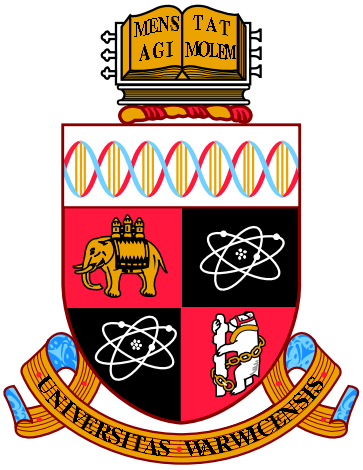

Manifold learning for emulations of computer models

by

Wei Xing

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

School of Engineering

September 2016

THE UNIVERSITY OF WARWICK

Contents

List of Figures

Acknowledgments

First and foremost I would like to express my utmost appreciation to my supervisor **Akeel Shah**. Without his support, academically and emotionally, I would not be writing this thesis. Thanks to his insight and guidance, I had the chance to learn many popular techniques and to make my modest contributions to the literature. Throughout, he supported me not only through his endless supply of ideas but also by providing beautiful mathematical insights into the research. He gave me the courage to get through the rocky road from rough ideas to polished publications. He taught me determination and the ability to grow as an independent researcher. He is also a mentor and a friend in life. There is so much I would like to thank him for but the list would go far beyond the scope of Ph.D. supervision.

I would like to thank my colleague and fellow student Vasilis Triantafyllidis. He is a very good friend and a great partner in research. Many thanks to him for helpful discussions and the work he contributed to my research.

I would also like to thank Prof. Nicholas Zabaras for some very interesting discussions, encouragement and help in drafting paper 3.

I would like to thank the School of Engineering and the China Scholarship Council for a doctoral scholarship that made my Ph.D. study possible.

I would also like to thank all of my friends, especially for their insight and for sharing their joy and knowledge with me. They made my life more than just about logic and maths.

Many thanks to my parents and brother who supported me in the pursuit of my dream of academia, both financially and emotionally.

Doing a Ph.D. overseas alone might seem difficult and depressing in many ways. However, it became the best time that I have ever had in my life largely because of my supervisor. So, lastly, I would like to thank my supervisor again.

Declarations

The refereed publications arising directly from this thesis are as follows:

1. A. A. Shah, W. W. Xing, V. Triantafyllidis. Reduced-order modelling of parameter-dependent, linear and nonlinear dynamic partial differential equation models. *Proc. R. Soc.* A 2017 473 20160809; DOI: 10.1098/ rspa.2016.0809. Published 26 April 2017

2. W. Xing, A. A. Shah, and P. B. Nair,"Reduced dimensional gaussian process emulators of parameterized partial differential equations based on Isomap, *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 471, p. 20140697, 2015.

3. W.W. Xing, V. Triantafyllidis, A.A Shah, P.B Nair, and N. Zabaras, "Manifold learning for the emulation of spatial fields from computational models", *Journal of Computational Physics*, vol. 326, pp. 666–690, 2016.

4. V. Triantafyllidis, W. Xing, A.A Shah, and P.B. Nair, "Neural network emulation of spatio-temporal data using linear and nonlinear dimensionality reduction", in *Advanced Computer and Communication Engineering Technology, Lecture Notes in Electrical Engineering*, pp. 1015–1029, Springer, 2016.

In the first three papers, the basic idea was provided by my supervisor, while the coding was done entirely by me, with the exception of the neural network and support vector machine methods. The extensions to local tangent space alignment and local linear embedding were entirely my idea. The idea for the pre-image for

kPCA and the general framework for the pre-image was mine. The analysis was conducted by my supervisor for the diffusion map pre-image solution in paper 2. I wrote the first drafts of both papers.

In paper 3, the initial idea was provided by my supervisor while the codes were adapted by V. Triantafyllidis from my codes on GPE and manifold learning and I conducted the simulations together with V. Triantafyllidis. The paper was written by my supervisor with a first draft from V. Triantafyllidis.

The emulation codes for paper 4 were mine, while the initial idea was again from my supervisor. The numerical methods for the PDEs were a joint effort with V. Triantafyllidis led by me (we worked independently on two methods for the concentration problem) before deciding to V. Triantafyllidis's finite-volume code in the paper over my finite-difference code (both were similar in accuracy). The finite element code was mine. The first draft of the paper was written by me and V. Triantafyllidis (the manifold learning sections by me and the POD/Galerkin sections by V. Triantafyllidis, with the results section and generation of results split between us), before being edited by my supervisor.

Abstract

Computer simulations are widely used in scientific research and engineering areas. Thought they could provide accurate result, the computational expense is normally high and thus hinder their applications to problems, where repeated evaluations are required, e.g, design optimization and uncertainty quantification. 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 thesis, manifold learning is applied to construct GP emulators for very high-dimensional output spaces arising from parameterised PDE model simulations. Artificial neural network (ANN) support vector machine (SVM) emulators using manifold learning are also studied. A general framework for the inverse map approximation and a new efficient method for diffusion maps were developed. The manifold learning based emulators are then to extend reduced order models (ROMs) based on proper orthogonal decomposition to dynamic, parameterized PDEs. A similar approach is used to extend the discrete empirical interpolation method (DEIM) to ROMs for nonlinear, parameterized dy-

namic PDEs.

Chapter 1

Introduction and literature review

Modelling and simulation play important roles in modern science and engineering. Mathematical models are typically based on conservation laws, expressed as ordinary or partial differential equations (ODEs or PDEs), together with constitutive relations and boundary/initial conditions. Such models can be implemented on a computer using a variety of numerical methods, e.g., the finite element or finite volume methods in the case of spatially-distributed systems, together with a time-stepping scheme for dynamic models. The resulting discretized (finite-dimensional) system is often referred to as a *computer model*, *simulator* or *high fidelity model*.

Simulations can often be used to estimate properties or provide insight into the behaviour of a physical system when such properties or behaviour cannot be studied or measured experimentally (e.g., *ab-initio* simulations). They can also dramatically reduce the timescales and costs associated with analysis, design and testing. Simulators, however, can be highly computationally expensive. For applications in which repeated calls to a simulator are required, e.g., design optimization, uncertainty quantification and sensitivity analysis, direct implementation of the simulator can be computationally infeasible, even when the computational time is only

moderately long [2, 3]. For example, 10,000 Monte Carlo runs of a simulator that takes 30 min per run would require more than 208 days of run time without a proper parallelisation.

In order to overcome this problem, it is common practice to employ a *surrogate model* (or simply 'surrogate'), also referred as an *emulator* or a meta model, in place of the simulator. These terms would be used interchangeably throughout this thesis. The surrogate approximates the simulator, with the advantage that it can be evaluated very cheaply, ideally in real-time [2–8].

Depending on the application, the target of interest and the form of the simulator, e.g., how the spatial grid is constructed, whether the simulator is dynamic or steady state, or whether stochastic or deterministic components (in some aspects of the specifications) are involved, surrogates can be constructed in different ways. Broadly speaking, there are three classes of surrogate models [9]. *Hierarchical* or *multi-fidelity* approaches construct surrogates from the original models *via* simplifying assumptions, such as spatial uniformity, using coarse or successively refined grids in the numerical implementations, or by relaxing error tolerances.

The most common approaches are *data-driven* (or *statistical* or *black-box*), which implement machine-learning techniques, e.g., polynomial response surfaces, artificial neural networks (ANN), Gaussian process regression or emulation (GPR or GPE) and support vector machines (SVM), to learn the mapping between the inputs and outputs of the high-fidelity model based on a few selected runs at judiciously selected design points [3].

A *model order reduction* approach can also yield a surrogate model. Such approaches directly reduce the size of the original model or the numerical formulation by projecting the system onto a space of lower dimension. The resulting *reduced order model* (ROM) can be thought of as an surrogate, although typically the terms 'emulator' and 'surrogate' are avoided and such models are simply referred to as ROMs.

In this thesis, data-driven emulators and ROMs would be the focus. From an algorithmic viewpoint, both involve two stages:

1. An *offline stage* is where a simulator is explored to provide knowledge of the system and to build a surrogate model. For data-driven methods, this stage involves evaluating the simulator at carefully chosen input parameter values ('design points') to generate solutions that are used as 'training' data, i.e. data to approximate ('learn') the mapping between the inputs and the simulator output of interest. For the ROM approaches, the solutions are used to find a low-dimensional subspace that approximates the space in which the simulator outputs lie. A ROM is then constructed by projecting the original system (the high fidelity model) onto the approximating subspace. Selection of the design points for both approaches is a process known as *design-of-experiment* (DOE).

2. An *online stage* in which the emulator is used as an alternative to the simulator, i.e., to provide predictions corresponding to new (*test* or *query*) inputs. This stage is typically very rapid.

In the next section, a literature review is provided for both types of emulators considered in this thesis, with the aim of outlining the current challenges with respect to the main problem under consideration in this thesis:

The emulation of parameterized spatial and spatio-temporal PDE models (involving high-dimensional output spaces).

The contributions of this thesis towards data-driven emulators and ROMs, in relation to the main problem above, are made clear in both cases.

1.1 Data driven surrogates for outputs in high dimensional spaces

A simulator can be represented as a mapping $\boldsymbol{\eta} : \mathcal{X} \to \mathbb{R}^d$, taking inputs $\boldsymbol{\xi} \in \mathcal{X} \subset \mathbb{R}^l$ in some space $\mathcal{X}$ and generating outputs $\boldsymbol{y} \in \mathbb{R}^d$. A data-driven (black-box) surrogate aims to approximate the function $\boldsymbol{\eta}(\cdot)$ within a supervised machine learning framework, i.e., based on so-called 'training' data in the form of inputs and corresponding outputs from the simulator at carefully chosen inputs/parameters ('design points') [2, 4, 10], Since data-driven surrogates are 'blind' to the underlying model and are only dependent on the training data and the machine learning techniques, they can be computationally cheap, are very versatile, and are able to handle nonlinearities readily to some extent when the problem is properly setted up. They are applicable to a broad range of applications, e.g., uncertainty quantification (UQ) [11–15] and sensitivity analysis as outlined in [16–19].

The machine learning method used is the most important factor in the development of a data-driven surrogate (together with the choice of design points) and should ideally be chosen according to our prior knowledge of the simulator [20, 21]. Among the many methods available (excluding simple polynomial interpolations), GPE/GPR, ANNs and SVMs are the most widely used, especially GP models since they are versatile (different covariance functions and structures, convenient analytical properties, explicit formulae in simple cases) and automatically provide a probabilistic (Bayesian) framework, making it easier to avoid overfitting. In this thesis, Gaussian process emulation (the term 'emulation' is more common in this context) is used as our main data-driven technique due to the aforementioned advantages, but ANNs and SVM regression are also implemented.

Emulation is usually limited to the approximation of a few scalar outputs as functions of input variables or parameters. These are summaries of the outputs, such as an average, a value at a specific point or time, a maximum or minimum, or

some other quantity derived from the model outputs. In practice, however, particularly when the simulator involves numerical solutions of a set of PDEs, the goal may require approximations to a spatial or spatio-temporal field as a function of the inputs/parameters since such a result provides informations across the problem domain. The solution (output) space of the simulator in this case is given by vectorized discrete field values $\boldsymbol{y} \in \mathbb{R}^d$, the coordinates of which are the field values at d locations on a spatial grid. The dimension d of the output space poses enormous challenges in terms of computational efficiency.

For even moderately coarse spatial discretizations, e.g., a $100 \times 100 \times 100$ grid in $\mathbb{R}^3$, the value of d is very large. In problems involving complex geometries or multiple spatial scales, a much finer grid may be required to adequately resolve small-scale characteristics. 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 [5] and to employ methods for scalar outputs. This approach is infeasible for more than a few spatial locations, even for parsimoniously selected training points [22]. On the other hand, with too few spatial locations, the emulator may not provide sufficient information. ANNs can of course handle multiple outputs, but are again impractical for very large values of d (the number of network weights and subsequent optimisation would be prohibitive).

In GPE, the output of a simulator is modelled as a GP indexed by the parameters [6, 23–26]. GP models were first used for emulation in [4, 10]. The GPE framework (as well as SVM regression) is not naturally extended to multiple outputs. Conti and O'Hagan [27] 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 [28]. 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 or a shock is formed, this assumption is invalid.

Recent extensions of this idea can be found in [29–31]. Fricker *et al.* [29] made use of the linear model of coregionalization (LMC), which constructs a p-variate GP using $d' \leq d$ independent univariate GPs [28, 32]. Since the number of hyperparameters increases with the value of d' in this method (the GP distributions are not identical), it is restricted to low dimensional outputs. Rougier [30] developed an outer-product emulator that takes advantage of factorizations of the covariance matrix to improve computational efficiency. The method is, on the other hand, based on a number of simplifying assumptions (including separability) in regard to the regression functions and correlation structure.

An alternative approach based on dimensionality reduction of the output space was developed by Higdon *et al.* [1], 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.* [33]. 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.

1.1.1 Thesis contributions on data-driven emulation

By combining manifold learning methods and scalar GPE, manifold-learning based Gaussian process emulators are constructed for emulating vectorized field values in very high dimensional spaces $\mathbb{R}^d$. The manifold learning is shown that it can be placed within the overall learning task. Essentially, the learning takes place

in a *feature space*, defined by the low-dimensional approximations (dependent on the method) of the physical points in $\mathbb{R}^d$. Approximate inverse maps (the map from reduced-dimensional space to physical space $\mathbb{R}^d$) are employed to yield the final predictions for new (test) inputs/parameters. Three different manifold learning techniques, namely, Isomap (in chapter 3), kernel PCA and diffusion maps (in chapter 4), are used in this thesis. Others (namely local tangent space alignment (LTSA) and local linear embedding (LLE)) are also implemented, but the results are omitted since further work is required to assess the results fully. The basic approach is then extended by implementing different machine techniques: ANN (in chapter 5) and SVM regression (in Chapter 4).

A crucial step is the inverse mapping (*pre-image*) for the manifold learning methods. For diffusion maps there are current two existing solutions. In this work, a new approach where only linear algebra is involved to construct the pre-image map (based on a continuous state space spectral analysis of certain Markov operators defined on a graph) is introduced . Unlike the existing solutions where complex optimization algorithms are involved, our method is computationally cheap and free from issues such as local optima trapping and sensitivity to initial guesses. The existing methods were developed for 2-D and 3-D shape analysis, and would be computationally infeasible for the problems considered in the thesis. Furthermore, for techniques with existing pre-image solutions, e.g., kernel PCA and Isomap, a new general framework for the pre-image map is also introduced (chapter 4).

1.2 Reduced order models

A *reduced order model* (ROM) is essentially a Galerkin projection of the simulator onto the subspace spanned by an approximating basis. In the ROM area, a simulator is often termed a *full order model* (FOM). For PDE models, the Galerkin projection can be performed on the original equations (including a weak form) or

on the spatially discretized system. The final form is an algebraic system for steady problems or an ODE system for dynamical problems. The approximating basis in ROMs can be obtained through balanced truncation [34], Krylov subspace methods [35] or proper orthogonal decomposition (POD) [36, 37]. The first two are largely restricted to linear, time-invariant (LTI) systems.

A recent survey of projection methods can be found in [38]. The most widely used technique for PDE systems is POD, in which the approximating subspace is obtained from solutions (*snapshots*) generated by the discretized full-order model (FOM) at selected time instances. Applications to dynamic, nonlinear parameterized PDEs presents a number challenges: (i) constructing a basis that is valid across parameter space; (ii) dealing with high dimensional parameter spaces; (iii) using data parsimoniously; (iv) efficiently computing the reduced-order system matrices and reduced-order nonlinearities (in the state variable) during use of the surrogate (online).

There are several approaches to incorporating parametric dependence: (a) to use a global basis (meaning across parameter space); (b) interpolation of the local basis (meaning for a particular parameter value); and (c) interpolation of local system matrices. For linear time-invariant (LTI) systems, the system matrices often take the form of affine combinations of constant matrices with parameter-dependent coefficients. In such cases the reduced order system is quickly and easily assembled for a new parameter value [39–41]. Affine forms can also be realised using Taylor series expansions [39, 42] or by using an empirical interpolation strategy [43]. Global basis methods extract a single basis by combining local snapshot matrices to form a global matrix [39, 44–46]. Obvious drawbacks are the violation of POD optimality and the growth in the size of the global matrix with the number of samples. There are, however, efficient sampling strategies for constructing global bases, such as the greedy approach of [39] or by using a local sensitivity analysis [47].

An alternative to global basis methods is interpolation of local bases or local

reduced model matrices. Lieu *et al.* [48] used the principal angles between two POD bases for different Mach numbers in a linearized fluid-structure ROM model to linearly interpolate a local basis for intermediate Mach numbers. This method is restricted to single-parameter systems and small parameter changes. Amsallem and Farhat considered the local bases as members of a Grassmann manifold, the set of all subspaces (of a chosen low dimension) of the state space [49, 50]. The local bases are mapped to a tangent space of the Grassman manifold using a logarithmic map and Lagrange interpolation is performed in the tangent space, with an inverse exponential map providing the required local basis.

Interpolation methods can also be used to approximate the reduced-order system matrices, in order to circumvent the problem of computing these matrices for each new parameter value. Degroote *et al* [46] proposed two methods: element-wise direct spline interpolation of the reduced-order matrices or spline interpolation of the matrices in a tangent space of a Riemannian manifold on which the matrices are assumed to lie (a similar method was proposed in [51]). When a global basis is not used to build the reduced-order model, a straightforward interpolation is not possible because the reduced-dimensional coordinates do not (in general) have the same physical meaning from one local basis to another. Thus a congruency transformation to a common basis is required before direct interpolation [52] or interpolation in a tangent space [50, 53].

Lieberman *et al.* used a greedy algorithm to construct projections for both the state variable and the parameters simultaneously, minimizing a measure of the error between the ROM and FOM outputs at each iteration [54] (different error measures are considered in [55]). Hay *et al* [56, 57] used sensitivities (derivatives) of the POD basis with respect to (w.r.t.) the parameters to either linearly extrapolate the POD basis for a new parameter value or to expand the POD basis by augmenting it with the corresponding sensitivities. The growth in the basis dimension with the number of parameters is a limitation of this approach.

The computational cost of evaluating a strong (high order polynomial or non-polynomial) nonlinearity in the state variable in a ROM depends on the dimension of the original state space. Expansions of the system nonlinearity using quadratic reduction [58], bi-linearization [59], or Volterra series [60] are only applicable to weak nonlinearities or confined regions of state space. Moreover, the computational cost grows exponentially with the order of the approximating expansion. Recently, a number of *hyper-reduction* methods have been developed to overcome the limitations of linearization techniques. An early method was developed by Astrid *et al.* [61, 62], based on selecting a subset of the FOM equations corresponding to heuristically chosen spatial grid points, followed by a Galerkin projection of the resulting reduced system onto the POD basis.

The empirical interpolation (EIM) method interpolates the nonlinear function at selected spatial locations using an empirically derived basis, and is applied to directly to the governing PDE [40, 41]. A discrete version, termed the discrete empirical interpolation method (DEIM), is applicable to general ODE or algebraic systems arising from a spatial discretization [63, 64]. Both methods construct a subspace for the approximation of the nonlinear term and use a greedy algorithm to select the interpolation points. An extension to DEIM [65] generates several local subspaces using a clustering algorithm and uses classification in the online phase to select one of the local subspaces. These approaches can also be used for approximating system matrices in non-affine cases by vectorization of the matrices [38]. The Gauss-Newton with approximated tensors (GNAT) method, on the other hand, operates at the fully discrete level in space and time, and is based on satisfying consistency and discrete-optimality conditions by solving a residual-minimization problem [66]. This leads to a Petrov-Galerkin (rather than Galerkin) problem with a test basis that depends on the residual derivatives w.r.t. the state variable.

1.2.1 Thesis contributions on POD ROMs

In this thesis, an extension of POD for dynamic, parameterized, linear and nonlinear PDEs is introduced. The method developed involves a computationally efficient approximation of the basis and nonlinearity for new parameter values. It can be used in conjunction with many of the methods described above, e.g., greedy sampling and methods for approximating non-affine system matrices. In order to avoid any loss of information and inconsistency in the construction of new bases, snapshots rather than the bases or system matrices are approximated directly.

The snapshots, however, lie in high-dimensional spaces so that direct approximations are computationally infeasible. This issue is overcomed by using the manifold learning based emulation methods developed (described section 1.1.1) to approximate the snapshots for new parameter values. To handle nonlinearities, the DEIM is extended by using the same emulation approach to approximate snapshots of the nonlinearity at desired locations in parameter space. The method is then implemented for a finite volume (FV) approximation of a 2-D linear convection-diffusion problem with a stochastic velocity field as the input. An UQ analysis is also performed to illustrate the application of our ROM. A second example, of a 1-D (nonlinear) Burger's equation, is also presented to demonstrate the use of the modified ROM with a simultaneously modified DEIM. In this case, finite element (FE) formulation is considered. The results are compared to a global basis approach, and are shown to yield a higher accuracy.

1.3 Thesis outline

In chapter 2 an outline of the techniques employed in this thesis is provided. These include the regression techniques, Bayesian regularization, linear dimension reduction and manifold learning methods, pre-image maps, design-of-experiment, numerical methods for PDEs, Galerkin projections and POD.

The main problem considered for the data-driven emulation is formulated in chapter 3. Linear (PCA) approach of Higdon *et al.* [1] is introduced in section 3.2 and motivate the implementation of Isomap and the improved version kernel Isomap in section 3.3, providing full details of the methods. The approach for Isomap based GP emulation is then presented, including the pre-image solution, and show results on three different problems, with comparisons to Higdon's method.

Chapter 4 extends the work in Chapter 3 by using kernel PCA and diffusion maps (full details in sections 4.1 and 4.2) in place of Isomap. The GPE emulation approaches using these methods are then presented. The pre-image problem is placed in a new general framework in section 4.4 for kernel PCA and diffusion maps. Our new pre-image solution for diffusion maps is presented in section 4.4.2. Numerical experiments are then shown, including comparisons with ANN and SVM regression in place of GPE in our framework, as well as comparisons to Higdon's method.

An ANN emulator is implemented in chapter 5, in which AAN with Bayesian regularization replaces GPE. The results of two numerical experiments shown are in section 5.2, alongside comparisons with the GPE based emulators. In chapter 7 full details of POD-based reduced order modelling are shown, with a specific example of a nonlinear parabolic PDE. Details of the emulation of the snapshots and the extension of DEIM to parametric cases are provided. Two examples (one linear and one nonlinear) and compare the results to a global basis approach are then presented as following. Finally, conclusions are drawn in Chapter 8.

Chapter 2

Methodologies

2.1 Introduction

In this chapter, the basic concepts and methods employed throughout the thesis are introduced. It starts with methods of regression (also known as function approximation), which can be classed as supervised machine learning techniques. That is, inferring a function given data that is labelled in some way by an input. The dimensionality reduction methods are then outlined. These can be classed as unsupervised learning techniques, i.e., inferring a function or mapping or categorisation that reveals hidden structures from unlabeled data (no corresponding inputs or labels). Linear methods are initially discussed before outlining nonlinear dimensionality reduction (also called manifold learning), including methods for inverse maps (mapping dimensional representations back to the original space). A discussion on design-of-experiments (choosing inputs in order to construct the data for building an emulator) is provided. Numerical approximations for PDEs, namely the finite difference, volume and element methods, are then introduced. This chapter are closed with a description of the Galerkin method and a detailed discussion of proper orthogonal decomposition (POD). The POD is presented from several perspectives. Given the numerous interpretations and labels given to POD in the literature, the

connections between POD and other (equivalent but sometimes subtly different) methods are detailed.

2.2 Regression methods

Regression methods can range from simple least squares fitting of polynomials to advanced methods of Bayesian inference. In this section, the three methods used in this thesis are outlined. Other than polynomial surface fitting, these methods are by the far the most widely used in emulation tasks. The discussion is kept to the univariate case. Extensions to multivariate (high-dimensional outputs) presented in later chapters are based on the univariate models below.

2.2.1 Univariate Gaussian process emulation

Suppose that one wishes to approximate a scalar valued function $y = \eta(\boldsymbol{\xi}) \in \mathbb{R}$, which maps inputs $\boldsymbol{\xi} = (\xi_i, \ldots, \xi_l)^T \in \mathcal{X} \subset \mathbb{R}^l$ to outputs $y \in \mathbb{R}$ given values of the function at selected (design) points.

In GPE, a *prior distribution* assumed over this function in the form of a GP indexed by $\boldsymbol{\xi} \in \mathcal{X}$. For each fixed $\boldsymbol{\xi}$, $\eta(\boldsymbol{\xi})$ is a random variable. A collection of values of $\eta(\boldsymbol{\xi})$ at different values of $\boldsymbol{\xi}$, on the other hand, is a partial realization (a deterministic function of $\boldsymbol{\xi}$) of the GP. The probability distribution of partial realizations can be expressed as follows: The joint distribution of $(\eta(\boldsymbol{\xi}_1), \ldots, \eta(\boldsymbol{\xi}_N))$ for an arbitrary finite collection $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N \in \mathcal{X}$ is multivariate Gaussian. A GP is denoted by $\mathcal{GP}\left(m(\boldsymbol{\xi}), c(\boldsymbol{\xi}, \boldsymbol{\xi}')\right)$, where the first and second arguments are the mean function and covariance function:

$$\begin{aligned} m(\boldsymbol{\xi}) &= \mathbb{E}[\eta(\boldsymbol{\xi})] \\ \mathfrak{c}(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}) &= \mathbb{C}\mathrm{ov}\left(\eta(\boldsymbol{\xi}), \eta(\boldsymbol{\xi}')\right) = \mathbb{E}[(\eta(\boldsymbol{\xi}) - m(\boldsymbol{\xi}))(\eta(\boldsymbol{\xi}') - m(\boldsymbol{\xi}'))], \end{aligned} \tag{2.1}$$

respectively. Here $\boldsymbol{\theta}$ is a vector of *hyperparameters* that appear in the covariance

function, i.e, in the specification of the probabilistic model. They are typically unknown *a priori* and must be estimated as part of the GPE framework. The most common choice for the mean function is a constant value (linear functions of $\boldsymbol{\xi}$ are also frequently used). In this work, a constant zero value was assumed by centring the data, that is $m(\boldsymbol{\xi}) = 0$ for all $\boldsymbol{\xi}$. It is worth to point out that this is a prior belief that is updated using the data to obtain a new 'posterior GP distribution', with updated mean and covariance functions. The data centring makes this assumption reasonable.

GPE is performed using training points $y^{(i)} = \eta(\boldsymbol{\xi}^{(i)})$ corresponding to design points $\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, m$. a vector of targets is defined as:

$$\boldsymbol{t} = (y^{(1)}), \ldots, y^{(m)}))^T. \tag{2.2}$$

The GP prior over $\eta(\boldsymbol{\xi})$ is:

$$\eta(\boldsymbol{\xi})|\boldsymbol{\theta} \sim \mathcal{GP}\left(0, \mathbb{c}(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta})\right), \tag{2.3}$$

where the notation indicates that $\eta(\boldsymbol{\xi})$ is distributed according to a GP with an identically zero mean function and a covariance function $\mathbb{c}(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta})$ *given* the values of $\boldsymbol{\theta}$. The GP prior and the forms of the covariance and mean functions specify a *distribution over functions* to which our desired mapping $\eta(\boldsymbol{\xi})$ belongs. A variety of covariance functions have been employed for GPE, including the Matérn class, piecewise polynomials, and the squared exponential, all of which define weak stationary processes. In this work, a square exponential covariance function is utilised, which is the most common form:

$$\mathbb{c}(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}) = \theta_0 \exp\left(-(\boldsymbol{\xi} - \boldsymbol{\xi}')^T \mathrm{diag}(\theta_1, \ldots, \theta_l)(\boldsymbol{\xi} - \boldsymbol{\xi}')\right) \tag{2.4}$$

, where $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_l)^T$. The parameters $\theta_1, \ldots, \theta_l$ are the inverse square correlation

lengths. Use of this covariance function expresses the belief that the realizations of the GP are continuous functions of $\boldsymbol{\xi}$ in a mean square sense.

GPE is also a Bayesian approach; the prior distribution (2.3) encodes the model and the prior beliefs about $\eta(\boldsymbol{\xi})$ (types of functions to which it belongs). The given data $\boldsymbol{t}$ is used to update the model in the form of another GP that represents an improved prediction of $\eta(\boldsymbol{\xi})$. Thus, a point estimate of $\eta(\boldsymbol{\xi})$ but rather distributions at each value of $\boldsymbol{\xi}$ over values that $\eta(\boldsymbol{\xi})$ may take, or equivalently a distribution over functions of $\boldsymbol{\xi}$ is obtained. The updated mean function yields $\mathbb{E}[\eta(\boldsymbol{\xi})]$ at each value of $\boldsymbol{\xi}$, while the covariance function provides predictive variances in this estimate. Rather than integrating over the space of functions, which requires a definition for a probability measure in the space of functions, the procedure in GPE is simplified by the standard conditioning properties of Gaussian distributions.

The likelihood $p(\boldsymbol{t}|\boldsymbol{\theta})$ is the probability distribution of the data given the hyperparameters, which by the properties of a GP is distributed as a multivariate Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{C}(\boldsymbol{\theta}))$ with mean $\mathbf{0}$ and *covariance matrix*:

$$\mathbf{C}(\boldsymbol{\theta}) = [C_{ij}], \quad \text{where} \quad C_{ij} = \mathbb{c}(\boldsymbol{\xi}^{(i)}, \boldsymbol{\xi}^{(j)}; \boldsymbol{\theta}), \quad i, j = 1, \ldots, m \tag{2.5}$$

The joint distribution $p(\eta(\boldsymbol{\xi}), \boldsymbol{t}|\boldsymbol{\theta})$ for $\boldsymbol{\xi} \in \mathcal{X}$ is likewise distributed according to $\mathcal{N}(\mathbf{0}, \mathbf{C}(\boldsymbol{\theta})')$, where:

$$\mathbf{C}'(\boldsymbol{\theta}) = \left[\begin{array}{c|c} \mathbf{C}(\boldsymbol{\theta}) & \boldsymbol{c}(\boldsymbol{\xi}) \\ \hline \boldsymbol{c}(\boldsymbol{\xi})^T & c(\boldsymbol{\xi}, \boldsymbol{\xi}; \boldsymbol{\theta}) \end{array}\right], \tag{2.6}$$

in which:

$$\boldsymbol{c}(\boldsymbol{\xi}) = (\mathbb{c}(\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}; \boldsymbol{\theta}), \ldots, \mathbb{c}(\boldsymbol{\xi}^{(m)}, \boldsymbol{\xi}; \boldsymbol{\theta}))^T, \tag{2.7}$$

The *conditional predictive distribution* at new inputs $\boldsymbol{\xi} \in \mathcal{X}$ is obtained from the

joint distribution $p(\eta(\boldsymbol{\xi}), \boldsymbol{t}|\boldsymbol{\theta})$ by conditioning on $\boldsymbol{t}$ to yield the updated GP:

$$\begin{aligned} \eta(\boldsymbol{\xi})|\boldsymbol{t}, \boldsymbol{\theta} &\sim \mathcal{GP}\left(m'(\boldsymbol{\xi};\boldsymbol{\theta}), c'(\boldsymbol{\xi}, \boldsymbol{\xi}';\boldsymbol{\theta})\right), \\ m'(\boldsymbol{\xi};\boldsymbol{\theta}) &= \boldsymbol{c}(\boldsymbol{\xi})^T \mathbf{C}(\boldsymbol{\theta})^{-1} \boldsymbol{t}, \\ c'(\boldsymbol{\xi}, \boldsymbol{\xi}';\boldsymbol{\theta}) &= c(\boldsymbol{\xi}, \boldsymbol{\xi}') - \boldsymbol{c}(\boldsymbol{\xi})^T \mathbf{C}(\boldsymbol{\theta})^{-1} \boldsymbol{c}(\boldsymbol{\xi}'). \end{aligned} \tag{2.8}$$

The expected value $\mathbb{E}[\eta(\boldsymbol{\xi})]$ of $\eta(\boldsymbol{\xi})$ at an arbitrary input is given by $m'(\boldsymbol{\xi};\boldsymbol{\theta})$ while the predictive variance $\mathbb{V}\text{ar}(\eta(\boldsymbol{\xi}))$ in the prediction is given by $c'(\boldsymbol{\xi}, \boldsymbol{\xi};\boldsymbol{\theta})$.

The hyperparameters $\boldsymbol{\theta}$ are, however, unknown. Point estimates [5, 67] such as the maximum likelihood estimate (MLE) are typically employed; that is, the predictive distribution is given by equation (2.8) using the MLE estimate. The MLE is obtained by assuming a uniform distribution for the hyperparameters and using Baye's rule to obtain $p(\boldsymbol{\theta}|\boldsymbol{t}) \propto p(\boldsymbol{t}|\boldsymbol{\theta})$, where the latter is the known likelihood. Thus, maximizing the (log of the) likelihood gives the the most likely value of $\boldsymbol{\theta}$:

$$\begin{aligned} \boldsymbol{\theta}_{MLE} &= \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{t}|\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \left(-\frac{1}{2} \ln |\mathbf{C}(\boldsymbol{\theta})| - \frac{1}{2} \boldsymbol{t}^T \mathbf{C}(\boldsymbol{\theta})^{-1} \boldsymbol{t} \right). \end{aligned} \tag{2.9}$$

A fully Bayesian inference requires integrating over $\boldsymbol{\theta}$ in the joint distribution of $\boldsymbol{\theta}$ and $\eta(\boldsymbol{\theta})$. The integration is analytically intractable but can be approximated using Monte Carlo (MC) integration, e.g., importance sampling, or a Markov Chain Monte Carlo (MCMC) [20] method to sample from the posterior over the hyperparameters $p(\boldsymbol{\theta}|\boldsymbol{t})$. This method is computationally intensive and does not always lead to improved results. In this thesis, MLE estimates is implemented.

2.2.2 Bayesian regularisation neural networks

Artificial neural networks (ANNs) are widely employed for a vast number of learning problems, due to their flexibility and accuracy ('universal approximators'). Recent developments in the form of deep learning networks [68, 69] have also revitalised

the area. ANNs use a fixed number of basis functions that can be adapted to different datasets to approximate an unknown mapping from inputs to outputs [20]. There are many types of ANN model, e.g., feed-forward neural networks, recurrent networks, polynomial networks and modular networks.

In this thesis, a feed-forward neural network, also known as multiplayer perceptron (MLP), is used as a regression model (a data-driven emulator) for multivariate emulation. The method is briefly introduced following Bishop *et al.* [20]. The basic idea of ANNs is to create a network of connected "neurons" that take inputs from a specified subset of the neurons and return outputs that are in turn used as inputs for another subset of neurons. They are, essentially, a complex expansion of a function in terms of a basis that depends on functions associated with the neurons and the number of neurons, as well as the way in which the neurons interact.

In MLPs, the output (activation) is defined as follows:

$$a_j^{(k+1)} = h\left(\sum_{i=1}^{N_k} w_{ji}^{(k)} a_i^{(k)} + w_{j0}^{(k)}\right), \tag{2.10}$$

where $h(\cdot)$ is an 'activation function' $a_i^{(k)}$ indicates the i-th output in the k-th layer, N_k is the total number of activation functions in the k-th layer, and $w_{ji}^{(k)}$ is the weight (or parameter) associated with $a_i^{(k)}$, connecting it to neuron j in layer k. The most commonly chosen activation function is the sigmoid function $h(x) = 1/(1+e^{-x})$. Other popular choices includes the hyperbolic tangent $h(x) = (e^x - e^{-x})/(e^x + e^{-x})$ and rectified linear function $h(x) = \max(0, x)$.

