Manifold learning for the emulation of spatial fields from computational models

W.W. Xing[a], V. Triantafyllidis[a], A.A. Shah[a,*], P.B. Nair[b], N. Zabaras[c,a]

[a] *Warwick Centre for Predictive Modelling, University of Warwick, Coventry CV4 7AL, UK*
[b] *University of Toronto Institute for Aerospace Studies, 4925 Dufferin Street, Toronto, Ontario, Canada M3H 5T6*
[c] *Department of Aerospace and Mechanical Engineering, University of Notre Dame, 365 Fitzpatrick Hall of Engineering, Notre Dame, IN 46556-5637, USA*

Abstract

Repeated evaluations of expensive computer models in applications such as design optimization and uncertainty quantification can be computationally infeasible. For partial differential equation (PDE) models, the outputs of interest are often spatial fields leading to high-dimensional output spaces. Although emulators can be used to find faithful and computationally inexpensive approximations of computer models, there are few methods for handling high-dimensional output spaces. For Gaussian process (GP) emulation, approximations of the correlation structure and/or dimensionality reduction are necessary. Linear dimensionality reduction will fail when the output space is not well approximated by a linear subspace of the ambient space in which it lies. Manifold learning can overcome the limitations of linear methods if an accurate inverse map is available. In this paper, we use kernel PCA and diffusion maps to construct GP emulators for very high-dimensional output spaces arising from PDE model simulations. For diffusion maps we develop a new inverse map approximation. Several examples are presented to demonstrate the accuracy of our approach.

Keywords: Parameterized partial differential equations, Gaussian process emulation, High dimensionality, Manifold learning, Inverse mapping, Kernel PCA, Diffusion maps

*Corresponding author. Tel: +44 (0)2476 151676. Fax: +44 (0)2476 418922. *E-mail address*: Akeel.Shah@warwick.ac.uk

1. Introduction

An emulator (or surrogate model) is an approximation of a high-fidelity computational model (simulator) that is employed in cases where use of the simulator is computationally impractical or simply infeasible [1, 2]. Applications include uncertainty quantification, design optimization and inverse parameter estimation, which demand repeated simulations in an input parameter space [3, 4]. Statistical (data-driven) emulators are based on supervised machine learning methods, notably polynomial response surface models, Gaussian process models and artificial neural networks, which are applied to input-output data generated by the simulator at judiciously selected design points [2].

In this paper, we are concerned with the emulation of outputs in *very high-dimensional spaces*, motivated by the problem of emulating spatial fields (e.g., velocity, temperature, electric) from *parameterized partial differential equation (PDE) models*. The dimensionality of the output space in this case is equal to the number of points in a spatial grid at which the field variable value is recorded. This number can be very high, especially for problems requiring a fine resolution of the spatial characteristics, e.g., multiple spatial scales or moving boundaries. Emulating such outputs poses unique challenges in terms of the computational cost.

In Gaussian process (GP) emulation, the output of a simulator is modelled as a GP indexed by the input parameters [5, 6, 7, 8, 9]. The GP emulation (GPE) framework is not naturally extended to multiple outputs. A naïve approach is to treat the output index (representing the field value at a particular location in the spatial domain) as an additional input parameter [10]. This approach is infeasible for more than a few spatial locations, even for parsimoniously selected training points [11]. Conti and O'Hagan [12] developed a method in which a multi-dimensional GP prior is placed over the outputs and separability of the covariance structure is assumed; that is, the between-output covariance and between-index correlations are factored, and it is assumed that a single set of

correlation lengths governs all points in the spatial domain. This is essentially the linear model of coregionalization with a so-called intrinsic formulation [13]. Although the resulting method is computationally practical, separability is a severe assumption to make. In many problems of practical interest, e.g., when a phase change takes place, this assumption is invalid. Recent extensions of this idea can be found in [14, 15, 16].

An alternative approach based on dimensionality reduction of the output space was developed by Higdon *et al.* [17], who used principal component analysis (PCA) combined with separate GPE of the coefficients in the PCA basis. Since the coefficients are uncorrelated, they can be treated as independent, with distinct sets of correlation lengths. A similar approach based on a wavelet decomposition was proposed by Bayarri *et al.* [18]. As we point out in [19], PCA will fail when the output space does not lie close to a linear subspace of the original space, e.g., if abrupt changes take place with variations in one or more input parameters. Other linear methods such as independent component analysis and multidimensional scaling suffer from the same issues.

Reduced-order models can also be employed as emulators for PDEs [20, 21, 22, 23]. The approach in this case is typically based on projecting the original PDE system onto a reduced-dimensional subspace obtained by forward runs of the simulator (snapshots), e.g., *via* proper orthogonal decomposition (POD). This procedure is combined with a numerical method (typically Galerkin finite element) to approximate the undetermined coefficients, e.g., in a POD basis. Such approaches are attractive since they maintain a direct link to the underlying physical principles and provide rigorous estimates of the error [23, 24, 25]. Dealing with nonlinearities, on the other hand, is not straightforward. Moreover, in methods based on weak formulations of the PDEs, affine approximations for the bilinear forms and functionals (in relation to the dependence on parameters) are key to computational efficiency [26].

Motivated by the work of Higdon *et al.*, in [19] we employ nonlinear dimensionality reduction (manifold learning) based on the Isomap [27] to perform GPE of the coefficients in a reduced-dimensional output space. Manifold learn-

ing refers to a class of methods that provide low-dimensional representations of data residing in a high-dimensional ambient space [28, 29]. The fundamental assumption is that the data lie on, or close to a manifold of low intrinsic dimensionality. The development of these methods was motivated by the failure of linear methods such as PCA to handle even relatively simple surfaces such as a swiss roll [28]. They can be categorized in a number of ways, e.g., as kernel, embedding, spectral or graphical methods, although these categories are not mutually exclusive [30, 31, 32]. Choosing amongst the methods for a given problem is not straightforward; performance on toy data sets can be misleading and each method requires tuning of free parameters to best approximate the given data set. In the present case, there is, moreover, a strict requirement for the existence of an inverse map from the reduced-dimensional space to the physical ambient space. This excludes the majority of methods, for which no such map, or even approximate map exists.

In this paper, we implement two manifold learning techniques (kernel PCA [33] and diffusion maps [34]) for GPE in high-dimensional spaces, each with their own challenges in terms of constructing a valid basis and finding an inverse map approximation. While a number of inverse map approximations exist for kernel PCA (kPCA), approximations for diffusion maps are limited to low-dimensional embeddings [35]. We outline a new approximation that is computationally efficient and stable. Its accuracy on a standard data set is demonstrated before it is used in the main algorithm developed in this paper: the emulation of spatial-field outputs (in high-dimensional spaces) from parameterized PDE models.

In the next section the problem is formulated and in Section 3 we outline the two manifold learning methods, defining the reduced-dimensional spaces and the quantities that are used later in the GP emulations and inverse map approximations. In Section 4, GPE is outlined and the targets for emulation are made clear. The overall strategy for emulating outputs in high-dimensional spaces is explained in Section 5 and the inverse map approximations are described in Section 6. Numerical examples are presented and discussed in Section 7. Brief concluding remarks are given in Section 8.

2. Statement of the problem

Consider a parameterized nonlinear, system of steady-state PDEs of arbitrary order for dependent variables (scalar fields) $u_i(\boldsymbol{x},\boldsymbol{\xi})$, $i = 1,\ldots,J$, where $\boldsymbol{\xi} \in \mathbb{R}^l$ is a vector of parameters and $\boldsymbol{x}$ is the spatial variable. To give a concrete example, the u_i could refer to velocity components (say $i = 1,2,3$) and pressure ($i = 4$) in a fluid flow model. The PDEs are permitted to be fully nonlinear and parameterized in an arbitrary fashion (including the initial and boundary conditions). It is assumed that the PDE model is well-posed (solutions exist and are unique) for the range of values of $\boldsymbol{\xi}$ considered.

The quantity or quantities of interest can include any or all of the u_i, or functions derived from the u_i. For the purposes of exposition, consider a single quantity of interest, denoted simply as $u(\boldsymbol{x};\boldsymbol{\xi})$. The simulator provides values of $u(\boldsymbol{x};\boldsymbol{\xi})$ at specified (fixed) locations, $\boldsymbol{x}^{(i)}$, $i = 1,\ldots,d$, on a spatial grid. For different inputs $\boldsymbol{\xi}^{(j)} \in \mathbb{R}^l$, $j = 1,\ldots,m$, the outputs of the simulator can be represented as vectors: $\boldsymbol{y}^{(j)} = (u(\boldsymbol{x}^{(1)};\boldsymbol{\xi}^{(j)}),\ldots,u(\boldsymbol{x}^{(d)};\boldsymbol{\xi}^{(j)}))^T \in \mathbb{R}^d$. This process can be repeated for other spatial fields of interest to derive multiple vectorized outputs in $\mathbb{R}^d$. An example of the simultaneous emulation of multiple field outputs is given in Section 7. It is assumed for now that a single output $\boldsymbol{y}$ (derived from a single scalar field $u(\boldsymbol{x};\boldsymbol{\xi})$) is the target for emulation.

The simulator can be considered as a mapping $\boldsymbol{\eta} : \mathcal{X} \to \mathcal{M}$ (assumed to be injective), where $\mathcal{M} \subset \mathbb{R}^d$ is the permissible output space and $\mathcal{X} \subset \mathbb{R}^l$ is the permissible input space. That is, $\boldsymbol{\eta}(\boldsymbol{\xi}) = \boldsymbol{y} = (u(\boldsymbol{x}^{(1)};\boldsymbol{\xi}),\ldots,u(\boldsymbol{x}^{(d)};\boldsymbol{\xi}))^T$ for an arbitrary input $\boldsymbol{\xi}$. The goal of statistical emulation is to approximate the mapping $\boldsymbol{\eta}$ given *training points* $\boldsymbol{y}^{(j)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(j)}) \in \mathcal{M}$, $j = 1,\ldots,m$. The corresponding inputs $\boldsymbol{\xi}^{(j)} \in \mathcal{X}$ are referred to as *design inputs* or *design points*.

To infer outputs of the simulator at new inputs, Conti and O'Hagan [12] took the approach of placing a d-dimensional GP prior over $\boldsymbol{\eta}$, indexed by $\boldsymbol{\xi}$. Effectively, the same assumption was made by Higdon *et al.* [17] but in that case the outputs were a linear combination of PCA basis vectors with coefficients treated as independent univariate GPs indexed by $\boldsymbol{\xi}$. In this paper, a

similar approach is adopted but rather than using PCA coefficients, we place GP priors over coefficients of a reduced-dimensional approximation of points in $\mathcal{M}$, obtained by manifold learning methods. We assume that $\mathcal{M}$ is a smooth manifold in $\mathbb{R}^d$. The high dimension d of the output space and the inability of linear dimensionality reduction methods such as PCA to capture complex response surfaces is the motivation for our approach.

3. Manifold learning methods

3.1. Kernel principal component analysis

kPCA [33] maps high-dimensional data in a space $\mathcal{M}$ to a higher-dimensional feature space $\mathscr{F}$ *via* a mapping $\boldsymbol{\phi} : \mathcal{M} \to \mathscr{F}$, in which linear PCA is performed. In our case, the data consists of the training data $\boldsymbol{y}^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)}) \in \mathcal{M} \subset \mathbb{R}^d$, $i = 1, \ldots, m$, i.e., simulator outputs at the design points $\boldsymbol{\xi}^{(i)} \in \mathcal{X} \subset \mathbb{R}^l$. The eigen-problem for the sample covariance matrix in $\mathscr{F}$ is:

$$\mathbf{C}_{\mathcal{F}}\boldsymbol{w} = \left(\frac{1}{m} \sum_{i=1}^{m} \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)}) \left(\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)}) \right)^T \right) \boldsymbol{w} = \lambda \boldsymbol{w}, \tag{1}$$

in which $\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)}) = \boldsymbol{\phi}(\boldsymbol{y}^{(i)}) - \overline{\boldsymbol{\phi}}$ is the i-th centred data point in feature space, where $\overline{\boldsymbol{\phi}} = (1/m) \sum_{j=1}^{m} \boldsymbol{\phi}(\boldsymbol{y}^{(j)})$. The mapping $\boldsymbol{\phi}(\cdot)$ is implicitly defined *via* a *kernel function* $\Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \boldsymbol{\phi}(\boldsymbol{y}^{(i)})^T \boldsymbol{\phi}(\boldsymbol{y}^{(j)})$, which generates a kernel matrix $\mathbf{K} = [K_{ij}]$ with entries $K_{ij} = \Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$. A centred kernel function $\widetilde{\Bbbk}(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)})^T \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)})$ and a centred kernel matrix $\widetilde{\mathbf{K}} = [\widetilde{K}_{ij}]$ with entries $\widetilde{K}_{ij} = \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)})^T \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)})$ are similarly defined. Note that $\widetilde{\mathbf{K}} = \mathbf{HKH}$, where $\mathbf{H} = \mathbf{I} - (1/m)\mathbf{1}\mathbf{1}^T$ is the centering matrix, in which $\mathbf{I}$ is the identity matrix and $\mathbf{1} = (1/m)(1, \ldots, 1)^T \in \mathbb{R}^m$. One of the most widely used kernel functions is the Gaussian kernel $\Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \exp\left(-||\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(j)}||^2/s^2\right)$, where s is a scale factor.

Equation (1) shows that the eigenvectors $\boldsymbol{w}$ are linear combinations of $\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)})$, i.e., $\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)})$. Using this expression in Eq. (1) and premultiplying by $\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)})^T$ (noting that $\widetilde{\mathbf{K}}$ is positive semidefinite), yields the eigenvalue problem $\widetilde{\mathbf{K}}\boldsymbol{\alpha} = m\lambda\boldsymbol{\alpha}$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^T$. Once computed, the orthonormal

$\boldsymbol{\alpha}_i$ are rescaled by $\boldsymbol{\alpha}_i \mapsto \boldsymbol{\alpha}_i/\sqrt{\lambda_i} = \widetilde{\boldsymbol{\alpha}}_i$. This defines orthonormal eigenvectors $\widetilde{\boldsymbol{w}}_i = \sum_{j=1}^m \widetilde{\alpha}_{ji}\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)})$, $i = 1, \ldots, m$, where $\widetilde{\alpha}_{ji} = \alpha_{ji}/\sqrt{\lambda_i}$ and α_{ji} denote the j-th components of $\widetilde{\boldsymbol{\alpha}}_i$ and $\boldsymbol{\alpha}_i$, respectively. Strictly speaking, there are $\min(\dim \mathscr{F}, m)$ basis vectors $\widetilde{\boldsymbol{w}}_i$, but we assume for the purposes of illustration that $\dim \mathscr{F} > m$, without loss of generality. A mapped training point $\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)})$ can be expressed in the basis $\{\widetilde{\boldsymbol{w}}_i\}_{i=1}^m \subset \mathcal{F}$ as $\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)}) = \sum_{i=1}^m z_i(\boldsymbol{y}^{(j)})\widetilde{\boldsymbol{w}}_i$, where the i-th coefficient is calculated as follows:

$$\begin{aligned} z_i(\boldsymbol{y}^{(j)}) = \widetilde{\boldsymbol{w}}_i^T\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)}) &= \sum_{l=1}^m \widetilde{\alpha}_{li}\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(l)})^T\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)}) \\ &= \sum_{l=1}^m \widetilde{\alpha}_{li}\widetilde{K}_{lj} = \widetilde{\boldsymbol{\alpha}}_i^T\widetilde{\boldsymbol{k}}_j = \widetilde{\boldsymbol{\alpha}}_i^T\mathbf{H}(\boldsymbol{k}_j - \mathbf{K1}), \end{aligned} \tag{2}$$

for $i = 1, \ldots, m$, where $\boldsymbol{k}_j = (K_{1j}, \ldots, K_{mj})^T$ and $\widetilde{\boldsymbol{k}}_j = (\widetilde{K}_{1j}, \ldots, \widetilde{K}_{mj})^T$. We can therefore define $\boldsymbol{z}(\boldsymbol{y}^{(j)}) = (z_1(\boldsymbol{y}^{(j)}), \ldots, z_m(\boldsymbol{y}^{(j)}))^T$, where the $z_i(\boldsymbol{y}^{(j)})$, $i = 1, \ldots, m$, are given by Eq. (2).

The main properties of PCA carry over to kPCA. With $\lambda_i < \lambda_{i-1}$, $i = 2, \ldots, m$, the variance in the data along $\widetilde{\boldsymbol{w}}_i$ (equal to λ_i) decreases as i increases and the coefficients in an expansion of a mapped training point in the basis $\{\widetilde{\boldsymbol{w}}_i\}_{i=1}^m$ are uncorrelated. The goal is to find an r-dimensional approximation of the points $\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)})$, where ideally $r \ll m$. The reconstruction error [36] of the projection $\widetilde{\boldsymbol{\phi}}_r(\boldsymbol{y}^{(j)}) = \sum_{i=1}^r z_i(\boldsymbol{y}^{(j)})\widetilde{\boldsymbol{w}}_i$ of $\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)})$ onto the subspace $\mathcal{F}_r = \operatorname{span}(\widetilde{\boldsymbol{w}}_1, \ldots, \widetilde{\boldsymbol{w}}_r)$ is given by $||\widetilde{\boldsymbol{\phi}}_r(\boldsymbol{y}^{(j)}) - \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)})||^2 = \sum_{i=r+1}^m \lambda_i^2$, where $||\cdot||$ is the standard Euclidean norm for $\dim \mathscr{F} < \infty$ or the $L^2(\mathcal{M})$ norm of (equivalence classes of) square integrable functions on $\mathcal{M}$ for $\dim \mathscr{F} = \infty$. The value of r is typically chosen according to a variance criterion (or modal energy) [36]: Select r such that $\sum_{i=1}^r \lambda_i / \sum_{i=1}^m \lambda_i > \varrho$ for some threshold ϱ.

We can now define a mapping $\widetilde{\boldsymbol{\phi}}_r : \mathcal{M} \to \mathcal{F}_r$ as the orthogonal projection of $\widetilde{\boldsymbol{\phi}}(\cdot)$ onto $\{\widetilde{\boldsymbol{w}}_i\}_{i=1}^r$:

$$\widetilde{\boldsymbol{\phi}}_r(\boldsymbol{y}^{(j)}) = \sum_{i=1}^r z_i(\boldsymbol{y}^{(j)})\widetilde{\boldsymbol{w}}_i. \tag{3}$$

We use the notation $\boldsymbol{z}_r(\boldsymbol{y}^{(j)}) = (z_1(\boldsymbol{y}^{(j)}), \ldots, z_r(\boldsymbol{y}^{(j)}))^T$, which, from Eq. (2), is given by $\boldsymbol{z}_r(\boldsymbol{y}^{(j)}) = [\widetilde{\boldsymbol{\alpha}}_1 \ldots, \widetilde{\boldsymbol{\alpha}}_r]^T\mathbf{H}(\boldsymbol{k}_j - \mathbf{K1})$. Algorithm 1 summarizes kPCA

for data $\{\boldsymbol{y}^{(i)}\}_{i=1}^m$.

