Clustered active-subspace based local Gaussian process emulator for high-dimensional and complex computer models

Xiong, Junda; Cai, Xin; Li, Jinglai

[Link to publication on Research at Birmingham portal](#)

Clustered active-subspace based local Gaussian Process emulator for high-dimensional and complex computer models

Junda Xiong [a]

[a]*School of Mathematical Sciences, Shanghai Jiao Tong University, 800 Dongchuan Rd, Shanghai 200240, P.R. China.*

Xin Cai [b]

[b]*School of Mathematical Sciences, Shanghai Jiao Tong University, 800 Dongchuan Rd, Shanghai 200240, P.R. China.*

Jinglai Li [c]

[c]*School of Mathematics, University of Birmingham, Birmingham B15 2TT, UK. (Corresponding author)*

Abstract

Quantifying uncertainties in physical or engineering systems often requires a large number of simulations of the underlying computer models that are computationally intensive. Emulators or surrogate models are often used to accelerate the computation in such problems, and in this regard the Gaussian Process (GP) emulator is a popular choice for its ability to quantify the approximation error in the emulator itself. However, a major limitation of the GP emulator is that it can not handle problems of very high dimensions, which is often addressed with dimension reduction techniques. In this work we hope to address an issue that the models of interest are so complex that they admit different low dimensional structures in different parameter regimes. Building upon the active subspace method for dimension reduction, we propose a clustered active subspace method which identifies the local low-dimensional structures as well as the parameter regimes they are in (represented as clusters), and then construct low dimensional and local GP emulators within the clusters. Specifically we design a clustering method based on the gradient information to identify these clusters, and a local GP construction procedure to construct the GP emulator within a local cluster. With numerical examples, we demonstrate

that the proposed method is effective when the underlying models are of complex low-dimensional structures.

1 Introduction

Computer models or simulators are one of the most important tools to study complex physical or biological processes and explore their behaviors, in many fields of science and engineering. As is well known, the computer models are inevitably subject to various sources of uncertainty: imprecise boundary or initial conditions, unknown model parameters, random perturbations, and so on. To characterize the impact of the uncertainties in the simulation results, a large number, e.g., tens of thousands or more, of simulations are required. On the other hand, in reality, the computer models are often computationally intensive, especially when the system of interest involves highly complex physical processes, and in this case, conducting a large number of simulations to quantify the impact of uncertainties becomes a prohibitive task. A commonly used approach to overcome this difficulty is to construct computationally inexpensive surrogate models, namely emulators, to approximate the computer simulators in a probabilistic way [26]. Substantial efforts have been devoted to this topic and many different types of surrogate models have been developed: the polynomial chaos expansion [34,33], radial basis functions (RBF) [13,24], adaptive sparse grid collocation [18], the Gaussian processes (GP) regression [31], and more recently, neural networks [36,22,37].

In this work, we choose to use the GP regression because of its probabilistic formulation, and we can take the advantage of the GP method's ability to epistemically quantify the uncertainty induced by all the random effects and limited number of simulations in a natural Bayesian framework [31]. In particular, it makes it possible to statistically quantify the approximation error in the emulator result itself, which then can be incorporated in the analysis of the total uncertainty, see [21,19,20]. For this reason, the GP emulators have been widely used in the field of uncertainty quantification (UQ): uncertainty propagation [6,12,35], parameter estimation [14,29], reliability analysis [3,30,32], just to name a few.

Email addresses: `jdxiong@sjtu.edu.cn` (Junda Xiong), `izumi_xin@sjtu.edu.cn` (Xin Cai), `j.li.10@bham.ac.uk` (Jinglai Li).

Despite the wide success of the GP emulator, a major limitation of the method, as well as practically any aforementioned surrogate models, is that it can not handle problems with high stochastic dimensionality. Within the context of the GP model, the incapability is due to the fact that the method relies on the Euclidean distance to define input-space correlations. Since the Euclidean distance becomes uninformative as the dimensionality of the input space increases [4], unless the number of data points available grows exponentially, an issue known as the curse of dimensionality. That is, directly reconstructing or approximating a function in a high dimensional space is not computationally feasible. To this end, considerable research efforts are focused on methodologies that can identify and exploit some special structure of the underlying mathematical model, and in particular some low dimensional structures. In such methods, it is assumed that the output of the model only depends on the inputs through its projection in a low dimensional subspace. One first identifies such a low-dimensional subspace and then construct the GP emulator in the obtained low dimensional subspace, where examples of this strategy include [11,17,27]. In this regard a variety of dimension reduction techniques can be used to identify the low dimensional subspace, such as the sliced inverse regression (SIR) [16], sliced average variance estimation (SAVE), among many others (see, e.g., [7]). Most of these dimension reduction methods only utilize the input-output data points to identify the low dimensional subspace. On the other hand, in many practical computer models, often the gradient of the output with respect to the input parameters is also available; for example, if the computer model is solved with a finite element method, the gradient can be readily computed using the adjoint method [5]. When the model gradients are available, such information can also be used to identify the low dimensional structure. To this end, a gradient based dimension reduction approach, the so-called active subspace (AS) method, has been developed [25,9,8]. Loosely speaking, the method identifies a low dimensional subspace of the original input space, termed as the active subspace, by considering the expectation of the gradient outer product [8]. Further details and theoretical analysis of the AS method are provided in Section 2.

It is important to note here that all the aforementioned methods, with or without using gradient, are all based on the assumption that a global low dimensional structure of the underlying simulation model exists. In other word, it is assumed that the same low dimensional structure of the simulation model exists across the state space of the input parameter. As a result, the emulator is constructed within this global low dimensional subspace. This assumption, however, is not always true in practice, as many computer models are highly complex and may exhibit substantially different behaviors in different parameter regimes. In this case, assuming a global low dimensional structure of the underlying model becomes inappropriate. Rather, it is more flexible and sensible to assume that, should such low dimensional structures exist, they may only be valid locally, i,e., in a subregion of the entire space. Based on

3

this idea, we propose a local dimensionality reduction scheme, which we call clustered active subspace method (CAS). The main idea of the method is to partition the original input space into a number of subdomains, where each subdomain has its own low dimensional structure. We proposed to identify such subdomains using a clustering method based on the data points. Finally, we construct a local and low dimensional GP emulator of the underlying simulation model in each subdomain, and as a result, we obtain a set of local GP emulators to mimic the behaviors of the underlying model.

The rest of the paper is organized as follows. In Section 2, we describe the problem and give reviews to the main technical ingredients of our methods. Then in Section 3 we present the clustered active subspace method and the construction of local GP based on it. In Section 4, we provide numerical results for three examples: two mathematical functions with known low dimensional structures and a PDE based model. Finally Section 5 offers some closing remarks.

2   Technical preliminaries

2.1   Problem setup

We consider in this work a computational expensive function $y = f(\mathbf{x})$, where $\mathbf{x}$ is a d dimensional random variable with distribution $\pi(\mathbf{x})$ and y is a scalar. Assume that the support of $\pi(\mathbf{x})$, Ω is a compact subspace of R^d. In practice, the function $f(\mathbf{x})$ can be described by a complex PDE or a large scale ODE system. When solving such systems, in addition to evaluating the function values, the gradient of the function can often be obtained as a byproduct. We assume that this is the case for problems considered here. Suppose that we have a set of data points

$$D = \{(\mathbf{x}^{(n)}, y^{(n)}) | y^{(n)} = f(\mathbf{x}^{(n)})\}_{n=1}^{N}, \tag{2.1}$$

with $\mathbf{x}^{(n)}$ drawn according to $\pi(\mathbf{x})$, and our goal here is to construct a surrogate model for $f(\mathbf{x})$ which can be used in other computational tasks. More precisely, we plan to use the GP regression to construct the surrogate, and the method is described in Section 2.2. A main challenge here is that in many practical problems the dimensionality d is large (for example $d \geq 100$), and constructing GP model or any other surrogates in such a high-dimensional setting is extremely difficult. As has been discussed in Section 1, we will first reduce the dimensionality of the problem before constructing the GP model.

4

2.2 Gaussian Process

The GP regression performs a nonparametric regression in a Bayesian framework [31]. The basic idea of the GP method is to assume that the function of interest $f(\mathbf{x}, \epsilon)$ is a realization from a Gaussian random field, whose mean is $\mu(\mathbf{x})$ and covariance is specified by a kernel function $k(\mathbf{x}, \mathbf{x}')$, namely,

$$\mathrm{Cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}').$$

The kernel $k(\mathbf{x}, \mathbf{x}')$ is positive semidefinite and bounded. Commonly used kernel functions include the squared exponential kernel, the Matern kernel and the rational quadratic kernel.