Eq. (2.10) corresponds to a *feedforward network*, in which the inputs to layer k are outputs of neurons from layers that strictly precede layer k. The number of neurons in a layer N_k and the number of layers n decide the complexity of a MLP. The inputs $\{\xi_j\}$ are incorporated by setting $a_j^{(0)} = \xi_j$ for all $j = 1, \ldots, N_0 = l$. This

gives the univariate output of a $n+1$ layer MLP as:

$$o(\boldsymbol{\xi}) = a_1^{(n+1)} = h\left(\sum_{i=1}^{N_n} w_{1i}^{(n)} a_i^{(n)} + w_{10}^{(n)}\right), \tag{2.11}$$

which is used to approximate $\eta(\boldsymbol{\xi})$. The MLP is naturally extended to multiple output problems, where $\boldsymbol{y} = (y_1, \ldots, y_d)^T = (\eta_1(\boldsymbol{\xi}), \ldots, \eta_d(\boldsymbol{\xi}))^T \in \mathbb{R}^d$, by setting $o_i(\boldsymbol{\xi}) = a_i^{(n+1)} \approx \eta_i(\boldsymbol{\xi})$ for $i = 1, \ldots, d$.

To train the model with a given dataset, a cost function is defined as:

$$E_D = \sum_{i=1}^{m} \frac{1}{2}\left(\boldsymbol{y}^{(i)} - \boldsymbol{o}(\boldsymbol{\xi}^{(i)}\right)^T \left(\boldsymbol{y}^{(i)} - \boldsymbol{o}(\boldsymbol{\xi}^{(i)}\right), \tag{2.12}$$

where $\boldsymbol{o}(\boldsymbol{\xi}^{(i)} = (o_i(\boldsymbol{\xi}^{(i)}), \ldots, o_d(\boldsymbol{\xi}^{(i)})^T$. One can define a vector of weights $\boldsymbol{w} = (w_{10}^{(0)}, \ldots, w_{dN_n}^{(n)})$. Finding the $\boldsymbol{w}$ that minimises the square sum error defined in Eq. (2.12) would give an optimal approximation to our training set $\{\boldsymbol{y}^{(i)}\}_{i=1}^m$. This, however, will not necessarily generate an accurate approximation to $\boldsymbol{\eta}(\boldsymbol{\xi})$ at test inputs (*generalization* of the model) as a consequence of over-fitting, which is a major issue in basic ANN formulations. The problem can be alleviated by adding a regularization term (a general method for optimization problems) to the cost function as follows:

$$E = \beta E_D + \alpha E_w = \beta E_D + \alpha \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w}, \tag{2.13}$$

where E_w is the sum of squares of the network weights and β and α are parameters determining the weighting of each cost term. A large β could lead to good approximations to the training data but may result in overfitting while a large α would improve generalization but underestimate the error in fitting the model to the training data.

Minimization of (2.13) can achieved with a gradient based optimisation algorithm, e.g., gradient descent. These approaches, however, involve computing the

partial derivatives of E w.r.t. each weight, which is computational intensive. A *back-propagation* is typically introduced to efficiently calculate the partial derivatives [20].

Other methods, e.g., early-stopping and cross-validation, could be implemented to improve generalization and avoid over-fitting. Another choice is Bayesian networks, in which a prior is placed on the weights, leading to improved generalization. An efficient approach, which avoids a full Bayesian estimation of all network weights (highly computationally intensive and therefore rarely adopted) is Bayesian regularization [70, 71], which is implemented in the thesis.

The weights are assumed to have zero-mean Gaussian prior distributions. Set α to the inverse variance of the zero-mean (assumed) Gaussian noise and β to the inverse variance of the weights. By Bayes law, the posterior density of the weights is:

$$P(\boldsymbol{w}|D,\alpha,\beta,\mathcal{M}) = \frac{P(D|\boldsymbol{w},\beta,\mathcal{M})P(\boldsymbol{w}|\alpha,\mathcal{M})}{P(D|\alpha,\beta,\mathcal{M})}, \tag{2.14}$$

where $D = \{y^{(i)}, \boldsymbol{\xi}^{(i)}\}_{i=1}^{m}$ is the data set, $\mathcal{M}$ indicates the MLP model, $P(\boldsymbol{w}|\alpha,\mathcal{M})$ is the prior density and $P(D|\boldsymbol{w},\beta,\mathcal{M})$ is the likelihood function. The optimal weights should maximize the posterior likelihood $P(\boldsymbol{w}|D,\alpha,\beta,\mathcal{M})$. It is assumed that the noise in the training data and the weight prior are both Gaussian:

$$\begin{aligned} P(D|\boldsymbol{w},\beta,\mathcal{M}) &= \frac{1}{Z_D(\beta)}\exp\left(-\beta E_D\right), \\ P(\boldsymbol{w}|\alpha,\mathcal{M}) &= \frac{1}{Z_w(\alpha)}\exp\left(-\alpha E_w\right), \end{aligned} \tag{2.15}$$

where $Z_D(\beta) = (\pi/\beta)^{m/2}$ and $Z_w(\alpha) = (\pi/\alpha)^{N/2}$ with N being the total number of weights. Substituting these assumptions into Eq. (2.14) yields:

$$\begin{aligned} P(\boldsymbol{w}|D,\alpha,\beta,\mathcal{M}) &= \frac{\left(Z_D(\beta)Z_w(\alpha)\right)^{-1}\exp\left(-(\beta E_D + \alpha E_w)\right)}{P(D|\alpha,\beta,\mathcal{M})} \\ &= \frac{\left(Z_F(\alpha,\beta)\right)^{-1}\exp\left(-F(\boldsymbol{w})\right)}{P(D|\alpha,\beta,\mathcal{M})}. \end{aligned} \tag{2.16}$$

In a Bayesian framework, the optimal weights should maximise the posterior, which is equivalent to minimising the regularised objective function $E = \beta E_D + \alpha E_w$. The posterior of α and β has the form (again using Bayes rule):

$$P(\alpha, \beta|D, \mathcal{M}) = \frac{P(D|\alpha, \beta, \mathcal{M})P(\alpha, \beta|\mathcal{M})}{P(D|\mathcal{M})}. \tag{2.17}$$

To derive the maximum of the posterior $P(\alpha, \beta|D, \mathcal{M})$ a uniform prior density $P(\alpha, \beta|\mathcal{M})$ is assumed so that the maximisation could be obtained by maximising the likelihood function $P(D|\alpha, \beta, \mathcal{M})$. Notice that the likelihood function is also the normalisation term shown in Eq. (2.14) and (2.16). One can thus solve Eq. (2.14) for the normalisation factor:

$$\begin{aligned} P(D|\alpha, \beta, \mathcal{M}) &= \frac{P(D|\boldsymbol{w}, \beta, \mathcal{M})P(\boldsymbol{w}|\alpha, \mathcal{M})}{P(D, \alpha, \beta, \mathcal{M})} \\ &= \frac{\left((Z_D(\beta))^{-1}\exp\left(-\beta E_D\right)\right)\left((Z_w(\alpha))^{-1}\exp\left(-\alpha E_w\right)\right)}{(Z_F(\alpha, \beta))^{-1}\exp\left(-F(\boldsymbol{w})\right)} \\ &= \frac{Z_F(\alpha, \beta)}{Z_D(\beta)Z_w(\alpha)}\frac{\exp(-\beta E_D - \alpha E_w)}{\exp(-F(\boldsymbol{w}))} \\ &= \frac{Z_F(\alpha, \beta)}{Z_D(\beta)Z_w(\alpha)}. \end{aligned} \tag{2.18}$$

To solve for Z_F, which is the only remaining unknown, a quadratic Taylor expansion of $F(\boldsymbol{w})$ around the minimum point $\boldsymbol{w}^{MP}$ (i.e. a Laplace approximation) is implemented. This yields:

$$Z_F \approx (2\pi)^{N/2}\left(\det\left(\left(\mathbf{H}^{MP}\right)^{-1}\right)\right)^{1/2}\exp(-F(\boldsymbol{w}^{MP})). \tag{2.19}$$

Where $\mathbf{H} = \beta\nabla^2_{\boldsymbol{w}}E_D + \alpha\nabla^2_{\boldsymbol{w}}E_w$ is the Hessian matrix of the objective function and $\mathbf{H}^{MP}$ is the Hessian matrix evaluated at $\boldsymbol{w} = \boldsymbol{w}^{MP}$. Substituting Eq. (2.19) into Eq. (2.18) and solving for the optimal values of α and β by setting the corresponding

derivatives to zero yields:

$$\begin{aligned} \alpha^{MP} &= \frac{\gamma}{2E_w(\boldsymbol{w}^{MP})}, \\ \beta^{MP} &= \frac{m-\gamma}{2E_D(\boldsymbol{w}^{MP})}, \end{aligned} \tag{2.20}$$

where:

$$\gamma = \frac{N - 2\alpha^{MP}}{\mathrm{tr}(\mathbf{H}^{MP})}. \tag{2.21}$$

To optimize α and β, the Hessian matrix $\mathbf{H}^{MP}$ is required. Using a Gauss-Newton approximation to the Hessian matrix and the Levenberg-Marquardt algorithm, these hyperparameters are calculated using an iterative procedure detailed in [72].

2.2.3 Support vector machine regression

Support vector machines (SVMs) are a widely used supervised machine learning technique for classification and regression. SVMs were introduced by Vapnik [73] and extended for non-linear problems using a 'kernel trick' by Boser [74]. Another significant improvement for SVMs was the standard incarnation (soft margin) introduced by [75]. In this section, idea of SVM regression for a univariate output is briefly introduced, following [76]. Multi-outputs remain challenging for SVM regression. Thanks to the framework that would be developing later, only the univariate version is needed and thus introduced here.

Suppose that one has a training data set $D = \{y^{(i)}, \boldsymbol{\xi}^{(i)}\}_{i=1}^{m}$ as before, where $y \in \mathbb{R}$ and $\boldsymbol{\xi} \in \mathcal{X} \subset \mathbb{R}^l$. A SVM regression [75] assumes the regression function $\eta(\boldsymbol{\xi})$ takes a linear form:

$$\eta(\boldsymbol{\xi}) = \boldsymbol{w}^T\boldsymbol{\xi} + b, \tag{2.22}$$

where $\boldsymbol{w} \in \mathcal{X}$, $b \in \mathbb{R}$. The aim of a so-called ε-SVM regression [75] is to solve the

following convex optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \\ \text{subject to} \quad & y^{(i)} - \boldsymbol{w}^T\boldsymbol{\xi}^{(i)} - b \leqslant \varepsilon \\ & \boldsymbol{w}^T\boldsymbol{\xi}^{(i)} + b - y^{(i)} \leqslant \varepsilon \end{aligned} \tag{2.23}$$

Eq. (2.23) is interpreted as follows: given an error precision ε, find $f(\boldsymbol{\xi})$ that has most ε deviation from the targets $y^{(i)}$, $\forall i$, and is as 'flat' as possible. A feasible solution (satisfying the constraints) is not always possible, or one may wish to allow for some margin in the error tolerance. A "soft margin" loss function [75] involving slack variables ζ_i, ζ_i^* is introduced as a consequence. The target formulation becomes:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + c\sum_{i=1}^{m}(\zeta_i + \zeta_i^*) \\ \text{subject to} \quad & y^{(i)} - \boldsymbol{w}^T\boldsymbol{\xi}^{(i)} - b \leqslant \varepsilon + \zeta_i \\ & \boldsymbol{w}^T\boldsymbol{\xi}^{(i)} + b - y^{(i)} \leqslant \varepsilon + \zeta_i^* \\ & \zeta_i, \zeta_i^* \geqslant 0 \end{aligned} \tag{2.24}$$

The constant c determine the balance between 'flatness' (a small w) and deviation of the function from the targets. Eq. (2.24) is solved by constructing a Lagrange function L and a set of dual variables as follows:

$$\begin{aligned} L = & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + c\sum_{i=1}^{m}(\zeta_i, \zeta_i^*) - \sum_{i=1}^{m}\alpha_i(\varepsilon + \zeta_i - y^{(i)} + \boldsymbol{w}^T\boldsymbol{\xi} + b) \\ & - \sum_{i=1}^{m}\alpha_i^*(\varepsilon + \zeta_i^* + y^{(i)} - \boldsymbol{w}^T\boldsymbol{\xi} - b) - \sum_{i=1}^{m}(\beta_i\zeta_i, \beta_i^*\zeta_i^*). \end{aligned} \tag{2.25}$$

The dual variables $\alpha_i, \alpha_i^*, \beta_i, \beta_i^*$ are strictly nonnegative. Optimization of Eq. (2.25) is achieved through the saddle point condition with respect to the target

variable $\boldsymbol{w}, b, \zeta_i, \zeta_i^*$:

$$\begin{aligned} \partial_b L &= \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) = 0, \\ \partial_{\boldsymbol{w}} L &= \boldsymbol{w} - \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) \boldsymbol{\xi}^{(i)} = 0, \\ \partial_{\zeta_i} L &= c - \alpha - \beta_i \qquad \partial_{\zeta_i^*} L = c - \alpha_i^* - \beta_i^*, \end{aligned} \tag{2.26}$$

Substituting Eq. (2.26) into Eq. (2.25) yields the dual optimization problem,

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(\boldsymbol{\xi}^{(i)})^T \boldsymbol{\xi}^{(j)}, \\ & - \varepsilon \sum_{i=1}^{m} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) y^{(i)}, \\ \text{subject to} \quad & \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) = 0, \\ & \alpha_i, \alpha_i^* \in [0, c]. \end{aligned} \tag{2.27}$$

From Eq. (2.26), it is known that $\boldsymbol{w} = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) \boldsymbol{\xi}^{(i)}$. To compute $\boldsymbol{b}$, the Karush-Kuhn-Tucker (KKT) conditions [77] demand that the product of dual variables and constrains must vanish:

$$\begin{aligned} & \alpha_i (\varepsilon + \zeta_i - y^{(i)} + \boldsymbol{w}^T \boldsymbol{\xi} + b) = 0, \\ & \alpha_i^* (\varepsilon + \zeta_i + y^{(i)} - \boldsymbol{w}^T \boldsymbol{\xi} - b) = 0, \\ & (c - \alpha_i) \zeta_i = 0, \\ & (c - \alpha_i^*) \zeta_i^* = 0. \end{aligned} \tag{2.28}$$

With these conditions one can eventually compute b as:

$$\begin{aligned} b &= y^{(i)} - \boldsymbol{w}^T \boldsymbol{\xi}^{(i)} - \varepsilon \quad \textbf{for} \quad \alpha_i \in (0, c), \\ b &= y^{(i)} - \boldsymbol{w}^T \boldsymbol{\xi}^{(i)} + \varepsilon \quad \textbf{for} \quad \alpha_i^* \in (0, c). \end{aligned} \tag{2.29}$$

Notice the inner product in Eq. (2.27) takes the linear form $\boldsymbol{\xi}^{(i)^T}\boldsymbol{\xi}^{(j)}$. One could, however, map the original data into a feature space and compute the inner product in that feature space in the form of a kernel function $\Bbbk(\boldsymbol{\xi}^{(i)},\boldsymbol{\xi}^{(j)})$ The advantage is that through the mapping one could allow for a nonlinear data structure and the mapping could be carried out implicitly. This is known as the kernel trick [74, 78]. The same idea (kernel trick) is also used in normal PCA to formulate kernel PCA, introduced later in section 2.4.1.

2.3 Linear dimensionality reduction

For many data sets of high dimensionality, the data points actually lie (exactly or approximately) in a subspace embedded in the original space. i.e., there are underlying hidden features that describe the observed data in a more compact form. Dimensionality reduction techniques approximate these features (the subspaces) and lead to the representation of the data using fewer degrees of freedom by preserving a preselected level of some important quantity, e.g., the 'variance' (a generalised measure of the spread of the points in different directions).

Within the vast array of dimensionality reduction methods, distinctions between linear and nonlinear methods are clear. Linear methods look for a linear subspace. All methods that do not fall into this category are nonlinear. It is begun below by introducing two commonly used linear dimensionality reduction techniques, namely, principal component analysis (PCA) [79] and multidimensional scaling (MDS).

2.3.1 Principal component analysis

Principal component analysis (PCA) is the most popular technique for dimension reduction. It has many applications, e.g., denoising, image compression and feature extraction. It is simple and especially effective when dealing with data that lives in

a linear subspace. Consider a dataset $\mathbf{Y} = [\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(m)}] \in \mathbb{R}^{d \times m}$, the columns of which are vector valued data points $\boldsymbol{y}^{(i)}$. PCA aims to find an r-dimensional linear subspace of the original space $\mathbb{R}^d$, spanned by a basis $\mathbf{V}_r = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r]$ such that the projections of the original data onto the subspace preserve as much variance as possible.

The projections are given by $\mathbf{Z} = \mathbf{V}_r\mathbf{Y}$, where $\mathbf{Z} = [\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}]$ is a matrix of the individual projections $\boldsymbol{z}^{(i)}$ corresponding to $\boldsymbol{y}^{(i)}$. The objective of PCA is thus:

$$\max_{\mathbf{V}_r : \mathbf{V}_r^T\mathbf{V}_r = \mathbf{I}} \mathbb{V}\mathrm{ar}\left(\mathbf{V}_r\mathbf{Y}\right). \tag{2.30}$$

When $r = d$, PCA is a basis transformation without dimension reduction or any *information loss* (a concept from information theory that I do not touch upon directly in this thesis). PCA requires $\mathbb{V}\mathrm{ar}(\boldsymbol{v}_1\mathbf{Y}) \geqslant \mathbb{V}\mathrm{ar}(\boldsymbol{v}_2\mathbf{Y}) \geqslant \cdots \geqslant \mathbb{V}\mathrm{ar}(\boldsymbol{v}_r\mathbf{Y})$ to guarantee that, with a specific r, the projections of the original data onto the subspace spanned by the basis $\mathbf{V}_r$ always preserve the maximum possible variance (among all possible orthonormal bases). The $\boldsymbol{v}_i$ are eigenvectors of the empirical *covariance matrix* $\mathbf{C}$ (described in detail in a later chapter) of the observed data:

$$\mathbf{C}\boldsymbol{v} = \lambda\boldsymbol{v}. \tag{2.31}$$

The projections of the original data onto the new basis are easily computed by $\boldsymbol{z}^{(i)} = \mathbf{V}^T\boldsymbol{y}^{(i)}$. These projections can be seen as the corresponding underlying coefficients or features, and the low-dimensional approximating subspace is $\mathrm{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r)$.

2.3.2 Multidimensional scaling

Rather than trying to use a new basis that preserves the variances of data as much as possible, which is the crux of PCA, MDS yields representative spatial configurations to describe the target dataset such that 'connectivity' relations between points are as close to the original ones as possible. When the connectivity relations are measured

by Euclidean distances, MDS is equivalent to PCA.

Classic MDS was first introduced by Torgerson *et al.* [80] to solve the problem of obtaining a descriptive map of data points given only the Euclidean distances between the points. The Euclidean distances can be generalised to connectivities/proximities that describe the (dis)similarity between data points in a way that is more suitable for the nature of the data (e.g., categorical) [81]. MDS then uses the proximities to find the optimal map (without any iterative procedures).

For the dataset $\boldsymbol{y}^{(i)} \in \mathcal{M} \subset \mathbb{R}^d, i = 1, \ldots, m$, the Euclidean distances between each pair are denoted by $d(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$. the shorthand $d(i,j) = d(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$ is used and a matrix $\mathbf{D}$ with entries $d(i,j)$ is defined. Classic MDS is designed to minimize the cost function:

$$E = \sum_{i \neq j} (d(i,j) - d'(i,j))^2, \tag{2.32}$$

where $d'(i,j)$ is the Euclidean distance between the low-dimensional representations of $\boldsymbol{y}^{(i)}$ and $\boldsymbol{y}^{(j)}$. The optimisation is achieved by solving the eigenproblem:

$$\mathbf{K}\boldsymbol{v} = \lambda \boldsymbol{v}, \tag{2.33}$$

where $\mathbf{K} = -\mathbf{H}\mathbf{P}_s\mathbf{H}/2$, $\mathbf{P}_s = \mathbf{D} \circ \mathbf{D}$ ($\circ$ denotes a Hadamard product) is the matrix of square distances, $\mathbf{H} = \mathbf{I} - \mathbf{1}\mathbf{1}^T/m$ is the known centring matrix, $\mathbf{I}$ denotes the identity matrix, and $\mathbf{1}$ is a vector of ones 1. The notation $\mathbf{K}$ is used since it could be seen as a kernel matrix, as will be shown in chapter 3.

The non-zero eigenvalues λ_i, $i = 1, \ldots, d$, of $\mathbf{K}$ are arranged in a non-increasing order and the corresponding eigenvectors $\boldsymbol{v}_i \in \mathbb{R}^m$ are normalized. This yields $\mathbf{Z} = \mathbf{V}\mathbf{\Lambda}^{1/2} \in \mathbb{R}^{m \times d}$, where the columns of $\mathbf{V}$ are the $\boldsymbol{v}_i$. Truncating $\mathbf{V}$ at the first r eigenvectors to form $\mathbf{V}_r \in \mathbb{R}^{m \times r}$ yields low-dimensional representations of the $\boldsymbol{y}^{(i)}$, as the rows of $\mathbf{Z}_r = \mathbf{V}_r\mathbf{\Lambda}_r^{1/2}$.

Other measurements could be adopted to generate the proximities, which

are the inputs to MDS. The proximities indicate the overall similarity or dissimilarity of the target data based on the measurement assumptions. There are two types of methods for obtaining the proximities: direct and indirect methods. The direct methods aim to assign a (scale) value or ranking of (dis)similarity for each pair of observations while the indirect methods derive the proximities from other measurements, e.g. confusion data or correlation matrices [82].

For the direct methods, the classic approach is to use metric MDS where a valid metric, e.g., Euclidean distances, is used to form the proximity matrix $\mathbf{D}$. Non-metric MDS, by contrast, assumes that only the ordering of proximities is meaningful. Non-metric MDS is helpful when dealing with non-classical data.

2.4 Nonlinear dimensionality reduction (manifold learning)

When the underlying (hidden or latent) space is a linear subspace or is close to a linear subspace of the original space, linear dimension reduction techniques, e.g., PCA and MDS, are effective and efficient methods. For complex data, however, linear dimension reduction cannot adequately reveal the underlying subspace. This motivated the development of many nonlinear dimensionality reduction techniques. Rather than locating a linear intrinsic subspace, nonlinear methods attempt to characterise a manifold on which the observed data lies [83]. For this reason, many of the nonlinear dimension reduction methods are also called manifold learning methods. In general, neither the geometry nor the intrinsic dimensionality of the data are known. Besides, in practice, the data is finite and is often corrupted with noise. Manifold learning is thus inherently ill-posed [84] and can only be solved with certain assumptions and heuristics.

From a machine learning point of view, dimensionality reduction and manifold learning techniques are unsupervised methods, attempting to locate the patterns

behind a dataset in the absence of corresponding inputs, in contrast to the regression techniques discussed earlier. Manifold learning methods can be classified as kernel based methods, embeddings, graph based methods and spectral methods (involving an eigenspectrum of some quantity, e.g., a kernel or covariance matrix), often falling into several categories.

Kernel methods, e.g. kernel PCA [85] and SVMs, apply a 'kernel trick', i.e., they introduce a kernel function that specifies the inner product between any two points mapped (implicitly) to a 'feature space' and deal only with quantities in the feature space, which is a computationally cheaper task. This idea can also extend linear methods to nonlinear counterparts. In contrast, graph based methods, e.g., local linear embedding (LLE) [86], Laplacian eigenmaps ([87]) and Hessian eigenmaps [88], aim to minimize some form of distortion of new configurations of data, which is normally achieved *via* a spectral analysis of Laplace-type operators (making them also spectral methods as well as embedding methods). In the section, general discussions on manifold learning methods implemented in this thesis are issued.

2.4.1 Kernel principal component analysis

The key ingredient of the kernel methods [89, 90] is to map the target data to a *feature space*, where the nonlinearity in the original data space can potentially be handled using linear methods. Rather than carrying out the mapping to a feature space explicitly, a kernel method uses a *kernel function* to define a dot product between points mapped to the feature space. Operations, e.g., feature extraction, regression, classification and clustering in the feature space are based on the dot product. Thus, kernel methods, unlike optimisation methods, are computationally inexpensive and effective.

The standard PCA fails when dealing with complex data sets (not lying in or close to a linear subspace). Suppose one maps the data points, lying in some

space $\mathcal{M}$, to a higher-dimensional (possibly infinite dimensional) feature space $\mathscr{F}$ by a mapping $\boldsymbol{\phi} : \mathcal{M} \to \mathscr{F}$. If the mapped points lie close to a linear subspace of $\mathscr{F}$, linear PCA can be performed successfully. The mapping $\boldsymbol{\phi}$ is specified implicitly *via* a kernel function $\Bbbk(\cdot, \cdot)$. By adopting different kernel functions, kernel PCA is capable of dealing with different types of nonlinear structures [74, 91].

Consider the dataset $\{\boldsymbol{y}^{(j)}\}_{j=1}^{m}$ where $\boldsymbol{y} = (y_1, \ldots, y_d)^T \in \mathcal{M} \subset \mathbb{R}^d$. For standard PCA, the basis is found by solving Eq. (2.31). Denote the mapped data as $\boldsymbol{\phi}(\boldsymbol{y}^{(i)})$ for $i = 1, \ldots, m$. Standard PCA could be carried out in the feature space by solving:

$$\mathbf{C}_{\mathcal{F}} \boldsymbol{w} = \lambda \boldsymbol{w}, \tag{2.34}$$

where $\mathbf{C}_{\mathcal{F}}$ is the covariance matrix of the mapped data points in feature space. Kernel PCA does not solve Eq. (2.34) directly. Instead, it is reformulated as an eigenvalue problem for a kernel matrix $\mathbf{K}$, with entries $\Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$, $i, j = 1, \ldots, M$. The projections are computed with reference only to the (known) kernel values. The choice of the kernel function is ideally based on prior knowledge of the data. In practice, however, such knowledges are usually unclear and the choice of kernel functions is difficult to made. In such cases, one usually makes an assumption that the data points are isotropic. The assumption does actually comply with our understanding of many real-word process and does work well in many realistic cases for this reason. One simple but commonly used kernel function to express such a isotropic belief is the Gaussian kernel,

$$\Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \exp\left(-\sum_{k=1}^{d} (y_k^{(i)} - y_k^{(j)})^2 / 2\sigma^2 \right), \tag{2.35}$$

where σ is a control parameter that is used to adjust the flexibility of the kernel, when prior knowledges are not available. The kernel function here shares the same definitions, and thus properties, as that defined previously for Gaussian process. Notice that in here only one length scale parameter is designed. This implies the

assumption that the data has euclidean distance indicating correlations which is usually true for spatial data. This, however, does not hinder one from choosing the more general form that is mentioned previously.

2.4.2 Isomap

Isomap was introduced by Tenenbaum *et al.* [92] as a method to find the nonlinear degrees of freedom behind complex natural observations, e.g., human writing and face images under different conditions. It is based on classic MDS, but uses the intrinsic geodesic distances between observations rather than the Euclidean distances as proximities. For neighbouring data points, the Euclidean distance could be used to well approximate the geodesic distance. The difficulty is to approximate the geodesic distance between two far-away observations. The original Isomap approximates such distances by finding a shortest path through data points lying between such two observations. The shortest path is obtained from Dijkstra's or Floyd's shortest-path algorithm [93, 94].

Balasubramanian *et al.* [95] pointed out that the work of [92] was not new since the framework had been explored in the context of flattening cortical surfaces using geodesic distances and MDS [96, 97], where Dijkstra's graph-based algorithm was applied to approximate the geodesic distances [98]. Moreover, there was some doubt as to the topological stability of the method, which in some cases can only be implemented after some preprocessing of the input data. It may also lead to erroneous connections on the neighbourhood graph. Attempts were made by Jenkins *et al.* [99] to overcome these issues, extending Isomap to spatio-temporal datasets (ST-Isomap).

Choi *et al.* [100] introduced a general framework called the kernel Isomap, which was naturally derived from kernel PCA as a generalisation of the standard Isomap. A method for eliminating critical outliers by investigating the network flow was developed to deal with the problem of topological instability by Saxena *et al.*

[101]. The main idea in their work was to remove the nearest neighbour that violates local linearity of the neighbourhood graph. Other problems associated with Isomap include the inability to handle 'holes' in the manifold, which can be dealt with by tearing the manifold [102], and failure when the manifold is nonconvex [92]. Despite these weaknesses, Isomap was widely and successfully applied in many tasks, e.g., head pose estimation [103], biomedical data visualization [104] and wood inspection [105].

Isomap firstly determines which points are neighbouring points based on Euclidean distances $d(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$. A simple approach is to connect each point to all other points that fall within some fixed radius or to connect the k closest points. A weighted graph $\mathcal{G}$ is defined with nodes representing data points and edges equal to Euclidean distances $d(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$ between neighbouring points. Distances between points that are not neighbours are defined as any arbitrarily large number.

The geodesic distances between non-neighbour points are estimated using the shortest path in the graph $\mathcal{G}$. The process is easily achieved through Floyd's algorithm which requires $\mathcal{O}(m^3)$ operations, where m is the number of data points. Define the graph distance matrix $\mathbf{D}_{\mathcal{G}}$ with entries equal to the geodesic distances $\{d_{\mathcal{G}}(i,j)\}_{i,j=1}^{m}$ among all data points. Then one simply apply MDS using $\mathbf{D}_{\mathcal{G}}$ as the proximity matrix to construct an r-dimensional spatial configuration. Isomap is thus designed to to minimize the cost function:

$$E = \sum_{i \neq j} (d_{\mathcal{G}}(i,j) - d'(i,j))^2, \tag{2.36}$$

where $d'(i,j)$ is the Euclidean distance between the low-dimensional representations of $\boldsymbol{y}^{(i)}$ and $\boldsymbol{y}^{(j)}$. As in classic MDS, the optimization problem is solved through eigen analysis of $\mathbf{D}_{\mathcal{G}}$. It is clear that the Euclidean distances among new configurations approximate the geodesic distances in the original data $\mathbf{Y}$. Thus, Isomap, like MDS, is an (approximate) optimal isometric embedding of the data in an r-dimensional

space.

2.4.3 Diffusion maps

As in Isomap, in diffusion maps the data points $\boldsymbol{y}^{(i)}$ are identified as vertices on a weighted undirected graph $\mathcal{G}$, where the weight is defined by a symmetric and positive definite kernel $\Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$. A diffusion process [106] on $\mathcal{G}$ is constructed by normalizing the connectivity (or kernel) matrix $\mathbf{K} = [\mathsf{K}_{ij}]$, where $\mathsf{K}_{ij} = \mathsf{k}(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)})$. The degree matrix is defined as:

$$\mathbf{D} = \operatorname{diag}\left(\sum_j \mathsf{K}_{1j}, \ldots, \sum_j \mathsf{K}_{mj}\right). \tag{2.37}$$

From this, one may define a *diffusion matrix* as $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$, with entries P_{ij}, which is a Markov matrix. That is, the entry P_{ij} is a transition probability from node i to j in a random walk on $\mathcal{G}$. A t step transition probability is given by the (i, j)-th entry of $\mathbf{P}^t$.

As is shown in [107], diffusion maps define a *diffusion distance* as:

$$D_t^2(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \sum_{k=1}^{m} (P_{ik}^t - P_{jk}^t)^2. \tag{2.38}$$

The diffusion distances are set equal to the Euclidean distances of the low (r) dimensional representations $\boldsymbol{\psi}_r(\boldsymbol{y}^{(i)})$ and $\boldsymbol{\psi}_r(\boldsymbol{y}^{(j)})$. The *diffusion map* $\boldsymbol{\psi}_r(\boldsymbol{y}) : \mathcal{M} \mapsto \mathbb{R}^r$ embeds the original data into an r-dimensional space by selecting the diffusion distance as the best distance to preserve (in contracts to Eclidean distance in PCA/MDS or geodesic distance in Isomap). The map is specified by conducting an eigen-analysis of the Markov matrix $\mathbf{P}$. Details are given in chapter 4.

2.4.4 Pre-image problems

For many applications, e.g., feature extraction, only the features, which are reduced-dimensional configurations, are of concern. But there are also many applications, e.g., image denoising, where a inverse mapping from the low-dimensional representations to the original space is crucial. Another example is the application of kernel-based clustering. In order to visualize the centroid, which exists in the reduced dimensional space, an inverse mapping is needed [108]. In our applications, the predictions are only meaningful in the original (physical) space. Thus, the inverse map is a necessity.

The problem of finding an inverse map is often referred to as the *pre-image problem*. For linear methods such as PCA, an inverse mapping is straight forward and does not require anything more complex than a matrix transpose. For manifold learning methods, the pre-image problem remains a challenge. Exact pre-image solutions generally do not exist. Mika *et al.* [109] suggested an approximate solution through optimization. This, along with other methods, is discussed below.

Since the exact pre-image solution $\widehat{\boldsymbol{y}}$ may not exist, Mika *et al.* [109] addressed this problem by minimizing the square distance between $\boldsymbol{\phi}_r(\widehat{\boldsymbol{y}})$ and $\boldsymbol{\phi}_r(\boldsymbol{y})$, where $\boldsymbol{\phi}_r(\widehat{\boldsymbol{y}})$ is the reduced-dimensional form of the point $\widehat{\boldsymbol{y}}$:

$$\arg\min_{\boldsymbol{y}} ||\boldsymbol{\phi}_r(\boldsymbol{y}) - \boldsymbol{\phi}_r(\widehat{\boldsymbol{y}})||^2 = \arg\min_{\boldsymbol{y}}(||\boldsymbol{\phi}_r(\widehat{\boldsymbol{y}})||^2 - 2\boldsymbol{\phi}_r(\boldsymbol{y})^T\boldsymbol{\phi}_r(\widehat{\boldsymbol{y}}) + \epsilon), \tag{2.39}$$

in which the remainder ϵ is independent of $\boldsymbol{y}$. The problem can be solved using optimization algorithms [110] (e.g., fixed-point iteration) but will suffer from trapping in local optima and sensitivity to initial guesses. Moreover, it is computationally expensive and unstable for complex and large data sets.

For many commonly used kernels, there is a simple relationship between distances in feature space and distances in the original space [111]. Kwok *et al.* [108] used such relationships to find a non-iterative solution using only linear algebra. Ma

et al. [112] improved this method to solve the pre-image problem for a stochastic input model. An efficient and effective pre-image solution combining these approaches will be introduced in chapter 4.

For graph based methods, Etyngier *et al.* [113] imposed a non-linear shape prior to estimate the pre-image solution in diffusion maps. Another method was introduced by Thorstensen *et al.* [114]. They solved the pre-image problem in the context of diffusion maps for shape and image denoising. Their method defines a pre-image solution as a Karcher-mean that is interpolated using neighbouring samples. They claimed that the resulting *Nystrom extension* is more meaningful, which leads to a better pre-image estimate even for heavy noise corrupted data. The method is a nonlinear optimization: $\widehat{\boldsymbol{y}} \approx \boldsymbol{y}^{(\theta)} = \arg\min_{\boldsymbol{y}} ||\Phi(\boldsymbol{y}^{(\theta)}) - \Phi(\widehat{\boldsymbol{y}})||^2$, where $\boldsymbol{y}^{(\theta)}$ is a Karcher mean for neighbourhood $\mathcal{N}$, comprising the $m+1$ closest (measured in terms of diffusion distances) sample points. The pre-image $\boldsymbol{y}$ is computed by gradient descent, generating a family of samples. As with other non-linear optimizations, such a solution may suffer from trapping in local optima, instablity and limited scalability. For the 2-D shapes considered in [113, 114] the method is feasible, but for very high-dimensional problems as considered in this thesis, it is not practical.

