Bayesian Proper Orthogonal Decomposition for Learnable Reduced-Order Models with Uncertainty Quantification

Shahin Boluki, Siamak Zamani Dadaneh, Edward R. Dougherty, *Fellow, IEEE,* Xiaoning Qian, *Senior Member, IEEE*

***Abstract*—Designing and/or controlling complex systems in science and engineering relies on appropriate mathematical modeling of systems dynamics. Classical differential equation based solutions in applied and computational mathematics are often computationally demanding. Recently, the connection between reduced-order models of high-dimensional differential equation systems and surrogate machine learning models has been explored. However, the focus of both existing reduced-order and machine learning models for complex systems has been how to best approximate the high fidelity model of choice. Due to high complexity and often limited training data to derive reduced-order or machine learning surrogate models, it is critical for derived reduced-order models to have reliable uncertainty quantification at the same time. In this paper, we propose such a novel framework of Bayesian reduced-order models naturally equipped with uncertainty quantification as it learns the distributions of the parameters of the reduced-order models instead of their point estimates. In particular, we develop *learnable Bayesian proper orthogonal decomposition* (BayPOD) that learns the distributions of both the POD projection bases and the mapping from the system input parameters to the projected scores/coefficients so that the learned BayPOD can help predict high-dimensional systems dynamics/fields as quantities of interest in different setups with reliable uncertainty estimates. The developed learnable BayPOD inherits the capability of embedding physics constraints when learning the POD-based surrogate reduced-order models, a desirable feature when studying complex systems in science and engineering applications where the available training data are limited. Furthermore, the proposed BayPOD method is an end-to-end solution, which unlike other surrogate-based methods, does not require separate POD and machine learning steps. The results from a real-world case study of the pressure field around an airfoil have shown the potential of learnable BayPOD as a new family of reduced-order models with reliable uncertainty estimates.**

***Impact Statement*—Surrogate machine learning models for complex engineering and science systems have gained popularity with the advancement of machine learning and data-driven methods, where they can aid in the design or control of the systems with a lower computational burden. However, most of the existing methods either lack principled uncertainty estimation capabilities or cannot effectively incorporate physical constraints along the observations, which become even more important when the amount of available data is limited with respect to the system complexity. In this work we introduce a method that tries to overcome these limitations. The presented method is equipped with uncertainty quantification and can embed physics constraints. Our tests on two case studies have shown better prediction accuracy with reliable uncertainty estimates compared with the baseline reduced-order models. The method can also pave the way for adaptive decision making and Bayesian experimental design guided by reduced-order models incorporating scientific principles.**

***Index Terms*—Bayesian proper orthogonal decomposition, partial differential equations, reduced-order model, uncertainty quantification.**

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Mathematical Multifaceted Integrated Capability Centers program under Award DE-SC0019303. XQ is supported by the National Science Foundation (NSF) Awards 1553281, 1812641, 1835690, 1934904, 2119103, 2212419 and 2215573.

Shahin Boluki, Siamak Zamani Dadaneh, Edward. R. Dougherty, and Xiaoning Qian are with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843, USA (e-mail: xqian@ece.tamu.edu).

I. Introduction

MACHINE learning and artificial intelligence (ML/AI) have been revolutionizing modeling and decision-making in many real-world applications [14]. If generalizable predictive models can be learned, typically from “big” data, ML/AI can greatly help effective and efficient decision making. However, when facing complex natural and engineered systems, where available data of observations are small with respect to the system complexity, deriving generalizable ML models can be challenging. On the other hand, in applied and computational mathematics, research in simulating high-dimensional complex systems has been studied extensively with rich knowledge in fundamental physics principles, such as conservation laws and other governing equations. Nonetheless, it is often computationally expensive to simulate high-dimensional systems dynamics, typically by solving the corresponding Ordinary or Partial Differential Equation Systems (ODE/PDEs). Many recent research efforts have been made to develop ML methods to speed up computational simulations based on differential equation systems.

For example, neural networks have been used as (black-box) surrogates for physical systems [13], [17], and have recently gained renewed interest [18], [25] due to widespread availability of more powerful computational resources. Physics-informed neural networks (PINN) [18] represent one of such models where the input to the neural network is the spatial coordinates (and also time if time-dependent) and the output is the predicted output field(s). In PINNs, the physics principles are added via regularization terms in addition to the reconstruction loss for training the surrogate to encourage it to respect the underlying governing equations and the initial/boundary conditions with the help of automatic differentiation. PINNs have been recently extended [23] by employing Bayesian neural networks, i.e. placing a prior on the network weights and calculating an approximate posterior, to have a notion of uncertainty estimate. The Bayesian version of PINNs can only use samples from the boundary conditions and not full knowledge of it. Also, the experiments in [23] have shown that the training of Bayesian PINNs can be challenging where simpler variational approximations do not usually work and they require the more computationally complex Hamiltonian Monte Carlo approximation in order to result in satisfactory performances. In [25], Bayesian convolutional neural networks for image to image regression are used as a surrogate model for flow through porous media. The approach taken there lacks any specific mechanism to enforce boundary conditions. All these methods lack an interpretable lower-dimensional embedding, need retraining if boundary/initial conditions are changed, and still require a quite significant amount of data for training. Other works like [6], [11] assume that all the underlying governing equations are fully known and utilize them to train a neural network to imitate them.

In this paper, motivated by recent efforts to derive reduced-order models to approximate the high-fidelity solutions of differential equation systems by physics-based ML to embed physics constraints [21], we leverage Bayesian learning to develop a new framework of Bayesian reduced-order models (ROMs). Besides searching for reduced-order models that best approximate the high-fidelity differential equation solutions, Bayesian ROMs emphasize naturally-equipped uncertainty quantification capability, which is critical when designing and controlling complex systems in science and engineering often with little-to-no observed data, to enable reliable estimates of prediction confidence for robust decision making. Moreover, when learning reduced-order models of differential equation systems, the underlying scientific principles can be naturally incorporated as shown in [21].

There exist a wide variety of model reduction methods [7], [9], [10], [15], [20] that search for the best low-dimensional approximations of an underlying high-fidelity model, which is typically a high-dimensional system of ordinary differential equations or a system of equations stemming from the discretization of partial differential equations characterizing the corresponding systems dynamics. In this paper, we focus on reduced-order models based on the proper orthogonal decomposition (POD) [3] as they are closely related to subspace learning in ML/AI. In addition, the projection-based POD can be derived with embedded physics constraints, including system geometry, system configuration, initial conditions, and boundary conditions [21]. In particular, we develop *learnable* Bayesian POD (**BayPOD**). In BayPOD, we propose to simultaneously learn the distributions of both the POD projection bases and the mapping from the system input parameters to the projected scores/coefficients from "snapshots," solutions computed with the high-fidelity model for different inputs, which can include both the settings for the parameters of the high-fidelity or full-order model (FOM) and initial or boundary conditions. BayPOD is different from existing machine learning methods, for example, probabilistic principal component analysis [22], which aims at learning the distribution of latent representations by deriving linear projection of training data. Besides learning projection bases in a Bayesian framework, BayPOD focuses on simultaneously deriving a mapping from input parameters of the full-order model to latent projection coefficients and it also involves embedding physics constraints including initial/boundary conditions.

Figure 1 provides a schematic illustration of BayPOD, which leverages the subspace learning and regression models into one unified Bayesian learning framework to help reliably predict high-dimensional systems dynamics/fields as quantities of interest with significantly improved scalability and computational efficiency compared to the original high-dimensional ODE/PDE solvers. As shown in the figure, BayPOD is trained with snapshots generated/observed from the full-order model for a set of input parameters, and can then be used to predict the high-dimensional systems dynamics/fields for new input settings. More critically, the learned BayPOD models, due to its generative nature, can provide reliable uncertainty estimates of predicted systems dynamics in different setups, which will be the enabler of optimal and adaptive decision making when studying and intervening complex systems of interest.

Compared to the existing reduced-order models, our BayPOD has the following advantages:

- Our framework provides a unified way for learning POD basis and coefficients without resorting to multiple independent steps, as originally implemented in [21].
- We can quantify the uncertainty about field prediction for new inputs through posterior distributions.
- By incorporating prior distributions, the POD basis parameters are regularized to mitigate the impacts of high-dimensional snapshots with small sample size.
- Flexible models, such as neural networks (NNs), can be integrated for mapping from systems inputs to POD coefficients when needed, using amortized variational inference [12].
- Our BayPOD enables Bayesian experimental design with reduced-order models based on scientific principles, instead of "black-box" surrogate models.

The organization of the rest of the paper is as follows. Section II briefly reviews the background of POD and its machine learning extensions with physics constraints. Section III presents BayPOD and the corresponding inference algorithms. In Sections IV and V, case studies of predicting the temperature field of a heated rod and the pressure field around an airfoil are performed with both prediction and uncertainty quantification performance evaluation. Finally, Section VI concludes the paper.

