathbf{\Xi}$ to generate the corresponding outputs $\mathbf{Y}^{(f)}$. The computational cost at this stage depends on the simulator and the fidelity setting. 2) We then apply Algorithm 2 to get the residual bases $\{\mathbf{\Psi}^{(f)}\}_{f=1}^{F}$ and corresponding projections $\{\mathbf{Z}^{(f)}\}_{f=1}^{F}$. The computational cost for fidelity-f data is $\mathcal{O}(\min(N_f^2 d, N_f d^2))$ using SVD. For the choice of low rank R_f, we can set a variance ratio as in Higdon et al. [4]. Specifically, the minimum number of bases to capture the variance ratio (calculated by the eigenvalues of the covariance matrix) is used. 3) We maximize the joint likelihood function of Eq. (19) w.r.t. the hyperparameters using the gradient-based optimization methods, e.g., L-BFGS-B. This optimization consists of $\sum_{f=2}^{F}(R_f(R_{f-1}+3))$ hyperparameters, and the computational complexity is $\mathcal{O}(R_f N_f^3)$ per iteration. In cases where we have very limited training data, we can instead place a conditional independent assumption [49] for $\hat{g}_r^{(f)}(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}))$, which reduces the hyperparameters to $\sum_{f=2}^{F}(R_{f-1}+3)$ and complexity $\mathcal{O}(N_f^3)$ compared to the fully independent approach. For large numbers of observations, sparse GPs [50] can be placed on the latent process to reduce the computational cost. 4) Given a new $\boldsymbol{\xi}_*$, we calculate each latent posterior using Eq. (21) and then compute the mean and variance of the highest fidelity outputs using Eq. (23).

4. Experiments

In this section, we first examine our model with three canonical PDEs and compare it with other methods in terms of model capacity and accuracy. We then apply our model to a fluid dynamic problem to demonstrate the advantages of our model in more complicated real-world applications.

Competing methods. We compared deep coregionalization (DC) with four low-rank GP models for emulations of spatial-temporal fields: 1) PCA-GP [4], the popular LMC-based GP model for high-dimensional simulation data via principal component analysis (PCA); 2) IsoMap-GP [5], an extension of PCA-GP based on IsoMap [51], a classic nonlinear dimension reduction method; 3) KPCA-GP [6], another extension of PCA-GP based on implicit nonlinear bases through kernel PCA [52]; and 4) HOGP [19], a recent approach that tensorizes the outputs and introduces latent coordinate features (in tensor space) to model the output correlations.

Parameter settings. All models were implemented in Matlab using L-BFGS-B with a maximum of 100 iterations for model training and using ARD for all kernel functions. For deep coregionalization, a single low rank R is equally applied to each fidelity. IsoMap-GP and KPCA-GP used 10% of the training data to solve the additional pre-image problems, as is suggested in Xing et al. [5,6]. For each dataset, deep coregionalization integrates the examples of all the fidelities for training. Since the other methods work only on single-fidelity data, we conducted their training based on the examples of each fidelity separately. For instance, PCA-GP-F1 denotes PCA-GP trained with samples of Fidelity-1, whereas PCA-GP-F2 with Fidelity-2.

Evaluation. We varied the number of bases and the number of training samples to compare the performance of each method. The performance was measured using the root-mean-square error (RMSE), defined as $RMSE=\sqrt{\sum_{i,j}(\hat{y}_{ij}-y_{ij})^2/Nd}$, where here $\hat{y}_{ij}$ is the j-th dimension of prediction $\hat{\boldsymbol{y}}_i$, $\boldsymbol{y}_i$ is the ground truth, and $i=1,\ldots,N$ is the index of the total N test points. We also used the mean absolute error (MAE) field, defined as $\sum_i(|\hat{\boldsymbol{y}}_i-\boldsymbol{y}_i|)/N$, to demonstrate the local error. Data was normalized beforehand to provide a fair comparison among all competing methods.

4.1. Modeling fundamental PDEs

We consider three canonical PDEs: Poisson's equation, the heat equation, and Burger's equation. These PDEs play important roles in scientific and engineering applications [53–55]. They provide some common scenarios in simulations, such as high-dimensional spatial-temporal field outputs, nonlinearities, and discontinuities; they are often used as benchmark problems for surrogate models [56–59]. For our experiments, Poisson's equation, $\Delta u = f$, considered a spatial domain $\boldsymbol{x} \in [0,1]\times[0,1]$ with Dirichlet boundary conditions and $f = c\cdot\delta(x_1-\frac{1}{2}, x_2-\frac{1}{2})$, where δ is the Dirac delta function. The simulator was parameterized by the initial condition of the four boundaries and c from 0.1 to 0.9. The 1D viscous Burger's equation, $\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}=v\frac{\partial^2 u}{\partial x^2}$, where u represents the volume, x indicates a spatial location, t denotes the time, and v represents the viscosity, considered solutions for $x\in[0,1]$, $t\in[0,3]$ with an initial condition of $u(x,t_0)=\sin(x\pi/2)$ and homogeneous Dirichlet boundary conditions. The simulator was parameterized by the viscosity $v\in[0.01,0.1]$. The 1D heat equation, $\frac{\partial u}{\partial t}+\alpha\Delta u=0$, where u represents the heat, α the thermal conductivity, and Δ the Laplace operator, considered the solutions for $x\in[0,1]$ and $t\in[0,5]$ with Neumann boundary conditions and initial conditions of $u(x,0)=H(x-0.25)-H(x-0.75)$, where $H(\cdot)$ is the Heaviside step function. The simulator was parameterized by the flux rate of the left boundary at $x=0$ (ranging from 0 to 1), the flux rate of the right boundary at $x=1$ (ranging from -1 to 0), and the thermal conductivity (ranging from 0.01 to 0.1).

Simulation and data generation. We used the second-order finite difference method (FDM) [60,61] on an uniform grid to solve these PDEs. For the 2D Poisson's equation, we used an explicite Euler in time and a second order finite difference

