Eric Herrison Gyamfi[1], Bledar A Konomi[1], Guang Lin[2], and Emily L. Kang[1*]

[1]Division of Statistics and Data Science, Department of Mathematical Sciences, University of Cincinnati
[2]Department of Mathematics, School of Mechanical Engineering, Purdue University

Enhancing Gaussian Process for Surrogate Modeling: A Review of Dimension Reduction Techniques for Input Variables

[1]Corresponding author: kangel@ucmail.uc.edu

0.1 Introduction

Computer models, also called simulators, are widely used to model complex processes in many fields such as engineering and climate modeling. These simulators are physics-based to reproduce the physical processes or systems of interest. However, due to their complexity, these simulators are often computationally expensive, making them costly or prohibitive to carry out. In addition, these simulators usually involve unknown parameters requiring calibration. To address these challenges, computationally efficient surrogate models have been commonly used to approximate such expensive-to-evaluate simulators in experiments and to facilitate computational savings in calibration. One prevailing surrogate modeling approach is to construct a statistical model as the surrogate, namely an emulator, based on a training dataset of the input and output pairs from the expensive-to-evaluate simulator. The emulators are then employed in various analyses related to uncertainty quantification (UQ), ranging from sensitivity analysis, parameter calibration, and uncertainty quantification for quantities of interest (QoIs).

Gaussian process (GP) regression is one mainstream statistical method for surrogate modeling (see Gramacy, 2020). GP involves a relatively small number of parameters. It provides explicit formulas for interpolation and prediction based on the training data, making it straightforward to implement. Furthermore, we obtain the predictive distribution based on the conditional distribution from GP. This provides a natural way to quantify the uncertainty, which is often crucial in computer experiments. In addition, GP can be integrated into hierarchical modeling structures, allowing for the modeling of more complicated computer experiments, including noisy data, multi-output, multi-fidelity, and multi-stage computer experiments (e.g., Kennedy and O'Hagan, 2001; Bayarri et al., 2007; Chen et al., 2015; Gu and Berger, 2016; Ma et al., 2022; Ji et al., 2023). When the size of training data n is large, fitting a GP model could be computationally challenging, since it requires $O(n^3)$ operations and $O(n^2)$ memory. However, there has been a rich literature suggesting various approaches to overcome this, such as imposing low-dimensional structure (Banerjee et al., 2008; Cressie and Johannesson, 2008), approximating with a local structure (laGP; Gramacy and Apley, 2015), or based on a directed acyclic graph (DAG) in Datta et al. (2016) and Katzfuss and Guinness (2021), and combinations of these methods (e.g. Sang and Huang, 2012; Ma and Kang, 2020). Overall, these advantages and new developments make GP regression a powerful tool for surrogate modeling, particularly in scenarios where computational efficiency, robustness to noise, and flexibility in building surrogates in complex computer experiments are essential considerations.

Although GP emulators are favorable in many UQ analyses, it has been shown that they struggle to handle high input dimensions. The accuracy of the GP emulator's approximation to the true simulator relies on the size of training data n. With an increasing number of training data, the GP emulator is anticipated to better approximate the behavior of the simulator. Various theoretical findings in the literature explore the effectiveness of the GP emulator, denoted as $\hat{f}$, in approximating the true but expensive-to-evaluate simulator f. For instance, De Marchi et al. (2011) shows that when the size of training data is n, and the training data with d-dimensional input variables are observed quasi-uniformly within the input space $\Omega \subset \mathbb{R}^d$, the approximation error for the GP emulator can be bounded by $C_d n^{-s/d} \|f\|_{\mathcal{H}}$, where s governs the smoothness of the simulator f in a Hilbert space over Ω, and C_d is a dimension-dependent constant. As d, the dimensionality of the input variables increases, the constant C_d grows, and $n^{-p/d}$ increases, unless we have the training sample size n increase simultaneously and exponentially. However, due to the computational cost to run the physics-based simulators, it is very rare to accommodate such exponential growth in the size of training data when d is large. Consequently, accuracy of the GP emulators to approximate the true simulators deteriorates as the input dimension d increases. Meanwhile, when we fit a GP model, we assume a covariance function (also called a kernel), $C : \Omega \times \Omega \to \mathbb{R}$, which in practice is often assumed to be separable and stationary, dependent on the Euclidean distance between input points. As the input dimension increases, the Euclidean distances lose their informativeness, and it becomes harder to identify covariance parameters from the training data (Bengio et al., 2005). These challenges associated with the high-dimensional input variables are termed the curse of dimensionality, and they render the task of directly employing GP regression with

high-dimensional input variables futile. This underscores the desirability of performing dimension reduction with GP regression for enhancing both accuracy and efficiency in surrogate modeling.

Several approaches have been developed to mitigate the curse of dimensionality in GP regression for surrogate modeling, focusing on identifying and exploiting specific structures of the input variables and then constructing a GP emulator based on the reduced input space. These dimension reduction techniques can be categorized into unsupervised and supervised methods. Unsupervised methods utilize only the input data to perform dimension reduction and aim to learn a lower-dimensional representation of the input variables with minimal information loss. Methods in this category range from the classic principal component analysis (PCA) to the more recent autoencoder approach (AE; Hinton et al., 2006). On the other hand, supervised dimension reduction methods learn a lower-dimensional structure of the input variables using both input and output data, with a focus on maintaining the mapping or conditional representation between the input and output variables. We discuss methods within this category including partial least squares (PLS; Wold, 1966; Geladi and Kowalski, 1986), the active subspace method (AS; Constantine et al., 2014), and the gradient-based kernel dimension reduction (gKDR; Fukumizu and Leng, 2014).

This chapter reviews the aforementioned dimension reduction approaches for input variables, embracing both unsupervised and supervised dimension reduction methods. In addition to elucidating these approaches, we present a numerical study illustrating their performance. While we consider univariate output simulators here, the framework and methods discussed can be applied to multi-output simulators as well (Ma et al., 2022; Lan et al., 2022). Our focus lies on GP surrogate modeling, showcasing various dimension reduction methods combined with GP in a numerical study. It's worth noting that these dimension reduction methods can also be employed with other surrogate modeling techniques, such as polynomial chaos expansion (PCE; Xiu and Karniadakis, 2002; Kontolati et al., 2022) and neural networks (NN; Lan et al., 2022; Lu et al., 2022).

The remainder of this chapter is organized as follows: In Section 0.2, we briefly review GP and explain the framework of GP surrogate modeling with dimension reduction. Then, in Section 0.3, we describe the use of unsupervised dimension reduction methods with the input variables, including PCA and autoencoder. In Section 0.4, we present several supervised dimension reduction methods, including PLS, AS, and gKDR. Section 0.6 contains a numerical study illustrating various dimension reduction techniques combined with GP in surrogate modeling. The chapter concludes with a discussion in Section 0.7.

0.2 Gaussian Process Regression for Surrogate Modeling

In this section, we provide a brief review of the basics of GP regression for surrogate modeling. For a more detailed and thorough description of GP regression and GP emulation, readers are referred to Rasmussen and Williams (2005) and Gramacy (2020).

Let f denote a simulator with d-dimensional input $\mathbf{x} = (x_1, \ldots, x_d)^T \in \Omega \subset \mathbb{R}^d$ and univariate output $Y \in \mathbb{R}$. In GP emulation, we assume that the output from this simulator is modeled with a Gaussian process,

$$f(\cdot)|\boldsymbol{\theta} \sim \mathcal{GP}(\mu(\cdot;\boldsymbol{\theta}), C(\cdot,\cdot;\boldsymbol{\theta})),$$

where $\mu(\cdot;\boldsymbol{\theta})$ and $C(\cdot,\cdot;\boldsymbol{\theta})$ are the mean function and covariance function known up to some parameters $\boldsymbol{\theta}$.

The mean function $\mu(\cdot;\boldsymbol{\theta})$ typically describes generic large-scale trends in the response surface, often modeled with a regression function $\mu(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta}$ in terms of q pre-specified covariates $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_q(\mathbf{x}))^T$ but unknown coefficients $\boldsymbol{\beta}$, and sometimes as a constant (i.e., $\mu(\mathbf{x}) = \mu$). The covariance function $C(\cdot,\cdot;\boldsymbol{\theta})$ gives the covariance between two outputs $Y = f(\mathbf{x})$ and $Y' = f(\mathbf{x}')$ as $\mathrm{Cov}[Y, Y'] = C(\mathbf{x}, \mathbf{x}';\boldsymbol{\theta})$. This covariance function, also called a kernel, is symmetric and positive-definite to ensure that for any $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Omega$ and $a_1, \ldots, a_n \in \mathbb{R}$, we have $\sum_{i=1}^n \sum_{j=1}^n a_i a_j C(\mathbf{x}_i, \mathbf{x}_j;\boldsymbol{\theta}) \geq 0$. Common

choices for this covariance function are typically stationary and separable, such as:

$$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \sigma^2 \prod_{i=1}^{d} c(|x_i - x_i'|; \rho_i)$$

where σ^2 is the variance parameter, $c_i(\cdot, \cdot; \rho_i)$ is a correlation function with some unknown length scale parameter ρ_i for the i-th input, $i = 1, \ldots, d$. Some common choices for the correlation function include the squared-exponential correlation function $c(x, x') = \exp[-(x - x')^2/\rho]$ and the family of Matérn correlation functions plus a nugget (Gramacy and Lee, 2012).

Suppose we have n runs from the simulator, providing training data $\{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n)\}$, where $Y_i = f(\mathbf{x}_i)$, for $i = 1, \ldots, n$. Define the $n \times q$ design matrix $\mathbf{H} = (\mathbf{h}(\mathbf{x}_1), \ldots, \mathbf{h}(\mathbf{x}_n))^T$. Assume that we are interested in the output at a set of unobserved input points $\mathbf{x}_1^*, \ldots \mathbf{x}_m^*$. Let $Y_i^* = f(\mathbf{x}_i^*)$ for $i = 1, \ldots, m$. Define $\mathbf{Y} = (Y_1, \ldots, Y_n)^T$, $\mathbf{Y}^* = (Y_1^*, \ldots, Y_m^*)^T$, and $\mathbf{H}^* = (\mathbf{h}(\mathbf{x}_1^*), \ldots, \mathbf{h}(\mathbf{x}_m^*))^T$. The assumption of GP renders that the joint distribution of $\mathbf{Y}$ and $\mathbf{Y}^*$ a multivariate Gaussian distribution:

$$\begin{pmatrix} \mathbf{Y} \\ \mathbf{Y}^* \end{pmatrix} \sim \mathcal{N}_{n+m}\left(\begin{pmatrix} \mathbf{H} \\ \mathbf{H}^* \end{pmatrix} \boldsymbol{\beta}, \begin{pmatrix} \boldsymbol{\Sigma} & \mathbf{C} \\ \mathbf{C}^T & \boldsymbol{\Sigma}^* \end{pmatrix} \right), \tag{1}$$

where $\boldsymbol{\Sigma}$, $\mathbf{C}$, and $\boldsymbol{\Sigma}^*$ are $n \times n$, $n \times m$, and $m \times m$ matrices, respectively with the associated (i, j)-th entries as $\Sigma_{ij} = C(\mathbf{x}_i, \mathbf{x}_j)$, $C_{ij} = C(\mathbf{x}_i, \mathbf{x}_j^*)$, and $\Sigma_{ij}^* = C(\mathbf{x}_i^*, \mathbf{x}_j^*)$. Assume that the parameters $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \sigma^2, \rho_1, \ldots, \rho_d\}$ are known. Then the predictive distribution of the outputs at the unobserved input points is the conditional distribution of $\mathbf{Y}*$ given the training data and $\boldsymbol{\theta}$:

$$\mathbf{Y}^* | \mathbf{Y}, \boldsymbol{\theta} \sim \mathcal{N}_m(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}), \tag{2}$$

where

$$\tilde{\boldsymbol{\mu}} = \mathbf{H}^* \boldsymbol{\beta} + \mathbf{C}^T \boldsymbol{\Sigma}^{-1} (\mathbf{Y} - \mathbf{H}\boldsymbol{\beta}), \tag{3}$$

$$\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}^* - \mathbf{C}^T \boldsymbol{\Sigma}^{-1} \mathbf{C}. \tag{4}$$

These explicit formulas in (3) and (4) can be used to obtain the prediction with the GP emulator together with associated uncertainty. It should be noted that (3) and (4) depend on the parameters $\boldsymbol{\theta}$ which are usually unknown. To estimate $\boldsymbol{\theta}$ in the GP model, we can use maximum likelihood estimation (MLE) by maximizing the likelihood $L(\boldsymbol{\theta}; \mathbf{Y}) = p(\mathbf{Y}|\boldsymbol{\theta})$ with $\mathbf{Y}|\boldsymbol{\theta} \sim \mathcal{N}_n(\mathbf{H}\boldsymbol{\beta}, \boldsymbol{\Sigma})$. Alternatively, we can impose priors on the parameters and make a fully Bayesian inference via a Markov chain Monte Carlo (MCMC) to obtain the posterior distribution of $\mathbf{Y}^*$ given data. When the size of training data n is large, various approaches including Gramacy and Apley (2015), Datta et al. (2016), and Katzfuss and Guinness (2021) can be used to facilitate efficient evaluation of the likelihood or sampling from a high-dimensional multivariate Gaussian distribution.

