Mesh-clustered Gaussian process emulator for partial differential equation boundary value problems

Chih-Li Sung*[,a], Wenjia Wang*[,b], Liang Ding[c], Xingjian Wang[d]

[a]Michigan State University
[b]Hong Kong University of Science and Technology (Guangzhou)
[c]Fudan University [d]Tsinghua University

Abstract

Partial differential equations (PDEs) have become an essential tool for modeling complex physical systems. Such equations are typically solved numerically via mesh-based methods, such as finite element methods, with solutions over the spatial domain. However, obtaining these solutions are often prohibitively costly, limiting the feasibility of exploring parameters in PDEs. In this paper, we propose an efficient emulator that simultaneously predicts the solutions over the spatial domain, with theoretical justification of its uncertainty quantification. The novelty of the proposed method lies in the incorporation of the mesh node coordinates into the statistical model. In particular, the proposed method segments the mesh nodes into multiple clusters via a Dirichlet process prior and fits Gaussian process models with the same hyperparameters in each of them. Most importantly, by revealing the underlying clustering structures, the proposed method can provide valuable insights into qualitative features of the resulting dynamics that can be used to guide further investigations. Real examples are demonstrated to show that our proposed method has smaller prediction errors than its main competitors, with competitive computation time, and identifies interesting clusters of mesh nodes that possess physical significance, such as satisfying boundary conditions. An R package for the proposed methodology is provided in an open repository.

Keywords: Computer Experiments; Mixture Model; Finite Element Method; Error Analysis; Functional Output; Dirichlet process; Uncertainty Quantification.

*These authors contributed equally to the manuscript. CS gratefully acknowledges funding from NSF DMS 2113407. WW gratefully acknowledges funding from NSFC grant 12101149.

1 Introduction

Computer models have become essential to study physical systems that are expensive or infeasible, and have been successfully applied in a variety of scientific research, ranging from cell adhesion (Sung et al., 2020) to designing a rocket injector (Mak et al., 2018). Typically, a physical system is described by a computer model consisting of a series of partial differential equations (PDEs) (Evans, 2010) and evaluated in a two- or three-dimensional space. For instance, in Wang et al. (2018), the governing PDEs are evaluated to simulate the characteristics of a three-dimensional swirling flow in a cylindrical chamber.

These PDEs are typically solved by numerical methods, such as finite element methods (FEMs) or a collocation method (Fornberg and Flyer, 2015), based on a mesh specification in a two- or three-dimensional space, and the numerical solutions are evaluated at these mesh node coordinates (also called *grid points* (Mak et al., 2018; Tan, 2018a)). The number of nodes is usually fairly large to ensure the numerical accuracy of PDE solutions. Such evaluations, however, are often prohibitively costly for input space exploration. For instance, the high-fidelity simulation in Mak et al. (2018) generates around 100,000 grid points and takes six days of computation time for a given input. Thus, it is essential to develop a cheaper statistical *emulator* as a surrogate model that approximates the solutions in a timely fashion.

The paper focuses on developing an efficient emulator for a series of solutions at *many* coordinates in a *PDE boundary value problem*, where the coordinates are fixed across inputs. A PDE boundary value problem involves solving a set of PDEs subject to specified conditions at the boundaries of the domain. A popular statistical emulator for computer simulations is through Gaussian process (GP) modeling (Santner et al., 2018; Gramacy, 2020), which provides a flexible approximation to the relationship between simulation output and inputs and quantifies uncertainty through its predictive variance; however, the GP is mainly for predicting a scalar output and is not directly applicable to the context of many outputs. One idea is to simultaneously emulate the output at each coordinate separately using independent GPs, which is discussed in Qian et al. (2008), Conti and O'Hagan (2010), and Lee et al. (2011, 2012), and another idea is using GPs with a special shared covariance structure (Gu and Berger, 2016). However, these methods do not incorporate the information of mesh node locations into their models, making it challenging to provide

predictions at *any* coordinates in the domain of interest beyond the mesh coordinates. Another idea is to perform dimension reduction to approximate the simulation outputs using basis expansion, such as functional principal component analysis (Ramsay and Silverman, 2005) or Karhunen-Loève expansion (Karhunen, 1947; Loève, 1955), and then fit GP models on the coefficients, the number of which is usually much less than the number of outputs. The methods adopting this idea include Higdon et al. (2008), Rougier (2008), Rougier et al. (2009), Marrel et al. (2011), Mak et al. (2018), and Tan (2018a). Such methods, however, achieve the dimension reduction by a *finite* truncation of the expansion in a function basis, and the approximation error can introduce additional bias to the predictions.

In this paper, we propose a novel emulator, called *mesh-clustered Gaussian process* (`mcGP`) emulator, for predicting the outputs at many fixed coordinates by incorporating the information of mesh node coordinates. Specifically, instead of fitting separate GPs with different hyperparameters at each node coordinate, the proposed method makes use of the divide-and-conquer idea, which segments the node coordinates into *clusters* with a soft-assignment clustering approach, within each of which, GPs with the *same* hyperparameters are fitted at each coordinate. In particular, the Dirichlet process (DP) prior is adopted here, facilitating a flexible clustering structure for the proposed mixture model without the need for specifying a fixed number of clusters. In addition, a basis expansion representation is employed for mesh-based numerical solutions and the GPs are used to model the coefficients. This approach enables us to make predictions across the entire spatial domain, extending beyond the mesh coordinates. Note that, this basis expansion *does not* perform dimension reduction, so no additional bias will be introduced to our predictions. Importantly, given such a sophisticated mixture model, the proposed method can be fitted efficiently by adopting the variational Bayesian inference method (Jordan et al., 1999; Wainwright and Jordan, 2008), which provides an analytical approximation to the posterior distribution of the latent variables and parameters, facilitating faster computation and scalability when compared with traditional approaches like Markov chain Monte Carlo (MCMC).

In addition to efficient emulation, our method also provides two important features. First, our method enables efficient uncertainty quantification for emulation with theoretical guarantees. In particular, we provide the error analysis that not only considers the uncertainty from the emulator given limited training samples, but also accounts for the numerical error arising from the discrete approximation of the continuous domain through a mesh

configuration in numerical methods for solving PDEs. Second, by revealing the clustering structures, the proposed method provides valuable insights into qualitative features of the resulting dynamics. Unlike traditional reduced-basis methods for flow simulations in physics and engineering, such as proper orthogonal decomposition (POD) (Lumley, 1967), which are unsupervised learning methods purely based on the flow data, the latent clustering structure by the proposed method is determined by both inputs and outputs as in Joseph and Mak (2021) and Sung et al. (2023), which can be used to separate a simulated flow into key instability structures, each with its corresponding spatial features.

It is important to note that recent developments in the field of finite element methods have led to the emergence of the *statistical finite elements* (statFEM) approach, which holds promise in advancing uncertainty quantification and model predictions through a Bayesian statistical construction of finite element methods. See, for example, Duffin et al. (2021), Girolami et al. (2021) and Akyildiz et al. (2022). Specifically, statFEM introduces a hierarchical statistical model to handle uncertainties in data, mathematical models, and finite element discretization. It decomposes data into three components—finite element, model misspecification, and noise—each treated as a random variable with a corresponding prior probability density. In contrast, our focus in this paper is primarily on the development of an efficient surrogate model, providing a faster alternative to costly finite element simulations.

The rest of this article is organized as follows. Section 2 provides a brief overview of PDE boundary value problems and mesh-based numerical methods. The proposed model is introduced in Section 3, and its error analysis is derived in Section 4. Real examples are demonstrated in Section 5. Section 6 concludes with directions for future work. Theoretical proofs, and the R (R Core Team, 2018) code for reproducing the numerical results, are provided in Supplementary Materials.

2 PDE boundary value problems and mesh-based numerical methods

PDEs are an essential tool in the description of complex systems drawing from scientific principles. A PDE boundary value problem typically can be expressed by a set of partial differential equations with boundary conditions, and the solutions are often dependent on

certain inputs, denoted as $\mathbf{x} \in \chi \subset \mathbb{R}^p$, typically representing parameters in the equations and boundary conditions, where χ is assumed to be compact and convex. Specifically, a PDE boundary value problem can be written in the form of

$$\begin{cases} \mathcal{L}(u(\mathbf{s}); \mathbf{x}) = f(\mathbf{s}; \mathbf{x}), & \mathbf{s} \in \Omega \\ \mathcal{G}(u(\mathbf{s}); \mathbf{x}) = g(\mathbf{s}; \mathbf{x}), & \mathbf{s} \in \partial\Omega, \end{cases} \tag{1}$$

where Ω is a compact domain in $\mathbb{R}^d$ with a Lipschitz boundary denoted by $\partial\Omega$, $\mathcal{L}(\cdot; \mathbf{x})$ and $\mathcal{G}(\cdot; \mathbf{x})$ are differential operators on Ω and $\partial\Omega$ given the input $\mathbf{x}$, respectively, $f(\cdot; \mathbf{x})$ and $g(\cdot; \mathbf{x})$ are two known functions on Ω and $\partial\Omega$ with the input $\mathbf{x}$, and $u(\mathbf{s})$ is the solution to the partial differential equations (1). Given the input $\mathbf{x}$, the exact solution to the equations when uniqueness holds, denoted by $u_0(\mathbf{s}; \mathbf{x})$, usually cannot be written explicitly. Instead, numerical methods are used to approximate the exact solution, for which mesh-based numerical methods are most widely used, including the finite difference method (FDM), finite volume method (FVM), and finite element method (FEM) (Brenner and Scott, 2007; Tekkaya and Soyarslan, 2019). The extensions to *mesh-free* methods, such as the collocation method (Golberg et al., 1999; Wendland, 1998; Fasshauer, 1999; Wendland, 1999; Fasshauer, 1996), are straightforward and will be discussed in the remarks. Specifically, mesh-based numerical methods subdivide a large system into smaller, simpler parts by a particular space discretization in the space dimensions, which is implemented by the construction of a *mesh* of the object. Figure 1 demonstrates a mesh specification for a 2-dimensional FEM problem, in which the triangular elements connect all characteristic points (called *nodes*) that lie on their circumference, and these connections are mathematically expressed through a set of functions called *shape functions*.

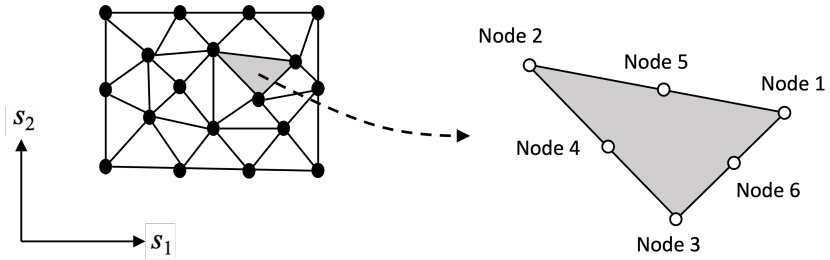

Figure 1: *Introduction of finite element method; similar to one from Tekkaya and Soyarslan (2019).*

Suppose that there are N nodes in this mesh-based numercial method, the coordinates of which are denoted by $\mathbf{S}_N = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N)$, where $\mathbf{s}_j \in \Omega \cup \partial\Omega$. We assume that $\mathbf{S}_N$ is fixed across different inputs $\mathbf{x}$. In the case of an adaptive mesh, where $\mathbf{S}_N$ may vary across different inputs, we explore potential methods in Section 6. The numerical solutions can be expressed as

$$u_N(\mathbf{s}; \mathbf{x}) = \sum_{j=1}^{N} \beta_j(\mathbf{x}) v_j(\mathbf{s}), \tag{2}$$

where $v_j(\mathbf{s})$ is the (given) shape function depending on the discretization, and $\beta_j(\mathbf{x})$ is the corresponding coefficient, which is independent of $\mathbf{s}$. The shape function has the Kronecker Delta property, that is, $v_j(\mathbf{s}_i) = 1$ if $i = j$ and $v_j(\mathbf{s}_i) = 0$ if $i \neq j$. This ensures that the function interpolates the solution at the mesh nodes, i.e., $u_N(\mathbf{s}_j; \mathbf{x}) = \beta_j(\mathbf{x})$ for $j = 1, \ldots, N$. In other words, the solution of the PDE at the mesh node $\mathbf{s}_j$ is equal to $\beta_j(\mathbf{x})$. As an example, for a triangular element with six nodes as in Figure 1, the *quadratic* shape functions, which will be adopted in our later implementation in Section 5, can be defined as:

$$\begin{gathered} v_1(\mathbf{s}) = \xi_1(\mathbf{s})(2\xi_1(\mathbf{s}) - 1), \quad v_2(\mathbf{s}) = \xi_2(\mathbf{s})(2\xi_2(\mathbf{s}) - 1), \quad v_3(\mathbf{s}) = \xi_3(\mathbf{s})(2\xi_3(\mathbf{s}) - 1), \\ v_4(\mathbf{s}) = 4\xi_1(\mathbf{s})\xi_2(\mathbf{s}), \quad v_5(\mathbf{s}) = 4\xi_2(\mathbf{s})\xi_3(\mathbf{s}), \quad v_6(\mathbf{s}) = 4\xi_1(\mathbf{s})\xi_3(\mathbf{s}), \end{gathered} \tag{3}$$

where ξ_1, ξ_2, and ξ_3 represent the barycentric coordinates associated with the three vertices (Nodes 1, 2, and 3, respectively), which are given by $\xi_j(\mathbf{s}) = a_j + b_j s_1 + c_j s_2$, where a_j, b_j, and c_j are determined based on the geometry and orientation of the triangle, and ensure that the shape functions satisfy the property that they are equal to 1 at the corresponding nodes and equal to 0 at the other nodes. This formulation can be easily extended to multiple triangular elements; we refer more detailed information and variations of shape functions to Bathe (2006).