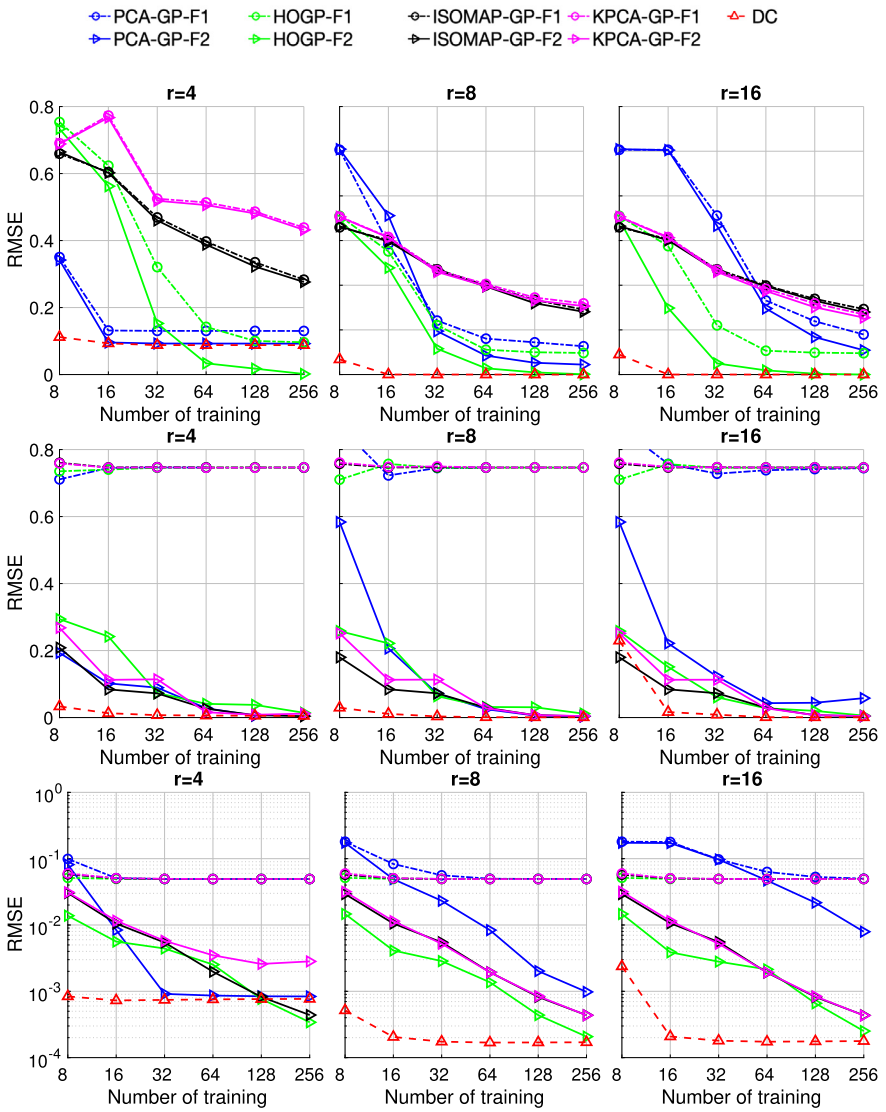

Fig. 2. RMSE for Poisson's equation (top row), heat equation (middle row), and Burger's equation (bottom row) using varying numbers of training samples. -F1, -F2 methods use Fidelity-1 and Fidelity-2 data, whereas deep coregionalization uses fixed 256 Fidelity-1 samples and varying Fidelity-2 samples; the low rank R varies from $\{4, 8, 16\}$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

stencil in space to ensure stability; for the 1D Burger's equation, we used first order upwinding for the convection terms and second order central differencing in space with explicite Euler in time; for the heat equation, we implemented the first order up-wind scheme in space and explicite Euler in time. The number of node/steps in the spatial (and temporal) grid used in the solver determines the data fidelity: the more nodes/steps in the solver, the higher the fidelity of the results. For each PDE, we first generated 256 design inputs using Sobol sequence [48] and 128 test inputs using uniform sampling. For each input, three fidelity field outputs were generated by running the simulation with three meshes, i.e., 16×16, 32×32, and 64×64 regular rectangular meshes. The simulation results were recorded using a 100×100 spatial-temporal (or just spatial) grid. Since random shuffling can invalidate the Sobol sequence for a better response surface coverage, we did not test each method using cross-validation.

We first conducted a two-fidelity test. We provided each model with 256 Fidelity-1 samples and varied the number of Fidelity-2 samples from $\{8, 16, 32, 64, 128, 256\}$ and the model low rank R from $\{4, 8, 16\}$. The Fidelity-2 test points were used as ground truths for validating the predictive results. The RMSE as a function of the number of training samples is shown in Fig. 2. Deep coregionalization uses Fidelity-1 and Fidelity-2 samples together, whereas the other methods use only Fidelity-1 or Fidelity-2 samples. In general, the Fidelity-1 based models show poor performance even though collecting the training samples is computationally cheap; as the number of latent processes increases (higher R), the model performance normally decreases with the same number of training samples due to the increasing model complexity. As we can see, deep coregionalization obtains the smallest prediction error in most cases. When the Fidelity-2 data is sufficient

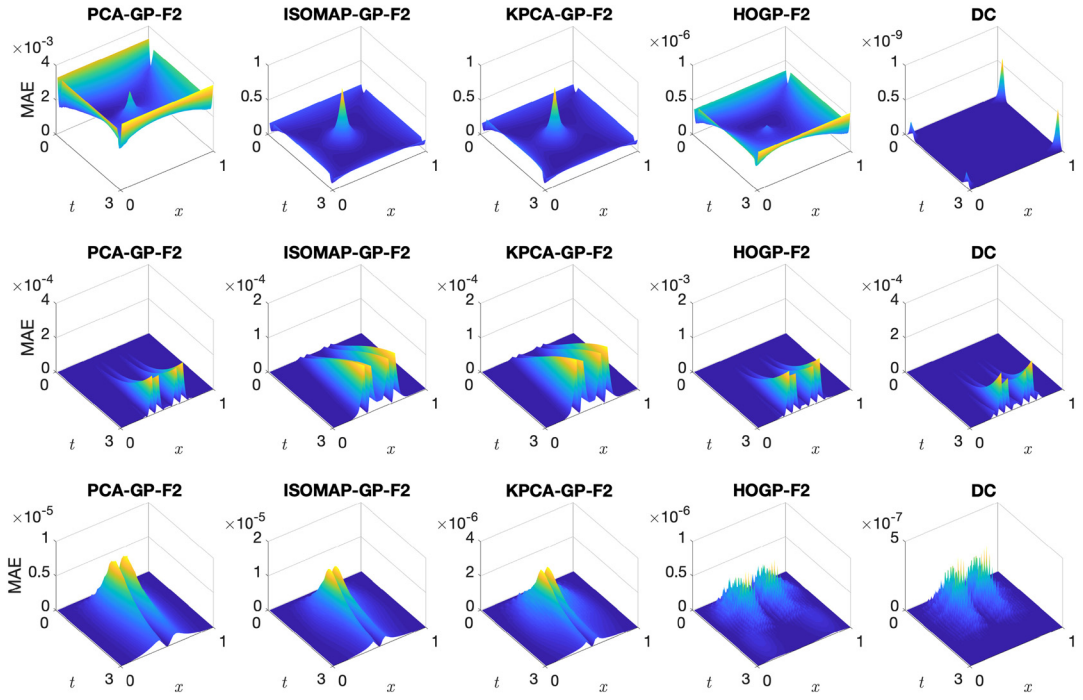

Fig. 3. MAE field for Poisson's equation (top row), heat equation (middle row) and Burger's equation (bottom row). Deep coregionalization uses 256 Fidelity-1 and 32 Fidelity-2 samples, whereas the other methods use 256 Fidelity-2 samples; the low rank is fixed at $R = 8$. The error fields show similar patterns for the same PDE. The superiority of deep coregionalization is clearly revealed by its error scale with 8 times fewer Fidelity-2 samples.

and the model low rank is low, i.e., $R = 4$, HOGP-F2 can achieve better performance due to its advantage of charactering the output correlations. With a sufficient low rank (beyond $R = 8$), deep coregionalization outperforms other methods by a large margin. The poor performance of deep coregionalization for the heat equation with $R = 16$ and 16 Fidelity-2 training samples may be caused by using too many latent processes (high R) in the deep structure, which makes model training difficult with limited data. As we have more Fidelity-2 samples, this fault disappears. We should emphasize that with only 16 Fidelity-2 samples, deep coregionalization can outperform most other methods with 16 times more, i.e., 256, Fidelity-2 samples.

To demonstrate the detailed error, we show the MAE fields produced by deep coregionalization using 256 Fidelity-1 samples and 32 Fidelity-2 samples, compared with those produced by Fidelity-2-based methods using 256 Fidelity-2 samples with a model low rank $R = 8$ in Fig. 3. The methods solely relying on Fidelity-1 data often produce significant errors as shown in Fig. 2; they are thus not included in this detailed demonstration. We can see that for different PDEs, the MAE fields have different patterns. For the Poisson's equation, the errors mainly gather around the energy sources, i.e., the boundaries and the center. {PCA,HOGP}-GP-F2 show good performances in reducing these errors, particularly in the center. Nevertheless, deep coregionalization completely outperforms them by orders of magnitudes despite that only an eighth of Fidelity-2 samples are used. For the heat equation, the overall errors increase along with time and gather at around the sharp corners of the initial conditions. Deep coregionalization shows a different pattern with faster error reduction with time and less overall error compared to other methods using 256 Fidelity-2 samples. For the Burger's equation, the errors tend to gather around the shock wave, which is consistent with existing studies [31]. The errors generated by deep coregionalization are not only smaller by an order of magnitude but also decay faster with time. To further assess the shock wave predictions, we show the absolute prediction error at t={1,2,3} s using a viscosity of 0.05 for all Fidelity-2 methods (using 256 Fidelity-2 samples) and deep coregionalization (using 256 Fidelity-1 samples + 32 Fidelity-2 samples) in Fig. 4. The Fidelity-1 methods are not included due to their poor performances. It is clear that all the methods produce reasonably accurate results. Particularly, the KPCA-GP-F2 shows accurate predictions due to its capacity to capture the nonlinearity of the Burger's equation. For most methods, as time increases, the error level remains similar, whereas deep coregionalization can further reduce the error level by encoding the correlations from the low fidelity.