When the input dimension d is high, we need n to increase as more runs of the expensive-to-evaluate simulators are required to construct a GP emulator that accurately approximates the simulator, which is usually computationally costly or infeasible. One way to address this curse of dimensionality is to perform dimension reduction on the input variables (e.g., Liu and Guillas, 2017; Ma et al., 2022; Lan et al., 2022). The procedure of constructing the GP emulator consists of two steps:

In Step 1, we perform dimension reduction on the original input variables to produce the lower-dimensional input in the reduced space $\mathbb{R}^D$ where $D < d$. This is achieved by finding a mapping $\boldsymbol{\Phi} : \mathbb{R}^d \to \mathbb{R}^D$, and the resulting low-dimensional input is denoted by $\boldsymbol{\Phi}(\mathbf{x}) = (\Phi_1(\mathbf{x}), \ldots, \Phi_D(\mathbf{x}))^T \in \mathbb{R}^D$. We review unsupervised and supervise methods for constructing this mapping $\boldsymbol{\Phi}$ in Sections 0.3 and 0.4, respectively.

In Step 2, we use the low-dimensional input points corresponding to the input points in the training data, $\boldsymbol{\Phi}_i = \boldsymbol{\Phi}(\mathbf{x}_i)$, for $i = 1, \ldots, n$, and construct a GP emulator based on the pairs, $\{(\boldsymbol{\Phi}_1, Y_1), \ldots, (\boldsymbol{\Phi}_n, Y_n)\}$. When making predictions, we first need to obtain $\{\boldsymbol{\Phi}_i^* : i = 1, \ldots, m\}$ corresponding to $\{\mathbf{x}_i^* : i = 1, \ldots, m\}$ with $\boldsymbol{\Phi}_i^* = \boldsymbol{\Phi}(\mathbf{x}_i^*)$, and then obtain the predictive distribution of $\mathbf{Y}^*$ associated with $\boldsymbol{\Phi}_i^*$, $i = 1, \ldots, m$.

It's important to note that GP modeling extends beyond computer models. For instance, it's widely employed in analyzing spatial and spatiotemporal data in environmental and climate science (e.g., Gelfand

and Schliep, 2016). However, effectively addressing the complexities of such data often entails special considerations. This may involve incorporating a more flexible covariance function to accommodate nonstationary dependence structures or represent specific geometric or geographical patterns realistically. We discuss potential extensions related to these considerations in Section 0.7.

0.3 Unsupervised dimension reduction methods

In this section, we present various unsupervised dimension reduction methods, such as PCA and AE, and explore some related extensions. They are termed unsupervised as they solely focus on the input variables, disregarding the input-output relationship, to unveil the lower-dimensional structure within the input space.

0.3.1 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical method widely employed for dimension reduction, data compression, feature extraction, and data visualization. In essence, PCA seeks to discover the orthogonal projection of the original input data into a lower-dimensional space while maximizing variance. Specifically, PCA identifies linear combinations of the original input variables that are mutually orthogonal:

$$\Phi_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x}, \text{ for } i = 1, \ldots, D$$

where the d-dimensional vector $\mathbf{w}_i$ is referred to as the i-th principal component (PC). It satisfies $\mathbf{w}_i^T \mathbf{w}_j = \mathbb{1}(i = j)$ for $i, j = 1, \ldots, D$, and $\Phi_i(\mathbf{x})$ represents the i-th principal component score associated with $\mathbf{x}$.

Given the input data $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$, PCA involves the computation of the $d \times d$ sample covariance matrix $\mathbf{R}$. The principal components (PCs) are then determined via the eigendecomposition of $\mathbf{R}$:

$$\mathbf{R} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T,$$

where $\mathbf{U}$ is a $d \times d$ matrix whose columns are normalized eigenvectors $\{\mathbf{u}_i : i = 1, \ldots, d\}$ associated with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$, and $\mathbf{\Lambda}$ is a diagonal matrix $\text{diag}(\lambda_1, \ldots, \lambda_d)$. The mapping $\mathbf{\Phi} : \mathbb{R}^d \to \mathbb{R}^D$ based on PCA then projects an input point $\mathbf{x}^*$ onto the subspace spanned by the first D PCs:

$$\mathbf{\Phi}(\mathbf{x}) = \mathbf{W}^T \mathbf{x},$$

where the $d \times D$ $\mathbf{W}$ consists of the D leading eigenvectors of $\mathbf{R}$: $\mathbf{W} = (\mathbf{u}_1, \ldots, \mathbf{u}_D)$.

To determine the optimal value of D, we often calculate the cumulative proportion of explained variance and visualize how much variance is retained as we increase the number of principal components (PCs). This process helps us decide on D based on the desired level of variance explained by the first D PCs.

The computational complexity of PCA can be analyzed in three main steps. First, computing the sample covariance matrix $\mathbf{R}$ from the input data $\mathbf{X} \in \mathbb{R}^{n \times d}$ requires computing means with $O(nd)$ operations and forming $\mathbf{R}$ with $O(nd^2)$ operations, leading to an overall complexity of $O(nd^2)$. Second, the eigendecomposition of the $d \times d$ covariance matrix, given by $\mathbf{R} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, is typically computed using iterative numerical methods such as the QR algorithm, which requires $O(d^3)$ operations. Lastly, once the projection matrix $\mathbf{W} \in \mathbb{R}^{d \times D}$ is obtained from the top D eigenvectors, projecting each data point $\mathbf{x}_i$ involves computing $\mathbf{\Phi}(\mathbf{x}_i) = \mathbf{W}^T \mathbf{x}_i$, which requires $O(dD)$ operations per data point, leading to a total complexity of $O(ndD)$. Thus, the total computational complexity of PCA is $O(nd^2 + d^3 + ndD)$, where the dominant term depends on the relative sizes of n, d, and D. If d is large, the eigendecomposition step $O(d^3)$ dominates, whereas if $n \gg d$, the covariance computation $O(nd^2)$ is the bottleneck.

PCA is widely used in engineering experiments to reduce the dimensionality of geometric or parametric inputs—and sometimes outputs—enabling more efficient surrogate modeling of aerodynamic or structural systems. For example, Koziel and Pietrenko-Dabrowska (2020) applied PCA to constrain and scale the input

space of a data-driven surrogate model used to predict the responses of a rat-race coupler and transformer. Higdon et al. (2008b) used PCA for output dimensionality reduction in the context of computer model calibration. PCA has also been applied to reduce input variables for calibration and surrogate modeling in hydrological models (Kamali et al., 2007) and aerodynamic models (Tao et al., 2020).

One limitation of PCA is its assumption of linearity in the mapping $\mathbf{\Phi}$. An extension aimed at addressing this limitation is kernel PCA (k-PCA; Schölkopf et al., 1997, 1998; Hoffmann, 2007). Briefly speaking, k-PCA maps the original input data into a higher-dimensional feature space using a kernel function and then performs eigendecomposition on the resulting $n \times n$ kernel matrix. The mapped input point $\mathbf{\Phi}(\mathbf{x}^*)$ depends on the eigenvectors derived from the $n \times n$ kernel matrix and the kernel function evaluated at the input data $\{\mathbf{x}_i\}$ and $\mathbf{x}^*$. Common choices for the kernel function include polynomial kernels and Gaussian kernels. For further details and applications of k-PCA, readers are referred to Schölkopf et al. (1997), Hoffmann (2007), Ma and Zabaras (2011), Lataniotis et al. (2018), and Kontolati et al. (2023).

Additionally, PCA has important limitations in the context of designed experiments commonly used in computer experiments, such as space-filling and orthogonal array-based Latin hypercube designs. These designs intentionally construct input variables to be uncorrelated or orthogonal, which conflicts with PCA's reliance on correlated structure to identify meaningful lower-dimensional projections. In such cases, PCA recovers directions aligned with the coordinate axes, offering no insight or dimension reduction. While PCA can be useful for reducing dimensionality in observational settings with naturally correlated inputs, it is generally not suitable for structured experimental designs. Recognizing this distinction is important for the appropriate application of PCA in computer experiments.

0.3.2 Autoencoder

In this subsection, we discuss autoencoders (AE) for dimensionality reduction (Hinton and Salakhutdinov, 2006). Autoencoders have found applications in diverse tasks such as feature learning, denoising, and anomaly detection spanning domains like computer vision, natural language processing, and signal processing. An autoencoder consists of two components: an encoder and a decoder. The encoder, denoted as $\mathbf{\Phi} : \mathbb{R}^d \to \mathbb{R}^D$, serves as a dimensionality reduction mapping. It employs a neural network (NN) structure to enact a sequence of transformations that reduce the dimensionality of the input data. Conversely, the decoder, denoted by $\mathbf{\Psi} : \mathbb{R}^D \to \mathbb{R}^d$, reconstructs the low-dimensional representation back to its original dimensionality, resembling the initial data. Both the encoder and decoder architectures in AE can adopt convolutional or recurrent NN structures (Ribeiro et al., 2018; Kingma and Welling, 2019).

Training an AE entails jointly optimizing the parameters within $\mathbf{\Phi}$ and $\mathbf{\Psi}$ to minimize the reconstruction error, often quantified by the mean squared error (MSE) loss function:

$$\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{\Psi}(\mathbf{\Phi}(\mathbf{x}_i))\|^2$$

This is achieved using gradient descent algorithms. Decisions regarding the lower dimensionality D and parameters and configurations related to the NN structures, such as the number of layers and the choice of non-linear activation functions, need to be specified and can be fine-tuned through cross-validation.

It's important to acknowledge that training AE often requires a relatively large dataset, which may not always be readily available, particularly when dealing with computationally expensive simulators. However, when abundant data are accessible, AE can provide a valuable tool in handling imaging or functional input variables, as demonstrated in Banerjee et al. (2016) and Lan et al. (2022).

0.4 Supervised dimension reduction methods

Unlike unsupervised dimension reduction methods that solely utilize input data, supervised dimension reduction methods leverage both input and output data to derive the mapping $\mathbf{\Phi}$. This approach aims to preserve important relationships or structures within the input-output pair $(\mathbf{x}, Y)$.

0.4.1 Partial least squares

Partial Least Squares (PLS) regression was introduced in Wold (1966) as an alternative to ordinary least squares (OLS) regression in situations where linear regression models become ill-conditioned. Helland (1990) provides a definition of PLS regression models. Frank and Friedman (1993) discuss and compare PLS with regression using PCA and ridge regression. Widely utilized initially in chemometrics, PLS regression has been applied in diverse fields such as social sciences and medicine (e.g., de Jong, 1993; Hulland, 1999; Boulesteix and Strimmer, 2006; Abdi, 2010).

In this subsection, we present PLS regression in the context of a multivariate response $\mathbf{Y} \in \mathbb{R}^q$ initially, but we will specifically describe the optimization procedure for dimension reduction when $q = 1$, i.e., univariate response. Let's recall the $n \times d$ input data matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$. We define an $n \times q$ output data matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)^T$. When we have a univariate output, $q = 1$, and $\mathbf{Y}$ is reduced to an n-dimensional vector. Assuming both $\mathbf{X}$ and $\mathbf{Y}$ are centered, i.e., with the sample means subtracted, PLS regression assumes a basic latent decomposition of both input and output matrices:

$$\mathbf{X} = \mathbf{TP} + \mathbf{E}; \tag{5}$$

$$\mathbf{Y} = \mathbf{TQ} + \mathbf{F}. \tag{6}$$

Here, $\mathbf{T}$ is an $n \times D$ matrix whose rows are the D-dimensional scores, $\mathbf{P}$ and $\mathbf{Q}$ are matrices of latent components (called loadings), and $\mathbf{E}$ and $\mathbf{F}$ are matrices of random errors. If we know $\mathbf{T}$, under typical regression assumptions, we can obtain the least squared estimator for $\mathbf{Q}$:

$$\hat{\mathbf{Q}} = (\mathbf{T}^T\mathbf{T})^{-1}\mathbf{T}^T\mathbf{Y}.$$

In PLS regression, the $n \times D$ matrix $\mathbf{T}$ is constructed as a linear combination of the column vectors in the $n \times d$ input data matrix $\mathbf{X}$:

$$\mathbf{T} = \mathbf{XW},$$

where $\mathbf{W}$ is a $d \times D$ matrix. Define $\mathbf{B} = \mathbf{WQ}$. Then when we replace $\mathbf{T}$ with $\mathbf{XW}$ in (6), we have

$$\mathbf{Y} = \mathbf{XB} + \mathbf{F},$$

and the estimated $\mathbf{B}$ is:

$$\hat{\mathbf{B}} = \mathbf{W}\hat{\mathbf{Q}} = \mathbf{W}(\mathbf{T}^T\mathbf{T})^{-1}\mathbf{T}^T\mathbf{Y}.$$

The fitted response can then be written as:

$$\hat{\mathbf{Y}} = \mathbf{T}\hat{\mathbf{Q}} = \mathbf{T}(\mathbf{T}^T\mathbf{T})^{-1}\mathbf{T}^T\mathbf{Y},$$

and the predicted response with a new input $\mathbf{x}^*$ is

$$\hat{\mathbf{Y}}^* = (\mathbf{x}^*)^T\hat{\mathbf{B}}.$$

In PLS regression, the key lies in constructing the $n \times D$ matrix $\mathbf{T} = \mathbf{XW}$. Unlike PCA, which seeks $\mathbf{W}$ to provide the maximum explanation of the variation in $\mathbf{X}$, PLS takes into consideration $\mathbf{Y}$ and constructs $\mathbf{T}$ to maximize the correlation between $\mathbf{T}$ and $\mathbf{Y}$. Let's denote the i-th column in $\mathbf{W}$ as $\mathbf{w}_i = (w_{i1}, \ldots, w_{id})$,

for $i = 1, \ldots, D$. Let $\mathbf{t}_i$ denote the i-th column in $\mathbf{T}$. We have $\mathbf{t}_i = \mathbf{X}\mathbf{w}_i$, for $i = 1, \ldots, D$. In the univariate output setting with $q = 1$, the sample covariance between $\mathbf{t}_i$ and $\mathbf{Y}$ is:

$$\mathrm{Cov}(\mathbf{Y}, \mathbf{t}_i) = \frac{1}{n}\mathbf{t}_i^T\mathbf{Y} = \frac{1}{n}\mathbf{w}_i^T\mathbf{X}^T\mathbf{Y}.$$

The sample covariance between $\mathbf{t}_i$ and $\mathbf{t}_j$ is:

$$\mathrm{Cov}(\mathbf{t}_i, \mathbf{t}_j) = \frac{1}{n}\mathbf{t}_i^T\mathbf{t}_j = \frac{1}{n}\mathbf{w}_i^T\mathbf{X}^T\mathbf{X}\mathbf{w}_j.$$

PLS specifies $\mathbf{T}$ by finding $\mathbf{w}_1, \ldots, \mathbf{w}_D$ through solving sequential optimization problems, for $i = 1, \ldots, D$:

$$\mathbf{w}_i = \arg\max_{\mathbf{w}} (\mathbf{w}^T\mathbf{X}^T\mathbf{Y})^2, \tag{7}$$

subject to

$$\mathbf{w}^T\mathbf{w} = 1 \text{ and } \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}_j = 0; \text{ for } j = 1, \ldots, i-1.$$

In other words, PLS successively specifies a unit-length $\mathbf{w}_i$ to ensure it maximizes the covariance between $\mathbf{t}_i$ and $\mathbf{Y}$, while $\mathbf{t}_i$ is empirically uncorrelated with $\mathbf{t}_j$'s. With $\mathbf{W}$ obtained from this successive but non-iterative algorithm, the resulting dimension reduction mapping $\mathbf{\Phi} : \mathbb{R}^d \to \mathbb{R}^D$ based on PLS is given by: $\mathbf{\Phi}(\mathbf{x}) = \mathbf{W}^T\mathbf{x}$.