Algorithm 1: kPCA

1: Form a kernel matrix $\mathbf{K}$ using a kernel function $\Bbbk(\cdot,\cdot)$:

$$\text{Centred kernel matrix: } \widetilde{\mathbf{K}} \leftarrow \mathbf{HKH}.$$

2: Solve eigenvalue problem: $\widetilde{\mathbf{K}}\boldsymbol{\alpha} = m\lambda\boldsymbol{\alpha} \rightarrow (\boldsymbol{\alpha}_i, \lambda_i)$, $i = 1, \ldots, m$,

$$\widetilde{\boldsymbol{\alpha}}_i \leftarrow \boldsymbol{\alpha}_i \sqrt{\lambda_i}.$$

3: Select $r < m$ according to $\sum_{i=1}^r \lambda_i / \sum_{i=1}^m \lambda_i > \varrho$. Then compute:

$$z_i(\boldsymbol{y}^{(j)}) \leftarrow \widetilde{\boldsymbol{\alpha}}_i^T \mathbf{H}(\boldsymbol{k}_j - \mathbf{K}\mathbf{1}) \quad i, j = 1, \ldots, m,$$

$$\boldsymbol{z}_r(\boldsymbol{y}^{(j)}) \leftarrow (z_1(\boldsymbol{y}^{(j)}), \ldots, z_r(\boldsymbol{y}^{(j)}))^T \quad j = 1, \ldots, m.$$

Remark 1. We assume that the training data captures the structure of $\mathcal{M}$ sufficiently well to (implicitly) define a representative basis, $\widetilde{\boldsymbol{w}}_i$, $i = 1, \ldots, m$, for the image $\widetilde{\boldsymbol{\phi}}[\mathcal{M}] \subset \mathcal{F}$ of the *entire* space $\mathcal{M}$ under $\widetilde{\boldsymbol{\phi}}$. Equation (3) then yields a reduced-dimensional approximation $\widetilde{\boldsymbol{\phi}}_r(\boldsymbol{y}) = \sum_{i=1}^r z_i(\boldsymbol{y})\widetilde{\boldsymbol{w}}_i$ for an arbitrary $\boldsymbol{y} \in \mathcal{M}$. Equivalently, by the injectivity of $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$, and assuming that the feature map is injective, Eq. (3) defines a map $(\widetilde{\boldsymbol{\phi}}_r \circ \boldsymbol{\eta})(\cdot) = \widetilde{\boldsymbol{\phi}}_r(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \rightarrow \mathcal{F}_r$, i.e., directly from the *entire* permissible input space $\mathcal{X}$ to $\mathcal{F}_r$. The basis vectors are, however, unknown without an explicit form for $\boldsymbol{\phi}$. For an arbitrary input $\boldsymbol{\xi} \in \mathcal{X}$, the coefficients $z_i(\boldsymbol{y})$ define computable maps $\mathbb{z}_i(\cdot) = z_i(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \rightarrow \mathbb{R}$ and $\boldsymbol{z}_r(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \rightarrow \mathbb{R}^r$. Thus:

$$\begin{aligned} \widetilde{\boldsymbol{\phi}}_r(\boldsymbol{\eta}(\boldsymbol{\xi})) &= \sum_{i=1}^r \mathbb{z}_i(\boldsymbol{\xi})\widetilde{\boldsymbol{w}}_i, \\ \boldsymbol{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi})) &= (\mathbb{z}_1(\boldsymbol{\xi}), \ldots, \mathbb{z}_r(\boldsymbol{\xi}))^T. \end{aligned} \tag{4}$$

The original problem of approximating $\boldsymbol{\eta} : \mathcal{X} \rightarrow \mathcal{M}$ given the training points $\{\boldsymbol{y}^{(j)}\}_{j=1}^m$ is replaced by the problem of approximating $\boldsymbol{z}_r(\boldsymbol{\eta}(\cdot))$.

A multivariate GP prior indexed by $\boldsymbol{\xi}$ is placed over $\boldsymbol{z}_r(\boldsymbol{\eta}(\cdot))$. Algorithm 1 applied to the original training set $\{\boldsymbol{y}^{(i)}\}_{i=1}^m$ yields the new training points for emulation: $\boldsymbol{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})) = \boldsymbol{z}_r(\boldsymbol{y}^{(j)}) = [\widetilde{\boldsymbol{\alpha}}_1 \ldots, \widetilde{\boldsymbol{\alpha}}_r]^T \mathbf{H}(\boldsymbol{k}_j - \mathbf{K}\mathbf{1})$, $j = 1, \ldots, m$.

3.2. Diffusion maps

In diffusion maps, the training data $\boldsymbol{y}^{(i)} \in \mathcal{M} \subset \mathbb{R}^d$, $i = 1, \ldots, m$ is mapped to a subset of $\mathbb{R}^m$ called the *diffusion space* from which a reduced-dimensional approximation is subsequently obtained [34, 37]. The mapping embeds the data points in diffusion space by preserving a *diffusion distance* defined between the points in physical space. The data points $\boldsymbol{y}^{(i)}$ are identified with nodes on a graph and a Markov chain is constructed by specifying a measure of 'connectivity' (or a 'kernel') between the nodes. Consider a weighted undirected graph $\mathcal{G}$ with vertex set $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(m)}\}$ representing the training points. Edge weights are defined by a symmetric and positive definite kernel $\mathsf{k}(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$ between the data points, e.g., the Gaussian kernel $\mathsf{k}(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \exp(-||\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(j)}||^2/s^2)$. We assume $\mathcal{G}$ is connected (otherwise the maps can be constructed separately on each connected component).

A diffusion process [38] on $\mathcal{G}$ is constructed by normalizing the connectivity (adjacency) matrix $\mathbf{K} = [\mathsf{K}_{ij}]$, where $\mathsf{K}_{ij} = \mathsf{k}(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$. The degree matrix is defined as $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_m)$, where $d_i = \sum_j \mathsf{K}_{ij}$, and an $m \times m$ *diffusion matrix* is defined by $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$. $\mathbf{P} = [P_{ij}]$ is a Markov matrix; the entry P_{ij} is considered to be a transition probability $p(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$ from node $\boldsymbol{y}^{(i)}$ to $\boldsymbol{y}^{(j)}$ in a random walk on $\mathcal{G}$. The corresponding t step transition probability $p_t(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$ (from $\boldsymbol{y}^{(i)}$ to $\boldsymbol{y}^{(j)}$ in $t \in \mathbb{N} = 1, 2, \ldots$ steps) is given by the (i, j)-th entry of $\mathbf{P}^t = \mathbf{P} \times \cdots \times \mathbf{P}$.

Since $\mathcal{G}$ is connected, $\mathbf{P}$ is ergodic and, therefore, possesses a unique stationary distribution $\boldsymbol{\pi}$ with entries $\pi_i = d_i / \sum_j d_j$ [34]. The symmetric matrix $\mathbf{P'} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{1/2}$ possesses the same eigenvalues γ'_i as $\mathbf{P}$. A spectral decomposition yields $\mathbf{P'} = \mathbf{S}\boldsymbol{\Gamma'}\mathbf{S}^T$, where the columns of $\mathbf{S}$ are the orthonormal eigenvectors $\boldsymbol{s}_i$, $i = 1, \ldots, m$, of $\mathbf{P'}$ and $\boldsymbol{\Gamma'} = \mathrm{diag}(\gamma'_1, \ldots, \gamma'_m)$. The eigenvalues are arranged such that $1 = \gamma'_1 > \cdots > \gamma'_m$ and the eigenvector $\boldsymbol{s}_1$ has entries $\sqrt{\pi_i}$ [39]. $\mathbf{P}$ has the spectral decomposition $\mathbf{P} = \mathbf{Q}\boldsymbol{\Gamma'}\mathbf{Q}^{-1}$, where $\mathbf{Q} = \mathbf{D}^{-1/2}\mathbf{S}$. The right and left eigenvectors of $\mathbf{P}$ are $\boldsymbol{r}_i = \mathbf{D}^{-1/2}\boldsymbol{s}_i$ and $\boldsymbol{l}_i = \mathbf{D}^{1/2}\boldsymbol{s}_i$, respectively. Therefore $\boldsymbol{l}_1 = \boldsymbol{\pi}\sqrt{\sum_j d_j}$ and $\boldsymbol{r}_1 = \mathbf{1}^T/\sqrt{\sum_j d_j}$. The right and left eigenvectors are bi-orthogonal, i.e., $\boldsymbol{l}_i^T \boldsymbol{r}_i = \delta_{ij}$, where δ_{ij} is the Kronecker delta.

By the orthogonality of $\mathbf{S}$, $\mathbf{P}^t = \mathbf{Q}\mathbf{\Gamma}^t\mathbf{Q}^{-1}$, or $\mathbf{P}^t = \sum_{i=1}^{m}(\gamma_i')^t \boldsymbol{r}_i \boldsymbol{l}_i^T$. The j-th row vector of $\mathbf{P}^t$, denoted $\boldsymbol{p}_j^t$, is:

$$\boldsymbol{p}_j^t = (p_t(\boldsymbol{y}^{(j)}, \boldsymbol{y}^{(1)}), \ldots, p_t(\boldsymbol{y}^{(j)}, \boldsymbol{y}^{(m)}))^T = \sum_{i=1}^{m}(\gamma_i')^t r_{ji} \boldsymbol{l}_i, \tag{5}$$

where r_{ji} is the j-th coordinate of $\boldsymbol{r}_i$. $\boldsymbol{p}_j^t$ can be considered as a probability mass function, where the i-th entry, $i = 1, \ldots, m$, is the probability of being at node $\boldsymbol{y}^{(i)}$ after t steps of a random walk that started at node $\boldsymbol{y}^{(j)}$.

A diffusion distance D_t (in physical space) is then defined as follows [34]:

$$D_t(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \left((\boldsymbol{p}_i^t - \boldsymbol{p}_j^t)^T \mathbf{D}^{-1} (\boldsymbol{p}_i^t - \boldsymbol{p}_j^t)\right)^{1/2}. \tag{6}$$

We can now define a family of diffusion maps $\boldsymbol{\psi}^t : \mathcal{M} \to \mathcal{D}^{(t)} \subset \mathbb{R}^m$ between the training points $\boldsymbol{y}^{(j)}$ and diffusion spaces $\mathcal{D}^{(t)}$ as follows [34, 37]:

$$\boldsymbol{\psi}^t(\boldsymbol{y}^{(j)}) = \left((\gamma_1')^t r_{j1}, \ldots, (\gamma_m')^t r_{jm}\right)^T. \tag{7}$$

The maps are indexed by the free parameter t. The coefficients of a mapped point $\boldsymbol{y}^{(j)}$ are the coefficients of $\boldsymbol{p}_j^t$ in the non-orthogonal basis $\{\boldsymbol{l}_i\}_{i=1}^m$. Diffusion maps embed the data points in $\mathcal{D}^{(t)}$ in the following sense [34, 37, 40]:

$$||\boldsymbol{\psi}^t(\boldsymbol{y}^{(i)}) - \boldsymbol{\psi}^t(\boldsymbol{y}^{(j)})|| = D_t(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}), \tag{8}$$

where $||\cdot||$ denotes the standard Euclidean norm. Equation (8) follows from the bi-orthogonality of the left and right eigenvectors. From Eq. (7) and the decay in the eigenvalues, we can define mappings $\boldsymbol{\psi}_r^t(\boldsymbol{y}^{(j)}) : \mathcal{M} \to \mathcal{D}_r^{(t)} \subset \mathbb{R}^r$ as follows:

$$\boldsymbol{\psi}_r^t(\boldsymbol{y}^{(j)}) = ((\gamma_1')^t r_{j1}, \ldots, (\gamma_r')^t r_{jr})^T, \tag{9}$$

which give approximations of the training data $\{\boldsymbol{y}^{(j)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})\}_{j=1}^m$ in $\mathbb{R}^r$, where ideally $r \ll m$.

In practice, the value of r is usually selected according to a criterion on the eigenvalues, e.g., as the largest index j such that $|(\gamma_j')^t| > \upsilon|(\gamma_2')^t|$ holds for a pre-selected precision υ [34]. The diffusion distance, and therefore the diffusion map, depends on t. As t increases, the diffusion distances between

Algorithm 2: Diffusion maps

1: Form a kernel matrix $\mathbf{K}$ using a kernel function $\mathsf{k}(\cdot,\cdot)$.

$$\text{Degree of node } i\text{: } d_i \leftarrow \textstyle\sum_j \mathsf{K}_{ij} \quad i = 1,\dots,m.$$
$$\text{Degree matrix: } \mathbf{D} \leftarrow \operatorname{diag}(d_1,\dots,d_m).$$
$$\mathbf{P}' \leftarrow \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{1/2}.$$

2: Eigenvalue problem: $\mathbf{P}'\boldsymbol{s} = \gamma\boldsymbol{s} \rightarrow (\boldsymbol{s}_i, \gamma_i')$, $i = 1,\dots,m$.

$$\boldsymbol{r}_i \leftarrow \mathbf{D}^{-1/2}\boldsymbol{s}_i \quad \text{and} \quad \boldsymbol{l}_i \leftarrow \mathbf{D}^{1/2}\boldsymbol{s}_i.$$

3: Select r as the largest index j such that $|(\gamma_j')^t| > \upsilon|(\gamma_2')^t|$ for a precision υ:

$$\boldsymbol{\psi}_r^t(\boldsymbol{y}^{(j)}) \leftarrow ((\gamma_1')^t r_{j1},\dots,(\gamma_r')^t r_{jr})^T \quad j = 1,\dots,m.$$

points decrease since each row of $\mathbf{P}^t$ approaches the stationary distribution (see Eq. (5)). Algorithm 2 summarizes diffusion maps for data $\{\boldsymbol{y}^{(i)}\}_{i=1}^m$.

In order to develop an inverse map approximation, we generalize diffusion maps to all points in $\mathcal{M}$ by taking the limit $m \to \infty$. In this limit, the random walk on the discrete graph using a Gaussian kernel converges to a discrete-time walk on the continuous state space $\mathcal{M}$ [34, 37, 40, 41].Full details of the following are provided in Appendix A. Here we define the key quantities needed to generalize diffusion maps for the analysis that follows in Section 6.2 on the inverse mapping. Let μ be a probability measure on $\mathcal{M}$ defining the density of points. In the limit $m \to \infty$, a one-step transition kernel for the Markov chain on $\mathcal{M}$ can be defined by $\mathbb{p}(\boldsymbol{y}', \boldsymbol{y}) = \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}')/\mathbb{d}(\boldsymbol{y}')$, from an arbitrary $\boldsymbol{y}' \in \mathcal{M}$ to an arbitrary $\boldsymbol{y} \in \mathcal{M}$, where $\mathbb{d}(\boldsymbol{y}') = \int_{\mathcal{M}} \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}')d\mu(\boldsymbol{y})$. A corresponding forward transfer operator is defined by $\mathcal{L}\varphi(\boldsymbol{y}) = \int_{\mathcal{M}} \mathbb{p}(\boldsymbol{y}', \boldsymbol{y})\varphi(\boldsymbol{y}')d\mu(\boldsymbol{y}')$ for $\varphi(\boldsymbol{y}) \in L^2(\mathcal{M}, \mu)$. This operator is the continuous analogue of multiplication of $\mathbf{P}$ from the left. The t-step operator $\mathcal{L}^t\varphi = \mathcal{L} \circ \mathcal{L} \circ \cdots \circ \mathcal{L}\varphi$ has a corresponding t-step transition kernel $\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}')$. We can similarly define a backward transfer operator $\mathcal{R}\varphi(\boldsymbol{y}) = \int_{\mathcal{M}} \mathbb{p}(\boldsymbol{y}, \boldsymbol{y}')\varphi(\boldsymbol{y}')d\mu(\boldsymbol{y}')$, which is the analogue of multiplication of $\mathbf{P}$ from the right.

The kernel $\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}')$ admits the decomposition $\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}') = \sum_{i=1}^{\infty} \gamma_i^t r_i(\boldsymbol{y}) l_i(\boldsymbol{y}')$,

where γ_i, $r_i(\boldsymbol{y})$ and $l_i(\boldsymbol{y})$ are the (common) eigenvalues and eigenfunctions of $\mathcal{L}$ and $\mathcal{R}$, respectively. They are, respectively, the continuous-space equivalents of γ_i', $\boldsymbol{r}_i$ and $\boldsymbol{l}_i$. Moreover $1 = \gamma_1 > \gamma_2 > \cdots$.The key to the inverse map we develop in Section 6.2 is the link between the eigenvalues/eigenvectors of $\mathbf{P}$ and the eigenvalues/eigenfunctions of $\mathcal{L}$ and $\mathcal{R}$.

For a fixed $\boldsymbol{y} \in \mathcal{M}$, $\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}')$ is the continuous version (a probability density in $\boldsymbol{y}' \in \mathcal{M}$) of the probability mass function defined by Eq. (5); in the latter case, $\boldsymbol{y} = \boldsymbol{y}^{(j)}$ and $\boldsymbol{y}' \in \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(m)}\}$, i.e., the finite set of states accessible from $\boldsymbol{y}^{(j)}$.As explained in Appendix A, the j-th components of $\boldsymbol{r}_i$ and $\boldsymbol{l}_i$ are, respectively, approximations of $r_i(\boldsymbol{y}^{(j)})$ and $l_i(\boldsymbol{y}^{(j)})$ based on the training data. The diffusion distances between any two points $\boldsymbol{y}, \boldsymbol{y}' \in \mathcal{M}$ are given by $D_t = ||\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}') - \mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}')||_{1/\mathbb{d}}$, where $||\varphi||_{1/\mathbb{d}}^2 = \int_{\boldsymbol{y}' \in \mathcal{M}} |\varphi(\boldsymbol{y}')|^2 / \mathbb{d}(\boldsymbol{y}') d\mu(\boldsymbol{y}')$ for functions $\{\varphi : \ ||\varphi||_{1/\mathbb{d}} < \infty\}$. In turn, diffusion maps $\boldsymbol{\psi}^t : \mathcal{M} \to \mathcal{D}^{(t)} \subset \ell^2$ are defined on the whole space $\mathcal{M}$ by $\boldsymbol{\psi}^t(\boldsymbol{y}) = (\gamma_1^t r_1(\boldsymbol{y}), \gamma_2^t r_2(\boldsymbol{y}), \ldots)$. Here, ℓ^2 denotes the space of sequences $\{(\mathrm{x}_1, \mathrm{x}_2 \ldots) : \sum_{j=1}^{\infty} \mathrm{x}_j^2 < \infty\}$. Truncating the expansion of $\mathbb{p}_t$ at the first r terms leads to r-dimensional approximations of the diffusion maps $\boldsymbol{\psi}_r^t : \mathcal{M} \to \mathcal{D}_r^{(t)} \subset \mathbb{R}^r$, i.e., $\boldsymbol{\psi}_r^t(\boldsymbol{y}) = (\gamma_1^t r_1(\boldsymbol{y}), \ldots, \gamma_r^t r_r(\boldsymbol{y}))^T$.

Given an isotropic kernel $\mathsf{k}(\boldsymbol{y}, \boldsymbol{y}')$, diffusion maps can be generalized by defining a family of anisotropic kernels $\mathsf{k}^{(\alpha)}(\boldsymbol{y}, \boldsymbol{y}') = \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}')/(\mathbb{d}(\boldsymbol{y}')^\alpha \mathbb{d}(\boldsymbol{y})^\alpha)$, for $\alpha \in \mathbb{R}$, and normalizing the resulting kernel to generalize $\mathbb{p}(\boldsymbol{y}', \boldsymbol{y})$ (or $\mathbf{P}$ in the discrete case) [34, 42, 43]. The standard algorithm described above corresponds to the limiting case of $\alpha = 0$ (isotropic kernel), and we do not consider anisotropic kernels in this paper due to limited space and the lack of inverse mappings for such special cases. In Section 6.2, we describe a new inverse map for the isotropic case only.

Remark 2. We can instead consider the mappings $(\boldsymbol{\psi}_r^t \circ \boldsymbol{\eta})(\cdot) = \boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \to \mathcal{D}_r^{(t)} \subset \mathbb{R}^r$ that map *all* points in the input space to $\mathcal{D}_r^{(t)}$. The mapped point is given by the first r coordinates of the transition kernel $\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}')$ (considering $\boldsymbol{y}$ to be fixed) in the basis $\{l_i\}_{i=1}^{\infty}$. The coefficients are the products of the eigenvalues and the corresponding eigenfunctions r_i evaluated at $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$. We

define composite functions $\mathbb{r}_i(\cdot) = r_i(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \to \mathbb{R}$ to obtain:

$$\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi})) = (\gamma_1^t \mathbb{r}_1(\boldsymbol{\xi}), \ldots, \gamma_r^t \mathbb{r}_r(\boldsymbol{\xi}))^T \in \mathcal{D}_r^{(t)}. \tag{10}$$

The original problem of approximating $\boldsymbol{\eta}(\cdot)$ is replaced with the problem of approximating $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\cdot))$ using the empirical eigenvalues γ_i' and empirical eigenfunctions (eigenvectors) $\boldsymbol{l}_i$ and $\boldsymbol{r}_i$.

A multivariate GP prior indexed by $\boldsymbol{\xi}$ is placed over $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi}))$. Algorithm 2 applied to the original data set $\{\boldsymbol{y}^{(i)}\}_{i=1}^m$ yields the new training points for emulation: $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})) = \boldsymbol{\psi}_r(\boldsymbol{y}^{(j)}) = ((\gamma_1')^t r_{j1}, \ldots, (\gamma_r')^t r_{jr})^T$, $j = 1, \ldots, m$, obtained from the empirical eigenfunctions and empirical eigenvalues.

4. Emulation of coefficients in reduced-dimensional approximations

As explained in Remarks 1 and 2, rather than emulating the outputs in $\mathcal{M}$ directly, we place multivariate GP priors over the reduced-dimensional representations in $\mathcal{F}_r$ or $\mathcal{D}_r^{(t)}$. In the actual approach described in Section 5, we place univariate GP priors over the individual coefficients $\mathbb{r}_i(\cdot)$ or $\mathbb{z}_i(\cdot)$ and emulate these coefficients separately. In this section, we therefore outline scalar GPE.