Predicting the Fidelity-2 can be easy considering how simple the Fidelity-2 simulations are. Thus, we extended the experiment to include three fidelities and to predict the 128 Fidelity-3 fields. Since it is difficult to show all the combinations of different training number settings for the three fidelities, we introduced the fidelity ratio, defined by the ratio of training points at different fidelities. We can change the fidelity ratio freely here to explore the influence of the training setting on the model performance. In practice, this fidelity ratio should be adjusted to reflect the ratio of simulation costs to maximize the efficiency. The RMSEs of fidelity ratios of $4:2:1$ and $16:4:1$ are shown in Figs. 5 and 6, respectively. The x-axis denotes the number of training samples at Fidelity-1 (and consequently samples at Fidelity-2 and Fidelity-3). Overall, the

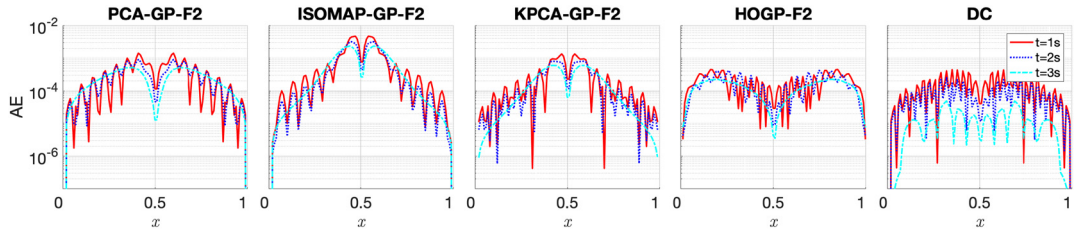

Fig. 4. Absolute error at t={1,2,3} s for Burgers' equation with viscosity=0.05 for deep coregionalization (with 256 Fidelity-1 + 32 Fidelity-2 samples) and other methods (with 256 Fidelity-2 samples); the row rank is fixed to $R = 8$.

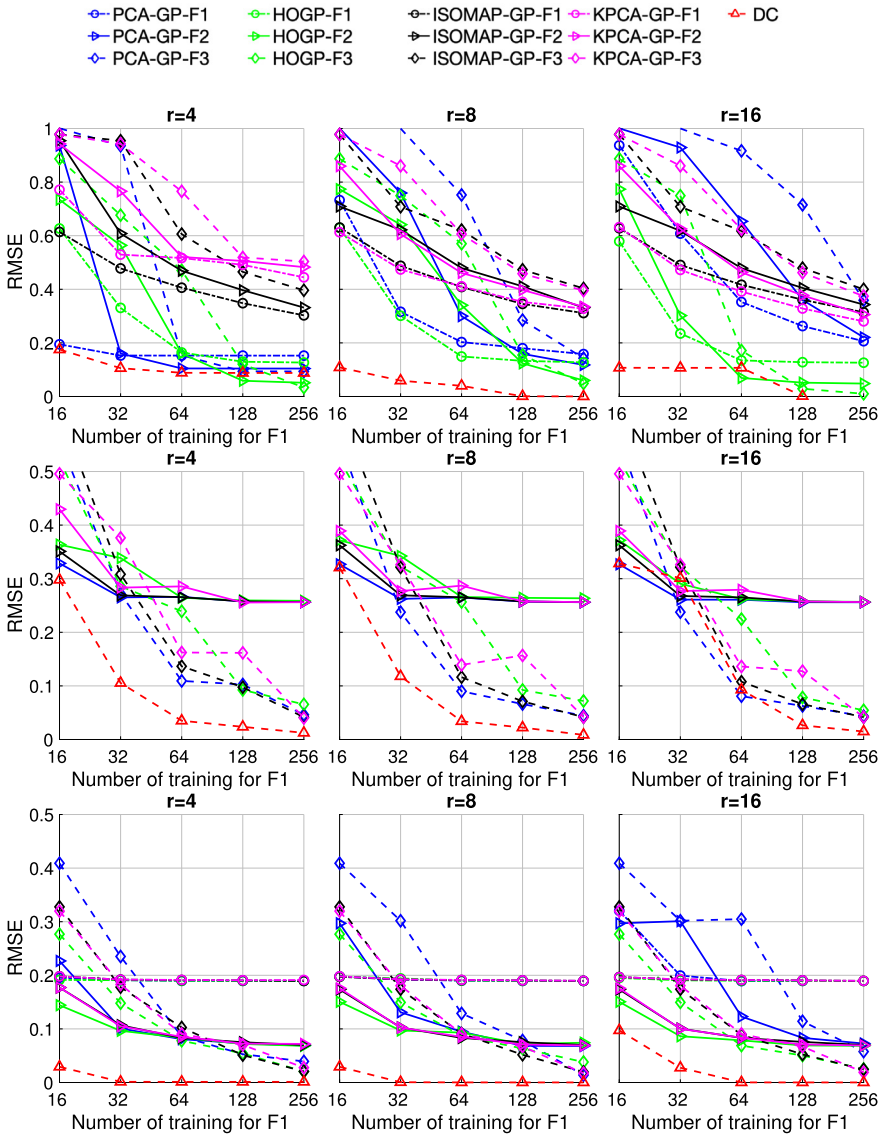

Fig. 5. RMSE for Poisson's equation (top row), heat equation (middle row), and Burger's equation (bottom row) with data of three fidelities, a low rank of $R = \{4, 8, 16\}$, and a fidelity ratio of $4 : 2 : 1$.

Fidelity-1- and Fidelity-2-based methods show limited improvements given more training data since the low-fidelity data do not contain high-fidelity information for the models to learn. The fidelity ratio has a significant impact on the Fidelity-3-based methods because these method rely merely on the Fidelity-3 samples; with only a few Fidelity-3 samples, they naturally perform badly. In contrast, the fidelity ratio has less influence on the performance of deep coregionalization; as the training samples increase, the performance converges to a similar level, which is also better than other methods with