The coefficients are obtained by solving a linear system,

$$L(\mathbf{x})\boldsymbol{\beta}_N(\mathbf{x}) = b(\mathbf{x}), \tag{4}$$

where $\boldsymbol{\beta}_N(\mathbf{x}) = (\beta_1(\mathbf{x}), \ldots, \beta_N(\mathbf{x}))^T$ is a vector of the coefficients, and $L(\mathbf{x}) \in \mathbb{R}^{N \times N}$ is the *stiffness matrix* and $b(\mathbf{x}) \in \mathbb{R}^N$ is the *load vector*, both of which are determined by the

numerical method.

The main challenge of mesh-based numerical methods is that directly solving the PDE boundary value problem for any input $\mathbf{x}$, or equivalently, solving the linear system (4), can be computationally demanding (e.g., the high-fidelity simulation in Mak et al. (2018)). Thus, an efficient emulator that can approximate the solution, $u_N(\mathbf{s};\mathbf{x})$, for any $\mathbf{s} \in \Omega, \mathbf{x} \in \chi$, is called for.

3 Mesh-clustered Gaussian process (mcGP) emulator

From (2), since emulating $u_N(\mathbf{s};\mathbf{x})$ is equivalent to emulating $\{\beta_j(\mathbf{x})\}_{j=1}^N$ (because v_j is a known function), we aim to build an efficient emulator that approximates $\boldsymbol{\beta}_N(\mathbf{x}) := \{\beta_j(\mathbf{x})\}_{j=1}^N$ for any $\mathbf{x} \in \chi$. Suppose that n computer simulations with the inputs, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, are conducted, and their corresponding solutions at the N nodes are $\{\boldsymbol{\beta}_N(\mathbf{x}_i)\}_{i=1}^n$. Clearly, this is a multi-output regression problem, because for each input $\mathbf{x}_i$, the output $\boldsymbol{\beta}_N(\mathbf{x}_i)$ returns a vector of size N, where N can be fairly large. To this end, we propose an efficient emulator that couples over clusters of Gaussian process (GP) emulators, with the aid of the mesh specification to find the clustering structure.

To begin, we briefly introduce the GP emulator in the following subsection.

3.1 Gaussian process (GP) emulator

A GP is a popular tool for building an emulator for computer experiments (Santner et al., 2018; Gramacy, 2020) due to its flexibility and the capability of uncertainty quantification through the predictive distribution. Specifically, suppose that we aim to emulate the single output $\beta_j(\mathbf{x})$, then the function β_j can be assumed to have a GP prior with zero mean and a positive-definite covariance function, $K_j(\cdot,\cdot) : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$. The covariance function often has the form of $K_j(\cdot,\cdot) = \tau_j^2 \Phi_{\boldsymbol{\theta}_j}(\cdot,\cdot)$, where τ_j^2 is a positive scale and $\Phi_{\boldsymbol{\theta}_j}$ is a positive-definite correlation function that depends on some hyperparameters $\boldsymbol{\theta}_j$. We denote such a GP as

$$\beta_j(\mathbf{x})|\tau_j^2, \boldsymbol{\theta}_j \sim \mathcal{GP}(0, \tau_j^2 \Phi_{\boldsymbol{\theta}_j}(\mathbf{x}, \mathbf{x}')).$$

The GP assumes that the random vector $\mathbf{b}_j = (\beta_j(\mathbf{x}_1), \ldots, \beta_j(\mathbf{x}_n))^T$ follows an n-dimensional multivariate normal distribution, $\mathcal{N}_n(0, \tau_j^2 \Phi_{\boldsymbol{\theta}_j}(\mathbf{X}_n, \mathbf{X}_n))$, where $\mathbf{X}_n = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and

$\Phi_{\boldsymbol{\theta}_j}(\mathbf{X}_n, \mathbf{X}_n)$ is an $n \times n$ matrix with $(\Phi_{\boldsymbol{\theta}_j}(\mathbf{X}_n, \mathbf{X}_n))_{i,k} = \Phi_{\boldsymbol{\theta}_j}(\mathbf{x}_i, \mathbf{x}_k)$. To emulate the N outputs $\boldsymbol{\beta}_N(\mathbf{x}) = (\beta_1(\mathbf{x}), \ldots, \beta_N(\mathbf{x}))^T$, we assume that the GPs are independent, implying that the N random vectors, $\mathbf{b}_1, \ldots, \mathbf{b}_N$, are independent.

For a new input $\mathbf{x}$, it can be shown that the posterior predictive distribution is a normal distribution with the mean

$$\mathbb{E}[\beta_j(\mathbf{x})|\mathbf{b}_j, \tau_j^2, \boldsymbol{\theta}_j] = \Phi_{\boldsymbol{\theta}_j}(\mathbf{x}, \mathbf{X}_n)\Phi_{\boldsymbol{\theta}_j}(\mathbf{X}_n, \mathbf{X}_n)^{-1}\mathbf{b}_j \tag{5}$$

and the variance

$$\mathbb{V}[\beta_j(\mathbf{x})|\mathbf{b}_j, \tau_j^2, \boldsymbol{\theta}_j] = \tau_j^2(1 - \Phi_{\boldsymbol{\theta}_j}(\mathbf{x}, \mathbf{X}_n)\Phi_{\boldsymbol{\theta}_j}(\mathbf{X}_n, \mathbf{X}_n)^{-1}\Phi_{\boldsymbol{\theta}_j}(\mathbf{x}, \mathbf{X}_n)^T), \tag{6}$$

where $\Phi_{\boldsymbol{\theta}_j}(\mathbf{x}, \mathbf{X}_n)$ is a $1 \times n$ matrix with $(\Phi_{\boldsymbol{\theta}_j}(\mathbf{x}, \mathbf{X}_n))_{1,i} = \Phi_{\boldsymbol{\theta}_j}(\mathbf{x}, \mathbf{x}_i)$. The posterior predictive mean can be used to predict $\beta_j(\mathbf{x})$ and the posterior predictive variance can be used to quantify the prediction uncertainty.

Two families of correlation functions are widely used in practice, which are the power exponential correlation functions and Matérn correlation function (Santner et al., 2018; Stein, 1999). For instance, the Matérn correlation function has the form of

$$\Phi_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}}(2\sqrt{\nu}\|\mathbf{x}_i - \mathbf{x}_j\|_{\boldsymbol{\theta}})^{\nu} B_{\nu}(2\sqrt{\nu}\|\mathbf{x}_i - \mathbf{x}_j\|_{\boldsymbol{\theta}}), \tag{7}$$

where $\|a\|_{\boldsymbol{\theta}}^2 = \sum_{k=1}^{p}(a_k/\theta_k)^2$ with the p-dimensional lengthscale hyperparameter $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$, $\nu > 0$ is the smoothness parameter (Cramér and Leadbetter, 1967), and B_{ν} is the modified Bessel function of the second kind.

3.2 Model specification

While it is possible to model multivariate GPs, instead of independent GPs, using a separable covariance function (Bonilla et al., 2007; Qian et al., 2008) or a nonseparable covariate function (Fricker et al., 2013; Svenson and Santner, 2016), fitting these models can be computationally prohibitive when N is large. In addition, recent studies (Zhang and Cai, 2015; Kleijnen and Mehdad, 2014; Li et al., 2020) have shown that such multivariate GPs could actually yield worse prediction accuracy than the independent (univariate) GPs

described in Section 3.1. On the other hand, Gu and Berger (2016) considers independent GPs that share the same lengthscale hyperparameters over the N outputs, i.e., $\boldsymbol{\theta}_1 = \cdots = \boldsymbol{\theta}_N$, but assume different τ_j's over the N outputs.

From the perspective of statistical learning, independent univariate GPs sharing the same hyperparameters over the N outputs could be underparametrized, while the one sharing different hyperparameters could be overparametrized. To this end, we propose a flexible model that serves as a compromise between the two models:

$$\begin{aligned} \beta_j(\mathbf{x})|\tau_j^2, \boldsymbol{\theta}_j &\sim \mathcal{GP}(0, \tau_j^2 \Phi_{\boldsymbol{\theta}_j}(\mathbf{x}, \mathbf{x}')) \quad \text{for} \quad j = 1, \ldots, N, \\ \tau_j^2, \boldsymbol{\theta}_j &\sim G, \\ G &\sim \mathcal{DP}(H, \alpha_0), \end{aligned} \tag{8}$$

where $\mathcal{DP}(H, \alpha_0)$ denotes a Dirichlet process (DP) prior (Ferguson, 1973) with a positive real scalar α_0 and H being a distribution over τ_j^2 and $\boldsymbol{\theta}_j$. The parameter α_0 is often called *concentration* parameter. The smaller α_0 yields more concentrated distributions. The DP prior is a Bayesian nonparametric model and is a popular tool for developing mixture models, which are often called *infinite* mixture models, because such mixture models have a countably infinite number of mixture components; therefore, these models do not require to pre-specify a fixed number of mixture components, which can be difficult to determine in practice. A DP can be constructively defined by the *stick-breaking* construction (Sethuraman, 1994), by which (8) can be equivalently expressed as

$$\beta_j(\mathbf{x})|z_j = k, \tau_k^2, \boldsymbol{\theta}_k \sim \mathcal{GP}(0, \tau_k^2 \Phi_{\boldsymbol{\theta}_k}(\mathbf{x}, \mathbf{x}')) \qquad \text{for} \quad j = 1, \ldots, N, \tag{9}$$

$$z_j|\gamma_1, \ldots, \gamma_\infty \overset{\text{iid}}{\sim} \text{Categorical}(\pi_1, \pi_2, \ldots, \pi_\infty) \qquad \text{for} \quad j = 1, \ldots, N, \tag{10}$$

$$\pi_k = \gamma_k \prod_{l=1}^{k-1} (1 - \gamma_l) \qquad \text{for} \quad k = 1, 2, \ldots, \infty, \tag{11}$$

$$\gamma_k \overset{\text{iid}}{\sim} \text{Beta}(1, \alpha_0) \qquad \text{for} \quad k = 1, 2, \ldots, \infty, \tag{12}$$

$$\tau_k^2, \boldsymbol{\theta}_k \overset{\text{iid}}{\sim} H, \qquad \text{for} \quad k = 1, 2, \ldots, \infty,$$

where "iid" denotes independently and identitically distributed, and "Categorical" and "Beta" denote a categorical distribution and a Beta distribution, respectively. From (9),

z_j is a latent variable indicating the assignment of $\beta_j(\mathbf{x})$ to the k-th GP having the hyperparameters $\boldsymbol{\theta}_k$ and τ_k^2, where the number of GPs is countably infinite. This forms an infinite mixture of GPs for multivariate outputs, with the mixing proportion $\Pr(z_j = k) = \pi_k$. The proportion π_k is given by (11), which provides the *stick-breaking* representation of a DP as $G = \sum_{k=1}^{\infty} \pi_k \delta_{(\tau_k^2, \boldsymbol{\theta}_k)}$, where $\delta_{(\tau_k^2, \boldsymbol{\theta}_k)}$ is the indicator function whose value is one at location $(\tau_k^2, \boldsymbol{\theta}_k)$ and zero elsewhere. The proportions $\{\pi_k\}_{k=1}^{\infty}$ always sum to one and can be resembling the breaking of a unit-length stick into a countably infinite number of pieces (hence the name). That is, a portion of a unit-length stick is broken off according to γ_k and assigned to π_k, so (11) can be understood by considering that after the first $k-1$ values have been assigned their portions, the length of the remaining stick is $\prod_{l=1}^{k-1}(1-\gamma_l)$, and this remaining piece is broken according to γ_k and assigned to π_k. It is important to note that because the π_k's decrease exponentially quickly, only a small number of components will be used to model the data a priori, which allows the number of clusters to be automatically determined. More details about DP applications can be found in Neal (1992), Lo (1984), and Rasmussen (2000).