In their work, Bhadra et al. (2024) employ PLS for dimension reduction before fitting a GP model, thereby enhancing the predictive performance of the resulting GP model. Meanwhile, Chun and Keleş (2010) and Polson et al. (2021) delve into theoretical properties and extensions of PLS, particularly in the realms of variable selection and deep learning.

The computational complexity of PLS regression can be analyzed by breaking it into key steps. Given the input data matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ and response matrix $\mathbf{Y} \in \mathbb{R}^{n\times q}$, the core of PLS lies in constructing the latent score matrix $\mathbf{T} = \mathbf{X}\mathbf{W}$, where $\mathbf{W} \in \mathbb{R}^{d\times D}$ is determined iteratively. The main computational steps involve: (1) computing the cross-product matrix $\mathbf{X}^T\mathbf{Y}$, which requires $O(ndq)$ operations, (2) iteratively solving the optimization problem $\mathbf{w}_i = \arg\max_{\mathbf{w}}(\mathbf{w}^T\mathbf{X}^T\mathbf{Y})^2$ for each component, which involves a power iteration or singular value decomposition (SVD) with complexity $O(dqD)$ per iteration, resulting in an overall complexity of $O(dqD)$, (3) computing the score vectors $\mathbf{t}_i = \mathbf{X}\mathbf{w}_i$ for each of the D components, requiring $O(ndD)$ operations, (4) updating $\mathbf{X}$ by deflation, involving matrix multiplications with complexity $O(ndD)$, and (5) computing the regression coefficients $\hat{\mathbf{B}} = \mathbf{W}(\mathbf{T}^T\mathbf{T})^{-1}\mathbf{T}^T\mathbf{Y}$, where forming and inverting $\mathbf{T}^T\mathbf{T}$ takes $O(D^3)$ operations, and the final multiplication adds $O(ndD)$. The total computational complexity of PLS regression is thus $O(ndq + dqD + ndD + D^3)$, where the dominant term depends on the relative sizes of n, d, q, and D. If D is small, the overall complexity is approximately $O(ndq + ndD)$, but for larger D, the inversion step $O(D^3)$ becomes significant.

PLS has been often applied in surrogate modeling for computer experiments, particularly to address high-dimensional input spaces. For instance, Bouhlel et al. (2016) used PLS to reduce the number of hyper-parameters prior to applying Kriging, enabling more efficient emulation of high-fidelity computer codes. In a case study on an ammonia synthesis reactor, Straus and Skogestad (2017) demonstrated that incorporating PLS significantly improved surrogate model accuracy. Additionally, Ehre et al. (2020) proposed a PLS-based Polynomial Chaos Expansion (PLS-PCE) method for global sensitivity analysis in high-dimensional problems, highlighting the utility of PLS in enhancing computational efficiency and interpretability.

0.4.2 Active subspace method

The objective of the active subspace (AS) method (Constantine et al., 2014) is to identify a low-dimensional linear manifold within the input space that captures maximal variation in the simulator $f : \mathbb{R}^d \to \mathbb{R}$. This method has found widespread application in surrogate modeling and Bayesian inverse problems (e.g., Constantine et al., 2016; Vohra et al., 2019; Ma et al., 2022). We offer a succinct overview of it in this subsection, while readers interested in a more extensive discussion are directed to Constantine (2015).

The AS method assumes that f is square integrable on the input $\mathbf{x}$ with a prior distribution $\mathbf{x} \sim \pi$. Let $\nabla_{\mathbf{x}} f(\mathbf{x}) = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d})^T$ denote the gradient of f with respect to $\mathbf{x}$. The AS method proposes exploring the $d \times d$ matrix:

$$\mathbf{H} = \mathrm{E}_{\mathbf{x}\sim\pi}[(\nabla_{\mathbf{x}} f(\mathbf{x}))(\nabla_{\mathbf{x}} f(\mathbf{x}))^T],$$

referred to as the active subspace matrix (AS matrix). In the eigendecomposition $\mathbf{H} = \tilde{\mathbf{W}}\mathbf{\Lambda}\tilde{\mathbf{W}}^T$, $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$, and the $d \times d$ matrix $\tilde{\mathbf{W}}$ contains eigenvectors $\mathbf{w}_i$ as its i-th column, $i = 1, \ldots, d$. Constantine et al. (2014) demonstrate that the mean-squared directional derivative of f with respect to the eigenvector $\mathbf{w}_i$ equals its corresponding eigenvalue λ_i. Therefore, with the eigenvalues arranged in decreasing order, the AS method utilizes the partitions:

$$\tilde{\mathbf{W}} = [\mathbf{W}\ \mathbf{V}], \text{ and } \mathbf{\Lambda} = \mathrm{diag}(\mathbf{\Lambda}_1, \mathbf{\Lambda}_2),$$

where $\mathbf{\Lambda}_{=}(\lambda_1, \ldots, \lambda_D)^T$, and $\mathbf{\Lambda}_2 = (\lambda_{D+1}, \ldots, \lambda_d)^T$; the eigenvectors in the $d \times D$ matrix $\mathbf{W}$ correspond to the D largest eigenvalues in $\mathbf{\Lambda}_1$, while those in the $d \times (d-D)$ matrix $\mathbf{V}$ correspond to the remaining $(d-D)$ eigenvalues in $\mathbf{\Lambda}_2$. Define:

$$\boldsymbol{\phi} = \mathbf{W}^T\mathbf{x}, \text{ and } \boldsymbol{\eta} = \mathbf{V}^T\mathbf{x}.$$

Constantine et al. (2014) prove that f varies more on average along the directions given by the eigenvectors in $\mathbf{W}$ than along the directions given by those in $\mathbf{V}$. The column space of $\mathbf{W}$ is thus termed the *active subspace*, and $\boldsymbol{\phi}$ is called the corresponding *active variables*. Additionally, Constantine et al. (2014) express $\mathbf{x} = \mathbf{W}\boldsymbol{\phi} + \mathbf{V}\boldsymbol{\eta}$ and derive that:

$$\mathrm{E}[(\nabla_{\boldsymbol{\phi}} f(\mathbf{x}))^T(\nabla_{\boldsymbol{\phi}} f(\mathbf{x}))] = \sum_{i=1}^{D} \lambda_i,$$

$$\mathrm{E}[(\nabla_{\boldsymbol{\eta}} f(\mathbf{x}))^T(\nabla_{\boldsymbol{\eta}} f(\mathbf{x}))] = \sum_{i=D+1}^{d} \lambda_i.$$

In particular, when $\lambda_{D+1}, \ldots, \lambda_d$ are all zero, the gradient $\nabla_{\boldsymbol{\eta}} f(\mathbf{x})$ is zero for any $\mathbf{x}$ in the input space.

It's important to note that the AS matrix $\mathbf{H}$ is typically not directly available. In practice, we need to estimate it from data. Suppose we sample $\mathbf{x}_i \sim \pi$, for $i = 1, \ldots, n$. We then evaluate the gradient of f at $\mathbf{x}_i$: $\nabla_{\mathbf{x}} f_i \equiv \nabla_{\mathbf{x}} f(\mathbf{x}_i)$, for $i = 1, \ldots, n$. The AS matrix is estimated by:

$$\hat{\mathbf{H}} = \frac{1}{n}\sum_{i=1}^{n} (\nabla_{\mathbf{x}} f_i)(\nabla_{\mathbf{x}} f_i)^T.$$

Let $\hat{\mathbf{W}}$ denote the $d \times D$ matrix consisting of the leading D eigenvectors of $\hat{\mathbf{H}}$. Consequently, the dimension reduction mapping $\mathbf{\Phi} : \mathbb{R}^d \to \mathbb{R}^D$ based on the AS method is given by: $\mathbf{\Phi}(\mathbf{x}) = \hat{\mathbf{W}}^T\mathbf{x}$.

To estimate $\mathbf{H}$ and implement the AS method, having access to the gradient of f is essential. However, extensions have been proposed to address scenarios where this gradient is not readily available. Constantine (2015) suggests performing linear regression with training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$. Another approach involves utilizing surrogate modeling for both f and its gradient (Zahm et al., 2020; Wycoff et al., 2021). Furthermore, the AS method has been extended to accommodate simulators with multivariate output: Ma et al. (2022) leverage the likelihood function of the output to derive the AS matrix, while Musayeva and Binois (2024) explore various approaches to identify a common structure based on the gradients of each component in the output vector. Additionally, Lam et al. (2020) extend the AS method to handle computer models of varying fidelities.

The computational complexity of the AS method is primarily determined by the estimation of the AS matrix $\mathbf{H}$, its eigendecomposition, and subsequent projection steps. Given n sampled points $\mathbf{x}_i \sim \pi$, for $i = 1, \ldots, n$, we first compute the gradients $\nabla_{\mathbf{x}} f_i = \nabla_{\mathbf{x}} f(\mathbf{x}_i)$, which requires evaluating f and its derivatives at n points, leading to an initial complexity of $O(nd)$ if automatic differentiation or finite difference methods are used. The AS matrix is then estimated as $\hat{\mathbf{H}} = \frac{1}{n}\sum_{i=1}^{n}(\nabla_{\mathbf{x}} f_i)(\nabla_{\mathbf{x}} f_i)^T$, which involves $O(nd^2)$ operations. The eigendecomposition of the $d \times d$ matrix $\hat{\mathbf{H}}$ is performed next, requiring $O(d^3)$ operations. Once the leading D

eigenvectors $\hat{\mathbf{W}}$ are extracted, projecting an input $\mathbf{x}$ onto the active subspace via $\mathbf{\Phi}(\mathbf{x}) = \hat{\mathbf{W}}^T\mathbf{x}$ takes $O(dD)$ operations per data point, leading to an additional complexity of $O(ndD)$ when performed for all n data points. The total computational complexity of the AS method is thus given by $O(nd+nd^2+d^3+ndD)$, where the dominant term depends on the relationship between n, d, and D. If d is large, the eigendecomposition step $O(d^3)$ dominates, whereas if $n \gg d$, the covariance matrix computation $O(nd^2)$ becomes the bottleneck.

AS has frequently been used in combination with Gaussian Process (GP) models and output dimension reduction, as demonstrated in Guy et al. (2019) and Ma et al. (2022). Tripathy and Bilionis (2019) introduced a scalable framework called Deep Active Subspaces, which integrates deep neural networks with AS to support high-dimensional uncertainty propagation. In the context of chemical reaction systems, Vohra et al. (2020) applied AS to construct efficient surrogate models for uncertainty quantification, showcasing the method's ability to reduce computational cost while preserving model fidelity.

0.4.3 gKDR

The gradient-based kernel dimension reduction (gKDR) approach builds upon the framework of the sufficient dimension reduction (SDR) framework, which was initially proposed and refined in seminal works by Cook (1994),Chiaromonte and Cook (2002) and Cook (2009). It operates under the assumption that the variables Y and $\mathbf{x}$ are conditionally independent given the projected inputs $\mathbf{W}^T\mathbf{x}$, as expressed by the equation:

$$Y \perp \mathbf{x} \,|\, \mathbf{W}^T\mathbf{x} \tag{8}$$

Here, $\mathbf{W} \in \mathbb{R}^{d\times D}$, and $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, and $\perp$ stands for independence. The space defined by the column vectors of $\mathbf{W}$ is termed the effective dimension reduction (EDR) space. Methods aimed at determining $\mathbf{W}$ include sliced inverse regression (SIR; Li, 1991) and the minimum average variance estimation (MAVE; Xia et al., 2002). However, these techniques either rely on strong assumptions regarding the distributions of $\mathbf{x}$ or are computationally intensive. The gKDR method is proposed by Fukumizu and Leng (2014) to overcome these limitations. Notably, unlike the AS method, gKDR does not require evaluations of the gradients of the simulator $\nabla_{\mathbf{x}} f(\mathbf{x})$. We provide an overview of gKDR in this subsection, while a comprehensive exposition of the method can be found in the original paper by Fukumizu and Leng (2014).