Now given the data points $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$, we want to predict the value of y at a new point $\mathbf{x}$. Now we let $\mathbf{X} := \left[\mathbf{x}^{(1)}; \ldots; \mathbf{x}^{(N)}\right]$, and $\mathbf{Y} = [y^{(1)}, \ldots, y^{(N)}]^{\mathrm{T}}$. Under the GP assumption, it is easy to see that the joint distribution of $(\mathbf{Y}, y)$ is Gaussian,

$$\begin{bmatrix} \mathbf{Y} \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{matrix} \mu(\mathbf{X}) \\ \mu(\mathbf{x}) \end{matrix}, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) + \sigma_M^2 I & K(\mathbf{X}, \mathbf{x}) \\ K(\mathbf{x}, \mathbf{X}) & K(\mathbf{x}, \mathbf{x}) \end{bmatrix} \right), \tag{2.2}$$

where σ_M^2 is the variance of observation noise, I is an identity matrix, and the notation $K(\mathbf{A}, \mathbf{B})$ denotes the matrix of the covariance evaluated at all pairs of points in set $\mathbf{A}$ and in set $\mathbf{B}$ using the kernel function $k(\cdot, \cdot)$. It follows immediately from Eq. (2.2) that the conditional distribution $\pi_{GP}(y|\mathbf{x}, \mathbf{X}, \mathbf{Y})$ is also Gaussian:

$$\pi_{GP}(y|\mathbf{x}, \mathbf{X}, \mathbf{Y}) = \mathcal{N}(\mu_{\mathrm{pos}}, \sigma_{\mathrm{pos}}^2), \tag{2.3a}$$

where the posterior mean and variance are,

$$\mu_{\mathrm{pos}}(\mathbf{x}) = \mu(\mathbf{x}) + k(\mathbf{x}, \mathbf{X})(k(\mathbf{X}, \mathbf{X}) + \sigma_M^2 I)^{-1}(\mathbf{Y} - \mu(\mathbf{x})), \tag{2.3b}$$

$$\sigma_{\mathrm{pos}}^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - k(\mathbf{x}, \mathbf{X})(k(\mathbf{X}, \mathbf{X}) + \sigma_M^2 I)^{-1} k(\mathbf{X}, \mathbf{x}). \tag{2.3c}$$

There are also a number of technical issues in the GP model, such as choosing the kernel function and determining the hyperparameters. For detailed discussion of these matters, we refer the readers to [31]. As is well known, the GP method fails when the dimensionality of data is extremely high. It is one of the key issues to be addressed when applying GP based methods to high dimensional problems.

2.3 Active Subspace method

We now discuss the dimension reduction technique that will be used in this work. Recall that it is assumed that the gradient of the target function is

available in the problems under consideration. Thus we use the so-called active subspapce (AS) method, which identifies the low dimensional subspace by using the gradient information. In what follows we give a brief introduction to AS, largely following [9].

Consider a target function $y = f(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^d$ that is absolutely continuous and square-integrable with respect to a probability density function $\pi : \mathbb{R}^d \to \mathbb{R}_+$. Now recall that we can compute the gradient of f denoted by the column vector $\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_d} \end{bmatrix}^{\mathrm{T}}$. Next we shall define the $d \times d$ matrix $\mathbf{C}$ as

$$\mathbf{C} = \mathbb{E}_{\pi(\mathbf{x})}[(\nabla_{\mathbf{x}} f)(\nabla_{\mathbf{x}} f)^{\mathrm{T}}] \tag{2.4}$$

where we assume that the products partial derivatives are integrable. Since $\mathbf{C}$ is symmetric positive definite, it can be decomposed as,

$$\mathbf{C} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{\mathrm{T}} \tag{2.5}$$

where $\boldsymbol{\Lambda} = diag(\lambda_1, \cdots, \lambda_d)$ is a diagonal matrix with the eigenvalues of $\mathbf{C}$ in decreasing order, $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$, and $\mathbf{V} \in \mathbb{R}^{d \times d}$ an orthonormal matrix consists of eigenvectors of $\mathbf{C}$.

Assuming that the reduced dimensionality is r, we can partition the eigenvalues and eigenvectors into two parts:

$$\boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\Lambda}_1 & \\ & \boldsymbol{\Lambda}_2 \end{bmatrix}, \ \mathbf{V} = [\mathbf{V}_1 \ \mathbf{V}_2], \tag{2.6}$$

where $\boldsymbol{\Lambda}_1 = diag(\lambda_1, \cdots, \lambda_r)$, $\mathbf{V}_1 = [\mathbf{v}_{11} \cdots \mathbf{v}_{1r}]$, and $\boldsymbol{\Lambda}_2$, $\mathbf{V}_2$ are defined analogously. We then define the rotated coordinates $\mathbf{z}_1 \in \mathbb{R}^r$ and $\mathbf{z}_2 \in \mathbb{R}^{d-r}$ by

$$\mathbf{z}_1 = \mathbf{V}_1^{\mathrm{T}} \mathbf{x}, \ \mathbf{z}_2 = \mathbf{V}_2^{\mathrm{T}} \mathbf{x}. \tag{2.7}$$

One then takes $\mathbf{V}_1^{\mathrm{T}}$ as the low dimensional projection matrix and $\mathbf{z}_1$ as the dimension reduced variable. That is to say, one can now approximate the original function $f(\mathbf{x})$ with a function, say $G(\mathbf{z}_1)$, defined on the reduced space $\{\mathbf{z}_1 = \mathbf{V}_1^{\mathrm{T}} \mathbf{x} | \mathbf{x} \in \Omega\}$. In particular we can define function $G(\mathbf{z}_1)$ as

$$G(\mathbf{z}_1) = \mathbb{E}[f|\mathbf{z}_1] = \int_{\mathbf{z}_2} f(\mathbf{V}_1 \mathbf{z}_1 + \mathbf{V}_2 \mathbf{z}_2) \pi_{Z_2|Z_1}(\mathbf{z}_2) d\mathbf{z}_2, \tag{2.8}$$

and G is the best mean-squared approximation of f given $\mathbf{z}_1$ which follows from the so-called *law of the unconscious statistician* [9]. Thus an approximation of $f(\mathbf{x})$ can be constructed via $G(\mathbf{z}_1)$:

$$f(\mathbf{x}) \approx F(\mathbf{x}) \equiv G(\mathbf{V}_1^{\mathrm{T}} \mathbf{x}). \tag{2.9}$$

The following theorem provides an error bound for F in terms of eigenvalues of $\mathbf{C}$ [9]:

Theorem 2.1 *The mean squared error of F defined in (2.9) satisfies*

$$\mathbb{E}[(f - F)^2] \leq \alpha(\lambda_{r+1} + \cdots + \lambda_d), \tag{2.10}$$

where α is a constant that depends on the domain $\mathcal{X}$ and the distribution π.

It should be clear that in practice it is usually not possible to evaluate Eq.(2.4) directly. Instead, one often approximates the integral via Monte Carlo simulation. That is, assuming that the observed inputs are drawn from $\pi(\mathbf{x})$, one approximates $\mathbf{C}$ using the observed gradients by:

$$\widetilde{\mathbf{C}} = \frac{1}{N} \sum_{i=1}^{N} (\nabla_{\mathbf{x}} f(\mathbf{x}^{(i)}))(\nabla_{\mathbf{x}} f(\mathbf{x}^{(i)}))^{\mathrm{T}}. \tag{2.11}$$

Next the eigenvalues and eigenvectors of $\widetilde{\mathbf{C}}$ are computed using the singular value decomposition (SVD). As is suggested in [9], the reduced dimensionality r can be determined by considering the spectrum of $\mathbf{C}_N$, and one possible approach is to require that

$$\sum_{i=1}^{r} \lambda_i \geq \rho \sum_{i=1}^{d} \lambda_d, \tag{2.12}$$

for a prescribed ratio $\rho \in [0, 1]$. We refer the readers to [9] for more details and other possible methods for determining r. Next we can define,

$$\hat{G}(\mathbf{z}_1) = \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{V}_1 \mathbf{z}_1 + \mathbf{V}_2 \mathbf{z}_2^{(i)}), \tag{2.13}$$

where the $\mathbf{z}_2^{(i)}$'s are drown independently from the conditional density $\pi(\mathbf{z}_2|\mathbf{z}_1)$, and it follows that we can obtain an approximation of f,

$$f(\mathbf{x}) \approx \hat{F}(\mathbf{x}) \equiv \hat{G}(\mathbf{V}_1^{\mathrm{T}} \mathbf{x}). \tag{2.14}$$

Similarly we can then derive an error bound for the Monte Carlo approximation $\hat{F}$, of f in terms of eigenvalues of $\mathbf{C}$ as follows [9]:

Theorem 2.2 *The mean squared error of $\hat{F}$ defined in Eq. (2.14) satisfies*

$$\mathbb{E}[(f - \hat{F})^2] \leq \alpha(1 + \frac{1}{N})(\lambda_{r+1} + \cdots + \lambda_d), \tag{2.15}$$

where α is a constant that depends on the domain $\mathcal{X}$ and the distribution π.