We further let the latent indicator variable be *mesh-dependent*, implying that the clustering structure is determined by the mesh coordinates. Specifically, assume that a node coordinate $\mathbf{s}$ for the cluster k follows a d-dimensional multivariate normal distribution,

$$\mathbf{s}|z = k \sim \mathcal{N}_d(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k^{-1}), \tag{13}$$

where $\boldsymbol{\mu}_k$ is the mean and $\boldsymbol{\Sigma}_k$ is the precision matrix, and their priors are a multivariate normal distribution and a Wishart distribution, respectively, that is,

$$\boldsymbol{\mu}_k \sim \mathcal{N}_d(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0^{-1}), \quad \boldsymbol{\Sigma}_k \sim \mathcal{W}(\mathbf{W}_0, \kappa_0), \tag{14}$$

where $\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, \mathbf{W}_0$ and κ_0 are fixed hyperparameters. Unlike a conditional model where $z|\mathbf{s}$ is employed to model the dependence of z and $\mathbf{s}$, the model (13) is a *generative* model (Ng and Jordan, 2001), which assumes that the mesh coordinates $\{\mathbf{s}_j\}_{j=1}^{N}$, while observed and fixed, are treated as instances or realizations generated from the underlying distribution $\mathbf{s}|z$. This conceptualization allows us to incorporate uncertainty related to distinct instances of mesh coordinates within the same cluster, and offers a consistent way of specifying each

component's responsibility for a given node coordinate (Meeds and Osindero, 2005; Sun and Xu, 2010). It is also worth noting that the multivariate normal assumption for $\mathbf{s}|z$ aligns with a well-known classification method—quadratic discriminant analysis (QDA), implying that mesh nodes will be categorized into distinct classes/clusters based on this model choice.

Remark 3.1. *Although the latent indicator variable is mesh-dependent particularly for mesh-based numerical methods, the idea can also be extended to mesh-free numerical methods, such as the collocation method (Wendland, 1998), where the basis function $v_j(\mathbf{s})$ becomes radial basis functions with some given knot locations and the knots can then be clustered in a similar manner. In addition, the idea can be naturally extended to other applications than solving PDEs that may have different dependence structures, such as the spatially-related dependence structure between proximate sites in Dahl and Bonilla (2019) and the network-related dependence structure in Wilson et al. (2012).*

Combining the above models, a graphical representation of the proposed model is given in Figure 2. It should be noted that the proposed `mcGP` is intrinsically different from the infinite mixture of GPs in Rasmussen and Ghahramani (2001), Meeds and Osindero (2005) and Sun and Xu (2010). In particular, the mixture model therein focuses on dividing the input space of $\mathbf{x}$ to address both the problems of computational complexity and stationary assumption for a univariate GP, whereas our proposed method addresses multi-output regression problems by dividing the mesh node coordinates, $\{\mathbf{s}_j\}_{j=1}^N$, into regions, within which separate univariate GPs with the same hyperparameters make predictions.

3.3 Parameter estimation: Variational inference

Although Markov chain Monte Carlo (MCMC) methods, such as Gibbs sampling, can be naturally used to draw the posterior distribution of hidden variables for DP mixture models (see, e.g., Rasmussen (2000); Neal (2000); Rasmussen and Ghahramani (2001)), they are often computationally demanding. Therefore, variational inference (VI) (Jordan et al., 1999; Ganguly and Earp, 2021) is adopted here to approximate the posterior, which leads to faster computation and efficient scalability.

Denote the parameters $\tilde{\boldsymbol{\gamma}} = (\gamma_1, \ldots, \gamma_\infty), \tilde{\boldsymbol{\mu}} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_\infty), \tilde{\boldsymbol{\Sigma}} = (\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_\infty)$ and the latent variable $\tilde{\mathbf{z}} = (z_1, \ldots, z_N)$, and denote the hyperparameters $\tilde{\boldsymbol{\tau}}^2 = (\tau_1^2, \ldots, \tau_\infty^2)$ and

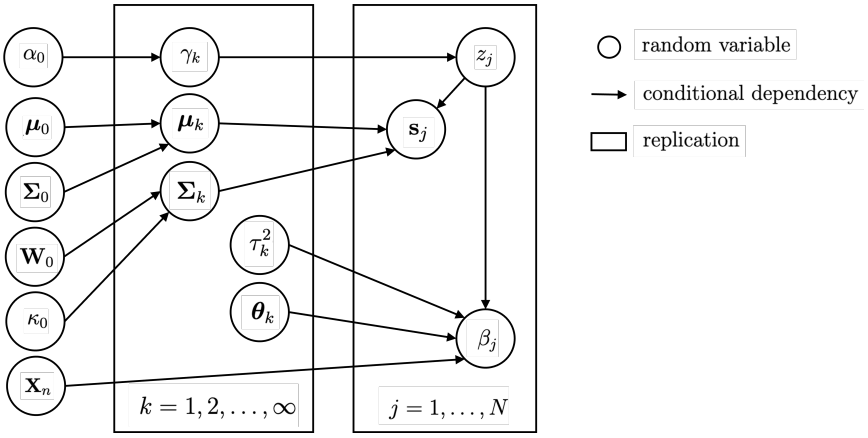

Figure 2: *Graphical model representation of* `mcGP`.

$\tilde{\boldsymbol{\theta}} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_\infty)$. Similar to Sun and Xu (2010), the hyperparameters $\tilde{\boldsymbol{\tau}}^2$ and $\tilde{\boldsymbol{\theta}}$ will be estimated via variational expectation maximization (EM). We first develop the posterior of the hidden variable vector, which is denoted by $\boldsymbol{\phi} := (\tilde{\boldsymbol{\gamma}}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}, \tilde{\mathbf{z}})$.

Denote $\mathcal{D}$ as the observational data, $\mathcal{D} = \{\beta_j(\mathbf{x}_1), \ldots, \beta_j(\mathbf{x}_n), \mathbf{s}_j\}_{j=1}^N = \{\mathbf{b}_j, \mathbf{s}_j\}_{j=1}^N$. By the graphical model representation in Figure 2, the joint distribution is

$$
\begin{aligned}
p(\mathcal{D}, \boldsymbol{\phi}) = p(\mathcal{D}|\boldsymbol{\phi})p(\boldsymbol{\phi}) &= \left(\prod_{j=1}^{N} p(\mathbf{b}_j|z_j)p(\mathbf{s}_j, z_j|\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}, \tilde{\boldsymbol{\gamma}})\right) \times \left(\prod_{k=1}^{\infty} p(\gamma_k)p(\boldsymbol{\mu}_k)p(\boldsymbol{\Sigma}_k)\right) \\
&= \left(\prod_{j=1}^{N} p(\mathbf{b}_j|z_j)p(\mathbf{s}_j|z_j, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}})p(z_j|\tilde{\boldsymbol{\gamma}})\right) \times \left(\prod_{k=1}^{\infty} p(\gamma_k)p(\boldsymbol{\mu})p(\boldsymbol{\Sigma}_k)\right), \qquad (15)
\end{aligned}
$$

where $p(\mathbf{b}_j|z_j)$ is the probability density function (pdf) of the n-dimensional multivariate normal distribution from (9), i.e., $\mathbf{b}_j|z_j = k \sim \mathcal{N}_n(0, \tau_k^2\Phi_{\boldsymbol{\theta}_k}(\mathbf{X}_n, \mathbf{X}_n))$, $p(\mathbf{s}_j|z_j, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}})$ is the pdf of the multivariate normal distribution from (13), $p(\gamma_k)$ is the beta distribution from (12), and $p(\boldsymbol{\mu}_k)$ and $p(\boldsymbol{\Sigma}_k)$ are the multivariate normal distribution and the Wishart distribution from (14), respectively.

Clearly, the posterior, $p(\boldsymbol{\phi}|\mathcal{D}) = p(\mathcal{D}, \boldsymbol{\phi})/p(\mathcal{D})$, has a complex probability density which cannot be represented in a closed form. To this end, we apply VI to provide an analytical approximation to $p(\boldsymbol{\phi}|\mathcal{D})$. Specifically, VI finds a distribution that is restricted to belong to a family of distributions of simpler forms, denoted by $q(\boldsymbol{\phi})$, such that $q(\boldsymbol{\phi}) \approx p(\boldsymbol{\phi}|\mathcal{D})$, which is called *variational distribution*. This can be done by finding the variational distribution

that minimizes the Kullback-Leibler (KL) divergence of $q(\boldsymbol{\phi})$ from $p(\boldsymbol{\phi}|\mathcal{D})$:

$$\begin{aligned}\mathrm{KL}(q(\boldsymbol{\phi})||p(\boldsymbol{\phi}|\mathcal{D})) &= \mathbb{E}_q[\log(q(\boldsymbol{\phi}))] - \mathbb{E}_q[\log p(\boldsymbol{\phi}|\mathcal{D})] \\ &= \mathbb{E}_q[\log(q(\boldsymbol{\phi}))] - \mathbb{E}_q[\log p(\mathcal{D}, \boldsymbol{\phi})] + \log p(\mathcal{D}),\end{aligned}$$

which is equivalent to maximizing

$$\mathrm{ELBO}(q) = \mathbb{E}_q[\log p(\mathcal{D}, \boldsymbol{\phi})] - \mathbb{E}_q[\log(q(\boldsymbol{\phi}))] \tag{16}$$

because $\log p(\mathcal{D}) = \mathrm{KL}(q(\boldsymbol{\phi})||p(\boldsymbol{\phi}|\mathcal{D})) + \mathrm{ELBO}(q)$ and $\log p(\mathcal{D})$ is fixed with respect to $q(\boldsymbol{\phi})$. The function is called *evidence lower bound (ELBO)*, which is a lower bound for the log-evidence of the data (hence the name), that is, $\log p(\mathcal{D}, \boldsymbol{\phi}) \geq \mathrm{ELBO}(q)$ for any q. This can be shown by the fact that $\mathrm{KL}(q(\boldsymbol{\phi})||p(\boldsymbol{\phi}|\mathcal{D})) \geq 0$ for any q.

Here we adopt the *mean field approximation* method (see Blei et al. (2017) for more details) to formulate the variational distribution $q(\boldsymbol{\phi})$, which considers the mean-field variational family where the variables $\boldsymbol{\phi}$ are mutually independent and each governed by a distinct factor in the variational density. In addition, we approximate the posterior DP by a *truncated* stick-breaking representation as in Ishwaran and James (2001), Blei and Jordan (2006), and Sun and Xu (2010). This can be done by setting $q(\gamma_K = 1) = 1$ with a fixed value K, indicating that the stick is no longer broken after $K - 1$ steps. The length of the remainder of the stick is assigned to π_K, while the lengths of the sticks beyond K become zero. This implies that the mixing proportion $\pi_k = 0$ for any $k > K$. Together, the mean field approximation method uses the following factorized variational distribution to approximate $p(\boldsymbol{\phi}|\mathcal{D})$:

$$q(\boldsymbol{\phi}) = \prod_{k=1}^{\infty} q(\gamma_k) q(\boldsymbol{\mu}_k) q(\boldsymbol{\Sigma}_k) \prod_{j=1}^{N} q(z_j) = \prod_{k=1}^{K-1} q(\gamma_k) \prod_{k=1}^{K} q(\boldsymbol{\mu}_k) q(\boldsymbol{\Sigma}_k) \prod_{j=1}^{N} q(z_j).$$

Given this mean-field variational family, the variational distribution for each $\omega \in \boldsymbol{\phi}$ can be optimized by maximizing the ELBO using coordinate ascent variational inference (Bishop, 2006), which gives the optimal variational distribution:

$$\log q(\omega) = \mathbb{E}_{\boldsymbol{\phi} \setminus \omega}[\log p(\mathcal{D}, \boldsymbol{\phi})] + \text{constant}, \tag{17}$$

where the expectation $\mathbb{E}_{\boldsymbol{\phi}\setminus\omega}$ is with respect to the variational distribution over all variables $\boldsymbol{\phi}$ except for ω. For example, from (15), the optimal variational distribution for γ_k is

$$\begin{aligned}\log q(\gamma_k) &= \mathbb{E}_{\boldsymbol{\phi}\setminus\gamma_k}[\log p(\mathcal{D}, \boldsymbol{\phi})] + \text{constant} \\ &= \sum_{j=1}^{N} \mathbb{E}_{\boldsymbol{\phi}\setminus\gamma_k}[\log p(z_j|\tilde{\boldsymbol{\gamma}})] + \log p(\gamma_k) + \text{constant}.\end{aligned}$$

The distribution $p(z_j|\tilde{\boldsymbol{\gamma}})$ can be found from (10). It follows that