A scalar valued simulator is a function $\eta : \mathcal{X} \to \mathbb{R}$ of inputs $\boldsymbol{\xi} \in \mathcal{X} \subset \mathbb{R}^l$. In univariate GPE, a GP prior indexed by $\boldsymbol{\xi} \in \mathcal{X}$ is placed over $\eta(\boldsymbol{\xi})$ and the emulator is trained using simulator outputs $\eta(\boldsymbol{\xi}^{(i)})$ at design points $\boldsymbol{\xi}^{(i)}$. We use the notation $\boldsymbol{t} = (\eta(\boldsymbol{\xi}^{(1)}), \ldots, \eta(\boldsymbol{\xi}^{(m)}))^T$. The prior is $\eta(\boldsymbol{\xi})|\boldsymbol{\theta}, \boldsymbol{\beta} \sim \mathcal{GP}\left(m(\boldsymbol{\xi}), \mathbb{c}(\boldsymbol{\xi}, \boldsymbol{\xi}')\right)$, where $\mathcal{GP}\left(m(\cdot), \mathbb{c}(\cdot, \cdot)\right)$ represents a GP with mean and covariance functions $m(\cdot)$ and $\mathbb{c}(\cdot, \cdot)$, respectively.The most common choices for the mean function are a linear function or a constant. In this work, $m \equiv 0$ was assumed by centering the data. $\boldsymbol{\theta}$ is a vector of hyperparameters (e.g., parameters in the covariance function) that are typically unknown *a priori*.

Remark 3. A GP noise term can be added to the model, in which case $\eta(\boldsymbol{\xi})$ is a latent function while the simulator outputs are the observables: $t(\boldsymbol{\xi}) = \eta(\boldsymbol{\xi}) + \epsilon(\boldsymbol{\xi})$, in which $\epsilon(\boldsymbol{\xi}) \sim \mathcal{GP}(0, \sigma_n^2 \delta(\boldsymbol{\xi}, \boldsymbol{\xi}'))$, where $\delta(\cdot, \cdot)$ is the Kronecker delta. The noise can represent modelling or simulation errors or can be included

for numerical stability. It can be included directly as an additional term in the covariance function $\mathbb{c}(\boldsymbol{\xi}, \boldsymbol{\xi}')$ (a so called 'jitter' or 'nugget' [44]), which leads to the same result for GP priors over the noise and latent function.

We use a square exponential covariance function:

$$\mathbb{c}(\boldsymbol{\xi}, \boldsymbol{\xi}') = \theta_0 \exp\left(-(\boldsymbol{\xi} - \boldsymbol{\xi}')^T \mathrm{diag}(\theta_1, \ldots, \theta_l)(\boldsymbol{\xi} - \boldsymbol{\xi}')\right) + \sigma_n^2 \delta(\boldsymbol{\xi}, \boldsymbol{\xi}'), \tag{11}$$

where the last term is the jitter, and set $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_l, \sigma_n^2)^T$. The parameters $\theta_1, \ldots, \theta_l$ are the inverse square correlation lengths. Alternatives to Eq. (11) include the Matérn class of functions and piecewise polynomials, which are also stationary [45].

The conditional predictive distribution at new inputs $\boldsymbol{\xi}$ is obtained in a straightforward manner from the joint distribution $p(\eta(\boldsymbol{\xi}), \boldsymbol{t}|\boldsymbol{\theta})$ [45]:

$$\begin{gathered} \eta(\cdot)|\boldsymbol{t}, \boldsymbol{\theta} \sim \mathcal{GP}\left(m'(\cdot;\boldsymbol{\theta}), \nu'(\cdot,\cdot;\boldsymbol{\theta})\right), \\ m'(\boldsymbol{\xi};\boldsymbol{\theta}) = \boldsymbol{c}(\boldsymbol{\xi})^T \mathbf{C}^{-1} \boldsymbol{t} \quad \text{and} \quad \nu'(\boldsymbol{\xi}, \boldsymbol{\xi}';\boldsymbol{\theta}) = \mathbb{c}(\boldsymbol{\xi}, \boldsymbol{\xi}') - \boldsymbol{c}(\boldsymbol{\xi})^T \mathbf{C}^{-1} \boldsymbol{c}(\boldsymbol{\xi}'), \end{gathered} \tag{12}$$

where $\mathbf{C} = [C_{ij}]$ is the covariance matrix with entries $C_{ij} = \mathbb{c}(\boldsymbol{\xi}^{(i)}, \boldsymbol{\xi}^{(j)})$, $i, j = 1, \ldots, m$, and $\boldsymbol{c}(\boldsymbol{\xi}) = (\mathbb{c}(\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}), \ldots, \mathbb{c}(\boldsymbol{\xi}^{(m)}, \boldsymbol{\xi}))^T$.

The hyperparameters $\boldsymbol{\theta}$ are unknown. Point estimates [10, 46] such as the maximum likelihood estimate (MLE) are employed in most cases; that is, the predictive distribution is given by Eq. (12) using the MLE estimate. The MLE is given by $\arg\max_{\boldsymbol{\theta}} \mathcal{R}(\boldsymbol{\theta})$, where $\mathcal{R}(\boldsymbol{\theta}) = \log p(\boldsymbol{t}|\boldsymbol{\theta})$ is the log likelihood:

$$\mathcal{R}(\boldsymbol{\theta}) = -\frac{1}{2} \ln |\mathbf{C}| - \frac{1}{2} \boldsymbol{t}^T \mathbf{C}^{-1} \boldsymbol{t} - \frac{m}{2} \ln(2\pi). \tag{13}$$

In a Bayesian inference approach, predictions at a new input $\boldsymbol{\xi}$ are made by integrating over $\boldsymbol{\theta}$ in the joint distribution of $\boldsymbol{\theta}$ and $\eta(\boldsymbol{\xi})$ given $\boldsymbol{t}$ (the posterior predictive distribution). The integral is analytically intractable but can be approximated using Monte Carlo integration, e.g., importance sampling, or Markov Chain Monte Carlo [47] to sample from the posterior over the hyperparameters $p(\boldsymbol{\theta}|\boldsymbol{t})$.In this paper, we use an MLE estimate.

5. Multi-output emulation using manifold learning

We replaced the problem of emulating $\boldsymbol{\eta} : \mathcal{X} \to \mathcal{M}$ with the problem of emulating the map $\boldsymbol{z}_r(\boldsymbol{\eta}(\cdot))$ defined by Eq. (4) or the map $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\cdot))$ defined by Eq. (10). Multivariate GP priors are placed over these maps, with training points for emulation given by Algorithms 1 and 2 for kPCA and diffusion maps, respectively. These multivariate GP priors take a particularly convenient form by assuming independence of the coordinates, as explained below.

The kPCA coefficients, $\mathbb{z}_i(\boldsymbol{\xi})$, $i = 1, \ldots, r$ are mutually uncorrelated; following Higdon *et al.* [17] (see also the wavelet decomposition approach in [18]) we therefore make the approximation that they arise from independent GPs. The diffusion map coefficients $\gamma_i \mathbb{r}_i(\boldsymbol{\xi})$, $i = 1, \ldots, r$, on the other hand, are not uncorrelated. As a simplification, however, we treat the underlying GPs as independent (see Remark 4). For both manifold learning methods, univariate GPE is then performed separately on each coefficient to approximate its value for a new input $\boldsymbol{\xi}$. The process is summarized below for each case, making clear the link between the notation of Sections 3 and 4.

1. **kPCA:** For a fixed $i = 1, \ldots, r$, we set $\eta(\boldsymbol{\xi}) = \mathbb{z}_i(\boldsymbol{\xi})$. The training points are given by Eq. (2): $\eta(\boldsymbol{\xi}^{(j)}) = \mathbb{z}_i(\boldsymbol{\xi}^{(j)}) = \widetilde{\boldsymbol{\alpha}}_i^T \mathbf{H}(\boldsymbol{k}_j - \mathbf{K1})$, $j = 1, \ldots, m$. Recall that $\mathbb{z}_i(\boldsymbol{\xi}^{(j)}) = z_i(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})) = z_i(\boldsymbol{y}^{(j)})$. The expected (mean) value at an input $\boldsymbol{\xi}$, given by Eq. (12), yields a prediction that is denoted $\mathbb{z}_i(\boldsymbol{\xi})$ (to avoid introducing new notation, we do not distinguish between $\mathbb{z}_i(\boldsymbol{\xi})$ and $\mathbb{E}[\mathbb{z}_i(\boldsymbol{\xi})]$). We set $\boldsymbol{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi})) = (\mathbb{z}_1(\boldsymbol{\xi}), \ldots, \mathbb{z}_r(\boldsymbol{\xi}))^T$. Again, this is the expected value $\mathbb{E}[\boldsymbol{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi}))]$.
2. **Diffusion maps:** For a fixed $i = 1, \ldots, r$, we set $\eta(\boldsymbol{\xi}) = \mathbb{r}_i(\boldsymbol{\xi})$. The training points are given by Eq. (9): $\eta(\boldsymbol{\xi}^{(j)}) = \mathbb{r}_i(\boldsymbol{\xi}^{(j)}) = r_{ji}$, $j = 1, \ldots, m$. Recall that $\mathbb{r}_i(\boldsymbol{\xi}^{(j)}) = r_i(\boldsymbol{\eta}(\boldsymbol{\xi})^{(j)})) = r_i(\boldsymbol{y}^{(j)})$. For a new input $\boldsymbol{\xi}$, Eq. (12) yields $\mathbb{E}[\mathbb{r}_i(\boldsymbol{\xi})]$, denoted simply as $\mathbb{r}_i(\boldsymbol{\xi})$. We then obtain (the expected value of) $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi})) = ((\gamma_1')^t \mathbb{r}_i(\boldsymbol{\xi}), \ldots, (\gamma_r')^t \mathbb{r}_r(\boldsymbol{\xi}))^T$, which approximates $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi})) = (\gamma_1^t \mathbb{r}_i(\boldsymbol{\xi}), \ldots, \gamma_r^t \mathbb{r}_r(\boldsymbol{\xi}))^T$. Note that while the GPE provides a prediction of the function $\mathbb{r}_i(\boldsymbol{\xi})$, it can provide no information on the

eigenvalues $\gamma_i = \lim_{m\to\infty} \gamma_i'$, which do not depend on $\boldsymbol{\xi}$. Thus, the γ_i' found from Algorithm 2 are used to compute the predicted value of $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi}))$.

Remark 4. To take account of the correlations between the coefficients when using diffusion maps, the linear model of coregionalization (LMC) [13, 48] could be used to emulate the coefficients simultaneously. Alternatively, the GP model could be replaced by an artificial neural network (ANN). For moderately sized r, neither approach is computationally expensive. In this paper, we compare the approach of univariate GPs with ANN using Bayesian regularization [49, 50].

To complete the emulation, we must approximate the inverse map from the reduced-dimensional space $\mathcal{F}_r$ or $\mathcal{D}_r^{(t)}$ to the physical space $\mathcal{M} \subset \mathbb{R}^d$. This so-called pre-image problem can be solved in a number of ways for kPCA but a stable, computationally efficient solution for diffusion maps in high-dimensional spaces does not exist. In the next section, we provide details of the inverse map approximations for both methods, including a new pre-image solution for diffusion maps. The main algorithm for GPE of outputs in high-dimensional spaces is given in Section 6.3.

Remark 5. The GPE framework furnishes predictive variances, given by Eq. (12). The variances pertain to the coefficients ($\mathbb{z}_i$ or $\mathbb{r}_i$) in an abstract space and there is no obvious method to translate this information into variances in the predictions $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi}) \in \mathcal{M}$. The inverse maps discussed below provide only the predictive means of the points $\boldsymbol{y}$. However, we can derive Monte Carlo (MC) estimates of higher-order statistics for a fixed input $\boldsymbol{\xi}$ by drawing samples from the posterior predictive Gaussian distribution (defined by Eq. (12)) over the coefficients $r_i(\boldsymbol{y}) = \mathbb{r}_i(\boldsymbol{\xi})$ or $z_i(\boldsymbol{y}) = \mathbb{z}_i(\boldsymbol{\xi})$ and using the deterministic inverse maps described below.

6. Inverse mappings: Reconstruction of points in $\mathcal{M}$

The final step is to find approximations of the inverse mappings $\boldsymbol{\phi}_r^{-1}(\cdot) : \mathcal{F}_r \to \mathcal{M}$ and $(\boldsymbol{\psi}_r^t)^{-1}(\cdot) : \mathcal{D}_r^{(t)} \to \mathcal{M}$ for kPCA and diffusion maps, respectively. Note

that these are the inverse mappings for the manifold learning methods (from the reduced dimensional space to physical space) and not the inverse mappings for the composite functions $\boldsymbol{\phi}_r(\boldsymbol{\eta}(\cdot))$ and $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\cdot))$. In practical terms (since the feature map is unknown), for kPCA we seek the mapping $\boldsymbol{z}_r^{-1}(\cdot) : \mathbb{R}^r \to \mathcal{M}$, or $\boldsymbol{z}_r \mapsto \boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$. This can be achieved *via* a closed-form least-squares solution [51, 52]. This method, however, can suffer from numerical instabilities if $m < d$ (number of training points is less than the dimension of $\mathcal{M}$), as can the fixed-point iterative algorithm of Mika *et al.* [53] and other minimization routines.

For diffusion maps there has been little progress towards finding an inverse map approximation. Etyngier *et al.* [35] proposed an optimization procedure designed for 2-d shapes embedded in $\mathbb{R}^3$ (a closely related method can be found in [54]). This method uses a Delaunay triangulation into r-simplices of the embedded points in $\mathcal{D}_r^{(t)}$ and takes the points in the simplex containing $\boldsymbol{\psi}_r^t(\boldsymbol{y}) = \boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi}))$ to be the mapped nearest $r+1$ neighbours of $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$ in $\mathcal{M}$. It then proceeds to minimize over the point $\boldsymbol{y}$ and its barycentric coordinates w.r.t. its $r+1$ closest neighbours. For large values of r and d, in particular for $d \gg m$, this procedure will be highly unstable and computationally expensive.

Given the reduced-dimensional representation $\boldsymbol{z}_r(\boldsymbol{y})$ or $\boldsymbol{\psi}_r^t(\boldsymbol{y})$ of an unknown point $\boldsymbol{y}$, a general method for finding the pre-image is to use a weighted average of N_n neighbouring (in some well defined sense) points of $\boldsymbol{y}$. The neighbouring points are taken from the data set, for which the reduced dimensional representations have been computed. In the present case, the data set consists of the m training points $\{\boldsymbol{y}^{(i)}\}_{i=1}^m$. The weighted average can be written as follows:

$$\boldsymbol{y} = \sum_{j \in \mathcal{J}} \vartheta(\boldsymbol{y}^{(j)}) \boldsymbol{y}^{(j)}, \tag{14}$$

in which the weight $\vartheta(\boldsymbol{y}^{(j)})$ is associated with the data point $\boldsymbol{y}^{(j)}$, $j \in \mathcal{J}$, and $\mathcal{J} \subseteq \{1, 2, \ldots, m\}$, which has cardinality N_n, defines the neighbouring points. For example, the weights can be defined in terms of the distances $d_{i,*}$, between $\boldsymbol{y}$ and the data points $\boldsymbol{y}^{(i)}$, $i = 1, \ldots, m$. The simplest approach, known as local linear interpolation [55, 56] is to take $\vartheta(\boldsymbol{y}^{(j)}) = d_{j,*}^{-1} / \sum_{j=1}^m d_{j,*}^{-1}$ and to select

the index set $\mathcal{J}$ according to the N_n points of $\{\boldsymbol{y}^{(j)}\}_{i=1}^m$ with the largest values of $\vartheta(\boldsymbol{y}^{(j)})$. A generalization of this approach uses an isotropic kernel density $\chi(\boldsymbol{y}, \boldsymbol{y}') = \chi(||\boldsymbol{y} - \boldsymbol{y}'||)$ to weight the samples [57]:

$$\vartheta(\boldsymbol{y}^{(j)}) = \frac{\chi(\boldsymbol{y}, \boldsymbol{y}^{(j)})}{\sum_{i=1}^m \chi(\boldsymbol{y}, \boldsymbol{y}^{(i)})} = \frac{\chi(d_{j,*})}{\sum_{i=1}^m \chi(d_{i,*})}, \tag{15}$$

The particular form of kernel density used in this paper is $\chi(\boldsymbol{y}, \boldsymbol{y}') = \exp(-||\boldsymbol{y} - \boldsymbol{y}'||^2)$, which was found to yield more stable and accurate results than local linear interpolation.

The problem is now reduced to finding the distances $d_{i,*}$, $i = 1, \ldots, m$, between $\boldsymbol{y}$ and the training points $\boldsymbol{y}^{(i)}$. For both manifold learning methods, these distances are calculated by finding the corresponding kernel values and exploiting relationships between the kernel function and distances in $\mathcal{M}$.

6.1. Kernel PCA

The data matrix $\Phi = [\boldsymbol{\phi}(\boldsymbol{y}^{(1)}), \ldots, \boldsymbol{\phi}(\boldsymbol{y}^{(m)})]$ can be centered in feature space by $\widetilde{\Phi} = \Phi\mathbf{H}$, yielding $\widetilde{\boldsymbol{w}}_i = \sum_{j=1}^m \widetilde{\alpha}_{ji}\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)}) = \widetilde{\Phi}\widetilde{\boldsymbol{\alpha}}_i = \Phi\mathbf{H}\widetilde{\boldsymbol{\alpha}}_i$, where the $\widetilde{\boldsymbol{\alpha}}_i$ are known from Algorithm 1. The uncentered projection $\boldsymbol{\phi}_r(\boldsymbol{y}) \in \mathcal{F}_r$ of $\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}) \in \mathcal{F}$ onto the first r basis vectors is given by:

$$\begin{aligned} \boldsymbol{\phi}_r(\boldsymbol{y}) = \sum_{i=1}^r z_i \widetilde{\boldsymbol{w}}_i + \overline{\boldsymbol{\phi}} &= \sum_{i=1}^r z_i \Phi\mathbf{H}\widetilde{\boldsymbol{\alpha}}_i + \Phi\mathbf{1} \\ &= \Phi\left(\mathbf{H}[\widetilde{\boldsymbol{\alpha}}_1 \ldots, \widetilde{\boldsymbol{\alpha}}_r]\boldsymbol{z}_r + \mathbf{1}\right) = \Phi\boldsymbol{\tau}. \end{aligned} \tag{16}$$

To find the distances $d_{i,*}$, we note that the distance $\widetilde{d}_{i,*}$ between $\boldsymbol{\phi}(\boldsymbol{y}^{(i)})$ and $\boldsymbol{\phi}(\boldsymbol{y})$ in $\mathcal{F}$ is given by:

$$\widetilde{d}_{i,*}^2 = \boldsymbol{\phi}(\boldsymbol{y})^T\boldsymbol{\phi}(\boldsymbol{y}) + \boldsymbol{\phi}(\boldsymbol{y}^{(i)})^T\boldsymbol{\phi}(\boldsymbol{y}^{(i)}) - 2\boldsymbol{\phi}(\boldsymbol{y})^T\boldsymbol{\phi}(\boldsymbol{y}^{(i)}). \tag{17}$$

Taking $\boldsymbol{\phi}(\boldsymbol{y}) \approx \boldsymbol{\phi}_r(\boldsymbol{y})$ and substituting Eq. (16) into Eq. (17) yields:

$$\widetilde{d}_{i,*}^2 \approx \boldsymbol{\tau}^T\mathbf{K}\boldsymbol{\tau} + \Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(i)}) - 2\boldsymbol{\tau}^T\boldsymbol{k}_i, \tag{18}$$

with $\boldsymbol{\tau}$ defined as in Eq. (16). Note that $\Phi^T\Phi = \mathbf{K}$ and $\boldsymbol{k}_i = \Phi^T\boldsymbol{\phi}(\boldsymbol{y}^{(i)})$. For an isotropic kernel normalized such that $\Bbbk(\boldsymbol{y}', \boldsymbol{y}') = 1$, Eq. (17) gives

$\widetilde{d}^2_{i,*} = 2 - 2\Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y})$, which, equating to the right hand side of Eq. (18), yields $\Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y})$. For the Gaussian kernel, therefore, we obtain $d^2_{i,*} = -s^2 \ln \Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y})$. Similar relationships exist for other commonly used kernel functions [58], e.g., the polynomial kernel $\Bbbk_n(\boldsymbol{y}, \boldsymbol{y}') = \left(\boldsymbol{y}^T \boldsymbol{y}' + c\right)^n$, $c \in \mathbb{R}$, $n = \mathbb{N}$. In combination with Eqs. (14) and (15), the values of $d_{i,*}$ yield an approximation of $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$.