The proofs of Theorems 2.1 and 2.2 can be found in [9] and will not be repeated here.

3 The Clustered Active subspace based local GP emulator

3.1 Clustered Active Subspace method

Using the AS method we can now construct a low dimensional subspace for function $f(\mathbf{x})$. However, as has been discussed in Section 1, in many practical problems with complex physics, a global low dimensional structure, i.e. one that is applicable in the entire parameter domain, may not exist. Instead, different parameter regimes may have distinct low dimensional structures. As a result, a global dimension reduction strategy may not apply. In what follows we present a clustered AS method to conduct dimension reduction for such problems.

The main idea of our method is the following. Suppose that the function of interest admits different low dimensional structures in different disjoint subdomains of the original input space, and ideally if we can identify these subdomains and the low dimensional subspace in each of them, we can expect that the GP emulator constructed locally (i.e., to only use data points in the same subdomain as the new point of interest) enjoys a better performance than the emulator constructed globally. First we can write the distribution $\pi(\mathbf{x})$ as a mixture:

$$\pi(\mathbf{x}) = \sum_{j=1}^{J} w_j \pi_j(\mathbf{x}), \quad \sum_{j=1}^{J} w_j = 1, \tag{3.1}$$

where each π_j has a support Ω_j satisfying

$$\Omega_j \cap \Omega_{j'} = \emptyset \text{ for } \forall \; j \neq j', \qquad \cup_{j=1}^{J} \Omega_j = \Omega.$$

In what follows we refer each π_j as a distribution cluster. Now we assume that the function $f(\mathbf{x})$ has different properties in different region Ω_j and for each π_j we can define

$$\mathbf{C}_j = \mathbb{E}_{\pi_j(\mathbf{x})}[(\nabla_{\mathbf{x}} f)(\nabla_{\mathbf{x}} f)^{\mathrm{T}}] \tag{3.2}$$

and compute the associated "clustered" active subspace accordingly. By going through the same procedure as described in Section 2.3, we can obtain a low dimensional projection matrix $\mathbf{V}_{1,j}$ and its compliment $\mathbf{V}_{2,j}$ for each distribution cluster π_j. As a result we obtain a set of J low-dimensional projections $\{\mathbf{V}_{1,j}\}_{j=1}^{J}$, and if the function's structure is very different with respect to different clusters/subdomains, these low-dimension projections are significantly different from each other. Defining $\mathbf{z}_{1,j} = \mathbf{V}_{1,j}^{\mathrm{T}} \mathbf{x}$ and $\mathbf{z}_{2,j} = \mathbf{V}_{2,j}^{\mathrm{T}} \mathbf{x}$ for $i = 1, \cdots, J$, we obtain a set of local approximations,

$$G_j(\mathbf{z}_{1,j}) = \mathbb{E}_{\pi_j}[f|\mathbf{z}_{1,j}] = \int_{\mathbf{z}_{2,j}} f(\mathbf{V}_{1,j}\mathbf{z}_{1,j} + \mathbf{V}_{2,j}\mathbf{z}_{2,j})\pi_j(\mathbf{z}_{2,j}|\mathbf{z}_{1,j})d\mathbf{z}_{2,j}, \tag{3.3}$$

8

and for any $\mathbf{x} \in \Omega_j$,

$$f(\mathbf{x}) \approx F_j(\mathbf{x}) \equiv G_j(\mathbf{V}_{1,j}^{\mathrm{T}}\mathbf{x}). \tag{3.4}$$

Consequently we can define a global approximation for $f(\mathbf{x})$ as,

$$f(\mathbf{x}) \approx F(\mathbf{x}) = \sum_{j=1}^{J} F_j(\mathbf{x}) I_{\Omega_j}(\mathbf{x}), \tag{3.5}$$

where I_{Ω_j} is an indicator function,

$$I_{\Omega_j}(\mathbf{x}) = \begin{cases} 1, \mathbf{x} \in \Omega_j, \\ 0, \mathbf{x} \notin \Omega_j. \end{cases} \tag{3.6}$$

The error bound of F defined in Eq. (3.5) is given by the following theorem.

Theorem 3.1 *Let $\{\lambda_{i,j}\}_{i=1}^{d}$ be the eigenvalues of C_j in a descending order and r_j be the reduced dimensionality associated with the j-th cluster. The mean squared error of F defined in Eq. (3.5) satisfies*

$$\mathbb{E}_\pi[(f - F)^2] \leq \sum_{j=1}^{J} \alpha_j(\lambda_{r_j+1,j} + \cdots + \lambda_{d,j}), \tag{3.7}$$

where α_j's are constants that depend on the domains Ω_j's and the distribution clusters π_j's.

Next we shall establish the error analysis for the Monte Carlo estimation as is done for the AS method. First suppose that we have samples $\mathbf{x}^{(1)}, \cdots, x^{(N)}$ drawn from the distribution cluster π_j, and we can estimate the sample variance,

$$\widetilde{\mathbf{C}}_j = \frac{1}{N} \sum_{i=1}^{N} (\nabla_\mathbf{x} f(\mathbf{x}^{(i)}))(\nabla_\mathbf{x} f(\mathbf{x}^{(i)}))^{\mathrm{T}}, \tag{3.8}$$

and once again we can obtain the two matrices $\mathbf{V}_{1,j}$ and $\mathbf{V}_{2,j}$ associated with $\widetilde{\mathbf{C}}_j$. Let $\mathbf{z}_{1,j} = \mathbf{V}_{1,j}^{\mathrm{T}}\mathbf{x}$ and $\mathbf{z}_{2,j} = \mathbf{V}_{2,j}^{\mathrm{T}}\mathbf{x}$ and we get

$$\hat{G}_k(\mathbf{z}_1^{(k)}) = \frac{1}{N_j} \sum_{i=1}^{N_j} f(\mathbf{V}_1^k \mathbf{z}_1^{(k)} + \mathbf{V}_2^k \mathbf{z}_2^{(k)^i}), \tag{3.9}$$

in which $\mathbf{z}_2^{(k)^i}$ are drawn i.i.d. from the conditional distribution $\pi_j(\mathbf{z}_{2,j}|\mathbf{z}_{1,j})$. Finally we obtain an approximation $\hat{F}$ of f are as follows:

$$f(\mathbf{x}) \approx \hat{F}(\mathbf{x}) = \sum_{j=1}^{J} \hat{F}_j(\mathbf{x}) I_{\Omega_j}(\mathbf{x}), \tag{3.10}$$

where

$$\hat{F}_j(\mathbf{x}) = \hat{G}_j(\mathbf{V}_{1,j}^{\mathrm{T}}\mathbf{x}), \tag{3.11}$$

and $I_{\Omega_j}(\cdot)$ is the indicator function defined in Eq. (3.6). Then we can derive an error bound for the Monte Carlo approximation $\hat{F}$ of f which is stated by the following theorem:

Theorem 3.2 *The mean squared error of $\hat{F}$ defined in Eq. (3.10) satisfies*

$$\mathbb{E}[(f - \hat{F})^2] \leq \sum_{j=1}^{J} \alpha_j (1 + \frac{1}{N_j})(\lambda_{r_j+1} + \cdots + \lambda_d), \qquad (3.12)$$

where r_j denotes the reduced dimensionality in Ω_j and α_j's are constants that depend on the domains Ω_j's and the distributions π_j's.

In the Appendix we provide a proof for Theorem 3.2 and the proof for Theorem 3.1 proceeds similarly and thus is omitted.

3.2 Gradient-based data clustering

In Section 3.1, we have provided the main framework of the clustered active subspace method. Clearly a key issue yet to be addressed is that the distribution clusters are not known in advance, and we need to identify the distribution clusters that represent different function structures. In particular in Section 3.1 we have assumed that it is known in advance that the samples $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N_j)}$ are drawn from each distribution cluster $\pi_j(\mathbf{x})$. However, in reality all the samples are drawn from the distribution π and we need to partition them into different data clusters, which implicitly defines the distribution clusters as well. In other word, we do not need to explicitly obtain the distribution clusters, and for implementation purpose we only need to cluster the data points.