II. Background

A. Proper Orthogonal Decomposition (POD)

Consider a system that maps an input onto a physical field such as pressure, temperature, stress, strain, etc. The physical field is the quantity of interest that we aim to predict. Denote a field as a function $f : \mathcal{X} \times \mathcal{T} \times \mathcal{P} \rightarrow \mathbb{R}$, with the spatial domain $\mathcal{X}$, time domain $\mathcal{T}$, and input domain $\mathcal{P}$. The field f varies in space and time, and depends on the input of the system. Given

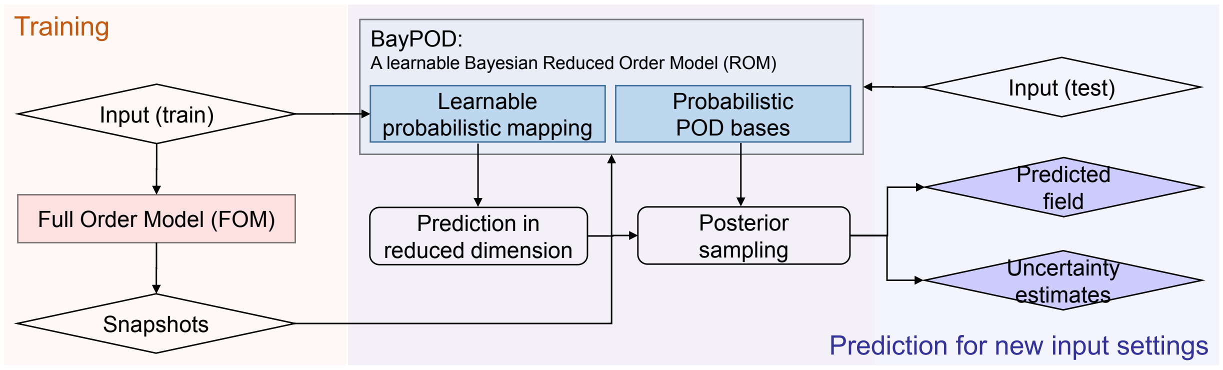


Fig. 1: Schematic diagram of BayPOD at training and for prediction. Inputs can include settings for the parameters of the full (high-fidelity) model and initial or boundary conditions.

the observed data $\mathcal{D} \subset \{f(\boldsymbol{x}, t; \boldsymbol{p}) | \boldsymbol{x} \in \mathcal{X}, t \in \mathcal{T}, \boldsymbol{p} \in \mathcal{P}\}$, we focus on learning approximate models f that respect the underlying physical constraints of the system.

Proper orthogonal decomposition (POD) is one of the most widely used model reduction methods which computes an expansion basis that enables a low-dimensional representation of the high-dimensional system state modeled by ODE/PDEs [21]. The goal in this paper is to facilitate learning a model to predict the high-dimensional output fields for new input settings of the corresponding high-fidelity ODE/PDEs while respecting physical constraints in the POD-based reduced-order model framework. Consider the field $f(\cdot, t; \boldsymbol{p})$ at time $t \in \mathcal{T}$ and with input $\boldsymbol{p} \in \mathcal{P}$. To calculate the POD basis, we introduce the finite-dimensional approximation $\boldsymbol{f}(t; \boldsymbol{p}) \in \mathbb{R}^{n_x}$ of $f(\cdot, t; \boldsymbol{p})$, where n_x is the dimension of the finite-dimensional discretization of the spatial domain. The approximate field $\boldsymbol{f}(t; \boldsymbol{p})$ is referred to as a *snapshot*, and it can be sensed data or a computational solution generated by a numerical model. The POD basis is computed using many such collected snapshots.

Let $\{\boldsymbol{f}(t_i; \boldsymbol{p}_j) | i = 1, ..., n_t, j = 1, ..., n_p\}$ be the set of $n_s = n_t n_p$ snapshots at n_t different time instances $\{t_1, ..., t_{n_t}\} \subset \mathcal{T}$ and for n_p different inputs $\{\boldsymbol{p}_1, ..., \boldsymbol{p}_{n_p}\} \subset \mathcal{P}$. The POD bases are then obtained by singular value decomposition (SVD) of the snapshot matrix $F = \left[\boldsymbol{f}(t_i; \boldsymbol{p}_j)\right]_{i,j} \in \mathbb{R}^{n_x \times n_s}$, which contains the snapshot vectors as its columns. More precisely, the SVD can be written as

$$F = V \Sigma W,$$

where the columns of the matrices $V \in \mathbb{R}^{n_x \times n_s}$ and $W \in \mathbb{R}^{n_s \times n_s}$ are the left and right singular vectors of F, respectively. The POD basis of dimension K , $V_K = [\boldsymbol{v}_1, ..., \boldsymbol{v}_K]$, is then defined as the K left singular vectors of F that correspond to the K largest singular values, where $K \ll n_x$.

B. Physical fields in the POD basis

After learning the POD basis from snapshot data, any field f can be approximated by a linear expansion as:

$$\tilde{\boldsymbol{f}}(t; \boldsymbol{p}) = \sum_{k=1}^{K} \boldsymbol{v}_k \alpha_k(t; \boldsymbol{p}), \tag{1}$$

where $\alpha_k(t; \boldsymbol{p})$ is the POD expansion coefficients and $\tilde{\boldsymbol{f}}(t; \boldsymbol{p})$ is the approximation of the field $f(\cdot, t; \boldsymbol{p})$ at time t and input $\boldsymbol{p}$. The POD expansion coefficients can be calculated as $\alpha_k(t; \boldsymbol{p}) = \boldsymbol{v}_k^T \boldsymbol{f}(t; \boldsymbol{p})$, for $k \in \{1, ..., K\}$.

The linear representation (1) provides a mechanism for embedding physical constraints. An approach to embed physical constraints into POD representation is by considering an alternative representation to (1) as:

$$\tilde{\boldsymbol{f}}(t; \boldsymbol{p}) = \bar{\boldsymbol{f}} + \sum_{k=1}^{K} \bar{\boldsymbol{v}}_k \alpha_k(t; \boldsymbol{p}), \tag{2}$$

where $\bar{\boldsymbol{f}}$ is a *particular solution*. The particular solution can be dependent on time and/or input parameters of the corresponding full-order models (ODE/PDEs). As an example, the particular solution $\bar{\boldsymbol{f}}$ is chosen to satisfy a particular set of prescribed inhomogeneous boundary conditions and the POD bases $\bar{\boldsymbol{v}}$ are defined so that they satisfy homogeneous boundary conditions.

C. Learning POD coefficients

The aim is to derive a function mapping from the input parameters to POD coefficients to approximate the high-dimensional field as the output of the full-order model for new inputs. Recently, machine learning methods have been employed to learn a surrogate model for the map $\boldsymbol{\alpha} : \mathcal{P} \to \mathcal{A}$ from inputs $\boldsymbol{p} \in \mathcal{P}$ to the POD coefficients $\boldsymbol{\alpha}(\boldsymbol{p}) \in \mathcal{A}$, where $\boldsymbol{\alpha}(\boldsymbol{p}) = [\alpha_1(\boldsymbol{p}), ..., \alpha_K(\boldsymbol{p})]$ and we assume inputs $\boldsymbol{p} = [p_1, ..., p_m]$ are m-dimensional system parameters [21], which can additionally include time as well. For brevity of notations, hereafter when referring to inputs for the map, time can be one dimension of the input vector $\boldsymbol{p}$. In the first step, we collect the inputs corresponding to the snapshots in a matrix $P \in \mathbb{R}^{n_s \times m}$, and their corresponding POD coefficients in a matrix $A \in \mathbb{R}^{n_s \times K}$. Note that the time-dependency of coefficient parameters is captured through snapshots. Then, input and output data are divided into training and test sets, and the map $\boldsymbol{\alpha} : \mathcal{P} \to \mathcal{A}$ is learned from the training data by applying supervised machine learning methods such as neural networks, decision trees or k-nearest neighbors regression model [21].

Due to high complexity and the approximation gap involved in the reduced-order model, principled (i.e. theoretically grounded, not *ad hoc*) uncertainty quantification for the prediction of the output fields of the full-order model for new input parameters becomes critical. Our proposed method in the next section, which is grounded in Bayesian inference, addresses this requirement.

III. Bayesian POD

In this section, we introduce our framework of Bayesian reduced-order models, BayPOD, which simultaneously learns the distributions of both POD projection bases and mapping from system inputs to projection coefficients. BayPOD is a Bayesian matrix factorization framework for simultaneously learning POD bases together with the relationship between input parameters and POD coefficients. The modeling of mapping from inputs to coefficients can be flexible. In this paper, we focus on linear parameter models (BayPOD-LM) first and then extend it to neural network models (BayPOD-NN) with amortized variational inference.

A. BayPOD – A Generative POD Model

We start by modeling the homogeneous field $\tilde{\boldsymbol{f}}$ in (1) using a multivariate normal distribution. The framework can be readily extended to (2) by adding the particular solution $\bar{\boldsymbol{f}}$.