6.2. Diffusion maps

We assume $t = 1$ (without loss of generality) to simplify the notation. At the practical level, we must work within the finite-dimensional setting in which we now have $m+1$ data points; the training points $\{\boldsymbol{y}^{(i)}\}_{i=1}^m$, and the unknown prediction $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$. The original kernel, degree and Markov matrices ($\mathbf{K}$, $\mathbf{D}$ and $\mathbf{P}$) based on the training points can be augmented to reflect the addition of the point $\boldsymbol{y}$. The augmented kernel matrix, denoted $\underline{\mathbf{K}}$, is:

$$\underline{\mathbf{K}} = \left[\begin{array}{c|c} \mathbf{K} & (\mathsf{k}(\boldsymbol{y}^{(1)}, \boldsymbol{y}), \ldots, \mathsf{k}(\boldsymbol{y}^{(m)}, \boldsymbol{y}))^T \\ \hline (\mathsf{k}(\boldsymbol{y}^{(1)}, \boldsymbol{y}), \ldots, \mathsf{k}(\boldsymbol{y}^{(m)}, \boldsymbol{y})) & \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}) \end{array}\right]. \tag{19}$$

The corresponding degree matrix, denoted $\underline{\mathbf{D}}$, is:

$$\underline{\mathbf{D}} = \left[\begin{array}{c|c} \widehat{\mathbf{D}} & 0 \\ \hline 0 & \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}) + \sum_j \mathsf{k}(\boldsymbol{y}^{(j)}, \boldsymbol{y}) \end{array}\right], \tag{20}$$

where $\widehat{\mathbf{D}} = \mathbf{D} + \mathrm{diag}(\mathsf{k}(\boldsymbol{y}^{(1)}, \boldsymbol{y}), \ldots, \mathsf{k}(\boldsymbol{y}^{(m)}, \boldsymbol{y}))$. The new Markov chain, denoted $\underline{\mathbf{P}} = \underline{\mathbf{D}}^{-1}\underline{\mathbf{K}}$, is given by:

$$\underline{\mathbf{P}} = \left[\begin{array}{c|c} \widehat{\mathbf{D}}^{-1}\mathbf{K} & \widehat{\mathbf{D}}^{-1}(\mathsf{k}(\boldsymbol{y}^{(1)}, \boldsymbol{y}), \ldots, \mathsf{k}(\boldsymbol{y}^{(m)}, \boldsymbol{y}))^T \\ \hline \dfrac{(\mathsf{k}(\boldsymbol{y}^{(1)}, \boldsymbol{y}), \ldots, \mathsf{k}(\boldsymbol{y}^{(m)}, \boldsymbol{y}))}{\mathsf{k}(\boldsymbol{y}, \boldsymbol{y}) + \sum_j \mathsf{k}(\boldsymbol{y}^{(j)}, \boldsymbol{y})} & \dfrac{\mathsf{k}(\boldsymbol{y}, \boldsymbol{y})}{\mathsf{k}(\boldsymbol{y}, \boldsymbol{y}) + \sum_j \mathsf{k}(\boldsymbol{y}^{(j)}, \boldsymbol{y})} \end{array}\right]. \tag{21}$$

The $(m+1)$-st row vector of $\underline{\mathbf{P}}$ is denoted $\underline{\boldsymbol{p}}_{m+1}$. The i-th entry in $\underline{\boldsymbol{p}}_{m+1}$ is the transition probability from $\boldsymbol{y}$ to $\boldsymbol{y}^{(i)}$, $i = 1, \ldots, m$ (the last entry is the transition probability from $\boldsymbol{y}$ to $\boldsymbol{y}$). From the discussion in Section 3.2 and Appendix A, we know that the i-th entry of $\underline{\boldsymbol{p}}_{m+1}$ approximates (based on the finite set $\{\boldsymbol{y}^{(i)}\}_{i=1}^m$) the value of the transition kernel $\mathbb{p}(\boldsymbol{y}, \boldsymbol{y}') = \sum_{j=1}^{\infty} \gamma_j r_j(\boldsymbol{y}) l_i(\boldsymbol{y}')$ with

$\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$ fixed, and with $\boldsymbol{y}' = \boldsymbol{y}^{(i)}$; the last entry is the value at $\boldsymbol{y}' = \boldsymbol{y}$. Thus:

$$\begin{aligned} \underline{\boldsymbol{p}}_{m+1} &\approx \sum_{j=1}^{\infty} \gamma_j r_j(\boldsymbol{y}) (l_j(\boldsymbol{y}^{(1)}), \ldots, l_j(\boldsymbol{y}^{(m)}), l_j(\boldsymbol{y}))^T \\ &\approx \sum_{j=1}^{r} \gamma_j r_j(\boldsymbol{y}) (l_j(\boldsymbol{y}^{(1)}), \ldots, l_j(\boldsymbol{y}^{(m)}), l_j(\boldsymbol{y}))^T, \end{aligned} \tag{22}$$

by virtue of the decay in γ_i. The value of $l_j(\boldsymbol{y}^{(i)})$, $i = 1, \ldots, m$, is approximated by the i-th component l_{ij} of $\boldsymbol{l}_j$ (the empirical eigenfunction obtained from the training points). The predicted diffusion coordinates satisfy:

$$\boldsymbol{\psi}_r(\boldsymbol{y}) = (\gamma_1' \mathbb{r}_1(\boldsymbol{\xi}), \ldots, \gamma_r' \mathbb{r}_r(\boldsymbol{\xi}))^T = (\gamma_1' r_1(\boldsymbol{y}), \ldots, \gamma_r' r_r(\boldsymbol{y}))^T. \tag{23}$$

Recall that $\mathbb{r}_i(\boldsymbol{\xi}) = r_i(\boldsymbol{\eta}(\boldsymbol{\xi}))$, which is numerically equal to $r_i(\boldsymbol{y})$ for $i = 1, \ldots, r$, and is thus known. Thus the i-th entry $\underline{p}_{m+1,i}$ of $\underline{\boldsymbol{p}}_{m+1}$ can be approximated as follows:

$$\underline{p}_{m+1,i} \approx \sum_{j=1}^{r} \gamma_j' \mathbb{r}_j(\boldsymbol{\xi}) l_{ij}, \quad i = 1, \ldots, m. \tag{24}$$

Equating this expression with that of the equivalent entry in Eq. (21), we obtain the following:

$$\sum_{j=1}^{r} \gamma_j' \mathbb{r}_j(\boldsymbol{\xi}) l_{ij} = \frac{\mathsf{k}(\boldsymbol{y}^{(i)}, \boldsymbol{y})}{\mathsf{k}(\boldsymbol{y}, \boldsymbol{y}) + \sum_{j=1}^{m} \mathsf{k}(\boldsymbol{y}^{(j)}, \boldsymbol{y})}, \quad i = 1, \ldots, m. \tag{25}$$

For a Gaussian kernel $\mathsf{k}(\boldsymbol{y}, \boldsymbol{y}) = 1$, so solving the system of m equations above yields the unknown kernel values $\mathsf{k}(\boldsymbol{y}^{(i)}, \boldsymbol{y})$, $i = 1, \ldots, m$. The Euclidean distances $d_{i,*}$ are recovered from the kernel values. For a Gaussian kernel, $d_{i,*}^2 = -s^2 \ln \mathsf{k}(\boldsymbol{y}^{(i)}, \boldsymbol{y})$. In combination with Eqs. (14) and (15), these values of $d_{i,*}$ yield an approximation of $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$.

The results of this inverse map approximation on a conical spiral are illustrated in Fig. 1. A conical spiral is a 1-d manifold embedded in 3-d, and is defined by the following equations:

$$x_1 = 4\pi \mathsf{t} \cos(4\pi \mathsf{t}), \quad x_2 = 4\pi \mathsf{t} \sin(4\pi \mathsf{t}), \quad x_3 = 40\pi \mathsf{t}, \tag{26}$$

for a single variable $\mathsf{t} \in \mathbb{R}$. A total of 500 points were sampled from the spiral by sampling 500 values of t from a standard uniform distribution $\mathcal{U}(0, 1)$.

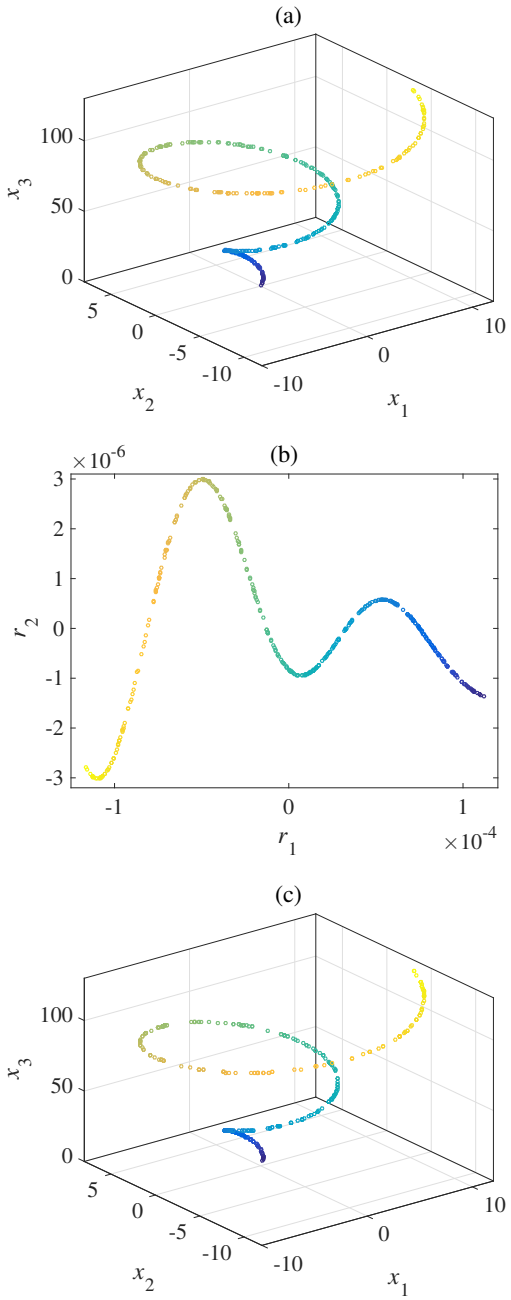

Figure 1: Illustration of the pre-image method for data lying on a conical spiral. 500 points were randomly sampled from the spiral, shown in Fig. (a). A 2-d approximation using diffusion maps is shown in Fig. (b). The reconstruction is illustrated in Fig. (c). Each point in Fig. (a) has a unique color, which is retained in Figs. (b) and (c).

Figure 1(a) shows the sampled points, Fig. 1(b) shows the 2-d ($r = 2$) approximation of the points using diffusion maps, and Fig. 1(c) shows the reconstruction of the original points using the inverse mapping described above. Here, we used $t = 1$ and a Gaussian kernel with s^2 given by the average square distance between observations in the original space [59], as detailed in Section 7.1. Similarly accurate results were obtained for other standard test sets, e.g., the swiss roll and a Gaussian surface.

6.3. Main algorithm

The proposed procedure for GPE of outputs in high-dimensional spaces is summarized in the pseudocode Algorithm 3, based on a Gaussian kernel for both kPCA and diffusion maps.

Algorithm 3: GPE for high-dimensional spaces using manifold learning.

1: Manifold Learning for reduced-dimensional space approximation

kPCA	Diffusion maps
Algorithm 1:	Algorithm 2:
$\left\{(z_1(\boldsymbol{y}^{(j)}),\ldots,z_r(\boldsymbol{y}^{(j)}))^T\right\}_{j=1}^m$	$\left\{(\gamma_1' r_{j1},\ldots,\gamma_r' r_{jr})^T\right\}_{j=1}^m$
$z_i(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})) \leftarrow z_i(\boldsymbol{y}^{(j)})$	$r_i(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})) \leftarrow r_i(\boldsymbol{y}^{(j)}) \leftarrow r_{ji}$

2: **for** $i \leftarrow 1$ to r **do**

kPCA	Diffusion maps
$\left\{\mathbb{z}_i(\boldsymbol{\xi}^{(j)}) \leftarrow z_i(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)}))\right\}_{j=1}^m$	$\left\{\mathbb{r}_i(\boldsymbol{\xi}^{(j)}) \leftarrow r_i(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)}))\right\}_{j=1}^m$
$\left\{\eta(\boldsymbol{\xi}^{(j)}) \leftarrow \mathbb{z}_i(\boldsymbol{\xi}^{(j)})\right\}_{j=1}^m$	$\left\{\eta(\boldsymbol{\xi}^{(j)}) \leftarrow \mathbb{r}_i(\boldsymbol{\xi}^{(j)})\right\}_{j=1}^m$
Scalar GPE: $\mathbb{z}_i(\boldsymbol{\xi}) \leftarrow \mathbb{E}[\eta(\boldsymbol{\xi})]$	Scalar GPE: $\mathbb{r}_i(\boldsymbol{\xi}) \leftarrow \mathbb{E}[\eta(\boldsymbol{\xi})]$

3: **end for**

kPCA	Diffusion maps
$\boldsymbol{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi})) \leftarrow (\mathbb{z}_1(\boldsymbol{\xi}),\ldots,\mathbb{z}_r(\boldsymbol{\xi}))^T$	$\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi})) \leftarrow (\gamma_1^t \mathbb{r}_i(\boldsymbol{\xi}),\ldots,\gamma_r^t \mathbb{r}_r(\boldsymbol{\xi}))^T$

4: Inverse map

$$\boldsymbol{y} \leftarrow \sum_{i=1}^{N_n} \left(\frac{\chi(d_{i,*})}{\sum_{i=1}^{N_n} \chi(d_{i,*})} \right) \boldsymbol{y}^{(i)}$$

kPCA	Diffusion maps ($t=1$)
$\mathbb{k}(\boldsymbol{y}^{(i)},\boldsymbol{y}) \leftarrow \frac{1}{2}\left(1 - \boldsymbol{\tau}^T\mathbf{K}\boldsymbol{\tau} + 2\boldsymbol{\tau}^T\boldsymbol{k}_i\right)$	$\sum_{j=1}^{r} \gamma_j' \mathbb{r}_j(\boldsymbol{\xi}) l_{ij} \leftarrow \frac{\mathsf{k}(\boldsymbol{y}^{(i)},\boldsymbol{y})}{1+\sum_{j=1}^m \mathsf{k}(\boldsymbol{y}^{(j)},\boldsymbol{y})}$
$d_{i,*} \leftarrow \sqrt{-s^2 \ln \mathbb{k}(\boldsymbol{y}^{(i)},\boldsymbol{y})}$	$d_{i,*} \leftarrow \sqrt{-s^2 \ln \mathsf{k}(\boldsymbol{y}^{(i)},\boldsymbol{y})}$

7. Results and discussion

In this section, we consider three examples. In the first example, a single field is emulated, while the second example is concerned with the emulation of three fields simultaneously. The final example considers a nonlinear 2-d model of a hydrogen fuel cell. Unless otherwise stated, for each example a total of 500 inputs were generated using a Sobol sequence. A Sobol sequence [60] is a quasi-random sequence that is specifically designed to generate samples as uniformly as possible over the unit hypercube [61]. For each input $\boldsymbol{\xi}^{(i)}$, $i = 1,\ldots,500$,

simulations were performed to yield data points $\boldsymbol{y}^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)}) \in \mathbb{R}^d$. Of the 500 data points, $m_t = 300$ were reserved for testing and the training points were selected from the remaining 200 ($m \leq 200$). We use $\boldsymbol{y}_p^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)})$ to denote the predicted value of $\boldsymbol{y}^{(i)}$ at a test input $\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, m_t$ using Algorithm 3. A relative error is defined as:

$$\text{Relative error} = \frac{||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2}{||\boldsymbol{y}^{(i)}||^2}, \tag{27}$$

where $||\cdot||$ is the standard Euclidean norm.

7.1. Computational details

Details of the scalar GPE, the manifold learning techniques and the software employed in the implementation of Algorithm 3 are provided below.

1. **kPCA.** A Gaussian kernel was used with the free parameter s^2 taken to be the average square distance between observations in the original space [59]: $s^2 = (1/m^2) \sum_{i,j=1}^{m} ||\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(j)}||^2$. Polynomial and multi-quadratic kernels were also tested but found to be inferior. A sigmoid kernel was found to give similar results to those obtained with a Gaussian kernel. In the inverse mapping, all m points were employed for the reconstruction in physical space (inverse mapping).
2. **Diffusion maps.** A Gaussian kernel was used, in which the value of s^2 was determined as described above. Again, all m points were employed for the reconstruction. A value of $t = 1$ was used in the results presented below. Higher values of t did not lead to any significant changes.
3. **Gaussian Process Emulation**. The square exponential covariance function Eq. (11) was used and the mean function was taken to be identically zero after centering the data (coefficients extracted from the manifold learning technique). The hyper parameters were estimated using the MLE method based on a gradient descent algorithm.

7.2. Free convection in porous media

Subsurface flow in a porous medium can be modelled by Brinkman's equation (with a Boussinesq buoyancy term) and a thermal energy balance [62]:

$$\begin{gathered} -\left(\omega\kappa^{-1}\boldsymbol{v}+\nabla p\right)-\nabla\cdot\omega\epsilon^{-1}\left(\nabla\boldsymbol{v}+\nabla\boldsymbol{v}^{T}\right)=\rho\boldsymbol{g}c_e(T-T_c), \\ \rho C_p\boldsymbol{v}\cdot\nabla T-\nabla\cdot(\overline{\lambda}\nabla T)=0, \\ \nabla\cdot\boldsymbol{v}=0, \end{gathered} \tag{28}$$

in which $\boldsymbol{v}$ is the flow velocity, T is temperature, p is pressure, $\boldsymbol{g}$ is the gravitational acceleration, ρ is the fluid density at a reference temperature T_c, ϵ and κ are the porosity and permeability of the medium, ω is the dynamic viscosity, c_e is the coefficient of volumetric thermal expansion, $\overline{\lambda}$ is the volume averaged thermal conductivity of the fluid-solid mixture, and C_p is the specific heat capacity of the fluid at constant pressure.

We consider a 2-d domain $(x_1, x_2) \in [0, 10] \times [0, 10]$ (in cm) filled with water. The temperature boundary conditions are illustrated in Fig. 2. The temperature ranges from T_h to $T_c < T_h$ along the outer edges. Buoyant flow is generated by the nonuniform temperature. No-slip conditions on all boundaries (with an arbitrary reference p) are assumed. The model was solved using the finite element method (FEM) with triangular elements and a quadratic Lagrange nodal basis. Details of the implementation and default parameter values can be found in [63].

Training and Testing. In this example, the input parameters were $\boldsymbol{\xi} = (c_e[\mathrm{K}^{-1}], T_h[^{\mathrm{o}}\mathrm{C}])^T \in [10^{-11}, 10^{-8}] \times [40, 60]$. For each input $\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, 500$, the magnitude $|\boldsymbol{v}|$ of the velocity was recorded at each grid point on a regular 100×100 square spatial grid and the $d = 10^4$ values of $|\boldsymbol{v}|$ were vectorized to yield the data points $\boldsymbol{y}^{(i)} \in \mathbb{R}^d$, $i = 1, \ldots, 500$. In the notation of Section 2, $u(\boldsymbol{x}; \boldsymbol{\xi}) = |\boldsymbol{v}|$, $J = 1$, $l = 2$ and $d = 10^4$.

Results. Figure 3 shows Tukey box plots of the relative errors for the 300 test cases as the number of training points m and the approximate manifold dimension r are increased. For each box, the central line is the median, the lower and upper edges signify the first (Q_1) and third (Q_3) quartiles. The lower

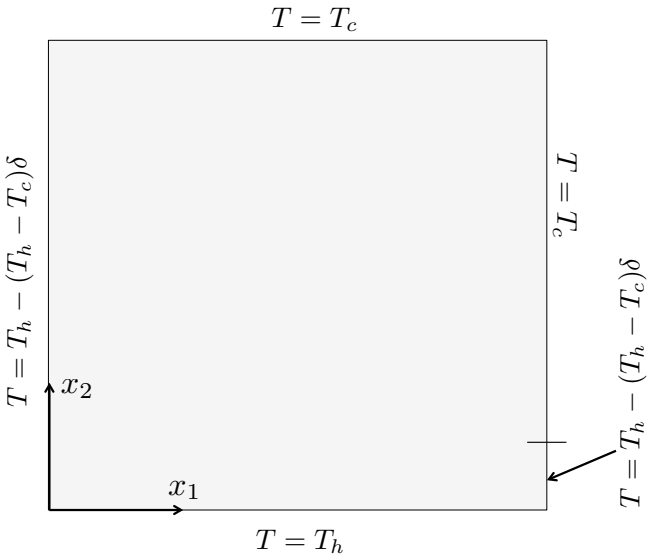

Figure 2: Temperature boundary conditions for the free-convection example. δ is a variable that represents the relative length of a boundary segment and goes from 0 to 1 along the segment as x_2 increases. The cut-off shown by the horizontal dash along $x_1 = 10$ cm is located at $x_2 = 1$ cm.

and upper lines (whiskers) define the errors within $1.5\times(Q_3-Q_1)$ of the first and third quartiles. All other points (considered outliers) are plotted individually using a '+' symbol. A decrease in the relative error for an increasing r is seen for both kPCA and diffusion maps. For both methods, the errors converge at around $r = 6$ dimensions. The median value of the error is marginally lower with kPCA, but it was found that the number of outliers was slightly higher using this method. For a high number of training points ($m \geq 80$), both methods provided accurate predictions and the differences in the errors were not significant.A comparison to Higdon's method [17] can be found in [19], where the same problem is considered and equivalent boxplots are provided. The performance of both kPCA and diffusion maps is far superior. To conserve space, we do not reproduce these results here.