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\phi}\setminus\gamma_k}[\log p(z_j|\tilde{\boldsymbol{\gamma}})] = \mathbb{E}_{\boldsymbol{\phi}\setminus\gamma_k}[\log \pi_{z_j}] &= \mathbb{E}_{\boldsymbol{\phi}\setminus\gamma_k}\left[\log \gamma_{z_j} + \sum_{l=1}^{z_j-1} \log(1-\gamma_l)\right] \\ &= \sum_{k'=1}^{\infty} q(z_j = k')\mathbb{E}_{\boldsymbol{\phi}\setminus\gamma_k}[\log \gamma_{k'}] + \sum_{k'=1}^{\infty} q(z_j = k') \sum_{l=1}^{k'-1} \mathbb{E}_{\boldsymbol{\phi}\setminus\gamma_k}[\log(1-\gamma_l)] \\ &= \sum_{k'=1}^{\infty} q(z_j = k')\mathbb{E}_{\boldsymbol{\phi}\setminus\gamma_k}[\log \gamma_{k'}] + \sum_{l=1}^{\infty} \sum_{k'=l+1}^{\infty} q(z_j = k')\mathbb{E}_{\boldsymbol{\phi}\setminus\gamma_k}[\log(1-\gamma_l)] \\ &= q(z_j = k)\log \gamma_k + q(z_j > k)\log(1-\gamma_k) + \text{constant},\end{aligned}$$

where the third equation is based on the variational distribution of z_j, i.e., $q(z_j)$. The fourth equation changes the order of summation, and the last equation holds because the terms $\mathbb{E}_{\boldsymbol{\phi}\setminus\gamma_k}[\log \gamma_{k'}]$ and $\mathbb{E}_{\boldsymbol{\phi}\setminus\gamma_k}[\log(1-\gamma_{k'})]$ are constant with respect to γ_k when $k' \neq k$. Since $p(\gamma_k)$ follows a Beta distribution as in (12), we have

$$\begin{aligned}\log q(\gamma_k) &= \sum_{j=1}^{N} q(z_j = k)\log \gamma_k + q(z_j > k)\log(1-\gamma_k) + (\alpha_0 - 1)\log(1-\gamma_k) + \text{constant} \\ &= \left(\sum_{j=1}^{N} q(z_j = k)\right)\log \gamma_k + \left(\sum_{j=1}^{N} q(z_j > k) + \alpha_0 - 1\right)\log(1-\gamma_k) + \text{constant},\end{aligned}$$

which implies that $q(\gamma_k)$ follows Beta$(\sum_{j=1}^{N} q(z_j = k) + 1, \sum_{j=1}^{N} q(z_j > k) + \alpha_0)$.

The optimal variational distributions for the remaining variables $\omega \in \boldsymbol{\phi}$ can be derived in a similar manner. The specific results are provided in the E-step of Algorithm 1, and for detailed derivations, we refer to Supplementary Materials S1. Note that the truncation level K is a variational parameter which can be freely set. Although it is possible to optimize K

with respect to the KL divergence, we hold it fixed as in Blei and Jordan (2006) and Sun and Xu (2010) throughout this paper.

Given the optimal variational distributions, the hyperparameters of the GPs, $\boldsymbol{\theta}_k$ and τ_k^2, can be estimated by maximizing the expected log-likelihood with respect to the approximated distributions, which is called *variational EM* in the literature (Blei et al., 2017). Specifically, since $\mathbf{b}_j|z_j = k \sim \mathcal{N}_n(0, \tau_k^2\Phi_{\boldsymbol{\theta}_k}(\mathbf{X}_n, \mathbf{X}_n))$, the estimates, denoted by $\hat{\boldsymbol{\theta}}_k$ and $\hat{\tau}_k^2$, can be solved by maximizing

$$\mathbb{E}_q[\log p(\mathcal{D}, \boldsymbol{\phi})] = \text{constant} - \frac{1}{2}\left(\sum_{j=1}^{N}\sum_{k=1}^{K} q(z_j = k)\left[n\log\tau_k^2 + \log|\Phi_{\boldsymbol{\theta}_k}(\mathbf{X}_n, \mathbf{X}_n)| + \frac{1}{\tau_k^2}\mathbf{b}_j^T\Phi_{\boldsymbol{\theta}_k}(\mathbf{X}_n, \mathbf{X}_n)^{-1}\mathbf{b}_j\right]\right) \quad (18)$$

with respect to $\boldsymbol{\theta}_k$ and τ_k^2. The estimators are given in the M-step of Algorithm 1, and the detailed derivations are provided in Supplementary Materials S1. The E- and M-steps repeat iteratively by updating the parameters of the variational distributions until the ELBO converges, where the explicit form of ELBO (derived from (16)) is provided in Supplementary Materials S1. In total, each iteration going through all the observations would take at most $O(KNn^3)$, which is linear with respect to the number of mesh nodes, N. We have developed an open-source R package `mcGP`, enabling the implementation of the variational EM algorithm described in Algorithm 1.

3.4 Prediction

For unknown $\mathbf{x} \in \chi$, the predictive posterior distribution of $\beta_j(\mathbf{x})$ can be constructed as:

$$\begin{aligned} p(\beta_j(\mathbf{x})|\mathcal{D}, \boldsymbol{\phi}, \{\hat{\boldsymbol{\theta}}_k, \hat{\tau}_k^2\}_{k=1}^K) &= \sum_{k=1}^{K} p(z_j = k|\mathcal{D}, \boldsymbol{\phi})p(\beta_j(\mathbf{x})|z_j = k, \mathcal{D}, \hat{\boldsymbol{\theta}}_k, \hat{\tau}_k^2) \\ &\approx \sum_{k=1}^{K} q(z_j = k)p(\beta_j(\mathbf{x})|z_j = k, \mathcal{D}, \hat{\boldsymbol{\theta}}_k, \hat{\tau}_k^2), \end{aligned}$$

where the approximation replaces $p(z_j = k|\mathcal{D}, \boldsymbol{\phi})$ with its variational distribution $q(z_j = k)$ as in Sun and Xu (2010), and $p(\beta_j(\mathbf{x})|z_j = k, \mathcal{D}, \{\hat{\boldsymbol{\theta}}_k, \hat{\tau}_k^2\}_{k=1}^K)$ is the pdf of a normal distribution with the mean (5) and the variance (6) by replacing $\boldsymbol{\theta}_j$ and τ_j^2 with $\hat{\boldsymbol{\theta}}_k$ and $\hat{\tau}_k^2$, respectively.

Thus, since $u_N(\mathbf{s};\mathbf{x}) = \sum_{j=1}^{N} \beta_j(\mathbf{x}) v_j(\mathbf{s})$, the posterior predictive mean of $u_N(\mathbf{s};\mathbf{x})$ for any $\mathbf{s} \in \Omega$ and $\mathbf{x} \in \chi$, can be approximated by

$$\hat{u}_N(\mathbf{s};\mathbf{x}) := \sum_{j=1}^{N} \sum_{k=1}^{K} q(z_j = k) v_j(\mathbf{s}) \Gamma_{\hat{\boldsymbol{\theta}}_k}(\mathbf{x}, \mathbf{X}_n) \mathbf{b}_j, \tag{19}$$

Algorithm 1 Variational expectation maximization for parameter estimation

- Set the truncation level $K > 1$ and the hyperparameters $\alpha_0, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, \mathbf{W}_0$, and κ_0.

repeat

E-step: variational distributions $q(\gamma_k), q(\boldsymbol{\mu}_k), q(\boldsymbol{\Sigma}_k)$ and $q(z_j)$.

- $\gamma_k \sim \text{Beta}(a_k, b_k)$ for $k = 1, \ldots, K-1$, where

$$a_k = \sum_{j=1}^{N} q(z_j = k) + 1, \quad b_k = \sum_{j=1}^{N} q(z_j > k) + \alpha_0.$$

- $\boldsymbol{\mu}_k \sim \mathcal{N}_d((\boldsymbol{\Sigma}_0 + \mathbf{R}_{k2})^{-1}(\boldsymbol{\Sigma}_0 \boldsymbol{\mu}_0 + \mathbf{R}_{k1}), (\boldsymbol{\Sigma}_0 + \mathbf{R}_{k2})^{-1})$ for $k = 1, \ldots, K$, where

$$\mathbf{R}_{k1} = \sum_{j=1}^{N} q(z_j = k) \mathbb{E}_q[\boldsymbol{\Sigma}_k] \mathbf{s}_j, \quad \mathbf{R}_{k2} = \sum_{j=1}^{N} q(z_j = k) \mathbb{E}_q[\boldsymbol{\Sigma}_k].$$

- $\boldsymbol{\Sigma}_k \sim \mathcal{W}(\mathbf{W}_k, \kappa_k)$, where $\kappa_k = \kappa_0 + \sum_{j=1}^{N} q(z_j = k)$ and

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + \sum_{j=1}^{N} q(z_j = k) \mathbb{E}_q[(\mathbf{s}_j - \boldsymbol{\mu}_k)(\mathbf{s}_j - \boldsymbol{\mu}_k)^T].$$

- $q(z_j = k) = r_{jk} / \sum_{k=1}^{K} r_{jk}$ for $j = 1, \ldots, N$ and $k = 1, \ldots, K$ with

$$\log r_{jk} = \mathbb{E}_q[\log \gamma_k] + \sum_{i=1}^{k-1} \mathbb{E}_q[\log(1 - \gamma_i)] + \frac{1}{2}(s_{jk} + t_{jk}), \quad \text{where}$$
$$s_{jk} = -d \log(2\pi) + \mathbb{E}_q[\log |\boldsymbol{\Sigma}_k|] - \mathbb{E}_q[(\mathbf{s}_j - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k (\mathbf{s}_j - \boldsymbol{\mu}_k)] \quad \text{and}$$
$$t_{jk} = -n \log(2\pi) - \log \tau_k^2 - \log |\Phi_{\boldsymbol{\theta}_k}(\mathbf{X}_n, \mathbf{X}_n)| - \frac{1}{\tau_k^2} \mathbf{b}_j^T \Phi_{\boldsymbol{\theta}_k}(\mathbf{X}_n, \mathbf{X}_n)^{-1} \mathbf{b}_j.$$

M-step: estimating the hyperparameters $\boldsymbol{\theta}_k$ and τ_k^2.

- $\boldsymbol{\theta}_k \leftarrow \text{argmin}_{\boldsymbol{\theta}_k} \log |\Phi_{\boldsymbol{\theta}_k}(\mathbf{X}_n, \mathbf{X}_n)| + n \log \sum_{j=1}^{N} q(z_j = k) \mathbf{b}_j^T \Phi_{\boldsymbol{\theta}_k}(\mathbf{X}_n, \mathbf{X}_n)^{-1} \mathbf{b}_j$
- $\tau_k^2 \leftarrow \left(\sum_{j=1}^{N} q(z_j = k) \mathbf{b}_j^T \Phi_{\boldsymbol{\theta}_k}(\mathbf{X}_n, \mathbf{X}_n)^{-1} \mathbf{b}_j \right) / \left(n \sum_{j=1}^{N} q(z_j = k) \right)$

until ELBO converges

- **return** $q(z_j = k), \boldsymbol{\theta}_k$ and τ_k^2 for each k and j.

where $\Gamma_{\hat{\boldsymbol{\theta}}_k}(\mathbf{x}, \mathbf{X}_n) = \Phi_{\hat{\boldsymbol{\theta}}_k}(\mathbf{x}, \mathbf{X}_n)\Phi_{\hat{\boldsymbol{\theta}}_k}(\mathbf{X}_n, \mathbf{X}_n)^{-1}$, and the posterior predictive variance is

$$\sum_{j=1}^{N} v_j(\mathbf{s})^2 \left(\sum_{k=1}^{K} q(z_j = k) \left[\hat{\tau}_k^2 (1 - \Gamma_{\hat{\boldsymbol{\theta}}_k}(\mathbf{x}, \mathbf{X}_n)\Phi_{\hat{\boldsymbol{\theta}}_k}(\mathbf{x}, \mathbf{X}_n)^T) + (\Gamma_{\hat{\boldsymbol{\theta}}_k}(\mathbf{x}, \mathbf{X}_n)\mathbf{b}_j)^2 \right] - \left(\sum_{k=1}^{K} q(z_j = k)\Gamma_{\hat{\boldsymbol{\theta}}_k}(\mathbf{x}, \mathbf{X}_n)\mathbf{b}_j \right)^2 \right). \tag{20}$$

The derivation of (20) is provided in Supplementary Materials S2. The posterior predictive mean $\hat{u}_N(\mathbf{s}; \mathbf{x})$ of (19) is used to predict $u_N(\mathbf{s}; \mathbf{x})$, and its error analysis is studied in the next section.