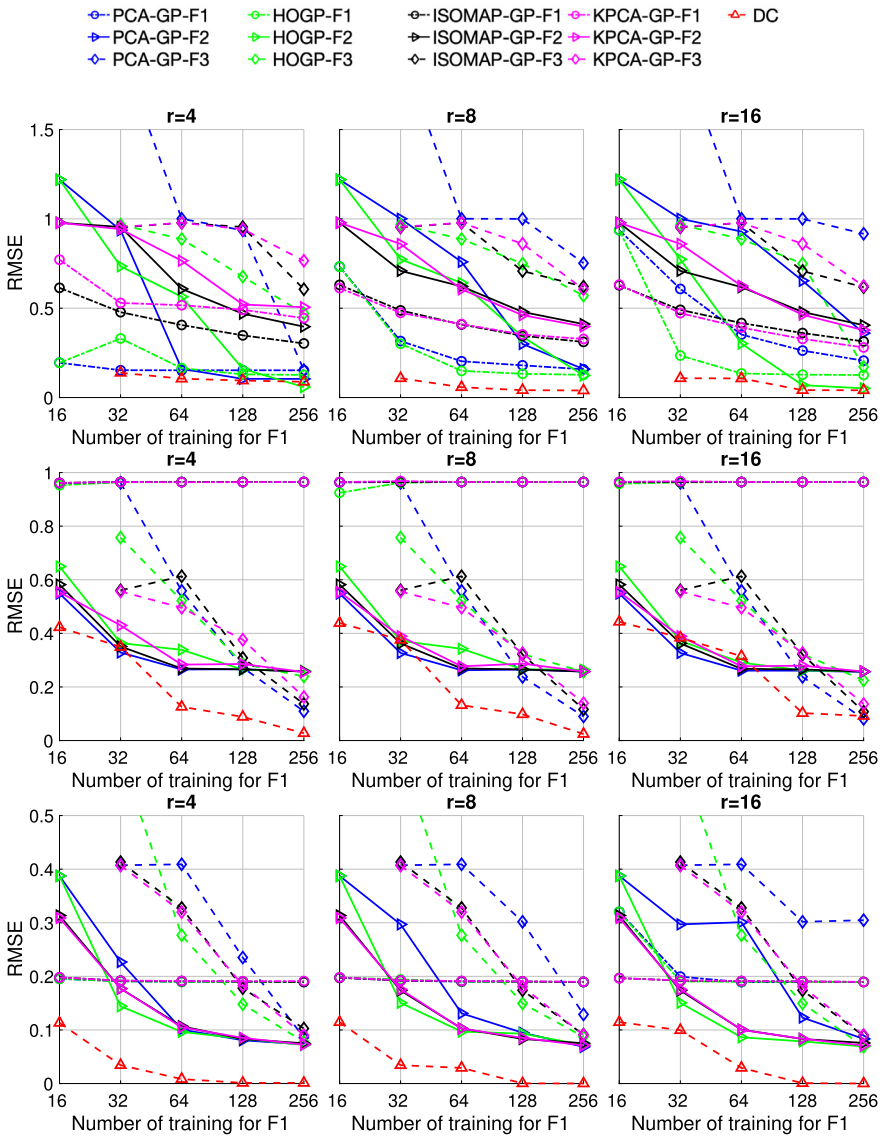

Fig. 6. RMSE for Poisson's equation (top row), heat equation (middle row), and Burger's equation (bottom row) with data of three fidelities, a low rank of $R = \{4, 8, 16\}$, and a fidelity ratio of $16 : 4 : 1$.

even 256 Fidelity-3 samples. Except for being stable, deep coregionalization outperforms other methods with a large margin at the same setting in most cases, especially with the fidelity ratio of $4 : 2 : 1$.

4.2. Applications to real simulation problems

Next, we applied deep coregionalization in a more difficult and large-scale physical simulation problem of fluid dynamics. Specifically, we emulated the spatial-temporal pressure field generated in a square 2D cavity $[0, 1] \times [0, 1]$ filled with a liquid that is driven by the top boundary representing a sliding lid. The problem is governed by the incompressible dimensionless Navier-Stokes equations [62]:

$$\frac{\partial \mathbf{u}}{\partial t}(\mathbf{u} \cdot \nabla)\mathbf{u} - Re^{-1}\nabla^2\mathbf{u} + \nabla p = 0, \quad \nabla \cdot \mathbf{u} = 0, \tag{24}$$

where $\mathbf{u} = (u_1, u_2)^T$ is the liquid velocity, p is the liquid pressure, and Re is the Reynolds number. The Navier-Stokes (NS) equation is known to be computationally challenging and has been a frequent benchmark problem for surrogate models [34, 6].

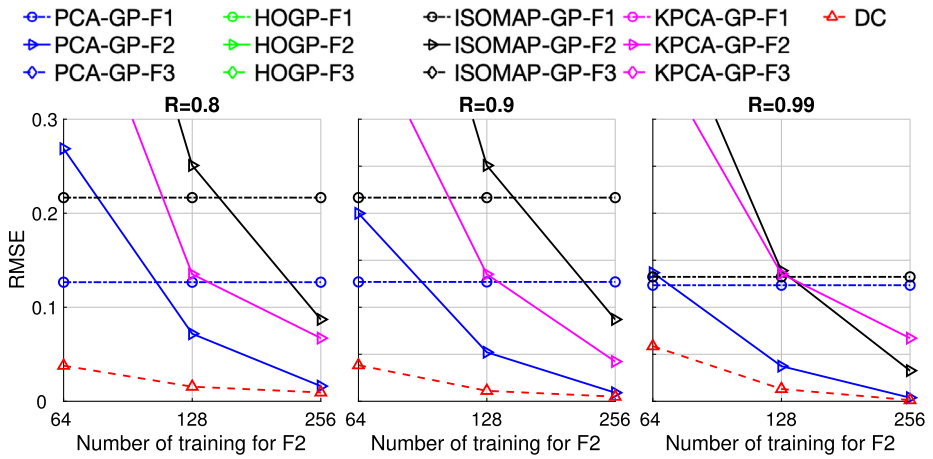

Fig. 7. RMSE for lid-driven cavity using training data of two fidelities. The number of Fidelity-2 samples varies from {64, 128, 256}, whereas the number of Fidelity-1 samples is fixed at 256. The low rank R varies from {0.8, 0.9, 0.99}.

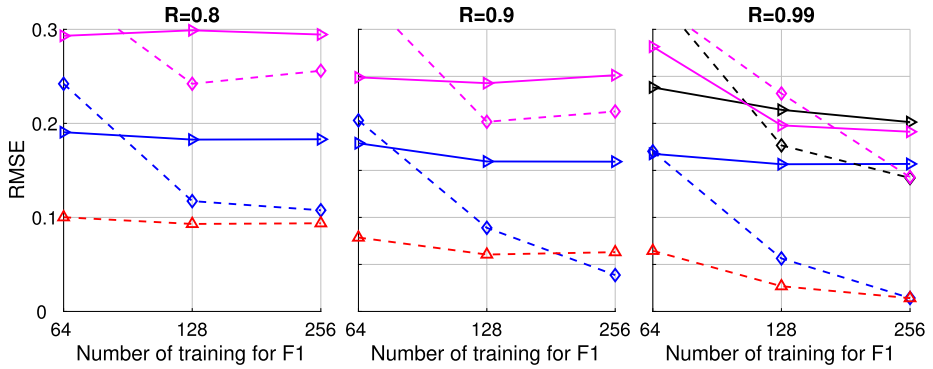

Fig. 8. RMSE for lid-driven cavity using data of three fidelities. The fidelity ratio is 16 : 4 : 1, and the low rank R varies from {0.8, 0.9, 0.99}.

Simulation setting. The Reynold's number and lid velocity were used as input parameters: $\boldsymbol{\xi} = (Re, u_{lid})^T \in [10, 1000] \times [0, 1]$. All other parameters were kept at the default values. We used the Sobol sequence to generate 256 design inputs for training and uniform sampling to generate 64 inputs for testing. For each input $\boldsymbol{\xi}_i$, we computed five fidelity pressure fields based on five spatial solver grids, 8×8, 16×16, 32×32, 64×64, and 128×128. We used finite differences on a staggered grid with implicit diffusion, a Chorin projection for the pressure [63], and 5000 fixed time steps to solve the PDEs. Each pressure field was recorded using a $100 \times 100 \times 100$ regular spatial-temporal grid, leading to a million degrees of freedom. According to our experiments, the average computational costs for generating a result of different fidelity were 0.3519 s, 0.5615 s, 1.4449 s, 5.7714 s and 32.2289 s for the Fidelity-1 to Fidelity-5 results, respectively. As mentioned previously, in real applications, the low rank R is normally determined by the preserved variance of the data. To reflect this choice, R, from here on, denotes the ratio of preserved variance, which implicitly decides the number of bases being used. HOGP requires significantly greater computational cost for very high-dimensional outputs and thus was not included in the following experiments.

We first conducted similar two- and three-fidelity experiments as before to examine the model capacities. The two-fidelity test was done with fixed 256 Fidelity-1 samples; the resulting RMSEs are shown in Fig. 7. Results of the three-fidelity test with a fidelity ratio of 16 : 4 : 1 are shown in Fig. 8. It is obvious that deep coregionalization outperforms others methods in most cases except for the second case in Fig. 8 with 256 Fidelity-1 samples.

Finally, we included samples from five fidelities to demonstrate our model performance when dealing with a more realistic application. Low-fidelity-based methods, i.e., the competing methods using only Fidelity-1 to Fidelity-4 data, are not demonstrated since they all show high errors even with 256 training samples. All methods use a low-rank variance ratio of $R = 0.99$. For deep coregionalization, we used two strategies to supply the training samples at each fidelity. DC(1) uses a fidelity ratio of 16 : 8 : 4 : 2 : 1 whereas DC(2) uses 256 : 64 : 16 : 4 : 1. We increased the training samples at Fidelity-5 (and consequently the number of training samples at other fidelities). The maximum number of training point for each fidelity was restricted to 256 due to the size of our dataset.