2.5 Design of experiment

Apart from the model chosen for an emulator, the most crucial factor in building an accurate emulator is the dataset that is used to train it. This training data are obtained by running a simulator at different parameter/input values (design points) to provide sufficient 'information' regarding the input-output relationship (response surface). Obviously the particular parameter set for which the simulator is evaluated should be carefully selected to cover the space of interest sufficiently well in order to provide the required information about the simulator response surface, ideally with a low number of simulator runs. *Design-of-experiment* (DOE) [115] is used to guide

this process.

Like the situation in optimisation, DOE is problem dependent and there is no systematic way to identify the best method. Of course the higher the number of samples available, the more information is revealed but the number of samples generated has to take into account the computational cost of the simulator. It is desirable if possible to limit the number of parameters using prior knowledge of the system behaviour w.r.t to the individual paramaters because the number of samples required to fill the design space rises exponentially with the number of parameters (the *curse of dimensionality*).

For continuous parameters, *space filling* DOE techniques [115] are widely used. These techniques choose design points as uniformly as possible over the space of interest. The Sobol sequence and Latin hypercube DOE are typically the best performing and the most robust methods, according to [115]. A Sobol sequence design is implemented in this thesis since it generates design points more uniformly, which is important if the number is low. The area of DOE is vast and I do not touch upon it much so a detailed discussion of methods is not provided. In the examples considered in this thesis, a simple space-filling method suffices.

2.6 Proper orthogonal decomposition

Let us first motivate the POD method. Consider the application of one of the spatial discretisation methods, e.g., finite difference, finite volume or finite element, to a linear, homogeneous parabolic PDE for a field $u(t, \boldsymbol{x})$. In any of these cases, a linear dynamical system is obtained:

$$\dot{\boldsymbol{u}}(t) = \mathbf{A}\boldsymbol{u}(t) \tag{2.40}$$

with some initial condition, where the *solution vector* $\boldsymbol{u}(t) = (u_1(t), \ldots, u_d(t))^T$ defines the values of $u(t, \boldsymbol{x})$ at d spatial locations (the number of degrees of freedom).

The value of d could be the number of grid points, the number of control volumes or the number of nodes in the FD, FV and FE (assuming it is nodal) schemes, respectively. The matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ is sparse.

A *Galerkin projection* approximates the problem (2.40) in a low dimensional space spanned by orthonormal vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$. That is, one can define the following approximation:

$$\boldsymbol{u} \approx \boldsymbol{u}_r(t) = \sum_{j=1}^{r} a_j(t)\boldsymbol{v}_j = \mathbf{V}_r \boldsymbol{a}(t) \tag{2.41}$$

where $\boldsymbol{a} = (a_1(t), \ldots, a_r(t))^T$ and $\mathbf{V}_r = [\boldsymbol{v}_1 \ldots \boldsymbol{v}_r]$. The Galerkin projection of equation (2.40) onto the basis vectors $\boldsymbol{v}_i$, $i = 1, \ldots, r$, consists of replacing $\boldsymbol{u}$ with $\boldsymbol{u}_r$ in (2.40) followed by left multiplication of each term by $\boldsymbol{v}_i$, repeating for each i. This leads to a new system with only r unknowns, $a_i(t)$, $i = 1, \ldots, r$, which can be written as follows:

$$\dot{\boldsymbol{a}}(t) = \mathbf{V}_r^T \mathbf{A} \mathbf{V}_r \boldsymbol{a}(t) \tag{2.42}$$

The basis $\mathbf{V}_r$ can be constructed in number of ways, e.g., balanced truncation, Krylov subspace methods or POD. The most widely used method is POD method, particularly for nonlinear problems, and is implemented in this thesis in chapter 6.

Suppose one has solved (2.40) and recorded the solution at times $\{t^{(i)}\}_{i=1}^m$. These solutions are labelled as $\{\boldsymbol{u}^{(i)} := \boldsymbol{u}(t^{(i)}) = (u_1(t^{(i)}), \ldots, u_d(t^{(i)}))^T\}_{i=1}^m$ and call them (discrete) *snapshots*. These snapshots are discrete time instances of $u(\boldsymbol{x}, t)$ at d particular locations in the spatial grid Ω, say $\{\boldsymbol{x}^{(i)}\}_{i=1}^d$. That is, $u_j(t^{(i)}) = u(\boldsymbol{x}_j, t^{(i)})$, $\forall i, j$.

POD extracts an optimal, in some well defined sense, representative basis for a field $u(\boldsymbol{x}, t)$, $(\boldsymbol{x}, t) \in \mathcal{D} \times [0, T]$, given an ensemble of 'snapshots' $\{u(\boldsymbol{x}, t^{(j)})\}_{j=1}^m$, $\boldsymbol{x} \in \mathcal{D}$. These are continuous (in space) equivalents of the discrete snapshots $\boldsymbol{u}^{(i)}$. The field $u(\boldsymbol{x}, t)$ can be regarded as a realisation of a random field indexed by $\boldsymbol{x}$ and t [36, 37, 116]. The underlying probability space is $(\Omega, \mathcal{A}, \mathbb{P})$, where Ω is the sample

space, $\mathcal{A}$ is the event space and $\mathbb{P}$ is a probability measure. It is assumed that:

1. The process is zero-mean (without loss of generality), $\mathbb{E}\left[|u(\boldsymbol{x},t)\right] = 0$

2. $u(\boldsymbol{x},t)$ is continuous in quadratic mean (q.m.) w.r.t. $\boldsymbol{x}$, i.e.:

$$\mathbb{E}\left[|u(\boldsymbol{x}+\delta\boldsymbol{x},t) - u(\boldsymbol{x},t)|^2\right] \to 0$$

   as $\delta\boldsymbol{x} \to \mathbf{0}$, $\forall t \in [0,T]$

3. $u(\boldsymbol{x},t)$ is stationary w.r.t. t, i.e., the autocovariance $\mathbb{E}\left[u(\boldsymbol{x},t)u(\boldsymbol{x}',t')\right]$ depends only on $t-t'$ so that the spatial autocovariance takes the form:

$$\mathbb{E}\left[u(\boldsymbol{x},t)u(\boldsymbol{x}',t)\right] = C(\boldsymbol{x},\boldsymbol{x}'), \quad \boldsymbol{x},\boldsymbol{x}' \in \mathcal{D}$$

For a fixed $t \in [0,T]$, $u(\boldsymbol{x},t)$ defines a *one-parameter* random field indexed by $\boldsymbol{x} \in \mathcal{D}$ [116]. Sample paths (for a fixed $\omega \in \Omega$) are deterministic functions $u(\cdot,t) : \mathcal{D} \to \mathbb{R}$. By the q.m. continuity assumption, $u(\cdot,t) \in L^2(\mathcal{D})$ for each $t \in [0,T]$ so that $u(\boldsymbol{x},t) \in L^2(0,T;L^2(\mathcal{D}))$. Applying Karhunen-Loéve (KL) theory [117] for a fixed t yields:

$$u(\boldsymbol{x},t) = \lim_{M\to\infty} \sum_{i=1}^{M} a_i(t)v_i(\boldsymbol{x}) \tag{2.43}$$

with convergence in q.m. In this separable form for $u(\boldsymbol{x},t)$, the randomness enters only through the dependence on t. The $v_i(\boldsymbol{x})$ form an orthonormal basis for $L^2(\mathcal{D})$ and are given by the eigenfunctions of the following integral operator:

$$\mathcal{C}v_i(\boldsymbol{x}) := \int_{\mathcal{D}} C(\boldsymbol{x},\boldsymbol{x}')v_i(\boldsymbol{x}')d\boldsymbol{x}' = \lambda_i' v_i(\boldsymbol{x}) \quad i \in \mathbb{N} \tag{2.44}$$

with corresponding real, positive eigenvalues $\lambda_i' > \lambda_{i+1}'$ $\forall i \in \mathbb{N}$. Furthermore, the coefficients $a_i(t) = (u, v_i)$ satisfy $\mathbb{E}[a_i(t)] = 0$ and from the definition of $v_i(\boldsymbol{x})$ and

λ_i' It is straightforward to obtain:

$$\mathbb{E}[a_i(t)a_j(t)] = \lambda_i'\delta_{ij} \tag{2.45}$$

where δ_{ij} is the kronecker-delta. Thus, the a_i are mutually uncorrelated, and ranging over t one obtains random processes $a_i(t)$. Implementations of the POD theory rely on an approximation to the expectation (averaging) operator:

$$\mathbb{E}[X] = \int_\Omega X(\omega)\mathbb{P}(d\omega) \tag{2.46}$$

The assumption of ergodicity allows us to estimate expectations as time averages (the *ergodic hypothesis*, which states that the system which reaches all possible states ω given infinite time). That is, one can average over snapshots to estimate expectations.

For an arbitrary orthonormal basis $\{\varphi_i\}_{i=1}^\infty$ of $L^2(\mathcal{D})$, one has:

$$\sum_{i=1}^r \mathbb{E}[(u, v_i)^2] = \sum_{i=1}^r \lambda_i' > \sum_{i=1}^r \mathbb{E}[(u, \varphi_i)^2] \tag{2.47}$$

for any $r \in \mathbb{N}$. This criterion is equivalent to:

$$\min_{\{\varphi_i\}} \mathbb{E}\left[\left|\left|u - \sum_{i=1}^r a_i\varphi_i\right|\right|^2\right] = \min_{\{\varphi_i\}} \left|\left|u - \sum_{i=1}^r a_i\varphi_i\right|\right|^2_{L^2(0,T;L^2(\mathcal{D}))} \tag{2.48}$$

where the latter equality is based on the ergodic hypothesis. This criterion leads to the eigenvalue problem (2.44) and alternatively defines the POD basis. It can be generalized to:

$$\min_{\{\varphi_i\}} \left|\left|u - \sum_{i=1}^r a_i\varphi_i\right|\right|^2_{L^2(0,T;\mathcal{H})} \tag{2.49}$$

for a general Hilbert space $\mathcal{H}$. In this case the POD basis is defined by the $\mathcal{H}$-

orthonormal eigenfunctions $v(\boldsymbol{x}) \in \mathcal{H}$ of the operator:

$$\mathcal{S}v := \mathbb{E}\left[u(u,v)_{\mathcal{H}}\right] = \int_0^T u(u,v)_{\mathcal{H}} dt \tag{2.50}$$

For $\mathcal{H} = L^2(\mathcal{D})$, $\mathcal{S} = \mathcal{C}$ using the commutativity of the time and spatial averaging operations.

To implement the method in practice a matrix of solution vectors (the discrete snapshots) is defined:

$$\mathbf{X} := \left[\boldsymbol{u}^{(1)} \dots \boldsymbol{u}^{(m)}\right] \tag{2.51}$$

and the *spatial* variance-covariance matrix:

$$\mathbf{C} = \mathbf{X}\mathbf{X}^T \approx \mathbb{E}\left[\boldsymbol{u}(t)\boldsymbol{u}(t)^T\right] \tag{2.52}$$

A numerical approximation of (2.44) at the quadrature points $\{\boldsymbol{x}^{(i)}\}_{i=1}^d$ and $\{t^{(i)}\}_{i=1}^m$ yields an eigenvalue problem in $\mathbb{R}^d$. For equally spaced quadrature points, a midpoint rule leads to a principal component analysis (PCA), i.e., $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$ for eigenvectors $\boldsymbol{v}_i \in \mathbb{R}^d$ and positive eigenvalues $\lambda_i > \lambda_{i+1}$, $\forall i = 1, \dots, d$. The j-th component $v_{i,j}$ of $\boldsymbol{v}_i$ can be identified with $v_i(\boldsymbol{x}^{(j)})$ and the (i,j)-th entry of $\mathbf{C}$ approximates $C(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ as a time average:

$$C(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) \approx \frac{1}{m} \sum_k u(\boldsymbol{x}^{(i)}, t^{(k)}) u(\boldsymbol{x}^{(j)}, t^{(k)})$$

In principle, any valid interpolation procedure could be implemented (in space and/or time) to discretize (2.44). For instance, in the FE formulation, one can approximate $u(\boldsymbol{x}, t)$ and $v_i(\boldsymbol{x})$ using a piecewise linear Lagrange basis $\{\psi_i(\boldsymbol{x})\}_{i=1}^d \subset L^2(\mathcal{D})$, which leads to a new eigenvalue problem:

$$\mathbf{C}\mathbf{M}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \tag{2.53}$$

where $\mathbf{M}$ is a mass matrix (as defined in the previous section) with entries $M_{ij} = (\psi_i(\boldsymbol{x}), \psi_j(\boldsymbol{x}))$. Defining $\overline{\boldsymbol{v}} = \mathbf{M}^{1/2}\boldsymbol{v}$, where $\mathbf{M}^{1/2}$ is a square root or Cholesky factor of $\mathbf{M}$, one obtains:

$$\mathbf{M}^{1/2}\mathbf{C}\mathbf{M}^{1/2}\overline{\boldsymbol{v}} = \lambda\overline{\boldsymbol{v}} \tag{2.54}$$

The eigenvalues/eigenvectors pairs $\{(\overline{\boldsymbol{v}}_i, \lambda_i)\}_{i=1}^{d}$ of $\mathbf{M}^{1/2}\mathbf{C}\mathbf{M}^{1/2}$ yield the POD basis vectors $\boldsymbol{v}_i = \mathbf{M}^{-1/2}\overline{\boldsymbol{v}}_i$ in the desired order.

In certain cases (depending on the relative sizes of m and d), it may be computationally convenient to use variants of POD or PCA to determine the $v_i(\boldsymbol{x})$ or $\boldsymbol{v}_i$. The *method of snapshots* is an indirect application of POD suitable for problems in which $m \ll d$. A temporal autocovariance function is defined:

$$K(t, t') = \int_{\mathcal{D}} u(\boldsymbol{x}, t)u(\boldsymbol{x}, t')d\boldsymbol{x} \tag{2.55}$$

with associated operator:

$$\mathcal{K}a_i(t) := \int_0^T K(t, t')a_i(t')dt' \tag{2.56}$$

The orthogonal eigenfunctions $a_i(t)$ of $\mathcal{K}$ are the POD coefficients and the eigenvalues are identical to those of $\mathcal{C}$. Using $\mathbb{E}[a_i(t)a_j(t)] = \lambda_i'\delta_{ij}$, the POD modes are given by:

$$v_i(\boldsymbol{x}) = \frac{1}{\lambda_i'}\int_0^T u(\boldsymbol{x}, t)a_i(t)dt \tag{2.57}$$

The discrete form (in space and time) of the eigenvalue problem is:

$$\mathbf{X}^T\mathbf{X}\boldsymbol{a}_i = \lambda_i\boldsymbol{a}_i \tag{2.58}$$

where $\mathbf{K} := \mathbf{X}^T\mathbf{X}$ is a kernel matrix with entries $K_{ij} = (\boldsymbol{u}^{(i)})^T\boldsymbol{u}^{(j)}$, i.e., the space-

discrete form of $K(t^{(i)}, t^{(j)})$. The eigendecomposition is:

$$\mathbf{K} = \mathbf{A}\mathbf{\Lambda}\mathbf{A}^T, \quad \mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_m) \tag{2.59}$$

where the columns of $\mathbf{A}$ are given by the $\boldsymbol{a}_i$. The j-th component $a_{i,j}$ of $\boldsymbol{a}_i$ approximates $a_i(t^{(j)})$ yielding the discrete-time approximation:

$$v_i(\boldsymbol{x}) = \frac{1}{\lambda_i} \sum_{j=1}^{m} u(\boldsymbol{x}, t^{(j)}) a_{i,j} \tag{2.60}$$

i.e., a linear combination of the snapshots. In the fully-discrete case, using the normalization $\boldsymbol{a}_i \mapsto \boldsymbol{a}_i'/\sqrt{\lambda_i}$, one obtains:

$$\boldsymbol{v}_i = \mathbf{X}\boldsymbol{a}_i'/\sqrt{\lambda_i} \tag{2.61}$$

These relationships are also evident from the singular value decomposition (SVD) of $\mathbf{X}$, that is:

$$\mathbf{X} = \mathbf{A}'\mathbf{\Lambda}^{1/2}\mathbf{V}^T \tag{2.62}$$

where the columns of $\mathbf{V}$ are given by the $\boldsymbol{v}_i$ and the columns of $\mathbf{A}'$ are given by the $\boldsymbol{a}_i'$. In this context, the columns of $\mathbf{A}'$ and $\mathbf{V}$, given respectively by the eigenvectors of $\mathbf{K}$ and $\mathbf{C}$, are referred to as left and right *singular vectors*. It is straightforward to show that:

$$\boldsymbol{v}_i = k\mathbf{X}\boldsymbol{a}_i', \quad k \in \mathbb{R} \tag{2.63}$$

Thus one recovers the earlier relationship by taking $k = 1/\sqrt{\lambda_i}$ to normalise the $\boldsymbol{v}_i$.

Chapter 3

Gaussian process emulators for high dimensional output spaces using Isomap

In this chapter, formulate the problem of data-driven emulation for spatial or spatio-temporal fields originating from parameterized PDE computer models (high-dimensional data) is firstly formulated. Followed by a discussion on the importance of correlation structures and the need for dimensionality reductions. A detailed description of existing methods is provided, particularly that of Higdon *et al.* [1], which is explained in detail. It is then the main concept: using a manifold learning method (in this chapter Isomap) to replace PCA as the dimensionality reduced method for GP emulation of parameterized spatio-temporal fields is introduced. Details of the inverse mapping and examples of the application, comparing to Higdon's method, are fully described.

This chapter is based on the publication [118]: W. Xing, A. A. Shah, and P. B. Nair,"Reduced dimensional gaussian process emulators of parameterized partial differential equations based on Isomap, *Proceedings of the Royal Society of London*

A: Mathematical, Physical and Engineering Sciences, vol. 471, p. 20140697, 2015.

3.1 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 quantities 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:

$$\boldsymbol{y}^{(j)} = \left(u(\boldsymbol{x}^{(1)};\boldsymbol{\xi}^{(j)}),\ldots,u(\boldsymbol{x}^{(d)};\boldsymbol{\xi}^{(j)})\right)^T \in \mathbb{R}^d. \tag{3.1}$$

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 3.4. 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. For a time dependent problem where the properties of interest are $\boldsymbol{u}(\boldsymbol{x}, t;\boldsymbol{\xi},)$ at specified (fixed) locations, $\boldsymbol{x}^{(i)}$, $i = 1,\ldots,d$ and time step, $t^{(j)}$, $j = 1,\ldots,n$, the vectorisation

process in the thesis is carried out in a similar manner as:

$$\begin{aligned}\boldsymbol{y}^{(j)} = \Big(&u(\boldsymbol{x}^{(1)}, t^{(1)}; \boldsymbol{\xi}^{(j)}), \ldots, (\boldsymbol{x}^{(d)}, t^{(1)}; \boldsymbol{\xi}^{(j)}), \ldots u(\boldsymbol{x}^{(1)}, t^{(2)}; \boldsymbol{\xi}^{(j)}), \ldots \\ &\ldots, u(\boldsymbol{x}^{(d)}, t^{(n)}; \boldsymbol{\xi}^{(j)})\Big)^T \in \mathbb{R}^{nd}.\end{aligned} \tag{3.2}$$

The simulator can be considered as a mapping $\boldsymbol{\eta} : \mathcal{X} \to \mathcal{M}$ (assumed to be injective), where $\mathcal{M} \subset \mathbb{R}^d$ ($\mathbb{R}^{nd}$ in the case of a time dependent system) is the permissible output space and $\mathcal{X} \subset \mathbb{R}^l$ is the permissible input space. That is,

$$\boldsymbol{\eta}(\boldsymbol{\xi}) = \boldsymbol{y} = (u(\boldsymbol{x}^{(1)}; \boldsymbol{\xi}), \ldots, u(\boldsymbol{x}^{(d)}; \boldsymbol{\xi}))^T. \tag{3.3}$$

(or in the form of Eq. (3.2) for a time-dependent system) for an arbitrary input $\boldsymbol{\xi}$. The goal of data-driven 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 *et al.* [27] 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.* [1] but in that case the outputs were a linear combination of PCA basis vectors with coefficients treated as independent univariate GPs indexed by $\boldsymbol{\xi}$. Since the method in [1] is closely related to our method and is used for the purposes of comparison, it is described in detail in the following section.

3.2 Principal component analysis based GPE surrogate modelling for simulators

3.2.1 Principal component analysis on the output space

The given training points $\{\boldsymbol{y}^{(j)}\}_{j=1}^m$ (values of $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$ at the design points) lie on a surface $\mathcal{M}$, which is a subset of the high dimensional Euclidean space $\mathbb{R}^d$. Without

loss of generality, one could assume the data is centred. Our aim is to approximate the random field $\boldsymbol{\eta}(\boldsymbol{\xi})$ using a deterministic basis for $\mathbb{R}^d$ and coefficients that are independent *scalar* Gaussian processes[1] indexed by the inputs $\boldsymbol{\xi}$.

Higdon *et al.* [1] used a linear subspace of $\mathbb{R}^d$, formed by the span (all linear combinations) of a small number of the basis vectors, to approximate all points in $\mathcal{M}$. This was achieved using principal component analysis (PCA) [79] applied to the observed data, i.e., a linear transformation $\boldsymbol{z} = \mathbf{V}^T\boldsymbol{y}$ of points $\boldsymbol{y} \in \mathcal{M}$, in which the $d \times d$ matrix $\mathbf{V} = [\boldsymbol{v}_1 \dots \boldsymbol{v}_d]$ is orthogonal and the principal components z_i, $i = 1, \dots, d$, of $\boldsymbol{z}$ are uncorrelated random variables. The d orthornormal principal directions $\boldsymbol{v}_i$ form a basis for $\mathbb{R}^d$ so that $z_i = \boldsymbol{v}_i^T\boldsymbol{y}$. The directions are chosen such that the z_i exhibit decreasing variance with i. Thus, it is easy to set:

$$\boldsymbol{v} = \underset{\boldsymbol{v}\in\mathbb{R}^d}{\arg\max}\, \mathbb{V}\mathrm{ar}[\boldsymbol{v}^T\boldsymbol{y}] = \underset{\boldsymbol{v}\in\mathbb{R}^d}{\arg\max}\, \boldsymbol{v}^T\boldsymbol{\Sigma}\boldsymbol{v} \quad \text{s.t.} \quad \boldsymbol{v}^T\boldsymbol{v} = 1, \tag{3.4}$$

in which:

$$\boldsymbol{\Sigma} = \mathbb{E}[\boldsymbol{y}\boldsymbol{y}^T] = \mathbb{E}[\boldsymbol{\eta}(\boldsymbol{\xi})\boldsymbol{\eta}(\boldsymbol{\xi})^T], \tag{3.5}$$

is the symmetric and positive definite *spatial* variance-covariance matrix; that is, the covariances between coordinates in $\boldsymbol{y}$, which correspond to the values of $\boldsymbol{y}$ at specific points on the spatial grid Ω. The expectation operator $\mathbb{E}[\cdot]$ in this definition is with respect to an underlying probability space consisting of a sample space, events defined on the sample space, and a probability measure. This space is common to the entire family of random vectors $\{\boldsymbol{\eta}(\boldsymbol{\xi}); \boldsymbol{\xi} \in \mathcal{X}\}$ generated by fixing the index $\boldsymbol{\xi}$. In practice, an estimate of $\boldsymbol{\Sigma}$ from the observed data is required, i.e., $\{\boldsymbol{y}^{(j)}\}_{j=1}^m$, which requires the assumption that the expected value of $\boldsymbol{\Sigma}$ can be estimated from different realizations of $\boldsymbol{y}^{(j)}$ as one ranges over values of the index. Such an assumption is referred to as homogeneity or ergodicity (in $\boldsymbol{\xi}$).

[1] They are usually referred to as 'Gaussian random fields' when $l > 1$, but the conventional term in the GPR literature is 'Gaussian process' even in this case. This convention is adopted from here on.

The constrained maximization problem (3.4) is equivalent to the eigenvalue problem $\boldsymbol{\Sigma}\boldsymbol{v} = \lambda\boldsymbol{v}$, which yields orthonormal eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$ and corresponding real, positive eigenvalues $\lambda_1, \ldots, \lambda_d$, arranged such that $\lambda_1 > \cdots > \lambda_d$. The principal directions are given by the eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$. The variance of z_i is given by (noting that $\mathbb{E}[z_i] = 0$):

$$\mathbb{V}\text{ar}[z_i] = \mathbb{V}\text{ar}[\boldsymbol{v}_i^T\boldsymbol{y}] = \boldsymbol{v}_i^T\mathbb{E}[\boldsymbol{y}\boldsymbol{y}^T]\boldsymbol{v}_i \quad = \boldsymbol{v}_i^T\boldsymbol{\Sigma}\boldsymbol{v}_i = \boldsymbol{v}_i^T\lambda_i\boldsymbol{v}_i = \lambda_i, \tag{3.6}$$

and thus decreases with i. Moreover:

$$\mathbb{E}[z_i z_j] = \boldsymbol{v}_i^T\boldsymbol{\Sigma}\boldsymbol{v}_j = \lambda_j\boldsymbol{v}_i^T\boldsymbol{v}_j = 0 \quad i \neq j, \tag{3.7}$$

so the components are uncorrelated. Since $\{\boldsymbol{v}_i\}_{i=1}^d$ forms a basis for $\mathbb{R}^d$, a point $\boldsymbol{y} \in \mathcal{M}$ can be expressed as:

$$\boldsymbol{y} = \sum_{i=1}^{d} z_i\boldsymbol{v}_i = \sum_{i=1}^{d}(\boldsymbol{v}_i^T\boldsymbol{y})\boldsymbol{v}_i, \tag{3.8}$$

An r-dimensional approximation $\boldsymbol{y}_r$ of $\boldsymbol{y}$ is given by truncating the expansion at the first r coefficients. Thus, PCA maps points $\boldsymbol{y} \in \mathcal{M}$ to points $\boldsymbol{y}_r = \boldsymbol{\eta}_r(\boldsymbol{\xi}) \in \mathcal{M}_r$, where $\mathcal{M}_r$ is a linear subspace of $\mathbb{R}^d$ formed by the span of $\{\boldsymbol{v}_i\}_{i=1}^r$. By the properties of the PCA basis and the coefficients, it can be demonstrated [79] that:

$$\mathbb{E}\left[\|\boldsymbol{y} - \boldsymbol{y}_r\|^2\right] = \sum_{i=r+1}^{d}\lambda_i, \tag{3.9}$$

which provides a method for selecting the value of r. For example:

$$\sum_{i=r+1}^{d}\lambda_i / \sum_{i=1}^{d}\lambda_i > \text{tol}, \tag{3.10}$$

for some tolerance tol.

The coefficients $z_i = \boldsymbol{v}_i^T \boldsymbol{y} = \boldsymbol{v}_i^T \boldsymbol{\eta}(\boldsymbol{\xi})$ depend on $\boldsymbol{\xi}$. An *ansatz* is assumed therefore that they are realizations $z_i(\boldsymbol{\xi})$ of scalar, uncorrelated random fields. if the assumptions that these random fields are GPs are valid, they are mutually independent. Thus, the reduced-dimensional approximation is defined as follows:

$$\begin{aligned} \boldsymbol{y}_r = \boldsymbol{\eta}_r(\boldsymbol{\xi}) = \textstyle\sum_{i=1}^{d} z_i(\boldsymbol{\xi}) \boldsymbol{v}_i = \sum_{i=1}^{d} (\boldsymbol{v}_i^T \boldsymbol{y}) \boldsymbol{v}_i \\ = \mathbf{V}_r (z_1(\boldsymbol{\xi}), \ldots, z_r(\boldsymbol{\xi}))^T = \mathbf{V}_r \boldsymbol{z}_r(\boldsymbol{\xi}), \end{aligned} \tag{3.11}$$

where $\mathbf{V}_r = [\boldsymbol{v}_1 \ldots \boldsymbol{v}_r]$. The approximations $\boldsymbol{y}_r$ lie in the r-dimensional subset $\mathcal{M}_r$ of $\mathbb{R}^d$. Let $\mathbf{Y} = [\boldsymbol{y}^{(1)} \cdots \boldsymbol{y}^{(m)}]$. To implement PCA in practice, the *sample* covariance matrix is implemented as followed:

$$\mathbf{C} = (1/m) \mathbf{Y} \mathbf{Y}^T, \tag{3.12}$$

to approximate $\boldsymbol{\Sigma}$ based on the given points $\boldsymbol{y}^{(j)} \in \mathcal{M}$. An eigendecomposition $\mathbf{C} = \mathbf{V} \boldsymbol{\Lambda} \mathbf{V}^T$ yields the decreasing set of eigenvalues $\lambda_i, \ldots, \lambda_d$, which are entries of the diagonal matrix $\boldsymbol{\Lambda}$, and the principal directions $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$, which are the columns of $\mathbf{V}$. It is important to realize that these eigenvalues and eigenvectors are approximations to the true values and the accuracy of the approximation depends on the number of training points m and their distribution on $\mathcal{M}$. For any point $\boldsymbol{y}^{(j)} \in \mathcal{M}$ in the data set one has an approximation:

$$\begin{aligned} \boldsymbol{y}_r^{(j)} = \boldsymbol{\eta}_r(\boldsymbol{\xi}^{(j)}) = \mathbf{V}_r \boldsymbol{z}_r(\boldsymbol{\xi}^{(j)}) \\ z_i(\boldsymbol{\xi}^{(j)}) = \boldsymbol{v}_i^T \boldsymbol{y}^{(j)}. \end{aligned} \tag{3.13}$$

The approximation $\boldsymbol{\eta}_r(\boldsymbol{\xi}) = \mathbf{V}_r \boldsymbol{z}_r(\boldsymbol{\xi})$ can also be interpreted as a *linear model of*

coregionalization (LMC). The cross-covariance matrix function takes the form:

$$\begin{aligned}\mathbb{C}\text{ov}\left(\boldsymbol{\eta}_r(\boldsymbol{\xi}), \boldsymbol{\eta}_r(\boldsymbol{\xi}')\right) &= \mathbb{E}\left[(\boldsymbol{\eta}_r(\boldsymbol{\xi}) - \mathbb{E}[\boldsymbol{\eta}_r(\boldsymbol{\xi})])(\boldsymbol{\eta}_r(\boldsymbol{\xi}') - \mathbb{E}[\boldsymbol{\eta}_r(\boldsymbol{\xi}')])^T\right] \\ &= \mathbf{V}_r \text{diag}(c_1(\boldsymbol{\xi}, \boldsymbol{\xi}'), \ldots, c_r(\boldsymbol{\xi}, \boldsymbol{\xi}'))\mathbf{V}_r^T,\end{aligned} \tag{3.14}$$

where:

$$c_i(\boldsymbol{\xi}, \boldsymbol{\xi}') = \mathbb{E}[(z_i(\boldsymbol{\xi}) - \mathbb{E}[z_i(\boldsymbol{\xi})])(z_i(\boldsymbol{\xi}') - \mathbb{E}[z_i(\boldsymbol{\xi}')])] \tag{3.15}$$

is the covariance function for $z_i(\boldsymbol{\xi})$. If one were to assume that the $z_i(\boldsymbol{\xi})$ are i.i.d., the covariance functions are identical, $c_i(\boldsymbol{\xi}, \boldsymbol{\xi}') = c(\boldsymbol{\xi}, \boldsymbol{\xi}')$, $\forall i$, yielding a cross-covariance matrix function $c(\boldsymbol{\xi}, \boldsymbol{\xi}')\mathbf{V}_r\mathbf{V}_r^T$. This is a *separable* model, as used by Conti and O'Hagan [27] (although in that case the matrix $\mathbf{V}_r$ is an arbitrary square root of a valid covariance matrix that it is integrated out of the posterior distribution). The model used here has a richer covariance structure (each of the z_i has a distinct covariance function) and simultaneously offers a justified reduction in dimensionality.

In summary, rather than approximating the outputs in $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi}) \in \mathcal{M}$, the reduced-dimensional approximations in $\boldsymbol{y}_r = \boldsymbol{\eta}_r(\boldsymbol{\xi}) \in \mathcal{M}_r$ is approximated. By our assumptions, the coefficients $z_i(\boldsymbol{\xi})$, $i = 1, \ldots, r$ are realizations of mutually independent GPs, so approximation of them separately for a chosen value of $\boldsymbol{\xi}$ using GP regression is adopted.

3.2.2 GPE for field high-dimensional outputs

One could easily replace the problem of learning $\boldsymbol{\eta}(\boldsymbol{\xi})$ with the problem of learning a reduced-dimensional approximation $\boldsymbol{\eta}_r(\boldsymbol{\xi})$. This is equivalent to learning the values of $z_i(\boldsymbol{\xi})$, $i = 1, \ldots, r$ defined by equation (3.11). A GP prior is placed over each of these coefficients and the data $z_i(\boldsymbol{\xi}^{(j)})$, $i = 1, \ldots, r$, $j = 1, \ldots, m$ (see equation (3.13)) are provided by PCA on the data matrix $\mathbf{Y} = [\boldsymbol{y}^{(1)} \cdots \boldsymbol{y}^{(m)}]$. For each $i = 1, \ldots, r$, the known coefficients $\{z_i(\boldsymbol{\xi}^{(j)})\}_{j=1}^m$ are centred by subtracting the mean $\overline{z}_i = \sum_{i=1}^m z_i(\boldsymbol{\xi}^{(j)})$, to obtain $\widetilde{z}_i^{(j)} = z_i(\boldsymbol{\xi}^{(j)}) - \overline{z}_i$. The coefficients $\widetilde{z}_i(\boldsymbol{\xi}) = z_i(\boldsymbol{\xi}) - \overline{z}_i$

are then treated as realizations of mutually independent, zero-mean GPs, with prior distribution:

$$\widetilde{z}_i(\boldsymbol{\xi})|\boldsymbol{\theta}_i \sim \mathcal{GP}\left(0, c_i(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}_i)\right), \tag{3.16}$$

where $c_i(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}_i)$ is the covariance function, with hyperparameters $\boldsymbol{\theta}_i$. Equivalently, $z_i(\boldsymbol{\xi})|\boldsymbol{\theta}_i \sim \mathcal{GP}\left(\overline{z}_i, c_i(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}_i)\right)$.