4 Error analysis of mcGP emulator

The error analysis is crucial for understanding the uncertainty of the emulator. By the triangle inequality, the norm of the difference between the posterior predictive mean $\hat{u}_N(\mathbf{s}; \mathbf{x})$ and true solution can be decomposed into *evaluation* and *emulation* components:

$$\|u_0 - \hat{u}_N\|_{L_2(\Omega,\chi)} \leq \underbrace{\|u_0 - u_N\|_{L_2(\Omega,\chi)}}_{\text{numerical error}} + \underbrace{\|u_N - \hat{u}_N\|_{L_2(\Omega,\chi)}}_{\text{emulation error}},$$

where the L_2-norm is defined as $\|f\|_{L_2(\Omega,\chi)} = (\int_\Omega \int_\chi f(\mathbf{s}; \mathbf{x})^2 \mathrm{d}\mathbf{s}\mathrm{d}\mathbf{x})^{1/2}$. The numerical error measures the discrepancy between the numerical solution $u_N(\mathbf{s}; \mathbf{x})$ and the ground truth $u_0(\mathbf{s}; \mathbf{x})$, and the emulation error is the error for the emulator $\hat{u}_N(\mathbf{s}; \mathbf{x})$ given limited evaluations of the simulator $u_N(\mathbf{s}; \mathbf{x})$. To save the space, we investigate the numerical error and emulation error in Supplementary Materials S3 and S4, respectively, and then apply the error bound to a common PDE problem, *elliptic equations*, in Supplementary Materials S5.

5 Numerical studies

Numerical studies are conducted in this section to examine the performance of the proposed method. Specifically, three real-world computer models comprised of PDEs are considered, which are solved via FEM using the quadratic shape functions as in (3).

In these numerical studies, the hyperparameters in the priors (12) and (14) can be quite

generic without the need of estimation. Similar to Yuan and Neubauer (2009) and Sun and Xu (2010), the hyperparameter $\boldsymbol{\mu}_0$ in (14) is set to the sample average of the node coordinates $\mathbf{S}_N$, and $\boldsymbol{\Sigma}_0$ are set to the sample inverse covariance of $\mathbf{S}_N$; the parameter κ_0 is the number of degrees of freedom under a Wishart distribution, which is set to the dimension of $\mathbf{s}_j$, that is, $\kappa_0 = d$; the scale parameter $\mathbf{W}_0$ of the Wishart distribution is set to $\boldsymbol{\Sigma}_0/d$ such that the mean of $\boldsymbol{\Sigma}_k$ is the sample inverse covariance of $\mathbf{S}_N$. The concentration parameter α_0 in (12) is set to 0.5.

For each individual GP, Matérn correlation functions (7) is considered. The smoothness parameter ν is set to $5/2$, which leads to a simplified form of (7):

$$\Phi_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j) = \left(1 + \sqrt{5}\|\mathbf{x}_i - \mathbf{x}_j\|_{\boldsymbol{\theta}} + \frac{5}{3}\|\mathbf{x}_i - \mathbf{x}_j\|^2_{\boldsymbol{\theta}}\right) \exp\left(-\sqrt{5}\|\mathbf{x}_i - \mathbf{x}_j\|_{\boldsymbol{\theta}}\right).$$

A small nugget parameter is added for numerical stability, which is set to $g \approx 1.5 \times 10^{-8}$. The truncation level K is set to 10. These numerical experiments were performed on a MacBook Pro laptop with Apple M1 Max of Chip and 32 GB of RAM.

Two performance measures are considered to examine the prediction performance. The first measure is the root mean squared error (RMSE) which is calculated as

$$\text{RMSE} = \left(\frac{\sum_{j=1}^{N}\sum_{i=1}^{n_{\text{test}}}(u_N(\mathbf{s}_j; \mathbf{x}_i^{\text{test}}) - \hat{u}_N(\mathbf{s}_j; \mathbf{x}_i^{\text{test}}))^2}{N n_{\text{test}}}\right)^{1/2}, \tag{21}$$

where n_{test} is the number of test input points, $u_N(\mathbf{s}_j; \mathbf{x}_i^{\text{test}})$ is the numerical solution of the i-th test input point, $\mathbf{x}_i^{\text{test}}$, at the j-th node location, $\mathbf{s}_i$, and $\hat{u}_N(\mathbf{s}_j; \mathbf{x}_i^{\text{test}})$ is the posterior predictive mean as in (19). The second measure is the average continuous ranked probability score (CRPS) (Gneiting and Raftery, 2007), which is a performance measure for predictive distribution of a scalar observation. Since the predictive distribution is a mixture of normal distributions as in Section 3.4, the CRPS can be computed by an analytical formula as in Grimit et al. (2006). For both RMSE and CRPS, lower values indicate better prediction accuracy.

To provide a comprehensive evaluation, we include the following methods for comparison: `uGP`, the independent univariate GPs sharing the same lengthscale hyperparameter ($\boldsymbol{\theta}_j$) but different scale hyperparameters (τ_j^2) across the N mesh nodes, which is similar to Gu

and Berger (2016); `iGP`, the independent GPs sharing different hyperparameters ($\boldsymbol{\theta}_j$ and τ_j^2) across the N mesh nodes; `pcaGP`, which uses a functional principal component analysis (FPCA) with *truncated* components (Wang et al., 2016):

$$u_N(\mathbf{s}_j; \mathbf{x}_i) \approx u_0(\mathbf{s}_j) + \sum_{l=1}^{M} \alpha_l(\mathbf{x}_i)\psi_l(\mathbf{s}_j),$$

with the leading M eigenfunctions $\{\psi_l(\mathbf{s})\}_{l=1}^M$, and the corresponding coefficients $\{\alpha_l(\mathbf{x}_i)\}$:

$$\psi_l(\mathbf{s}) = \underset{\substack{\|\phi\|_2=1,\\ \langle\phi,\psi_l\rangle=0, \forall l<j}}{\operatorname{argmax}} \sum_{i=1}^{n} \left\{ \int u_N(\mathbf{s}; \mathbf{x}_i)\phi(\mathbf{s})d\mathbf{s} \right\}^2,$$
$$\alpha_l(\mathbf{x}_i) = \int u_N(\mathbf{s}; \mathbf{x}_i)(\psi_l(\mathbf{s}) - u_0(\mathbf{s}))\, d\mathbf{s},$$

where $u_0(\mathbf{s})$ is the mean function, which can be estimated by $\sum_{i=1}^n u_N(\mathbf{s}; \mathbf{x}_i)/n$. The number of components, M, is selected by finding the leading M eigenfunctions that explain 99% of variance over all n training cases. Then, `iGP` is applied to the M coefficients $\{\alpha_l(\cdot)\}_{l=1}^M$. This approach is similar to Dancik and Dorman (2008) and Mak et al. (2018).

5.1 Poisson's Equation

In this subsection, we explore the performance for emulating a PDE boundary value problem on an L-shaped membrane based on Poisson's equation, which has broad applicability in electrostatics and fluid mechanics (Evans, 2010). The model is represented as:

$$\Delta u = (x^2 - 2\pi^2)e^{xs_1}\sin(\pi s_1)\sin(\pi s_2) + 2x\pi e^{xs_1}\cos(\pi s_1)\sin(\pi s_2), \quad \mathbf{s} = (s_1, s_2) \in \Omega,$$

where x is the one-dimensional input variable with $x \in [-1, 1]$, the operator Δ is defined by $\Delta = \frac{\partial^2}{\partial s_1^2} + \frac{\partial^2}{\partial s_2^2}$, Ω is an L-shaped membrane, and the Dirichlet boundary condition, $u = 0$ on $\partial\Omega$, is considered. The geometry and mesh, along with the solutions through FEM when $x = -1, 0, 1$, are demonstrated in Figure 3. Partial Differential Equation Toolbox of MATLAB (2021) is used to create the geometry and mesh to solve the equation.

We conduct a computer experiment of size $n = 5$, where the input locations are equally spaced in the input space $[-1, 1]$, i.e., $x_i = 0.4i - 1.2$ for $i = 1, \ldots, 5$. The mesh size is set

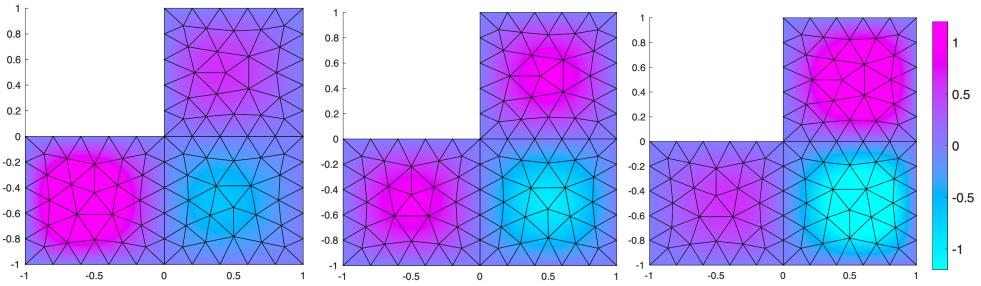

Figure 3: *Illustration of geometry, mesh, and solutions via FEM for Poisson's equation with (left)* $x = -1$*, (middle)* $x = 0$*, and (right)* $x = 1$*.*

to 0.2, yielding $N = 401$ mesh nodes as shown in Figure 3. The proposed method is applied to this experimental data. Figure 4 illustrates the variational distribution, $q(z_j = k)$, for $k = 1, \ldots, 4$, and Figure 5 presents the corresponding hyperparameter estimates, $\hat{\tau}_k^2$ and $\hat{\boldsymbol{\theta}}_k$. Note that here $k = 5, 6, \ldots, 10$ are not shown in Figures 4 and 5 because $q(z_j = k) < 0.001$ for all j, indicating that only four mixture components are only needed in this mixture model. This shows that, even though the proposed model considers an infinite mixture of GPs throughout a DP prior, the mixing proportions decrease exponentially so quickly that only a small number of components are used to model the data a priori. The fitting result reveals interesting scientific insights. First, the cluster $k = 3$ has higher probability mass on the nodes at the boundaries and locations of $s_1 = 0$ and $s_2 = 0$ (Figure 4), and the corresponding hyperparamters of the GP are $\hat{\tau}_3^2 \approx 0$ and $\hat{\theta}_3 \approx 7.5$ (Figure 5), which makes sense because these nodes are related to the solution of 0 based on the boundary condition. Second, the cluster $k = 1$ features higher probability mass on the nodes in the regions where the magnitude of solutions is the highest and their shapes are similar. The cluster shares high estimates $\hat{\tau}_1^2$ and $\hat{\theta}_1$, indicating the input-output relationship is smooth but the output values in these regions have relatively high variability across different input settings. More interestingly, the cluster $k = 2$ covers a group of the nodes $s_1 = 0$ in the neighbourhood of the cluster $k = 3$, while the cluster $k = 4$ contains a group of nodes in the vicinity of the cluster $k = 1$. The example shows that, unlike the traditional reduced-basis methods like proper orthogonal decomposition (POD) (Lumley, 1967), the clustering structures by the proposed method not only provide a useful insight for discovering the underlying physics, but also reveal their shared input-output relationship.

To illustrate the prediction performance, Figure 6 shows the FEM solution (left) at

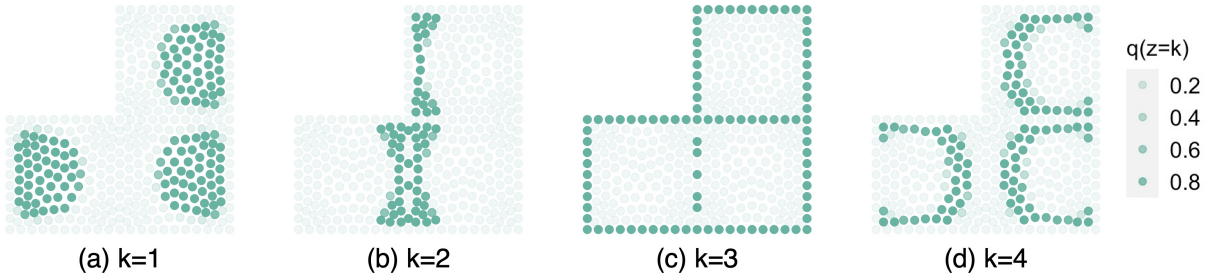

Figure 4: *Variational distribution $q(z_j = k)$ in the Poisson's equation experiment.*

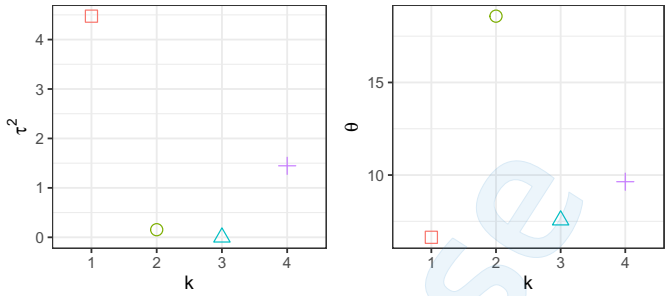

Figure 5: *Hyperparameter estimates in the Poisson's equation experiment: (left) $\hat{\tau}_k$; (right) $\hat{\theta}_k$.*

the test input point, $x^{\text{test}} = -0.25$, the `mcGP` posterior predictive mean (middle), and the `mcGP` posterior predictive standard deviation (right). From visual comparison, the `mcGP` posterior predictive mean accurately captures the spatial structure of the FEM solution. The prediction uncertainty can be quantified by the posterior predictive standard deviation, where the most uncertain predictions are located in the nodes of cluster $k = 1$, which is expected provided that the variation of the outputs are most dynamic with large magnitude in this cluster.