Examples of the predictions are shown in Fig. 4 for 120 training points and $r = 5$. For both kPCA and diffusion maps, the error with respect to the first

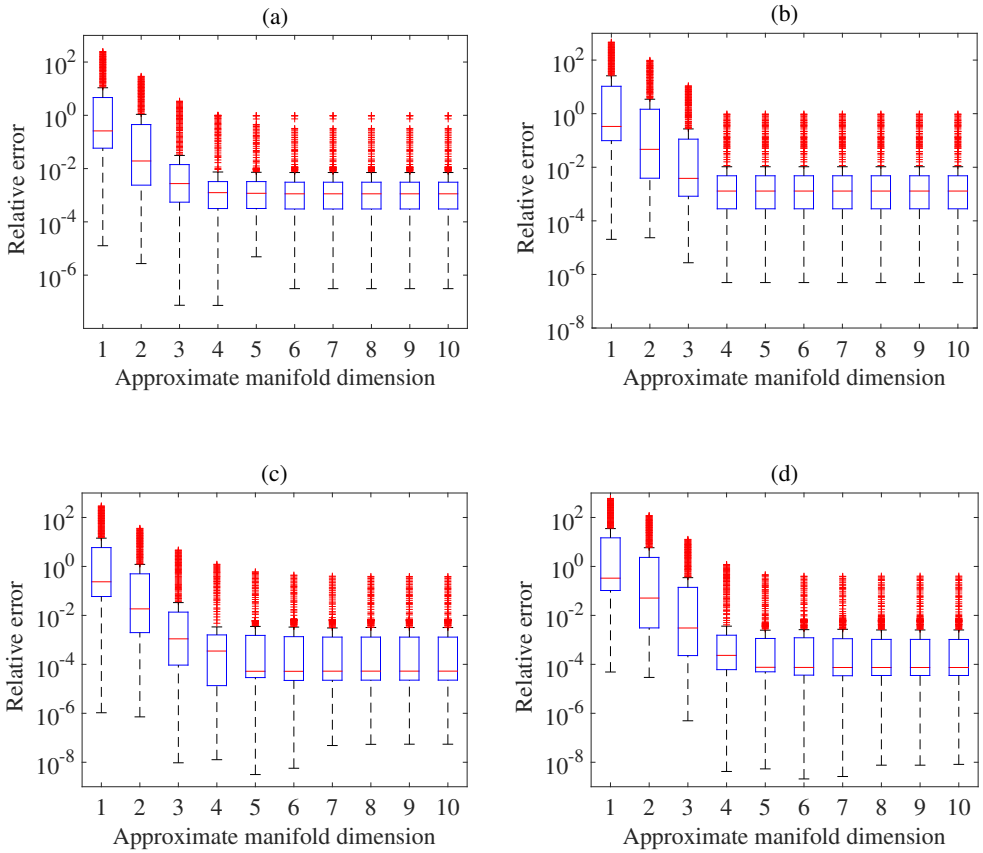

Figure 3: Tukey box plots of the relative error $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2/||\boldsymbol{y}^{(i)}||^2$ in the free-convection example using Algorithm 3 with increasing approximate manifold dimension r on the 300 test points for: (a) kPCA with 40 training points; (b) diffusion maps with 40 training points; (c) kPCA with 120 training points; (d) diffusion maps with 120 training points.

test example (Figs. 4(a)-(c)) lies around the median of the $r = 5$ boxplot in Fig. 3. The errors with respect to the second test example are close to the upper whiskers in the same boxplots. In both cases, Algorithm 3 with either kPCA or diffusion maps yields highly accurate predictions. An example of the outliers for both methods in the $r = 5$ boxplots in Figs. 3(c) and 3(d) is shown in Fig. 5. This figure demonstrates the worst level of prediction, which, nevertheless, captures the qualitative features of the velocity field and remains quantitatively accurate to a reasonable level.

Boxplots of the errors using an ANN and support vector machine regression (SVMR) for emulation of the coefficients, rather than GPE, are shown in Fig. 6 for $m = 120$. In the first case, the correlations between the coefficients are naturally taken into account by approximating the r coefficients simultaneously. To avoid overtraining and cross validation, Bayesian regularization [49, 50] was used for the ANN, implemented in the Matlab Neural Network Toolbox. In this method, zero-mean Gaussian priors are placed over the network weights and an additive noise. Estimates of the weights and hyperparameters (variances in the priors) are found by an iterative procedure based on a Laplace approximation to the posterior over the weights and an evidence approximation for the hyperparameters [47]. A single hidden layer was employed and the number of neurons was selected using a sequential network construction [50].For the SVMR, we tested Gaussian and polynomial kernels (with varying order), together with an L^1 loss function.

Comparing with Figs. 3(a) and (b), we see that GPE and ANN exhibit similar levels of accuracy. This indicates that in this example the assumption of independent GPs for the coefficients in diffusion maps in the GPE framework did not significantly affect the accuracy. The same was true of the other examples (the results are omitted given the limited space). Although this will not be true in general, either ANN or LMC can be used to rigorously incorporate the correlations. For SVRM (implemented in the Statistics and Machine Learning Toolbox in Matlab), a Gaussian kernel gave the best results for both kPCA and diffusion maps. Fig. 6 indicates that, at least in this example, GPE and ANN

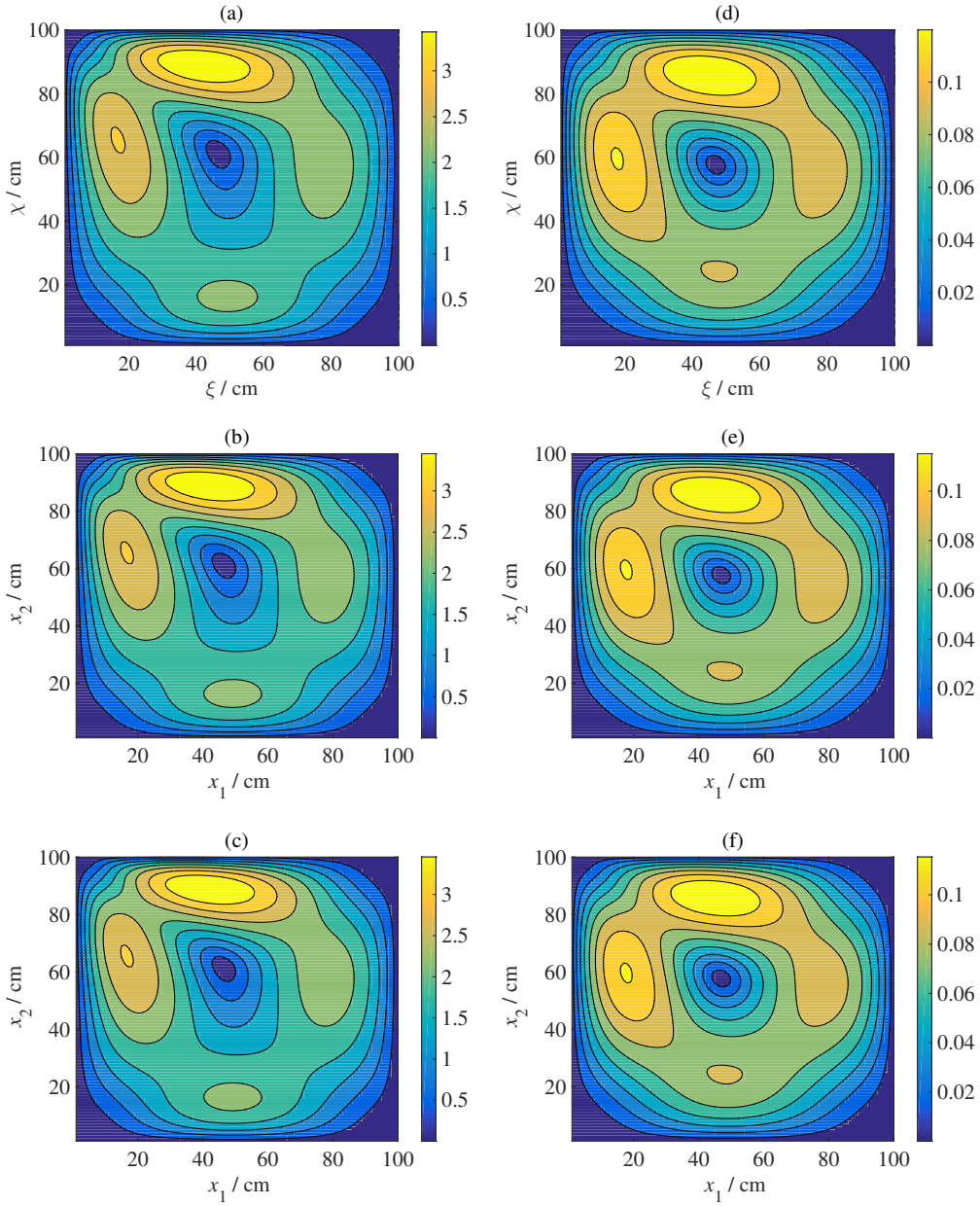

Figure 4: Predictions of the velocity field using 120 training points and $r = 5$ coefficients in the free-convection example. Figure (a) is the test point corresponding to $\boldsymbol{\xi} = (3.18 \times 10^{-9}[\mathrm{K}^{-1}], 56.7[^{\mathrm{o}}\mathrm{C}])^T$, while Figs. (b) and (c) are the corresponding predictions using kPCA and diffusion maps, respectively. Figure (d) is the test point corresponding to $\boldsymbol{\xi} = (7 \times 10^{-11}[\mathrm{K}^{-1}], 46.7[^{\mathrm{o}}\mathrm{C}])^T$, while Figs. (e) and (f) are the corresponding predictions using kPCA and diffusion maps, respectively.

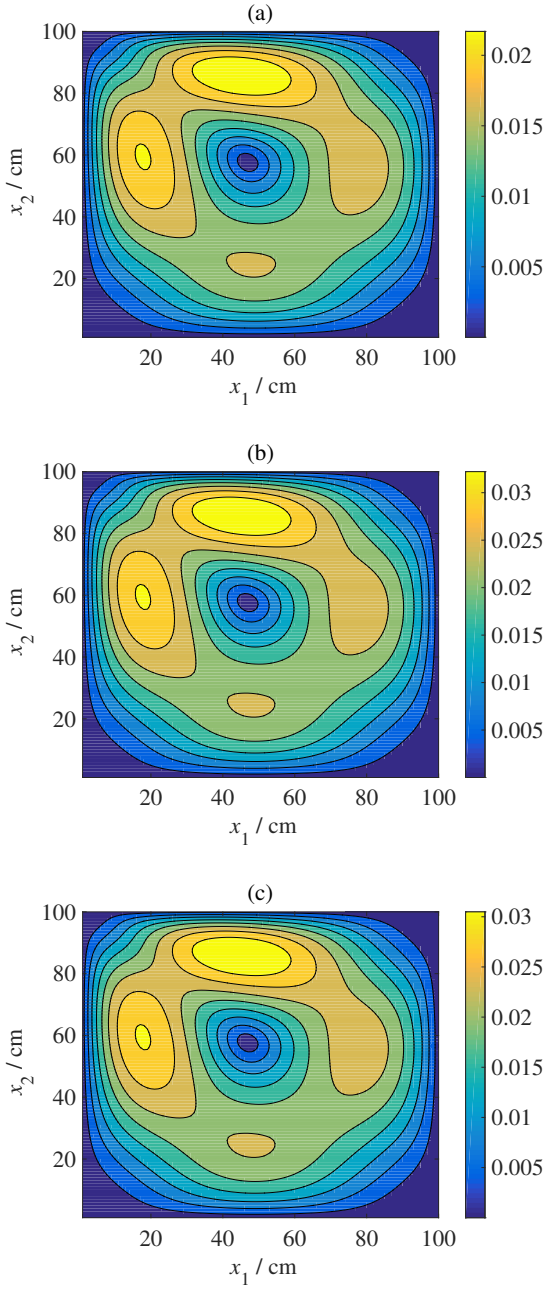

Figure 5: Predictions of the velocity field using 120 training points and $r = 5$ coefficients in the free-convection example in the case of an outlier. Figure (a) is the test point corresponding to $\boldsymbol{\xi} = (1 \times 10^{-9}[\mathrm{K}^{-1}], 40.7[^{\mathrm{o}}\mathrm{C}])^T$, while Figs. (b) and (c) are the predictions using kPCA and diffusion maps, respectively.

are superior.

7.3. Lid driven cavity

We consider a square 2-d cavity $(x_1, x_2) \in [0,1]\times[0,1]$ filled with liquid water. The top boundary represents a sliding lid, which drives the liquid flow. The problem is governed by the steady-state, dimensionless Navier-Stokes equations:

$$(\boldsymbol{v}\cdot\nabla)\boldsymbol{v} - Re^{-1}\nabla^2\boldsymbol{v} + \nabla p = 0, \quad \nabla\cdot\boldsymbol{v} = 0, \tag{29}$$

where $\boldsymbol{v} = (v_1, v_2)^T$ is the liquid velocity, p is the liquid pressure and Re is the Reynolds number. The boundary conditions are $\boldsymbol{v} = (v_1^0, 0)$ for $x_2 = 1$, where v_1^0 is the lid velocity, and $\boldsymbol{v} = 0$ on the other three boundaries. The model was solved using finite differencing on a staggered grid with implicit diffusion and a Chorin projection for the pressure [64].

Training and Testing. The Reynold's number and lid velocity were used as input parameters: $\boldsymbol{\xi} = (Re, v_1^0)^T \in [700, 1200]\times[0.01, 10]$. All other parameters were kept at the default values. For each input $\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, 500$, the pressure p and the component velocities v_1 and v_2 were recorded at each grid point on a regular 100×100 spatial grid. The $d/3 = 10^4$ values of each field variable were vectorized to yield vector outputs $\boldsymbol{y}_{v_1}^{(i)} \in \mathbb{R}^{d/3}$, $\boldsymbol{y}_{v_2}^{(i)} \in \mathbb{R}^{d/3}$ and $\boldsymbol{y}_p^{(i)} \in \mathbb{R}^{d/3}$. The three vectors were then combined into a single vector $\boldsymbol{y}^{(i)} = [\boldsymbol{y}_{v_1}^{(i)} \;\; \boldsymbol{y}_{v_2}^{(i)} \;\; \boldsymbol{y}_p^{(i)}] \in \mathbb{R}^d$ to account for the correlations between the fields. In the notation of Section 2, $J = 3$, $l = 2$ and $d = 3\times10^4$. This is a multiple field example discussed in Section 2, with, e.g., $u_1 = v_1$, $u_2 = v_2$ and $u_3 = p$.

Results. Tukey box plots of the relative error on the 300 test points are shown in Fig. 7 for an increasing r (approximate manifold dimension) and m. Around $r = 5$ is sufficient for both values of m using both methods. In this case, the differences between the methods was almost negligible, except that again there were fewer outliers for diffusion maps, particularly for low numbers of training points.Figure 8 shows the equivalent boxplots using Higdon's method [17]. For this example, Higdon's method also performed well, with superior performance

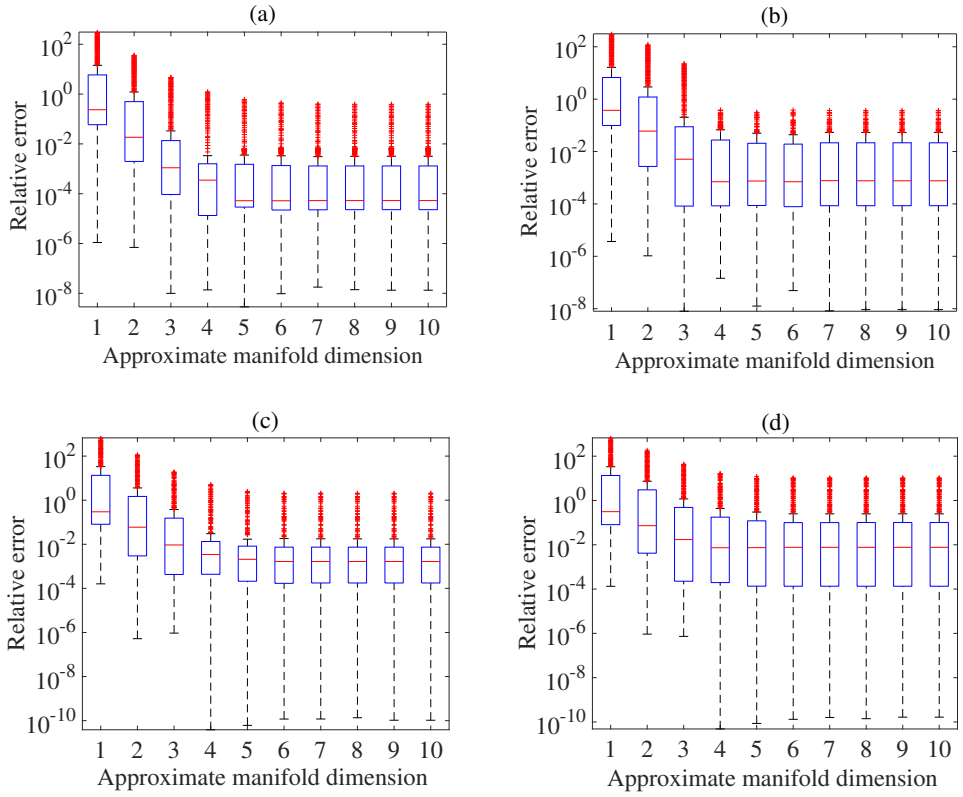

Figure 6: Tukey box plots of the relative error $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2/||\boldsymbol{y}^{(i)}||^2$ in the free-convection example using Algorithm 3 with an ANN and SVMR for an increasing approximate manifold dimension r on the 300 test points. In both cases, 120 training points were used. (a) kPCA with ANN; (b) diffusion maps with ANN; (c) kPCA with SVMR; (d) diffusion maps with SVMR.

at the lower number of training points and slightly inferior performance at a higher number of training points.

Two examples of the predictions are shown in Fig. 9 for 120 training points and $r = 5$. Here, the normalized velocity field is shown as a quiver plot and the surface plot is the pressure field, with contours in black. Note that since only ∇p is meaningful, homogeneous Neumann conditions are prescribed for the pressure Poisson equation, so p is defined only up to a constant (hence the negative values). Stream lines representing contour lines of a stream function ζ are also shown, in white. The stream function is defined by $-\nabla^2\zeta = \partial_{x_2} v_1 - \partial_{x_1} v_2$. For both kPCA and diffusion maps, the error with respect to the first test example (Figs. 9(a)-(c)) lies close to the median in the $r = 5$ boxplot in Fig. 7. The second test example corresponds to an outlier for both methods (relative error around 0.07). The results of Algorithm 3 remain accurate, especially for diffusion maps. The error in kPCA is primarily due to the prediction of the pressure field, in particular the maximum value in the top right corner. Nevertheless, the profile is well captured.