Let $\tilde{f}_{sx}$ denote the field response for snapshot $s \in \{1, 2, ..., n_s\}$ at the spatial point $x \in \{1, 2, ..., n_x\}$. We model this response as a normally-distributed random variable:

$$\tilde{f}_{sx} \sim \mathrm{N}(\boldsymbol{u}_x^T \boldsymbol{\alpha}_s, \gamma_x^{-1}), \tag{3}$$

where $\boldsymbol{u}_x = [u_{x1}, ..., u_{xK}] \in \mathbb{R}^K$ is the K-dimensional POD basis vector at position x and $\boldsymbol{\alpha}_s = [\alpha_{s1}, ..., \alpha_{sK}] \in \mathbb{R}^K$ represents the K POD coefficients for snapshot s. The variance γ_x^{-1} can be considered as the model uncertainty at position x. In other words, the corresponding probability density function (PDF) $p(\tilde{f}_{sx}|\boldsymbol{u}_x, \boldsymbol{\alpha}_s, \gamma_x) = \mathrm{N}(\tilde{f}_{sx}; \boldsymbol{u}_x^T\boldsymbol{\alpha}_s, \gamma_x^{-1}) = \sqrt{\frac{\gamma_x}{2\pi}} \exp\big[-\frac{1}{2}\gamma_x(\tilde{f}_{sx} - \boldsymbol{u}_x^T\boldsymbol{\alpha}_s)^2\big]$, following the commonly used notations. In what follows, the random variables are explicitly indicated on the left side of the punctuation mark ";" and the parameters or fixed variables are denoted on the right side of ";" of the corresponding PDFs.

We place independent zero-mean normal priors on POD basis and coefficients:

$$\begin{aligned} \boldsymbol{u}_x &\sim \mathrm{N}(0, I_K), \\ \boldsymbol{\alpha}_s &\sim \mathrm{N}(0, \gamma_\alpha^{-1} I_K), \end{aligned} \tag{4}$$

where I_K is the identity matrix, and γ_α is the precision parameter for $\boldsymbol{\alpha}_s$. Note that K is the dimension of subspace (POD bases/factors/principal components). Employing the priors in (4) has multiple benefits. First, by placing zero-mean priors on $\boldsymbol{u}$ and $\boldsymbol{\alpha}$, we ensure that the marginal distribution of $\tilde{\boldsymbol{f}}$ is zero mean, and thus physical constraints can be applied through the particular solution. Second, normal priors enhance the robustness of our model in the presence of small sample size data, as they play a role similar to ridge regularization [8]. Finally, by using identity covariance matrix for POD basis $\boldsymbol{u}$ in the prior distribution, we aim to reduce the scale non-identifiability of $\boldsymbol{u}$ from $\boldsymbol{\alpha}$ in the model as $\tilde{\boldsymbol{f}}$ is related to $\boldsymbol{u}$ and $\boldsymbol{\alpha}$ through their multiplication. To complete the model, we place conjugate gamma distributions over the position and coefficient precision parameters:

$$\gamma_\alpha, \gamma_x \sim \mathrm{Gamma}(1, 1). \tag{5}$$

1) Inference model: A primary goal of model reduction is to predict the system response to new input parameters by leveraging the learned basis vectors. We attain this goal by introducing an inference (recognition) network, widely used in variational inference literature [2], [12], [19], [24].

For variational inference, we introduce variational distributions $q(\cdot)$ over model parameters as approximations for intractable posterior distributions. For our Bayesian reduced-order model, to simplify deriving the variational parameters, we assume the following independence structure for variational distributions:

$$q(\boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) = q(\boldsymbol{u})q(\boldsymbol{\alpha})q(\boldsymbol{\gamma}). \tag{6}$$

To establish amortized inference of POD coefficients $\boldsymbol{\alpha}_s$ for $s \in \{1, ..., n_s\}$, we define their variational distributions as

$$q(\boldsymbol{\alpha}_s) = \mathrm{N}(\boldsymbol{\alpha}_s; \boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{p}_s), \Sigma_{\boldsymbol{w}}(\boldsymbol{p}_s)), \tag{7}$$

where $\boldsymbol{\mu}_{\boldsymbol{w}}$ and $\Sigma_{\boldsymbol{w}}$ are the mean vector and covariance matrix which take the form of some mapping with weights $\boldsymbol{w}$ from input parameters $\boldsymbol{p}$. Hence, for new input parameters $\boldsymbol{p}^*$, the variational posterior mean $\boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{p}^*)$ can be considered as an estimate of the POD coefficients.

Finally, to exploit the conjugate priors, we let the variational posteriors for POD basis and precision parameters to be normal and gamma distributions, respectively:

$$\begin{aligned} q(\boldsymbol{u}_x) &= \mathrm{N}(\boldsymbol{u}_x; \boldsymbol{\mu}_x, \Sigma_x), \\ q(\gamma_x) &= \mathrm{Gamma}(\gamma_x; \lambda_x, 1/r_x), \\ q(\gamma_\alpha) &= \mathrm{Gamma}(\gamma_\alpha; \lambda_\alpha, 1/r_\alpha). \end{aligned} \tag{8}$$

To obtain the optimal variational parameters $\boldsymbol{\Theta} = \{\boldsymbol{\mu}, \Sigma, \boldsymbol{\gamma}, \boldsymbol{\lambda}, \boldsymbol{r}, \boldsymbol{w}\}$, our variational inference procedure minimizes the Kullback-Leibler (KL) divergence between the variational posteriors and the true posteriors, or equivalently maximizes the evidence lower bound (ELBO) of the marginal log-likelihood $\log p(\tilde{\boldsymbol{f}})$ [1], [4]:

$$\begin{aligned} &\mathcal{L}(\boldsymbol{\Theta}|\mathcal{D} = \{\tilde{\boldsymbol{f}}_s, \boldsymbol{p}_s\}_{s=1}^{n_s}) = \\ &\mathbb{E}_{q(\boldsymbol{u},\boldsymbol{\alpha},\boldsymbol{\gamma})}\Big[\log \frac{p(\tilde{\boldsymbol{f}}|\boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\gamma})p(\boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\gamma})}{q(\boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\gamma})}\Big] \leq \log p(\tilde{\boldsymbol{f}}), \end{aligned} \tag{9}$$

where we have the conditional independence assumption for $p(\tilde{\boldsymbol{f}}|\boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) = \prod_{s,x} p(\tilde{f}_{sx}|\boldsymbol{u}_x, \boldsymbol{\alpha}_s, \gamma_x)$, and $p(\boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) = p(\boldsymbol{u})p(\boldsymbol{\alpha}|\boldsymbol{\gamma})p(\boldsymbol{\gamma})$, with the corresponding components defined in (4) and (5). Note that here, $\log p(\tilde{\boldsymbol{f}})$ denotes the marginal log-likelihood of field responses, integrated over the parameter space. The variational distribution in the demonimator $q(\boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\gamma})$ is defined in equations (6) to (8), and $\mathbb{E}_{q(\boldsymbol{u},\boldsymbol{\alpha},\boldsymbol{\gamma})}$ denotes the expectation with respect to the variational distributions $q(\boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\gamma})$. More specifically,

$$p(\tilde{\boldsymbol{f}}|\boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) = \prod_{s=1}^{n_s}\prod_{x=1}^{n_x} \mathrm{N}(\tilde{f}_{sx}; \boldsymbol{u}_x^T\boldsymbol{\alpha}_s, \gamma_x^{-1}),$$

$$
\begin{aligned}
p(\boldsymbol{u},\boldsymbol{\alpha},\boldsymbol{\gamma}) &= p(\boldsymbol{u})p(\boldsymbol{\alpha}|\boldsymbol{\gamma})p(\boldsymbol{\gamma}) = \\
&\prod_{x=1}^{n_x} \mathrm{N}(\boldsymbol{u}_x;0,I_K)\mathrm{Gamma}(\gamma_x;1,1) \\
&\prod_{s=1}^{n_s} \mathrm{N}(\boldsymbol{\alpha}_s;0,\gamma_\alpha^{-1}I_K)\ \mathrm{Gamma}(\gamma_\alpha;1,1), \\
q(\boldsymbol{u},\boldsymbol{\alpha},\boldsymbol{\gamma}) &= q(\boldsymbol{u})q(\boldsymbol{\alpha})q(\boldsymbol{\gamma}) = \\
&\prod_{x=1}^{n_x} \mathrm{N}(\boldsymbol{u}_x;\boldsymbol{\mu}_x,\Sigma_x)\mathrm{Gamma}(\gamma_x;\lambda_x,1/r_x) \\
&\prod_{s=1}^{n_s} \mathrm{N}(\boldsymbol{\alpha}_s;\boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{p}_s),\Sigma_{\boldsymbol{w}}(\boldsymbol{p}_s))\ \mathrm{Gamma}(\gamma_\alpha;\lambda_\alpha,1/r_\alpha).
\end{aligned}
$$

Below, we present the update equations for the variational parameters.

a) Update $\boldsymbol{u}$: Using the conjugacy property of normal distributions, we can derive the closed form of variational parameters for $\boldsymbol{u}_x$ as follows:

$$
\begin{aligned}
\Sigma_x &= \left(<\gamma_x> \sum_{s=1}^{n_s} <\boldsymbol{\alpha}_s\boldsymbol{\alpha}_s^T> + I_K \right)^{-1}, \\
\boldsymbol{\mu}_x &= \Sigma_x \big[<\gamma_x> \sum_{s=1}^{n_s} \tilde{f}_{sx} <\boldsymbol{\alpha}_s> \big], \\
<\gamma_x> &= \lambda_x/r_x, \\
<\boldsymbol{\alpha}_s> &= \boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{p}), \\
<\boldsymbol{\alpha}_s\boldsymbol{\alpha}_s^T> &= \boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{p}_s)\boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{p}_s)^T + \Sigma_{\boldsymbol{w}}(\boldsymbol{p}_s), \qquad (10)
\end{aligned}
$$

where $<\cdot>$ denotes expectation with respect to the variational distributions.