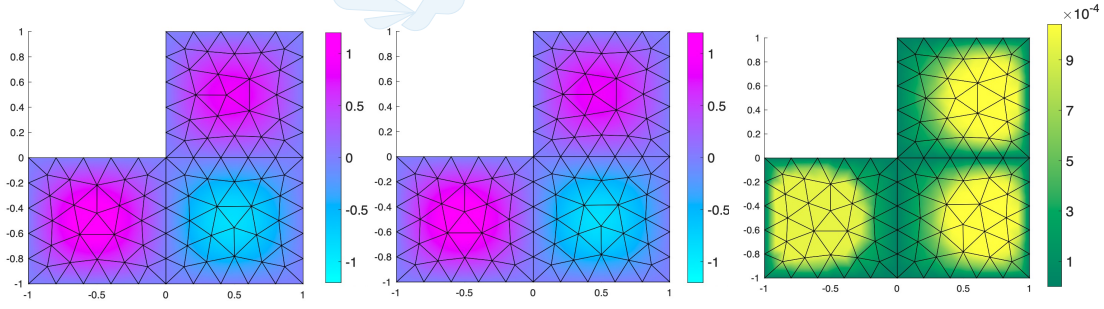

Figure 6: *Validation performance of* `mcGP` *prediction: (left) the real FEM solution when $x = -0.25$; (middle) the* `mcGP` *posterior predictive mean; (right) the* `mcGP` *posterior predictive standard deviation.*

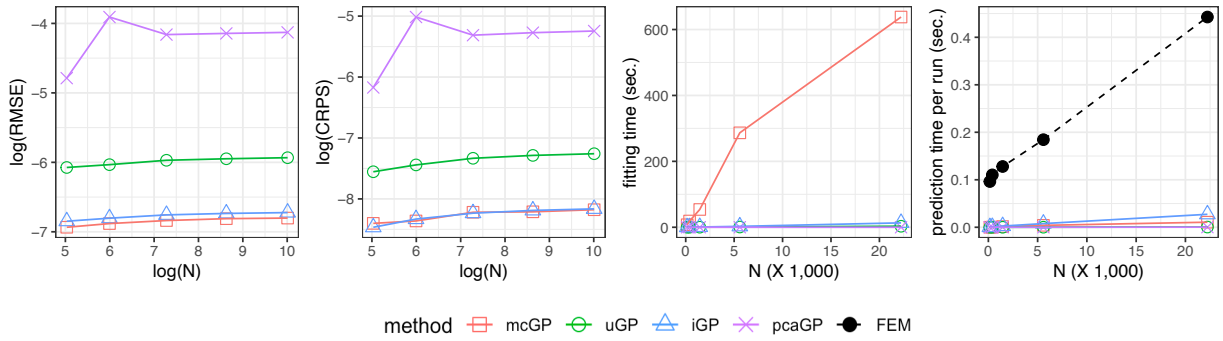

Figure 7: *Performance comparison in terms of prediction accuracy and computational cost in the Poisson's equation experiment. The left two panels are the prediction performance in terms of RMSE and CRPS in logarithm, and the right two panels are the computational cost in model fitting and predictions.*

We further examine the prediction and computation performance on 201 test input points equally spaced in $[-1, 1]$ with various mesh size settings, which are 0.4, 0.2, 0.1, 0.05, and 0.025. The results are presented in Figure 7. From the left two panels, it appears that `uGP` and `pcaGP` perform worse than the other two in terms of prediction accuracy (both RMSE and CRPS). While there is no significant difference between `iGP` and `mcGP` in the predictive scores, it can be seen that `mcGP` generally yields lower RMSEs than other methods. The right two panels present the computational cost. It is of no surprise that the proposed method requires more fitting time due to the complications of a mixture model; however, as indicated in Section 3.3, the third panel (from the left) shows that the computational cost for fitting `mcGP` is linear with respect to N, which is reasonably tractable in practice. Besides, in the context of emulation, the computation for predictions is more of interest, for which it can be seen that the proposed method (and its competitors) can predict much faster than conducting a real FEM simulation. It is important to note that while in Figure 7, `iGP` demonstrates comparable prediction accuracy, in the problems presented in Sections 5.2 and 5.3, it performs less accurately compared to `uGP` and `mcGP`. On the other hand, `mcGP` generally exhibits better prediction accuracy than `iGP` and `uGP`.

5.2 Laminar flow past a cylinder

In this subsection, we investigate a system of a two-dimensional flow past a circular cylinder, which is a classical and interesting problem in fluid mechanics (Rajani et al., 2009; Seo

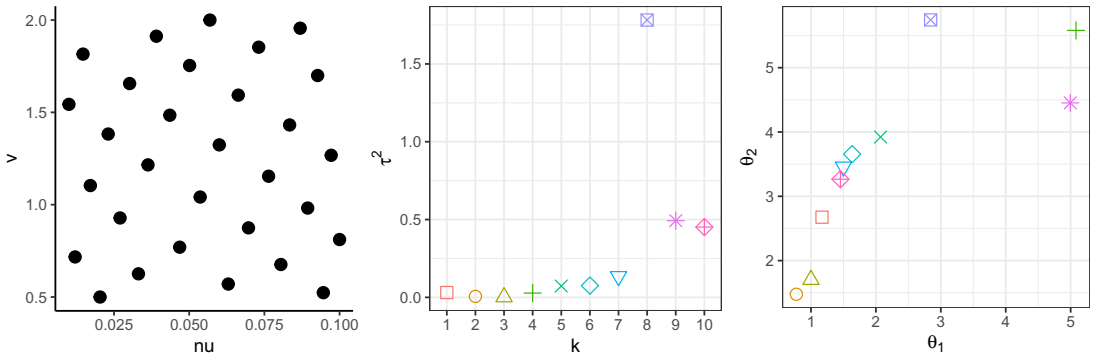

Figure 8: *The left panel presents the design points in the laminar flow application, and the middle and right panels show the hyperparameter estimates of* `mcGP`*:* $\hat{\tau}_k$ *(middle) and* $\hat{\boldsymbol{\theta}}_k$ *(right).*

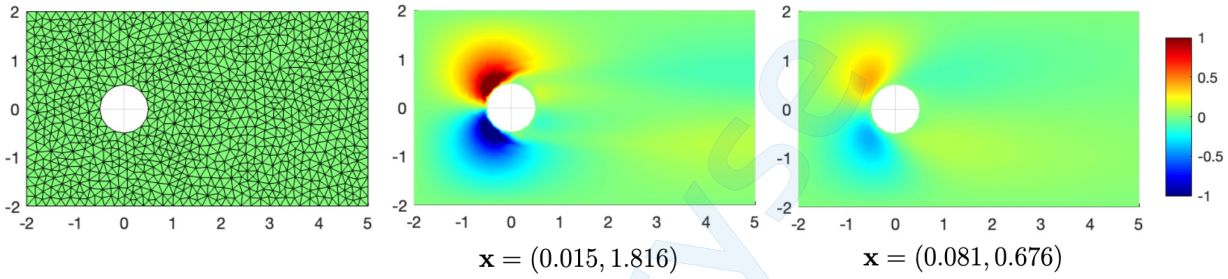

Figure 9: *The left panel demonstrates the geometry and mesh in the laminar flow application, and the middle and right panels are the two (out of 30) training examples of the simulations, where the input settings are indicated by the values of* $\mathbf{x}$*.*

and Song, 2012). The problem is described by the incompressible Navier-Stokes equations. Two input variables are considered in this study, which are the kinematic viscosity of the fluid (x_1) and the freestream velocity in the s_1 direction (x_2), where the input space is $\mathbf{x} = (x_1, x_2) \in [0.01, 0.1] \times [0.5, 2]$, which results in laminar flows with Reynolds number between 0.5 and 200. The objective is to predict the velocity component in the s_2 direction (v) using the `mcGP` method. The sample size is prescribed as $n = 30$. The sample points are distributed uniformly in the input space following the maximum projection (MaxPro) design method (Joseph et al., 2015), as shown in the left panel of Figure 8. The computer simulations at these points are performed using the FEM solver in the QuickerSim CFD Toolbox (Ltd., 2022). Figure 9 shows the computational mesh and computer solutions of v at two different input settings. The total number of mesh nodes is $N = 3778$.

The proposed `mcGP` method is applied to this experimental data. Figure 10 illustrates the variational distribution, $q(z_j = k)$, for $j = 1, \ldots, N$, and $k = 1, \ldots, 5, 8, 9, 10$, and the

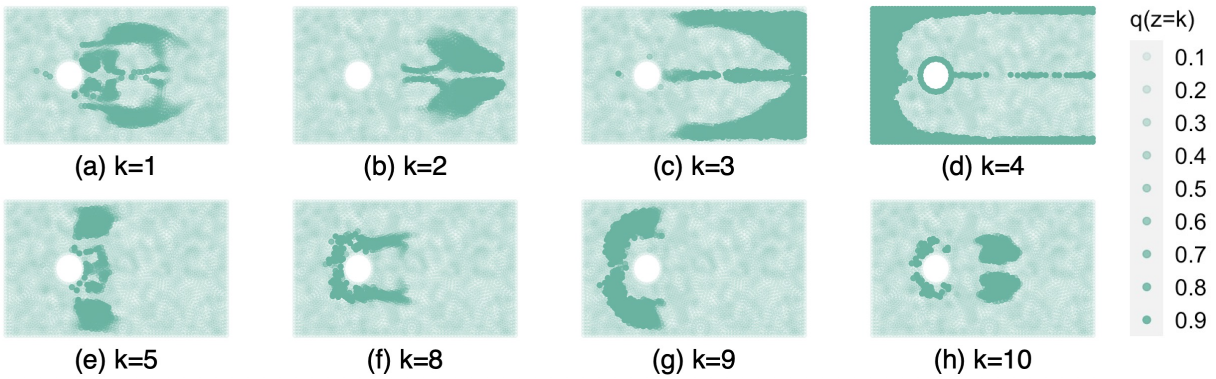

Figure 10: *Variational distribution $q(z_j = k)$ in the laminar flow application.*

corresponding hyperparameter estimates are presented in the middle and right panels of Figure 8 for each k. Note that the clusters $k = 6$ and $k = 7$ are not shown here because $q(z_j = k) < 0.1$ for all j at these clusters. The result shows that different clusters exhibit high probability mass on the nodes in different fluid regions. For example, the cluster $k = 4$ has higher probability mass on the nodes at the upstream and vertical boundaries with small $\hat{\tau}_4^2$ and large $\hat{\boldsymbol{\theta}}_4$, while the cluster $k = 3$ imposes higher probability mass in the downstream boundary and associated neighbouring regions. These two clusters share a common feature of low variation in the vertical velocity, as manifested by Figure 9. The components $k = 8$ and $k = 9$ puts higher probability mass in the frontal area of the cylinder with the highest output magnitude, and this comes with the large hyperparameter estimates $\hat{\tau}_8^2$ and $\hat{\tau}_9^2$. In addition, the clusters $k = 1, 2$, and 10 have higher probability mass in the wake region. It turns out that the regions of high probability mass exerted by different clusters constitute the entire computational domain, implying the present clustering strategy is efficient.

The prediction and computation performance is examined on 100 uniformly random test input points from the input space. Three (out of 100) test FEM simulations along with the `mcGP` predictions are presented in Figure 11. From visual inspection, it appears that the point-wise predictions of `mcGP` are fairly accurate at the three input points. All dynamic structures in the flow are well captured, including the large-magnitude region in the front of the cylinder and the wake region behind the cylinder. Table 1 shows the results of the 100 test data in comparison of other GP models, indicating that the proposed `mcGP` outperforms the others in terms of prediction accuracy with reasonable computation time. Similar to the previous subsection, `pcaGP` performs the worst, which is not surprising because the

approximation error of the dimension reduction approach can introduce additional bias to the predictions (Sung et al., 2024b). Unlike the previous subsection, `uGP` outperforms `iGP` in terms of both RMSE and CRPS. This again demonstrates that `mcGP` can serve as a middle ground between these two models.