Fig. 9 demonstrates the RMSE as a function of simulation cost (in hours), which is the computational cost to generate the training data. It is clear that deep coregionalization can achieve better performance with reduced computational cost

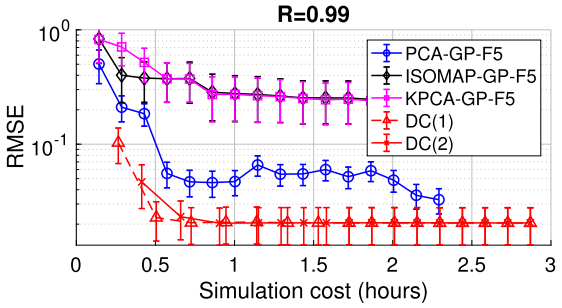

Fig. 9. RMSE as a function of the computational cost of generating the training data.

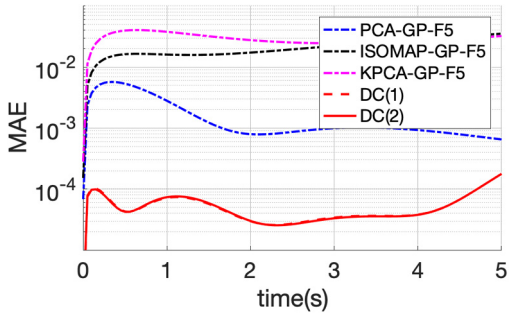

Fig. 10. MAE as a function of time using a simulation cost of approximately 0.7 hour and low rank $R = 0.99$.

compared with other methods. At the simulation cost of about 0.5 hours, deep coregionalization can achieve a performance that the other methods cannot beat even with about five times more simulation cost. Note that the fidelity ratio has only a small influence at low simulation cost in terms of model accuracy. PCA-GP is overall a stable method, but it requires more expensive high-fidelity data to slowly improve. ISOMAP-GP and KPCA-GP do not perform well even compared with PCA-GP. According to Xing et al. [5,6], these two methods improve slowly after the low rank exceeds a certain value, which is clearly the case here as we are preserving 99% of output variance. Another interesting discovery is that deep coregionalization converges not only quicker but also to a much lower error bound. According to Perdikaris et al. [34], due to the Markov properties, the high-fidelity predictions can benefit from the low-fidelity predictions only with the same input. When using 256 samples for all fidelities, we should see no improvement compared to directly using an LMC on 256 Fidelity-5 samples, i.e., the PCA-GP-F5. We believe this is due to the residual additive structure, which not only learns correlations between fidelities but also improves model capacity via the deep structure. The MAE as a function of time at the cost of around 0.7 hour is shown in Fig. 10. KPCA-GP-F5 and ISOMAP-GP-F5's error accumulates as time increases and propagates from the top (the driven lid) to the bottom boundary. PCA-GP-F5 shows a better performance in reducing the error, which is particularly significant in the first two seconds due to the drastic change in the pressure field. In contrast, deep coregionalization shows more than an order of magnitude lower error compared to PCA-GP-F5 before $t = 4$ s, after which the error increases, possibly due to the lack of high-fidelity samples to capture the evolving dynamics.

To further assess the predictions, we show the MAE fields at $t = \{1, 2, 3, 4, 5\}$ s in Fig. 11. For all methods, the major error transfers from the center to the top-right corner as the pressure center is transfered by the driving lid. Compared with KPCA- and ISOMAP-GP-F5, PCA-GP-F5 shows similar error fields with smaller levels. However, its error transformation lacks some smoothness at $t = 2$ s, possibility due to its incapacity of modeling a nonlinear trend as a linear-based model. Deep coregionalization shows predictions with significantly reduced errors by orders of magnitudes. Furthermore, it suppresses the error area during the process. In particular, at $t = 5$ s the large error area is reduced to a much smaller area compared with the other methods.

5. Discussion and conclusion

We have presented a novel framework for efficient emulations of spatial-temporal fields of an expensive simulator by utilizing its multifidelity setting. Our model can be seen as a fundamental generalization of the classic autoregressive model for high-dimensional problems based on the general assumption of LMC. To focus on the model framework itself, our implementation is based on a simple yet successful two-stage method [4]. Nevertheless, deep coregionalization is also a natural extension of LMC for multifidelity data; thus, it is readily implemented with the state-of-the-art methods based on LMC. For instance, joint learning of the bases can be implemented as suggested by Goovaerts et al. [39], Bonilla et al. [26], and Alvarez et al. [44] to potentially improve the model performance. Tensor decomposition can be applied to the high-dimensional outputs to improve the model scalability when conducting joint learning [19]. To improve model flexibility, the

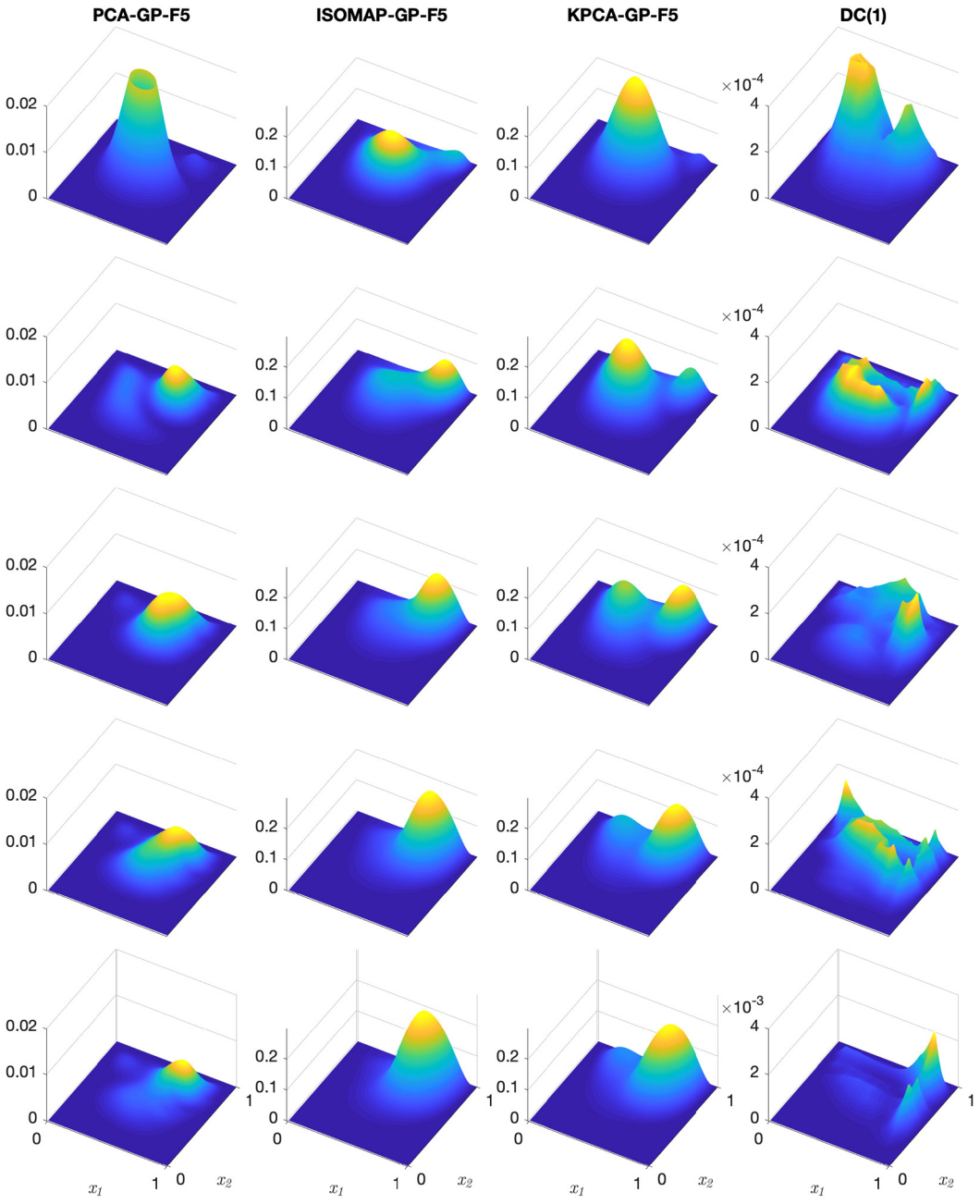

Fig. 11. MAE field for the NS equation of t={1,2,3,4,5} s (top to bottom rows) using a simulation cost of approximately 0.7 hour and low rank $R = 0.99$. The major error transfers from the center to the top-right corner where the pressure gathers. The superiority of deep coregionalization is revealed by the error scale and the error areas.

bases can be assumed functions of the inputs $\boldsymbol{\xi}$ [28]. The process convolution [23,24] can be implemented for nonstationary output correlations. Manifold learning [5,6] can be used to derive implicit bases to capture the nonlinear output correlations. Sparse GPs can be implemented for the latent process to reduce computational costs when dealing with large datasets [50, 64].