As a further test, we consider a modification of this example, in which the number of inputs is increased to 13 ($l = 13$) using the following boundary conditions:

$$\begin{aligned} v_1(x_1, 1) &= 5c_1 \sin(c_2 \pi x_1) e^{-c_3 x_1}, \quad v_2(x_1, 1) = 0, \\ v_1(x_1, 0) &= 5c_4 \sin(c_5 \pi x_1) e^{-c_6 x_1}, \quad v_2(x_1, 0) = 0, \\ v_2(1, x_2) &= 5c_7 \sin(c_8 \pi x_2) e^{-c_9 x_2}, \quad v_1(1, x_2) = 0, \\ v_2(0, x_2) &= 5c_{10} \sin(c_{11} \pi x_2) e^{-c_{12} x_2}, \quad v_1(0, x_2) = 0, \end{aligned} \tag{30}$$

for constants $c_1, \ldots, c_{12}$. The inputs were defined as $\boldsymbol{\xi} = (Re, c_1, \ldots, c_{12})^T \in [500, 1000] \times (0, 1) \times (0, 1) \times \cdots \times (0, 1)$. Inputs $\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, 1000$ were generated using a Sobol sequence and simulations were performed to yield 1000 data points. Of the 1000 data points, $m_t = 300$ were reserved for testing and the training points were selected from the remaining 700. Both kPCA and diffusion maps exhibited excellent performance, as illustrated in the boxplots in Fig. 10,

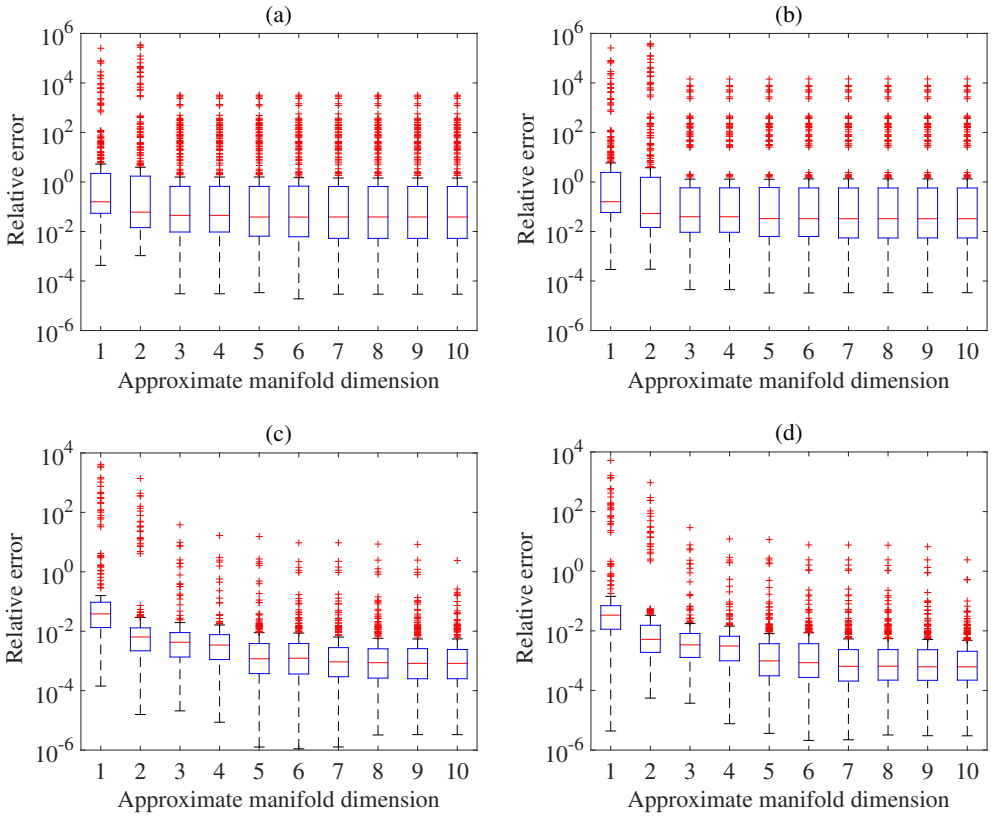

Figure 7: Tukey box plots of the relative error $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2/||\boldsymbol{y}^{(i)}||^2$ in the lid-driven cavity example using Algorithm 3 with an increasing approximate manifold dimension r on the 300 test points for: (a) kPCA with 80 training points; (b) diffusion maps with 80 training points; (c) kPCA with 120 training points; (d) diffusion maps with 120 training points.

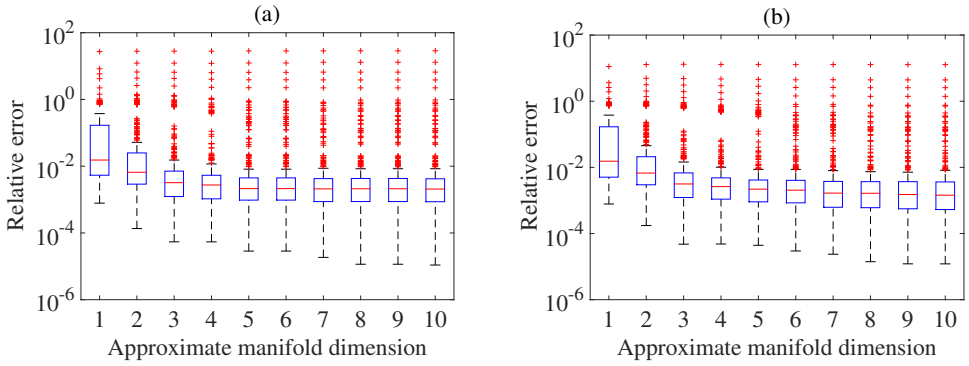

Figure 8: Tukey box plots of the relative error $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2/||\boldsymbol{y}^{(i)}||^2$ in the lid-driven cavity example using Higdon's method [17] with an increasing approximate manifold dimension r on the 300 test points for: (a) 80 training points; (b) 120 training points.

showing the relative error on the 300 test points for an increasing r (approximate manifold dimension) with $m = 500$. Two examples of the fields are shown in Fig. 11 using kPCA with $r = 10$ and $m = 500$. The first example corresponds to an error near the median (for $r = 10$) and the second example is an outlier with a large relative error in the corresponding boxplot. As expected, for a higher dimensional input space, more training points are needed to capture the surface $\mathcal{M}$ accurately. In this case, any lower than 400 training points led to poor performance from all methods.

7.4. Hydrogen fuel cell model

In this example, we consider a hydrogen/oxygen polymer electrolyte membrane (PEM) fuel cell model that incorporates species conservation, charge conservation and a momentum balance in the porous layers. The 2-d domain includes the porous gas diffusion layers (GDLs), through which the species (oxygen, water and hydrogen) are transported from the channels to the reaction sites in the catalyst layers, which are adjacent to the PEM (Fig. 12).

The oxidation reaction in the anode is $2\mathrm{H}_4 \rightarrow 2\mathrm{H}^+ + 4\mathrm{e}^-$ and the reduction reaction in the cathode is $2\mathrm{O}_2 + 4\mathrm{H}^+ + 4\mathrm{e}^- \rightarrow 2\mathrm{H}_2\mathrm{O}$, both of which are assumed to be governed by a modified Butler-Volmer law for charge transfer [65].The

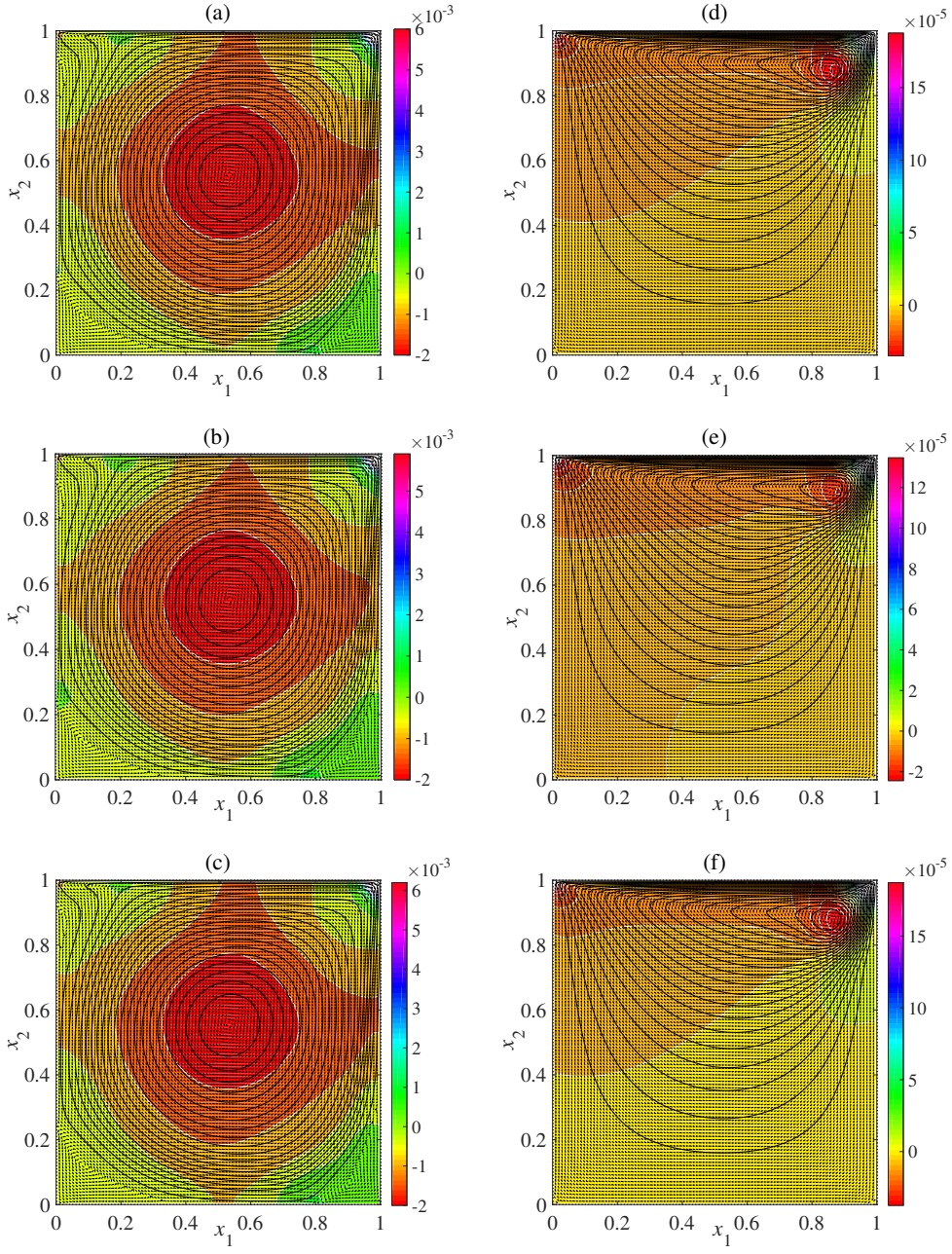

Figure 9: Predictions of the velocity field using 120 training points and $r = 5$ coefficients in the lid driven cavity example. Figure (a) is the test point corresponding to $\boldsymbol{\xi} = (874.8, 7.79)^T$, while Figs. (b) and (c) are the corresponding predictions using kPCA and diffusion maps, respectively. Figure (d) is the test point corresponding to $\boldsymbol{\xi} = (773.24, 0.77)^T$, while Figs. (e) and (f) are the corresponding predictions using kPCA and diffusion maps, respectively.

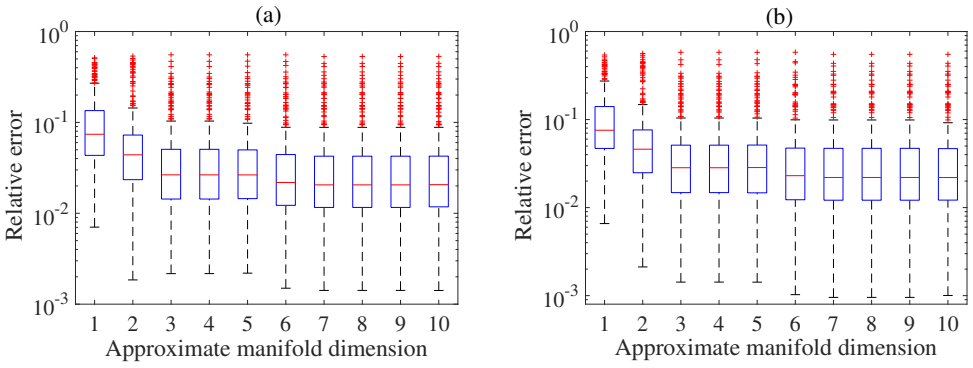

Figure 10: Tukey box plots of the relative error $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2/||\boldsymbol{y}^{(i)}||^2$ in the lid-driven cavity example with boundary conditions as in Eq. (30). The trends are shown for an increasing approximate manifold dimension r using 600 training points and 300 test points for: (a) kPCA and (b) diffusion maps.

catalyst layer morphology is approximated as clusters (agglomerates) of carbon-supported platinum coated with the electrolyte. The transfer current densities are expressed as follows [66]:

$$
\begin{aligned}
j_c &= -\frac{12 L_{act} F D_{agg}}{R_{agg}^2} C_{\mathrm{O}_2,agg}(1-\epsilon_{mac})(1-\lambda_c \coth \lambda_c), \\
j_a &= -\frac{6 L_{act} F D_{agg}}{R_{agg}^2} C_{\mathrm{H}_2,agg}\left(1-e^{-\frac{2F}{RT}\eta_a}\right)(1-\epsilon_{mac})(1-\lambda_a \coth \lambda_a), \\
\lambda_c &= \sqrt{\frac{i_{0c} S R_{agg}^2}{4F C_{\mathrm{O}_2,ref} D_{agg}} e^{\frac{F}{2RT}\eta_c}} \quad \lambda_a = \sqrt{\frac{i_{0a} S R_{agg}^2}{2F C_{\mathrm{H}_2,ref} D_{agg}}},
\end{aligned}
\tag{31}
$$

where $j_a(\eta_a)$ and $j_c(\eta_c)$ are the anode and cathode transfer current densities (overpotentials); R_{agg} and D_{agg} are the radius of the agglomerate and the diffusion coefficient of the reactant through the agglomerate; L_{act} is the catalyst layer thickness (same in both half cells); i_{0a} and i_{0c} are the exchange current densities of the anode and cathode reactions; $C_{\mathrm{O}_2,ref}$ and $C_{\mathrm{H}_2,ref}$ are reference reactant concentrations; $C_{\mathrm{O}_2,agg}$ and $C_{\mathrm{H}_2,agg}$ are the (catalyst) surface concentrations of the reactants; T is temperature, F is Faraday's constant and R is the universal gas constant. The reactants dissolve in the electrolyte at the

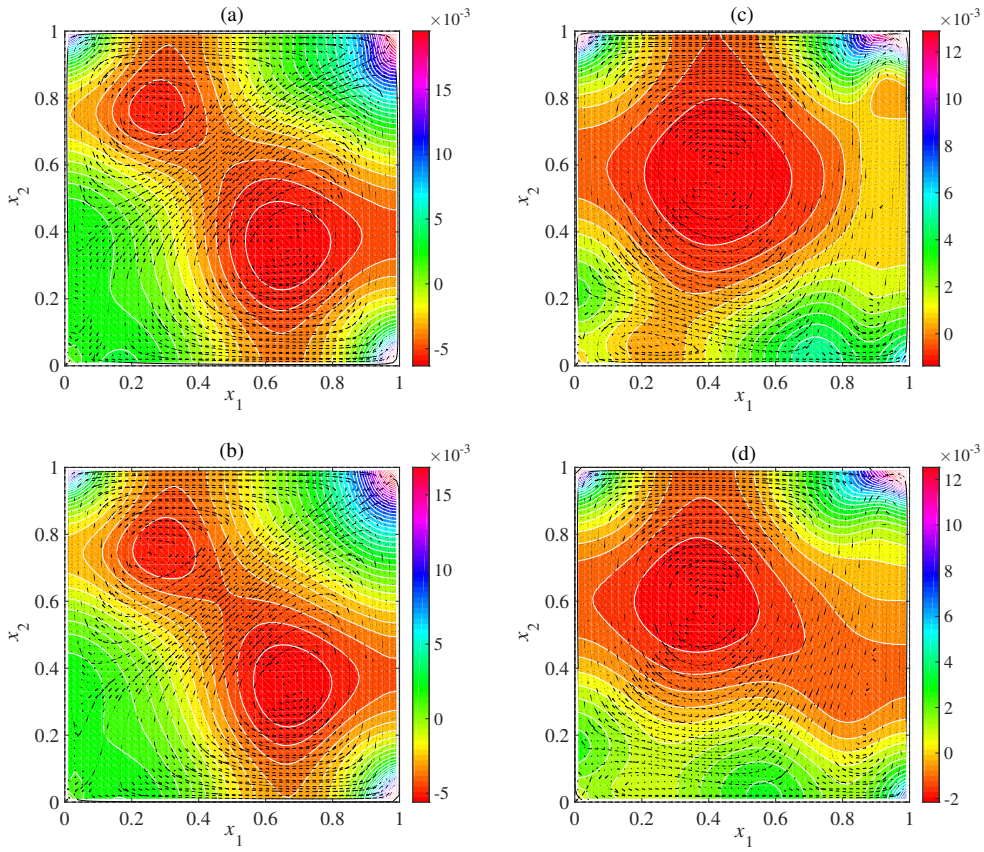

Figure 11: Predictions of the velocity and pressure fields using $m = 500$ training points and $r = 10$ coefficients in the lid driven cavity example with the boundary conditions of Eq. (30). Figure (a) is a test point and Fig. (b) is the corresponding prediction using kPCA, with a relative error of 0.0244. Figure (c) is a second test point and Fig. (d) is the corresponding prediction using kPCA, with a relative error of 0.2275.

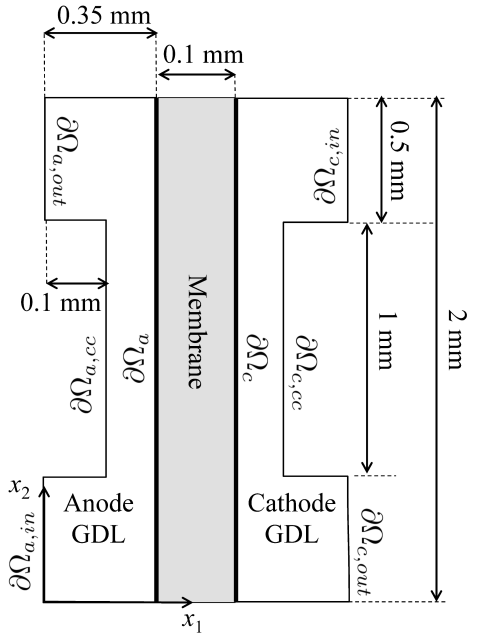

Figure 12: A schematic of the PEM fuel cell and the components that form the model domain.

agglomerate surfaces at a rate governed by Henry's law, so that:

$$C_{\mathrm{H}_2,agg} = pX_{\mathrm{H}_2}/K_{\mathrm{H}_2} \quad C_{\mathrm{O}_2,agg} = pX_{\mathrm{O}_2}/K_{\mathrm{O}_2}, \tag{32}$$

where $X_i(K_i)$ is the mole fraction (Henry constant) of species i and p is the gas pressure.

The charge balances are given by:

$$-\nabla\cdot(\sigma_e\nabla\phi_e) = 0 \quad \text{and} \quad -\nabla\cdot(\sigma_s\nabla\phi_s) = 0, \tag{33}$$

in which $\phi_e(\sigma_e)$ and $\phi_s(\sigma_s)$ are the ionic and electronic potentials (conductivities), respectively. These equations apply to the GDLs. The catalyst layers are approximated by infinitesimally thin surfaces, depicted by $\partial\Omega_a$ and $\partial\Omega_c$ in Fig. 12. The overpotentials (defined only on these boundaries) take the form:

$$\eta_a = \phi_s - \phi_e - E_{eq,a} \quad \text{and} \quad \eta_c = \phi_s - \phi_e - E_{eq,c}, \tag{34}$$

in which $E_{eq,a}$ and $E_{eq,c}$ are the equilibrium potentials for the reactions.

Flow through the GDLs is governed by continuity and Darcy's law:

$$\nabla \cdot (\rho \boldsymbol{v}) = 0, \quad \boldsymbol{v} = -k_p \omega^{-1} \nabla p, \tag{35}$$

where ω is the gas viscosity and k_p is the GDL permeability. The ideal gas law is used to determine the density: $\rho = (p/RT) \sum_i M_i X_i$, in which M_i is the molecular weight of species $i \in \{\mathrm{H_2}, \mathrm{O_2}, \mathrm{H_2O}, \mathrm{N_2}\}$. The transport of species through the GDLs is governed by convection and multicomponent diffusion (Stefan-Maxwell) [67]. In the cathode, the species are $\mathcal{I}_1 = \{\mathrm{O_2}, \mathrm{H_2O}, \mathrm{N_2}\}$ and in the anode the species are $\mathcal{I}_2 = \{\mathrm{H_2}, \mathrm{H_2O}, \mathrm{N_2}\}$. The transport equations in the cathode are given by:

$$\begin{gathered} -\nabla \cdot \left\{ \rho Y_i \sum_{\substack{j \in \mathcal{I}_1 \\ j \neq i}} D_{i,j} \left(\nabla X_j + (X_j - Y_j) \nabla p / p \right) \right\} = -\rho \boldsymbol{v} \cdot \nabla Y_i, \\ Y_{\mathrm{N_2}} = 1 - Y_{\mathrm{O_2}} - Y_{\mathrm{H_2O}}, \end{gathered} \tag{36}$$

for $i \in \{\mathrm{O_2}, \mathrm{H_2O}\}$. Y_i is the mass fraction of species i and the $D_{i,j}$ are binary diffusivities [67]. Identical equations for species $\mathcal{I}_2$ are solved in the anode.