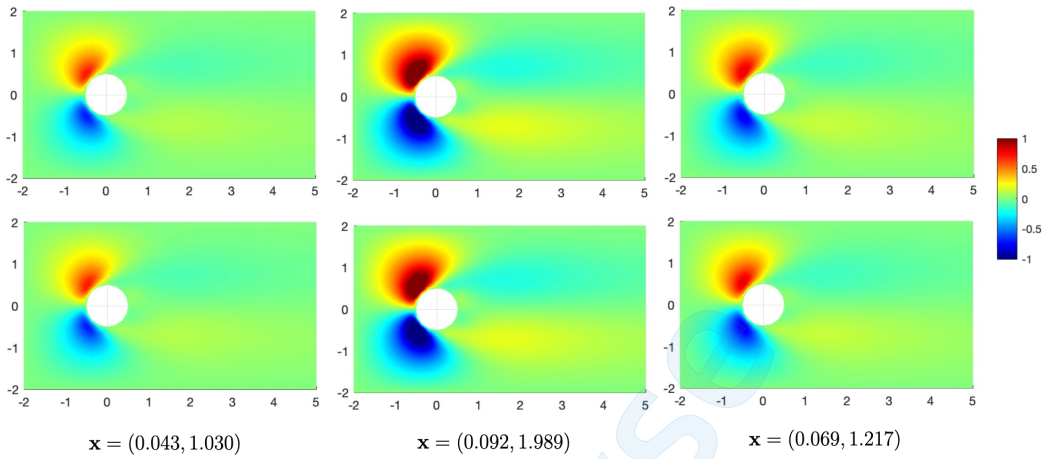

Figure 11: *Validation performance of* `mcGP` *prediction in the laminar flow application. The upper panels present three (out of 100) test FEM simulations (with the input settings indicated by the values of* $\mathbf{x}$*) and the bottom panels are the corresponding posterior predictive means of* `mcGP`*.*

method	`mcGP`	`uGP`	`iGP`	`pcaGP`
RMSE ($\times 10^{-4}$)	**8.741**	9.064	15.385	24.880
CRPS ($\times 10^{-4}$)	**2.276**	2.410	5.537	15.178
fitting time (sec.)	98	10	10	**0.13**
prediction time per run (msec.)	3	0.1	19	**0.06**

Table 1: *Performance comparison in terms of prediction accuracy and computational cost in the laminar flow application, in which the better performances are boldfaced. Note that the test FEM simulations take, on average, 1068 milliseconds per run.*

5.3 Thermal stress analysis of jet engine turbine blade

In this subsection, we investigate the performance of the proposed method on a thermal stress analysis application for a jet turbine engine blade in steady-state operating condition. More details can be found in Wright and Han (2006), Carter (2005) and Sung et al. (2024a). In this study, we aim to emulate the thermal stress and deformation of a turbine with the

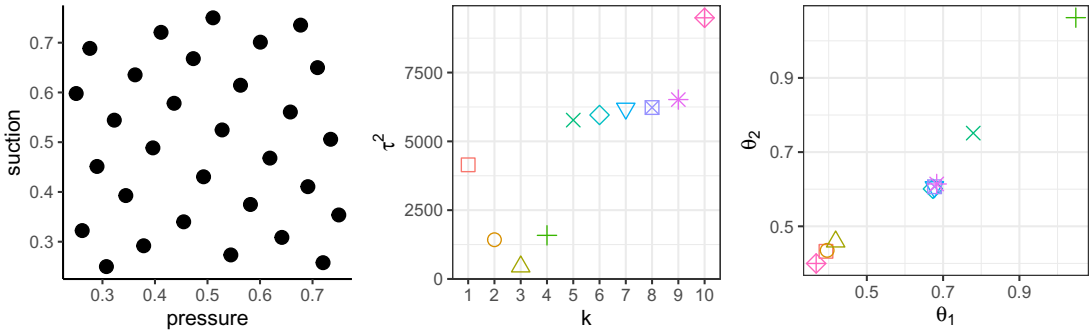

Figure 12: *The left panel presents the design points in the turbine blade application, and the middle and right panels show the hyperparameter estimates of* `mcGP`*:* $\hat{\tau}_k$ *(middle) and* $\hat{\boldsymbol{\theta}}_k$ *(right).*

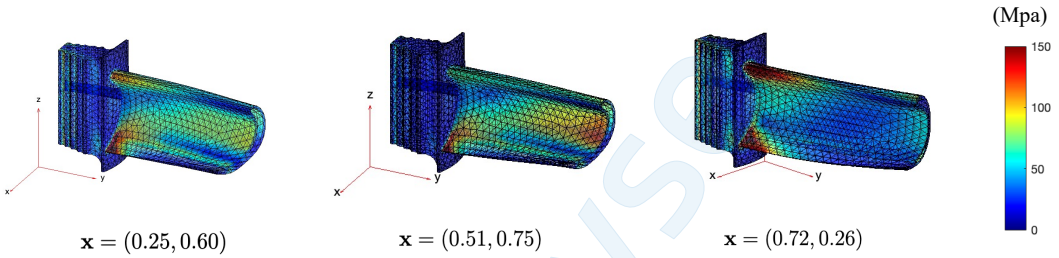

Figure 13: *Three (out of 30) examples of the training FEM simulations in the turbine blade application, where the input settings are indicated by the values of* $\mathbf{x}$*.*

effect of the thermal stress and pressure of the surrounding gases on turbine blades. The problem is analyzed as a static structural model, which can be numerically solved using FEM. Two input variables are considered, which are the pressure load on the pressure (x_1) and suction (x_2) sides of the blade, both of which range from 0.25 to 0.75 MPa, i.e., $(x_1, x_2) \in [0.25, 0.75]^2$. FEM simulations of sample size $n = 30$ for the thermal stress are conducted via the Partial Differential Equation Toolbox (MATLAB, 2021). The sample points over the input space are allocated using the MaxPro design, which are presented in the left panel of Figure 12. The mesh size is set to 0.01, yielding $N = 21158$ mesh nodes. Three (out of 30) training examples are presented in Figure 13. They show that larger suction pressure ($x_2 = 0.60$ and 0.69) tends to impose higher thermal stress on the central area of the blade airfoil and the blade leading and trailing edges near the platform.

Figure 14 illustrates the variational distribution of the selected components, $q(z_j = k)$, and the corresponding hyperparameter estimates, $\hat{\tau}_k^2$ and $\hat{\boldsymbol{\theta}}_k$, are shown in the middle and right panels of Figure 12. The clusters $k = 6, 7, 8$ and 9 are not shown here because

$q(z_j = k) < 0.3$ for all j at these clusters. The distribution of the nodes in resulting clusters correspond to the distribution of thermal stress in a physically meaning manner. The cluster $k = 4$ has higher probability mass on the top center of the blade airfoil, where the thermal load is relatively large with high variability ($\hat{\tau}_4^2 \approx 1584$) across different input settings; on the other hand, the cluster $k = 3$ put higher probability mass on the platform area, where the thermal load is relatively small with subtle variability (small $\hat{\tau}_3^2$). In addition, the cluster $k = 1$ has higher probability mass on the leading and trailing edges of the blade next to the platform, which also tends to have a high variability ($\tau^2 \approx 4147$) of the thermal stress.

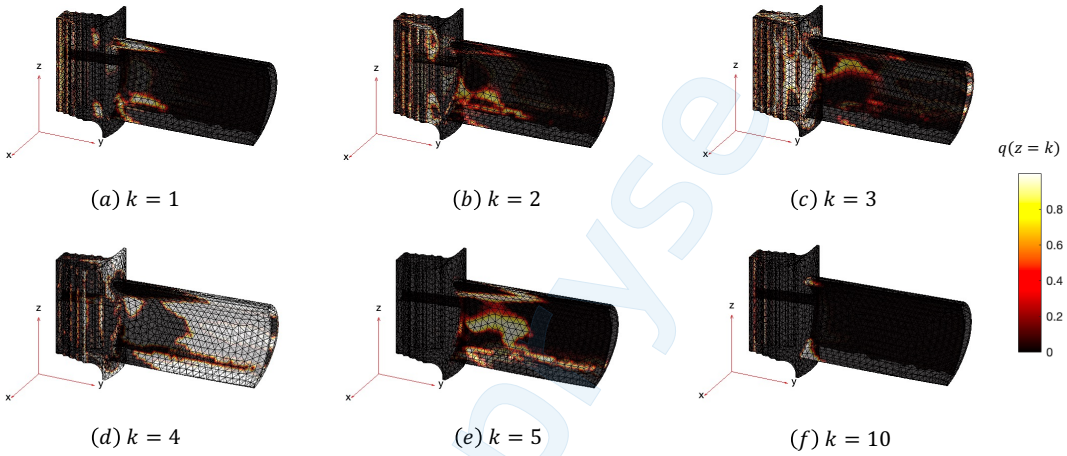

Figure 14: *Variational distribution $q(z_j = k)$ in the turbine blade application.*

Similar to Section 5.2, 100 test FEM simulations are conducted at uniformly random test input locations to examine the prediction and computation performance. Three (out of 100) test examples are demonstrated in Figure 15, which shows that, from visual inspection, the predicted thermal stress (upper panels) closely mimics the simulated thermal stress (lower panels). In comparison with the competitors, the results are presented in Table 2, which again shows that the proposed method can outperform others in terms of prediction accuracy with reasonable computation time.

6 Concluding remarks

PDE simulations based on mesh-based numerical methods have become essential in various applications, ranging from engineering to health care. In this paper, we propose a new

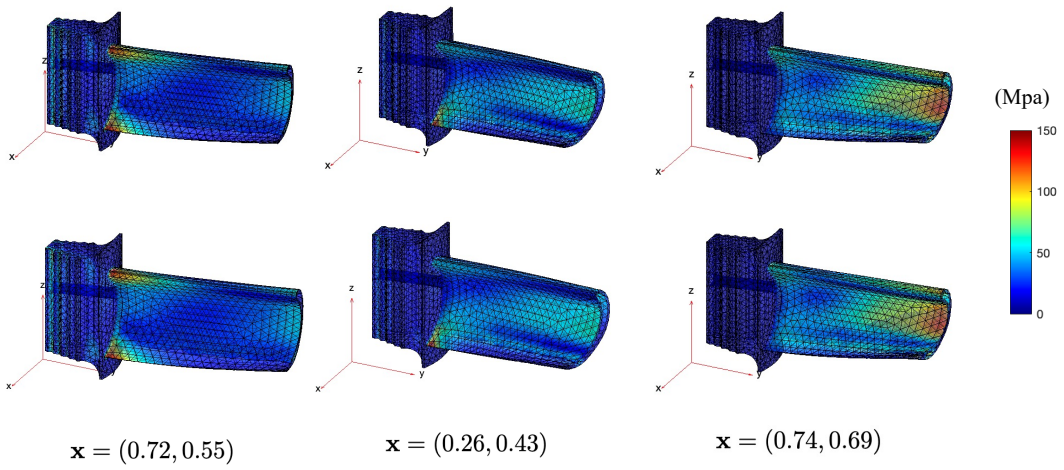

Figure 15: *Validation performance of* `mcGP` *prediction in the turbine blade application. The upper panels present three (out of 100) test FEM simulations (with the input settings indicated by the values of* $\mathbf{x}$*) and the bottom panels are the corresponding posterior predictive means of* `mcGP`*.*

method	`mcGP`	`uGP`	`iGP`	`pcaGP`
RMSE ($\times 10^{-1}$)	**9.800**	9.823	11.073	13.986
CRPS ($\times 10^{-1}$)	**3.204**	3.236	3.617	6.753
fitting time (sec.)	313	134	71	**0.19**
prediction time per run (msec.)	19	0.7	97	**0.3**

Table 2: *Performance comparison in terms of prediction accuracy and computational cost in the turbine blade application, in which the better performances are boldfaced. Note that the test FEM simulations take, on average, 3819 milliseconds per run.*

surrogate model for expensive PDE boundary value problems that simultaneously emulates the PDE solutions over a spatial domain. An important innovation of this work lies in its incorporation of mesh node locations into a statistical model, forming a mixture model that compromises the bias-variance trade-off in the context of many outputs. Furthermore, we develop a rigorous theoretical error analysis for the proposed emulator, which provides an important insight about its uncertainty quantification. Three real examples show that the method not only has advantages in prediction accuracy, but also enables discovery of interesting physics by interpreting the mixture clusters.