The main idea here is to cluster the data points based on the gradient information which may reveal the local structure of the function $f(\mathbf{x})$. For this purpose we choose to use the hierarchical clustering algorithm, which groups data over a variety of scales by creating a cluster tree or dendrogram. An important feature of hierarchical clustering is that the tree is not a single set of clusters, but rather a multilevel hierarchy, where clusters at one level are joined as clusters at the next level. This feature is important as it allows users to decide the level or scale of clustering based on the specific application. We refer to [15] for more details of the hierarchical clustering methods.

In hierarchical clustering, a key is to define an appropriate distance that can measure the similarity or dissimilarity between the data points, and this distance should represent the properties of the data points that users hope to distinguish. In our problem, as has been mentioned, we expect that different clusters represent different low-dimensional structures of the function, which

in the AS framework is encoded in the gradient information. This motivates us to cluster the data based on the gradient. To do so, we define the absolute cosine distance measure as follows:

$$\Delta(\mathbf{x}, \mathbf{x}') = 1 - |\cos(\mathbf{g_x}, \mathbf{g_{x'}})|, \qquad (3.13)$$

where $\mathbf{g_x}, \mathbf{g_{x'}}$ are the corresponding gradient of $\mathbf{x}, \mathbf{x}'$, and we conduct a clustering of the data points based on this distance. This distance function can be further generalized, taking into account of the Euclidean distance between points and combining it with the gradient similarity. Namely we set the distance to be

$$\Delta_\eta(\mathbf{x}, \mathbf{x}') = \eta * (1 - |\cos(\mathbf{g_x}, \mathbf{g_x}')|) + (1 - \eta) * (||\mathbf{x} - \mathbf{x}'||_2 / \sqrt{d}), \qquad (3.14)$$

where $\eta \in [0, 1]$ is a hyper-parameter to balance the influence between the discrepancy on locations and gradients of $\mathbf{x}$ and d is the dimension of $\mathbf{x}$. We divide the euclidean distance by $\sqrt{d}$ to adjust the second term to be in the same level of scales with the first absolute cosine distance. It should also be clear that, when $\eta = 1$, the distance is reduced to the absolute cosine distance in Eq. (3.13) which only uses the gradient information. Indeed, in most of the problems, we can simply use the gradient-based distance measure, and the Euclidean distance is introduced as an insurance policy to prevent certain extreme scenarios where the gradient based distance may fail. Thus in practice we recommend to choose η to be close to 1.

Another important issue in hierarchical clustering is to choose the linkage criterion, and in our numerical tests we have found that the unweighted average linkage clustering [15] has the best performance overall thus is used in this work. That said, the method proposed does not reply on any specific choice of the linkage function. Next we discuss the procedure of the gradient based clustering. We modify the notation defined by Eq. (2.1), extending the training set to include the gradient:

$$D = \{(\mathbf{x}^{(n)}, y^{(n)}, \mathbf{g}^{(n)}) | y^{(n)} = f(\mathbf{x}^{(n)}), \ \mathbf{g}^{(n)} = \nabla_\mathbf{x} f(\mathbf{x}^{(n)})\}_{n=1}^{N}. \qquad (3.15)$$

and present the complete procedure of the clustered active subspace method in Algorithm 1. Finally one can see that the CAS method requires the number of clusters J as an input, and in our method it is determined via a k-fold cross-validation [1] where details are provided in B.

3.3   Local GP emulator with clustered dimensionality reduction

In Section 3.2 we have discussed the method to obtain the data clusters as well as the dimension reduction projection matrices associated to them. To

---
Algorithm 1 Clustered active subspace dimensionality reduction(CAS)

Require: Training set D; the number of clusters J; the dimension reduction ratio ρ.

Ensure: $\{D_j, \mathbf{V}_{1,j}\}_{j=1}^J$, where D_j are the data clusters and $\mathbf{B}_j$ are the associated DR projection matrices.

   $\{D_j\}_{j=1}^J \leftarrow$ cluster data set D into J clusters according to distance (3.14);

   for $j = 1, \cdots, J$ **do**

      $\mathbf{C}_j = \frac{1}{card(D_j)} \sum_{(\mathbf{x}^{(n)}, y^{(n)}, \mathbf{g}^{(n)}) \in D_j} (\mathbf{g}^{(n)})(\mathbf{g}^{(n)})^{\mathrm{T}}$;

      Conduct SVD to matrix $\mathbf{C}_j$ obtaining $\mathbf{C}_j = \mathbf{V}_j \mathbf{\Lambda}_j \mathbf{V}_j^{\mathrm{T}}$;

      Determine the reduced dimensionality r_j based on ratio ρ;

      $\mathbf{V}_{1,j} = [\mathbf{v}_{j1} \cdots \mathbf{v}_{jr_j}]$;

   end for

   return $\{D_j, \mathbf{V}_{1,j}\}_{j=1}^J$
---

make use of them, another key issue is to determine which cluster a new point belongs to, which can be posed as a classification problem. It is important to note that, practically we only have the knowledge of the input parameter itself, and we do not know the function value or the gradient. Since the training data points have been clustered and labelled, we construct a classifier based on the parameter value $\mathbf{x}$ only, in a supervised manner. In particular we choose to construct the classifier with the Support Vector Machine (SVM) model [28] mainly for its simplicity, while noting that other classification methods can also be used here. To start, we construct the data set for the classification problem as

$$D_c = \{(\mathbf{x}^{(n)}, b^{(n)})\}_{n=1}^N, \tag{3.16}$$

where $b^{(n)}$ is the label assigned to cluster D_j if $\mathbf{x}^{(n)} \in D_j$. Using the classification data set D_c we are able to train a SVM classifier denoted as $b = \mathrm{SVM}(\mathbf{x})$. We omit the training procedure of the SVM model and interested readers may consult [28]. Now for any given new point $\mathbf{x}^*$, we first use the trained SVM model to determine which cluster it belongs to, and then we conduct dimension reduction of all the data points in the chosen cluster with the associated dimension reduction projection matrix. Finally a local GP emulator is constructed with the dimension-reduced data points for the new point $\mathbf{x}^*$. More precisely, suppose that the cluster predicted by the SVM model is D_{j^*}, with dimension reduction matrix $\mathbf{V}_{1,j^*}$, and we define the following data set:

$$D_{GP} = \{(\mathbf{z}^{(i)}, y^{(i)}) | \mathbf{z}^{(i)} = \mathbf{V}_{1,j^*}^{\mathrm{T}} \mathbf{x}^{(i)}, \ \forall \mathbf{x}^{(i)} \in D_{j^*}\}, \tag{3.17}$$

for the construction of the GP model. Finally the GP model is constructed using D_{GP} with the procedure outlined in Section 2.2, which is then used to predict the value of $f(\mathbf{x}^*)$. We reinstate here that the GP model is constructed on the low dimensional subspace obtained by $\mathbf{V}_{1,j^*}$ and only with data points in D_{j^*}, and so it is regarded as a local GP model. We summarize the complete procedure for constructing such a local GP model in Alg. 2. Finally a number

of remarks are listed in order:

- An important question is the number of data points used in the method. While noting that the actual number of data points needed is problem-dependent, we reinstate that here we mainly consider the situation where the number of data points is typically limited, i.e., insufficient to conduct regression in the high-dimensional space. The dimension reduction technical should be able to improve the regression performance in such a situation.
- Compared to the standard AS, the proposed CAS method is more computationally expensive mainly because the additional clustering procedure is needed. However, as has been mentioned earlier, we consider in this work problems with exceedingly expensive computer models, and as such the dimension reduction procedure is not a main contributor to the total computational cost, as long as it does not involve simulating the computer model.

Algorithm 2 The construction of the low dimensional local GP model

---

Require: D_c classification data set; test point x^*.
Ensure: testing prediction y^*.

Train a SVM model based on the training set D_c, denoted as $b = \text{SVM}(\mathbf{x})$;
Compute $b^* = \text{SVM}(\mathbf{x}^*)$ and let $D_{j^*}, \mathbf{V}_{1,j^*}$ respectively be the data cluster and projection matrix corresponding to label b^*;
Obtain the training set for GP:

$$D_{GP} = \{(\mathbf{z}^{(i)}, y^{(i)}) | \mathbf{z}^{(i)} = \mathbf{V}_{1,j^*}^{\mathrm{T}} \mathbf{x}^{(i)}, \ \forall \mathbf{x}^{(i)} \in D_{j^*}\};$$

Construct the GP model f_{j^*} with data set D_{GP}, and use $f_{j^*}(\mathbf{V}_{1,j^*}^{\mathrm{T}} \mathbf{x}^*)$ to predict the value of $f(\mathbf{x}^*)$.