Consider a random variable $(\mathbf{x}, Y)$ defined on the spaces $\mathcal{X} \times \mathcal{Y}$ with the probability distribution $\pi_{\mathbf{x}y}$, where $(\mathcal{X}, \mathcal{B}_{\mathcal{X}}, \mu_{\mathcal{X}})$ and $(\mathcal{Y}, \mathcal{B}_{\mathcal{Y}}, \mu_{\mathcal{Y}})$ are measure spaces. Let $k_{\mathcal{X}}$ and $k_{\mathcal{Y}}$ be positive-definite kernels on $\mathcal{X}$ and $\mathcal{Y}$, respectively, with associated reproducing kernel Hilbert spaces (RKHS) denoted as $\mathcal{H}_{\mathcal{X}}$ and $\mathcal{H}_{\mathcal{Y}}$.

The cross-covariance operator $C_{YX} : \mathcal{H}_{\mathcal{X}} \to \mathcal{H}_{\mathcal{Y}}$ is defined by:

$$\langle g, C_{YX} h\rangle_{\mathcal{H}_{\mathcal{Y}}} = \mathrm{E}[h(\mathbf{x})g(Y)],$$

for any $h \in \mathcal{H}_{\mathcal{X}}$ and $g \in \mathcal{H}_{\mathcal{Y}}$. Similarly, the covariance operator C_{XX} on $\mathcal{H}_{\mathcal{X}}$ is defined as:

$$\langle h_2, C_{XX} h_1\rangle_{\mathcal{H}_{\mathcal{X}}} = \mathrm{E}[h_1(\mathbf{x})h_2(\mathbf{x})],$$

for any $h_1, h_2 \in \mathcal{H}_{\mathcal{X}}$. Fukumizu et al. (2004) establish that when $\mathrm{E}[g(Y)|\mathbf{x}] \in \mathcal{H}_{\mathcal{X}}$ and C_{XX} is injective, we have:

$$\mathrm{E}[g(Y)|\mathbf{x}] = C_{XX}^{-1} C_{XY} g.$$

Furthermore, Fukumizu et al. (2004) introduce a conditional covariance operator $C_{YY|\mathbf{x}}$ and propose to minimize the determinant of the corresponding empirical conditional covariance matrix, calculated from data, with respect to the choice of $\mathbf{W}$. This method is termed kernel dimension reduction (KDR). However, the numerical optimization involving a nonconvex objective function in KDR is generally computationally expensive, limiting its applicability to large and high-dimensional datasets.

The gKDR method enhances KDR by integrating the principles of gradient-based dimension reduction methods (Samarov, 1993; Hristache et al., 2001). Specifically, Fukumizu and Leng (2014) demonstrate that when (8) is satisfied, subject to certain mild conditions, the following expressions hold:

$$\frac{\partial}{\partial x_i}\mathrm{E}[g(Y)|\mathbf{x}] = \langle g, C_{YX} C_{XX}^{-1} \frac{\partial k_{\mathcal{X}}(\cdot, \mathbf{x})}{\partial x_i}\rangle_{\mathcal{H}_{\mathcal{Y}}},$$

and

$$C_{YX}C_{XX}^{-1}\frac{\partial k_{\mathcal{X}}(\cdot,\mathbf{x})}{\partial x_i}=\sum_{j=1}^{D}W_{ij}\frac{\partial}{\partial \phi_j}\mathrm{E}[k_{\mathcal{Y}}(\cdot,Y)|\mathbf{W}^T\mathbf{x}=\boldsymbol{\phi}].$$

We define the $d\times d$ matrix-valued function $M(\mathbf{x})$ with the (i,j)-th element given by:

$$M_{ij}(\mathbf{x})=\langle C_{YX}C_{XX}^{-1}\frac{\partial k_{\mathcal{X}}(\cdot,\mathbf{x})}{\partial x_i},C_{YX}C_{XX}^{-1}\frac{\partial k_{\mathcal{X}}(\cdot,\mathbf{x})}{\partial x_j}\rangle_{\mathcal{H}_{\mathcal{Y}}},$$

for $i,j=1,\ldots,d$. Fukumizu and Leng (2014) demonstrate that $M_{ij}(\mathbf{x})$ is equivalent to:

$$\sum_{k=1}^{D}\sum_{l=1}^{D}W_{ik}W_{jl}\langle\frac{\partial}{\partial \phi_k}\mathrm{E}[k_{\mathcal{Y}}(\cdot,Y)|\mathbf{W}^T\mathbf{x}=\boldsymbol{\phi}],\frac{\partial}{\partial \phi_l}\mathrm{E}[k_{\mathcal{Y}}(\cdot,Y)|\mathbf{W}^T\mathbf{x}=\boldsymbol{\phi}]\rangle_{\mathcal{H}_{\mathcal{Y}}}.$$

Therefore, the eigenvectors of $M(\mathbf{x})$ encompass the EDR space defined by the column vectors of $\mathbf{W}$.

With data $(\mathbf{x}_1,Y_1),\ldots,(\mathbf{x}_n,Y_n)$, the estimator for $M(\mathbf{x})$ in Fukumizu and Leng (2014) is given by:

$$\hat{M}(\mathbf{x})=\nabla\mathbf{k}_{\mathcal{X}}(\mathbf{x})^T(\mathbf{G}_{\mathcal{X}}+n\epsilon_n\mathbf{I})^{-1}\mathbf{G}_{\mathcal{Y}}(\mathbf{G}_{\mathcal{X}}+n\epsilon_n\mathbf{I})^{-1}\nabla\mathbf{k}_{\mathcal{X}}(\mathbf{x}),$$

where

$$\nabla\mathbf{k}_{\mathcal{X}}(\mathbf{x})=\left(\frac{\partial k_{\mathcal{X}}(\mathbf{x}_1,\mathbf{x})}{\partial\mathbf{x}},\ldots,\frac{\partial k_{\mathcal{X}}(\mathbf{x}_n,\mathbf{x})}{\partial\mathbf{x}}\right)^T\in\mathbb{R}^{n\times d},$$

and $\mathbf{G}_{\mathcal{X}}$ and $\mathbf{G}_{\mathcal{Y}}$ represent the $n\times n$ Gram matrices with the (i,j)-th entry as $k_{\mathcal{X}}(\mathbf{x}_i,\mathbf{x}_j)$ and $k_{\mathcal{Y}}(Y_i,Y_j)$, respectively. Here, ϵ_n denotes the regularization coefficient (Bach and Jordan, 2002).

Fukumizu and Leng (2014) define the average of $\hat{M}(\mathbf{x}_i)$ over the data set $\{\mathbf{x}_i:i=1,\ldots,n\}$ as:

$$\tilde{M}_n\equiv\frac{1}{n}\hat{M}(\mathbf{x}_i)$$

Subsequently, the $d\times D$ matrix $\mathbf{W}$ is constructed with the unit-length eigenvectors corresponding to the D largest eigenvalues of the $d\times d$ matrix $\tilde{M}_n$. This approach is termed gKDR in Fukumizu and Leng (2014). In their work, Fukumizu and Leng (2014) also propose two additional variants:

1. The first variant involves gradually reducing the dimensionality in an iterative algorithm. Initially, $D^{(0)}=d$. In the l-th iteration, a $D^{(l-1)}\times D^{(l)}$ matrix $\mathbf{W}^{(l)}$ is obtained from gKDR, where $D^{(l)}<D^{(l-1)}$, for $l=1,2,\ldots,L$, with the condition $D^{(L)}=D$. The final $d\times D$ dimension reduction matrix $\mathbf{W}$ is formed as $\mathbf{W}=\mathbf{W}^{(1)}\mathbf{W}^{(2)}\cdots\mathbf{W}^{(L)}$. This variant is referred to as gKDR-i in Fukumizu and Leng (2014).

2. The second variant involves firstly partitioning the data into L subsets. Then, a $d\times D$ matrix $\mathbf{W}_{(l)}$ is computed from each subset, for $l=1,\ldots,L$. The final dimension reduction matrix $\mathbf{W}$ is constructed using the eigenvectors corresponding to the D largest eigenvalues of the matrix $\mathbf{P}=\frac{1}{L}\sum_{l=1}^{L}\mathbf{W}_{(l)}\mathbf{W}_{(l)}^T$. This is called gKDR-v in Fukumizu and Leng (2014).

To implement gKDR and its variants, it's essential to specify the kernel functions, tune parameters within these functions, select the regularization coefficient ϵ_n, and determine the low dimensionality D. Fukumizu and Leng (2014) suggest employing the Gaussian or Laplace kernels and utilizing cross-validation, particularly with k-nearest neighbor (kNN) regression, to set these parameters efficiently. Additionally, Liu and Guillas (2017) suggest determining D based on the predictive performance of the resulting emulator, particularly if gKDR is integrated into surrogate modeling frameworks.

The computational complexity of the gradient-based kernel dimension reduction (gKDR) method is primarily determined by the estimation of the kernel Gram matrices, computation of the empirical matrix $\tilde{M}_n$, its eigendecomposition, and the iterative or ensemble-based variants. Given a dataset $(\mathbf{x}_i,Y_i)$ for $i=1,\ldots,n$, gKDR requires computing the Gram matrices $\mathbf{G}_{\mathcal{X}}$ and $\mathbf{G}_{\mathcal{Y}}$, each of size $n\times n$, by evaluating kernel functions on all pairs of data points, leading to a complexity of $O(n^2d)$ and $O(n^2q)$, respectively, where q is the output dimension. Constructing the gradient matrix $\nabla\mathbf{k}_{\mathcal{X}}(\mathbf{x})$, which involves differentiating the kernel function with respect to input variables, requires $O(nd^2)$ operations. Solving the system $(\mathbf{G}_{\mathcal{X}}+n\epsilon_n\mathbf{I})^{-1}$ requires

computing a Cholesky decomposition or matrix inversion, which has a complexity of $O(n^3)$. The empirical estimate $\hat{M}(\mathbf{x})$ is then computed as a quadratic form involving the inverse Gram matrices and the gradient matrix, leading to an overall complexity of $O(n^3 + nd^2)$. Finally, obtaining the dimension reduction matrix $\mathbf{W}$ requires performing the eigendecomposition of the $d \times d$ matrix $\tilde{M}_n$, which has complexity $O(d^3)$. The total computational complexity of gKDR is thus given by $O(n^2 d + n^2 q + n^3 + nd^2 + d^3)$, where the dominant term depends on the relative sizes of n, d, and q. If $n \gg d$, then the inversion step $O(n^3)$ dominates, while if d is large, the eigendecomposition $O(d^3)$ becomes the bottleneck. The iterative variant gKDR-i introduces additional matrix multiplications across iterations, contributing an extra factor of L, making the complexity approximately $O(L(n^3 + d^3))$, whereas the ensemble-based gKDR-v method requires computing separate eigenvalue decompositions for each subset, increasing the complexity to $O(L(n^3/L^3 + d^3))$. The choice of kernel function and regularization parameter ϵ_n also influences computational costs, with cross-validation or parameter tuning adding an extra complexity of $O(T(n^3 + d^3))$, where T is the number of parameter evaluations.

gKDR has been shown to be effective in capturing complex, nonlinear relationships between inputs and outputs. For example, Liu and Guillas (2016) applied gKDR within a Gaussian Process emulator to assess the influence of bathymetry on tsunami heights, demonstrating its ability to handle intricate dependencies. More recently, Yang et al. (2025) showed that surrogate models incorporating gKDR outperformed those based on Active Subspaces (AS) in the construction of a surrogate for the forward model in a remote sensing instrument. These studies highlight gKDR's potential as a powerful tool for dimension reduction in challenging scientific applications.

0.5 Practical Guidelines for Dimension Reduction in Computer Experiments

Before presenting the numerical results, we offer a set of practical recommendations for handling high-dimensional input settings in computer experiments. A summary of the recommended workflow is provided in Figure 1.

When dealing with high-dimensional inputs, the first step is to clearly define the objectives of the study – whether they involve prediction, uncertainty quantification, surrogate modeling, or calibration. It is also important to understand the characteristics of the input variables, including their domains, dependence structures, and potential influence on the outputs. An initial assessment should be made to determine whether the problem is susceptible to the curse of dimensionality, which can degrade model accuracy and increase computational cost. In the context of computer experiments, where the size of the training dataset is typically limited, even a moderate number of input variables (e.g., 10–50) can pose challenges.

In such cases, applying dimension reduction techniques can be highly beneficial. Unsupervised methods such as Principal Component Analysis (PCA) or autoencoders are effective at capturing dominant structures in the input space, but because they do not incorporate output information, they are generally less suitable for surrogate modeling. Supervised approaches—such as Partial Least Squares (PLS), Active Subspaces (AS), and gradient-based Kernel Dimension Reduction (gKDR)—are better suited for this purpose, as they are designed to preserve input-output relationships and can handle complex, nonlinear dependencies.

After reducing the dimensionality, the transformed inputs can be used in surrogate modeling frameworks, such as Gaussian Process (GP) emulators. Model performance can be assessed through a train-test split and appropriate metrics, such as Root Mean Squared Prediction Error (RMSPE). Cross-validation can be employed to determine the optimal dimension of the reduced input space. This process may require iteration to refine the reduction method or its parameters. Throughout, domain knowledge should guide decisions about which inputs are most relevant and how to interpret the results. Finally, it is essential to document the dimension reduction process clearly, including the techniques used, the rationale for their selection, and their impact on model performance. This ensures that the reduced representation remains interpretable and meaningful in the context of the original scientific problem.