b) Update $\boldsymbol{\gamma}$: Similar to $\boldsymbol{u}$, we exploit the conjugacy to obtain the variational parameters for both γ_x and γ_α. For γ_x, we have:

$$
\begin{aligned}
\lambda_x &= 1 + n_s/2, \qquad (11) \\
r_x &= 1 + \frac{1}{2}\sum_{s=1}^{n_s} <(\tilde{f}_{sx} - \boldsymbol{u}_x^T\boldsymbol{\alpha}_s)^2>,
\end{aligned}
$$

where the expectations in the second line can be calculated using the following equations:

$$
\begin{aligned}
<\boldsymbol{u}_x> &= \boldsymbol{\mu}_x, \qquad (12) \\
<\boldsymbol{u}_x\boldsymbol{u}_x^T> &= \boldsymbol{\mu}_x\boldsymbol{\mu}_x^T + \Sigma_x, \\
<\boldsymbol{\alpha}_s^T A\boldsymbol{\alpha}_s> &= \boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{p}_s)^T A\boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{p}_s) + tr(A\Sigma_{\boldsymbol{w}}(\boldsymbol{p}_s)).
\end{aligned}
$$

Similarly, for γ_α, we can update the variational distribution's parameters as:

$$
\begin{aligned}
\lambda_\alpha &= 1 + \frac{n_s K}{2}, \qquad (13) \\
r_\alpha &= 1 + \frac{1}{2}\sum_{s=1}^{n_s} <\boldsymbol{\alpha}_s^T\boldsymbol{\alpha}_s>.
\end{aligned}
$$

Variational inference alternates the updates of these model parameters ($\boldsymbol{u}$ and $\boldsymbol{\gamma}$), and parameters of the mapping ($\boldsymbol{\alpha}$) by: $\log q(\theta_i) \propto \mathbb{E}_{q(-i)}[\log p(\mathcal{D},\boldsymbol{\theta})]$, where $\boldsymbol{\theta} = \{\boldsymbol{u},\boldsymbol{\gamma},\boldsymbol{\alpha}\}$, $\theta_i \in \boldsymbol{\theta}$ denotes the parameter of interest to update, and $(-i)$ indicates the set of all the remaining parameters over which the expectation is taken with respect to the variational distribution, and $p(\mathcal{D},\boldsymbol{\theta})$ is the joint density of data and model parameters. The main implementation difference with different mappings is to update $\boldsymbol{\alpha}$ according to the model.

c) Update $\boldsymbol{\alpha}$: To update the parameters of the mapping from the inputs to the variational distribution, we optimize the evidence lower-bound with respect to the parameters of $q(\boldsymbol{\alpha})$, where the objective function can be expressed as:

$$
\begin{aligned}
&\mathcal{L}(\boldsymbol{w}|\mathcal{D}) = \mathbb{E}_{q(\boldsymbol{\alpha})q(\boldsymbol{u})q(\boldsymbol{\gamma})}\Big[\log\prod_{s,x}\mathrm{N}(\tilde{f}_{sx};\boldsymbol{u}_x^T\boldsymbol{\alpha}_s,\gamma_x^{-1})\Big] - \\
&\mathbb{E}_{q(\boldsymbol{u})q(\boldsymbol{\gamma})}\Big[\textstyle\sum_s \mathrm{KL}\big[\mathrm{N}(\boldsymbol{\alpha}_s;\boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{p}_s),\Sigma_{\boldsymbol{w}}(\boldsymbol{p}_s))||\mathrm{N}(\boldsymbol{\alpha}_s;0,\gamma_\alpha^{-1}I_K)\big]\Big], \qquad (14)
\end{aligned}
$$

where $\mathrm{KL}(p||q) = \int p(x)\log\frac{p(x)}{q(x)}dx$ denotes the Kullback-Leibler (KL) divergence. Note here that the functional relationships from the input parameters to the variational distribution is through $\boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{p})$ and $\Sigma_{\boldsymbol{w}}(\boldsymbol{p})$, which can be modeled flexibly with different complexity levels to balance the model expressiveness and computational as well as sample complexity. In this paper, we illustrate two implementation options with simple linear models and non-parametric neural networks.

2) Bayesian POD with linear mappings (BayPOD-LM): In the first scenario, we employ a linear mapping from the input $\boldsymbol{p}$ to $\boldsymbol{\alpha}$, which we call **BayPOD-LM**, with

$$
q(\boldsymbol{\alpha}_s) = \mathrm{N}(\boldsymbol{\alpha}_s;\boldsymbol{\mu}_{\boldsymbol{w}}(\boldsymbol{p}_s),\Sigma_{\boldsymbol{w}}(\boldsymbol{p}_s)) = \mathrm{N}(\boldsymbol{\alpha}_s;\boldsymbol{W}\boldsymbol{p}_s,\Sigma_{\boldsymbol{\alpha}}), \qquad (15)
$$

where $\boldsymbol{W} \in \mathbb{R}^{K\times m}$. Note that here, the mean in (7) is a linear function of the input parameters and the covariance is shared across different inputs. For BayPOD-LM, we can express (14) by keeping only the terms depending on $\boldsymbol{W}$ and $\Sigma_{\boldsymbol{\alpha}}$ as:

$$
\begin{aligned}
\mathcal{L} &= \sum_{s=1}^{n_s} <\gamma_\alpha> \boldsymbol{p}_s^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{p}_s + \frac{n_s}{2}\log|\Sigma_{\boldsymbol{\alpha}}| \qquad (16) \\
&- \sum_{s,x}\frac{<\gamma_x>}{2}\big[\boldsymbol{p}_s^T\boldsymbol{W}^T <\boldsymbol{u}_x\boldsymbol{u}_x^T> \boldsymbol{W}\boldsymbol{p}_s \\
&+ tr(<\boldsymbol{u}_x\boldsymbol{u}_x^T>\Sigma_{\boldsymbol{\alpha}}) - 2\tilde{f}_{sx}\boldsymbol{p}_s^T\boldsymbol{W}^T <\boldsymbol{u}_x>\big],
\end{aligned}
$$

where $|\cdot|$ is the determinant. We can calculate the gradients of the objective function in (16) with respect to $\boldsymbol{W}$ and $\Sigma_{\boldsymbol{\alpha}}$ in closed form. Hence, we have the following update equations integrated into the inference procedure of general BayPOD:

a) Update $\boldsymbol{\alpha}$: To update the parameters of the linear model which outputs $\boldsymbol{\alpha}$:

$$
\begin{aligned}
\Sigma_{\boldsymbol{\alpha}} &= \Big(\sum_x <\gamma_x><\boldsymbol{u}_x\boldsymbol{u}_x^T> + <\gamma_\alpha> I_K\Big)^{-1}, \qquad (17) \\
\boldsymbol{W} &= \Sigma_{\boldsymbol{\alpha}}\big(\sum_{s,x} <\gamma_x> \tilde{f}_{sx} <\boldsymbol{u}_x> \boldsymbol{p}_s^T\big)\big(\sum_s \boldsymbol{p}_s\boldsymbol{p}_s^T\big)^{-1}.
\end{aligned}
$$

3) Bayesian POD with neural networks (BayPOD-NN): The linearity assumption for the mean in (7) is potentially limiting the model expressiveness. Therefore, as a more flexible model, we let $\boldsymbol{\mu}_{\boldsymbol{w}}(\cdot)$ and $\Sigma_{\boldsymbol{w}}(\cdot)$, i.e. the mean and covariance matrix mapping, take the form of a neural network with weights $\boldsymbol{w}$ and input parameters $\boldsymbol{p}$. Neural networks have been widely used as the inference model in amortized variational inference [2], [12], [19], [24]. We denote this model with **BayPOD-NN**. Note that here, both the mean and covariance matrix are flexible functions of the input parameters. For BayPOD-NN, the corresponding inference procedure adopts the following amortized variational inference to update $\boldsymbol{\alpha}$:

a) Update $\boldsymbol{\alpha}$*:* To update the parameters of the neural network, which outputs the variational distribution over $\boldsymbol{\alpha}$, we adopt the stochastic gradient variational Bayes (SGVB) algorithm [12] to optimize (14).

The parameters of the mapping from the inputs to the variational distribution (i.e. $\boldsymbol{w}$) and the other variational parameters are alternately updated in the inference procedure.

IV. Heated Rod Example

The first case study considers predicting the evolution of the temperature in a one-dimensional heated rod given time-dependent boundary conditions. Our output quantity of interest is the discretized temperature distribution along a rod of length L. Having the initial conditions and the specified boundary conditions, the evolution of the temperature field f over the rod is governed by the heat equation with the diffusivity parameter, κ as an input:

$$\frac{\partial f}{\partial t} = \kappa \frac{\partial^2 f}{\partial \boldsymbol{x}^2}. \tag{18}$$

The temperature field varies as a function of time and distance along the rod. The initial condition and Dirichlet boundary conditions are defined as $f(\boldsymbol{x} = 0, t) = 3\sin(2t)$, $f(\boldsymbol{x} = L, t) = 3$, $f(\boldsymbol{x}, t = 0) = 0$, where we have a time-varying boundary condition at the left end of the rod and a fixed temperature value at the right end.