---

4 Numerical results

4.1 Example 1: a piece-wise function

To illustrate the effectiveness of the proposed approach, we first consider a piece-wise function which has four distinct low dimensional structures in four disjoint regions. Specifically the function is defined on $\mathbf{x} \in [-1, 1]^{50}$, and admits the form,

$$f(\mathbf{x}) = \begin{cases} (1 + x_3 + x_4)x_5, & x_1 < 0 \ and \ x_2 < 0 \ (\text{Region1}), \\ (1 + x_6 + x_7)(x_8 + x_9), & x_1 < 0 \ and \ x_2 \geq 0 \ (\text{Region2}), \\ 1 + x_{10} + x_{11}, & x_1 \geq 0 \ and \ x_2 < 0 \ (\text{Region3}), \\ (1 + x_{12})x_{13}, & x_1 \geq 0 \ and \ x_2 \geq 0 \ (\text{Region4}). \end{cases} \quad (4.1)$$

This function relies on completely different dimensions of the input space in the four different regions, and we want to test if the ability to identify these regions and their individual low dimensional structures can improve the performance of the GP emulator. Also since the low dimensional structure of this problem is analytically available, we can use it to validate the dimension reduction results. To this end we hope that the proposed CAS method can correctly identify the regions (in the form of clusters) and the reduced dimensions in each of them. In the numerical tests, the distribution of $\mathbf{x}$ is taken to be uniform defined on $\mathbf{x} \in [-1, 1]^{50}$, and 1000 training samples and 5000 testing samples are drawn from the distribution. Our experiments involves two main steps: conducting a dimension reduction and constructing the GP model in the low dimensional subspace. First we want to examine if the standard AS and the CAS methods can correctly identify the DR directions of this function (one can see from Eq. (4.1) that the function admits totally 7 actual DR directions). To do so, we compare the actual DR directions with those identified with both AS and CAS (with 4 clusters) methods: for regions 1, 2 and 4, each has two DR directions and region 3 only has one. In Fig. 1 we show the DR directions in all the four regions, and for the regions with two DR directions they are marked with different colors blue and red. As one can see from the figure, the directions computed with both AS and CAS agree well with the actual ones, while, in addition to the DR directions, CAS can also correctly identify the clusters or regions where the low dimensional structures are different.

Next we shall demonstrate that the ability of CAS to identify these different clusters can help improve the performance of the GP emulator. In addition to the CAS-LGP algorithm, we also employ AS, SIR [16] and SAVE [10] to reduce the dimensionality globally and then construct the GP emulator in the resulting dimension-reduced space. For comparison purposes, we also conduct the test of direct GP emulator in the original space, without dimension reduction. In CAS-LGP, we consider two cases of the distance function used in the clustering: $\eta = 1$ and $\eta = 0.5$, where the first case only considers the gradient information while the 2nd combines gradient similarity with the Euclidean distance. In all the examples, the predictive performance of GP emulator is measured by the normalized mean-square-error (NMSE):

$$NMSE = \frac{\sum_{n=1}^{N}(f(\mathbf{x}^{(n)}) - \hat{f}(\mathbf{x}^{(n)}))^2}{\sum_{n=1}^{N} f(\mathbf{x}^{(n)})^2}, \qquad (4.2)$$

where $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}$ are test samples drawn from the distribution $\pi(\mathbf{x})$, $f(\mathbf{x})$ is the actual function and $\hat{f}$ is the posterior mean of the GP model. For the global dimension reduction methods, we test five different numbers of reduced dimensions: $r = 1, 2, 3, 4, 7$ (we choose 7 because the total number of intrinsic dimensions is 7), and for CAS-LGP we compute the results with 2 to 4 clusters each with $r = 1, 2, 3, 4$ reduced dimensions, where we note that when the number of clusters is taken to be 1, the method reduces to the
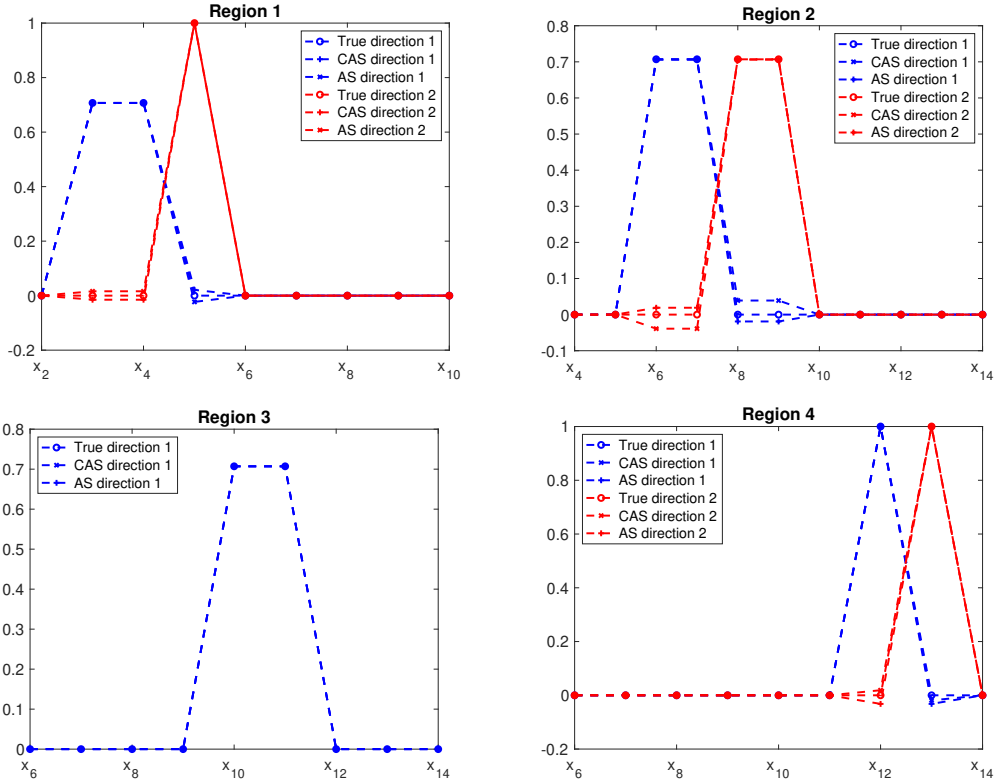
Fig. 1. DR directions in four regions.

standard AS. We compute the NMSE for each case and summarize all the results in Table 1. First of all, as we can see from the table, for all the cases where the reduced dimensionality is from 1 to 4, CAS-LGP with four clusters yield by far the best results. Moreover, interestingly the table also illustrates that, even though the standard AS can correctly identify the 7 DR directions, the resulting GP emulator is severely inaccurate, which is even less accurate than the GP constructed in the original space. Lastly we note that the distance function in this example seems to have little impact on the performance, as the results of the two cases ($\eta = 1$ and $\eta = 0.5$) are nearly identical. This example demonstrates that, if the model of interest has clearly different low dimensional structure in different regions, the proposed method can effectively identify these regions and their associated low dimensional structures, which in turn produces an accurate local GP emulator.

4.2 A Gaussian mixture

The first example is a piece-wise function and it is therefore non-smooth. To test the performance with smooth functions, we consider the following mixture

15

Table 1
The NMSE results of the piece-wise example.