Scalar GPR as outlined in section 2.2.1 is then performed *separately* for each $\widetilde{z}_i(\boldsymbol{\xi})$, $i = 1, \ldots, r$ by setting:

$$\begin{aligned} \widetilde{z}_i(\boldsymbol{\xi}) &= \zeta(\boldsymbol{\xi}), \\ \boldsymbol{t} = \boldsymbol{t}_i &= \{\widetilde{z}_i^{(j)}\}_{j=1}^m, \end{aligned} \tag{3.17}$$

with inputs $\boldsymbol{\xi}^{(j)}$, $j = 1, \ldots, m$. The posterior distribution:

$$\widetilde{z}_i(\boldsymbol{\xi})|\boldsymbol{t}_i, \boldsymbol{\theta}_i \sim \mathcal{GP}\left(\widetilde{m}_i'(\boldsymbol{\xi}; \boldsymbol{\theta}), c_i'(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}_i)\right), \tag{3.18}$$

is given by equation (2.8). The posterior mean $\widetilde{m}_i'(\boldsymbol{\xi})$ and covariance function $c_i'(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}_i)$ are given by $m'(\boldsymbol{\xi})$ and $c'(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta})$ in equations (2.8). The posterior distribution for $z_i(\boldsymbol{\xi})$ is:

$$z_i(\boldsymbol{\xi})|\boldsymbol{t}_i, \boldsymbol{\theta}_i \sim \mathcal{GP}\left(m_i'(\boldsymbol{\xi}; \boldsymbol{\theta}), c_i'(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}_i)\right), \tag{3.19}$$

where:

$$m_i'(\boldsymbol{\xi}) = \mathbb{E}[z_i(\boldsymbol{\xi})] = \mathbb{E}[\widetilde{z}_i(\boldsymbol{\xi}) + \overline{z}_i] = \widetilde{m}_i'(\boldsymbol{\xi}) + \overline{z}_i, \tag{3.20}$$

Defining $\boldsymbol{z}_r(\boldsymbol{\xi}) = (z_1(\boldsymbol{\xi}), \ldots, z_r(\boldsymbol{\xi}))^T$, one obtains:

$$\mathbb{E}[\boldsymbol{z}_r(\boldsymbol{\xi})] = (m_1'(\boldsymbol{\xi}), \ldots, m_r'(\boldsymbol{\xi}))^T = \boldsymbol{m}'(\boldsymbol{\xi}). \tag{3.21}$$

The predicted variance of each coefficient is:

$$\mathbb{V}\mathrm{ar}(z_i(\boldsymbol{\xi})) = \mathbb{V}\mathrm{ar}(\widetilde{z}_i(\boldsymbol{\xi})) = c_i'(\boldsymbol{\xi}, \boldsymbol{\xi}; \boldsymbol{\theta}). \tag{3.22}$$

The distribution over the model outputs is given by:

$$\boldsymbol{y}_r = \boldsymbol{\eta}_r(\boldsymbol{\xi}) = \sum_{i=1}^{r} z_i(\boldsymbol{\xi}) \boldsymbol{v}_i = \mathbf{V}_r \boldsymbol{z}_r(\boldsymbol{\xi}), \tag{3.23}$$

with expected value (vector mean function for the distribution (3.23)):

$$\mathbb{E}[\boldsymbol{y}_r] = \mathbb{E}[\boldsymbol{\eta}_r(\boldsymbol{\xi})] = \sum_{i=1}^{r} m_i'(\boldsymbol{\xi}) \boldsymbol{v}_i = \mathbf{V}_r \boldsymbol{m}'(\boldsymbol{\xi}), \tag{3.24}$$

and predictive variance:

$$\begin{aligned} \mathbb{V}\mathrm{ar}[\boldsymbol{y}_r] &= \mathbb{V}\mathrm{ar}\left(\mathbf{V}_r \boldsymbol{z}_r(\boldsymbol{\xi})\right) = \mathbf{V}_r \mathbb{V}\mathrm{ar}\left(\boldsymbol{z}_r(\boldsymbol{\xi})\right) \mathbf{V}_r^T \\ &= \mathbf{V}_r \,\mathrm{diag}\left(c_1'(\boldsymbol{\xi}, \boldsymbol{\xi}; \boldsymbol{\theta}), \ldots, c_r'(\boldsymbol{\xi}, \boldsymbol{\xi}; \boldsymbol{\theta})\right)\right) \mathbf{V}_r^T, \end{aligned} \tag{3.25}$$

by virtue of the fact that $\mathbb{C}\mathrm{ov}\left(z_i(\boldsymbol{\xi}), z_j(\boldsymbol{\xi})\right) = 0$ for $i \neq j$. The process is summarized in the pseudocode given in Algorithm 1.

Algorithm 1 GPE for field outputs.

1: Perform PCA on $\mathbf{Y} = [\boldsymbol{y}^{(1)} \cdots \boldsymbol{y}^{(m)}]$

- Eigenvectors and eigenvalues: $(\boldsymbol{v}_i, \lambda_i)$
- $\mathbf{V}_r \leftarrow [\boldsymbol{v}_1 \dots \boldsymbol{v}_r]$
- $\boldsymbol{y}_r^{(j)} \leftarrow \mathbf{V}_r(z_1(\boldsymbol{\xi}^{(j)}), \dots, z_r(\boldsymbol{\xi}^{(j)}))^T$
- $z_i(\boldsymbol{\xi}^{(j)}) \leftarrow \boldsymbol{v}_i^T \boldsymbol{y}^{(j)}, \quad i = 1, \dots, r,\ j = 1, \dots, m$

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

- $\eta(\boldsymbol{\xi}^{(j)}) \leftarrow z_i(\boldsymbol{\xi}^{(j)})$
- $\boldsymbol{t} \leftarrow (z_i(\boldsymbol{\xi}^{(1)}), \dots, z_i(\boldsymbol{\xi}^{(m)}))^T$

Perform GPR:

- $\boldsymbol{\theta}_{MLE} \leftarrow \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{t}|\boldsymbol{\theta})$
- (equation (2.8)) $\quad \mathbb{E}[z_i(\boldsymbol{\xi})] \leftarrow \mathbb{E}[\zeta(\boldsymbol{\xi})] \quad \mathbb{V}\text{ar}(z_i(\boldsymbol{\xi})) \leftarrow \mathbb{V}\text{ar}(\zeta(\boldsymbol{\xi}))$

3: **end for**

4: Prediction

- $\boldsymbol{y}_r = \boldsymbol{\eta}_r(\boldsymbol{\xi}) \leftarrow \sum_{i=1}^{r} z_i(\boldsymbol{\xi}) \boldsymbol{v}_i = \mathbf{V}_r \boldsymbol{z}_r(\boldsymbol{\xi})$
- $\mathbb{E}[\boldsymbol{y}_r] \leftarrow \sum_{i=1}^{d} m_i'(\boldsymbol{\xi}) \boldsymbol{v}_i$
- $\mathbb{V}\text{ar}\,(\boldsymbol{y}_r) \leftarrow \mathbf{V}_r \,\text{diag}\,(c_1'(\boldsymbol{\xi}, \boldsymbol{\xi}; \boldsymbol{\theta}), \dots, c_r'(\boldsymbol{\xi}, \boldsymbol{\xi}; \boldsymbol{\theta}))\, \mathbf{V}_r^T$

Note that each of the coefficients is approximated without making an i.i.d. assumption, i.e., they come from independent GPs with *different* covariance parameters. For complex response surfaces, PCA (and other linear methods such as multidimensional scaling) may lead to poor results, and in many cases it could fail altogether. A simple example is illustrated in Fig. 3.1, which shows low-dimensional representations of data lying on a 'swiss roll' in $\mathbb{R}^3$. Fig. 3.1 (a) shows 2000 points randomly sampled from a swiss roll manifold, while Fig. 3.1 (b) shows the approximation of the data set using 2 principal components in PCA. Fig. 3.1 (c) shows the reconstruction using only 1 component (defined later) in Isomap. Clearly, this simple data set cannot be approximately well by a linear subspace of $\mathbb{R}^3$ (further examples, comparisons and detailed analyses can be found in. e.g., [119–121]). Replacing PCA with a nonlinear dimensionality reduction method is, therefore, a natural extension, which forms the motivation for the method developed below.

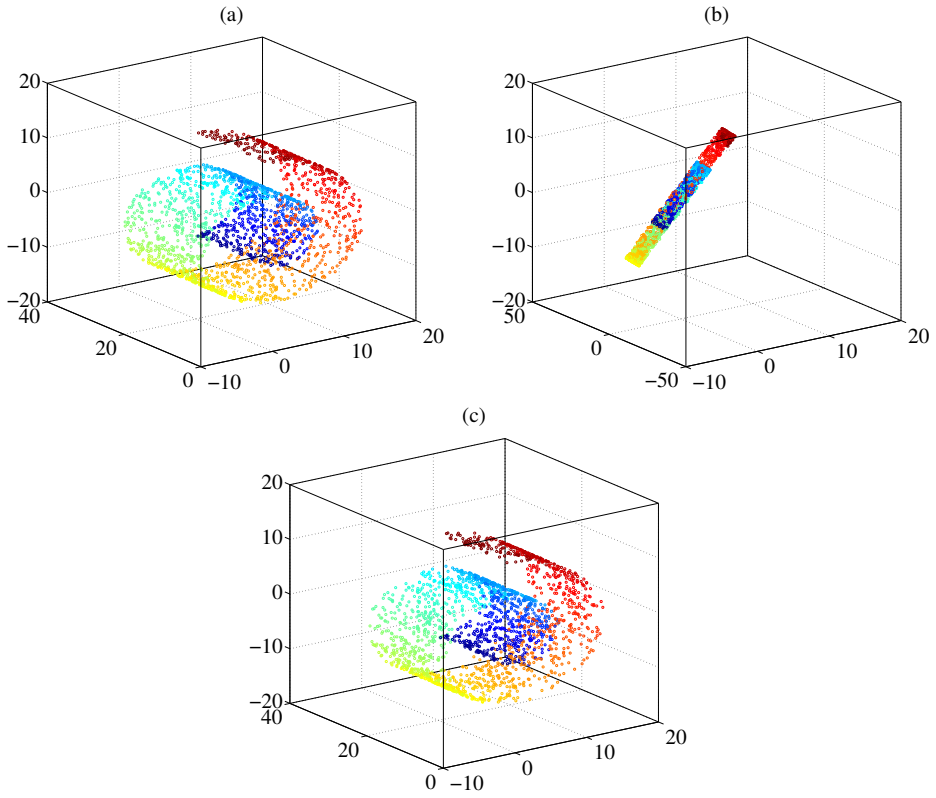

Figure 3.1: Swiss roll data and reconstruction. Fig.(a) is the original Swiss roll data. Fig.(b) is reconstructed data using PCA with 2 preserved dimensions. Fig.(c) is reconstructed data via Isomap with 2 preserved dimensions.

3.3 Dimensionality reduction via Isomap

3.3.1 Multidimensional scaling (MDS)

Classical MDS, which was motivated by the work of [80], provides a low-dimensional Euclidean space representation of a data set that lies in a high-dimensional ambient space. It does so by relating the distances δ_{ij} between data points in the low-dimensional Euclidean space to dissimilarities d_{ij} between the data points in the ambient space. In 'classical scaling', the dissimilarities are Euclidean distances and the Euclidean distances in the low-dimensional space satisfy $\delta_{ij} = d_{ij}$ (in a least squares sense). This can be viewed as an approximate *isometric embedding* of the data in the low-dimensional space. Let $\mathbf{D} = [d_{ij}]$ (dissimilarity matrix) denote the

matrix of dissimilarities between data points $\boldsymbol{y}^{(i)} \in \mathbb{R}^d$, $i = 1 \ldots, m$. In classical scaling:

$$\delta_{ij} = ||\boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)}|| = d_{ij}, \tag{3.26}$$

for the representations $\boldsymbol{z}^{(i)}$, $i = 1, \ldots, m$ of $\boldsymbol{y}^{(i)}$ in the low dimensional Euclidean space ($||\cdot||$ is the standard Euclidean norm). Applying the centering matrix $\mathbf{H} = \mathbf{I} - (1/m)\mathbf{1}\mathbf{1}^T$ to the matrix $\mathbf{Z}$ with column corresponding to $\boldsymbol{z}^{(i)}$ yields:

$$\widetilde{\mathbf{Z}} = \mathbf{Z}\mathbf{H} = \left[\widetilde{\boldsymbol{z}}^{(1)} \ldots \widetilde{\boldsymbol{z}}^{(m)}\right], \tag{3.27}$$

in which $\widetilde{\boldsymbol{z}}^{(i)} = \boldsymbol{z}^{(i)} - \overline{\boldsymbol{z}}$, with $\overline{\boldsymbol{z}} = \sum_{i=1}^{m} \boldsymbol{z}^{(i)}$. It is shown by [95] that:

$$-\frac{1}{2}\mathbf{H}(\mathbf{D} \circ \mathbf{D})\mathbf{H} = \widetilde{\mathbf{Z}}^T\widetilde{\mathbf{Z}} = \mathbf{K}, \tag{3.28}$$

in which the right-hand side is the matrix of inner products of centred data points in the low-dimensional space, i.e., a centred kernel matrix ($\circ$ denotes a Hadamard product). The spectral decomposition of $\mathbf{K}$ is $\mathbf{K} = \mathbf{V}'\Lambda\mathbf{V}'^T$, where:

$$\Lambda = \mathrm{diag}(\lambda_1, \ldots \lambda_d) \in \mathbb{R}^{d \times d}, \quad V' = [\boldsymbol{v}'_1, \ldots \boldsymbol{v}'_d] \in \mathbb{R}^{m \times d}. \tag{3.29}$$

The non-zero eigenvalues λ_i, $i = 1, \ldots, d$, are arranged in a non-increasing order and the corresponding eigenvectors $\boldsymbol{v}'_i \in \mathbb{R}^m$ are normalized. This procedure yields a (non-unique) representation $\widetilde{\mathbf{Z}} = \boldsymbol{\Lambda}^{1/2}\mathbf{V}'^T \in \mathbb{R}^{d \times m}$ of the data. Dimensionality reduction (i.e., the embedding of the data in an r dimensional manifold of $\mathbb{R}^d$) is achieved by selecting the first r eigenvectors corresponding to the largest r eigenvalues, i.e.:

$$\widetilde{\mathbf{Z}}_r = \Lambda_r^{1/2}\mathbf{V}_r'^T, \tag{3.30}$$

in which:

$$\begin{aligned} \mathbf{V}'_r &= (\boldsymbol{v}'_1, \dots \boldsymbol{v}'_r), \\ \boldsymbol{\Lambda}_r &= \text{diag}(\lambda_1, \dots \lambda_r), \\ \widetilde{\mathbf{Z}}_r &= (\widetilde{z}_1, \dots, \widetilde{z}_m). \end{aligned} \tag{3.31}$$

The columns $\widetilde{\boldsymbol{z}}_r^{(i)}$ of $\widetilde{\mathbf{Z}}_r$ are the low-dimensional representations of the data points. When Euclidean distances are used for the dissimilarities, MDS is equivalent to PCA , the latter of which computes a representation $\mathbf{Z}_r = \mathbf{V}_r^T \mathbf{Y} \in \mathbb{R}^{r \times m}$, where $\mathbf{V}_r$ is formed from the first r eigenvectors $\boldsymbol{v}_i$, $i = 1, \cdots, d$, of the sample covariance matrix (as columns) corresponding to the largest r eigenvalues. These eigenvalues are identical to those of the kernel matrix. The singular value decomposition of $\mathbf{Y}$ is given by $\mathbf{Y} = \mathbf{V}\boldsymbol{\Lambda}^{1/2}\mathbf{V}'^T$, where $\mathbf{V} = [\boldsymbol{v}_1 \dots \boldsymbol{v}_d]$, which gives:

$$\mathbf{Z}_r = \mathbf{V}_r^T(\mathbf{V}\Lambda^{1/2}\mathbf{V}'^T) = \Lambda_r^{1/2}\mathbf{V}_r'^T = \widetilde{\mathbf{Z}}_r. \tag{3.32}$$

In other words, the coordinates of $\widetilde{\boldsymbol{z}}_r^{(i)}$, $i = 1, \dots, m$, computed by classical scaling are identical to the first r coordinates using the principal directions $\boldsymbol{v}_i$ as a basis for $\mathbb{R}^d$. The equivalence also arises from the least-squares optimality of both methods in terms of reconstructing both data sets in a new orthogonal basis.

3.3.2 Isomap and kernel Isomap

Classical metric MDS is a linear method and in common with PCA it will fail to accurately represent many surfaces. To overcome this weakness, Tenenbaum *et al.* [92] developed the Isomap method, which uses geodesic distances rather than straight line (e.g., Euclidean) distances as the dissimilarities. For neighbouring points, the Euclidean distance offers a decent approximation to the geodesic distance. For far-away points, the geodesic distance can be approximated by finding a shortest path distance. Applying classical MDS to a dissimilarity matrix based on such geodesic distances leads to a low dimensional (approximately isometric) embedding that cap-

tures the intrinsic geometry of the data. The main steps are given below [92]:

Algorithm 2 Isomap

1: Given data points $\boldsymbol{y}^{(i)} \in \mathbb{R}^d$, $i = 1, \ldots, m$, choose neighbour points on the manifold based on Euclidean distances: (i) any point lying within an ϵ ball is considered a neighbour; or (ii) the N (neighbourhood number) closest points are considered neighbours. Distances between non-neighbour points are defined as infinity.

2: Construct the proximity matrix $P = [d_{ij}]$ using the dissimilarities. The proximities between neighbours are equal to their Euclidean distances. Distances between non-neighbour points are computed as the shortest path distances through neighbouring points.

3: Apply MDS on the kernel matrix $\mathbf{K} = -(1/2)\mathbf{H}(\mathbf{P} \circ \mathbf{P})\mathbf{H}$ to obtain a low dimensional representation $\boldsymbol{z}^{(i)} \in \mathbb{R}^r$, $i = 1, \ldots, m$, for some integer $r \ll d$.

The connection between this method and kPCA is worth elucidating ([100, 122, 123]) to explain the significance of the Isomap coordinates. The dissimilarity matrix D can be considered to represent distances between points in a feature space $\mathcal{F}$, i.e.:

$$d_{ij}^2 = (\boldsymbol{\phi}(\boldsymbol{y}^{(i)}) - \boldsymbol{\phi}(\boldsymbol{y}^{(j)}))^T(\boldsymbol{\phi}(\boldsymbol{y}^{(i)}) - \boldsymbol{\phi}(\boldsymbol{y}^{(j)})). \tag{3.33}$$

where $\boldsymbol{\phi} : \mathbb{R}^d \to \mathcal{F}$ is the feature map. A kernel function can be defined as:

$$\Bbbk(i, j) = \boldsymbol{\phi}(\boldsymbol{y}^{(i)})^T\boldsymbol{\phi}(\boldsymbol{y}^{(j)}). \tag{3.34}$$

For an isotropic kernel scaled so that $\Bbbk(i, i) = 1$, one has:

$$d_{ij}^2 = 2 - 2\Bbbk(i, j) \quad \text{or} \quad -\frac{1}{2}(\mathbf{D} \circ \mathbf{D}) = \mathbf{K}' - \mathbf{1}\mathbf{1}^T. \tag{3.35}$$

where $\mathbf{K}'$ is a kernel matrix. The centered kernel matrix K is obtained from

$$-\frac{1}{2}\mathbf{H}(\mathbf{D} \circ \mathbf{D})\mathbf{H} = \mathbf{H}\mathbf{K}'\mathbf{H}. \tag{3.36}$$

Classical scaling using this centred kernel matrix (derived from the dissimilarity

matrix) is then equivalent to finding coordinates in a basis consisting of the normalized eigenvectors of the sample (centred) covariance matrix *in a feature space*, i.e., exactly as in kPCA.

In kPCA, the eigenvectors of the centred covariance matrix in feature-space are generally not known since the feature map is not specified. This covariance matrix eigenproblem, in a possibly infinite dimensional feature space, can be recast as the eigenproblem for the positive definite (centred) kernel matrix. Assuming for simplicity that both matrices have a full set of eigenvectors, using only the eigenvectors of the centred kernel matrix, $\boldsymbol{v}_j$, $j = 1, \ldots, m$, and the specified kernel values, the projections of a data point $\boldsymbol{y}^{(i)} \in \mathbb{R}^d$ onto the new basis formed by the generally unknown eigenvectors of the covariance matrix in feature space, $\boldsymbol{f}_j$, $j = 1, \ldots, \min\{\dim(\mathcal{F}), m\}$, can be computed as follows (full details and a derivation are provided in Chapter 4):

$$\widetilde{z}_j^{(i)} = \boldsymbol{f}_j^T \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)}) = \sum_{l=1}^{m} v_{lj} K_{li} \tag{3.37}$$

in which $\widetilde{\boldsymbol{\phi}}(\boldsymbol{u}^{(i)})$ is the centered value of the data point in feature space, v_{lj} is the lth component of the eigenvector $\boldsymbol{v}_j$ and K_{li} is the dot product (kernel value) between two points $\boldsymbol{y}^{(l)}$ and $\boldsymbol{y}^{(i)}$ centred in feature space.

There is no theoretical guarantee, however, that the kernel matrix is positive semi definite (p.s.d.), which would guarantee that the low-dimensional manifold, onto which the data is mapped, is a Euclidean space (where the distances between mapped points equals to their dissimilarities), and that a feature space exists. Dissimilarity matrices that lead to p.s.d. kernel matrices are said to have an 'exact Euclidean representation'. To overcome the problem of non-Euclidean dissimilarity matrices, [100] developed a variant of Isomap termed *kernel Isomap*, which exhibits greater robustness. The algorithm is described below [123]: Step 4 is related to the additive constant problem, which can be defined in the following way: find a value

Algorithm 3 kernel Isomap

1: Determine the neighbours as in Isomap.
2: Form the proximity matrix using euclidean distance $\mathbf{P} = [d_{ij}]$.
3: Construct the matrices $\mathbf{K} = -(1/2)\mathbf{H}(\mathbf{P} \circ \mathbf{P})\mathbf{H}$ and $\mathbf{J} = -(1/2)\mathbf{HPH}$.
4: Compute the largest eigenvalue, c^*, of the matrix $\widetilde{\mathbf{K}}$:

$$\begin{pmatrix} 0 & 2\mathbf{K} \\ -\mathbf{I} & -4\mathbf{J} \end{pmatrix} \tag{3.38}$$

5: Apply classical scaling on $\widetilde{\mathbf{K}}$ to obtain $\widetilde{\boldsymbol{z}}_r^{(i)} \in \mathbb{R}^r$, $i = 1, \ldots, m$, for some integer $r \ll d$.

of the constant c such that the modified dissimilarities matrices:

$$[\widetilde{d}_{ij}] \equiv [d_{ij} + c(1 - \delta_{ij})], \tag{3.39}$$

have a Euclidean representation for all $c \geq c^*$ (δ_{ij} is the Kronecker-delta function). An exact solution to this problem was given by [124] as the largest eigenvalue of the matrix (3.38).

3.3.3 The Iso-GPE algorithm

For design points $\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, m$, the simulator $\boldsymbol{\eta}(\cdot)$ in Section 3.1 yields outputs $\boldsymbol{y}^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)})$. Analogously to the PCA method, one can view the simulator as instead giving realizations $\widetilde{\boldsymbol{z}}_r^{(i)} \in \mathbb{R}^r$, $i = 1, \ldots, m$, of a random vector. These realizations lie on or near an r-dimensional space (embedding) and are extracted from the simulator outputs $\boldsymbol{y}^{(i)}$ as described in section (3.3.2). For fixed j, univariate GPE is performed on the set of coefficients $\widetilde{z}_j^{(i)}$, with corresponding design inputs $\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, m$, to learn the mapping $\boldsymbol{\xi} \mapsto \widetilde{z}_j$ for a test input $\boldsymbol{\xi}$. In the notation of section 2.2.1, the training points are $y^{(i)} = \widetilde{z}_j^{(i)}$, $i = 1, \ldots, m$, and the test output is $y = \widetilde{z}_j$. This procedure is repeated for $j = 1, \ldots, r$, with $r \ll d$, to yield $\widetilde{\boldsymbol{z}}_r = (\widetilde{z}_1, \ldots, \widetilde{z}_r)^T$, which is a low dimensional representation of the desired output $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$. The predicted coefficients $\widetilde{z}_j$, $j = 1, \ldots, r$, are precisely the coefficients

in an expansion using the PCA basis in a feature space, as demonstrated in the preceding subsection (equation (3.37)). They are therefore uncorrelated and can be treated as independent given the GP-prior assumption. The final step is to find the inverse mapping from the Isomap representation $\widetilde{\boldsymbol{z}}_r \in \mathbb{R}^r$ to the physical space $\boldsymbol{y} \in \mathbb{R}^d$, which is explained below. The Iso-GPE method can then be summarized in the pseudo code given below.

Constructing solutions from the predicted Isomap coordinates

Given the predicted coordinates of a point $\widetilde{\boldsymbol{z}}_r$ for a test input $\boldsymbol{\xi}$ (as described above), the distances between such a point and the other points $\widetilde{\boldsymbol{z}}_r^{(i)}$, $i = 1, \ldots, m$, can be computed. Let $d_{i,*}$ denote the Euclidean distance between $\widetilde{\boldsymbol{z}}_r^{(i)}$ and $\widetilde{\boldsymbol{z}}_r$. Then $d_{i,*}$ approximates the geodesic distance between $\boldsymbol{y}$ (the inverse image of $\widetilde{\boldsymbol{z}}_r$) and $\boldsymbol{y}^{(i)}$ by the nature of Isomap. For neighbourhood points of $\boldsymbol{y}$, these geodesic distances are approximately equal to Euclidean distances. Local linear interpolation can be used to approximate the coordinates of $\boldsymbol{u}$ by using the geodesic distances as weights ([112, 125]). Let $w_i = d_{i,*}$; then $\boldsymbol{y}$ is approximately given by:

$$\boldsymbol{y} = \sum_{i=1}^{N_n} \frac{w_i}{W} \boldsymbol{y}^{(i)}, \tag{3.40}$$

where $W = \sum_{i=1}^{N_n} w_i$ and N_n is the number of nearest neighbours selected for the reconstruction (mapping of $\widetilde{\boldsymbol{z}}_r$ to $\boldsymbol{y}$). The natural choice for N_n is the neighbourhood number N used in Isomap (recall that for neighbourhood points the geodesic distances are set equal to the Euclidean distances in step 1 of Algorithm 2). The accuracy of this procedure relies on: (a) the design-of-experiment (DOE) for selecting the inputs (see section 3.4) to provide 'representative' outputs in the window of input space that is of interest; and (b) the number of neighbours selected for the reconstruction. In some cases, e.g., precipitous changes in the outputs with small changes in the inputs and/or small sample sizes, a more informed (based on prior

information) DOE than uniform sampling may be required, although this problem was not encountered with the examples considered in section 3.4.

Algorithm 4 Iso-GPE Algorithm

1: Select design points $\boldsymbol{\xi}^{(i)} \in \mathcal{X} \subset \mathbb{R}^l$, $i = 1, \ldots, m$, using design of experiments
2: Construct outputs $\boldsymbol{y}^{(i)} = \boldsymbol{\eta}(\boldsymbol{x}^{(i)}) \in \mathbb{R}^d$, $i = 1, \ldots, m$, from the simulator (section 2.2.1)
3: Perform Isomap (Algorithms 2 or 3) on $\boldsymbol{y}^{(i)}$, $i = 1, \ldots, m$, to obtain $\widetilde{\boldsymbol{z}}_r^{(i)} = (\widetilde{z}_1^{(i)}, \ldots, \widetilde{z}_r^{(i)})^T$, $i = 1, \ldots, m$.
4: Select a test point $\boldsymbol{\xi}$ for prediction.
5: **for** $j = 1 : r$ **do**
6: perform scalar GPE on the training set $\widetilde{z}_j^{(i)}$, $\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, m$, to obtain $\widetilde{z}_j$. (find $\boldsymbol{\theta}_{MLE}$ from (2.9) and use (2.8) with $\boldsymbol{t} = (\widetilde{z}_j^{(1)}, \ldots, \widetilde{z}_j^{(m)})^T$ and $\boldsymbol{y} = \widetilde{z}_j$)
7: **end for**
8: Define $\widetilde{\boldsymbol{z}}_r = (\widetilde{z}_1, \ldots, \widetilde{z}_r)^T$ and $w_i^2 = d_{i,*}^2 = ||\widetilde{\boldsymbol{z}}_r^{(i)} - \widetilde{\boldsymbol{z}}_r||$, $i = 1, \ldots, N_n$. Approximate simulator output $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$ using equation (3.40)

3.4 Results and Discussion

The methodology developed in the previous section (Iso-GPE) was applied to data sets from three problems: (i) a steady-state parameterized PDE model, (ii) an unsteady parameterized PDE model and (iii) a parameterized ODE model in time. The results were compared in each case to PCA method, also called HH in this thesis.

Training and Testing. In each case, the data set consisted of 500 data points (vectors), with inputs selected using a Sobol sequence design-of-experiment (DOE) [126], which is specifically designed to generate samples as uniformly as possible over the unit hypercube [127]. 400 points were reserved for testing and different numbers (in an increasing manner) of the remaining 100 points were used for training. A

relative error is defined to measure the generalization error as:

$$\text{Relative error} = \frac{||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2}{||\boldsymbol{y}^{(i)}||^2}, \tag{3.41}$$

where $||\cdot||$ is the standard Euclidean norm. Selection of the design points using DOE is vital for ensuring that an accurate response surface can be constructed from the data [3, 4, 128]. In the examples below, inputs were sampled uniformly. If prior knowledge of the input-output relationship is available, a more sophisticated approach could be used. Since our focus is on the output space dimensionality, and uniform sampling of the inputs suffices for the examples considered, the readers are referred to [3, 4, 127, 128] for detailed discussions on DOE issues, which are common to all emulation methods.

Results are shown for different numbers of components in the two dimensionality reduction methods. In the case of PCA, the first r components are the r principal components corresponding to the r largest eigenvalues of the covariance matrix. In the case of Isomap (or kernel Isomap) the first r components are the r Isomap coordinates corresponding to the r largest eigenvalues of the kernel matrix (corresponding to the r in the previous section and in the Iso-GPE algorithm).

3.4.1 Free convection in porous media

This example concerns subsurface flow in a porous medium driven by density variations that result from temperature changes [129]. The temperature varies from a high value T_h to a low value T_c along the outer edges in a two-dimensional domain $(x_1, x_2) \in \Omega = [0, 10] \times [0, 10]$ (in cm), which is filled with water (see Fig. 3.2). Temperature gradients alter the fluid density and buoyant flow is generated. A Boussinesq buoyancy term in Brinkmann's equation accounts for the lifting force

due to thermal expansion. The model equations are:

$$\begin{gathered}\frac{\mu}{\kappa}\boldsymbol{w} + \nabla p - \nabla \cdot \frac{\mu}{\epsilon}\left(\nabla \boldsymbol{w} + \nabla \boldsymbol{w}^T\right) = \rho \mathbf{g}\beta(T - T_c), \\ \rho C_p \boldsymbol{w} \cdot \nabla T - \nabla \cdot (\lambda \nabla T) = 0.\end{gathered} \tag{3.42}$$

which are combined with the continuity equation $\nabla \cdot \boldsymbol{w} = 0$. T represents temperature, p is pressure, $\mathbf{g}$ denotes gravitational acceleration, ρ is the fluid density at the reference temperature T_c, ϵ and κ are the porosity and permeability of the medium, β is the coefficient of volumetric thermal expansion of the fluid, μ is the dynamic viscosity, λ 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. The Brinkman equations are subject to no-slip conditions on all boundaries.

The model was solved in COMSOL Multiphysics 4.3b ('Free convection in porous media' under the Subsurface Flow (Heat Transfer) module) without modification. The input parameters were $\boldsymbol{\xi} = (\beta, T_h)^T \in [10^{-11}, 10^{-8}] \times [40, 60]$ in units of K^{-1} and $^{\mathrm{o}}\mathrm{C}$ respectively. A total of 500 numerical experiments were performed using a Sobol sequence for sampling the input space. 400 of the data points were reserved for testing. For each input $\boldsymbol{\xi}^{(i)} \in \mathbb{R}^2$, $i = 1, \ldots, 500$, the magnitude $|\boldsymbol{w}|$ of the velocity was recorded on a regular 100×100 square spatial grid. The 10000 points in the 2D spatial domain Ω were re-ordered into vector form (as described in section 3.1) to give data points $\boldsymbol{y}^{(i)} \in \mathbb{R}^d$, where $d = 10000$. In the notation of section 3.1, $|\boldsymbol{w}| = u(\boldsymbol{\xi}; \boldsymbol{x})$. To give some indication of the range of results, Fig. 3.3 shows the $|\boldsymbol{w}|$ fields in 6 different cases. The general trend was an increase in the average value of $|\boldsymbol{w}|$ with increases in β and T_h.

Results. For 40 training points, which are the first 40 of the reserved training dataset, (Tukey) box plots of the relative errors (defined as: $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2 / ||\boldsymbol{y}^{(i)}||^2$) are shown in Fig.s 3.4 (a) and (b), up to 5 components for both HH and Iso-GPE (the horizontal axis is the number of components for the respective method, as

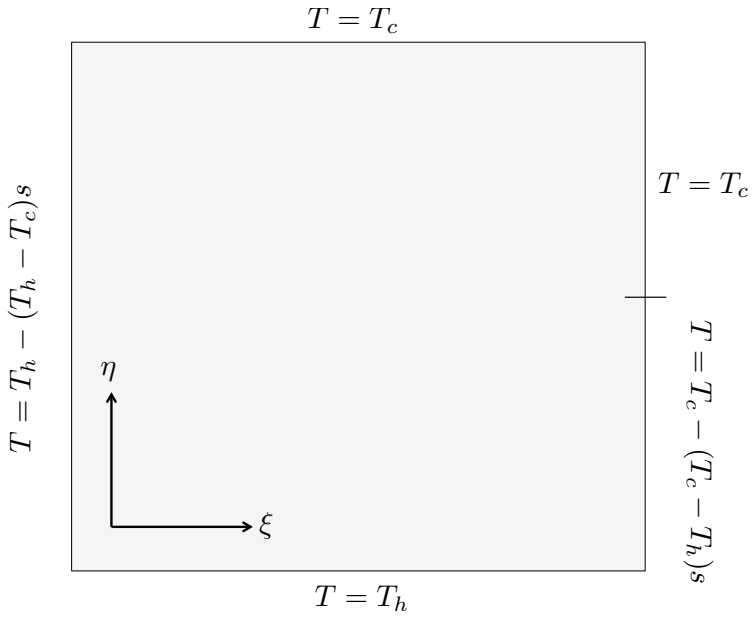

Figure 3.2: Temperature boundary conditions for the free-convection example. δ is a variable that represents the relative length of a boundary segment and goes from 0 to 1 along the segment as x_2 increases. The cut-off shown by the horizontal dash along $x_1 = 10$ cm is located at $x_2 = 1$ cm.

previously defined). On each box, the central line is the median, the lower and upper edges signify the first (Q_1) and third (Q_3) quartiles, and the lower 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. For this number of training points, the two methods exhibit similar accuracy, with little variation in the errors beyond the use of 1 component (linear PCA shows that the first component contains over 95 % of the modal energy). For higher numbers of training points, the Iso-GPE method is superior, as shown in Fig.s 3.4 (c) and (d), which correspond to 80 training points. For the HH, the MLE for the hyperparameters was very sensitive to the algorithm used. The steepest descent algorithm (used for Fig. 3.4) gave the most stable results, while the other methods tested (conjugate gradient, L-BFGS, Hessian free Newton, Barzilai & Borwein, all implemented using the Matlab library *minFunc* (available from http://www.di.ens.fr/ mschmidt/Software/minFunc.html) typically led to complete

failure, irrespective of the initial guess for the hyperparameters. The MLE problem for Iso-GPE was found to be much more robust, in terms of both the method used and the initial guess. Representative examples in the case of 80 training samples are shown in Fig. 3.5. In both cases, 5 components were used. Iso-GPE performs well in both cases, while HH exhibits noticeable quantitative and qualitative differences from the test output in both cases.