The input parameter vector for this example is $\boldsymbol{p} = [t, \kappa]$, and each snapshot is a discretized temperature field with $n_x = 200$ that corresponds to a set of values for the input vector, i.e. a fixed diffusivity and time point. Each entry in a snapshot vector represents a spatial location along the rod.

Similar as in [21], we incorporate the constraints to satisfy the (time-dependent) boundary conditions through a particular solution $\bar{\boldsymbol{f}}$ as in (2). For this, we can solve two auxiliary problems, one with boundary conditions $f(\boldsymbol{x} = 0, t) = 0$, $f(\boldsymbol{x} = L, t) = 1$ to get the steady-state solution $\bar{f}_L(\boldsymbol{x})$, and the other with boundary conditions $f(\boldsymbol{x} = 0, t) = 1$, $f(\boldsymbol{x} = L, t) = 0$ to get the steady-state solution $\bar{f}_0(\boldsymbol{x})$. The particular solution can then be defined as

$$\bar{\boldsymbol{f}} = 3\sin(2t)\bar{\boldsymbol{f}}_0(\boldsymbol{x}) + 3\bar{\boldsymbol{f}}_L(\boldsymbol{x}). \tag{19}$$

By subtracting the particular solution that corresponds to a snapshot from it, we get a modified snapshot with homogeneous boundary conditions. The different POD learning methods are then applied to the modified (training) snapshots to learn the reduced-order models and predict the temperature field for the unseen (test) snapshots satisfying the homogeneous boundary conditions. Adding the corresponding particular solution to each snapshot prediction guarantees satisfying the original inhomogeneous boundary conditions.

Snapshots are generated for six different diffusivity parameters κ =[0.25, 0.35, 0.45, 0.55, 0.65, 0.75]. For solving the heat equation, 628 equally spaced temporal points in $[0, 2\pi]$ are used, and 157 time points are randomly selected for snapshot generation. Overall, we have 942 snapshots ($n_s = 942$) corresponding to the 6 diffusivity values ($n_p = 6$) at 157 different time points ($n_t = 157$). We solve the two auxiliary problems to get $\bar{f}_L(\boldsymbol{x})$ and $\bar{f}_0(\boldsymbol{x})$ only once for $\kappa = 2$. Although this introduces an approximation, it does not affect the enforcement of boundary conditions through the particular solution and the homogeneous boundary conditions of the modified snapshots.

Method	mean	std	min	max
Polynomial Regression	0.846	0.370	0.213	1.685
BayPOD-LM	0.847	0.367	0.198	1.687
Neural Net Regression	0.041	0.019	0.004	0.079
BayPOD-NN	0.030	0.017	0.003	0.067

TABLE I: Mean, standard deviation (std), minimum (min), and maximum (max) of mean absolute errors on all the different testing snapshots by Polynomial Regression, BayPOD-LM, Neural Network Regression, and BayPOD-NN for the heated rod case study.

For evaluating different methods, 31 of the generated snapshots are randomly selected and withheld from POD learning as the test set. The competing methods are trained on the remaining snapshots and make predictions for the withheld test snapshots. We set the dimension of POD bases, K, to be 5, which is the lowest number that results in less than 1% reconstruction error of the training snapshots by the classical POD analysis.

A. Results and discussion

We test the performance of the proposed BayPOD-LM and BayPOD-NN for this case study and compare them with the original two-step approach using the corresponding polynomial regression (quadratic) and neural network regression (NN Regression) in [21]. In BayPOD-LM, the same polynomial features that are utilized in the original two-step Polynomial Regression approach are used to have quadratic regression and the variational parameters are initialized with the corresponding parameters from the original approach. In other words, in BayPOD-LM, for each snapshot $\boldsymbol{p} = [\kappa, t, t\kappa, \kappa^2, t^2]$ as in the original two-step Polynomial Regression to have a fair comparison. Both BayPOD-NN and the two-step NN Regression employ the same NN architecture, having two hidden layers each with 50 nodes and ReLU activation functions. BayPOD-NN has outputs with the softplus activation for the covariance of $\boldsymbol{\alpha}$ ($\Sigma_{\boldsymbol{w}}(\cdot)$) in addition to the outputs for the mean of $\boldsymbol{\alpha}$ ($\boldsymbol{\mu}_{\boldsymbol{w}}(\cdot)$) for uncertainty quantification.

We calculate the mean absolute prediction error of each method for each test snapshot. The mean, standard deviation, minimum, and maximum values of the mean absolute errors of each method are provided in Table I. Moreover, four test snapshots as representatives of the different patterns observed in all the test snapshots, their corresponding predictions by all the methods, and uncertainty estimates (as shaded regions) from the proposed BayPOD methods are shown in Figure 2.

BayPOD-LM and the two-step Polynomial Regression have virtually the same error statistics as shown in Table I. In Figure 2, we can see that models with the quadratic mapping from the inputs to the projection coefficients, i.e. Polynomial Regression and BayPOD-LM, have a relatively higher error compared with the methods with a more flexible mapping

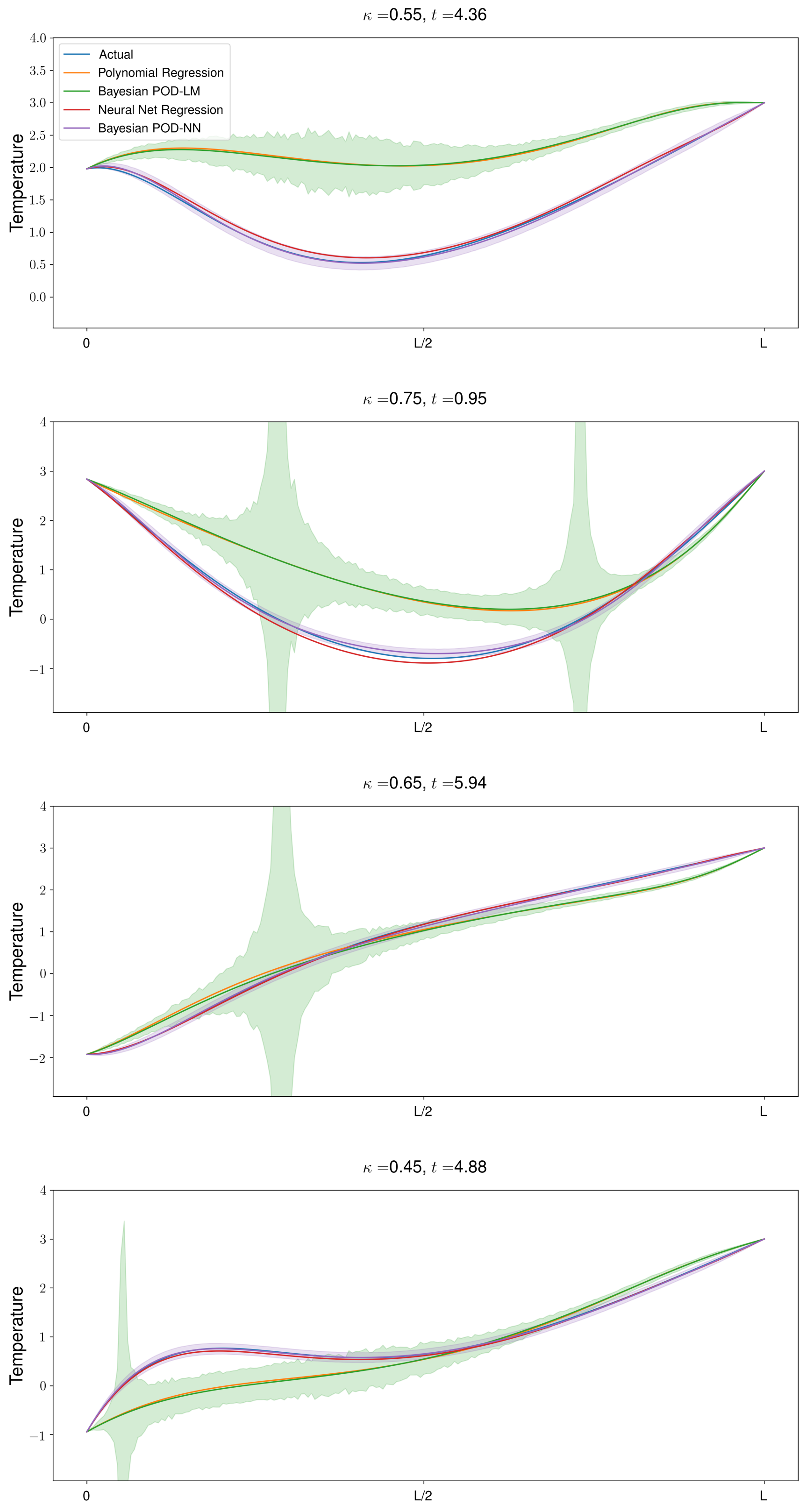


Fig. 2: Four testing examples of comparing the actual temperature field snapshots and predictions from Polynomial Regression, BayPOD-LM, Neural Network Regression, and BayPOD-NN.

using neural networks. One of the critical advantages of BayPOD-LM is its uncertainty quantification capability compared with the two-step Polynomial Regression. The estimated 95% posterior confidence intervals by BayPOD-LM in Figure 2 for the bottom three testing examples include the true temperature values across many spatial points. Moreover, in the top plot of Figure 2, although both BayPOD-LM and Polynomial Regression have very large errors and the 95% posterior confidence interval from BayPOD-LM is far from the true values, we can see that the prediction uncertainty is highly correlated with the prediction error over the length of the rod.