$\eta = 1$							
Method	CAS-LGP			AS-GP	SIR-GP	SAVE-GP	GP
k d	2	3	4				
1	73.1%	68.5%	**41.1%**	91.6%	72.6%	88.6%	
2	72.1%	57.6%	**20.3%**	86.1%	73.2%	85.3%	
3	65.1%	56.6%	**20.3%**	85.3%	72.7%	86.0%	67.5%
4	61.9%	55.7%	**20.3%**	80.5%	72.2%	86.2%	
7				81.7%	71.6%	82.5%	
$\eta = 0.5$							
Method	CAS-LGP			AS-GP	SIR-GP	SAVE-GP	GP
k d	2	3	4				
1	73.1%	68.5%	**41.1%**	91.6%	72.6%	88.6%	
2	72.1%	57.6%	**20.3%**	86.1%	73.2%	85.3%	
3	65.1%	56.6%	**20.3%**	85.3%	72.7%	86.0%	67.5%
4	61.9%	55.7%	**20.3%**	80.5%	72.2%	86.2%	
7				81.7%	71.6%	82.5%	

of Gaussian functions:

$$f(\mathbf{x}) = \sum_{j=1}^{J} \theta_j \exp(-\frac{(\mathbf{x} - \mathbf{c}_j)B_j B_j^{\mathrm{T}}(\mathbf{x} - \mathbf{c}_j)^{\mathrm{T}}}{2\sigma_j^2}) \qquad (4.3)$$

with B_j being the low dimensionality projection matrix of size $d \times l$ in which $l \ll d$. Here we set $d = 50$, $l = 2$ and $J = 3$ in this experiment, which means that the mixture has three components, each mixture component admits two intrinsic dimensions and totally six intrinsic dimensions exists in this function. To keep it simple and clear, we specify the columns in each B_j to be orthogonal where entries are randomly drawn according to a binomial distribution such that it is either 0 or 1. It should be clear that the columns of these matrices represent the DR directions in each mixture, which are shown in Fig. 2. Centers $\mathbf{c}_j$ are also randomly chosen from a uniform distribution $U[0,1]^d$, σ_j's are all set to be 0.2, and the weights are taken to be $\theta = [0.41, 0.44, 0.26]$. The distribution of $\mathbf{x}$ is set to be $U[0,1]^d$.

Table 2
The NMSE results of the Gaussian mixture example.

$\eta = 1$							
Method	CAS-LGP			AS-GP	SIR-GP	SAVE-GP	GP
d \ k	2	3	4				
1	85.7%	**85.4**%	97.2%	89.8%	93.2%	95.6%	95.8%
2	25.2%	**23.6**%	36.6%	37.6%	92.5%	94.8%	
3	24.7%	**23.6**%	36.6%	37.3%	92.3%	94.3%	
4	24.9%	**23.6**%	36.6%	36.5%	92.9%	94.6%	
$\eta = 0.5$							
Method	CAS-LGP			AS-GP	SIR-GP	SAVE-GP	GP
d \ k	2	3	4				
1	85.7%	**85.4**%	113.0%	89.8%	93.2%	95.6%	95.8%
2	25.2%	**23.6**%	36.7%	37.6%	92.5%	94.8%	
3	24.7%	**23.6**%	36.6%	37.3%	92.3%	94.3%	
4	24.9%	**23.6**%	50.1%	36.5%	92.9%	94.6%	

In the numerical tests, once again 1000 samples are used as the training set and another 5000 are used as the test set for evaluating the accuracy of GP. In Figs. 2 we plot the actual DR directions as well as the directions computed with CAS and AS, where we can see that the two methods can both identify the DR directions rather accurately. Next we compare the performance of the GP emulators constructed with the same class of methods used in the first example and we summarize all the results in Table 2. These results are qualitatively similar to those for the first example. In particular CAS with 2, 3 or 4 clusters yield substantially lower approximation errors than AS and other methods, which demonstrates that CAS-LGP can take advantage of the local low dimensional structure to achieve better performance than the GP constructed on the global low dimensional subspace, even though, in both AS and CAS, the global subspace is correctly identified. This example shows that the proposed CAS-LGP method performs well in problems where the underlying model does not have a piecewise structure (i.e., the low dimensional structures are not clearly separated in disjoint subdomains).

Fig. 2. Principal directions in three different regimes
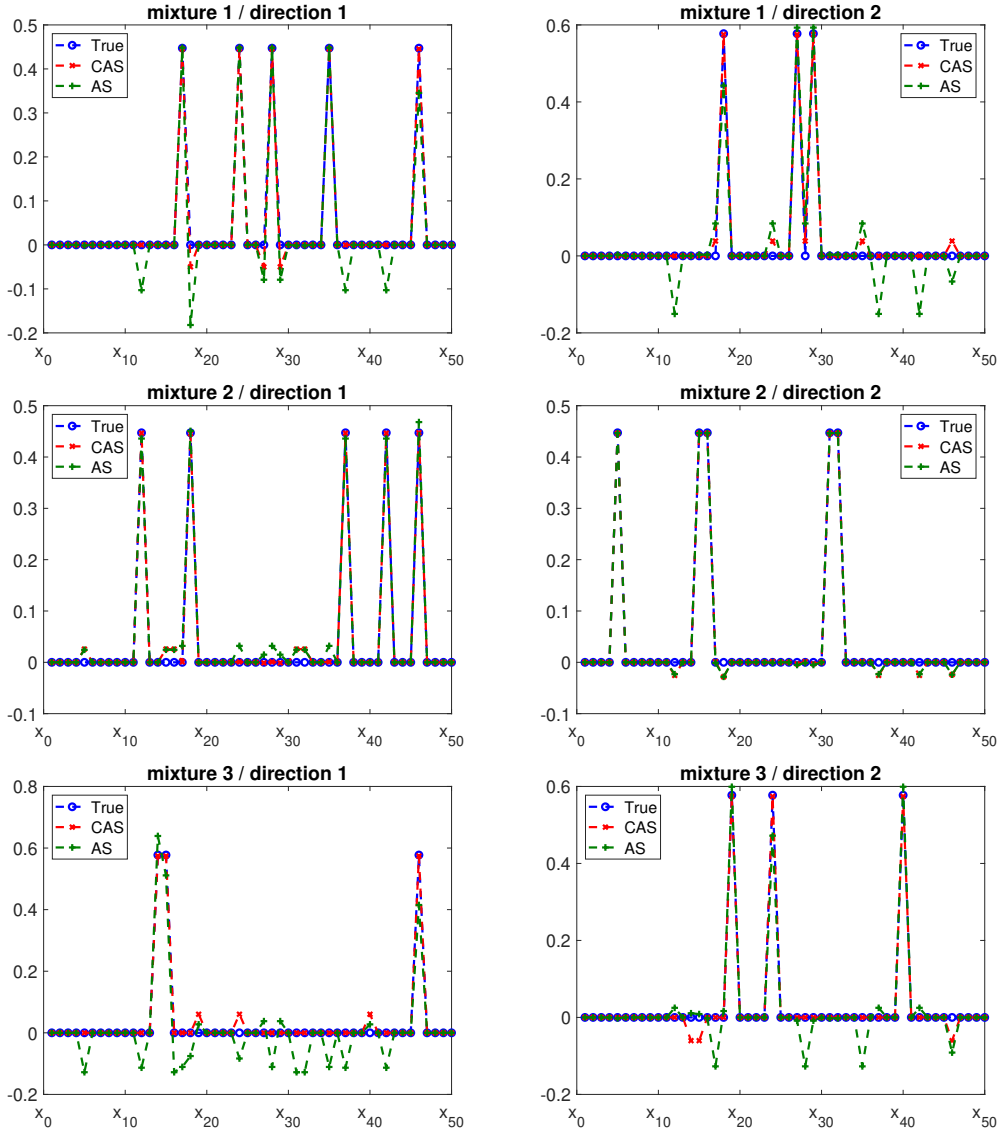
4.3 Elliptic PDE

Our last example is the following elliptic partial differential equation studied in [9] with slight modification:

$$-\nabla_{\mathbf{s}} \cdot (a(\mathbf{s}, \mathbf{x})\nabla_{\mathbf{s}}\mathbf{u}) = 1, \ \mathbf{s} \in [\mathbf{0}, \mathbf{1}]^{\mathbf{2}}. \tag{4.4}$$

We set homogeneous Dirichlet boundary conditions on the left, top, and bottom of the spatial domain; denote this boundary by Γ_1. The right side of the spatial domain denoted Γ_2 has a homogeneous Neumann boundary condition. That is,

$$\begin{aligned} u(\mathbf{s}) = 0, \ \mathbf{s} \in \Gamma_1, \\ \nabla u(\mathbf{s}) \cdot \mathbf{n} = 0, \ \mathbf{s} \in \Gamma_2. \end{aligned} \tag{4.5}$$

18

In this problem we assume that the coefficients $a = a(\mathbf{s})$ of the differential operator is a log-Gaussian random field. Moreover we represent $a(\mathbf{s})$ by a truncated Karhunen-Loève (KL) type expansion:

$$\log(a(\mathbf{s})) = \sum_{i=1}^{d} x_i \gamma_i \phi_i(\mathbf{s}), \qquad (4.6)$$

where the x_i are independent, identically distributed standard normal random variables, and in principle the $\{\phi_i, \gamma_i^2\}$ are the eigenpairs of a correlation operator. In this example we will modify the standard setting and the eigenpairs $\{\phi_i, \gamma_i^2\}$ will be specified later. Our function of interest is a linear functional of the solution [9]

$$f(\mathbf{x}) = \int_{\Gamma_2} u(\mathbf{s}, \mathbf{x})/|\Gamma_2| d\mathbf{s}. \qquad (4.7)$$