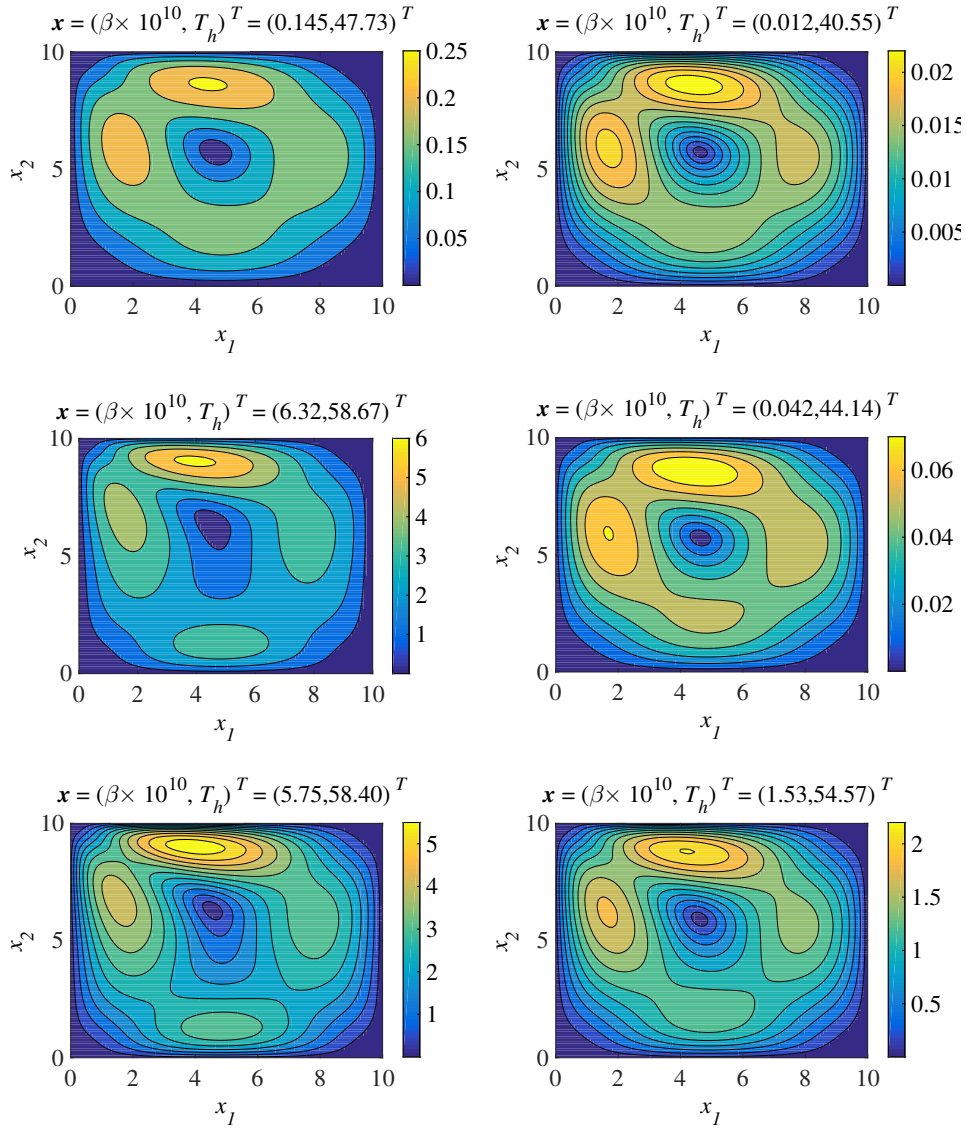

Figure 3.3: The simulator velocity field in 6 different cases as indicated for the free-convection example.

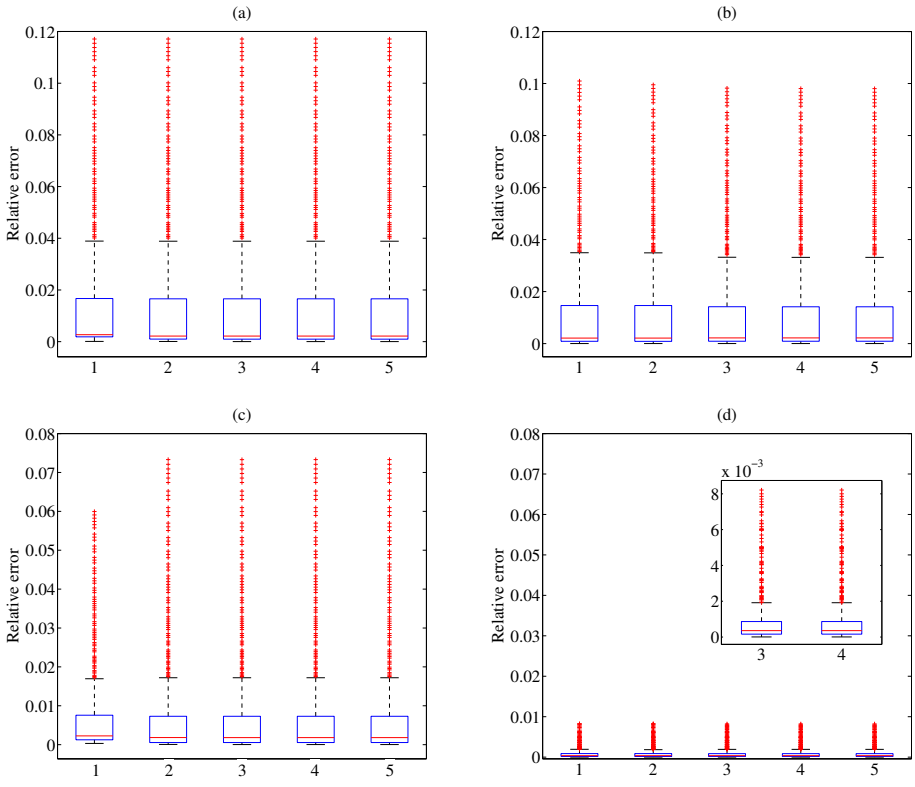

Figure 3.4: Tukey box plots of the relative errors for HH and Iso-GPE in the free-convection example. Fig.(a) and Fig.(b) are HH and Iso-GPE with 40 training point, respectively, while Fig.(c) and Fig.(d) are HH and Iso-GPE with 80 training point

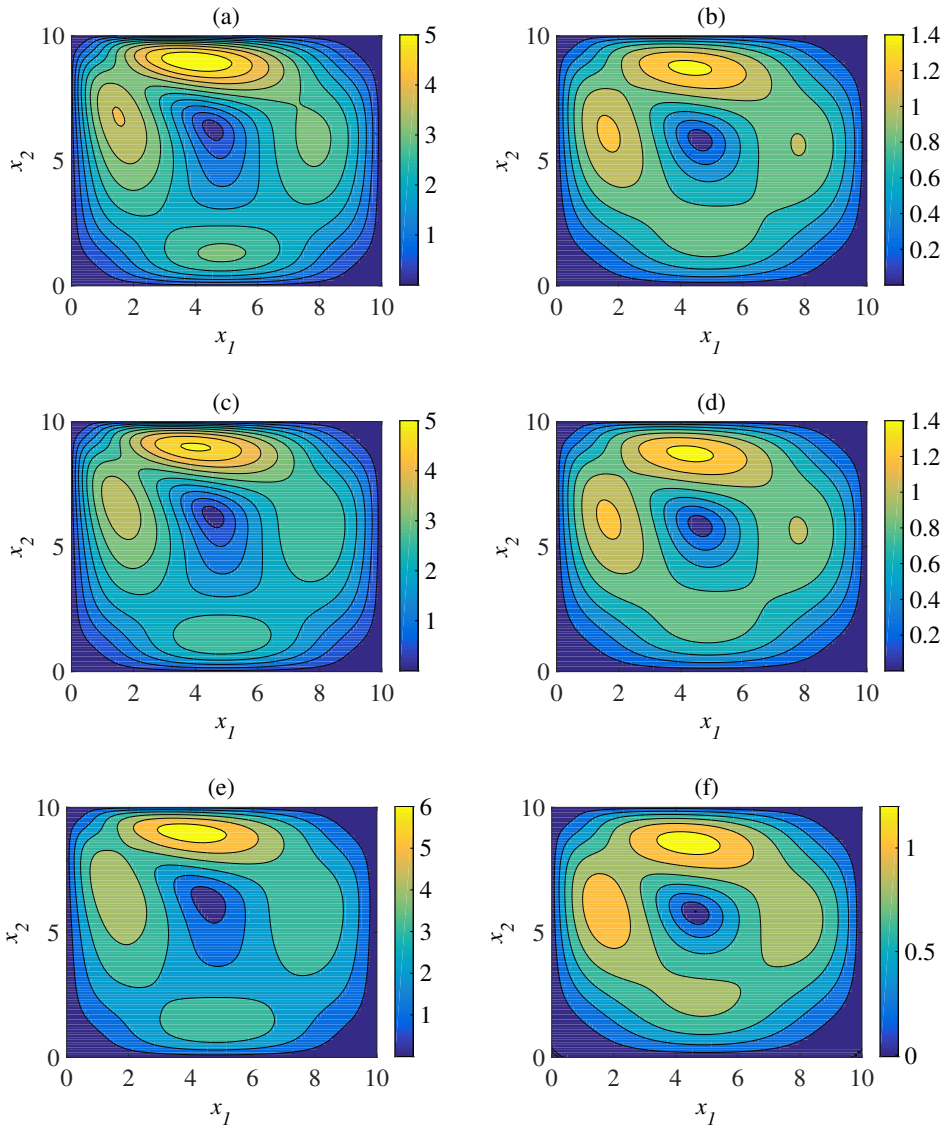

Figure 3.5: Predictions of the velocity field using 80 training points and r = 5 coefficients in the free-convection example for HH and Iso-GPE. Fig.(a) and Fig.(b) are two actual velocity field using the simulator. Fig.(c) and Fig.(d) are the predictions of Iso-GPE corresponding to Fig.(a) and Fig.(b) case, respectively. Fig.(e) and Fig.(f) are the predictions of HH in the same corresponding to Fig.(a) and Fig.(b) case, respectively.

3.4.2 Continuously stirred tank reactor

The temperature profile of the startup phase of a continuously stirred tank reactor (CSTR) used to produce propylene glycol (PrOH) from the reaction of propylene oxide (PrO) with water in the presence of an acid catalyst (H_2SO_4): PrO+$H_2O \xrightarrow{H_2SO_4}$ PrOH is considered. The liquid phase reaction takes place in a CSTR, equipped with a heat-exchanger. Methanol (MeOH) is added to the mixture but does not react. The mass balances for the species are:

$$V_r \frac{dc_i}{dt} = v_f(c_{f,i} - c_i) + \nu_i V_r \mathcal{R}, \qquad i = \text{PrO, MeOH, PrOH, } H_2O, \tag{3.43}$$

in which: c_i is the concentration, ν_i is the stoichiometric coefficient and $c_{f,i}$ is the feed stream concentration of species i, respectively; v_f is the volumetric flow rate; and $\mathcal{R} = Ac_{\text{PrO}}e^{-E/(RT)}$ is the rate of reaction, with activation energy E and pre-exponential factor A (R is the universal gas constant and T is the temperature of the reactor). The energy balance is expressed as follows:

$$\begin{aligned}\sum_i c_i C_{p,i} \frac{dT}{dt} = -H\mathcal{R} + \frac{F_x C_{p,x}(T_x - T)}{V_r}\left(1 - e^{U/(F_x C_{p,x})}\right) \\ + \sum_i \frac{v_f c_{f,i}(h_{f,i} - h_i)}{V_r},\end{aligned} \tag{3.44}$$

in which the specific heat capacity is molar averaged over the specific heat capacities of the individual species ($C_{p,i}$) and H is the enthalpy of reaction. The first term on the right hand side (rhs) represents the heat of reaction. The second term on the rhs represents the heat loss to the heat exchanger, where F_x is the flow rate, $C_{p,x}$ is the specific heat capacity and T_x is the inlet temperature of the heat exchanger medium. U (J s^{-1} K^{-1}) is the heat exchange parameter. The third term on the rhs is the enthalpy change due to the flow of species through the reactor. The molar enthalpy of species i is given by $h_i = C_{p,i}(T - T_{ref}) + h_{i,ref}$, in which $h_{i,ref}$ is the standard heat of formation of i at the reference temperature T_{ref}. The feed stream

molar enthalpies $h_{f,i}$ are calculated similarly. The model is completed by initial values for T and c_i.

Solutions were obtained using the 'cstr startup' model in the 'Chemical Reaction Engineering Module' of COMSOL Multiphysics 4.3b without modification. A total of 500 numerical experiments were performed using a Sobol sequence for sampling the input space. 400 of the data points were reserved for testing. The inputs were defined as $\boldsymbol{\xi} = (T_0, c_0, U)^T \in [297, 360] \times [100, 1500] \times [1000, 10000]$, where T_0 (K) is the initial temperature and c_0 (mol m^{-3}) is the initial concentration of PrO. For each input $\boldsymbol{\xi}^{(i)} \in \mathbb{R}^3$, $i = 1, \ldots, 500$, the temperature was recorded at intervals of 14 s up to 7000 s, yielding 501 values for each of the 500 cases. The 501 values were re-ordered into vector form (as described in Eq.(3.2) to give data points $\boldsymbol{y}^{(i)} \in \mathbb{R}^{nd}$, where $nd = 501$. in the notation of section refsec:problem defin.

Results. Fig. 3.6 shows the boxplots of the relative errors for HH and Iso-GPE up to 5 components for both 40 and 80 training points. The best performance is seen with Iso-GPE. In fact HH failed to provide satisfactory for any number of training points. Despite appearing to give similar results to Iso-GPE for 40 training points, based on Fig.s 3.6 (a) and (b), HH exhibited spurious oscillations. A clear difference in the errors is seen in Fig.s 3.6 (a) and (b), which correspond to 80 training points. Examples of the predictions for each method are shown in Fig. 3.7 (for 80 training points using 5 components) demonstrating the failure of HH. Iso-GPE provided a good fit to the trends and although it exhibited only a reasonably good quantitative fit, the method did not suffer from spurious oscillations. For a higher number of training points up to 140, Iso-GPE exhibited slight improvements in the accuracy of the predictions, while HH continued to fail.

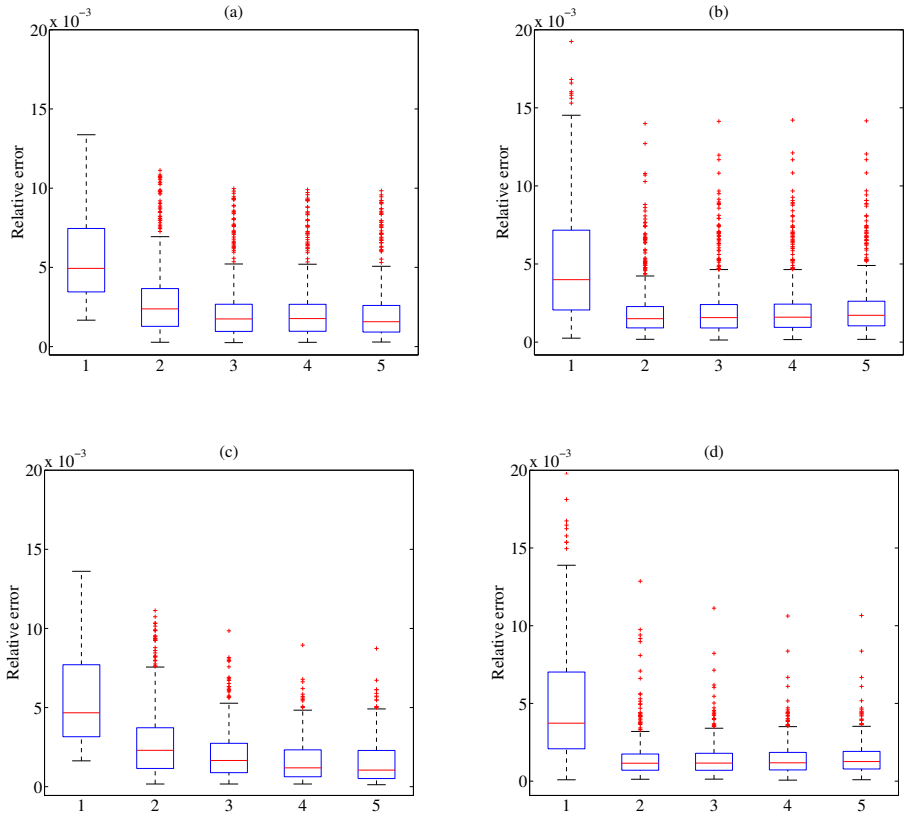

Figure 3.6: Tukey box plots of the relative errors for HH and Iso-GPE in the CSTR example. Fig.(a) and Fig.(b) are HH and Iso-GPE with 40 training point, respectively, while Fig.(c) and Fig.(d) are HH and Iso-GPE with 80 training point

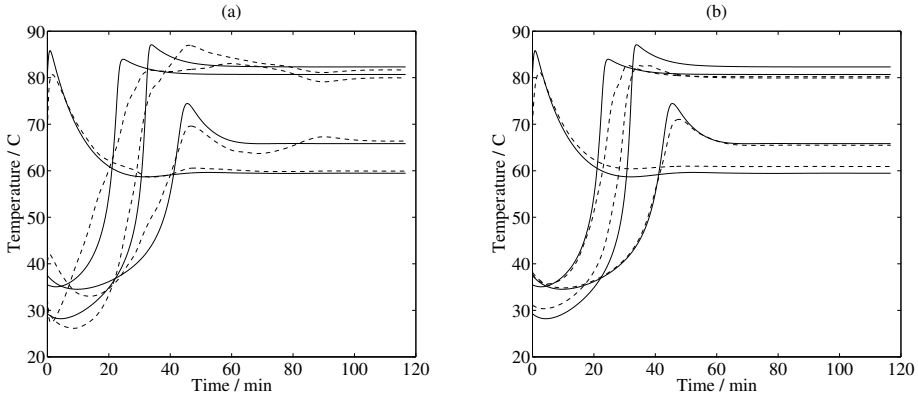

Figure 3.7: Actual response curves and predictions using 80 training points and r = 5 coefficients in the CSTR example for HH (Fig.(a)) and Iso-GPE(Fig.(b)) at 4 different parameter sets. The solid lines are the actual response curves while the dash lines indicate the prediction using emulation.

3.4.3 Metal melting front

A square cavity, $(x_1, x_2) \in \Omega = [0, 10] \times [0, 10]$ (in units of cm), containing both solid and liquid is submitted to a temperature difference between the left and right boundaries. The fluid and solid phases are treated as separate domains sharing a moving melting front. The position of the melting front is calculated according to a Stefan condition ([130]). The liquid domain is governed by the Navier-Stokes equations using a Boussinesq approximation:

$$\begin{gathered} \rho_0 \frac{\partial \boldsymbol{w}}{\partial t} + \rho_0(\boldsymbol{w} \cdot \nabla)\boldsymbol{w} + \nabla p - \nabla \cdot \mu \left(\nabla \boldsymbol{w} + \nabla \boldsymbol{w}^T\right) - \rho \mathbf{g} = 0, \\ \nabla \cdot \boldsymbol{w} = 0, \end{gathered} \tag{3.45}$$

and the heat balances are:

$$\begin{gathered} \rho C_p \frac{\partial T_l}{\partial t} + \rho C_p \boldsymbol{w} \cdot \nabla T_l - \nabla \cdot (\lambda \nabla T_l) = 0, \\ \rho C_p \frac{\partial T_s}{\partial t} - \nabla \cdot (\lambda \nabla T) = 0, \end{gathered} \tag{3.46}$$

in which $\boldsymbol{w}$ is the liquid velocity, T_l (T_s) is the liquid (solid) temperature, μ is the dynamic viscosity, λ is the (common) thermal conductivity of the liquid and solid, C_p is the heat capacity of the liquid at constant pressure, $\mathbf{g}$ denotes gravitational acceleration, ρ_0 is the reference density of the fluid, $\rho = \rho_0 \beta (T_l - T_f)$ is the linearized density, where β is the metal coefficient of thermal expansion, and T_f denotes the fusion temperature of the solid. The liquid-solid front is defined by $\boldsymbol{x} = (x_1, x_2)$, along which $T = T_f$. An energy balance at the melting front is given by [130]:

$$\rho_0 \Delta h_f \frac{\partial a}{\partial t} = \left(1 + \left(\frac{\partial a}{\partial y}\right)^2\right) \left(\lambda \frac{\partial T_s}{\partial y} - \lambda \frac{\partial T_l}{\partial x}\right), \tag{3.47}$$

where Δh_f is the latent heat of fusion. A no slip condition is applied at the other boundaries.

The model was solved in COMSOL Multiphysics 4.3b ('Tin Melting Front'

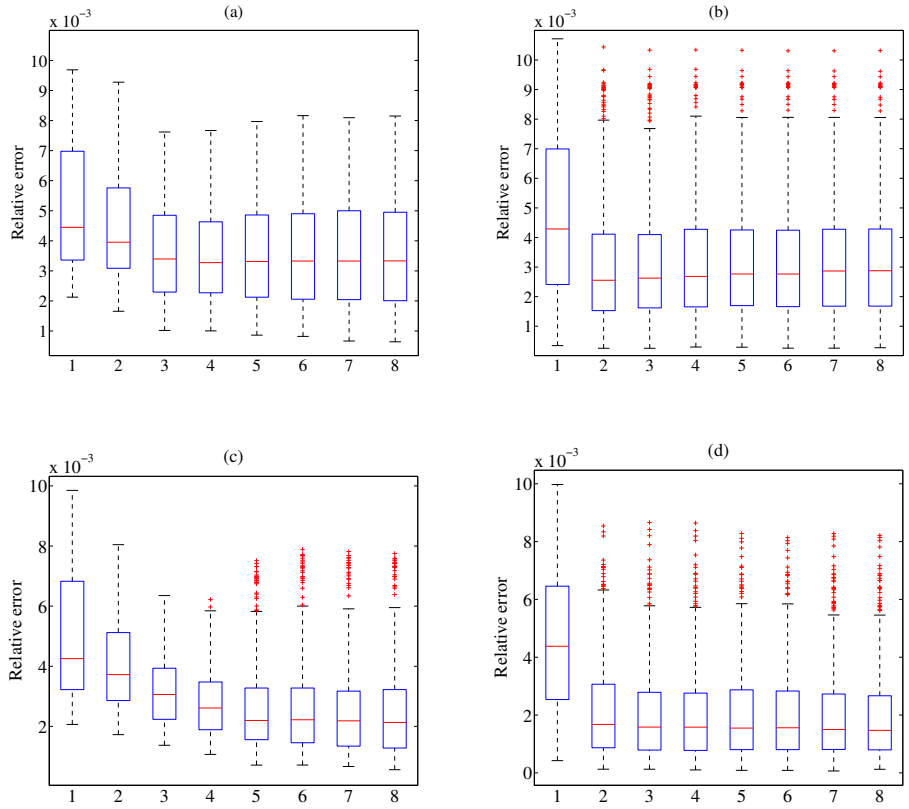

Figure 3.8: Tukey box plots of the relative square errors up to 8 components for Iso-GPE and HH. Fig.(a) and Fig.(b) are result of Iso-GPE and HH with 50 training points while Fig.(a) and Fig.(b) are result of Iso-GPE and HH with 80 training points.

under the Heat Transfer (Phase change) module) without modification. A total of 100 numerical experiments were performed by selecting parameters $\widetilde{\boldsymbol{\xi}}^{(i)} = (\Delta h_f, \lambda)^T \in [200, 1200] \times [30, 100]$, $i = 1, \ldots, 100$, using a Sobol sequence (the units are kJ kg^{-1} and W m^{-1} K^{-1} respectively). For each value of $\widetilde{\boldsymbol{\xi}}^{(i)}$, 5 snapshots of the velocity field were recorded for $t_i = 50 + 50(i-1)$, $i = 1, \ldots, 5$ (in seconds). For each of the 500 snapshots, the magnitude $|\boldsymbol{w}|$ of the velocity was recorded on a regular 100×100 square spatial grid. The 10000 points in the 2D snapshots were re-ordered into vector form (as described in section 3.1) to give data points $\boldsymbol{y}^{(i)} \in \mathbb{R}^d$ as a instantaneous state which is treated as a steady state , where $d = 10000$. In the notation of section 3.1, $|\boldsymbol{w}| = u(\boldsymbol{x}, t; \boldsymbol{\xi})$, time t is treated as an input parameter. This gave a total of 500 data points (400 reserved for testing), with corresponding inputs $\boldsymbol{\xi}^{(i+100(k-1))} = (\widetilde{\boldsymbol{\xi}}^{(i)}, t_k) \in \mathbb{R}^3$, $i = 1, \ldots, 100$, $k = 1, \ldots, 5$.

Results. For 40 training points, box plots of the relative square errors are shown in Fig. 3.8 (a) and (b), up to 8 components for both Iso-GPE and HH. Iso-GPE clearly exhibits the lowest errors of the two methods. The differences in the errors are significant, as demonstrated by the two representative examples shown in Fig. 3.9, corresponding to cases 145 and 301 of the 400 reserved for testing. In both cases, 5 components were used. The performance of Iso-GPE in the first example is very good, while it performs poorly in the second. HH is noticeably worse than Iso-GPE, particularly in the second test, in which the fit is both qualitatively poor and negative values are predicted. For 80 training examples, the boxplots of the relative square errors are shown in Fig. 3.8 (c) and (d), again up to 8 components for both Iso-GPE and HH. Both methods show improvement. Iso-GPE exhibited very good performance with 80 training points in almost all cases, while the fit with HH was still poor in many of the test cases. In fact, significant improvements were seen with Iso-GPE using just 60 training points.

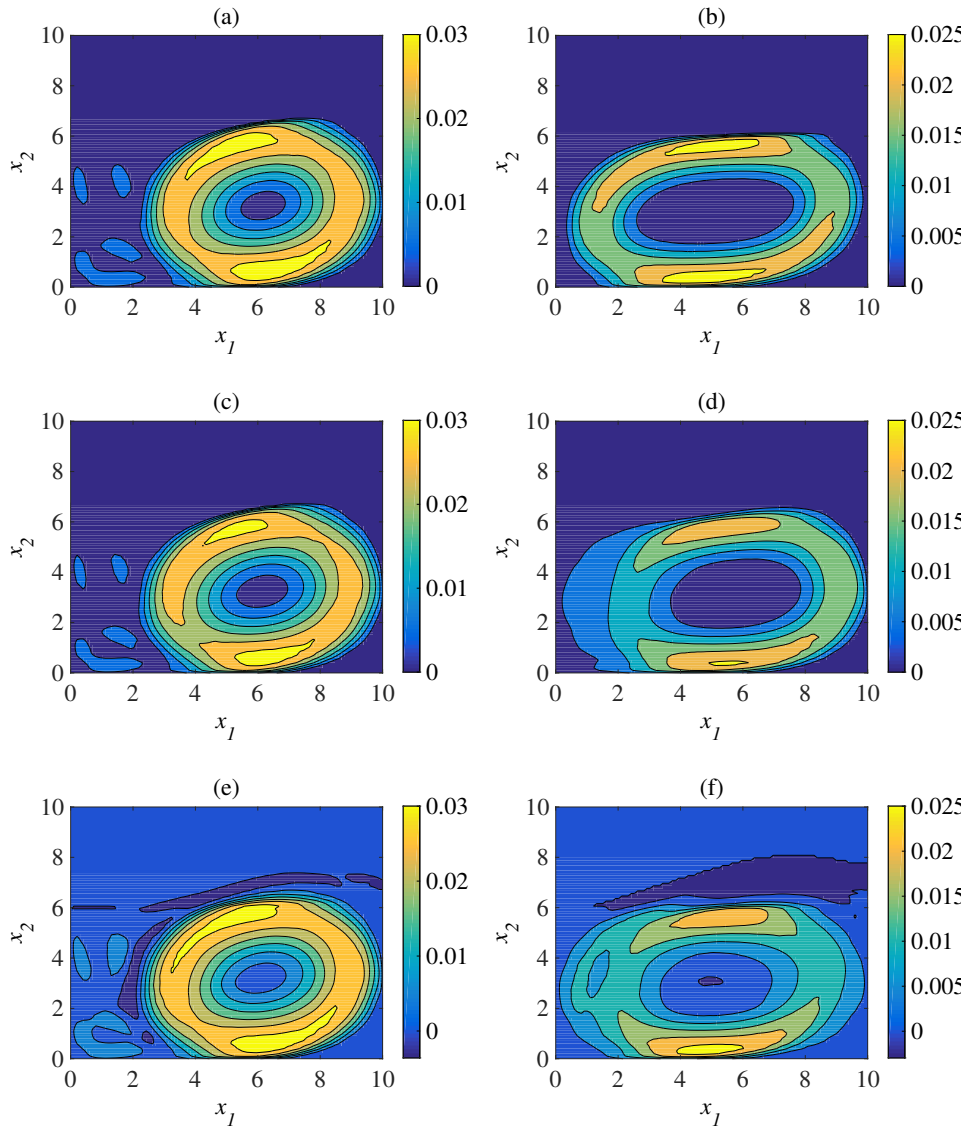

Figure 3.9: Actual response fields and predictions using 40 training points and r = 5 coefficients in the melting front example for 145^{th} (left column) and 301^{th} (right column) observations of the 400 reserved testing points. Fig.(a) and Fig.(b) are the actual response fields of the simulators. Fig.(c) and Fig.(d) are the predictive fields of the Iso-GPE while Fig.(e) and Fig.(f) are the predictions of HH

3.4.4 Computational details

The results of HH were highly dependent on the MLE algorithm and the initial guess for the hyperparameters. Iso-GPE also showed sensitivity to the MLE algorithm and the initial guess, but a much lesser degree; reasonable results were obtained with all algorithms tested. At present, no reasons for these observations is clear, which will require further investigations. One of many possible explanations would be the way reduced-order data are orientated changes the gradient surface completely and complicate (or maybe simplify) the optimisation problem. Again, this would require a further look into the problem of dimension reductions and optimisations. The computational times for training the algorithms using 80 training points and 9 components in the examples considered (on a 2.7 GHz i7core, 16 GB RAM laptop) were all below 100 s when using the interior point method for the MLE (the most computationally expensive). These times are determined primarily by the solution to the MLE problem, and thus vary from method to method. The main limitation with the MLE (or fully Bayesian approach) is the inversion of the covariance matrix $\mathbf{C}$ and the calculation of $|\mathbf{C}|$ in equation (2.9), both of which are $O(m^3)$ calculations, where m is the number of samples. As the input space dimensionality grows, m will generally increase. The examples considered here have relatively low numbers of inputs L, but much higher numbers can be used provided the number of training points does not grow too large. A main consideration will be the DOE [3, 4, 127, 128] as discussed earlier. Since the limitations on the input space dimensionality and sample number are common to all GP emulation methods (including univariate), a more detailed discussion is beyond the scope of this work, which is focused on reducing the dimensionality of the output space.

The method used to determine the neighbouring points for Isomap (step 1 in Algorithms 2 and 3 of section3.3 and 3.3.2) was found not to influence the results significantly. The results discussed above were generated using the neighbourhood number method ($N = 10$ in all cases). Increasing the neighbourhood number above

10 did not noticeably alter the results as is shown in previous experiments. Several parameters and choices in the Iso-GPE algorithm will need to be optimized, on a case by case basis. These parameters include the covariance function, the MLE algorithm and the neighbourhood number (or ball radius) for Isomap.

3.5 Concluding remarks

In this chapter, a new method for GPE emulation of high-dimensional data sets arising from parameter-dependent PDEs (dynamic and steady) is presented. The method extends that of Higdon *et al.* [1] (HH) by replacing PCA with a manifold learning method, able to overcome many of the limitations of PCA. The generalization to nonlinear dimensionality reduction is significant because many response surfaces will not admit to a low-dimensional approximation using a linear subspace of the ambient space. Implement of the methods is also shown, including an inverse map approximation and applied the method to several data sets. Compared to HH, our method performs better in the cases presented, for which HH can fail. The method is also very efficient in terms of computational cost, with dimensionality reduction of several orders of magnitude.

Although the motivation was parameterized nonlinear PDE models, the method developed can be applied to any problem involving vector-valued (steady-state or time-dependent) targets and vector-valued inputs. It can be potentially employed to dramatically reduce the time costs of numerical algorithms for uncertainty quantification, optimal design, control and inverse parameter estimation (any application that involves repeated solutions of high-fidelity computational models where a spatial domain is of interest).

There are several powerful approaches to manifold learning other than Isomap, including kernel PCA, diffusion maps, Laplacian eigenmaps, local linear embedding ([86]) and local tangent space alignment ([131]). Predicting which manifold

learning method works best on a given data set is not straightforward. There are cases where one may work very well, while another works poorly. Moreover, 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 context, there must also exist an approximation to the pre-image map. In many cases, this is not the case. In the next chapter, the idea of this chapter is extended by replacing Isomap with other manifold learning methods, namely kernel PCA and diffusion maps.

Chapter 4

Gaussian process emulators for high dimensional output spaces using kernel PCA and diffusion maps

In this chapter, the Isomap based GPE introduced in chapter 3 is extended by replacing Isomap with diffusion maps and kernel PCA, 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 [113]. A new approximation that is computationally efficient and stable is outlined. Its accuracy on a standard data set is demonstrated before it is used in the main algorithm developed in this chapter.

The problem under consideration is the same as that outlined in Section 3.1. In relation to the problem formulated in section 3.1, it is stared by implementing the two manifold learning methods on the field output data, defining the reduced-

dimensional spaces and the quantities that are used later in the GP emulations and inverse map approximations. The overall strategy for emulating outputs in high-dimensional spaces is explained in Section 4.3 and the inverse map approximations are described in Section 4.4. Numerical examples are presented and discussed in Section 4.5.

This chapter is based on the publication [132]: W.W. Xing, V. Triantafyllidis, A.A Shah, P.B Nair, and N. Zabaras, "Manifold learning for the emulation of spatial fields from computational models", *Journal of Computational Physics*, vol. 326, pp. 666–690, 2016.