Table I and Figure 2 clearly show the advantage of the more flexible BayPOD-NN modeling compared with BayPOD-LM. It is clear that BayPOD-NN can achieve significantly more accurate predictions in addition to narrower and better uncertainty estimates. Both BayPOD-NN and NN Regression outperform the corresponding linear parametrized models. From Table I, we see that BayPOD-NN also performs better than the two-step NN Regression in terms of the error statistics while providing uncertainty quantification as shown in Figure 2. We can see in all testing examples in Figure 2 that the estimated 95% posterior confidence intervals by BayPOD-NN contain the true temperature values in all the spatial locations for the four depicted snapshots. The results of this case study clearly show that BayPOD-NN is more accurate than the deterministic two-step NN regression while providing principled and accurate uncertainty estimates.

V. Airfoil Example

This case study considers learning a Bayesian reduced-order model for the prediction of the flow around an airfoil. The learned model can predict the flow for new input settings without running the full-order PDE solvers. Both the POD bases and input mappings are trained using data generated from a large-scale computational fluid dynamics (CFD) simulator for the NACA 0012 airfoil [16].

The input parameters for this example are the freestream Mach number, M, and the airfoil lift coefficient, c_l. Our input parameter vector is $\boldsymbol{p} = [M, c_l] \in \mathbb{R}^2$. The output quantity of interest is the pressure field around the airfoil, which varies as a function of the input parameters. In this example, we use the `SU2` CFD tool suite, a multi-purpose open-source solver, specifically developed for aerospace applications. `SU2` uses a finite volume method to discretize the underlying partial differential equations. Here we use the Euler equations to model the inviscid steady flow over the airfoil. We consider a range of Mach numbers, spanning subsonic and transonic flow regimes. Flow tangency boundary conditions are imposed on the airfoil surface and the farfield boundary is approximately 20 chord lengths away from the airfoil.

`SU2`'s discretization of the pressure field has $n_x = 9027$ degrees of freedom; that is, each SU2 pressure field solution is a vector of dimension $n_x = 9027$, where each entry corresponds to the predicted pressure at a different spatial location in the computational domain.

We refer to each pressure field solution vector as a snapshot. Snapshots are generated by the CFD solver for the NACA 0012 airfoil and a domain of Mach numbers from $M = 0.6$ to $M = 0.8$ in increments of 0.01. At each Mach number, the following seven lift coefficients are used: $c_l = [0.4, 0.5, 0.6, 0.7, 0.8, 0.85, 0.9]$. This provides a total of $n_s = 147$ snapshots, where each snapshot is a high-fidelity pressure field solution, represented as a high-dimensional vector.

For evaluation, we withhold all data corresponding to a single Mach number, train the models with the remaining data, and test on the withheld data one sample at a time. We set the dimension of POD bases, K, to be 20, which is the lowest number that results in less than 1% reconstruction error of the training snapshots by the classical POD analysis for all the data splits.

A. Results of BayPOD-LM and discussion

Figure 3 illustrates the comparison of BayPOD-LM after five iterations of updates with the original results using polynomial regression (quadratic) in [21]. In this figure, for each Mach number, the plots show the minimum, mean, and maximum of the mean absolute error (MAE) over the entire field, across the seven lift coefficient values. Note that in BayPOD-LM, we use polynomial combinations of the features with degree less than or equal to 2 as in Polynomial Regression, and initialize the variational parameters with the corresponding parameters from the original two-step approach. We can see from the figure that BayPOD-LM has a similar or slightly better performance compared with the two-step Polynomial Regression for the different Mach numbers.

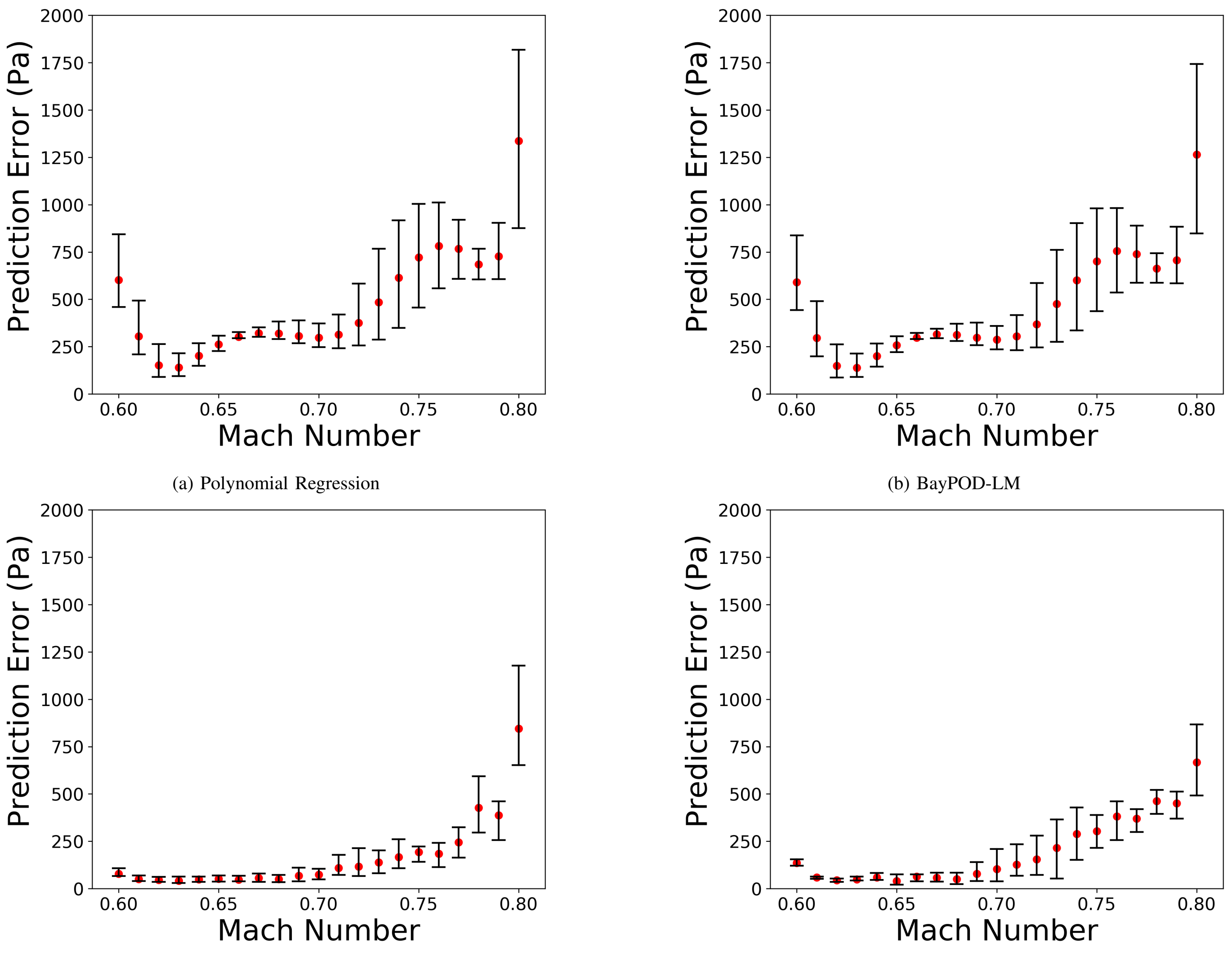
For the case when the Mach number $M = 0.7$ and lift coefficient $c_l = 0.7$, Figure 4 shows the point-wise absolute error of the field predictions. All pressure fields produced with Mach 0.7 have been held out of the training set used by each method for making the predictions. This figure again shows that BayPOD-LM performs slightly better in terms of MAE. Critically, the advantage of BayPOD-LM is its uncertainty quantification capability. In Figure 5, the point-wise posterior predictive standard deviation times two is illustrated, where the regions with higher uncertainty are overlapping with some of the regions with the highest MAE in Figure 4, demonstrating the effectiveness of the uncertainty quantification capability of BayPOD.

B. Results of BayPOD-NN and discussion

Next, we test the more flexible BayPOD-NN with the same setup of this case study. We use a NN with two hidden layers, each with 50 nodes and ReLU activation functions, shared by both $\boldsymbol{\mu}_{\boldsymbol{w}}(\cdot)$ and $\Sigma_{\boldsymbol{w}}(\cdot)$. The output layer for the mean inference network does not have any activation while we use softplus for the covariance network.

For direct comparison, we here illustrate the results with the same Mach number $M = 0.7$ and lift coefficient $c_l = 0.7$. The point-wise absolute error of the field predictions by BayPOD-NN and the posterior predictive standard deviation times two as a measure of uncertainty are provided in Figure 6. We can clearly see that BayPOD-NN improves upon BayPOD-LM.