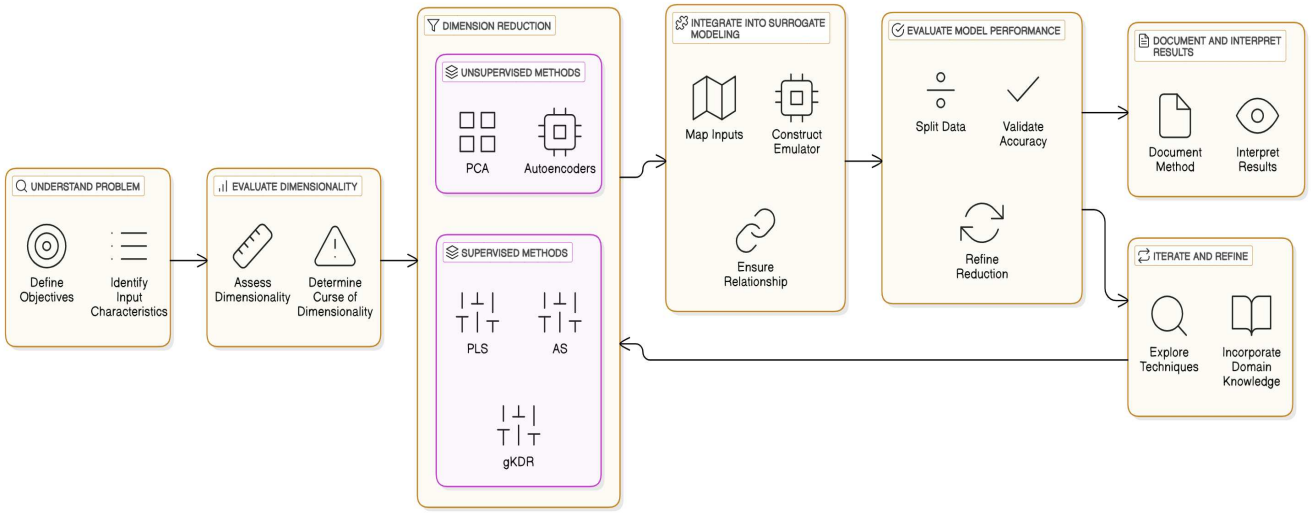


FIGURE 1
A summary of workflow when dealing with high-dimensional inputs.

0.6 A numerical study

In this section, we demonstrate the efficacy of GP surrogate modeling using different dimension reduction techniques in a numerical study. Section 0.6.1 elucidates the process of generating synthetic data. It's worth noting that we adopt a two-step approach in surrogate modeling with the synthetic data: initially reducing the dimensionality of input variables, followed by constructing a GP model between the reduced-dimension input and output. Section 0.6.2 outlines the implementation of various dimension reduction methods and evaluates their performance. The details of GP modeling implementation and the corresponding results are delineated in Section 0.6.3. The computation is executed using either R or Matlab on a 4-core HP system equipped with 12 Gigabytes of memory. The code for this experiment can be accessed at https://github.com/UCStat/DimensionR.

0.6.1 Synthetic Data

We generate synthetic data to evaluate the efficacy of dimension reduction methods, including PCA, PLS, AS, and gKDR. Subsequently, we employ a GP model utilizing the lower-dimensional input variables obtained from these dimension reduction techniques, alongside the synthetic output data. Specifically, we assume that the input vector $\mathbf{x}$ follows a multivariate standard normal distribution $\mathcal{N}_d(0, \mathbf{I})$, where each component x_i is

independently sampled from a standard normal distribution. Moreover, we model the function $f : \mathbb{R}^d \to \mathbb{R}$ as follows:

$$f(\mathbf{x}) = \gamma(\mathbf{W}^T\mathbf{x}),$$

where $\mathbf{W} \in \mathbb{R}^{d\times D}$, and $\gamma : \mathbb{R}^D \to \mathbb{R}$ represents a quadratic function defined as:

$$\gamma(\boldsymbol{\phi}) = a_0 + \mathbf{a}^T\boldsymbol{\phi} + \boldsymbol{\phi}^T\mathbf{A}\boldsymbol{\phi},$$

with a_0 being the intercept, $\mathbf{a}$ the D-dimensional linear coefficient vector, and $\mathbf{A}$ the $D \times D$ matrix. The output Y is generated based on this model below:

$$Y = f(\mathbf{x}) + \epsilon,$$

where ϵ follows a normal distribution $N(0, \sigma_\epsilon^2)$. This setup considers the output as f contaminated with Gaussian white noise having zero mean and variance σ_ϵ^2. The term ϵ can also be interpreted as a nugget term, as suggested even for deterministic simulators (Gramacy and Lee, 2012).

In this numerical investigation, we adopt a setup similar to Tripathy et al. (2016). Specifically, we set $d = 10$ and $D = 1$. The $d \times D$ matrix $\mathbf{W}$ is then condensed into a d-dimensional vector, which we define as:

$$\mathbf{W} = (-0.0091, -0.0579, -0.1877, 0.4774, 0.4559, -0.6714, -0.1264, -0.0082, 0.0724, -0.2308)^T.$$

With $D = 1$, the remaining parameters, including $\mathbf{a}$ and $\mathbf{A}$ in $\gamma(\cdot)$, are all scalars and specified as follows:

$$a_0 = -0.16113,\ \mathbf{a} = (-0.97483),\ \mathbf{A} = (-1.66526),\ \text{and } \sigma_\epsilon^2 = 0.01.$$

We initially generate N samples from $\mathbf{x} \sim \mathcal{N}_d(0, \mathbf{I})$ independently. For each sampled $\mathbf{x}$, we compute $f(\mathbf{x}) = \gamma(\mathbf{W}^T\mathbf{x})$. The corresponding synthetic output Y is obtained by adding a sampled value from $N(0, \sigma_\epsilon^2)$ to $f(\mathbf{x})$. Subsequently, we randomly partition the resulting data into a training set of size $n = 80\%N$ and a test set of size $N - n$. The former is utilized for dimension reduction of $\mathbf{x}$ and fitting a GP model, while the latter validates the predictive performance of the fitted GP models employing various dimension reduction methods, namely PCA, PLS, AS, and gKDR. We set the total sample size $N = 150$, 350, and 600, with corresponding training set sizes of $n = 120$, 280, and 480, respectively.

0.6.2 Implementation and Results for Dimension Reduction

We apply the dimension reduction methods PCA, PLS, AS, gKDR, as well as its two variants, gKDR-i and gKDR-v, to the synthetic data. In this subsection, we first outline implementation specifics for these methods before proceeding to a comparison of their results.

0.6.2.1 Implementation of PCA

In PCA, we derive the $d \times D$ dimension reduction matrix $\mathbf{W}$ by computing the D eigenvalues of the sample covariance matrix corresponding to the largest D eigenvalues. However, determining the optimal value of D in PCA is essential. In practice, this is often achieved by examining the cumulative proportion of variance explained by the leading D eigenvectors, denoted as $r(D)$, calculated as:

$$r(D) = \frac{\sum_{i=1}^{D}\lambda_i}{\sum_{j=1}^{d}\lambda_j},$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$ are the eigenvalues of the sample covariance matrix. In the left panels of Figure 2, we illustrate the proportion $r(D)$ against D for sample sizes $n = 120$, 280, and 480, respectively. If we desire $r(D)$ to be substantial, for instance, exceeding 80%, we would need to select D to be at least 8. However, compared to the original dimensionality $d = 10$, opting for $D = 8$ does not yield significant dimension reduction. This outcome aligns with expectations because we generate $\mathbf{x}_i$ independently from the distribution $\mathcal{N}_d(0, \mathbf{I})$. Consequently, the elements in $\mathbf{x}$ are independent $N(0, 1)$, indicating no inherent

structure in the input space. Subsequently, when constructing the GP model, we utilize PCA with $D = 2$ and $D = 8$ respectively: the former allows for an examination of GP performance with PCA at a comparable lower dimensionality to other methods, while the latter represents a choice of D following common practices with $r(D)$ at least 80%.

0.6.2.2 Implementation of PLS

To determine the appropriate value of D in PLS implementation, we employ leave-one-out cross-validation (LOOCV), a widely adopted method for PLS analysis (e.g., Liland et al., 2020). In our numerical study, we depict the LOOCV root-mean-squared error (RMSPE) against D for varying training data sizes of $n = 120$, 280, and 480, respectively, in the right panels of Figure 2. These plots reveal that additional components beyond the first do not lead to an improvement in RMSPE. Hence, we opt for $D = 1$ for PLS in this numerical study.

0.6.2.3 Implementation of AS

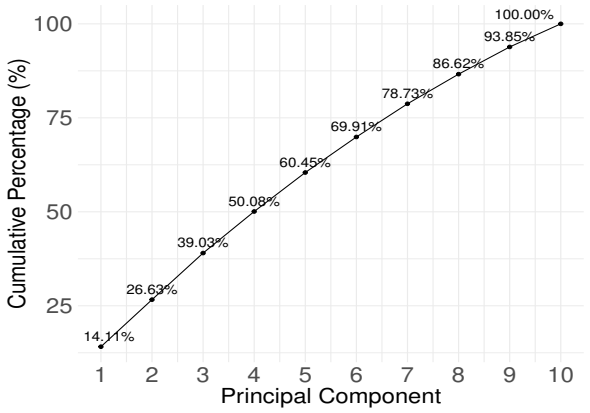
To implement the AS method, we require the gradients $\nabla_{\mathbf{x}} f$. For the data-generating model, the corresponding gradient function is given by $\nabla_{\mathbf{x}} f(\mathbf{x}) = \mathbf{W}\mathbf{a} + 2\mathbf{W}\mathbf{A}\mathbf{W}^T\mathbf{x}$. During the AS method implementation, we compute the gradients corresponding to the training data, $\nabla_{\mathbf{x}} f(\mathbf{x}_1), \ldots, \nabla_{\mathbf{x}} f(\mathbf{x}_n)$. The AS matrix is then estimated as:

$$\hat{\mathbf{H}} = \frac{1}{n}\sum_{i=1}^{n}(\nabla_{\mathbf{x}} f_i)(\nabla_{\mathbf{x}} f_i)^T.$$

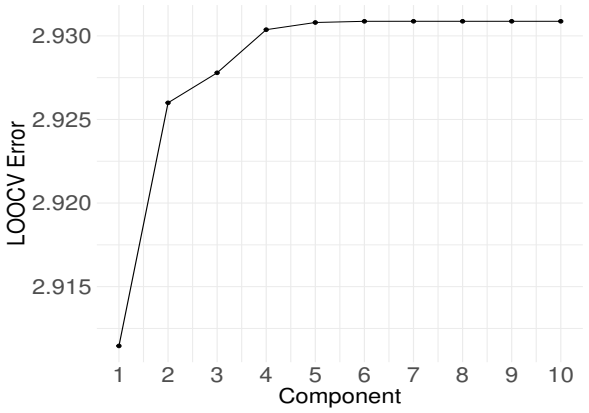
The dimension reduction projection $\mathbf{W}$ is derived as the $d \times D$ matrix comprising the unit-length eigenvectors of $\hat{\mathbf{H}}$ associated with the largest D eigenvalues from the AS matrix $\mathbf{H}$. To select the value of D, we plot $\sum_{i=1}^{D}\lambda_i / \sum_{j=1}^{D}\lambda_j$, where λ_i denotes the i-th largest eigenvalue from $\hat{\mathbf{H}}$, and choose $D = 1$ such that the ratio exceeds 80% (refer to Figure 3).

0.6.2.4 Implementation of gKDR and its variants

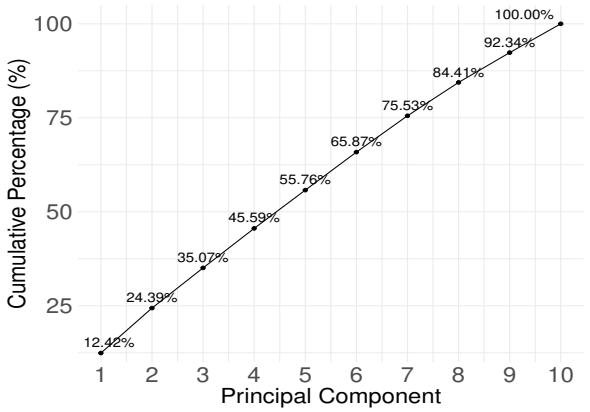
When implementing the gKDR method and its variants (i.e., gKDR-i and gKDR-v) as described in Section 0.4.3, several specifications are required, including the kernel function and its parameters, regularization parameter, and the choice of D. For the kernel functions, we adopt the Gaussian radial basis function (RBF) kernel for both $\mathcal{H}_{\mathcal{X}}$ and $\mathcal{H}_{\mathcal{Y}}$, as recommended in Fukumizu and Leng (2014). Specifically, we define $k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2/(2\sigma_x^2)\right)$ and $k_{\mathcal{Y}}(Y, Y') = \exp\left(-(Y - Y')^2/(2\sigma_Y^2)\right)$. To determine the parameters σ_x^2 and σ_Y^2, we follow the procedure outlined in Fukumizu and Leng (2014): considering candidate values (in our study, $(0.25, 0.5, 0.75, 1, 2)$) for both parameters, we perform k-nearest neighbor (kNN) regression cross-validation. For each pair of candidate values for the kernel parameters, we compute the cross-validation RMSPE with the kNN method and select the ones that yield the least cross-validation error. The regularization parameter is specified as $\epsilon_n = 10^{-5}$, following Fukumizu and Leng (2014). For the gKDR-i variant, we iteratively reduce dimensionality in $d - D$ iterations. In the l-th iteration, a $D^{(l-1)} \times D^{(l)}$ matrix $\mathbf{W}^{(l)}$ is obtained from gKDR, where $D^{(l)} = D^{(l-1)} - 1$, for $l = 1, 2, \ldots, L$, with $D^{(0)} = d$ and $D^{(L)} = D$. The final $d \times D$ dimension reduction matrix $\mathbf{W}$ is formed as $\mathbf{W} = \mathbf{W}^{(1)}\mathbf{W}^{(2)}\cdots\mathbf{W}^{(L)}$. For the gKDR-v variant, we partition the training data into two subsets and apply gKDR on each of them, obtaining resulting dimension reduction matrices $\mathbf{W}_{(1)}$ and $\mathbf{W}_{(2)}$. The final dimension reduction matrix $\mathbf{W}$ from gKDR-v is constructed using the eigenvectors corresponding to the D largest eigenvalues of the matrix $\mathbf{P} = \frac{1}{2}(\mathbf{W}_{(1)}\mathbf{W}_{(1)}^T + \mathbf{W}_{(2)}\mathbf{W}_{(2)}^T)$. To determine the value of D, we once again analyze cross-validation errors. Figure 4 illustrates that we select $D = 1$ to minimize cross-validation errors for gKDR, gKDR-i, and gKDR-v when $n = 120$, 280, and 480, respectively. We utilize the MATLAB code provided at (http://www.ism.ac.jp/ fukumizu/) to implement gKDR and its variants.