The boundary conditions for the potential impose a cell voltage V_{cell}:

$$\begin{gathered} \phi_s = 0 \quad \boldsymbol{x} \in \partial\Omega_{a,cc}, \\ \phi_s = V_{cell} \quad \boldsymbol{x} \in \partial\Omega_{c,cc}, \\ -\boldsymbol{n} \cdot \nabla \phi_s = 0 \quad \text{otherwise}, \end{gathered} \tag{37}$$

where $\boldsymbol{n}$ is the outwardly pointing unit normal. At the inlets ($\partial\Omega_{a,in}$ and $\partial\Omega_{c,in}$) and outlets ($\partial\Omega_{a,out}$ and $\partial\Omega_{c,out}$), the total gas pressures and the mole fractions of the reactants are specified. At $\partial\Omega_a$ and $\partial\Omega_c$, the gas velocity is calculated from the total mass flow based on Faraday's law [65]:

$$\begin{aligned} -\boldsymbol{n} \cdot \boldsymbol{v} &= j_a \left(M_{\mathrm{H_2}}/2 + \lambda_{\mathrm{H_2O}} M_{\mathrm{H_2O}} \right) / (\rho F) \quad \boldsymbol{x} \in \partial\Omega_a, \\ -\boldsymbol{n} \cdot \boldsymbol{v} &= j_c \left(M_{\mathrm{O_2}}/2 + \left[1/2 + \lambda_{\mathrm{H_2O}} \right] M_{\mathrm{H_2O}} \right) / (\rho F) \quad \boldsymbol{x} \in \partial\Omega_c, \end{aligned} \tag{38}$$

where $\lambda_{\mathrm{H_2O}}$ is the water drag number [65]. At the other boundaries except the inlets and outlets $-\boldsymbol{n} \cdot (\rho \boldsymbol{v}) = 0$ is imposed. At the catalyst layer surfaces the

mass fluxes of reactants are determined by Faraday's law:

$$\begin{aligned} -\boldsymbol{n}\cdot\boldsymbol{N}_{\mathrm{H_2}} &= M_{\mathrm{H_2}} j_a/(2F) \quad \boldsymbol{x}\in\partial\Omega_a, \\ -\boldsymbol{n}\cdot\boldsymbol{N}_{\mathrm{O_2}} &= M_{\mathrm{O_2}} j_c/(4F) \quad \boldsymbol{x}\in\partial\Omega_c, \\ -\boldsymbol{n}\cdot\boldsymbol{N}_{\mathrm{H_2O}} &= M_{\mathrm{H_2O}} j_c\left(1/2+\lambda_{\mathrm{H_2O}}\right)/F \quad \boldsymbol{x}\in\partial\Omega_c, \end{aligned} \tag{39}$$

where $\boldsymbol{N}_i = -\rho Y_i \sum_{j\neq i} D_{i,j}(\nabla X_j + (X_j - Y_j)\nabla p/p) + \rho \boldsymbol{v} Y_i$ is the flux of species i. At all other boundaries except the inlets and outlets, $\boldsymbol{N}_i = 0$. The model was solved using the FEM with 10236 triangular domain elements, 582 boundary elements and a Lagrange basis of order 2. Details of the implementation and the default parameter values can be found in [68].

Training and Testing. The cell voltage V_{cell} and the membrane/electrolyte conductivity σ_e were used as input parameters: $\boldsymbol{\xi} = (V_{cell}[\mathrm{V}], \sigma_e[\mathrm{S\ m^{-1}}])^T \in [0.2, 0.8]\times[1, 15]$. For each input $\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, 500$, the mole fraction of water $X_{\mathrm{H_2O}}$ was recorded at each point on a regular 150×300 spatial grid in the cathode GDL. $X_{\mathrm{H_2O}}$ in the cathode (where water is produced) is a key quantity. High values can lead to flooding of the electrode, which would prevent the fuel cell from operating. The $d = 4.5\times10^4$ values of $X_{\mathrm{H_2O}}$ were re-ordered into vector form to yield vectors $\boldsymbol{y}^{(i)} \in \mathbb{R}^d$. In the notation of Section 2, $u(\boldsymbol{x};\boldsymbol{\xi}) = X_{\mathrm{H_2O}}$, $J = 1$, $l = 2$ and $d = 4.5 \times 10^4$.

Results. Figure 13 shows the Tukey box plots of the relative error for increasing r (approximate manifold dimension) and m. The results using both methods are highly accurate, particularly for $m = 120$ (in fact, $m = 80$ was found to give a similar level of performance). The performance with diffusion maps is better for $m = 60$, while the performance with kPCA is slightly superior with $m = 120$. Again there are more outliers in the box plots for kPCA. Examples of the predictions are shown in Fig. 14 for 120 training points and $r = 7$. In the first example (Figs. 14(a)-(c)), the error with respect to the test case lies close to the median in the $r = 7$ boxplot for kPCA (Fig. 13(c)), while for diffusion maps the error is near the upper whisker in the corresponding boxplot ($m = 120$, $r = 7$ in Fig. 13(d)). The second example (Figs. 14(d)-(f)) is an outlier for both kPCA and diffusion maps (second and third highest errors, respectively). Even

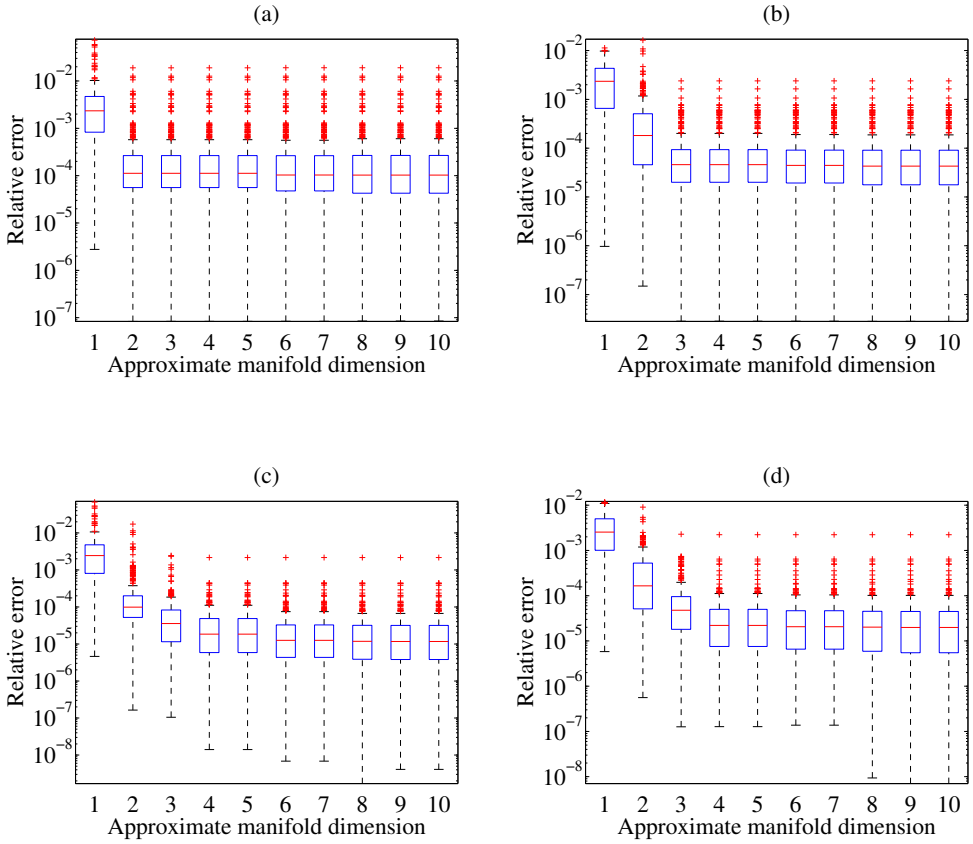

Figure 13: Tukey box plots of the relative error $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2/||\boldsymbol{y}^{(i)}||^2$ in the PEM fuel cell example using Algorithm 3 with increasing approximate manifold dimension r on the 300 test points for: (a) kPCA with 40 training points; (b) diffusion maps with 40 training points; (c) kPCA with 120 training points; (d) diffusion maps with 120 training points.

in the latter case, the predictions are accurate.

8. Concluding Remarks

We have developed an approach to the GP emulation of outputs in very high dimensional spaces. We use manifold learning methods to exploit patterns in the permissible output space. The motivation is parameter dependent spatial fields governed by PDE models, but the approach can be applied to any problem involving vector-valued targets (with or without noise) and vector-valued inputs. For both kPCA and diffusion maps we show how the coefficients can be used as targets for emulation, with a subsequent inverse mapping of the predicted points in an reduced-dimensional space to the original (physical) space. In particular, we have developed an approximate inverse mapping for diffusion maps.

There are several powerful approaches to manifold learning other than those used in this paper, including Laplacian eigenmaps, local linear embedding (LLE) [69] and local tangent space alignment (LSTA) [70]. These methods may offer improved accuracy if solutions to the associated pre-image problems can be found.

Acknowledgments

The work at the Warwick Centre for Predictive Modelling was supported by EPSRC (Grant No. EP/L027682/1). W.W.X. acknowledges the Chinese Scholarship Council for a doctoral scholarship. A.A.S. acknowledges support from the EU Framework Programme 7 (Grant No. 314159). N.Z. acknowledges support from the College of Engineering at the University of Notre Dame and the Computer Science and Mathematics Division of ORNL under the DARPA EQUiPS program. N.Z. also acknowledges the Royal Society for support through a Wolfson Research Merit Award and the Technische Universität München, Institute for Advanced Study for support through a Hans Fisher Senior Fellowship.

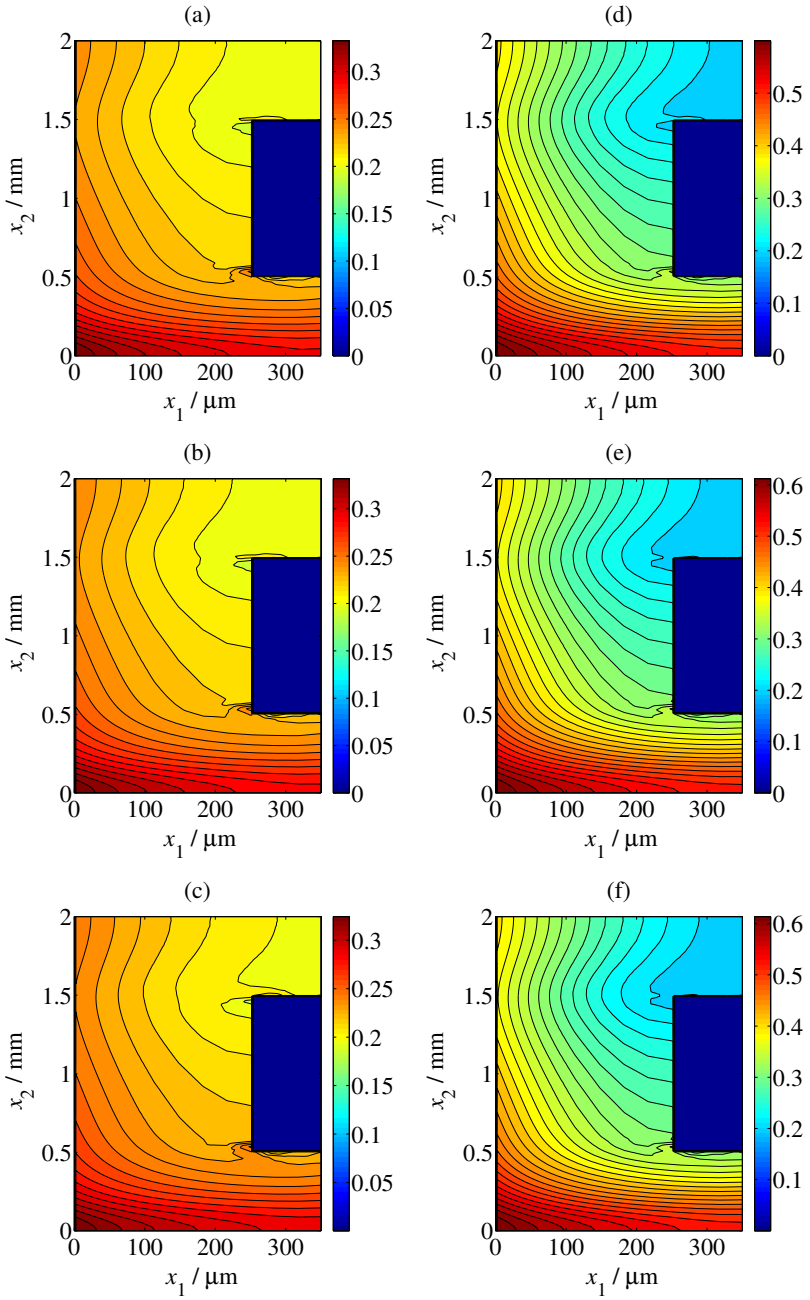

Figure 14: Predictions of the water mole fraction using 120 training points and $r = 7$ coefficients in the PEM fuel cell example. Figure (a) is the test point corresponding to $\boldsymbol{\xi} = (0.525[\text{V}], 1.492[\text{S m}^{-1}])^T$, while Figs. (b) and (c) are the corresponding predictions using kPCA and diffusion maps, respectively. Figure (d) is the test point corresponding to $\boldsymbol{\xi} = (0.301[\text{V}], 9.039[\text{S m}^{-1}])^T$ obtained using direct simulation, while Figs. (e) and (f) are the corresponding predictions using kPCA and diffusion maps, respectively.

9. Appendix A: Continuous state space diffusion maps

Full details of the following can be found in the references [34, 37, 40, 41, 71]. In the limit $m \to \infty$, the Markov chain with transition matrix $\mathbf{P}$ generated from a Gaussian kernel (with scale parameter s^2) converges towards a Markov chain on the continuous state space $\mathcal{M}$ [34, 37, 40, 41, 71], with a discrete-time step s^2. Let μ be a probability measure on $\mathcal{M}$ defining the density of points, e.g., the Lebesgue measure for a uniform density. In the limit $m \to \infty$, a one-step (from $\boldsymbol{y}' \in \mathcal{M}$ to $\boldsymbol{y} \in \mathcal{M}$) transition kernel for the Markov chain on $\mathcal{M}$ can be defined by $\mathbb{p}(\boldsymbol{y}', \boldsymbol{y}) = \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}')/\mathbb{d}(\boldsymbol{y}')$, where $\mathbb{d}(\boldsymbol{y}') = \int_{\mathcal{M}} \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}') d\mu(\boldsymbol{y})$ is a normalization factor. $\mathbb{p}(\boldsymbol{y}', \boldsymbol{y})$ is the continuous equivalent of the elements of $\mathbf{P}$. The evolution of a probability distribution $\varphi(\boldsymbol{y})$ is determined by the Markov operator $\mathcal{L}$ (forward transfer operator or propagator) defined as follows [40, 41]:

$$\mathcal{L}\varphi(\boldsymbol{y}) = \int_{\mathcal{M}} \mathbb{p}(\boldsymbol{y}', \boldsymbol{y}) \varphi(\boldsymbol{y}') d\mu(\boldsymbol{y}'). \tag{A1}$$

for $\varphi(\boldsymbol{y}') \in L^2(\mathcal{M}, \mu)$The distribution after t steps is given by $\mathcal{L}^t\varphi = \mathcal{L} \circ \mathcal{L} \circ \cdots \circ \mathcal{L}\varphi$. $\mathcal{L}$ is equivalent to multiplication of $\mathbf{P}$ from the left in the case of a finite state space. The adjoint of $\mathcal{L}$ under the $L^2(\mathcal{M}, \mu)$ inner product $\langle \varphi_1, \varphi_2 \rangle = \int_{\mathcal{M}} \varphi_1(\boldsymbol{y}) \varphi_2(\boldsymbol{y}) d\mu(\boldsymbol{y})$ is given by the following backward transfer operator [40, 41]:

$$\mathcal{R}\varphi(\boldsymbol{y}) = \int_{\mathcal{M}} \mathbb{p}(\boldsymbol{y}, \boldsymbol{y}') \varphi(\boldsymbol{y}') d\mu(\boldsymbol{y}'), \quad \langle \mathcal{L}\varphi_1, \varphi_2 \rangle = \langle \varphi_1, \mathcal{R}\varphi_2 \rangle, \tag{A2}$$

for $\varphi_1, \varphi_2 \in L^2(\mathcal{M}, \mu)$. In Eq. (A2), if $\varphi(\boldsymbol{y})$ is a function defined on $\mathcal{M}$, then $\mathcal{R}\varphi(\boldsymbol{y})$ is the mean value of the function after one step of a random walk that started at $\boldsymbol{y}$. $\mathcal{R}^t\varphi$ gives the mean value after t steps. The action of $\mathcal{R}$ is equivalent to multiplication of $\mathbf{P}$ from the right in finite state space.

By defining a symmetric transition kernel $\mathbb{p}_s(\boldsymbol{y}', \boldsymbol{y}) = \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}')/\sqrt{\mathbb{d}(\boldsymbol{y}')}\sqrt{\mathbb{d}(\boldsymbol{y})}$, we obtain the following self-adjoint, compact operator $\mathcal{S}$ [34, 37]:

$$\mathcal{S}\varphi(\boldsymbol{y}) = \int_{\mathcal{M}} \mathbb{p}_s(\boldsymbol{y}, \boldsymbol{y}') \varphi(\boldsymbol{y}') d\mu(\boldsymbol{y}'), \quad \langle \mathcal{S}\varphi_1, \varphi_2 \rangle = \langle \varphi_1, \mathcal{S}\varphi_2 \rangle, \tag{A3}$$

for $\varphi_1, \varphi_2 \in L^2(\mathcal{M}, \mu)$. $\mathcal{S}$ is the continuous space equivalent of the action of $\mathbf{P}' = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{1/2}$. From the spectral theory for compact, self-adjoint operators, $\mathcal{S}$

admits a discrete eigendecomposition $\mathcal{S}s_i = \gamma_i s_i$, $i \in \mathbb{N}$, the eigenvalues (all positive) can be ordered such that $1 = \gamma_1 > \gamma_2 > \cdots$, and the eigenfunctions form an orthonormal basis for $L^2(\mathcal{M}, \mu)$. Moreover, we obtain the expansion $\mathbb{p}_s(\boldsymbol{y}, \boldsymbol{y}') = \sum_{i=1}^{\infty} \gamma_i s_i(\boldsymbol{y}) s_i(\boldsymbol{y}')$. Since $\mathcal{S}$ is obtained *via* conjugation of the kernel $\mathbb{p}(\boldsymbol{y}', \boldsymbol{y})$ with $\sqrt{\mathbb{d}(\boldsymbol{y})}$, the operators $\mathcal{L}$, $\mathcal{R}$ and $\mathcal{S}$ share the same eigenvalues γ_i, while the eigenfunctions of $\mathcal{L}$ and $\mathcal{R}$ are given by $l_i = s_i(\boldsymbol{y})\sqrt{\mathbb{d}(\boldsymbol{y})}$ and $r_i = s_i(\boldsymbol{y})/\sqrt{\mathbb{d}(\boldsymbol{y})}$, $i \in \mathbb{N}$, respectively.

From the spectral expansion of $\mathbb{p}_s(\boldsymbol{y}, \boldsymbol{y}')$ and the above relationships between the eigenfunctions, we obtain the expansion $\mathbb{p}(\boldsymbol{y}, \boldsymbol{y}') = \sum_{i=1}^{\infty} \gamma_i r_i(\boldsymbol{y}) l_i(\boldsymbol{y}')$ [34]. The t-step transition probabilities $p_t(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$ defined in Eq. (5), i.e., elements of $\mathbf{P}^t$, are now given by the transition kernel $\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}')$ of $\mathcal{R}^t = \mathcal{R} \circ \cdots \circ \mathcal{R}$, which admits the expansion $\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}') = \sum_{i=1}^{\infty} \gamma_i^t r_i(\boldsymbol{y}) l_i(\boldsymbol{y}')$.Considering $\boldsymbol{y} \in \mathcal{M}$ to be fixed, this gives a function of $\boldsymbol{y}' \in \mathcal{M}$ that is a continuous equivalent of the vector of probabilities $\boldsymbol{p}_j^t$ given by Eq. (5), in which $\boldsymbol{y} = \boldsymbol{y}^{(j)}$ and $\boldsymbol{y}' \in \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(m)}\}$ belongs to the finite set of states accessible from $\boldsymbol{y}^{(j)}$. The basis $\{\boldsymbol{l}_i\}_{i=1}^m$ is replaced with $\{l_i\}_{i=1}^{\infty}$(defined on the whole of $\mathcal{M}$) and the i-th coordinate $(\gamma_i')^t r_{ji}$ is now replaced by the function $\gamma_i^t r_i$ evaluated at thegeneral starting location $\boldsymbol{y} \in \mathcal{M}$. Given the decay in the eigenvalues, the expansion for $\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}')$ can likewise be truncated at the first few eigenfunctions $\{l_i\}_{i=1}^r$.