Moreover, we compare BayPOD-NN with the original Polynomial Regression approach and also Neural Network Regression (NN Regression) in terms of the error statistics over the entire field for the lift coefficients and the different



(a) Polynomial Regression

(b) BayPOD-LM

(c) Neural Network Regression

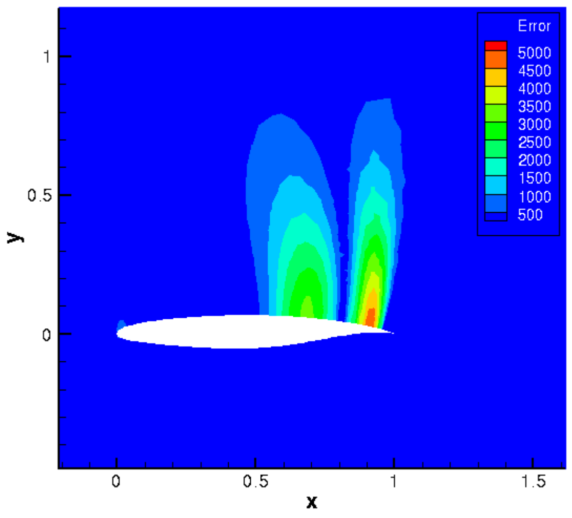
(d) BayPOD-NN

Fig. 3: The minimum, mean, and maximum mean absolute error across lift coefficients for each Mach number by Polynomial Regression, BayPOD-LM, Neural Network Regression, and BayPOD-NN for the airfoil case study.

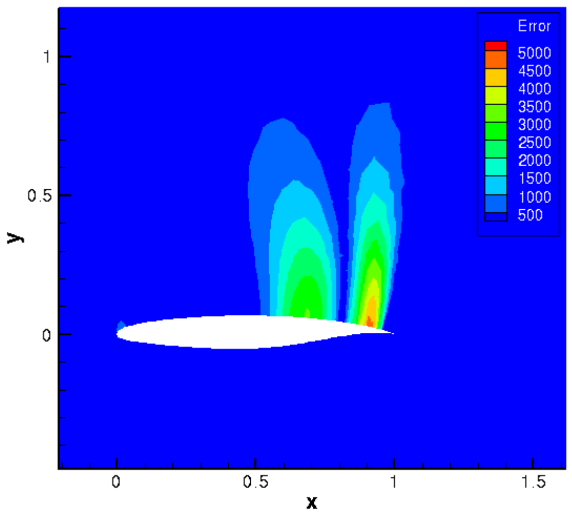
Mach numbers, as shown in Figure 3(d). The figure depicts the advantage of the more flexible BayPOD-NN modeling compared with the linear model in BayPOD-LM in Figure 3(b), where we see consistent improvement for all Mach numbers.

In Figure 3(c) and (d), the two-step NN Regression approach and BayPOD-NN overall show comparable performance in terms of the error statistics over the entire field for the lift coefficients and different Mach numbers. The NN architecture is the same for both NN Regression and BayPOD-NN (i.e. two hidden layers with width 50), with BayPOD-NN having the additional outputs corresponding to the covariance of $\boldsymbol{\alpha}$ ($\Sigma_{\boldsymbol{w}}(\cdot)$). We can see that for smaller Mach numbers their error statistics are virtually the same, while for a few of the mid-range Mach numbers NN Regression has slightly better error statistics and for larger Mach numbers BayPOD-NN performs better. These results clearly show that BayPOD-NN is a flexible unified approach for learning projection bases and coefficients that does not lose accuracy compared with the deterministic two-step NN Regression, while providing a mechanism for principled input-dependent uncertainty estimates. It is worth emphasizing that as opposed to the NN Regression approach where NNs are used as the regressor in the two-step procedure, for BayPOD-NN, we are integrating NNs in the variational distribution, where in addition to providing uncertainty estimates, the structure of the model and the prior distributions automatically impose inherent model regularization obviating the need for additional fine-tuned regularization.

It is worth noting that BayPOD-NN does not have much higher complexity than NN Regression at prediction/inference. BayPOD-NN only requires one pass through the neural network to get the variational parameters for the coefficients ($\boldsymbol{\alpha}$) to reconstruct the homogeneous component of the field. Furthermore, we have the additional benefit of having full access to the posterior distribution of $\tilde{\boldsymbol{f}}$. BayPOD-NN has computation overhead during training. We have provided more detailed discussions in the Appendix. Additionally, results for another two-step baseline with fixed bases and a neural network mapping from inputs to coefficients with MC



(a) Polynomial Regression



(b) BayPOD-LM

Fig. 4: The error field produced by the deterministic polynomial regression and our BayPOD predictions

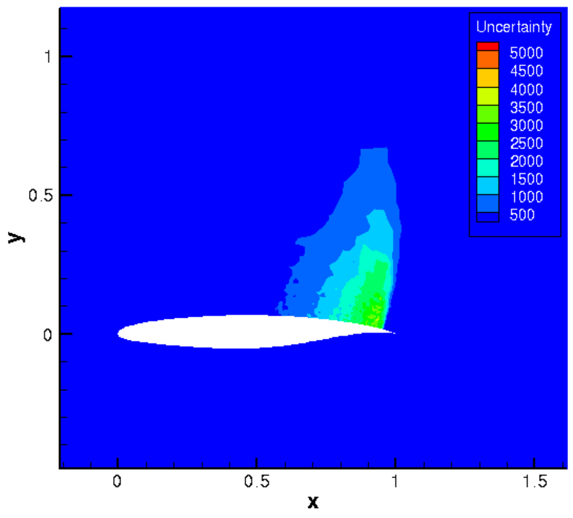


Fig. 5: Uncertainty quantification map by visualizing twice of the posterior predictive standard deviation from BayPOD-LM.

Dropout [5] are included in the Appendix. The results show that BayPOD-NN outperforms this baseline in terms of both accuracy and uncertainty estimate.

VI. Conclusions

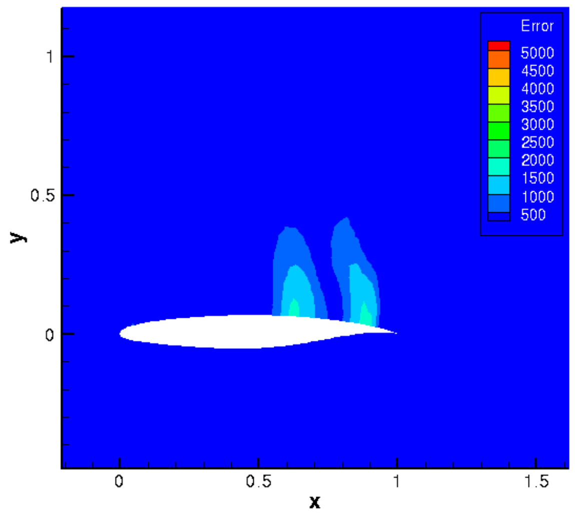
The critical contributions of developing learnable Bayesian reduced-order models are to not only seek the best low-dimensional subspace for approximating high-dimensional dynamics, but also allow uncertainty estimates by learning distributions of reduced-order model parameters. By modeling both projections and mappings from system inputs to projection coefficients in one unified model with seamless integration of Bayesian inference for both components, our experimental results with the heated rod and airfoil examples clearly show the advantages over non-Bayesian reduced-order models on both prediction accuracy and uncertainty estimation of high-dimensional system dynamics. With the developed BayPOD framework, Bayesian experimental design guided by efficient predictive models constrained by scientific principles can be developed for science and engineering applications where training data are difficult or costly to generate and the involved decision making based on predictions can have significant consequences, which are the main challenges in the recent emerging scientific machine learning (SciML) research.

Acknowledgments

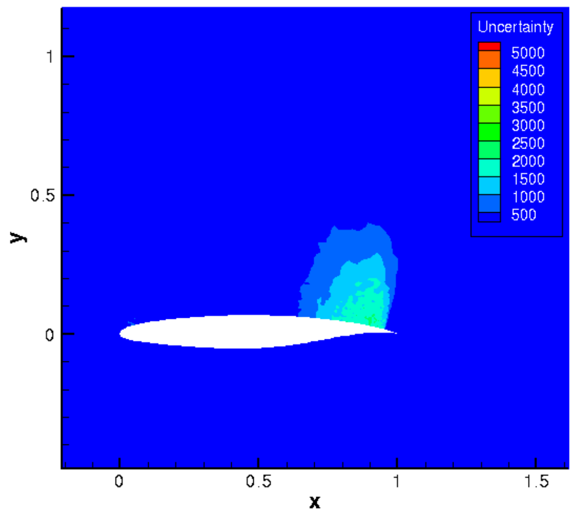
The authors truly appreciate the discussions with Dr. Francis J. Alexander at the Brookhaven National Laboratory and Prof. Karen E. Willcox in the Oden Institute for Computational Engineering and Sciences at the University of Texas at Austin. We thank Prof. Willcox's group, in particular Renee Swischuk, for sharing their code and PDE solution data as *snapshots*. The presented research has been performed when XQ visited the Oden Institute for Computational Engineering and Sciences at the University of Texas at Austin as a J.T. Oden Faculty Fellow. Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.

References

[1] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.
[2] Shahin Boluki, Randy Ardywibowo, Siamak Zamani Dadaneh, Mingyuan Zhou, and Xiaoning Qian. Learnable Bernoulli dropout for Bayesian deep learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3905–3916. PMLR, 2020.
[3] Anindya Chatterjee. An introduction to the proper orthogonal decomposition. *Current science*, pages 808–817, 2000.
[4] Siamak Zamani Dadaneh and Xiaoning Qian. Bayesian module identification from multiple noisy networks. *EURASIP Journal on Bioinformatics and Systems Biology*, 2016(1):5, 2016.