The proposed method shows several avenues for future research. First, in addition to the fine mesh in a spatial domain, the proposed method can be modified for simulations having fine grids over both the spatial and temporal domains. This can be naturally done

by incorporating the time information into the latent model (13) to segment the temporal domain, and it is conceivable that the resulting clustering structures can reveal interesting *dynamic* features. Second, although the method developed herein assumes that the mesh specifications are identical across different input settings, the proposed method can be modified to tackle different mesh specifications. This can be done by utilizing the idea of *common grid* by Mak et al. (2018). Specifically, we recommend first determining a common grid, denoted as $\bar{\mathbf{S}} = (\bar{\mathbf{s}}_1, \ldots, \bar{\mathbf{s}}_N)^T$, which serves as a reference for predictions. By evaluating training outputs $u_N(\bar{\mathbf{s}}_j; \mathbf{x}_i)$ for each $i = 1, \ldots, n$ and $j = 1, \ldots, N$ via (2), we can subsequently model $\{u_N(\bar{\mathbf{s}}_j; \mathbf{x})\}_{j=1,\ldots,N}$ as an `mcGP`, as introduced in Section 3. This strategy enables predictions at untried points $\mathbf{x} \in \chi$ on the common grid $\bar{\mathbf{S}}$. Lastly, it is worthwhile investigating the incorporation of the boundary conditions into the proposed model as in Tan (2018a,b), to further improve the prediction accuracy. We leave these to our future work.

Supplemental Materials Additional supporting materials can be found in Supplemental Materials, including the detailed derivations for Algorithm 1, the derivation of (20), the detailed error analysis in Section 4, and the R code for reproducing the results in Section 5.

Acknowledgements The authors express sincere gratitude for the conscientious efforts of the associate editor and two anonymous reviewers whose insightful comments greatly strengthen this paper.

References

Akyildiz, Ö. D., Duffin, C., Sabanis, S., and Girolami, M. (2022). Statistical finite elements via Langevin dynamics. *SIAM/ASA Journal on Uncertainty Quantification*, 10(4):1560–1585.

Bathe, K.-J. (2006). *Finite Element Procedures*. Klaus-Jurgen Bathe.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer New York.

Blei, D. M. and Jordan, M. I. (2006). Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Bonilla, E. V., Chai, K., and Williams, C. (2007). Multi-task gaussian process prediction. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

Brenner, S. C. and Scott, L. R. (2007). *The Mathematical Theory of Finite Element Methods (Third Edition),*. New York: Springer.

Carter, T. J. (2005). Common failures in gas turbine blades. *Engineering Failure Analysis*, 12(2):237–247.

Conti, S. and O'Hagan, A. (2010). Bayesian emulation of complex multi-output and dynamic computer models. *Journal of Statistical Planning and Inference*, 140(3):640–651.

Cramér, H. and Leadbetter, M. R. (1967). *Stationary and Related Stochastic Processes: Sample Function Properties and Their Applications*. Courier Corporation.

Dahl, A. and Bonilla, E. V. (2019). Grouped Gaussian processes for solar power prediction. *Machine Learning*, 108(8):1287–1306.

Dancik, G. M. and Dorman, K. S. (2008). mlegp: statistical analysis for computer models of biological systems using r. *Bioinformatics*, 24(17):1966–1967.

Duffin, C., Cripps, E., Stemler, T., and Girolami, M. (2021). Statistical finite elements for misspecified models. *Proceedings of the National Academy of Sciences*, 118(2):e2015006118.

Evans, L. C. (2010). *Partial Differential Equations (Second Edition)*, volume 19. American Mathematical Society.

Fasshauer, G. E. (1996). Solving partial differential equations by collocation with radial basis functions. In *Proceedings of Chamonix*, volume 1997, pages 1–8. Vanderbilt University Press Nashville, TN.

Fasshauer, G. E. (1999). Solving differential equations with radial basis functions: multilevel methods and smoothing. *Advances in Computational Mathematics*, 11(2-3):139–159.

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230.

Fornberg, B. and Flyer, N. (2015). Solving pdes with radial basis functions. *Acta Numerica*, 24:215–258.

Fricker, T. E., Oakley, J. E., and Urban, N. M. (2013). Multivariate Gaussian process emulators with nonseparable covariance structures. *Technometrics*, 55(1):47–56.

Ganguly, A. and Earp, S. W. (2021). An introduction to variational inference. *arXiv preprint arXiv:2108.13083*.

Girolami, M., Febrianto, E., Yin, G., and Cirak, F. (2021). The statistical finite element method (statfem) for coherent synthesis of observation data and model predictions. *Computer Methods in Applied Mechanics and Engineering*, 375:113533.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378.

Golberg, M., Chen, C., and Bowman, H. (1999). Some recent results and proposals for the use of radial basis functions in the bem. *Engineering Analysis with Boundary Elements*, 23(4):285–296.

Gramacy, R. B. (2020). *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*. CRC Press.

Grimit, E. P., Gneiting, T., Berrocal, V. J., and Johnson, N. A. (2006). The continuous ranked probability score for circular variables and its application to mesoscale forecast ensemble verification. *Quarterly Journal of the Royal Meteorological Society*, 132(621C):2925–2942.

Gu, M. and Berger, J. O. (2016). Parallel partial gaussian process emulation for computer models with massive output. *The Annals of Applied Statistics*, 10(3):1317–1347.

Higdon, D., Gattiker, J., Williams, B., and Rightley, M. (2008). Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583.

Ishwaran, H. and James, L. F. (2001). Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Joseph, V. R., Gul, E., and Ba, S. (2015). Maximum projection designs for computer experiments. *Biometrika*, 102(2):371–380.

Joseph, V. R. and Mak, S. (2021). Supervised compression of big data. *Statistical Analysis and Data Mining*, 14(3):217–229.

Karhunen, K. (1947). *Ueber lineare Methoden in der Wahrscheinlichkeitsrechnung*, volume 37. Universitat Helsinki.

Kleijnen, J. P. C. and Mehdad, E. (2014). Multivariate versus univariate kriging metamodels for multi-response simulation models. *European Journal of Operational Research*, 236(2):573–582.

Lee, L., Carslaw, K., Pringle, K., and Mann, G. (2012). Mapping the uncertainty in global ccn using emulation. *Atmospheric Chemistry and Physics*, 12(20):9739–9751.

Lee, L., Carslaw, K., Pringle, K., Mann, G., and Spracklen, D. (2011). Emulation of a complex global aerosol model to quantify sensitivity to uncertain parameters. *Atmospheric Chemistry and Physics*, 11(23):12253–12273.

Li, M., Liu, M.-Q., Wang, X.-L., and Zhou, Y.-D. (2020). Prediction for computer experiments with both quantitative and qualitative factors. *Statistics & Probability Letters*, page 108858.

Lo, A. Y. (1984). On a class of Bayesian nonparametric estimates: I. density estimates. *The Annals of Statistics*, 12(1):351–357.

Loève, M. (1955). *Probability Theory: Foundations, Random Sequences*. New York: D. Van Nostrand Company.

Ltd., Q. (2022). QuickerSim CFD Toolbox. https://old.quickersim.com/cfdtoolbox/. [Online; accessed 10-November-2022].

Lumley, J. L. (1967). The structure of inhomogeneous turbulent flows. *Atmospheric Turbulence and Radio Wave Propagation*, pages 166–178.

Mak, S., Sung, C.-L., Wang, X., Yeh, S.-T., Chang, Y.-H., Joseph, V. R., Yang, V., and Wu, C. F. J. (2018). An efficient surrogate model for emulation and physics extraction of large eddy simulations. *Journal of the American Statistical Association*, 113(524):1443–1456.

Marrel, A., Iooss, B., Jullien, M., Laurent, B., and Volkova, E. (2011). Global sensitivity analysis for models with spatially dependent outputs. *Environmetrics*, 22(3):383–397.

MATLAB (2021). *MATLAB version 9.11.0.1769968 (R2021b)*. The Mathworks, Inc., Natick, Massachusetts.

Meeds, E. and Osindero, S. (2005). An alternative infinite mixture of Gaussian process experts. *Advances in Neural Information Processing systems*, 18.

Neal, R. M. (1992). Bayesian mixture modeling. In *Maximum Entropy and Bayesian Methods*, volume 11, pages 197–211. Springer.

Neal, R. M. (2000). Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265.

Ng, A. and Jordan, M. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems*, 14.

Qian, P. Z. G., Wu, H., and Wu, C. F. J. (2008). Gaussian process models for computer experiments with qualitative and quantitative factors. *Technometrics*, 50(3):383–396.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rajani, B. N., Kandasamy, A., and Majumdar, S. (2009). Numerical simulation of laminar flow past a circular cylinder. *Applied Mathematical Modelling*, 33(3):1228–1247.

Ramsay, J. O. and Silverman, B. W. (2005). *Functional Data Analysis (Second Edition)*. Springer New York, NY.

Rasmussen, C. (2000). The infinite Gaussian mixture model. *Advances in Neural Information Processing systems*, 12.

Rasmussen, C. and Ghahramani, Z. (2001). Infinite mixtures of Gaussian process experts. *Advances in Neural Information Processing Systems*, 14.

Rougier, J. (2008). Efficient emulators for multivariate deterministic functions. *Journal of Computational and Graphical Statistics*, 17(4):827–843.

Rougier, J., Guillas, S., Maute, A., and Richmond, A. D. (2009). Expert knowledge and multivariate emulation: The thermosphere–ionosphere electrodynamics general circulation model (tie-gcm). *Technometrics*, 51(4):414–424.

Santner, T. J., Williams, B. J., and Notz, W. I. (2018). *The Design and Analysis of Computer Experiments*. Springer Science & Business Media.

Seo, I. W. and Song, C. G. (2012). Numerical simulation of laminar flow past a circular cylinder with slip conditions. *International Journal for Numerical Methods in Fluids*, 68(12):1538–1560.

Sethuraman, J. (1994). A constructive definition of Dirichlet priors. *Statistica Sinica*, 4(2):639–650.

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media.

Sun, S. and Xu, X. (2010). Variational inference for infinite mixtures of Gaussian processes with applications to traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):466–475.

Sung, C.-L., Haaland, B., Hwang, Y., and Lu, S. (2023). A clustered Gaussian process model for computer experiments. *Statistica Sinica*, 33(2):893–918.

Sung, C.-L., Hung, Y., Rittase, W., Zhu, C., and Wu, C. F. J. (2020). Calibration for computer experiments with binary responses and application to cell adhesion study. *Journal of the American Statistical Association*, 115(532):1664–1674.

Sung, C.-L., Ji, Y., Tang, T., and Mak, S. (2024a). Stacking designs: designing multi-fidelity computer experiments with confidence. *SIAM/ASA Journal on Uncertainty Quantification*. To appear.

Sung, C.-L., Wang, W., Cakoni, F., Harris, I., and Hung, Y. (2024b). Functional-input gaussian processes with applications to inverse scattering problems. *Statistica Sinica*, 34(4). To appear.

Svenson, J. and Santner, T. (2016). Multiobjective optimization of expensive-to-evaluate deterministic computer simulator models. *Computational Statistics & Data Analysis*, 94(1):250–264.

Tan, M. H. (2018a). Gaussian process modeling of a functional output with information from boundary and initial conditions and analytical approximations. *Technometrics*, 60(2):209–221.

Tan, M. H. Y. (2018b). Gaussian process modeling with boundary information. *Statistica Sinica*, 28(2):621–648.

Tekkaya, A. E. and Soyarslan, C. (2019). Finite element method. In Chatti, S., Laperrière, L., Reinhart, G., and Tolio, T., editors, *CIRP Encyclopedia of Production Engineering*, pages 677–683. Springer Berlin Heidelberg.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

Wang, J.-L., Chiou, J.-M., and Müller, H.-G. (2016). Functional data analysis. *Annual Review of Statistics and Its Application*, 3:257–295.

Wang, Y., Wang, X., and Yang, V. (2018). Evolution and transition mechanisms of internal swirling flows with tangential entry. *Physics of Fluids*, 30(1):013601.

Wendland, H. (1998). Numerical solution of variational problems by radial basis functions. *Approximation Theory IX*, 2:361–368.

Wendland, H. (1999). Meshless galerkin methods using radial basis functions. *Mathematics of Computation of the American Mathematical Society*, 68(228):1521–1531.

Wilson, A. G., Knowles, D. A., and Ghahramani, Z. (2012). Gaussian process regression networks. In *International Conference on Machine Learning*.

Wright, L. M. and Han, J.-C. (2006). Enhanced internal cooling of turbine blades and vanes. *The Gas Turbine Handbook*, 4:1–5.

Yuan, C. and Neubauer, C. (2009). Variational mixture of Gaussian process experts. *Advances in Neural Information Processing Systems*, 21:1897–1904.

Zhang, H. and Cai, W. (2015). When doesn't cokriging outperform kriging? *Statistical Science*, 30(2):176–180.