A continuous version of the diffusion distance can now be defined as [34]:

$$D_t^2(\boldsymbol{y}_1, \boldsymbol{y}_2) = ||\mathbb{p}_t(\boldsymbol{y}_1, \boldsymbol{y}') - \mathbb{p}_t(\boldsymbol{y}_2, \boldsymbol{y}')||_{1/\mathbb{d}}^2 = \sum_{i=1}^{\infty} \gamma_i^{2t} [r_i(\boldsymbol{y}_1) - r_i(\boldsymbol{y}_2)]^2, \quad \text{(A4)}$$

where $||\varphi||_{1/\mathbb{d}}^2 = \langle \varphi, \varphi \rangle_{1/\mathbb{d}} = \int_{\boldsymbol{y}' \in \mathcal{M}} |\varphi(\boldsymbol{y}')|^2 / \mathbb{d}(\boldsymbol{y}') d\mu(\boldsymbol{y}')$ for functions $\{\varphi : ||\varphi||_{1/\mathbb{d}} < \infty\}$. The last step in Eq. (A4) follows immediately from the orthonormality of $\{l_i\}_{i=1}^{\infty}$ w.r.t. the inner product $\langle \cdot, \cdot \rangle_{1/\mathbb{d}}$.

Thus, the diffusion maps can be generalized to maps $\boldsymbol{\psi}^t : \mathcal{M} \to \mathcal{D}^{(t)} \subset \ell^2$ on the continuous state space $\mathcal{M}$ as follows: $\boldsymbol{\psi}^t(\boldsymbol{y}) = (\gamma_1^t r_1(\boldsymbol{y}), \gamma_2^t r_2(\boldsymbol{y}), \ldots)$. Here, ℓ^2 denotes the space of sequences $\{(\mathrm{x}_1, \mathrm{x}_2 \ldots) : \sum_{j=1}^{\infty} \mathrm{x}_j^2 < \infty\}$. Restricting the expansion of $\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}')$ to the first r eigenfunctions l_i, we can define the maps

$\boldsymbol{\psi}_r^t : \mathcal{M} \to \mathcal{D}_r^{(t)} \subset \mathbb{R}^r$ as follows:

$$\boldsymbol{\psi}_r^t(\boldsymbol{y}) = (\gamma_1^t r_1(\boldsymbol{y}), \ldots, \gamma_r^t r_r(\boldsymbol{y})) \in \mathcal{D}_r^{(t)}. \tag{A5}$$

[1] M. C. Kennedy, A. O'Hagan, Predicting the output from a complex computer code when fast approximations are available, Biometrika 87 (1) (2000) 1–13.
URL http://www.jstor.org/stable/2673557

[2] T. Santner, B. Williams, W. Notz, The Design and Analysis of Computer Experiments, Springer, 2003.
URL http://www.springer.com/us/book/9780387954202

[3] I. Bilionis, N. Zabaras, B. A. Konomi, G. Lin, Multi-output separable Gaussian process: Towards an efficient, fully Bayesian paradigm for uncertainty quantification, Journal of Computational Physics 241 (2013) 212 – 239. doi:http://dx.doi.org/10.1016/j.jcp.2013.01.011.
URL http://www.sciencedirect.com/science/article/pii/S0021999113000417

[4] A. Keane, P. Nair, Computational Approaches for Aerospace Design, John-Wiley and Sons, 2005.
URL http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470855401.html

[5] J. Oakley, A. O'Hagan, Bayesian inference for the uncertainty distribution of computer model outputs, Biometrika 89 (4) (2002) 769–784. arXiv:http://biomet.oxfordjournals.org/content/89/4/769.full.pdf+html, doi:10.1093/biomet/89.4.769.
URL http://biomet.oxfordjournals.org/content/89/4/769.abstract

[6] M. C. Kennedy, C. W. Anderson, S. Conti, A. O'Hagan, Case studies in Gaussian process modelling of computer codes, Reliability Engineering & System Safety 91 (10) (2006) 1301–1309.

[7] J. Rougier, D. M. H. Sexton, J. M. Murphy, D. Stainforth, Analyzing the climate sensitivity of the HadSM3 climate model using ensembles from different but related experiments, Journal of Climate 22 (13) (2009)

3540–3557. arXiv:http://dx.doi.org/10.1175/2008JCLI2533.1, doi:10.1175/2008JCLI2533.1.
URL http://dx.doi.org/10.1175/2008JCLI2533.1

[8] P. M. Tagade, B.-M. Jeong, H.-L. Choi, A Gaussian process emulator approach for rapid contaminant characterization with an integrated multizone-CFD model, Building and Environment 70 (2013) 232 – 244. doi:http://dx.doi.org/10.1016/j.buildenv.2013.08.023.
URL http://www.sciencedirect.com/science/article/pii/S0360132313002497

[9] L. A. Lee, K. J. Pringle, C. L. Reddington, G. W. Mann, P. Stier, D. V. Spracklen, J. R. Pierce, K. S. Carslaw, The magnitude and causes of uncertainty in global model simulations of cloud condensation nuclei, Atmospheric Chemistry and Physics 13 (17) (2013) 8879–8914. doi:10.5194/acp-13-8879-2013.
URL http://www.atmos-chem-phys.net/13/8879/2013/

[10] M. C. Kennedy, A. O'Hagan, Bayesian calibration of computer models, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63 (3) (2001) 425–464. doi:10.1111/1467-9868.00294.
URL http://dx.doi.org/10.1111/1467-9868.00294

[11] J. McFarland, S. Mahadevan, V. Romero, L. Swiler, Calibration and uncertainty analysis for computer simulations with multivariate output, AIAA journal 46 (5) (2008) 1253–1265.

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

[13] H. Wackernagel, Multivariate Geostatistics: An Introduction with Applications, Springer, Berlin, 1995.
URL http://link.springer.com/book/10.1007%2F978-3-662-05294-5

[14] Bledar Konomi and Georgios Karagiannis and Avik Sarkar and Xin Sun and Guang Lin, Bayesian treed multivariate Gaussian process with adaptive design: Application to a carbon capture unit, Technometrics 56 (2) (2014) 145–158. arXiv:http://dx.doi.org/10.1080/00401706.2013.879078, doi:10.1080/00401706.2013.879078.
URL http://dx.doi.org/10.1080/00401706.2013.879078

[15] T. E. Fricker, J. E. Oakley, N. M. Urban, Multivariate Gaussian process emulators with nonseparable covariance structures, Technometrics 55 (1) (2013) 47–56. arXiv:http://dx.doi.org/10.1080/00401706.2012.715835, doi:10.1080/00401706.2012.715835.
URL http://dx.doi.org/10.1080/00401706.2012.715835

[16] J. Rougier, Efficient emulators for multivariate deterministic functions, Journal of Computational and Graphical Statistics 17 (4) (2008) 827–843. arXiv:http://dx.doi.org/10.1198/106186008X384032, doi:10.1198/106186008X384032.
URL http://dx.doi.org/10.1198/106186008X384032

[17] D. Higdon, J. Gattike, B. Williams, M. Rightley, Computer model calibration using high-dimensional output, Journal of the American Statistical Association 103 (482) (2008) 570–583.
URL http://www.jstor.org/stable/27640080

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

[19] W. Xing, A. A. Shah, P. B. Nair, Reduced dimensional Gaussian process emulators of parametrized partial differential equations based on Isomap, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 471 (2174). arXiv:http://rspa.royalsocietypublishing.org/content/471/2174/20140697.full.pdf,

doi:10.1098/rspa.2014.0697.
URL http://rspa.royalsocietypublishing.org/content/471/2174/20140697

[20] T. Bui-Thanh, K. Willcox, O. Ghattas, Model reduction for large-scale systems with high-dimensional parametric input space, SIAM Journal on Scientific Computing 30 (6) (2008) 3270–3288. arXiv:http://dx.doi.org/10.1137/070694855, doi:10.1137/070694855.
URL http://dx.doi.org/10.1137/070694855

[21] S. Deparis, G. Rozza, Reduced basis method for multi-parameter-dependent steady Navier-Stokes equations: Applications to natural convection in a cavity, Journal of Computational Physics 228 (12) (2009) 4359 – 4378. doi:http://dx.doi.org/10.1016/j.jcp.2009.03.008.
URL http://www.sciencedirect.com/science/article/pii/S0021999109001284

[22] M. D. Gunzburger, J. S. Peterson, J. N. Shadid, Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data, Computer Methods in Applied Mechanics and Engineering 196 (46) (2007) 1030 – 1047. doi:http://dx.doi.org/10.1016/j.cma.2006.08.004.
URL http://www.sciencedirect.com/science/article/pii/S0045782506002337

[23] D. J. Knezevic, N.-C. Nguyen, A. T. Patera, Reduced basis approximation and a posteriori error estimation for the parametrized unsteady Boussinesq equations, Mathematical Models and Methods in Applied Sciences 21 (07) (2011) 1415–1442.

[24] N. Nguyen, A posteriori error estimation and basis adaptivity for reduced-basis approximation of nonaffine-parametrized linear elliptic partial differential equations, Journal of Computational Physics 227 (2) (2007) 983 – 1006. doi:http://dx.doi.org/10.1016/j.jcp.2007.08.031.
URL http://www.sciencedirect.com/science/article/pii/S0021999107003749

[25] A. Quarteroni, G. Rozza, A. Manzoni, Certified reduced basis approximation for parametrized partial differential equations and applications, Journal of Mathematics in Industry 1 (1) (2011) 1–49. doi:10.1186/2190-5983-1-3.
URL http://dx.doi.org/10.1186/2190-5983-1-3

[26] F. Ballarin, A. Manzoni, A. Quarteroni, G. Rozza, Supremizer stabilization of POD-Galerkin approximation of parametrized steady incompressible Navier-Stokes equations, International Journal for Numerical Methods in Engineering 102 (5) (2015) 1136–1161. doi:10.1002/nme.4772.
URL http://dx.doi.org/10.1002/nme.4772

[27] J. B. Tenenbaum, Mapping a manifold of perceptual observations, in: M. I. Jordan, M. J. Kearns, S. A. Solla (Eds.), Advances in Neural Information Processing Systems 10, MIT Press, 1998, pp. 682–688.
URL http://papers.nips.cc/paper/1332-mapping-a-manifold-of-perceptual-observations.pdf

[28] T. Lin, H. Zha, Riemannian manifold learning, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (5) (2008) 796–809. doi:10.1109/TPAMI.2007.70735.

[29] C. J. C. Burges, Dimension Reduction: A Guided Tour, Foundations and Trends in Machine Learning 2 (4) (2010) 275–365. doi:10.1561/2200000002.
URL http://dx.doi.org/10.1561/2200000002

[30] J. Ham, D. D. Lee, S. Mika, B. Schölkopf, A kernel view of the dimensionality reduction of manifolds, in: Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04, ACM, New York, NY, USA, 2004, pp. 369–376. doi:10.1145/1015330.1015417.
URL http://doi.acm.org/10.1145/1015330.1015417

[31] L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, D. D. Lee, Spectral

methods for dimensionality reduction, Semisupervised learning (2006) 293–308.

[32] Y. Bengio, O. Delalleau, N. Le Roux, J.-F. Paiement, P. Vincent, M. Ouimet, Spectral Dimensionality Reduction, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 519–550. doi:10.1007/978-3-540-35488-8_28.
URL http://dx.doi.org/10.1007/978-3-540-35488-8_28

[33] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319. doi:10.1162/089976698300017467.
URL http://dx.doi.org/10.1162/089976698300017467

[34] D. Donoho, C. Chui, R. R. Coifman, S. Lafon, Special Issue: Diffusion Maps and Wavelets Diffusion maps, Applied and Computational Harmonic Analysis 21 (1) (2006) 5 – 30. doi:http://dx.doi.org/10.1016/j.acha.2006.04.006.
URL http://www.sciencedirect.com/science/article/pii/S1063520306000546

[35] P. Etyngier, F. Ségonne, R. Keriven, Shape priors using manifold learning techniques, in: IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007, 2007, pp. 1–8. doi:10.1109/ICCV.2007.4409040.
URL http://dx.doi.org/10.1109/ICCV.2007.4409040

[36] I. Jolliffe, Principal Component Analysis, Springer Series in Statistics, Springer, 2002.
URL http://www.springer.com/us/book/9780387954424

[37] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, S. W. Zucker, Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps, Proceedings of the National

Academy of Sciences of the United States of America 102 (21) (2005) 7426–7431. arXiv:http://www.pnas.org/content/102/21/7426.full.pdf, doi:10.1073/pnas.0500334102.
URL http://www.pnas.org/content/102/21/7426.abstract

[38] F. R. Chung, Spectral Graph Theory, Vol. 92, American Mathematical Soc., 1997.
URL http://bookstore.ams.org/cbms-92

[39] R. Bellman, Introduction to Matrix Analysis, Second Edition.

[40] B. Nadler, S. Lafon, R. R. Coifman, I. G. Kevrekidis, Diffusion maps, spectral clustering and eigenfunctions of Fokker–Planck operators, in: Y. Weiss, B. Schölkopf, J. Platt (Eds.), in Advances in Neural Information Processing Systems 18, MIT Press, Cambridge, MA, 2005, pp. 955–962.

[41] B. Nadler, S. Lafon, R. Coifman, I. Kevrekidis, Diffusion maps, spectral clustering and reaction coordinates of dynamical systems, Appl. Comput. Harmon. Anal. 21 (1) (2006) 113 – 127. doi:http://dx.doi.org/10.1016/j.acha.2005.07.004.
URL http://www.sciencedirect.com/science/article/pii/S1063520306000534

[42] D. Kushnir, A. Haddad, R. R. Coifman, Anisotropic diffusion on sub-manifolds with application to earth structure classification, Applied and Computational Harmonic Analysis 32 (2) (2012) 280 – 294. doi:http://dx.doi.org/10.1016/j.acha.2011.06.002.
URL http://www.sciencedirect.com/science/article/pii/S1063520311000777

[43] R. Talmon, R. R. Coifman, Empirical intrinsic geometry for nonlinear modeling and time series filtering, Proceedings of the National Academy of Sciences 110 (31) (2013) 12535–12540. arXiv:http://www.pnas.org/content/110/31/12535.full.pdf, doi:10.1073/pnas.1307298110.
URL http://www.pnas.org/content/110/31/12535.abstract

[44] I. Andrianakis, P. G. Challenor, The effect of the nugget on Gaussian process emulators of computer models, Computational Statistics & Data Analysis 56 (12) (2012) 4215 – 4228. doi:http://dx.doi.org/10.1016/j.csda.2012.04.020.
URL http://www.sciencedirect.com/science/article/pii/S0167947312001879

[45] C. E. Rasmussen, C. K. I. Williams, Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning), The MIT Press, 2005.

[46] L. S. Bastos, A. O'Hagan, Diagnostics for Gaussian process emulators, Technometrics 51 (4) (2009) 425–438. arXiv:http://dx.doi.org/10.1198/TECH.2009.08019, doi:10.1198/TECH.2009.08019.
URL http://dx.doi.org/10.1198/TECH.2009.08019

[47] C. Bishop, Pattern Recognition and Machine Learning, Information Science and Statistics, Springer-Verlag New York, 2006.

[48] A. E. Gelfand, A. M. Schmidt, S. Banerjee, C. F. Sirmans, Nonstationary multivariate process modeling through spatially varying coregionalization, Test 13 (2) (2004) 263–312. doi:10.1007/BF02595775.
URL http://dx.doi.org/10.1007/BF02595775

[49] F. D. Foresee, M. T. Hagan, Gauss–Newton approximation to Bayesian learning, in: International Conference on Neural Networks, 1997, Vol. 3, 1997, pp. 1930–1935 vol.3. doi:10.1109/ICNN.1997.614194.

[50] J. Moody, Prediction Risk and Architecture Selection for Neural Networks, Springer Berlin Heidelberg, Berlin, Heidelberg, 1994, pp. 147–165. doi:10.1007/978-3-642-79119-2_7.
URL http://dx.doi.org/10.1007/978-3-642-79119-2_7

[51] P. Arias, G. Randall, G. Sapiro, Connecting the out-of-sample and pre-image problems in kernel methods, in: 2007 IEEE Conference on Computer

Vision and Pattern Recognition, 2007, pp. 1–8. doi:10.1109/CVPR.2007.383038.

[52] J. T. Y. Kwok, I. W. H. Tsang, The pre-image problem in kernel methods, IEEE Transactions on Neural Networks 15 (6) (2004) 1517–1525. doi:10.1109/TNN.2004.837781.

[53] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, G. Rätsch, Kernel PCA and De-noising in feature spaces, in: Advances in Neural Information Processing Systems 11, Max-Planck-Gesellschaft, MIT Press, Cambridge, MA, USA, 1999, pp. 536–542.

[54] N. Thorstensen, F. Segonne, R. Keriven, Pre-image as Karcher Mean Using Diffusion Maps: Application to Shape and Image Denoising, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 721–732. doi:10.1007/978-3-642-02256-2_60.
URL http://dx.doi.org/10.1007/978-3-642-02256-2_60

[55] X. Ma, N. Zabaras, Kernel principal component analysis for stochastic input model generation, Journal of Computational Physics 230 (19) (2011) 7311 – 7331. doi:http://dx.doi.org/10.1016/j.jcp.2011.05.037.
URL http://www.sciencedirect.com/science/article/pii/S0021999111003494

[56] B. Ganapathysubramanian, N. Zabaras, A non-linear dimension reduction methodology for generating data-driven stochastic input models, Journal of Computational Physics 227 (13) (2008) 6612 – 6637. doi:http://dx.doi.org/10.1016/j.jcp.2008.03.023.
URL http://www.sciencedirect.com/science/article/pii/S0021999108001721

[57] E. A. Nadaraya, On estimating regression, Theory of Probability & Its Applications 9 (1) (1964) 141–142. arXiv:http://dx.doi.org/10.1137/1109020, doi:10.1137/1109020.
URL http://dx.doi.org/10.1137/1109020

[58] C. K. Williams, On a connection between kernel PCA and metric multidimensional scaling, Machine Learning 46 (1) (2002) 11–19. doi:10.1023/A:1012485807823.
URL http://dx.doi.org/10.1023/A:1012485807823

[59] Y. Rathi, S. Dambreville, A. Tannenbaum, Statistical shape analysis using kernel PCA, Vol. 6064, 2006, pp. 60641B–60641B–8. doi:10.1117/12.641417.
URL http://dx.doi.org/10.1117/12.641417

[60] I. Sobol, Uniformly distributed sequences with an additional uniform property, USSR Computational Mathematics and Mathematical Physics 16 (5) (1976) 236 – 242. doi:http://dx.doi.org/10.1016/0041-5553(76)90154-3.
URL http://www.sciencedirect.com/science/article/pii/0041555376901543

[61] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, S. Tarantola, Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index, Computer Physics Communications 181 (2) (2010) 259 – 270. doi:http://dx.doi.org/10.1016/j.cpc.2009.09.018.
URL http://www.sciencedirect.com/science/article/pii/S0010465509003087

[62] M. Hossain, M. Wilson, Natural convection flow in a fluid–saturated porous medium enclosed by non-isothermal walls with heat generation, International Journal of Thermal Sciences 41 (5) (2002) 447 – 454. doi:http://dx.doi.org/10.1016/S1290-0729(02)01337-6.
URL http://www.sciencedirect.com/science/article/pii/S1290072902013376

[63] COMSOL Multiphysics 5.0, Free Convection in Porous Media, last accessed 29 January 2016.
URL http://www.comsol.com/model/download/285651/models.ssf.convection_porous_medium.pdf

[64] B. Seibold, A compact and fast Matlab code solving the incompressible Navier-Stokes equations on rectangular domains, last accessed 29 January 2016.
URL http://math.mit.edu/~gs/cse/

[65] J. Newman, K. Thomas-Alyea, Electrochemical Systems, Third Edition, John Wiley & Sons, Hoboken, 2004.

[66] K. Broka, P. Ekdunge, Modelling the PEM fuel cell cathode, Journal of Applied Electrochemistry 27 (3) (1997) 281–289. doi:10.1023/A:1018476612810.
URL http://dx.doi.org/10.1023/A:1018476612810

[67] R. B. Bird, W. E. Stewart, E. N. Lightfoot, Transport Phenomena, Revised 2nd Edition, John Wiley & Sons, Inc., 2006.

[68] COMSOL Multiphysics 5.0, Species Transport in the Gas Diffusion Layers of a PEM, last accessed 29 January 2016.
URL https://www.comsol.com/model/download/267211/models.bfc.pem_gdl_species_transport_2d.pdf

[69] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326. arXiv:http://science.sciencemag.org/content/290/5500/2323.full.pdf, doi:10.1126/science.290.5500.2323.
URL http://science.sciencemag.org/content/290/5500/2323

[70] Z. Zhang, H. Zha, Principal manifolds and nonlinear dimensionality reduction via tangent space alignment, SIAM Journal on Scientific Computing 26 (1) (2004) 313–338. arXiv:http://dx.doi.org/10.1137/S1064827502419154, doi:10.1137/S1064827502419154.
URL http://dx.doi.org/10.1137/S1064827502419154

[71] U. von Luxburg, O. Bousquet, M. Belkin, On the Convergence of Spectral Clustering on Random Samples: The Normalized Case, Springer Berlin

Heidelberg, Berlin, Heidelberg, 2004, pp. 457–471. doi:10.1007/978-3-540-27819-1_32.

URL http://dx.doi.org/10.1007/978-3-540-27819-1_32