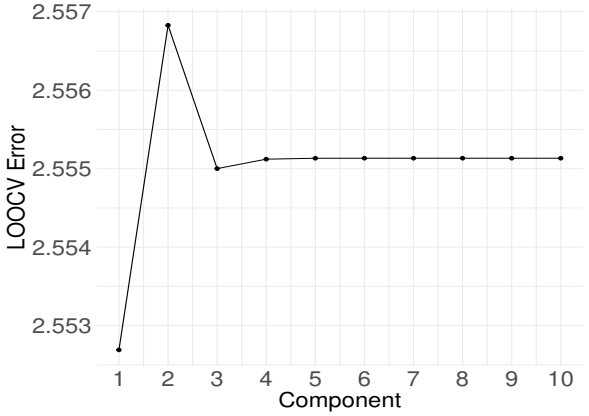
(a) PCA: $n = 120$



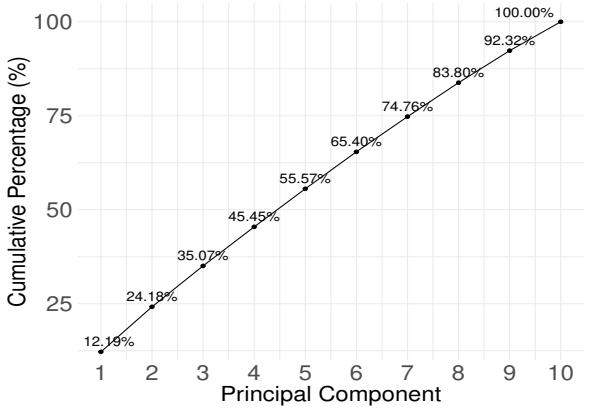
(b) PLS: $n = 120$



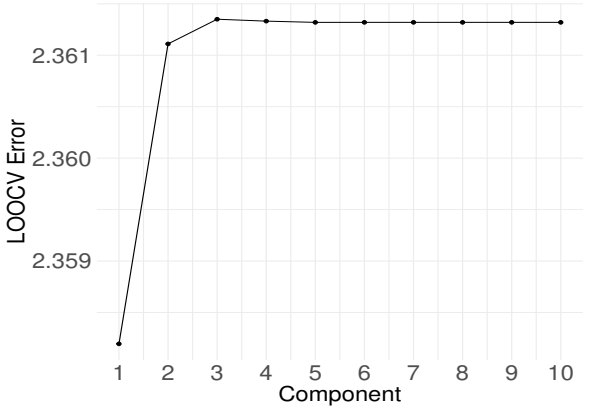
(c) PCA: $n = 280$



(d) PLS: $n = 280$



(e) PCA: $n = 480$

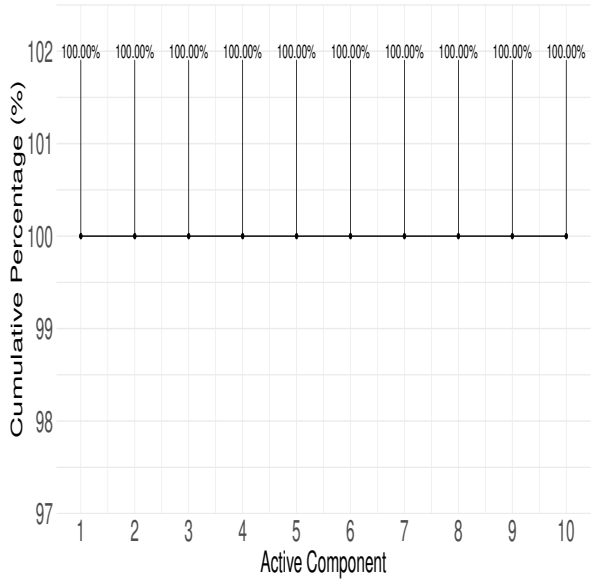


(f) PLS: $n = 480$

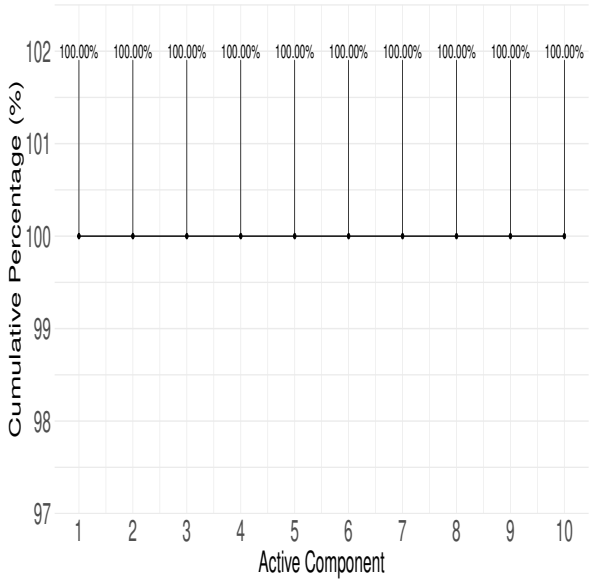
FIGURE 2
Left: plots of the cumulative proportion of variation explained by the leading D PCs in PCA, when $n = 120$, 280, and 480, respectively (from top to bottom). Right: plots of the root mean squared prediction error in LOOCV for PLS for different values of D when $n = 120$, 280, and 480, respectively (from top to bottom).

0.6.2.5 Results from Dimension Reduction

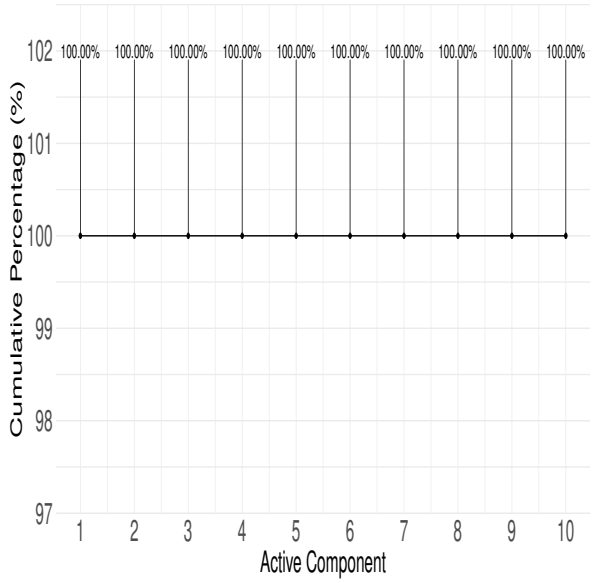
We implement the aforementioned dimension reduction methods using the synthetic data. We denote the estimated dimension reduction matrix from method A as $\mathbf{W}_A$, where A represents one of the six methods



(a) $n = 120$



(b) $n = 280$



(c) $n = 480$

FIGURE 3
Plots showing the cumulative proportion of variation explained, $\sum_{i=1}^{D} \lambda_i / \sum_{j=1}^{d} \lambda_j$, where λ_i is the largest eigenvalue of the estimated AS matrix $\hat{\mathbf{H}}$ for $n = 120$ (panel a), 280 (panel b), and 480 (panel c), respectively.

(PCA, PLS, AS, gKDR, gKDR-i, and gKDR-v), respectively. To assess the quality of $\mathbf{W}_A$ compared to the true $\mathbf{W}$, we utilize the norm proposed in Li and Wang (2007):

$$e(\mathbf{W}_A, \mathbf{W}) = ||\mathbf{W}(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T - \mathbf{W}_A(\mathbf{W}_A^T\mathbf{W}_A)^{-1}\mathbf{W}_A^T||_F,$$

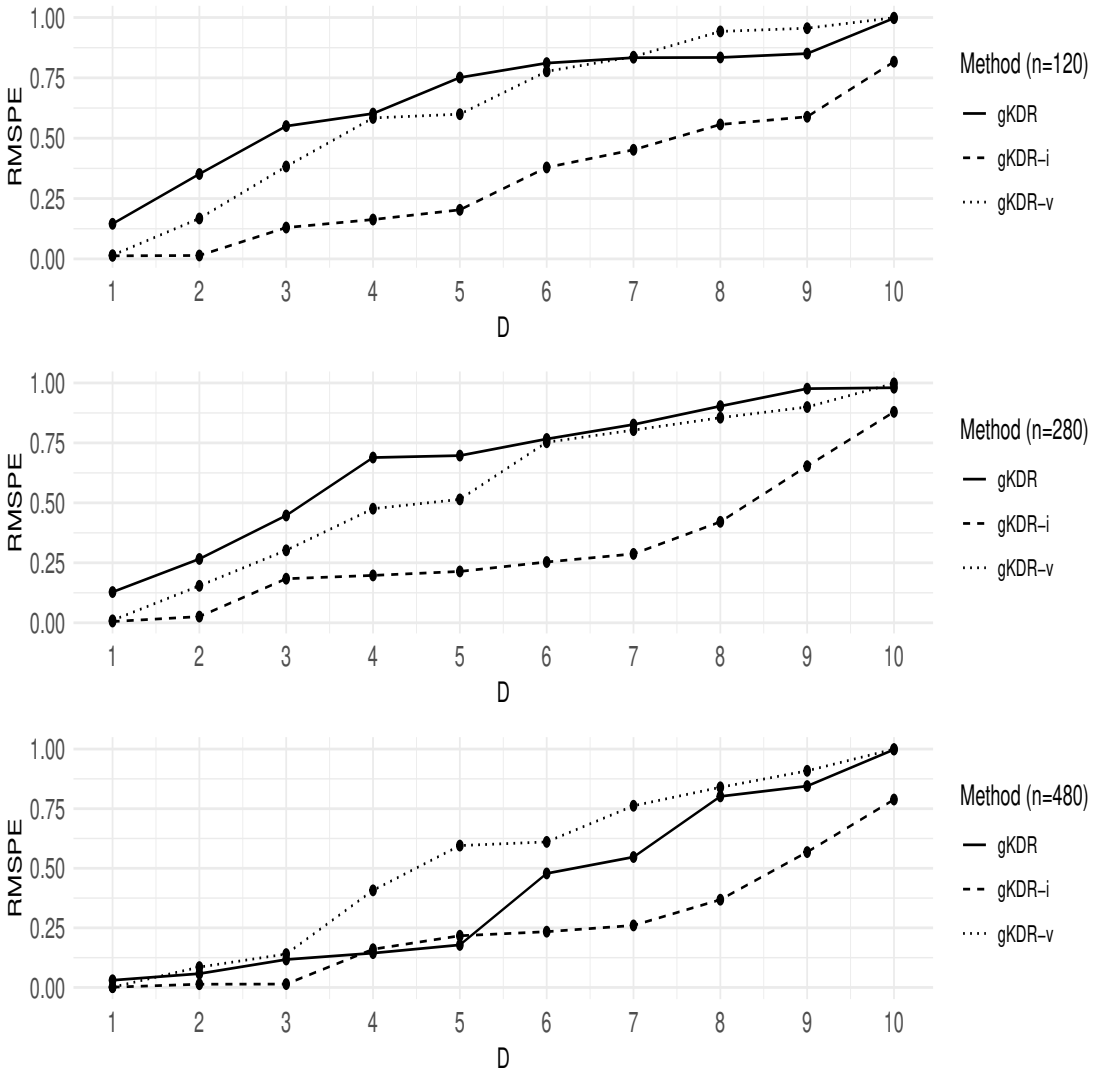


FIGURE 4
Plots of CV error associated with various values of D in gKDR (in solid line), gKDR-i (in dashed line), and gKDR-v (in dotted line), for $n = 120$ (top), 280 (middle), and 480 (bottom), respectively.

which measures the discrepancy between $\mathbf{W}_A$ and $\mathbf{W}$ using the Frobenius norm for the difference of the projection matrices for the spaces spanned by the columns in $\mathbf{W}_A$ and $\mathbf{W}$. Consequently, a smaller value of this metric $e(\mathbf{W}_A, \mathbf{W})$ indicates a better estimate of the mapping $\mathbf{\Phi}(\mathbf{x}) = \mathbf{W}^T\mathbf{x}$.

Table 0.1 presents a summary of $e(\mathbf{W}_A, \mathbf{W})$ for all different methods, where A = PCA with 8 PCs, PCA with 2 PCs, PLS, AS, gKDR, gKDR-i, and gKDR-v, with respective sample sizes of $n = 120$, 280, and 480. Since the goal in PCA is to maximize the variance in the resulting subspace, which is not directly relevant to how the output is specifically related to the input, PCA, whether with 2 or 8 PCs, fails to provide a good estimate for $\mathbf{W}$, resulting in the largest discrepancy $e(\mathbf{W}_A, \mathbf{W})$. Moreover, this discrepancy for PCA does not improve with an increase in sample size n.