The PDE is discretized and solved using a finite element method on a triangulation mesh; then f and $\nabla_{\mathbf{x}} f$ can be computed as a forward and adjoint problem (see [2] for details). Recall that by the KL representation (4.6), we can specify $a(\mathbf{s}, \mathbf{x})$ by providing $\{\phi_i, \gamma_i\}$. First the KL bases ϕ_i are taken to be the eigenfunctions of the following covariance kernel function:

$$C(\mathbf{s}, \mathbf{s}') = \exp(-\frac{\|\mathbf{s} - \mathbf{s}'\|_1}{\beta}), \qquad (4.8)$$

where β is taken to be 1, and d is taken to be 100, implying that the dimensionality of the problem is 100. As is mentioned earlier, we modify the eigenvalues γ_i's so that they become a function of $\mathbf{x}$. Specifically we assume that the vector-valued function

$$\boldsymbol{\gamma}(\mathbf{x}) = (\gamma_1(\mathbf{x}), ..., \gamma_{100}(\mathbf{x})),$$

takes the following form,

$$\boldsymbol{\gamma}(\mathbf{x}) = \begin{cases} \gamma_{3,4,5,6,7,8,9,10}(\mathbf{x}) = 100 \ \& \ \gamma_{\Gamma \setminus \{3,4,5,67,8,9,10\}}(\mathbf{x}) = 0, & if \ x_1 < 0, x_2 < 0, \\ \gamma_{11,12,13,14,15,16,17,18}(\mathbf{x}) = 100 \ \& \ \gamma_{\Gamma \setminus \{11,12,13,14,15,16,17,18\}}(\mathbf{x}) = 0, & if \ x_1 \geq 0, x_2 < 0, \\ \gamma_{19,20,21,22,23,24,25,26}(\mathbf{x}) = 100 \ \& \ \gamma_{\Gamma \setminus \{19,20,21,22,23,24,25,26\}}(\mathbf{x}) = 0, & if \ x_1 < 0, x_2 \geq 0, \\ \gamma_{27,28,29,30,31,32,33,34}(\mathbf{x}) = 100 \ \& \ \gamma_{\Gamma \setminus \{27,28,29,30,31,32,33,34\}}(\mathbf{x}) = 0, & if \ x_1 \geq 0, x_2 \geq 0, \end{cases}$$
$$(4.9)$$

where Γ in this equation means the full index set $\{1, \cdots, 100\}$. Regarding the data, we first generate random samples from distribution $\pi(\mathbf{x})$, solve the PDE model, and finally evaluate the function of interest (4.7), yielding the input-output pairs, where 1000 pairs are used as the training set and 400 are used for testing. It is important to mention here that for the 1000 training data, the gradient of the target function is also obtained.

First we conduct the comparison of the NMSE results for the same set of methods as those in the first two examples, which are shown in Table 3. We
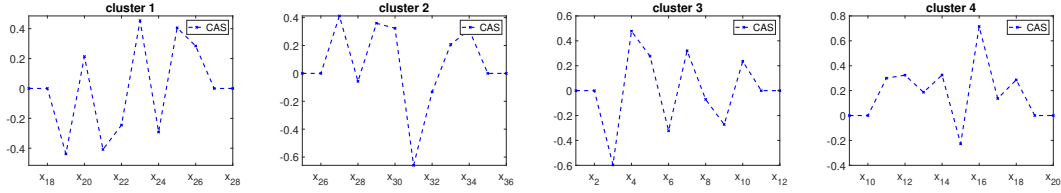
Fig. 3. The leading DR directions in four different clusters calculated by the CAS method. The dimensions that are not shown in the figure are all of zero value.

reinstate that in the case the cluster number is one, CAS reduces to the standard AS. From the table we can see that all the global dimension reduction methods produce comparable results while the performance of CAS is clearly better than those. Within CAS, the use of four clusters provides the best results. We can also see from the table that with 4 clusters, keeping 1 dimension seems to be sufficient for the local GP emulator and increasing the dimensionality does not improve the performance. Moreover, the table also shows that results of CAS are rather robust with respect to the value of α. Finally for illustration purposes, we plot the leading DR direction in each of the four clusters in Fig. 3.

Another important issue in CAS method is to determine the number of clusters J and the reduced dimensionality r_j in each cluster. In Fig. 4 we show the relation between the reduced dimensionality r_j and the value of ρ, where one can see that, while r_j depends on ρ, the dependence is not highly sensitive. In what follows we conduct the numerical experiments using Eq. (2.12) with $\rho = 0.85$ and 0.95. The number of clusters is determined by a ten-fold cross validation procedure, the results of which are shown in Table 4, from which we can see that, in both cases the NMSE is decreasing as J increases from 1 to 4 and it remains about the same at 4 and 5. Based on the the principle of Occam's razor [23] for model selection, we choose $J = 4$ here and the final NMSE is 20.4% for $\rho = 0.85$ and 21.6% for $\rho = 0.95$. Note that these are the results where all the r_j's are determined automatically, which is different from those in Table 3.

5 Conclusions

In this work, we consider the construction of GP emulators for computationally intensive models of large dimensions. In such problems, the construction of GP emulator directly in the original input space is usually not feasible due to the fact that GP models can not handle very high dimensionality. Thus, a common practice is to first reduce the dimensionality of the original model and then construct the GP emulator in the dimension-reduced parameter space. To this end, the AS method is particularly effective as it utilize the gradient information. To deal with models that do not have a simple globally low di-

20

Table 3
The NMSE results of the PDE example.

$\eta = 1$							
Method	CAS-LGP			AS-GP	SIR-GP	SAVE-GP	GP
k d	2	3	4				
1	27.4%	27.3%	**20.5%**	41.0%	37.4%	41.0%	41.0%
2	27.8%	27.9%	**20.1%**	41.0%	38.1%	41.0%	
3	28.5%	28.4%	**20.4%**	41.0%	41.0%	41.0%	
4	29.4%	29.3%	**21.6%**	41.0%	41.0%	41.0%	
$\eta = 0.5$							
Method	CAS-LGP			AS-GP	SIR-GP	SAVE-GP	GP
k d	2	3	4				
1	29.2%	23.1%	**20.5%**	41.0%	37.4%	41.0%	41.0%
2	30.0%	23.8%	**20.1%**	41.0%	38.1%	41.0%	
3	29.7%	24.4%	**20.4%**	41.0%	41.0%	41.0%	
4	30.4%	25.2%	**21.6%**	41.0%	41.0%	41.0%	

Table 4
The Cross Validation results.

CAS-LGP	CV-NMSE					NMSE
J ρ	1	2	3	**4**	5	4
0.85	36.4%	32.8%	31.3%	**30.3%**	30.2%	20.4%
0.95	36.4%	36.3%	32.4%	**30.3%**	30.3%	21.6%

mensional structure, we proposed a clustered AS method, which first identifies the "clusters" and then computes local low dimensional structure associated to them. Finally a set of local and low dimensional GP emulators are obtained for the underlying simulation model. We apply the proposed method to several examples which do not possess a global low dimensional structure, and the numerical results show that the CAS based LGP can provide more accurate results than those based on global dimension reduction.

Several extensions of the proposed method are possible. First, one can see that the CAS method does not rely on the GP emulator and can be incorporated
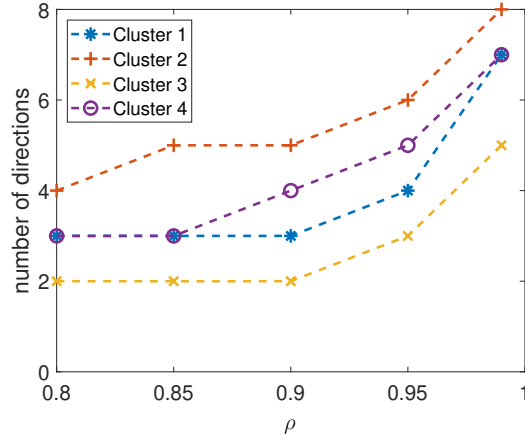
Fig. 4. The number of directions determined by the value of ρ.

with many other surrogates such as polynomial or RBF based regression models. To this end a very straightforward extension is to investigate the advantages and potential issues of combining CAS with different surrogate models. Second, the performance of the clustering method depends critically on the distance function used. In this work we proposed a distance function which combines the gradient and the location information, and it is certainly interesting to explore other possible distance functions which can further improve the performance. Finally just like the standard AS method, the proposed CAS method also relies on the gradient information. However, in many real-world applications, the gradient information may be unavailable or available with significant error or noise. To this end, we hope to extend the present CAS approach so that it can identify the low-dimensional structure, using the noisy gradient or without using the gradient information at all.