4.1 Kernel principal component analysis

kPCA [133] maps high-dimensional data in a space $\mathcal{M}$ to a higher-dimensional feature space $\mathscr{F}$ *via* a mapping:

$$\boldsymbol{\phi} : \mathcal{M} \to \mathscr{F} \tag{4.1}$$

and performs linear PCA in $\mathscr{F}$. In our case, the data consist 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:

$$\mathbf{C}_{\mathcal{F}}\boldsymbol{w} = \left(\frac{1}{m} \sum_{i=1}^{m} \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)}) \left(\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)}) \right)^T \right) \boldsymbol{w} = \lambda \boldsymbol{w} \tag{4.2}$$

in which, $\boldsymbol{w}$ is the principal basis in the feature space, $\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)}) = \boldsymbol{\phi}(\boldsymbol{y}^{(i)}) - \overline{\boldsymbol{\phi}}$ is the i-th centred data point in feature space, with $\overline{\boldsymbol{\phi}} = (1/m) \sum_{j=1}^{m} \boldsymbol{\phi}(\boldsymbol{y}^{(j)})$. The mapping $\boldsymbol{\phi}(\cdot)$ is implicitly defined *via* a *kernel function*:

$$\Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \boldsymbol{\phi}(\boldsymbol{y}^{(i)})^T \boldsymbol{\phi}(\boldsymbol{y}^{(j)}), \tag{4.3}$$

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. $\widetilde{\mathbf{K}}$ can be obtained from:

$$\widetilde{\mathbf{K}} = \mathbf{HKH}, \tag{4.4}$$

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 $\mathbf{1} = (1/m)(1, \ldots, 1)^T \in \mathbb{R}^m$. As mentioned previously, the most widely used kernel functions is the Gaussian kernel:

$$\Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \exp\left(-||\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(j)}||^2/s^2\right), \tag{4.5}$$

where s is a scale factor. This kernel function is particularly useful when one has no prior knowledge on the data. This, however, does not imply such a kernel function would always work. Guidelines on how to find proper kernel functions could be found in [134] where a automatic process is proposed. Equation (4.2) 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. (4.2) and premultiplying by $\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)})^T$ (noting that $\widetilde{\mathbf{K}}$ is positive semidefinite), yields the eigenvalue problem:

$$\widetilde{\mathbf{K}}\boldsymbol{\alpha} = m\lambda\boldsymbol{\alpha}, \tag{4.6}$$

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:

$$\widetilde{\boldsymbol{w}}_i = \sum_{j=1}^m \widetilde{\alpha}_{ji} \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)}), \tag{4.7}$$

for $i = 1, \ldots, m$ 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

assuming 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:

$$\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)}) = \sum_{i=1}^{m} z_i(\boldsymbol{y}^{(j)})\widetilde{\boldsymbol{w}}_i, \tag{4.8}$$

where the i-th coefficient is calculated as follows:

$$\begin{aligned} z_i(\boldsymbol{y}^{(j)}) = \widetilde{\boldsymbol{w}}_i^T\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)}) \quad &= \sum_{l=1}^{m} \widetilde{\alpha}_{li}\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(l)})^T\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)}) \\ &= \sum_{l=1}^{m} \widetilde{\alpha}_{li}\widetilde{K}_{lj} \\ &= \widetilde{\boldsymbol{\alpha}}_i^T\widetilde{\boldsymbol{k}}_j \\ &= \widetilde{\boldsymbol{\alpha}}_i^T\mathbf{H}(\boldsymbol{k}_j - \mathbf{K}\mathbf{1}), \end{aligned} \tag{4.9}$$

for $i = 1, \ldots, m$, where:

$$\begin{aligned} \boldsymbol{k}_j &= (K_{1j}, \ldots, K_{mj})^T, \\ \widetilde{\boldsymbol{k}}_j &= (\widetilde{K}_{1j}, \ldots, \widetilde{K}_{mj})^T. \end{aligned} \tag{4.10}$$

One can therefore define:

$$\boldsymbol{z}(\boldsymbol{y}^{(j)}) = (z_1(\boldsymbol{y}^{(j)}), \ldots, z_m(\boldsymbol{y},^{(j)}))^T, \tag{4.11}$$

where the $z_i(\boldsymbol{y}^{(j)})$, $i = 1, \ldots, m$, are given by Eq. (4.9).

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 [79] 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 $\mathcal{F}_r = \text{span}(\widetilde{\boldsymbol{w}}_1, \ldots, \widetilde{\boldsymbol{w}}_r)$

is given by:

$$\sum_{j=1}^{m} ||\widetilde{\boldsymbol{\phi}}_r(\boldsymbol{y}^{(j)}) - \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)})||^2 = \sum_{i=r+1}^{m} \lambda_i^2, \tag{4.12}$$

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) [79]: Select r such that $\sum_{i=1}^{r} \lambda_i / \sum_{i=1}^{m} \lambda_i > \varrho$ for some threshold ϱ.

One 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 $\{\widetilde{\boldsymbol{w}}_i\}_{i=1}^{r}$:

$$\widetilde{\boldsymbol{\phi}}_r(\boldsymbol{y}^{(j)}) = \sum_{i=1}^{r} z_i(\boldsymbol{y}^{(j)}).\widetilde{\boldsymbol{w}}_i. \tag{4.13}$$

Here the notation is used:

$$\boldsymbol{z}_r(\boldsymbol{y}^{(j)}) = (z_1(\boldsymbol{y}^{(j)}), \ldots, z_r(\boldsymbol{y}^{(j)}))^T, \tag{4.14}$$

which, from Eq. (4.9), is given by:

$$\boldsymbol{z}_r(\boldsymbol{y}^{(j)}) = [\widetilde{\boldsymbol{\alpha}}_1 \ldots, \widetilde{\boldsymbol{\alpha}}_r]^T \mathbf{H}(\boldsymbol{k}_j - \mathbf{K}.\mathbf{1}) \tag{4.15}$$

Algorithm 5 summarizes kPCA for data $\{\boldsymbol{y}^{(i)}\}_{i=1}^{m}$.

It is assumed 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 (4.13) then yields a reduced-dimensional approximation:

$$\widetilde{\boldsymbol{\phi}}_r(\boldsymbol{y}) = \sum_{i=1}^{r} z_i(\boldsymbol{y}) \widetilde{\boldsymbol{w}}_i, \tag{4.16}$$

for an arbitrary $\boldsymbol{y} \in \mathcal{M}$. Equivalently, by the injectivity of $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$, and assuming

Algorithm 5 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:

$$z_i(\boldsymbol{y}^{(j)}) \leftarrow \widetilde{\boldsymbol{\alpha}}_i^T \mathbf{H}(\boldsymbol{k}_j - \mathbf{K1}) \quad i, j = 1, \ldots, m,$$

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

that the feature map is injective, Eq. (4.13) defines a map:

$$(\widetilde{\boldsymbol{\phi}}_r \circ \boldsymbol{\eta})(\cdot) = \widetilde{\boldsymbol{\phi}}_r(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \rightarrow \mathcal{F}_r, \tag{4.17}$$

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:

$$\mathbb{z}_i(\cdot) = z_i(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \rightarrow \mathbb{R} \quad \text{and} \quad \boldsymbol{z}_r(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \rightarrow \mathbb{R}^r. \tag{4.18}$$

Thus:

$$\begin{aligned} \widetilde{\boldsymbol{\phi}}_r(\boldsymbol{\eta}(\boldsymbol{\xi})) &= \sum_{i=1}^{r} \mathbb{z}_i(\boldsymbol{\xi}) \widetilde{\boldsymbol{w}}_i, \\ \boldsymbol{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi})) &= (\mathbb{z}_1(\boldsymbol{\xi}), \ldots, \mathbb{z}_r(\boldsymbol{\xi}))^T. \end{aligned} \tag{4.19}$$

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:

$$\boldsymbol{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})) = \boldsymbol{z}_r(\boldsymbol{y}^{(j)}) = [\widetilde{\boldsymbol{\alpha}}_1 \ldots, \widetilde{\boldsymbol{\alpha}}_r]^T \mathbf{H}(\boldsymbol{k}_j - \mathbf{K}\mathbf{1}), \quad j = 1, \ldots, m. \tag{4.20}$$

4.2 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 [107, 135]. 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 $\mathsf{k}(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \exp(-||\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(j)}||^2/s^2)$. It is assumed that $\mathcal{G}$ is connected (otherwise the maps can be constructed separately on each connected component).

A diffusion process [106] 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:

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}, \tag{4.21}$$

where $\mathbf{P} = [P_{ij}]$ 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 station-

ary distribution $\boldsymbol{\pi}$ with entries [107]:

$$\pi_i = \frac{d_i}{\sum_j d_j}. \tag{4.22}$$

Define a symmetric matrix:

$$\mathbf{P'} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{1/2}, \tag{4.23}$$

which possesses the same eigenvalues γ'_i as $\mathbf{P}$. A spectral decomposition yields:

$$\mathbf{P'} = \mathbf{S}\boldsymbol{\Gamma'}\mathbf{S}^T, \tag{4.24}$$

where the columns of $\mathbf{S}$ are the orthonormal eigenvectors $\boldsymbol{s}_i$, $i = 1, \ldots, m$, of $\mathbf{P'}$ and $\boldsymbol{\Gamma'} = \text{diag}(\gamma'_1, \ldots, \gamma'_m)$. The eigenvalues are arranged such that $1 = \gamma'_1 > \cdots > \gamma'_m$ and the eigenvector $\boldsymbol{s}_1$ has entries $\sqrt{\pi_i}$ [136]. $\mathbf{P}$ has the spectral decomposition:

$$\mathbf{P} = \mathbf{Q}\boldsymbol{\Gamma'}\mathbf{Q}^{-1}, \tag{4.25}$$

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:

$$\boldsymbol{l}_1 = \boldsymbol{\pi}\sqrt{\sum_j d_j}, \qquad \boldsymbol{r}_1 = \mathbf{1}^T/\sqrt{\sum_j d_j}. \tag{4.26}$$

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 $\mathbf{S}$, one has:

$$\mathbf{P}^t = \mathbf{Q}\boldsymbol{\Gamma}^t\mathbf{Q}^{-1} \quad \text{or} \quad \mathbf{P}^t = \sum_{i=1}^{m}(\gamma'_i)^t\boldsymbol{r}_i\boldsymbol{l}_i^T, \tag{4.27}$$

where the j-th row vector of $\mathbf{P}^t$, denoted $\boldsymbol{p}_j^t$, is:

$$\boldsymbol{p}_j^t = (p_t(\boldsymbol{y}^{(j)}, \boldsymbol{y}^{(1)}), \ldots, p_t(\boldsymbol{y}^{(j)}, \boldsymbol{y}^{(m)}))^T = \sum_{i=1}^{m}(\gamma'_i)^t r_{ji}\boldsymbol{l}_i, \tag{4.28}$$

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 [107]:

$$D_t(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \left((\boldsymbol{p}_i^t - \boldsymbol{p}_j^t)^T \mathbf{D}^{-1} (\boldsymbol{p}_i^t - \boldsymbol{p}_j^t)\right)^{1/2}. \tag{4.29}$$

One 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 [107, 135]:

$$\boldsymbol{\psi}^t(\boldsymbol{y}^{(j)}) = \left((\gamma_1')^t r_{j1}, \ldots, (\gamma_m')^t r_{jm}\right)^T. \tag{4.30}$$

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 [107, 135, 137]:

$$||\boldsymbol{\psi}^t(\boldsymbol{y}^{(i)}) - \boldsymbol{\psi}^t(\boldsymbol{y}^{(j)})|| = D_t(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}), \tag{4.31}$$

where $||\cdot||$ denotes the standard Euclidean norm. Equation (4.31) follows from the bi-orthogonality of the left and right eigenvectors. From Eq. (4.30) and the decay in the eigenvalues, one can define mappings $\boldsymbol{\psi}_r^t(\boldsymbol{y}^{(j)}) : \mathcal{M} \to \mathcal{D}_r^{(t)} \subset \mathbb{R}^r$ as follows:

$$\boldsymbol{\psi}_r^t(\boldsymbol{y}^{(j)}) = ((\gamma_1')^t r_{j1}, \ldots, (\gamma_r')^t r_{jr})^T, \tag{4.32}$$

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:

$$|(\gamma_j')^t| > \upsilon |(\gamma_2')^t|, \tag{4.33}$$

holds for a pre-selected precision υ [107]. The diffusion distance, and therefore the diffusion map, depends on t. As t increases, the diffusion distances between points decrease since each row of $\mathbf{P}^t$ approaches the stationary distribution (see Eq. (4.28)). Algorithm 6 summarizes diffusion maps for data $\{\boldsymbol{y}^{(i)}\}_{i=1}^m$.

Algorithm 6 Diffusion maps

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

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

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

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

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

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

In order to develop an inverse map approximation, diffusion maps is generalised to all points in $\mathcal{M}$ by taking the limit $m \rightarrow \infty$. 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 $\mathcal{M}$ [107, 137, 138]. Full details of the following are provided in Appendix 8.1. Here the key quantities needed to generalize diffusion maps for the analysis that follows in Section 4.4.2 on the inverse mapping is clearly defined. Let μ be a probability measure on $\mathcal{M}$ defining the density of points. In the limit $m \rightarrow \infty$, a one-step transition kernel for the Markov chain on $\mathcal{M}$ can be defined by:

$$\mathbb{p}(\boldsymbol{y}', \boldsymbol{y}) = \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}')/\mathbb{d}(\boldsymbol{y}'), \tag{4.34}$$

from an arbitrary $\boldsymbol{y}' \in \mathcal{M}$ to an arbitrary $\boldsymbol{y} \in \mathcal{M}$, where $\mathbb{d}(\boldsymbol{y}')$ is defined as:

$$\mathbb{d}(\boldsymbol{y}') = \int_{\mathcal{M}} \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}') d\mu(\boldsymbol{y}). \tag{4.35}$$

A corresponding forward transfer operator is defined by:

$$\mathcal{L}\varphi(\boldsymbol{y}) = \int_{\mathcal{M}} \mathbb{p}(\boldsymbol{y}', \boldsymbol{y}) \varphi(\boldsymbol{y}') d\mu(\boldsymbol{y}'), \tag{4.36}$$

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 $\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}')$. One can similarly define a backward transfer operator:

$$\mathcal{R}\varphi(\boldsymbol{y}) = \int_{\mathcal{M}} \mathbb{p}(\boldsymbol{y}, \boldsymbol{y}') \varphi(\boldsymbol{y}') d\mu(\boldsymbol{y}'), \tag{4.37}$$

which is the analogue of multiplication of $\mathbf{P}$ from the right. The kernel $\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}')$ admits the decomposition:

$$\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}') = \sum_{i=1}^{\infty} \gamma_i^t r_i(\boldsymbol{y}), l_i(\boldsymbol{y}'), \tag{4.38}$$

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 developed in Section 4.4.2 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. (4.28); 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 8.1, 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:

$$D_t = ||\mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}') - \mathbb{p}_t(\boldsymbol{y}, \boldsymbol{y}')||_{1/\mathbb{d}}, \tag{4.39}$$

where:

$$||\varphi||^2_{1/\mathbb{d}} = \int_{\boldsymbol{y}' \in \mathcal{M}} |\varphi(\boldsymbol{y}')|^2 / \mathbb{d}(\boldsymbol{y}') d\mu(\boldsymbol{y}'), \tag{4.40}$$

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:

$$\boldsymbol{\psi}^t(\boldsymbol{y}) = (\gamma_1^t r_1(\boldsymbol{y}), \gamma_2^t r_2(\boldsymbol{y}), \ldots). \tag{4.41}$$

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.:

$$\boldsymbol{\psi}_r^t(\boldsymbol{y}) = (\gamma_1^t r_1(\boldsymbol{y}), \ldots, \gamma_r^t r_r(\boldsymbol{y}))^T. \tag{4.42}$$

Given an isotropic kernel $\mathsf{k}(\boldsymbol{y}, \boldsymbol{y}')$, diffusion maps can be generalized by defining a family of anisotropic kernels:

$$\mathsf{k}^{(\alpha)}(\boldsymbol{y}, \boldsymbol{y}') = \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}') / (\mathbb{d}(\boldsymbol{y}')^\alpha \mathbb{d}(\boldsymbol{y})^\alpha), \tag{4.43}$$

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) [107, 139, 140]. The standard algorithm described above corresponds to the limiting case of $\alpha = 0$ (isotropic kernel), and the anisotropic kernels is not considered in this thesis due to limited space and the lack of inverse mappings for such special cases. In Section 4.4.2, a new inverse map is developed for the isotropic case only.

Consider the mappings:

$$(\boldsymbol{\psi}_r^t \circ \boldsymbol{\eta})(\cdot) = \boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \to \mathcal{D}_r^{(t)} \subset \mathbb{R}^r, \tag{4.44}$$

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

$$\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi})) = (\gamma_1^t \mathbb{r}_1(\boldsymbol{\xi}), \ldots, \gamma_r^t \mathbb{r}_r(\boldsymbol{\xi}))^T \in \mathcal{D}_r^{(t)}. \tag{4.45}$$

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:

$$\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})) = \boldsymbol{\psi}_r^t(\boldsymbol{y}^{(j)}) = ((\gamma_1')^t r_{j1}, \ldots, (\gamma_r')^t r_{jr})^T, \tag{4.46}$$

for $j = 1, \ldots, m$, obtained from the empirical eigenfunctions and empirical eigenvalues.

4.3 Multi-output emulation using manifold learning

Replace 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. (4.19) or the map $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\cdot))$ defined by Eq. (4.45). Multivariate GP priors are placed over these maps, with training points for emulation given by Algorithms 5 and 6 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 details of a scalar GPE refers to section 2.2.1.

The kPCA coefficients, $\mathbb{z}_i(\boldsymbol{\xi})$, $i = 1, \ldots, r$ are mutually uncorrelated; following Higdon *et al.* [1] (see also the wavelet decomposition approach in [33]). The approximations, therefore, are assumed arising 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, the underlying GPs are treated as independent. 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 4.1,4.2 and 2.2.1.

1. **kPCA:** For a fixed $i = 1, \ldots, r$, one sets $\eta(\boldsymbol{\xi}) = \mathbb{z}_i(\boldsymbol{\xi})$. The training points are given by Eq. (4.9): $\eta(\boldsymbol{\xi}^{(j)}) = \mathbb{z}_i(\boldsymbol{\xi}^{(j)}) = \widetilde{\boldsymbol{\alpha}}_i^T \mathbf{H}(\boldsymbol{k}_j - \mathbf{K1})$, $j = 1, \ldots, m$. 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. (2.8), yields a prediction that is denoted $\mathbb{z}_i(\boldsymbol{\xi})$ (to avoid introducing new notation, $\mathbb{z}_i(\boldsymbol{\xi})$ and $\mathbb{E}[\mathbb{z}_i(\boldsymbol{\xi})]$) are not distinguished. $\boldsymbol{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi})) = (\mathbb{z}_1(\boldsymbol{\xi}), \ldots, \mathbb{z}_r(\boldsymbol{\xi}))^T$ is setted as a notation. Again, this is the expected value $\mathbb{E}[\boldsymbol{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi}))]$.

2. **Diffusion maps:** For a fixed $i = 1, \ldots, r$, one sets $\eta(\boldsymbol{\xi}) = \mathbb{r}_i(\boldsymbol{\xi})$. The training points are given by Eq. (4.32): $\eta(\boldsymbol{\xi}^{(j)}) = \mathbb{r}_i(\boldsymbol{\xi}^{(j)}) = r_{ji}$, $j = 1, \ldots, m$. 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. (2.8) yields $\mathbb{E}[\mathbb{r}_i(\boldsymbol{\xi})]$, denoted simply as $\mathbb{r}_i(\boldsymbol{\xi})$. One then obtains (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 6 are used to compute the predicted value of $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi}))$.

To take account of the correlations between the coefficients when using diffusion maps, the linear model of coregionalization (LMC) [28, 32] could be used to emulate the coefficients simultaneously. Alternatively, the GP model could be replaced by an artificial neural network (ANN). Implementation details would be drafted in chapter 5. For moderately sized r, neither approach is computationally expensive. In this chapter, the approach of univariate GPs with ANN using Bayesian regularization [70, 72] are compared. To complete the emulation, one 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, details of the inverse map approximations are provided 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 4.4.3.

The GPE framework furnishes predictive variances, given by Eq. (2.8). 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, one 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. (2.8)) 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.

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

The final step, as in chapter 3, is to find approximations of the inverse mappings:

$$\boldsymbol{\phi}_r^{-1}(\cdot) : \mathcal{F}_r \to \mathcal{M} \quad \text{and} \quad (\boldsymbol{\psi}_r^t)^{-1}(\cdot) : \mathcal{D}_r^{(t)} \to \mathcal{M}, \tag{4.47}$$

for kPCA and diffusion maps, respectively. Note, as in chapter 3, 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, one seek the mapping:

$$\boldsymbol{z}_r^{-1}(\cdot) : \mathbb{R}^r \to \mathcal{M} \quad \text{or} \quad \boldsymbol{z}_r \mapsto \boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi}). \tag{4.48}$$

This can be achieved *via* a closed-form least-squares solution [108, 141]. 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.* [109] and other minimization routines.

For diffusion maps there has been little progress towards finding an inverse map approximation. Etyngier *et al.* [113] proposed an optimization procedure designed for 2-d shapes embedded in $\mathbb{R}^3$ (a closely related method can be found in [114]). 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. The particular choice of closest neighbours is problem and data density dependent and there are no "best" way of choosing such a number. A pre-seeking process based on a given dataset using optimisation, e.g., stochastic gradient descent or greedy algorithm, could be used to assist making such a decision. The result,however, could be strongly biased or unstable, especially when the dataset is considered small to reveal the data structure. The choice of $r+1$ neighbours is decided based on our reconstruction experiments including different toy dataset, e.g., Swiss roll and hyper-sphere, and real experimental dataset showed in 4.5 as it shows stable and fairly accurate result with modest computational expense. For large val-

ues 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, a special case of which was used in chapter 3, 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:

$$\boldsymbol{y} = \sum_{j \in \mathcal{J}} \vartheta(\boldsymbol{y}^{(j)}) \boldsymbol{y}^{(j)}, \tag{4.49}$$

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 is the local linear interpolation [112, 125] in chapter 3, i.e., to set:

$$\vartheta(\boldsymbol{y}^{(j)}) = \frac{d_{j,*}^{-1}}{\sum_{j=1}^m d_{j,*}^{-1}} \tag{4.50}$$

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 [142]:

$$\vartheta(\boldsymbol{y}^{(j)}) = \frac{\chi(\boldsymbol{y}, \boldsymbol{y}^{(j)})}{\sum_{i=1}^m \chi(\boldsymbol{y}, \boldsymbol{y}^{(i)})} = \frac{\chi(d_{j,*})}{\sum_{i=1}^m \chi(d_{i,*})}, \tag{4.51}$$

The particular form of kernel density used in this chapter 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 $\mathcal{M}$.

4.4.1 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:

$$\widetilde{\boldsymbol{w}}_i = \sum_{j=1}^{m} \widetilde{\alpha}_{ji} \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)}) = \widetilde{\Phi}\widetilde{\boldsymbol{\alpha}}_i = \Phi\mathbf{H}\widetilde{\boldsymbol{\alpha}}_i, \tag{4.52}$$

where the $\widetilde{\boldsymbol{\alpha}}_i$ are known from Algorithm 5. 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:

$$\begin{aligned} \boldsymbol{\phi}_r(\boldsymbol{y}) = \sum_{i=1}^{r} z_i \widetilde{\boldsymbol{w}}_i + \overline{\boldsymbol{\phi}} &= \sum_{i=1}^{r} z_i \Phi\mathbf{H}\widetilde{\boldsymbol{\alpha}}_i + \Phi\mathbf{1} \\ &= \Phi\left(\mathbf{H}[\widetilde{\boldsymbol{\alpha}}_1 \ldots, \widetilde{\boldsymbol{\alpha}}_r]\boldsymbol{z}_r + \mathbf{1}\right) = \Phi\boldsymbol{\tau}. \end{aligned} \tag{4.53}$$

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

$$\widetilde{d}_{i,*}^2 = \boldsymbol{\phi}(\boldsymbol{y})^T\boldsymbol{\phi}(\boldsymbol{y}) + \boldsymbol{\phi}(\boldsymbol{y}^{(i)})^T\boldsymbol{\phi}(\boldsymbol{y}^{(i)}) - 2\boldsymbol{\phi}(\boldsymbol{y})^T\boldsymbol{\phi}(\boldsymbol{y}^{(i)}). \tag{4.54}$$

Taking $\boldsymbol{\phi}(\boldsymbol{y}) \approx \boldsymbol{\phi}_r(\boldsymbol{y})$ and substituting Eq. (4.53) into Eq. (4.54) yields:

$$\widetilde{d}_{i,*}^2 \approx \boldsymbol{\tau}^T\mathbf{K}\boldsymbol{\tau} + \Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(i)}) - 2\boldsymbol{\tau}^T\boldsymbol{k}_i, \tag{4.55}$$

with $\boldsymbol{\tau}$ defined as in Eq. (4.53). 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. (4.54) gives:

$$\widetilde{d}_{i,*}^2 = 2 - 2\Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y}) \tag{4.56}$$

which, equating to the right hand side of Eq. (4.55), yields $\Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y})$. For the Gaussian kernel, therefore, one obtains $d_{i,*}^2 = -s^2 \ln \Bbbk(\boldsymbol{y}^{(i)}, \boldsymbol{y})$. Similar relationships exist for other commonly used kernel functions [111], 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. (4.49) and (4.51), the values of $d_{i,*}$ yield an approximation of $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$.

4.4.2 Diffusion maps

It is assumed $t = 1$ (without loss of generality) to simplify the notation. At the practical level, one must work within the finite-dimensional setting in which there now is $m+1$ 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:

$$\underline{\mathbf{K}} = \left[\begin{array}{c|c} \mathbf{K} & (\mathsf{k}(\boldsymbol{y}^{(1)}, \boldsymbol{y}), \ldots, \mathsf{k}(\boldsymbol{y}^{(m)}, \boldsymbol{y}))^T \\ \hline (\mathsf{k}(\boldsymbol{y}^{(1)}, \boldsymbol{y}), \ldots, \mathsf{k}(\boldsymbol{y}^{(m)}, \boldsymbol{y})) & \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}) \end{array}\right]. \tag{4.57}$$

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

$$\underline{\mathbf{D}} = \left[\begin{array}{c|c} \widehat{\mathbf{D}} & 0 \\ \hline 0 & \mathsf{k}(\boldsymbol{y}, \boldsymbol{y}) + \sum_j \mathsf{k}(\boldsymbol{y}^{(j)}, \boldsymbol{y}) \end{array}\right], \tag{4.58}$$

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:

$$\underline{\mathbf{P}} = \left[\begin{array}{c|c} \widehat{\mathbf{D}}^{-1}\mathbf{K} & \widehat{\mathbf{D}}^{-1}(\mathsf{k}(\boldsymbol{y}^{(1)}, \boldsymbol{y}), \ldots, \mathsf{k}(\boldsymbol{y}^{(m)}, \boldsymbol{y}))^T \\ \hline \dfrac{(\mathsf{k}(\boldsymbol{y}^{(1)}, \boldsymbol{y}), \ldots, \mathsf{k}(\boldsymbol{y}^{(m)}, \boldsymbol{y}))}{\mathsf{k}(\boldsymbol{y}, \boldsymbol{y}) + \sum_j \mathsf{k}(\boldsymbol{y}^{(j)}, \boldsymbol{y})} & \dfrac{\mathsf{k}(\boldsymbol{y}, \boldsymbol{y})}{\mathsf{k}(\boldsymbol{y}, \boldsymbol{y}) + \sum_j \mathsf{k}(\boldsymbol{y}^{(j)}, \boldsymbol{y})} \end{array}\right]. \tag{4.59}$$

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 4.2 and Appendix 8.1, 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:

$$\begin{aligned} \underline{\boldsymbol{p}}_{m+1} &\approx \sum_{j=1}^{\infty} \gamma_j r_j(\boldsymbol{y}) (l_j(\boldsymbol{y}^{(1)}), \ldots, l_j(\boldsymbol{y}^{(m)}), l_j(\boldsymbol{y}))^T \\ &\approx \sum_{j=1}^{r} \gamma_j r_j(\boldsymbol{y}) (l_j(\boldsymbol{y}^{(1)}), \ldots, l_j(\boldsymbol{y}^{(m)}), l_j(\boldsymbol{y}))^T, \end{aligned} \tag{4.60}$$

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:

$$\boldsymbol{\psi}_r(\boldsymbol{y}) = (\gamma_1' \mathbb{r}_1(\boldsymbol{\xi}), \ldots, \gamma_r' \mathbb{r}_r(\boldsymbol{\xi}))^T = (\gamma_1' r_1(\boldsymbol{y}), \ldots, \gamma_r' r_r(\boldsymbol{y}))^T. \tag{4.61}$$

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:

$$\underline{p}_{m+1,i} \approx \sum_{j=1}^{r} \gamma_j' \mathbb{r}_j(\boldsymbol{\xi}) l_{ij}, \quad i = 1, \ldots, m. \tag{4.62}$$

Equating this expression with that of the equivalent entry in Eq. (4.59), one obtains the following:

$$\sum_{j=1}^{r} \gamma_j' \mathbb{r}_j(\boldsymbol{\xi}) l_{ij} = \frac{\mathsf{k}(\boldsymbol{y}^{(i)}, \boldsymbol{y})}{\mathsf{k}(\boldsymbol{y}, \boldsymbol{y}) + \sum_{j=1}^{m} \mathsf{k}(\boldsymbol{y}^{(j)}, \boldsymbol{y})}, \quad i = 1, \ldots, m. \tag{4.63}$$

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. (4.49) and (4.51), these values of $d_{i,*}$ yield an approximation

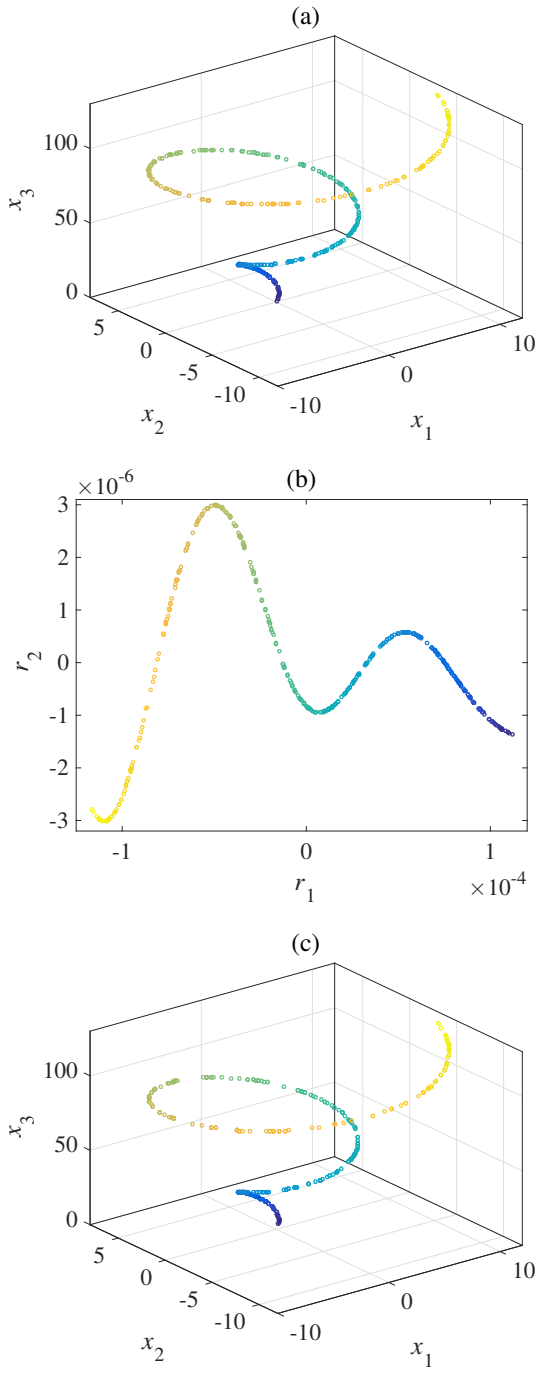

Figure 4.1: Illustration of the pre-image method for data lying on a conical spiral. 500 points were randomly sampled from the spiral, shown in Fig. (a). A 2-d approximation using diffusion maps is shown in Fig. (b). The reconstruction is illustrated in Fig. (c). Each point in Fig. (a) has a unique color, which is retained in Figs. (b) and (c).

of $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$.

The results of this inverse map approximation on a conical spiral are illustrated in Fig. 4.1. A conical spiral is a 1-d manifold embedded in 3-d, and is defined by the following equations:

$$x_1 = 4\pi \mathtt{t} \cos(4\pi \mathtt{t}), \quad x_2 = 4\pi \mathtt{t} \sin(4\pi \mathtt{t}), \quad x_3 = 40\pi \mathtt{t}, \tag{4.64}$$

for a single variable $\mathtt{t} \in \mathbb{R}$. A total of 500 points were sampled from the spiral by sampling 500 values of $\mathtt{t}$ from a standard uniform distribution $\mathcal{U}(0, 1)$. Fig. 4.1(a) shows the sampled points, Fig. 4.1(b) shows the 2-d ($r = 2$) approximation of the points using diffusion maps, and Fig. 4.1(c) shows the reconstruction of the original points using the inverse mapping described above. Here, $t = 1$ and a Gaussian kernel with s^2 given by the average square distance between observations in the original space [110] are adopted , as detailed in Section 4.5.1. The SSE (square sum error) is 2087, the average SSE, which is value of SSE divided by 500 in this case, is 4.17. The relative SSE is the average SSE divided by the square sum of true value (also see Eq. 3.41) and the result is 0.049. These quantities show an very positive reconstruction result.

4.4.3 Main algorithm

The proposed procedure for GPE of outputs in high-dimensional spaces is summarized in the pseudocode Algorithm 7, based on a Gaussian kernel for both kPCA and diffusion maps.

Algorithm 7 GPE for high-dimensional spaces using manifold learning.

1: Manifold Learning for reduced-dimensional space approximation

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

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

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

4.5 Results and discussion

In this section, three examples are considered and demonstrated. 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 [126] is a quasi-random sequence that is specifically designed to generate samples as uni-

formly as possible over the unit hypercube [127]. 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 (in an increasing manner) from the remaining 200 ($m \leq 200$). $\boldsymbol{y}_p^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)})$ is used to denote the predicted value of $\boldsymbol{y}^{(i)}$ at a test input $\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, m_t$ using Algorithm 7.

4.5.1 Computational details

Details of the scalar GPE, the manifold learning techniques and the software employed in the implementation of Algorithm 7 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 [110]: $s^2 = (1/m^2) \sum_{i,j=1}^{m} ||\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(j)}||^2$. Other popular kernels, e.g., 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 lead to no significant change in this case. One reason may be that a GP with Gaussian kernel is essentially an exponential metric on the dataset while t serves exactly as the power term in a exponential transformations through diffusion maps. Thus the GP always successfully find the mapping (if it is reasonably possible to the learn the mapping). For the pre-image stage th t would be cancelled out thus makes no difference. A performance drop was observed in practice when using very large or small t. This is likely to related

to computer memory overflow issues.

3. **Gaussian Process Emulation**. The square exponential covariance function Eq. (2.4) 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 as it works well in this case.

4.5.2 Free convection in porous media

This is the same experiment demonstrated in section 3.4.1.

Training and Testing. In this example, the input parameters were $\boldsymbol{\xi} = (\beta[\mathrm{K}^{-1}], T_h[^{\mathrm{o}}\mathrm{C}])^T \in [10^{-11}, 10^{-8}] \times [40, 60]$. 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 3.1, $u(\boldsymbol{x}; \boldsymbol{\xi}) = |\boldsymbol{v}|$, $J = 1$, $l = 2$ and $d = 10^4$.

Results. Fig. 4.2 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. 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.