Another contribution of this work is the proposed ResPCA algorithm, which presents a new dimension reduction treatment for multifidelity data based on an addictive structure across different fidelities. This method can potentially be used for efficient base extractions for applications such as reduced order models [58,65,66] and sensitivity analysis [67,68] when multifidelity data is available.

The proposed method, deep coregionalization, consistently shows superior performance compared to the state-of-the-art GP-based high-dimensional emulators. Considering the extremely high computational cost for high-fidelity simulations in

real-life applications, we believe the proposed method should serve as a baseline for the emulations of simulation-based spatial-temporal fields.

CRediT authorship contribution statement

Wei W. Xing: Conceptualization, Investigation, Methodology, Software, Validation, Visualization. **Robert M. Kirby:** Funding acquisition, Project administration, Supervision, Writing – review & editing. **Shandian Zhe:** Conceptualization, Methodology, Supervision, Writing – review & editing.

Declaration of competing interest

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

Acknowledgement

This work was done when Dr. Wei W. Xing was a postdoc researcher at the University of Utah. This work has been supported by DARPA TRADES Award HR0011-17-2-0016.

Appendix A. Proof of Lemma 1

Firstly, we can write the full covariance specified by Eq. (9) in a matrix form,

$$\begin{aligned}\mathrm{Cov}\left[\boldsymbol{y}^{(f)}(\boldsymbol{\xi}),\boldsymbol{y}^{(f)}(\boldsymbol{\xi}')\right] &= \mathbf{K}_d^{(f)}\cdot k_x^{(f)}(\boldsymbol{\xi},\boldsymbol{\xi}')\cdot k_y^{(f)}\left([\boldsymbol{\xi},\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})],[\boldsymbol{\xi}',\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi}')]\right)\\ &= \mathbf{K}_d^{(f)}\cdot k_{xy}^{(f)}\left([\boldsymbol{\xi},\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})],[\boldsymbol{\xi}',\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi}')]\right)\end{aligned} \tag{25}$$

where $[\mathbf{K}_d^{(f)}]_{ij}=k_d^{(f)}(i,j)$ is the covariance matrix of outputs and $k_{xy}^{(f)}$ is a representation of the kernel function depending on $\boldsymbol{\xi}$ and $\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})$. Since a covariance matrix is always p.s.d., $\mathbf{K}_d^{(f)}$ admits an eigendecomposition of

$$\mathbf{K}_d^{(f)}=\sum_{r=1}\boldsymbol{v}_r^{(f)}\lambda_r^{(f)}\boldsymbol{v}_r^{(f)^T}. \tag{26}$$

For $\boldsymbol{y}^{(f)}(\boldsymbol{\xi})$, the separable assumption enables a general decomposition as follows:

$$\boldsymbol{y}^{(f)}(\boldsymbol{\xi})=\sum_{r=1}^{\infty}\boldsymbol{\psi}_r^{(f)}z_r^{(f)}\left(\boldsymbol{\xi},\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})\right). \tag{27}$$

Place a GP prior for each $\boldsymbol{z}^{(f)}\left(\boldsymbol{\xi},\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})\right)$, i.e.,

$$z_r^{(f)}\left(\boldsymbol{\xi},\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})\right)\sim\mathcal{N}\left(0,k_r([\boldsymbol{\xi},\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})],[\boldsymbol{\xi}',\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi}')])\right). \tag{28}$$

The sum of GPs is still a GP. Likewise, a linear transformation of a GP is also a GP [36]. From Eq. (27), we can derive

$$\boldsymbol{y}^{(f)}(\boldsymbol{\xi})\sim\mathcal{N}\left(\mathbf{0},\boldsymbol{\Psi}^{(f)}\mathrm{Cov}\left[\boldsymbol{z}(\boldsymbol{\xi}),\boldsymbol{z}(\boldsymbol{\xi}')\right]\boldsymbol{\Psi}^{(f)^T}\right), \tag{29}$$

where $\mathrm{Cov}\left[\boldsymbol{z}(\boldsymbol{\xi}),\boldsymbol{z}(\boldsymbol{\xi}')\right]$ is the covariance between the multivariate GPs $\{z_r(\boldsymbol{\xi},\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi}))\}_{r=1}^{\infty}$. We can further place a simplifying assumption that all outputs are independent, i.e., noncorrelated, such that the total covariance becomes

$$\mathrm{Cov}[\boldsymbol{z}(\boldsymbol{\xi}),\boldsymbol{z}(\boldsymbol{\xi}')]=\mathbf{I}\otimes k\left([\boldsymbol{\xi},\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})],[\boldsymbol{\xi}',\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi}')]\right), \tag{30}$$

where $\otimes$ is the Kronecker product. With this simplification, we can derive

$$\boldsymbol{y}^{(f)}(\boldsymbol{\xi})\sim\mathcal{N}\left(\mathbf{0},\sum_{r=1}^{\infty}\boldsymbol{\psi}_r^{(f)}\boldsymbol{\psi}_r^{(f)^T}k\left([\boldsymbol{\xi},\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})],[\boldsymbol{\xi}',\boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi}')]\right)\right). \tag{31}$$

If we let the bases in Eq. (31) be the product of the eigenvector and square root of eigenvalue in Eq. (26), i.e., $\boldsymbol{\psi}_r^{(f)}=\boldsymbol{v}_r^{(f)}\sqrt{\lambda_r^{(f)}}$, we can clearly see that Eq. (31) has the exact kernel function of Eq. (25). Note that Eq. (31) is achieved with a few simplifying assumptions. Thus, a covariance function of Eq. (9) is implicitly a special case of the decomposition of Eq. (27). Consequently, we can say that Eq. (31) is equivalent to a GP with a covariance function of the form of Eq. (9).

Appendix B. Proof of Lemma 2

A general kernel, $k_y^{(f)}(\boldsymbol{\psi}^{(f-1)}z_*^{(f-1)}(\boldsymbol{\xi}), \boldsymbol{\psi}^{(f-1)}z_*^{(f-1)}(\boldsymbol{\xi}'))$, is a valid stationary kernel iff

$$\begin{aligned} &k_y^{(f)}(\boldsymbol{\psi}^{(f-1)}z_*^{(f-1)}(\boldsymbol{\xi}), \boldsymbol{\psi}^{(f-1)}z_*^{(f-1)}(\boldsymbol{\xi}')) \\ &\quad = \int_{\mathbb{R}^d} \cos\left(\boldsymbol{\omega}^T\left(\boldsymbol{\psi}^{(f-1)}z_*^{(f-1)}(\boldsymbol{\xi}) - \boldsymbol{\psi}^{(f-1)}z_*^{(f-1)}(\boldsymbol{\xi}')\right)\right) F(d\boldsymbol{\omega}), \end{aligned} \tag{32}$$

where F is a positive finite measure [69]. Letting $\boldsymbol{\omega}^T\boldsymbol{\psi}^{(f-1)} = \boldsymbol{\phi}^T$, we have

$$\begin{aligned} &k_y^{(f)}(\boldsymbol{\psi}^{(f-1)}z_*^{(f-1)}(\boldsymbol{\xi}), \boldsymbol{\psi}^{(f-1)}z_*^{(f-1)}(\boldsymbol{\xi}')) \\ &= \int_{\mathbb{R}^d} \cos\left(\boldsymbol{\phi}^T\left(z_*^{(f-1)}(\boldsymbol{\xi}) - z_*^{(f-1)}(\boldsymbol{\xi}')\right)\right) F(\boldsymbol{\psi}^{(f)}d\boldsymbol{\phi}) \\ &= \int_{\mathbb{R}^d} \cos\left(\boldsymbol{\phi}^T\left(z_*^{(f-1)}(\boldsymbol{\xi}) - z_*^{(f-1)}(\boldsymbol{\xi}')\right)\right) \hat{F}(d\boldsymbol{\phi}) \\ &= k_z^{(f)}(z_*^{(f-1)}(\boldsymbol{\xi}), z_*^{(f-1)}(\boldsymbol{\xi}')). \end{aligned} \tag{33}$$

$\hat{F}(d\boldsymbol{\phi})$ must exist because $\boldsymbol{\psi}^{(f)}d\boldsymbol{\phi}$ represents a subset of $d\boldsymbol{\phi}$ and $\hat{F}$ is an arbitrary positive measure. Thus, $k_z^{(f)}(z_*^{(f-1)}(\boldsymbol{\xi}), z_*^{(f-1)}(\boldsymbol{\xi}'))$ exists and it is also stationary.