Among the supervised dimension reduction methods, gKDR and its variants (gKDR-i and gKDR-v) achieve smaller values of $e(\mathbf{W}_a, \mathbf{W})$ compared to PLS. Active Subspaces (AS) delivers the best performance;

however, it should be noted that AS relies on gradient information, which was not provided to the other methods. In this case, the underlying function is a relatively simple second-order polynomial, so the availability of gradient information provides a substantial advantage to AS in model fitting. Among the gKDR-based methods, gKDR and gKDR-v perform similarly, while gKDR-i shows the smallest discrepancy from the true subspace.

The $e(\mathbf{W}_a, \mathbf{W})$ and computational complexity associated with AS varies slightly with n. For all the rest supervised dimension reduction methods, we observe a decreasing pattern of $e(\mathbf{W}_a, \mathbf{W})$ as the sample size n increases, indicating that more data are beneficial for inferring the lower structure of inputs related to the output. In terms of computation, gKDR is more time-consuming than the other methods, and its computational cost substantially increases with the sample size n. Among the three gKDR methods, gKDR-i requires the most computation time, as expected, since it involves running gKDR iteratively. It is worth noting that although many of these dimension reduction methods are originally specified through the eigendecomposition of a $d \times d$ matrix, this can be rewritten into a singular value decomposition of a corresponding matrix of size $n \times d$ and some approximation can be adopted with large n as pointed out by Fukumizu and Leng (2014). Additionally, it should be mentioned that these dimension reduction methods are not all implemented in the same software: gKDR and its variants are executed in Matlab, while the other methods are implemented in R, which may also contribute to differences in computation time.

	$n = 120$		$n = 280$		$n = 480$	
Method A	Time	$e(\mathbf{W}_A, \mathbf{W})$	Time	$e(\mathbf{W}_A, \mathbf{W})$	Time	$e(\mathbf{W}_A, \mathbf{W})$
PCA(8)	0.46	7.1077	0.53	6.4953	0.55	5.2201
PCA(2)	0.46	2.0046	0.53	2.2187	0.55	2.1758
PLS (1)	0.64	0.5172	1.26	0.21805	1.43	0.0841
AS (1)	0.70	$9.7884e-32$	0.72	$1.3679e-31$	0.80	$2.2994e-31$
gKDR (1)	1.70	0.0111	7.03	0.0081	19.10	0.0022
gKDR-i (1)	4.37	0.0069	21.69	0.0024	78.61	0.0004
gKDR-v (1)	2.07	0.0993	3.05	0.0901	4.34	0.0533

TABLE 0.1
Summary of results from dimension reduction, presenting $e(\mathbf{W}_A, \mathbf{W})$ and the associated computation time (in seconds) for method $A(a)$, representing the method A with $D = a$, including PCA with 8 PCs, PCA with 2 PCs, PLS with $D = 1$, AS with $D = 2$, and gKDR, gKDR-i, and gKDR-v with $D = 1$ for $n = 120$, 280, and 480, respectively.

0.6.3 Implementation and Results for GP Modeling

Next, we fit a GP using the lower-dimensional inputs $\boldsymbol{\phi}_i = \mathbf{W}^T \mathbf{x}_i$ and outputs Y_i, $i = 1, \ldots, n$, derived from all the dimension reduction methods reported in Section 0.6.2. We assume a zero mean GP associated with a separable squared exponential correlation function, defined as:

$$c_\rho(\boldsymbol{\phi}, \boldsymbol{\phi}') = \exp\left\{ -\sum_{i=1}^{D} \frac{(\phi_i - \phi_i')^2}{\rho_i} \right\},$$

where $\boldsymbol{\rho} = (\rho_1, \cdots, \rho_D)$ represents the set of length-scale parameters. The covariance function of the GP is then given by:

$$C(\boldsymbol{\phi}, \boldsymbol{\phi}') = \sigma^2 \left[c_\rho(\boldsymbol{\phi}, \boldsymbol{\phi}') + \tau^2 \mathbb{1}(\boldsymbol{\phi} = \boldsymbol{\phi}') \right],$$

where τ^2 denotes the nugget as a proportion of σ^2. We estimate the parameters $\boldsymbol{\rho}$, σ^2, and τ^2 using maximum likelihood estimation. Notably, with a larger value of D, we have more parameters to estimate, as $\boldsymbol{\rho}$ becomes a D-dimensional vector of length-scale parameters.

For a test input $\mathbf{x}_j^*$, $j = 1, \ldots, M - n$, we first compute its corresponding lower-dimensional input $\boldsymbol{\phi}_j^* = \mathbf{W}^T \mathbf{x}_j^*$ by utilizing the estimated dimension reduction matrix from the methods discussed in Section 0.6.2.

Subsequently, we calculate the GP prediction for its corresponding output Y_i^*, based on the conditional distribution given in (2), (3), and (4). We assess the predictive performance of the fitted GP combined with different dimension reduction methods in terms of the root mean squared prediction errors (RMSPE), defined as:

$$RMSPE_A = \sqrt{\frac{1}{M-n}\sum_{i=1}^{M-n}(Y_i - \hat{Y}_{i,A})^2},$$

where A denotes a specific dimension reduction method used to obtain the lower-dimensional inputs. Additionally, we include computation time as a metric to evaluate the cost incurred in GP fitting and prediction. The summary of these results is provided in Table 0.2.

As shown in Table 0.2, the GP emulator using AS achieves the best overall performance. This result is expected, as AS provides the most accurate subspace estimate and leverages additional gradient information, allowing the GP to effectively model the underlying second-order polynomial function. For the other methods, the RMSPE of the GP emulator decreases with increasing training sample size. Excluding AS, the GP emulators based on inputs reduced by gKDR-related methods consistently yield the best predictive performance, as indicated by the lowest RMSPE values. Among the three gKDR variants, gKDR-i achieves both the smallest discrepancy in estimating the true subspace (Table 0.1) and the lowest RMSPE (Table 0.2), albeit at a higher computational cost. The GP emulator using inputs reduced by PLS is less accurate than those based on other supervised dimension reduction methods, but it still outperforms PCA. Notably, the GP emulator built on the 8 principal components from PCA performs the worst among the methods considered and also requires more time to fit.

In terms of computation, the calculation of the likelihood requires $\mathcal{O}(n^3)$ floating point operations (flops) to invert the covariance matrix of the GP model. Therefore, we observe an increase in computational time as n increases for all the methods in Table 0.2. Moreover, we observe that fitting a GP model with 8 PCs is computationally more expensive because the parametric space of the correlation parameters is larger than other reduced input space. As the parametric space of the correlation function increases so does the evaluation of the likelihood in the optimization process to find the MLEs. Apart from PCA, the other methods seem to have comparable computational times, although the AS method with $D = 2$ costs more time than the other methods with $D = 1$. Additionally, it should be noted that the computation time of the entire analysis,i.e., dimension reduction, GP fitting, and prediction, is the summation of the computation time from both Table 0.1 and Table 0.2. From this, we can observe that AS costs similar or less computationally than other supervised methods. Overall, the empirical results from this numerical study demonstrate that the GP emulator on the gKDR input dimension reduction yields accurate estimations of $\mathbf{W}$ and achieves the best predictions compared to other methods, albeit at the expense of increased computation time.

	$n = 120$		$n = 280$		$n = 480$	
Method $A(a)$	RMSPE	Time	RMSPE	Time	RMSPE	Time
PCA(8)	5.1095	4.01	2.8689	10.55	0.7986	59.64
PCA(2)	3.9771	0.68	2.1325	4.17	2.0279	11.1
PLS (1)	2.2051	0.22	1.2437	2.99	0.7486	15.23
AS (1)	0.0038	0.24	0.0005	2.81	0.0014	15.07
gKDR (1)	0.3656	0.48	0.2394	2.58	0.115	15.41
gKDR-i (1)	0.3224	0.39	0.1177	3.20	0.0553	19.65
gKDR-v (1)	0.2757	0.37	0.2393	3.83	0.1451	16.58

TABLE 0.2
Summary of results from GP modeling and prediction, presenting the root mean squared prediction error (RMSPE) the associated computation time (in seconds) for method $A(a)$, representing the method A, with $D = a$, including PCA with 8 PCs, PCA with 2 PCs, PLS with $D = 1$, AS with $D = 2$, and gKDR, gKDR-i, and gKDR-v with $D = 1$ for $n = 120$, 280, and 480, respectively.

0.7 Discussion

In summary, we review various dimension reduction techniques that have been applied in surrogate modeling. We conducted a comparative analysis involving PCA, PLS, AS, and gKDR through a numerical study. The results from this empirical investigation underscore the efficacy of employing supervised dimension reduction approaches on input variables, enhancing the precision of resulting GP surrogates compared to unsupervised counterparts such as PCA. It would be interesting to conduct a more comprehensive comparison across these methods, utilizing simulated and real-world datasets across diverse scenarios. This could encompass varying dimensions (d/D), levels of low-dimensional structural complexity, and dataset sizes, thereby enriching our understanding of their performance under different conditions.

In this chapter, we delve into the integration of GP regression with various dimension reduction techniques. However, it's worth noting that these dimension reduction methods can be applied with other surrogate modeling approaches. For instance, Kontolati et al. (2023) utilize k-PCA for dimension reduction, employing the resultant lower-dimensional inputs with manifold-based polynomial chaos expansion (m-PCE) and the deep neural operator, known as DeepONet (Lu et al., 2021), to construct surrogates. Meanwhile, Lan et al. (2022) employ a convolutional neural network (CNN) as an emulator, implementing the autoencoder approach for dimension reduction within the context of solving the Bayesian inverse problem using MCMC.

The framework adopted in this chapter for combining input dimension reduction and surrogate modeling aligns closely with methods presented in Liu and Guillas (2017) and Ma et al. (2022). This approach involves a sequential two-step analysis, as outlined in Section 0.2: Step 1 entails dimension reduction, followed by Step 2, where the lower-dimensional input and output are utilized to construct a surrogate. It's essential to underscore that employing a stochastic model like the GP in Step 2 facilitates various UQ tasks in the analysis pipeline, such as UQ for quantities of interest (QoIs) and model calibration. However, it's noteworthy that the two steps are executed separately in this framework, with uncertainties associated with Step 1's dimension reduction not directly propagated into Step 2. Some may argue that the dimension reduction in Step 1 serves primarily as a preprocessing or data transformation step, and thus the uncertainty in this phase might be of less interest compared to that in the surrogate modeling step (Bhadra et al., 2024). Nevertheless, a unified modeling or inference framework connecting these steps could offer inferential advantages. For example, Tripathy et al. (2016) propose a probabilistic framework for the AS method, jointly inferring the matrix $\mathbf{W}$ and fitting the GP model. They demonstrate that this extends the AS method beyond reliance on simulator gradients, while still yielding satisfactory results. Furthermore, developing information-based criteria for determining D would be of significant interest. An additional extension could involve incorporating D into such a joint probabilistic framework and estimating it alongside other unknown quantities. However, these potential extensions are likely to introduce substantial complexity to the models and make computation, whether numerical optimization or Monte Carlo Bayesian inference, more challenging. Addressing these challenges remains an open problem.

It's worth noting that while the GP model with a stationary covariance function is effective for many applications, it can be extended with a more complicated covariance function to approximate more complex simulators. In fields such as environmental and climate studies, geophysical processes often exhibit nonstationary or non-Gaussian structures. To accommodate such complexities, a generalized GP model can be constructed by embedding the GP within a hierarchical structure and employing an appropriate link function (e.g., Sung et al., 2020). Moreover, for modeling nonstationary covariance structures, it's possible to introduce additional layers of models to enhance flexibility. Approaches such as the deep GP (e.g., Damianou and Lawrence, 2013; Annie Sauer and Higdon, 2023) and the Kernel Flows (KF) method (Owhadi and Yoo, 2019) propose incorporating layers of simple stationary GPs as intermediate layers to facilitate warping, thereby constructing a nonstationary GP model. Additionally, leveraging the transport map framework offers another avenue to enhance flexibility in modeling the dependence structure and joint distribution of $Y_1, \ldots, Y_n$ (Parno and Marzouk, 2018; Katzfuss and Schäfer, 2023). These advancements provide powerful tools for addressing the intricacies of complex simulators for real-world processes in various domains.

When dealing with simulators featuring large datasets n, or multivariate, and even high-dimensional out-

puts, the utilization of cross-covariance functions or the implementation of dimension reduction for output variables becomes useful in constructing multi-output GP surrogates. For instance, in scenarios involving high-dimensional outputs, techniques such as PCA or functional PCA are often employed (Higdon et al., 2008a; Ma et al., 2022). Moreover, Gu and Berger (2016) propose modeling each output at individual coordinates as conditionally independent GPs, with certain shared parameters.

Incorporating experimental design and screening into dimension reduction algorithms and surrogate modeling remains a challenging yet crucial open problem. Drawing inspiration from Bayesian optimization and sequential designs could offer promising solutions. Additionally, the exploration of more intricate computer experiments and simulation studies, encompassing multi-fidelity and multi-stage setups, is essential for effectively capturing the complexities of real-world systems. Investigating how dimension reduction methods can be integrated into UQ studies within such experiments, including sensitivity and reliability analysis, as well as optimization, represents another avenue for future research. We look forward to the emergence of work in this area.

Acknowledgements The research of Konomi and Kang was supported in part by National Science Foundation grant (NSF DMS-2053668) and the Taft Research Center at the University of Cincinnati. The research of Lin was supported in part by National Science Foundation grant (NSF DMS-2053746, DMS-2134209, ECCS-2328241, and OAC-2311848), and U.S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research program DE-SC0023161. Kang was also supported in part by Simons Foundation's Collaboration Awards (#317298 and #712755).