Examples of the predictions are shown in Fig. 4.3 for 120 training points and $r = 5$. For both kPCA and diffusion maps, the error with respect to the first test example (Figs. 4.3(a)-(c)) lies around the median of the $r = 5$ boxplot in Fig. 4.2. The errors with respect to the second test example are close to the upper

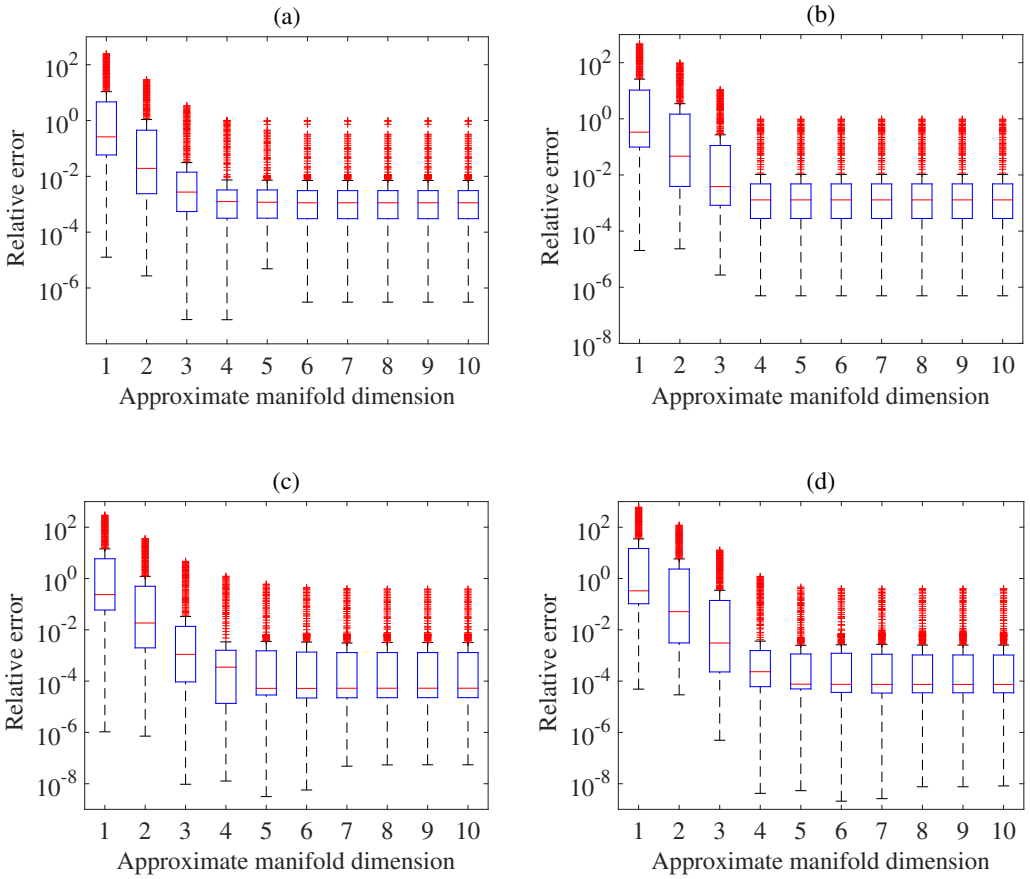

Figure 4.2: Tukey box plots of the relative error $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2/||\boldsymbol{y}^{(i)}||^2$ in the free-convection example using Algorithm 7 with increasing approximate manifold dimension r on the 300 test points for: (a) kPCA with 40 training points; (b) diffusion maps with 40 training points; (c) kPCA with 120 training points; (d) diffusion maps with 120 training points.

whiskers in the same boxplots. In both cases, Algorithm 7 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. 4.2(c) and 4.2(d) is shown in Fig. 4.4. 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 (visually no difference. see Fig.4.4 for a full one-to-one compare).

Boxplots of the errors using an ANN with Bayesian regularization [70, 72] and support vector machine regression (SVMR) for emulation of the coefficients, rather than GPE as comparison, are shown in Fig. 5.1 for $m = 120$ in chapter 5.

4.5.3 Lid driven cavity

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

$$\begin{aligned} (\boldsymbol{v} \cdot \nabla)\boldsymbol{v} - Re^{-1}\nabla^2\boldsymbol{v} + \nabla p = 0, \\ \nabla \cdot \boldsymbol{v} = 0, \end{aligned} \tag{4.65}$$

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 [143].

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

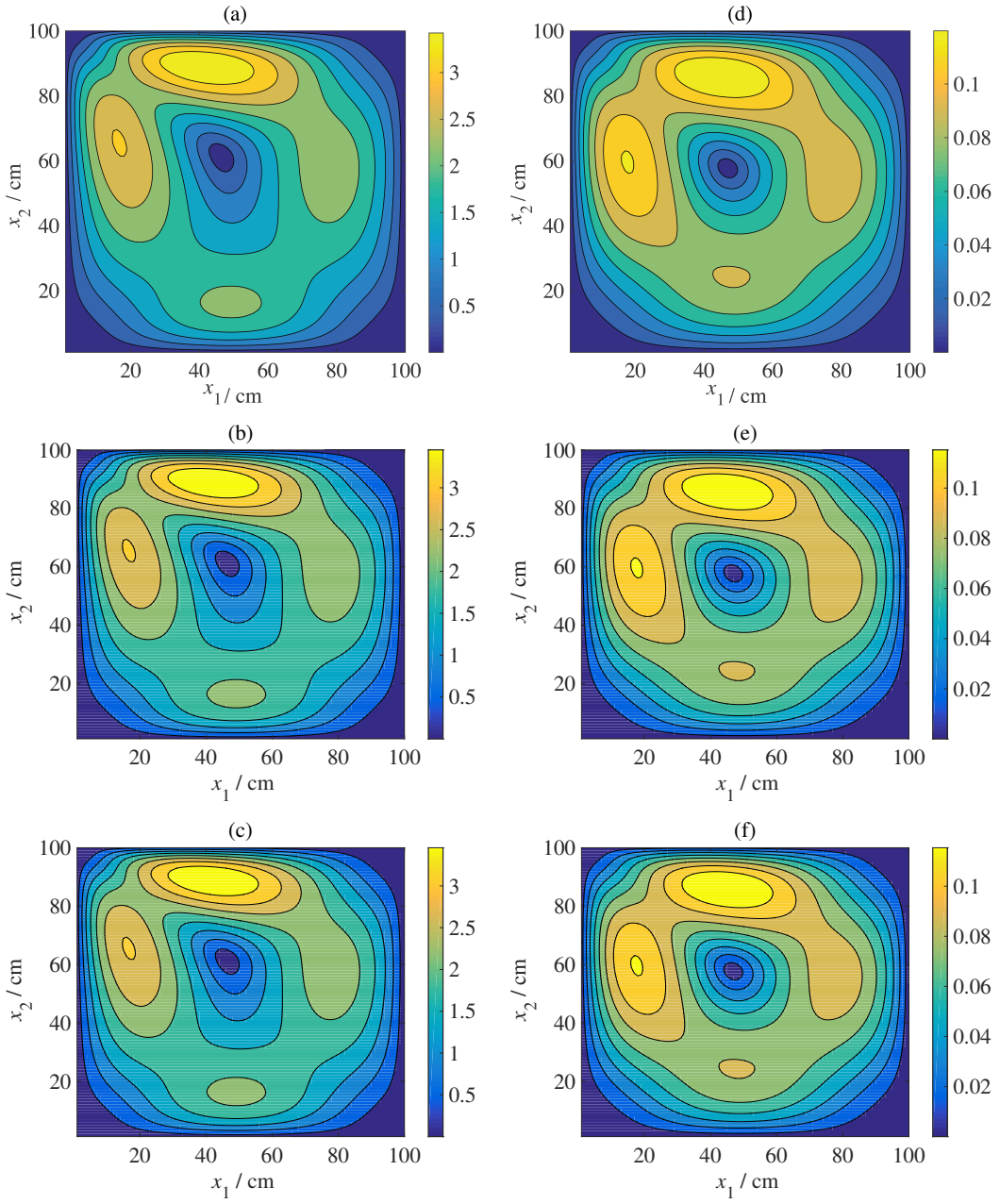

Figure 4.3: Predictions of the velocity field using 120 training points and $r = 5$ coefficients in the free-convection example. Fig. (a) is the test point corresponding to $\boldsymbol{\xi} = (3.18 \times 10^{-9}[\mathrm{K}^{-1}], 56.7[^{\mathrm{o}}\mathrm{C}])^T$, while Figs. (b) and (c) are the corresponding predictions using kPCA (relative error=6.31×10^{-4}) and diffusion maps (relative error=7.76×10^{-4}), respectively. Fig. (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 (relative error=2.01×10^{-2}) and diffusion maps (relative error=1.25×10^{-2}), respectively.

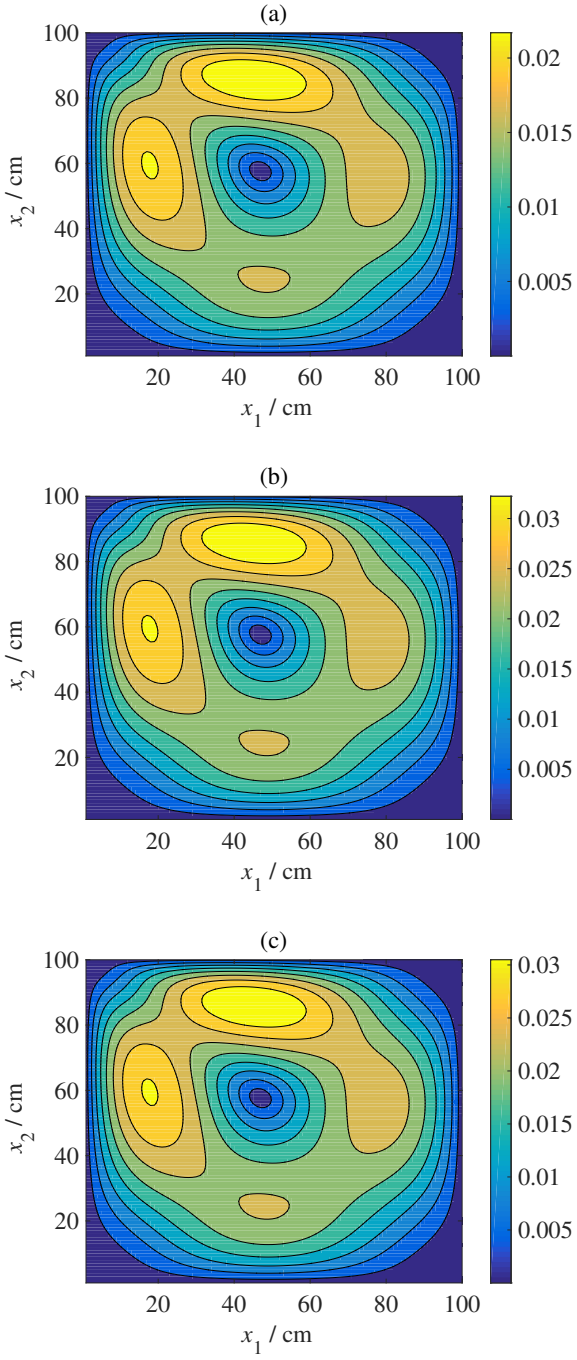

Figure 4.4: Predictions of the velocity field using 120 training points and $r = 5$ coefficients in the free-convection example in the case of an outlier. Fig. (a) is the test point corresponding to $\boldsymbol{\xi} = (1 \times 10^{-9}[\mathrm{K}^{-1}], 40.7[^{\circ}\mathrm{C}])^T$, while Figs. (b) and (c) are the predictions using kPCA (relative error=0.057) and diffusion maps (relative error=0.062), respectively.

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 3.1 , $J = 3$, $l = 2$ and $d = 3 \times 10^4$. This is a multiple field example discussed in Section 3.1, 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. 4.5 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 were almost negligible, except that again there were fewer outliers for diffusion maps, particularly for low numbers of training points. Fig. 4.6 shows the equivalent boxplots using Higdon's method [1]. For this example, Higdon's method also performed well, with superior performance 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. 4.7 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. 4.7(a)-(c)) lies close to the median in the $r = 5$ boxplot in Fig. 4.5. The second test example corresponds to an outlier for both methods (relative error around 0.07). The results of Algorithm 7 remain accurate, especially for diffusion maps. The error in kPCA

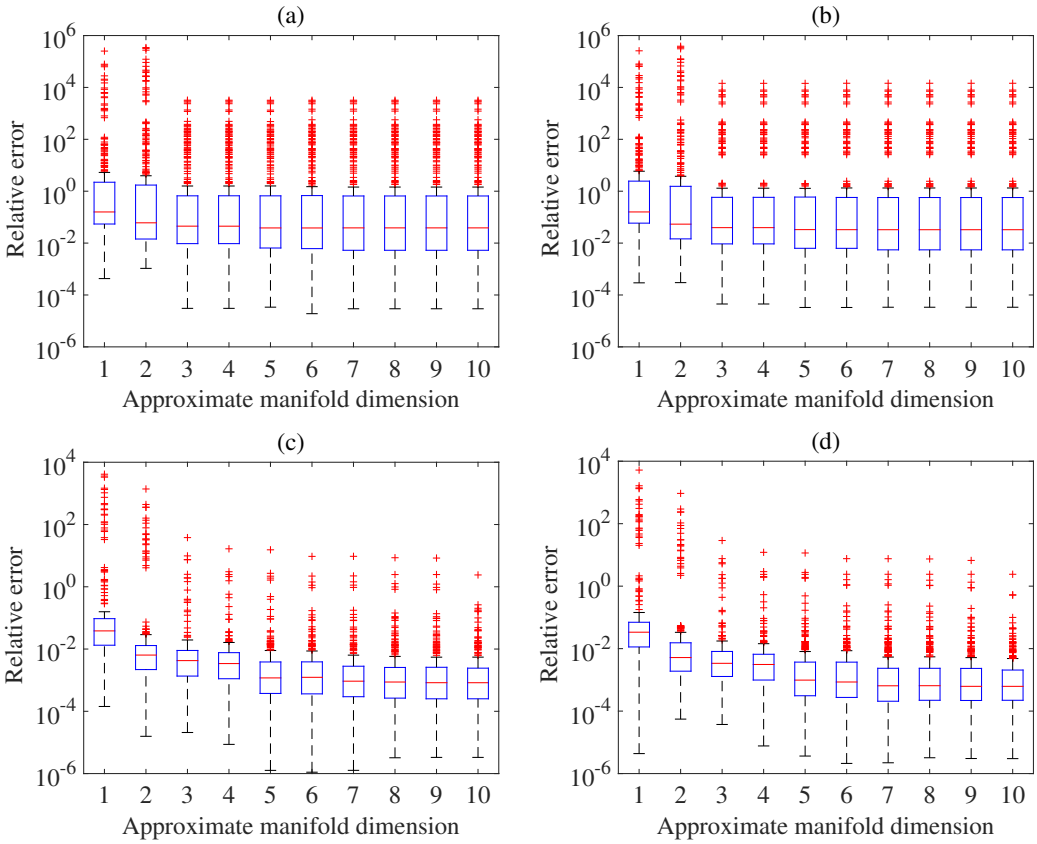

Figure 4.5: Tukey box plots of the relative error $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2/||\boldsymbol{y}^{(i)}||^2$ in the lid-driven cavity example using Algorithm 7 with an increasing approximate manifold dimension r on the 300 test points for: (a) kPCA with 80 training points; (b) diffusion maps with 80 training points; (c) kPCA with 120 training points; (d) diffusion maps with 120 training points.

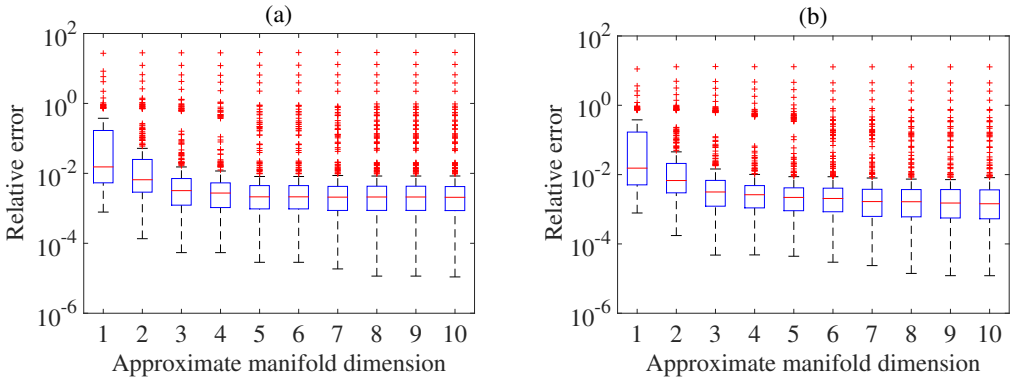

Figure 4.6: Tukey box plots of the relative error $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2/||\boldsymbol{y}^{(i)}||^2$ in the lid-driven cavity example using Higdon's method [1] with an increasing approximate manifold dimension r on the 300 test points for: (a) 80 training points; (b) 120 training points.

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, a modification of this example is also considered. in this case, the number of inputs is increased to 13 ($l = 13$) using the following boundary conditions:

$$\begin{aligned} v_1(x_1, 1) &= 5c_1 \sin(c_2 \pi x_1) e^{-c_3 x_1}, \quad v_2(x_1, 1) = 0, \\ v_1(x_1, 0) &= 5c_4 \sin(c_5 \pi x_1) e^{-c_6 x_1}, \quad v_2(x_1, 0) = 0, \\ v_2(1, x_2) &= 5c_7 \sin(c_8 \pi x_2) e^{-c_9 x_2}, \quad v_1(1, x_2) = 0, \\ v_2(0, x_2) &= 5c_{10} \sin(c_{11} \pi x_2) e^{-c_{12} x_2}, \quad v_1(0, x_2) = 0, \end{aligned} \tag{4.66}$$

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, $m_t = 300$ (rather than 100 that is used before since this is more complex problem where more training points are needed) were reserved for testing and the training points were selected from the remaining 700.

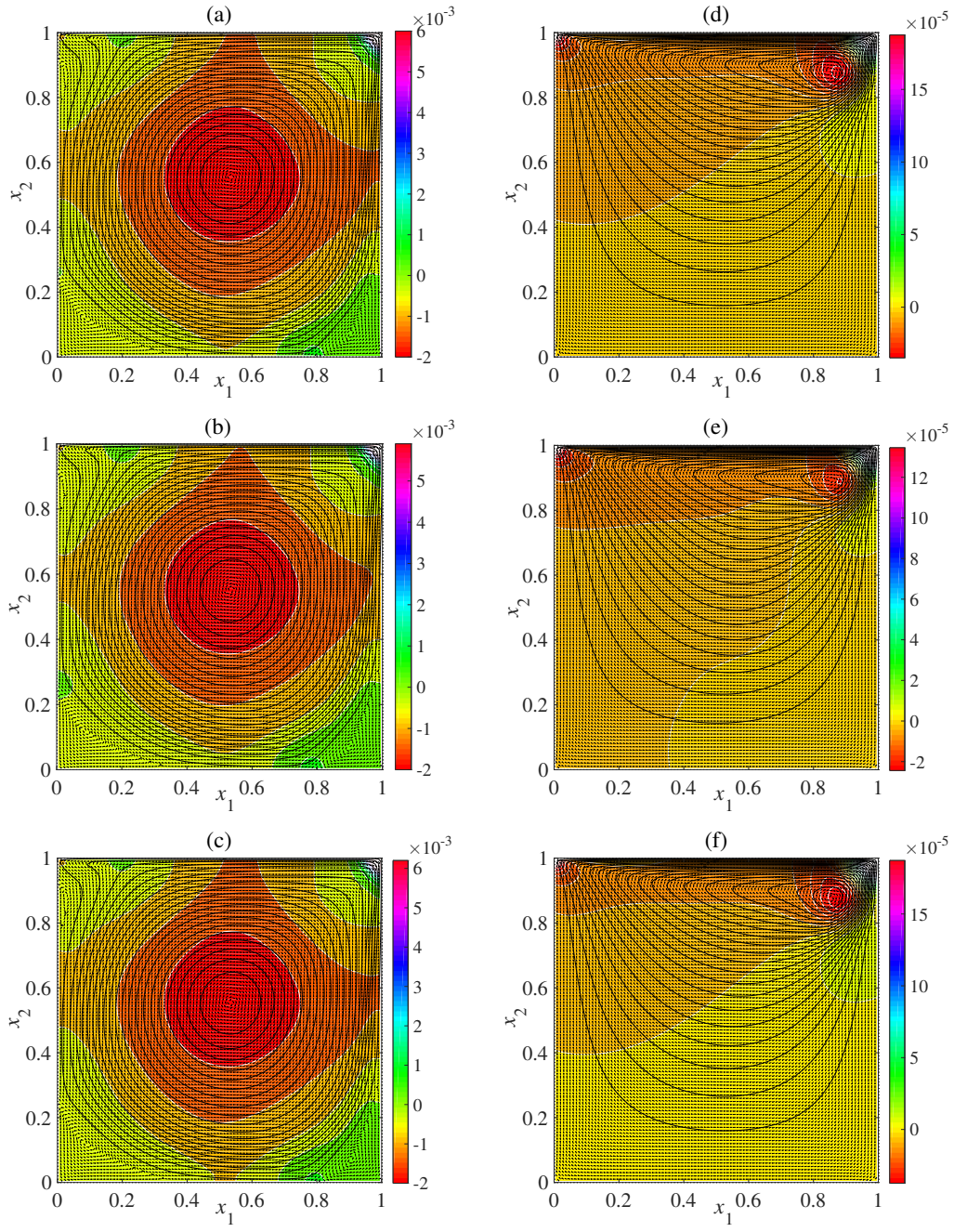

Figure 4.7: Predictions of the velocity field using 120 training points and $r = 5$ coefficients in the lid driven cavity example. Fig. (a) is the test point corresponding to $\boldsymbol{\xi} = (874.8, 7.79)^T$, while Figs. (b) and (c) are the corresponding predictions using kPCA (relative error 1.58×10^{-2}) and diffusion maps(relative error 1.33×10^{-2}) , respectively. Fig. (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 (relative error 0.076) and diffusion maps(relative error 0.062), respectively.

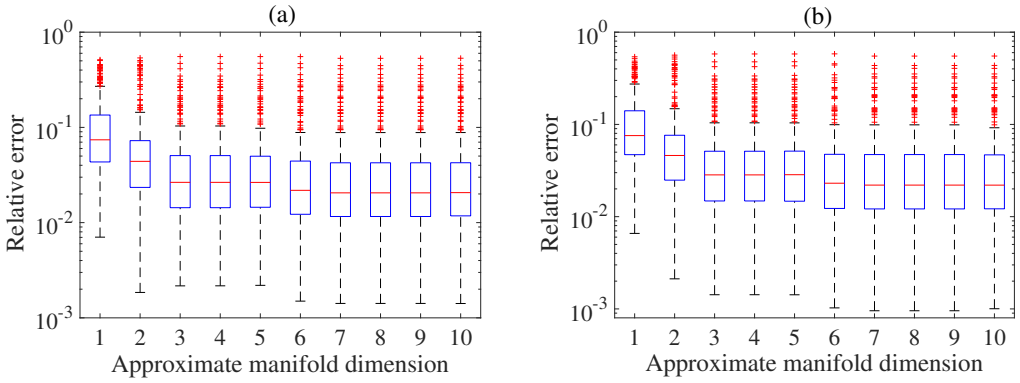

Figure 4.8: Tukey box plots of the relative error $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2/||\boldsymbol{y}^{(i)}||^2$ in the lid-driven cavity example with boundary conditions as in Eq. (4.66). The trends are shown for an increasing approximate manifold dimension r using 600 training points and 300 test points for: (a) kPCA and (b) diffusion maps.

Both kPCA and diffusion maps exhibited excellent performance, as illustrated in the boxplots in Fig. 4.8, 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. 4.9 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.

4.5.4 Hydrogen fuel cell model

In this example, 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 is under consideration. 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. 4.10).

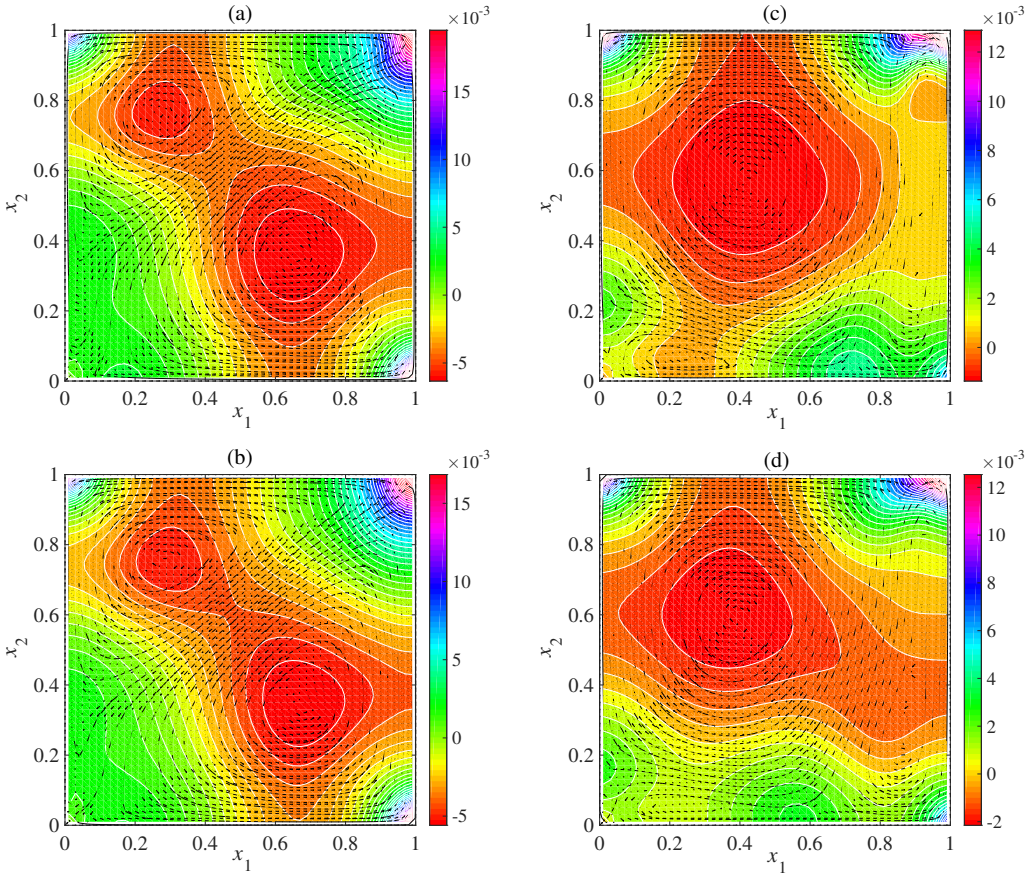

Figure 4.9: Predictions of the velocity and pressure fields using $m = 500$ training points and $r = 10$ coefficients in the lid driven cavity example with the boundary conditions of Eq. (4.66). Fig. (a) is a test point and Fig. (b) is the corresponding prediction using kPCA, with a relative error of 0.0244. Fig. (c) is a second test point and Fig. (d) is the corresponding prediction using kPCA, with a relative error of 0.2275.

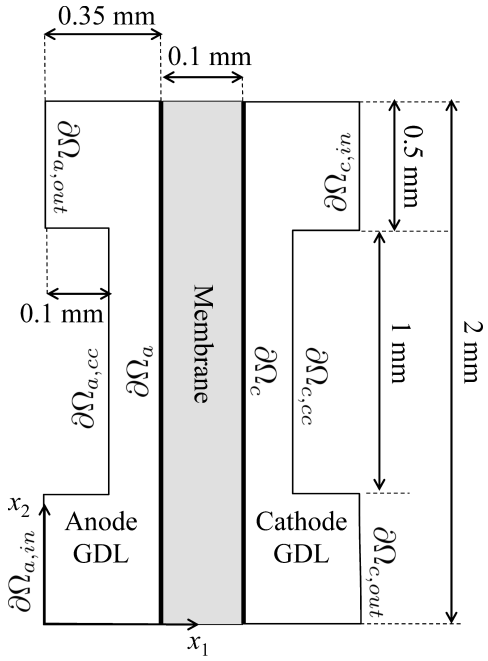

Figure 4.10: A schematic of the PEM fuel cell and the components that form the model domain.

The oxidation reaction in the anode is $2H_4 \rightarrow 2H^+ + 4e^-$ and the reduction reaction in the cathode is $2O_2 + 4H^+ + 4e^- \rightarrow 2H_2O$, both of which are assumed to be governed by a modified Butler-Volmer law for charge transfer [144]. The catalyst layer morphology is approximated as clusters (agglomerates) of carbon-supported platinum coated with the electrolyte. The transfer current densities are expressed as follows [145]:

$$
\begin{aligned}
j_c &= -\frac{12 L_{act} F D_{agg}}{R_{agg}^2} C_{\mathrm{O}_2,agg}(1-\epsilon_{mac})(1-\lambda_c \coth \lambda_c), \\
j_a &= -\frac{6 L_{act} F D_{agg}}{R_{agg}^2} C_{\mathrm{H}_2,agg}\left(1-e^{-\frac{2F}{RT}\eta_a}\right)(1-\epsilon_{mac})(1-\lambda_a \coth \lambda_a), \\
\lambda_c &= \sqrt{\frac{i_{0c} S R_{agg}^2}{4FC_{\mathrm{O}_2,ref} D_{agg}} e^{\frac{F}{2RT}\eta_c}} \quad \lambda_a = \sqrt{\frac{i_{0a} S R_{agg}^2}{2FC_{\mathrm{H}_2,ref} D_{agg}}},
\end{aligned}
\tag{4.67}
$$

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 coef-

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

$$C_{\mathrm{H}_2,agg} = \frac{pX_{\mathrm{H}_2}}{K_{\mathrm{H}_2}} \quad C_{\mathrm{O}_2,agg} = \frac{pX_{\mathrm{O}_2}}{K_{\mathrm{O}_2}}, \tag{4.68}$$

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:

$$\begin{aligned} -\nabla \cdot (\sigma_e \nabla \phi_e) &= 0, \\ -\nabla \cdot (\sigma_s \nabla \phi_s) &= 0, \end{aligned} \tag{4.69}$$

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. 4.10. The overpotentials (defined only on these boundaries) take the form:

$$\begin{aligned} \eta_a &= \phi_s - \phi_e - E_{eq,a}, \\ \eta_c &= \phi_s - \phi_e - E_{eq,c}, \end{aligned} \tag{4.70}$$

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:

$$\begin{aligned} \nabla \cdot (\rho \boldsymbol{v}) &= 0, \\ \boldsymbol{v} &= -k_p \omega^{-1} \nabla p, \end{aligned} \tag{4.71}$$

where ω is the gas viscosity and k_p is the GDL permeability. The ideal gas law is used to determine the density:

$$\rho = \frac{p}{RT} \sum_i M_i X_i, \tag{4.72}$$

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) [146]. 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:

$$\begin{aligned} -\nabla \cdot \left\{ \rho Y_i \textstyle\sum_{\substack{j \in \mathcal{I}_1 \\ j \neq i}} D_{i,j} \left(\nabla X_j + (X_j - Y_j) \nabla p / p \right) \right\} &= -\rho \boldsymbol{v} \cdot \nabla Y_i, \\ Y_{\mathrm{N_2}} &= 1 - Y_{\mathrm{O_2}} - Y_{\mathrm{H_2O}}, \end{aligned} \tag{4.73}$$

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 [146]. Identical equations for species $\mathcal{I}_2$ are solved in the anode.

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

$$\begin{aligned} \phi_s &= 0 \quad \boldsymbol{x} \in \partial\Omega_{a,cc}, \\ \phi_s &= V_{cell} \quad \boldsymbol{x} \in \partial\Omega_{c,cc}, \\ -\boldsymbol{n} \cdot \nabla \phi_s &= 0 \quad \text{otherwise}, \end{aligned} \tag{4.74}$$

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 [144]:

$$\begin{aligned} -\boldsymbol{n}\cdot\boldsymbol{v} &= \frac{j_a}{\rho F}\left(\frac{M_{\mathrm{H_2}}}{2}+\lambda_{\mathrm{H_2O}}M_{\mathrm{H_2O}}\right) \quad \boldsymbol{x}\in\partial\Omega_a, \\ -\boldsymbol{n}\cdot\boldsymbol{v} &= \frac{j_c}{\rho F}\left(\frac{M_{\mathrm{O_2}}}{2}+\left[\frac{1}{2}+\lambda_{\mathrm{H_2O}}\right]M_{\mathrm{H_2O}}\right) \quad \boldsymbol{x}\in\partial\Omega_c, \end{aligned} \tag{4.75}$$

where $\lambda_{\mathrm{H_2O}}$ is the water drag number [144]. 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:

$$\begin{aligned} -\boldsymbol{n}\cdot\boldsymbol{N}_{\mathrm{H_2}} &= \frac{M_{\mathrm{H_2}}j_a}{2F} \quad \boldsymbol{x}\in\partial\Omega_a, \\ -\boldsymbol{n}\cdot\boldsymbol{N}_{\mathrm{O_2}} &= \frac{M_{\mathrm{O_2}}j_c}{4F} \quad \boldsymbol{x}\in\partial\Omega_c, \\ -\boldsymbol{n}\cdot\boldsymbol{N}_{\mathrm{H_2O}} &= \frac{M_{\mathrm{H_2O}}j_c\,(1/2+\lambda_{\mathrm{H_2O}})}{F} \quad \boldsymbol{x}\in\partial\Omega_c, \end{aligned} \tag{4.76}$$

where:

$$\boldsymbol{N}_i = -\rho Y_i\sum_{j\neq i}D_{i,j}\left(\nabla X_j+(X_j-Y_j)\frac{\nabla p}{p}\right)+\rho\boldsymbol{v}Y_i \tag{4.77}$$

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 [147].

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 vec-

tors $\boldsymbol{y}^{(i)} \in \mathbb{R}^d$. In the notation of Section 3.1, $u(\boldsymbol{x};\boldsymbol{\xi}) = X_{\mathrm{H_2O}}$, $J = 1$, $l = 2$ and $d = 4.5 \times 10^4$.

Results. Fig. 4.11 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

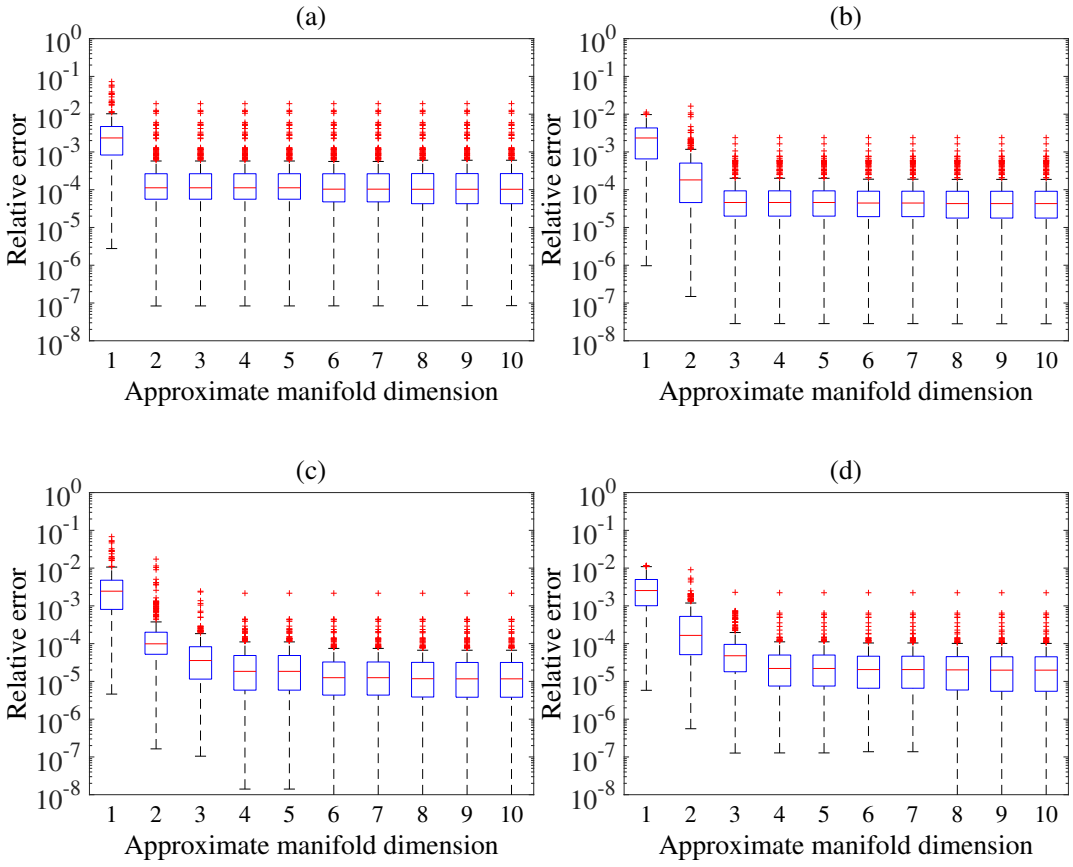

Figure 4.11: Tukey box plots of the relative error $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2/||\boldsymbol{y}^{(i)}||^2$ in the PEM fuel cell example using Algorithm 7 with increasing approximate manifold dimension r on the 300 test points for: (a) kPCA with 40 training points; (b) diffusion maps with 40 training points; (c) kPCA with 120 training points; (d) diffusion maps with 120 training points.

in Fig. 4.12 for 120 training points and $r = 7$. In the first example (Figs. 4.12(a)-(c)), the error with respect to the test case lies close to the median in the $r = 7$ boxplot for kPCA (Fig. 4.11(c)), while for diffusion maps the error is near the upper whisker in the corresponding boxplot ($m = 120$, $r = 7$ in Fig. 4.11(d)). The second example (Figs. 4.12(d)-(f)) is an outlier for both kPCA and diffusion maps (second and third highest errors, respectively). Even in the latter case, the predictions are accurate.