Appendix C. Proof of Theorem 3.1

By the LMC formulation of Eq. (14), we know that $\boldsymbol{y}^{(f)}(\boldsymbol{\xi}) = \mathbf{g}^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{\Psi}^{(f-1)}\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right)$ is a multivariate GP by its definition (a linear transformation of a GP is still a GP). Let's denote the unknown kernel function of $\mathbf{g}^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{\Psi}^{(f-1)}\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right)$ via a general kernel function

$$\mathrm{Cov}\left[y_i^{(f)}(\boldsymbol{\xi}), y_i^{(f)}(\boldsymbol{\xi})\right] = k\left([i, \boldsymbol{\xi}, \boldsymbol{\Psi}^{(f-1)}\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})], [j, \boldsymbol{\xi}', \boldsymbol{\Psi}^{(f-1)}\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}')]\right),$$

which can be further expanded as

$$\begin{aligned} &k\left(\left[i, \boldsymbol{\xi}, \boldsymbol{\Psi}^{(f-1)}\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right], \left[j, \boldsymbol{\xi}', \boldsymbol{\Psi}^{(f-1)}\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}')\right]\right) \\ &= \left\langle \varphi\left(\left[i, \boldsymbol{\xi}, \boldsymbol{\Psi}^{(f-1)}\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right]\right), \varphi\left(\left[j, \boldsymbol{\xi}', \boldsymbol{\Psi}^{(f-1)}\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right]\right)\right\rangle_{\mathcal{V}} \\ &= \left\langle \hat{\varphi}\left(\left[i, \boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right]\right), \hat{\varphi}\left(\left[j, \boldsymbol{\xi}', \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right]\right)\right\rangle_{\mathcal{V}} \\ &= \hat{k}\left(\left[i, \boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right], \left[, \boldsymbol{\xi}', \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}')\right]\right), \end{aligned} \tag{34}$$

where φ is an implicit feature mapping, $\langle\cdot,\cdot\rangle_{\mathcal{V}}$ is a proper inner product in the inner product space $\mathcal{V}$, $\hat{\varphi}$ is the composite function of φ absorbing $\boldsymbol{\Psi}^{(f-1)}$ and $\hat{k}$ is the composite kernel function. $\hat{\varphi}$ always exists since the input space $([i, \boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})])$ is a subspace of $[i, \boldsymbol{\xi}, \boldsymbol{y}_*^{(f-1)}(\boldsymbol{\xi})]$. Similarly, as k satisfies Mercer's condition, $\hat{k}$ also satisfies Mercer's condition and thus $\hat{k}$ must exist. Thus $\mathbf{g}^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{\Psi}^{(f-1)}\boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right)$ has a compact representation $\tilde{g}^{(f)}\left(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})\right)$. For the independence,

$$\begin{aligned} &\mathrm{Cov}\left[\hat{g}_i(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})), \hat{g}_j(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}))\right] \\ &\quad = \boldsymbol{\psi}_i^{(f)^T}\mathrm{Cov}\left[\tilde{g}_i(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})), \tilde{g}_j(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}))\right]\boldsymbol{\psi}_j^{(f)} \\ &\quad = \delta_{ij}\cdot\mathrm{Cov}\left[\tilde{g}_i(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi})), \tilde{g}_j(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}))\right] \end{aligned} \tag{35}$$

because $\boldsymbol{\psi}_i^{(f)}$ and $\boldsymbol{\psi}_j^{(f)}$ are orthogonal and thus $\hat{g}_i(\boldsymbol{\xi}, \boldsymbol{z}_*^{(f-1)}(\boldsymbol{\xi}))$ can be treated as independent an univariate GP, which matches the independent assumption for $\boldsymbol{z}^{(f)}(\boldsymbol{\xi})$.