(a) Error field



(b) Uncertainty map

Fig. 6: The prediction results by BayPOD-NN

[5] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[6] Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020.

[7] Mengwu Guo, Shane A McQuarrie, and Karen E Willcox. Bayesian operator inference for data-driven reduced-order modeling. *arXiv preprint arXiv:2204.10829*, 2022.

[8] Trevor Hastie. Ridge regularization: An essential concept in data science. *Technometrics*, 62(4):426–433, 2020.

[9] Seth M Hirsh, David A Barajas-Solano, and J Nathan Kutz. Sparsifying priors for bayesian uncertainty quantification in model discovery. *Royal Society Open Science*, 9(2):211823, 2022.

[10] Philip Holmes, John L Lumley, Gahl Berkooz, and Clarence W Rowley. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 2012.

[11] Sharmila Karumuri, Rohit Tripathy, Ilias Bilionis, and Jitesh Panchal. Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks. *Journal of Computational Physics*, 404:109120, 2020.

[12] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[13] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[15] KK Nomura and SE Elghobashi. The structure of inhomogeneous turbulence in variable density nonpremixed flames. *Theoretical and Computational Fluid Dynamics*, 5(4-5):153–175, 1993.

[16] Max M. J. Opgenoord, Mark Drela, and Karen E. Willcox. Physics-based low-order model for transonic flutter prediction. *AIAA Journal*, 56(4):1519–1531, 2018.

[17] Dimitris C Psichogios and Lyle H Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992.

[18] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[19] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

[20] L Sirovich. Turbulence and the dynamics of coherent structures. part 1: Coherent structures. *Quart Appl Math*, 45:561–571, 1987.

[21] Renee Swischuk, Laura Mainini, Benjamin Peherstorfer, and Karen Willcox. Projection-based model reduction: Formulations for physics-based machine learning. *Computers & Fluids*, 179:704–717, 2019.

[22] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[23] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.

[24] Siamak Zamani Dadaneh, Shahin Boluki, Mingzhang Yin, Mingyuan Zhou, and Xiaoning Qian. Pairwise supervised hashing with bernoulli variational auto-encoder and self-control gradient estimator. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 540–549. PMLR, 2020.

[25] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

Appendix A
Discussion on Computational Complexity

Here we discuss the complexity and run time of the proposed methods in comparison with the baselines. Both BayPOD-LM and BayPOD-NN have the same complexity and run time as their respective baselines, Polynomial Regression and Neural Network Regression, for prediction/inference. During inference for BayPOD-NN, only one pass of the neural network is required to obtain the parameters of the basis coefficients $\boldsymbol{\alpha}$. The estimated $\boldsymbol{\alpha}$ is then leveraged to reconstruct the homogeneous component of the field. Moreover, we have the extra benefit of having full access to the posterior distribution of $\tilde{\boldsymbol{f}}$. Obtaining samples from the predictive distribution can also be easily done by regular sampling from normal distributions without any additional run of the neural network.

The proposed methods involve additional computation during training which increases training time. As mentioned in the case studies in sections V and IV, we run BayPOD-LM updates only for a few iterations after initializing the parameters with regular regression, and since all the updates are in closed forms, the overhead is negligible. For BayPOD-NN, we observe that each iteration in training takes twice and five times the iteration time of the baseline for the heated rod and airfoil examples, respectively. As the iteration times are in the order of thousandth or hundredth of a second, the overall training time is reasonable and in the order of a few minutes. The evolution of training error versus training iterations is shown in Figure 7 for the heated rod example and one training run of the airfoil case study.

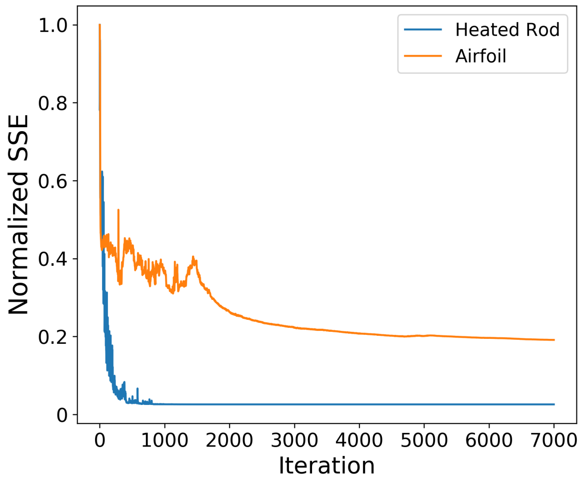


Fig. 7: Normalized training sum of squared errors (SSE) for BayPOD-NN versus the training iteration

Appendix B
Additional Baseline for Airfoil Example

In this section we provide results for an additional baseline for the airfoil example, where Neural Network Regression is combined with MC Dropout [5]. In particular, we add dropout to the hidden layers of the neural network architecture used for the map from inputs to the POD coefficients in Neural Network Regression, where dropout is active both during training and prediction. The dropout rates are tuned based on the data. In this approach, the POD bases are learned first and fixed as in Neural Network Regression, but the map from inputs to coefficients can provide some form of uncertainty estimate, albeit in a limited fashion. This is in contrast to BayPOD-NN which learns the bases and the coefficient mapping simultaneously in a principled probabilistic way. We refer to this baseline as Neural Network Regression + MC Dropout.

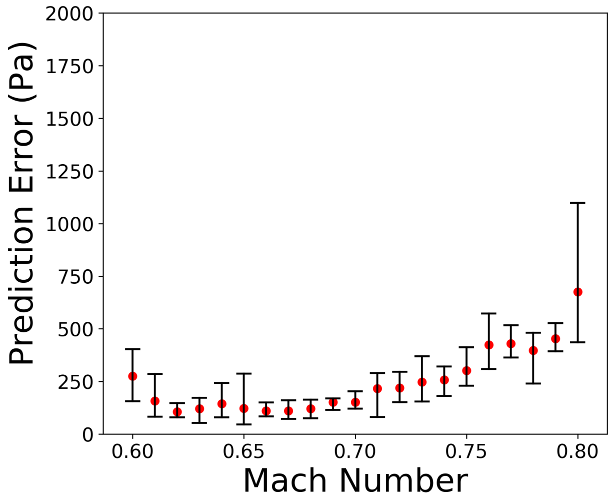
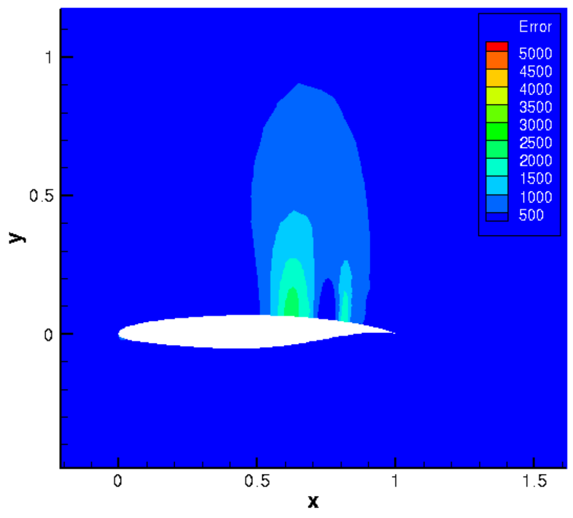
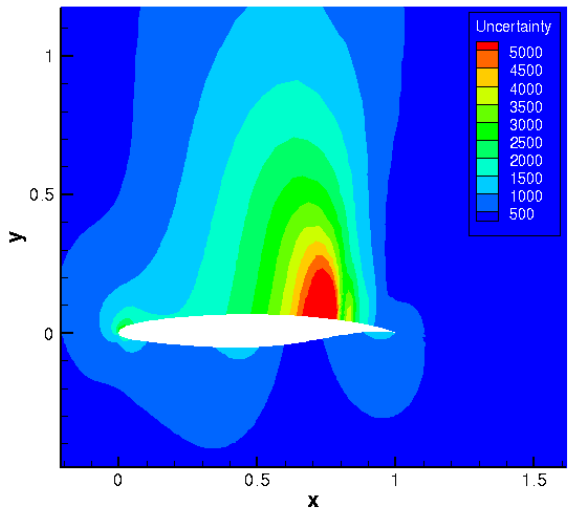


Fig. 8: The minimum, mean, and maximum mean absolute error across lift coefficients for each Mach number by Neural Network Regression + MC Dropout

The field prediction error statistics across the lift coefficients for each Mach number are shown in Figure 8. We can see from the Figure and by comparing it to Figure 3 that for some of the Mach numbers the error statistics across the different lift coefficients are worse than BayPOD-NN and Neural Network Regression. Moreover, in Figure 9, point-wise absolute error of the field predictions and the predictive standard deviation times two as a measure of uncertainty are shown for Mach number $M = 0.7$ and lift coefficient $c_l = 0.7$. The Figure is directly comparable with Figure 6, which shows the same results for BayPOD-NN. It is clear that the error of Neural Network Regression + MC Dropout is somewhat larger than BayPOD-NN and its uncertainty estimate is very wide, even in locations with very small prediction errors.



(a) Error field



(b) Uncertainty map

Fig. 9: The prediction results by Neural Network Regression + MC Dropout