Acknowledgments

A   Proof of Theorem 3.2

proof A.1 *First for $j = 1, ..., J$ we define*

$$f_j(\mathbf{x}) = \begin{cases} f(\mathbf{x}), \ \mathbf{x} \in \Omega_j, \\ 0, \ \mathbf{x} \notin \Omega_j, \end{cases}$$

22

and it should be clear that

$$f(\mathbf{x}) = \sum_{j=1}^{J} f_j(\mathbf{x}).$$

Next we have

$$\mathbb{E}[(f - \hat{F})^2] = \sum_{j=1}^{J} w_k \mathbb{E}_{\pi_j}[(f - \hat{F})^2]$$

$$= \sum_{j=1}^{J} w_j \mathbb{E}_{\pi_j}[(f_j - \hat{F}_j)^2]$$

$$\leq \sum_{j=1}^{J} w_j (1 + \frac{1}{N_j}) \alpha'_j (\lambda_{r_j+1} + \cdots + \lambda_d),$$

where the last inequality is a direct application of Theorem (2.2). Finally letting $\alpha_j = w_j \alpha'_j$ *completes the proof.*

B Cross validation for determining J

Here we describe a k-fold cross validation (CV) procedure for determining the number of clusters J. For each $J = 1$ to $J_{\max}$ (i,e., the maximum number of clusters allowed), we perform the following procedure to calculate the average NMSE associated to J:

- First one randomly splits the training set D into k equal groups: $D_1, ..., D_k$.
- For $i = 1$ to k
  · Take D_i as a CV test set, and the rest combined as the CV training set denoted as D_{-i}.
  · Apply the CAS-LGP method to the data set D_{-i} and one obtains a local GP emulator, which is then tested on D_i, yielding a NMSE result, denoted as $NMSE_i$.
- Calculate the average NMSE: $NMSE = \frac{1}{k} \sum_{i=1}^{k} NMSE_i$.

Finally we choose J that yields the smallest $NMSE$.

C Supplemental result for the PDE example

In the following table (C.1), we provide some the NMSE results of the PDE example with $\beta = 0.01$. As one can see from the table, in this case, dimension reduction does not improve the regression accuracy at all, which provides a good example of the limitation of dimension reduction techniques in general.

Table C.1

The NMSE results of the PDE example.

$\eta = 0.5,\ \beta = 0.01$							
Method	CAS-LGP			AS-GP	SIR-GP	SAVE-GP	GP
k ╲ d	2	3	4				
1	12.57%	15.39%	15.48%	**12.44%**	15.39%	15.39%	15.39%
2	15.64%	**15.39%**	15.48%	15.42%	15.39%	15.39%	
3	15.64%	**15.39%**	15.48%	15.42%	15.39%	15.39%	
4	15.64%	**15.39%**	15.48%	15.42%	15.39%	15.39%	

References

[1] Sylvain Arlot, Alain Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[2] Ivo Babuska, Raúl Tempone, and Georgios E Zouraris. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 42(2):800–825, 2004.

[3] Julien Bect, David Ginsbourger, Ling Li, Victor Picheny, and Emmanuel Vazquez. Sequential design of computer experiments for the estimation of a probability of failure. *Statistics and Computing*, 22(3):773–793, 2012.

[4] Yoshua Bengio, Olivier Delalleau, and Nicolas L Roux. The curse of highly variable functions for local kernel machines. In *Advances in neural information processing systems*, pages 107–114, 2006.

[5] Lorenz T Biegler, Omar Ghattas, Matthias Heinkenschloss, and Bart van Bloemen Waanders. Large-scale pde-constrained optimization: an introduction. In *Large-Scale PDE-Constrained Optimization*, pages 3–13. Springer, 2003.

[6] Ilias Bilionis and Nicholas Zabaras. Multi-output local gaussian process regression: Applications to uncertainty quantification. *Journal of Computational Physics*, 231(17):5718–5746, 2012.

[7] Li Bing. *Sufficient Dimension Reduction: Methods and Applications with R.* 2018.

[8] Paul G Constantine. *Active subspaces: Emerging ideas for dimension reduction in parameter studies*. SIAM, 2015.

[9] Paul G Constantine, Eric Dow, and Qiqi Wang. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014.

[10] R. Dennis Cook and Sanford Weisberg. Comment. *Journal of the American Statistical Association*, 86(414):328–332, 1991.

[11] Josip Djolonga, Andreas Krause, and Volkan Cevher. High-dimensional gaussian process bandits. In *Advances in Neural Information Processing Systems*, pages 1025–1033, 2013.

[12] Alex Gorodetsky and Youssef Marzouk. Mercer kernels and integrated variance experimental design: connections between gaussian process regression and polynomial approximation. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):796–828, 2016.

[13] H-M Gutmann. A radial basis function method for global optimization. *Journal of global optimization*, 19(3):201–227, 2001.

[14] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. Bayesian active learning for posterior estimation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3605–3611, 2015.

[15] Leonard Kaufman and Peter J Rousseeuw. Agglomerative nesting (program agnes). *Finding Groups in Data: An Introduction to Cluster Analysis*, pages 199–252, 2008.

[16] Ker-Chau Li. Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, 86(414):316–327, 1991.

[17] Xiaoyu Liu and Serge Guillas. Dimension reduction for gaussian process emulation: An application to the influence of bathymetry on tsunami heights. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):787–812, 2017.

[18] Xiang Ma and Nicholas Zabaras. An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations. *Journal of Computational Physics*, 228(8):3084–3113, 2009.

[19] Jeremy Oakley and Anthony O'Hagan. Bayesian inference for the uncertainty distribution of computer model outputs. *Biometrika*, 89(4):769–784, 2002.

[20] Jeremy E Oakley and Anthony O'Hagan. Probabilistic sensitivity analysis of complex models: a bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(3):751–769, 2004.

[21] Anthony O'Hagan, JM Bernardo, JO Berger, AP Dawid, AFM Smith, et al. Uncertainty analysis and other inference tools for complex computer codes. 1998.

[22] Tong Qin, Kailiang Wu, and Dongbin Xiu. Data driven governing equations approximation using deep neural networks. *Journal of Computational Physics*, 395:620–635, 2019.

[23] Carl Edward Rasmussen and Zoubin Ghahramani. Occam's razor. In *Advances in neural information processing systems*, pages 294–300, 2001.

[24] Rommel G Regis and Christine A Shoemaker. Constrained global optimization of expensive black box functions using radial basis functions. *Journal of Global optimization*, 31(1):153–171, 2005.

[25] Trent Michael Russi. *Uncertainty quantification with experimental data and complex system models*. PhD thesis, UC Berkeley, 2010.

[26] Jerome Sacks, William J Welch, Toby J Mitchell, and Henry P Wynn. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989.

[27] Rohit Tripathy, Ilias Bilionis, and Marcial Gonzalez. Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation. *Journal of Computational Physics*, 321:191–223, 2016.

[28] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[29] Hongqiao Wang and Jinglai Li. Adaptive gaussian process approximation for bayesian inference with expensive likelihood functions. *Neural computation*, 30(11):3072–3094, 2018.

[30] Hongqiao Wang, Guang Lin, and Jinglai Li. Gaussian process surrogates for failure detection: A bayesian experimental design approach. *Journal of Computational Physics*, 313:247–259, 2016.

[31] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[32] Keyi Wu and Jinglai Li. A surrogate accelerated multicanonical monte carlo method for uncertainty quantification. *Journal of Computational Physics*, 321:1098–1109, 2016.

[33] Dongbin Xiu. Efficient collocational approach for parametric uncertainty analysis. *Communications in computational physics*, 2(2):293–309, 2007.

[34] Dongbin Xiu and George Em Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM journal on scientific computing*, 24(2):619–644, 2002.

[35] Xiu Yang, Xueyu Zhu, and Jing Li. When bifidelity meets cokriging: An efficient physics-informed multifidelity method. *SIAM Journal on Scientific Computing*, 42(1):A220–A249, 2020.

[36] Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[37] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.