References

[1] I. Bilionis, N. Zabaras, B. Konomi, G. Lin, Multi-output separable Gaussian process: towards an efficient, fully Bayesian paradigm for uncertainty quantification, J. Comput. Phys. 241 (2013) 212–239.
[2] A. Keane, P. Nair, Computational Approaches for Aerospace Design, John-Wiley and Sons, 2005.
[3] M. Kennedy, A. O'Hagan, Predicting the output from a complex computer code when fast approximations are available, Biometrika 87 (2000) 1–13.
[4] D. Higdon, J. Gattiker, B. Williams, M. Rightley, Computer model calibration using high-dimensional output, J. Am. Stat. Assoc. 103 (2008) 570–583.
[5] W. Xing, A.A. Shah, P.B. Nair, Reduced dimensional Gaussian process emulators of parametrized partial differential equations based on isomap, Proc. R. Soc. Lond., Ser. A, Math. Phys. Eng. Sci. 471 (2015) 20140697.
[6] W. Xing, V. Triantafyllidis, A. Shah, P. Nair, N. Zabaras, Manifold learning for the emulation of spatial fields from computational models, J. Comput. Phys. 326 (2016) 666–690.
[7] T.J. Santner, B.J. Williams, W. Notz, B.J. Williams, The Design and Analysis of Computer Experiments, vol. 1, Springer, 2003.
[8] M. Kennedy, C. Anderson, C. Conti, A. O'Hagan, Case studies in Gaussian process modelling of computer codes, Reliab. Eng. Syst. Saf. 91 (2006) 1301–1309.
[9] J. Rougier, D. Sexton, J. Murphy, D. Stainforth, Analyzing the climate sensitivity of the HadSM3 climate model using ensembles from different but related experiments, J. Climate 22 (2009) 3540–3557.
[10] P. Tagade, B. Jeong, H. Choi, A Gaussian process emulator approach for rapid contaminant characterization with an integrated multizone-CFD model, Build. Environ. 70 (2013) 232–244.
[11] L. Lee, K. Pringle, C. Reddington, G. Mann, P. Stier, D. Spracklen, J. Pierce, K. Carslaw, The magnitude and causes of uncertainty in global model simulations of cloud condensation nuclei, Atmos. Chem. Phys. 13 (2013) 8879–8914.
[12] M.C. Kennedy, A. O'Hagan, Bayesian calibration of computer models, J. R. Stat. Soc., Ser. B, Stat. Methodol. 63 (2001) 425–464.
[13] J. McFarland, S. Mahadevan, V. Romero, L. Swiler, Calibration and uncertainty analysis for computer simulations with multivariate output, AIAA J. 46 (2008) 1253–1265.
[14] G. Matheron, Pour une analyse krigeante des données régionalisées, Report N-732, Centre de Géostatistique, Fontainebleau, 1982.
[15] M. Goulard, M. Voltz, Linear coregionalization model: tools for estimation and choice of cross-variogram matrix, Math. Geol. 24 (1992) 269–286.
[16] B. Konomi, G. Karagiannis, A. Sarkar, X. Sun, G. Lin, Bayesian treed multivariate Gaussian process with adaptive design: application to a carbon capture unit, Technometrics 56 (2014) 145–158.
[17] T. Fricker, J. Oakley, N. Urban, Multivariate Gaussian process emulators with nonseparable covariance structures, Technometrics 55 (2013) 47–56.
[18] J. Rougier, Efficient emulators for multivariate deterministic functions, J. Comput. Graph. Stat. 17 (2008) 827–843.
[19] S. Zhe, W. Xing, R.M. Kirby, Scalable high-order Gaussian process regression, in: The 22nd International Conference on Artificial Intelligence and Statistics, 2019, pp. 2611–2620.
[20] S. Conti, A. O'Hagan, Bayesian emulation of complex multi-output and dynamic computer models, J. Stat. Plan. Inference 140 (2010) 640–651.
[21] H. Wackernagel, Multivariate Geostatistics, Springer, Berlin, 1995.
[22] A. Wilson, H. Nickisch, Kernel interpolation for scalable structured Gaussian processes (kiss-gp), in: International Conference on Machine Learning, 2015, pp. 1775–1784.
[23] D. Higdon, Space and space-time modeling using process convolutions, in: Quantitative Methods for Current Environmental Issues, Springer, 2002, pp. 37–56.
[24] P. Boyle, M. Frean, Dependent Gaussian processes, in: Advances in Neural Information Processing Systems, 2005, pp. 217–224.
[25] Y.-W. Teh, M. Seeger, M. Jordan, Semiparametric Latent Factor Models, Artificial Intelligence and Statistics, vol. 10, 2005, EPFL-CONF-161317.
[26] E.V. Bonilla, K.M. Chai, C. Williams, Multi-task Gaussian process prediction, in: Advances in Neural Information Processing Systems, 2008, pp. 153–160.
[27] A.G. Wilson, Z. Hu, R. Salakhutdinov, E.P. Xing, Deep kernel learning, in: Artificial Intelligence and Statistics, 2016, pp. 370–378.
[28] A.G. Wilson, D.A. Knowles, Z. Ghahramani, Gaussian process regression networks, arXiv preprint, arXiv:1110.4411, 2011.
[29] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[30] I. Macêdo, R. Castro, Learning Divergence-Free and Curl-Free Vector Fields with Matrix-Valued Kernels, IMPA, 2010.
[31] A. Shah, W. Xing, V. Triantafyllidis, Reduced-order modelling of parameter-dependent, linear and nonlinear dynamic partial differential equation models, Proc. R. Soc. A, Math. Phys. Eng. Sci. 473 (2017) 20160809.
[32] B. Peherstorfer, K. Willcox, M. Gunzburger, Survey of multifidelity methods in uncertainty propagation, inference, and optimization, SIAM Rev. 60 (2018) 550–591.
[33] M.C. Kennedy, A. O'Hagan, Predicting the output from a complex computer code when fast approximations are available, Biometrika 87 (2000) 1–13.
[34] P. Perdikaris, M. Raissi, A. Damianou, N. Lawrence, G.E. Karniadakis, Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling, Proc. R. Soc. A, Math. Phys. Eng. Sci. 473 (2017) 20160751.
[35] K. Cutajar, M. Pullin, A. Damianou, N. Lawrence, J. González, Deep Gaussian processes for multi-fidelity modeling, arXiv preprint, arXiv:1903.07320, 2019.
[36] C.E. Rasmussen, Gaussian processes in machine learning, in: Summer School on Machine Learning, Springer, 2003, pp. 63–71.
[37] L. Le Gratiet, Multi-fidelity Gaussian process regression for computer experiments, Ph.D. thesis, Université Paris-Diderot-Paris VII, 2013.
[38] A. Damianou, N. Lawrence, Deep Gaussian processes, in: Artificial Intelligence and Statistics, 2013, pp. 207–215.
[39] P. Goovaerts, et al., Geostatistics for Natural Resources Evaluation, Oxford University Press on Demand, 1997.
[40] X. Yang, G. Tartakovsky, A. Tartakovsky, Physics-informed kriging: a physics-informed Gaussian process regression method for data-model convergence, arXiv preprint, arXiv:1809.03461, 2018.
[41] X. Yang, D. Barajas-Solano, G. Tartakovsky, A.M. Tartakovsky, Physics-informed cokriging: a Gaussian-process-regression-based multifidelity method for data-model convergence, J. Comput. Phys. (2019).
[42] A. Narayan, C. Gittelson, D. Xiu, A stochastic collocation algorithm with multifidelity models, SIAM J. Sci. Comput. 36 (2014) A495–A521.
[43] M.E. Tipping, C.M. Bishop, Probabilistic principal component analysis, J. R. Stat. Soc., Ser. B, Stat. Methodol. 61 (1999) 611–622.
[44] M.A. Alvarez, L. Rosasco, N.D. Lawrence, et al., Kernels for vector-valued functions: a review, Found. Trends Mach. Learn. 4 (2012) 195–266.
[45] J.O. Ramsey, B.W. Silverman, Functional Data Analysis, Springer, 1997.
[46] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[47] A. Girard, C.E. Rasmussen, J.Q. Candela, R. Murray-Smith, Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting, in: Advances in Neural Information Processing Systems, 2003, pp. 545–552.
[48] I.M. Sobol, Uniformly distributed sequences with an additional uniform property, USSR Comput. Math. Math. Phys. 16 (1976) 236–242.
[49] N. Lawrence, The Gaussian process latent variable model, Technical Report CS-06-03, The University of Sheffield, 2006.
[50] E. Snelson, Z. Ghahramani, Sparse Gaussian processes using pseudo-inputs, in: Advances in Neural Information Processing Systems, 2005, pp. 1257–1264.

[51] M. Balasubramanian, E.L. Schwartz, The isomap algorithm and topological stability, Science 295 (2002) 7.
[52] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (1998) 1299–1319.
[53] S.C. Chapra, R.P. Canale, et al., Numerical Methods for Engineers, McGraw-Hill Higher Education, Boston, 2010.
[54] T. Chung, Computational Fluid Dynamics, Cambridge University Press, 2010.
[55] K. Burdzy, Z.-Q. Chen, J. Sylvester, et al., The heat equation and reflected Brownian motion in time-dependent domains, Ann. Probab. 32 (2004) 775–804.
[56] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (1998) 987–1000.
[57] R. Tuo, C.J. Wu, D. Yu, Surrogate modeling of computer experiments with different mesh densities, Technometrics 56 (2014) 372–380.
[58] M.O. Efe, H. Ozbay, Proper orthogonal decomposition for reduced order modeling: 2d heat flow, in: Proceedings of 2003 IEEE Conference on Control Applications, CCA 2003, vol. 2, IEEE, 2003, pp. 1273–1277.
[59] M. Raissi, P. Perdikaris, G.E. Karniadakis, Machine learning of linear differential equations using Gaussian processes, J. Comput. Phys. 348 (2017) 683–693.
[60] O.C. Zienkiewicz, R.L. Taylor, O.C. Zienkiewicz, R.L. Taylor, The Finite Element Method, vol. 36, McGraw-Hill, London, 1977.
[61] A.R. Mitchell, D.F. Griffiths, The Finite Difference Method in Partial Differential Equations, John Wiley, 1980.
[62] A.J. Chorin, Numerical solution of the Navier-Stokes equations, Math. Comput. 22 (1968) 745–762.
[63] B. Seibold, A compact and fast Matlab code solving the incompressible Navier-Stokes equations on rectangular domains mit18086 navierstokes, Massachusetts Institute of Technology, 2008.
[64] J. Hensman, N. Fusi, N.D. Lawrence, Gaussian processes for big data, arXiv preprint, arXiv:1309.6835, 2013.
[65] C. Audouze, P.B. Nair, Galerkin reduced-order modeling scheme for time-dependent randomly parametrized linear partial differential equations, Int. J. Numer. Methods Eng. 92 (2012) 370–398.
[66] A. Hay, J. Borggaard, I. Akhtar, D. Pelletier, Reduced-order models for parameter dependent geometries based on shape sensitivity analysis, J. Comput. Phys. 229 (2010) 1327–1352.
[67] V. Triantafyllidiis, W. Xing, P. Leung, A. Rodchanarowan, A. Shah, Probabilistic sensitivity analysis for multivariate model outputs with applications to Li-ion batteries, J. Phys., Conf. Ser. 1039 (2018) 012020.
[68] A. Saltelli, K. Chan, E.S. (Eds.), Sensitivity Analysis, John-Wiley and Sons, New York, 2000.
[69] M.G. Genton, Classes of kernels for machine learning: a statistics perspective, J. Mach. Learn. Res. 2 (2001) 299–312.