Bibliography

Abdi, H. (2010). Partial least squares regression and projection on latent structure regression (pls regression). *WIREs Computational Statistics*, 2(1):97–106.

Annie Sauer, R. B. G. and Higdon, D. (2023). Active learning for deep gaussian process surrogates. *Technometrics*, 65(1):4–18.

Bach, F. and Jordan, M. (2002). Learning graphical models with mercer kernels. *Advances in Neural Information Processing Systems*, 15.

Banerjee, A., Ding, W., Dy, J., Lyubchich, V., and Rhines, A. (2016). Proceedings of the 6th international workshop on climate informatics: Ci 2016. NCAR Technical Note NCAR/TN-529+PROC, National Center for Atmospheric Research (NCAR).

Banerjee, S., Gelfand, A. E., Finley, A. O., and Sang, H. (2008). Gaussian Predictive Process Models for Large Spatial Data Sets. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 70(4):825–848.

Bayarri, M. J., Berger, J. O., Cafeo, J., Garcia-Donato, G., Liu, F., Palomo, J., Parthasarathy, R. J., Paulo, R., Sacks, J., and Walsh, D. (2007). Computer model validation with functional output. *The Annals of Statistics*, 35(5):1874–1906.

Bengio, Y., Delalleau, O., and Roux, N. (2005). The curse of highly variable functions for local kernel machines. In Weiss, Y., Schölkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press.

Bhadra, A., Datta, J., Polson, N. G., Sokolov, V., and Xu, J. (2024). Merging two cultures: Deep and statistical learning. *WIREs Computational Statistics*, 16(2):e1647.

Bouhlel, M. A., Bartoli, N., Otsmane, A., and Morlier, J. (2016). An improved approach for estimating the hyperparameters of the kriging model for high-dimensional problems through the partial least squares method. *Mathematical Problems in Engineering*, 2016:1–11.

Boulesteix, A.-L. and Strimmer, K. (2006). Partial least squares: a versatile tool for the analysis of high-dimensional genomic data. *Briefings in Bioinformatics*, 8(1):32–44.

Chen, P., Zabaras, N., and Bilionis, I. (2015). Uncertainty propagation using infinite mixture of gaussian processes and variational bayesian inference. *Journal of Computational Physics*, 284:291–333.

Chiaromonte, F. and Cook, R. D. (2002). Sufficient dimension reduction and graphics in regression. *Annals of the Institute of Statistical Mathematics*, 54(4):768–795.

Chun, H. and Keleş, S. (2010). Sparse Partial Least Squares Regression for Simultaneous Dimension Reduction and Variable Selection. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 72(1):3–25.

Constantine, P. G. (2015). *Active Subspaces*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Constantine, P. G., Dow, E., and Wang, Q. (2014). Active subspace methods in theory and practice: Applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524.

Constantine, P. G., Kent, C., and Bui-Thanh, T. (2016). Accelerating markov chain monte carlo with active subspaces. *SIAM Journal on Scientific Computing*, 38(5):A2779–A2805.

Cook, R. D. (1994). On the interpretation of regression plots. *Journal of the American Statistical Association*, 89(425):177–189.

Cook, R. D. (2009). *Regression graphics: Ideas for studying regressions through graphics*. John Wiley & Sons.

Cressie, N. and Johannesson, G. (2008). Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):209–226.

Damianou, A. and Lawrence, N. D. (2013). Deep Gaussian processes. In Carvalho, C. M. and Ravikumar, P., editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 207–215, Scottsdale, Arizona, USA. PMLR.

Datta, A., Banerjee, S., Finley, A. O., and Gelfand, A. E. (2016). Hierarchical nearest-neighbor gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111(514):800–812. PMID: 29720777.

de Jong, S. (1993). Simpls: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18(3):251–263.

De Marchi, S., Buhmann, M. D., and Plonka-Hoch, G. (2011). Kernel functions and meshless methods. *Dolomites Research Notes on Approximation*, 4(DRNA Volume 4.2):1–63.

Ehre, M., Papaioannou, I., and Straub, D. (2020). Global sensitivity analysis in high dimensions with pls-pce. *Reliab. Eng. Syst. Saf.*, 198:106861.

Frank, I. E. and Friedman, J. H. (1993). A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135.

Fukumizu, K., Bach, F. R., and Jordan, M. I. (2004). Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5(Jan):73–99.

Fukumizu, K. and Leng, C. (2014). Gradient-based kernel dimension reduction for regression. *Journal of the American Statistical Association*, 109(505):359–370.

Geladi, P. and Kowalski, B. R. (1986). Partial least-squares regression: a tutorial. *Analytica Chimica Acta*, 185:1–17.

Gelfand, A. E. and Schliep, E. M. (2016). Spatial statistics and gaussian processes: A beautiful marriage. *Spatial Statistics*, 18:86–104. Spatial Statistics Avignon: Emerging Patterns.

Gramacy, R. (2020). *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press.

Gramacy, R. B. and Apley, D. W. (2015). Local gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, 24(2):561–578.

Gramacy, R. B. and Lee, H. K. H. (2012). Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22(3):713–722.

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.

Guy, H., Alexanderian, A., and Yu, M. (2019). A distributed active subspace method for scalable surrogate modeling of function valued outputs. *Journal of Scientific Computing*, 85.

Helland, I. S. (1990). Partial least squares regression and statistical models. *Scandinavian Journal of Statistics*, 17(2):97–114.

Higdon, D., Gattiker, J., Williams, B., and Rightley, M. (2008a). Computer Model Calibration Using High-Dimensional Output. *Journal of the American Statistical Association*, 103(482):570–583.

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

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hoffmann, H. (2007). Kernel pca for novelty detection. *Pattern Recognition*, 40(3):863–874.

Hristache, M., Juditsky, A., Polzehl, J., and Spokoiny, V. (2001). Structure adaptive approach for dimension reduction. *The Annals of Statistics*, 29(6):1537–1566.

Hulland, J. (1999). Use of partial least squares (pls) in strategic management research: a review of four recent studies. *Strategic Management Journal*, 20(2):195–204.

Ji, Y., Mak, S., Soeder, D., Paquet, J.-F., and Bass, S. A. (2023). A graphical multi-fidelity gaussian process model, with application to emulation of heavy-ion collisions. *Technometrics*, 0(0):1–15.

Kamali, M., Ponnambalam, K., and Soulis, E. (2007). Integration of surrogate optimization and pca for calibration of hydrologic models, a watclass case study. In *2007 IEEE International Conference on Systems, Man and Cybernetics*, pages 2733–2737.

Katzfuss, M. and Guinness, J. (2021). A General Framework for Vecchia Approximations of Gaussian Processes. *Statistical Science*, 36(1):124 – 141.

Katzfuss, M. and Schäfer, F. (2023). Scalable bayesian transport maps for high-dimensional non-gaussian spatial fields. *Journal of the American Statistical Association*, 0(0):1–15.

Kennedy, M. C. and O'Hagan, A. (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464.

Kingma, D. P. and Welling, M. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392.

Kontolati, K., Goswami, S., Shields, M. D., and Karniadakis, G. E. (2023). On the influence of over-parameterization in manifold based surrogates and deep neural operators. *Journal of Computational Physics*, 479:112008.

Kontolati, K., Loukrezis, D., Giovanis, D. G., Vandanapu, L., and Shields, M. D. (2022). A survey of unsupervised learning methods for high-dimensional uncertainty quantification in black-box-type problems. *Journal of Computational Physics*, 464:111313.

Koziel, S. and Pietrenko-Dabrowska, A. (2020). Low-cost data-driven modelling of microwave components using domain confinement and pca-based dimensionality reduction. *IET Microwaves, Antennas & Propagation*, 14(13):1643–1650.

Lam, R. R., Zahm, O., Marzouk, Y. M., and Willcox, K. E. (2020). Multifidelity dimension reduction via active subspaces. *SIAM Journal on Scientific Computing*, 42(2):A929–A956.

Lan, S., Li, S., and Shahbaba, B. (2022). Scaling up bayesian uncertainty quantification for inverse problems using deep neural networks. *SIAM/ASA Journal on Uncertainty Quantification*, 10(4):1684–1713.

Lataniotis, C., Marelli, S. P., and Sudret, B. (2018). Extending classical surrogate modeling to high dimensions through supervised dimensionality reduction : A data-driven approach. *International Journal for Uncertainty Quantification*.

Li, B. and Wang, S. (2007). On directional regression for dimension reduction. *Journal of the American Statistical Association*, 102(479):997–1008.

Li, K.-C. (1991). Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, 86(414):316–327.

Liland, K. H., Stefansson, P., and Indahl, U. G. (2020). Much faster cross-validation in plsr-modelling by avoiding redundant calculations. *Journal of Chemometrics*, 34(3):e3201. e3201 cem.3201.

Liu, X. and Guillas, S. (2016). Dimension reduction for gaussian process emulation: An application to the influence of bathymetry on tsunami heights. *SIAM/ASA J. Uncertain. Quantification*, 5:787–812.

Liu, X. and Guillas, S. (2017). 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.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. (2021). Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229.

Lu, L., Meng, X., Cai, S., Mao, Z., Goswami, S., Zhang, Z., and Karniadakis, G. E. (2022). A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778.

Ma, P. and Kang, E. L. (2020). A fused gaussian process model for very large spatial data. *Journal of Computational and Graphical Statistics*, 29(3):479–489.

Ma, P., Mondal, A., Konomi, B. A., Hobbs, J., Song, J. J., and Kang, E. L. (2022). Computer model emulation with high-dimensional functional output in large-scale observing system uncertainty experiments. *Technometrics*, 64(1):65–79.

Ma, X. and Zabaras, N. (2011). Kernel principal component analysis for stochastic input model generation. *Journal of Computational Physics*, 230(19):7311–7331.

Musayeva, K. and Binois, M. (2024). Shared active subspace for multivariate vector-valued functions.

Owhadi, H. and Yoo, G. R. (2019). Kernel flows: From learning kernels from data into the abyss. *Journal of Computational Physics*, 389:22–47.

Parno, M. D. and Marzouk, Y. M. (2018). Transport map accelerated markov chain monte carlo. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):645–682.

Polson, N., Sokolov, V., and Xu, J. (2021). Deep learning partial least squares.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press.

Ribeiro, M., Lazzaretti, A. E., and Lopes, H. S. (2018). A study of deep convolutional auto-encoders for anomaly detection in videos. *Pattern Recognition Letters*, 105:13–22. Machine Learning and Applications in Artificial Intelligence.

Samarov, A. M. (1993). Exploring regression structure using nonparametric functional estimation. *Journal of the American Statistical Association*, 88(423):836–847.

Sang, H. and Huang, J. Z. (2012). A full scale approximation of covariance functions for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(1):111–132.

Schölkopf, B., Smola, A., and Müller, K.-R. (1997). Kernel principal component analysis. In Gerstner, W., Germond, A., Hasler, M., and Nicoud, J.-D., editors, *Artificial Neural Networks — ICANN'97*, pages 583–588, Berlin, Heidelberg. Springer Berlin Heidelberg.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319.

Straus, J. and Skogestad, S. (2017). Use of latent variables to reduce the dimension of surrogate models. In Espuña, A., Graells, M., and Puigjaner, L., editors, *27th European Symposium on Computer Aided Process Engineering*, volume 40 of *Computer Aided Chemical Engineering*, pages 445–450. Elsevier.

Sung, C.-L., Hung, Y., Rittase, W., Zhu, C., and Wu, C. F. J. (2020). A generalized gaussian process model for computer experiments with binary time series. *Journal of the American Statistical Association*, 115(530):945–956.

Tao, J., SUN, G., GUO, L., and WANG, X. (2020). Application of a pca-dbn-based surrogate model to robust aerodynamic design optimization. *Chinese Journal of Aeronautics*, 33(6):1573–1588.

Tripathy, R. and Bilionis, I. (2019). Deep active subspaces – a scalable method for high-dimensional uncertainty propagation. *arXiv: Computational Physics*.

Tripathy, R., Bilionis, I., and Gonzalez, M. (2016). Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation. *Journal of Computational Physics*, 321:191–223.

Vohra, M., Alexanderian, A., Guy, H., and Mahadevan, S. (2019). Active subspace-based dimension reduction for chemical kinetics applications with epistemic uncertainty. *Combustion and Flame*, 204:152–161.

Vohra, M., Nath, P., Mahadevan, S., and Tina Lee, Y. (2020). Fast surrogate modeling using dimensionality reduction in model inputs and field output: Application to additive manufacturing. *Reliability Engineering and System Safety*, 201. Publisher Copyright: © 2020 Elsevier Ltd.

Wold, H. (1966). Estimation of principal components and related models by iterative least squares. *Multivariate Analysis*, pages 391–420.

Wycoff, N., Binois, M., and Wild, S. M. (2021). Sequential learning of active subspaces. *Journal of Computational and Graphical Statistics*, 30(4):1224–1237.

Xia, Y., Tong, H., Li, W. K., and Zhu, L.-X. (2002). An adaptive estimation of dimension reduction space. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 64(3):363–410.

Xiu, D. and Karniadakis, G. E. (2002). The wiener–askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing*, 24(2):619–644.

Yang, G., Konomi, B., Hobbs, J., , , and Kang, E. L. (2025). A data driven statistical emulation for large-scale remote sensing observing systems. Technical report, Department of Mathematical Sciences, University of Cincinnati, Cincinnati, OH. Technical Report.

Zahm, O., Constantine, P. G., Prieur, C., and Marzouk, Y. M. (2020). Gradient-based dimension reduction of multivariate vector-valued functions. *SIAM Journal on Scientific Computing*, 42(1):A534–A558.