4.6 Concluding remarks

The approach of chapter 3 is extended to other manifold learning methods, namely diffusion maps and kPCA. In order to do so, the interpretation of the basis is firstly demonstrated. This is followed by a development of an inverse map for both methods, and generalizing the local linear interpolation. For diffusion map, a new method to derive distances, and therefore to perform the pre-image map, is developed. The method is both computationally efficient and stable. In particular, it is applicable to high-dimensional spatial problems. The accuracy of the methods was assessed on several data sets. Both methods perform well, particularly kPCA. A more extensive test on more complex data sets is needed to ascertain when one method is more appropriate than another. It is also mentioned that other manifold learning methods could be tested, e.g., LTSA, which has been implemented, and shows promise. The results were not shown here since further studies are required to fully assess the accuracy.

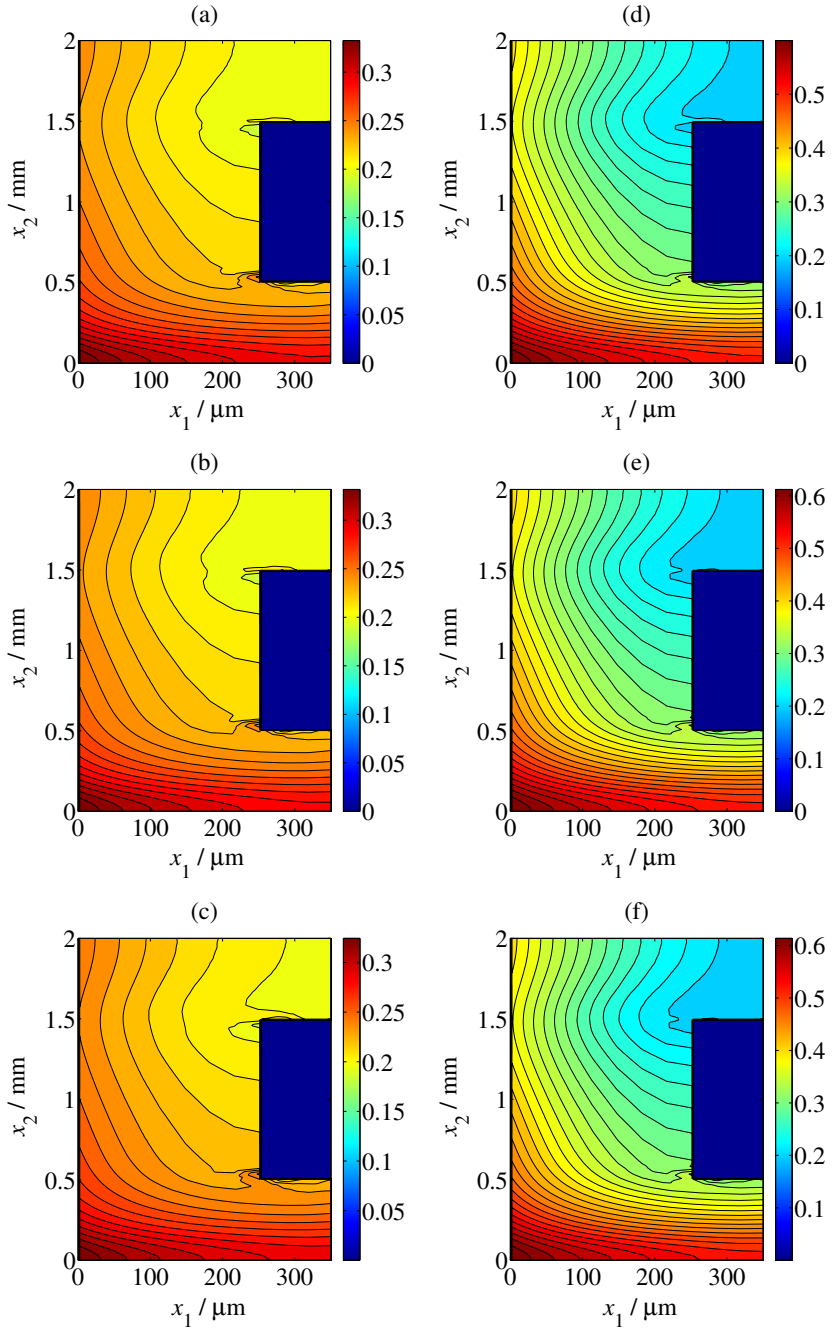

Figure 4.12: Predictions of the water mole fraction using 120 training points and $r = 7$ coefficients in the PEM fuel cell example. Fig. (a) is the test point corresponding to $\boldsymbol{\xi} = (0.525[\text{V}], 1.492[\text{S m}^{-1}])^T$, while Figs. (b) and (c) are the corresponding predictions using kPCA (relative error 1.16×10^{-5}) and diffusion maps (relative error 1.73×10^{-5}), respectively. Fig. (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 (relative error 7.23×10^{-4}) and diffusion maps (relative error 8.93×10^{-4}), respectively.

Chapter 5

Manifold learning based Bayesian neural network emulators

In this chapter, linear and nonlinear dimensionality reductions are combined with artificial neural networks to develop an efficient approach to emulate high-dimensional spatio-temporal models, without making any assumptions regarding correlations between coefficients. The approach is tested on models of electromagnetic wave propagation. The necessity of nonlinear dimensionality reduction is highlighted.

The advantages of ANNs are that they are extremely versatile and learn rapidly. They can also be used to learn multiple coefficients in a reduced basis simultaneously, in contrast to GPE. This has particular advantages in terms of learning multiple spatio-temporal outputs from a model, accounting naturally and efficiently for correlations between the different outputs. Unlike GPE, ANN is a parametric approach and it can suffer from overfitting, as discussed in section 5.3. It is therefore natural to use a Bayesian version of ANNs in which a prior is placed over the weights, leading to a modified form of back propagation with a penalty term

(Bayesian regularization); see section 2.2.2. This improves generalisation and leads to stable results (the optimisation problem for the weights is far less sensitive to the initialisation compared to standard back propagation with k-fold cross-validation).

This chapter is based on the publication [148]: V. Triantafyllidis, W. Xing, A.A Shah, and P.B. Nair, "Neural network emulation of spatio-temporal data using linear and nonlinear dimensionality reduction", in *Advanced Computer and Communication Engineering Technology, Lecture Notes in Electrical Engineering*, pp. 1015–1029, Springer, 2016.

5.1 Main algorithm

The emulation algorithm employing manifold learning on the output space is similar to those in the previous two chapters. It is described in the pseudo code for multiple spatio-temporal datasets. Details of the manifold manifold learning methods, namely, kPCA, diffusion maps and Isomap were given in chapters 3 and 4.

Algorithm 8 Manifold learning based ANN for spatio-temporal models

1: Select design points $\boldsymbol{\xi}^{(i)} \in \mathcal{X} \subset \mathbb{R}^l$, $i = 1, \ldots, m$, using DOE and construct outputs $\boldsymbol{y}^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)}) \in \mathcal{M} \subset \mathbb{R}^d$, $i = 1, \ldots, m$, from the computer model.

2: Perform manifold learning (PCA, MDS, Isomap, kPCA or diffusion maps) on $\boldsymbol{y}^{(i)}$, $i = 1, \ldots, m$, to obtain coordinates in a low-dimensional representation: $\boldsymbol{z}_r^{(i)} = (z_1^{(i)}, \ldots, z_r^{(i)})^T$, $i = 1, \ldots, m$, with $r \ll d$ (for multiple fields $\boldsymbol{y}_b^{(i)}$, $b = 1, \ldots, B$, this would lead to B sets of coefficients $\boldsymbol{z}_{r,b}^{(i)} = (z_{1,b}^{(i)}, \ldots, z_{r,b}^{(i)})^T$.

3: Select a test point $\boldsymbol{\xi}$ for prediction and perform ANN on the training set $(\boldsymbol{z}_r^{(i)}, \boldsymbol{\xi}^{(i)})$, $i = 1, \ldots, m$, to obtain $\boldsymbol{z}_r = (z_1, \ldots, z_r)^T$. For multiple fields the training set is $((\boldsymbol{z}_{r,1}^{(i)}, \ldots, \boldsymbol{z}_{r,B}^{(i)}), \boldsymbol{\xi}^{(i)})$, $i = 1, \ldots, m$, which yields $\boldsymbol{z}_r = (z_{1,1}, \ldots, z_{r,1}, \ldots, z_{1,B}, \ldots, z_{r,B})^T \in \mathbb{R}^{rB}$ for a test point $\boldsymbol{\xi}$.

4: Using $\boldsymbol{z}_r$ approximate the computer model output $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$ by solving the pre-image problem (see section 4.4 and 3.3.3) for details.

5.2 Results and discussion

5.2.1 Details of training and testing

In three examples below, the data set consisted of 500 points ($\boldsymbol{y}^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)})$), with inputs $\boldsymbol{\xi}^{(i)}$ selected using a Sobol sequence. 300 points were used for training in example 5.2.2 to have a like-to-like comparison to that in section 4.5.2. For the last two examples, 400 points were reserved for testing and up to 100 points were used for training ($m \leq 100$). This number is selected since more than 100 training point will not effect the result significantly as would be shown. The relative square errors (total square error divided by the number of grid points and the magnitudes of the average values of the test points) were used to assess the generalization error. Results are shown for different numbers of components r in the manifold learning methods. In the case of PCA (kPCA), the first r 'components' are the r principal components corresponding to the r largest eigenvalues of the (feature space) covariance matrix. For Isomap, the first r components are the r Isomap coordinates corresponding to the r largest eigenvalues of the kernel matrix.

The neighbourhood number method (10 neighbours) was used for Isomap as it shows a stable and accurate result. The number is decided by testing on the dataset with pre-image reconstruction to see the error level and Empirically 10% of total data points,which in the case equals 10, generate stable result. Though our experiment the number would make no significant change as far as it is not too large or small. For kPCA, a Gaussian kernel was used, with a shape parameter dependent on the data set. For reconstruction, $N_n = 10$ points were used for both Isomap and kPCA. Again such a number is decided by pre-testing over the dataset to ensure the results are stable. The ANN architecture in all cases contained a single hidden layer with 10 hidden units of sigmoid function. The ANN was trained using Bayesian regularization [70]. The number of neurons for each example was selected using sequential network construction [72]. In general, an arbitrary ANN architecture can

be used within the framework.

5.2.2 Example: Free convection in porous media

This model considered here is exactly the same as that presented in section 4.5.2. For more details, please refer to section 4.5.2. It use the same training and testing points. 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. 5.1 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 [70, 72] 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 [20]. A single hidden layer was employed and the number of neurons was selected using a sequential network construction [72]. For the SVMR, Gaussian and polynomial kernels (with varying order) are tested, together with an L^1 loss function.

Comparing with Figs. 4.2(a) and (b), it is easy to 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. 5.1 indicates that, at least in this example, GPE and ANN are superior.

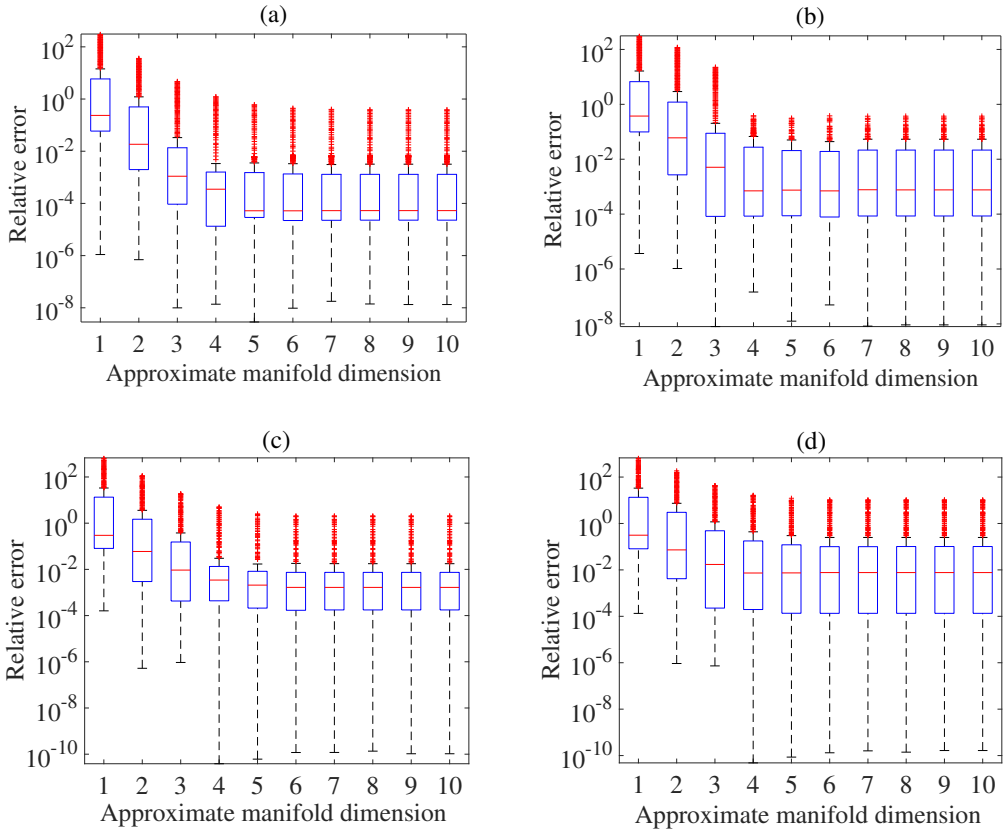

Figure 5.1: Tukey box plots of the relative error $||\boldsymbol{y}_p^{(i)} - \boldsymbol{y}^{(i)}||^2/||\boldsymbol{y}^{(i)}||^2$ in the free-convection example using Algorithm 7 with an ANN and SVMR for an increasing approximate manifold dimension r on the 300 test points. In both cases, 120 training points were used. (a) kPCA with ANN; (b) diffusion maps with ANN; (c) kPCA with SVMR; (d) diffusion maps with SVMR.

5.2.3 Example: 2D h-bend waveguide

This model examines a transversal electric (TE) wave in a h-bend waveguide with a 90 degree bend. The frequencies f are restricted so that TE_{10} is the single propagating mode. The electric field has only one nonzero component E_z in the transversal direction z. The model computes the electromagnetic field by solving Hemholtz equation:

$$-\nabla^2 E_z - n^2 k_0^2 E_z = 0, \tag{5.1}$$

in which n is the refractive index, k_0 is the free space wave number, and x_1 and x_2 are the in-plane directions.

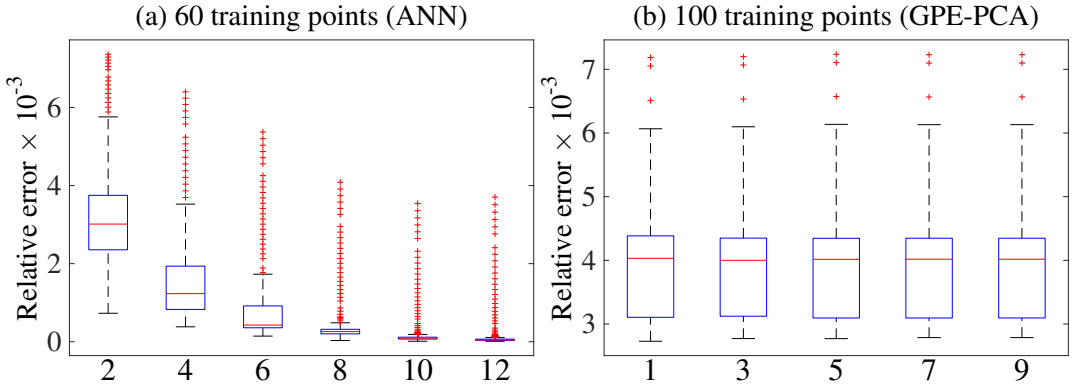

Figure 5.2: Boxplots of the relative errors for different numbers of components (r) using ANN with PCA ($M = 60$) and GPE with PCA ($M = 100$) in the waveguide example.

On the domain walls, the tangential component of the electric field is zero. The input wave is determined by the boundary conditions for Maxwell's equations:

$$\widehat{\boldsymbol{n}} \times \boldsymbol{E} = 0, \tag{5.2}$$

where $\widehat{\boldsymbol{n}}$ is the unit normal. The incident field has the form:

$$\boldsymbol{E} = \left(0, 0, \sin\left(\frac{\pi(b/2 - \xi}{b}\right)\right) = \Re(\boldsymbol{E} e^{i\omega t}), \tag{5.3}$$

in which b is the width of the rectangular sections of the waveguide and ω is the angular frequency of the incident wave. The model was solved ('H-Bend Waveguide 2D' in the 'RF Module' of COMSOL Multiphysics 5.0) for 500 frequency values f between 4 and 6 GHz ($\boldsymbol{\xi}^{(i)} = f^{(i)}$, $i = 1, \ldots, 500$). For each simulation, the magnitude of the electric field $\boldsymbol{E}$ was recorded on a 100×100 regular grid in (x_1, x_2). The $d = 10^4$ values of $|\boldsymbol{E}(x_1, x_2)|$ for each $\boldsymbol{\xi}^{(i)}$ at locations (x_{1i}, x_{2j}), $i, j = 1, \ldots, 100$, were vectorized (see section 3.1) to give 500 data points $\boldsymbol{y}^{(i)}$ in $\mathbb{R}^d$. Up to 100 were used for training and the remainder for testing.

Results

All three dimension reduction methods (PCA, Isomap, kPCA) using ANN gave excellent results for at least 20 training points ($m = 20$). In the case of PCA, box plots of the relative errors for different numbers of principal components (on the horizontal axis) are shown in Fig. 5.2 (a) for 60 training points. The other methods gave similar results (with less than 5% measured in different rate). Also remember that PCA is always recommended if it works since the pre-image solution is more general, only the PCA method is demonstrated in this case. Higdon's method [1] using a maximum likelihood estimate (MLE) for the hyperparameters failed to provide meaningful results, as demonstrated in Fig. 5.2 (b) for $m = 100$. An example of the worst predictions for $m = 60$ using ANN with PCA ($r = 12$) is shown in Fig. 5.3. The relative error is 2.3×10^{-3}.

5.2.4 Example: 2D radar interaction with a boat (radar cross section)

The interaction between a boat and the incident field from a radar transmitter is simulated. The transmitter is distant enough that the field can be treated as a plane wave (only the boat and its immediate surroundings are considered). The

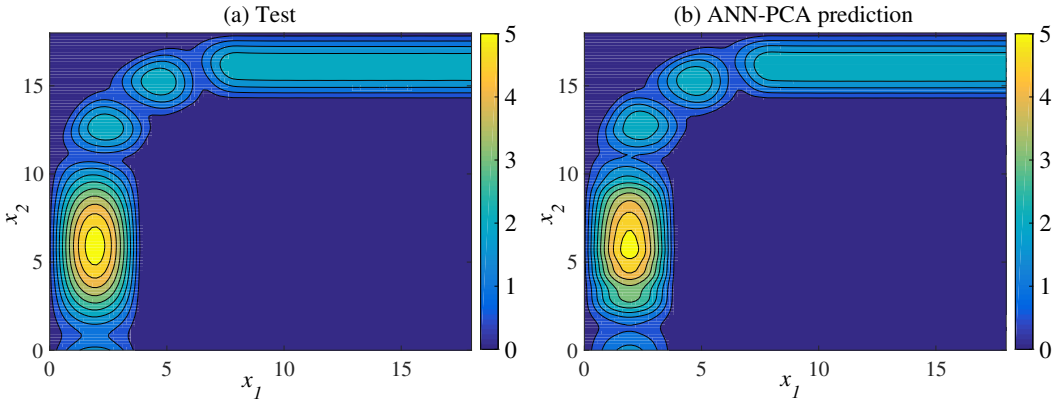

Figure 5.3: Representative examples of prediction using ANN with PCA ($r = 5$, $M = 60$) for the 2D waveguide.

background field is swept over a range of angles of incidence and the far-field and radar cross section (RCS) are computed. The 2D geometry consists of an inner circle containing the boat and the surrounding air, together with an outer circle representing a perfectly matched layer (PML). The background electromagnetic field from the radar is described by its out-of-plane electric field component:

$$\boldsymbol{E}_b = \exp(ik_0(\xi\cos\theta + \eta\sin\theta))\boldsymbol{e}_z, \tag{5.4}$$

where $k_0 = 2\pi f/c$ is the wave number in vacuum, c is the speed of light, f is the frequency and θ is the angle of incidence. The time-harmonic wave equation is then solved for the relative field, $\boldsymbol{E}_{rel} = \boldsymbol{E} - \boldsymbol{E}_b$, where $\boldsymbol{E}$ is the total field:

$$\nabla \times \left(\mu_r^{-1}\nabla \times \boldsymbol{E}_{rel}\right) - \left(\epsilon_r - \frac{i\sigma}{2\pi f\epsilon_0}\right) k_0^2 \boldsymbol{E}_{rel} = 0, \tag{5.5}$$

in which ϵ, μ and σ denote the permittivity, permeability, and conductivity of air, respectively (subscripts r denote a 'relative' quantity). The RCS per unit length is defined as

$$\sigma_{2D} = \lim_{r\to\infty} 2\pi r \frac{|\boldsymbol{E}_{rel}|^2}{|\boldsymbol{E}|^2}$$

The model was solved ('Radar Cross Section' under the Radio Frequency module in COMSOL Multiphysics 5.0) for 500 combinations of f and θ as input values; that is $\boldsymbol{\xi}^{(i)} = (f^{(i)}, \theta^{(i)})^T$, $i = 1, \ldots, 500$. The magnitude of the electric field $\boldsymbol{E}$ was recorded on a regular 500×500 square spatial grid in (ξ, η). The $d = 2.5 \times 10^5$ values of $|\boldsymbol{E}(x_1, x_2)|$ for each $\boldsymbol{x}^{(i)}$ at locations (x_{1i}, x_{2j}), $i, j = 1, \ldots, 500$, were vectorized to form the data points $\boldsymbol{y}^{(i)} \in \mathbb{R}^d$ used for testing and training.

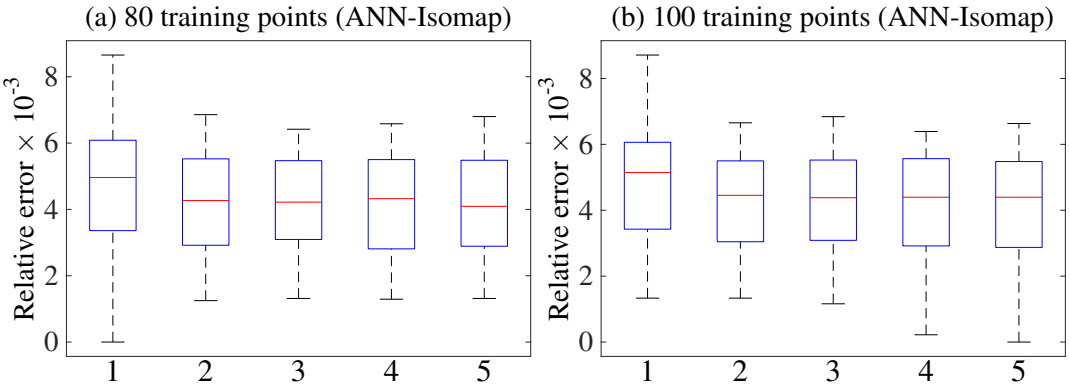

Figure 5.4: Boxplots of the relative errors for different numbers of components (r) using ANN with Isomap ($M = 80$ and 100) in the RCS example.

Results

PCA with both ANN and GPE (method of [1]) failed to provide meaningful results for any number of training points m or components r. ANN with Isomap and kPCA, on the other hand, exhibited good results, especially in the case of Isomap for $m > 60$, which captured the trends precisely. Box plots of the relative square errors are shown in Fig. 5.4 (a) and (b), up to 5 components (beyond which no improvements were visible). Fig. 5.5 shows two representative examples of the predictions using ANN with Isomap ($r = 5$ and $m = 100$). In the first case, the relative error is 6.4×10^{-3} (near the maximum) and in the second case the relative error is 2.2×10^{-3}.

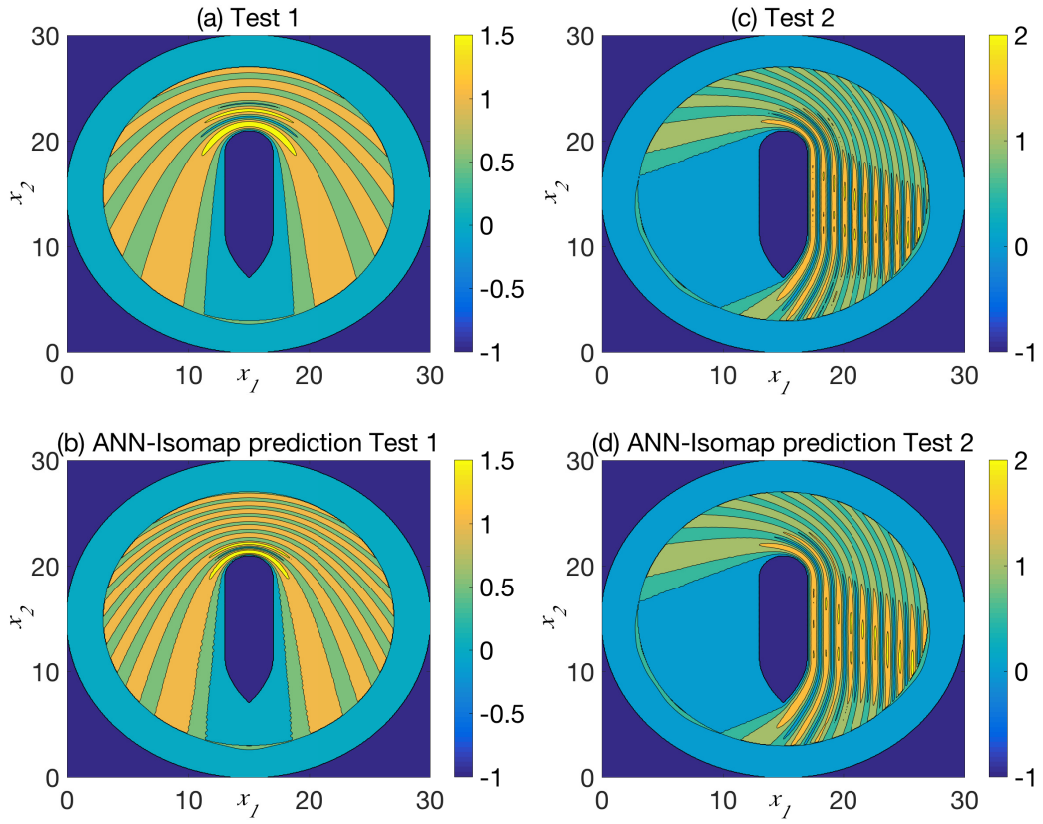

Figure 5.5: Representative examples of the predictions using ANN with Isomap ($r = 5$) and 100 training points in the RCS example.

5.3 Concluding remarks

Unlike GPE, ANNs do not place assumptions on the interdependency of the coefficients generated by manifold learning algorithms. This can potentially lead to more accurate predictions and is particularly useful when using manifold learning methods does not generate independent coefficients, e.g., diffusion maps. The total learning and prediction time for all methods is on the order of a few minutes, which is a dramatic reduction in computational effort compared to the emulation of d outputs simultaneously, which, in many cases, would not be computationally feasible.

Results were compared in test cases to Higdon's method [1]. In both cases

(and in many other data sets tested) this standard method fails, while the linear and/or non-linear methods exhibit accurate results. The second example demonstrates again that manifold learning is necessary for some complex datasets as the linear method such as PCA would fail or perform badly unless increasing the reduced dimensionality to a very high number. This may somehow defeat the purpose of reducing the computational cost as our motivations. There are also cases where manifold learning method would be inferior to linear method, e.g., PCA or MDS. Usually the response surfaces in these cases are 'flat' where linear method would suffice. In such a situation, a linear method could outperform a manifold learning method easily to some extent due to the advantages of its existing general pre-image solutions.

Chapter 6

Reduced order modelling via manifold learning based Gaussian process emulators

Proper orthogonal decomposition (POD) (see chapter 2) is a popular reduced-order model (ROM) method for solving PDE models. As discussed in Chapter 1, for dynamic parameterized PDE models, the main challenge is to accurately and efficiently approximate the POD basis for new parameter values, in order to use the ROM model for applications such as uncertainty quantification and real-time control. A further challenge is posed by nonlinearities, which impede computational efficiency in the absence of further approximations. In this chapter, linear and nonlinear parameterized dynamic PDE models are considered. A method that extends POD (in time) is presented to parametric problems by using our GP manifold learning framework to predict the snapshots for new parameter values. The same method is also used to extend the discrete empirical interpolation method (DEIM) to parameterized nonlinearities.

Firstly, procedures for generating ROMs and POD bases are outlined and

details of the DEIM method are also provided. The issues associated with parameterized and/or nonlinear problems and outline our approach are explained. In Section 6.3, brief details of the feature-space emulation and inverse mapping (already explained fully in previous chapters) are provided. The approach to both linear and nonlinear parameterized problems is summarised in Algorithm 9. In section 6.5, two examples are presented, one linear and one nonlinear. The results are compared to a global basis approach. Conclusion are drawn in Section 6.6.

This chapter forms the basis for the paper, A. A. Shah, W. W. Xing, V. Triantafyllidis. Reduced-order modelling of parameter-dependent, linear and nonlinear dynamic partial differential equation models. *Proc. R. Soc.* A 2017 473 20160809; DOI: 10.1098/rspa.2016.0809. Published 26 April 2017.

6.1 Problem definition and Galerkin projection

Let $\boldsymbol{x} = (x_1, \ldots, x_L)$ denote a point in a bounded, regular domain in $\mathcal{D} \subset \mathbb{R}^L$ $(L = 1, 2, 3)$, let $t \in [0, T]$ denote time and let $\boldsymbol{\xi} \in \mathcal{X} \subset \mathbb{R}^l$ denote a vector of parameters. For the purposes of illustration, consider a parameterised, parabolic PDE for a dependent variable (field) $u(\boldsymbol{x}, t; \boldsymbol{\xi})$:

$$\begin{aligned} \partial_t u + \mathcal{L}(\boldsymbol{\xi})u + \mathcal{N}(\boldsymbol{\xi})u = g(\boldsymbol{x};\boldsymbol{\xi}) \quad (\boldsymbol{x}, t) \in \mathcal{D} \times (0, T], \\ u(\boldsymbol{x}, 0; \boldsymbol{\xi}) = u_0(\boldsymbol{x};\boldsymbol{\xi}) \quad \boldsymbol{x} \in \mathcal{D}, \end{aligned} \tag{6.1}$$

augmented by linear boundary conditions. Here, $\mathcal{L}(\boldsymbol{\xi})$ and $\mathcal{N}(\boldsymbol{\xi})$ are parameter dependent linear and nonlinear partial differential operators, respectively. The dependence on the parameters can be through the operators, the source term $g(\boldsymbol{x};\boldsymbol{\xi})$ or the initial/boundary conditions.

Let $\mathcal{H}$ be a separable Hilbert space with inner product $(\cdot, \cdot)_{\mathcal{H}}$ and induced norm $||\cdot||_{\mathcal{H}}$, e.g., $L^2(\mathcal{D})$, the space of square integrable equivalence classes of func-

tions with inner product:

$$(v, v')_{L^2(\mathcal{D})} = \int_{\mathcal{D}} v(\boldsymbol{x})v'(\boldsymbol{x})d\boldsymbol{x}. \tag{6.2}$$

From hereon, the subscript in the notation is dropped for the inner product and norm in $L^2(\mathcal{D})$. It is assumed that for each $\boldsymbol{\xi}$, $u \in L^2(0,T;\mathcal{H})$, i.e., $t \mapsto u(\cdot,t;\boldsymbol{\xi})$ is a measurable map from $(0,T)$ to $\mathcal{H}$ with finite norm:

$$||u||_{L^2(0,T;\mathcal{H})} := \int_0^T ||u(\cdot,t;\boldsymbol{\xi})||_{\mathcal{H}}. \tag{6.3}$$

Then $u(\cdot,t;\boldsymbol{\xi}) \in \mathcal{H}$ for each $t \in (0,T)$. A spatial discretization of (6.1) leads to a system of ODEs in time:

$$\begin{aligned} \dot{\boldsymbol{u}}(t;\boldsymbol{\xi}) &= \mathbf{A}(\boldsymbol{\xi})\boldsymbol{u}(t;\boldsymbol{\xi}) + \boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\xi});\boldsymbol{\xi}), \\ \boldsymbol{u}(0;\boldsymbol{\xi}) &= \boldsymbol{u}_0(\boldsymbol{\xi}), \end{aligned} \tag{6.4}$$

for a discrete state variable $\boldsymbol{u}(t;\boldsymbol{\xi}) = (u_1(t;\boldsymbol{\xi}),\ldots,u_d(t;\boldsymbol{\xi}))^T$, which is called the *solution vector*. d is the number of degrees of freedom, e.g., the number of grid points in a finite difference (FD) approximation, the number of cells in a cell-centred finite volume (FV) approximation or the number of nodes (basis functions) in a finite-element (FE) approximation. The matrix $\mathbf{A}(\boldsymbol{\xi}) \in \mathbb{R}^{d\times d}$ arises from the linear term $\mathcal{L}(\boldsymbol{\xi})$ and $\boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\xi});\boldsymbol{\xi}) \in \mathbb{R}^d$ arises from $\mathcal{N}(\boldsymbol{\xi})u$, $g(\boldsymbol{x};\boldsymbol{\xi})$ and possibly the boundary conditions. It is nonlinear for $\mathcal{N}(\boldsymbol{\xi})u \neq 0$.

The precise relationship between $\boldsymbol{u}(t;\boldsymbol{\xi})$ and $u(\boldsymbol{x},t;\boldsymbol{\xi})$, the forms of $\mathbf{A}(\boldsymbol{\xi})$ and $\boldsymbol{f}(\boldsymbol{u};\boldsymbol{\xi})$ and the incorporation of boundary conditions depends on the method used. For a FD approximation, problem (6.1) is solved directly and the boundary conditions are incorporated in $\boldsymbol{f}(\boldsymbol{u};\boldsymbol{\xi})$. In a FE approximation a weak form is solved with test functions in $\mathcal{H}$ or a dense subspace $\mathcal{V}$ of $\mathcal{H}$, with boundary conditions incorporated in $\boldsymbol{f}$ and/or the definition of $\mathcal{H}$. Solutions are then sought in a finite

dimensional subspace of $\mathcal{H}$ or $\mathcal{V}$.

The form of $\mathbf{A}(\boldsymbol{\xi})$ is determined by the form of $\mathcal{L}(\boldsymbol{\xi})u$. In some cases it admits an affine form or can be approximated as such:

$$\mathbf{A}(\boldsymbol{\xi}) = \sum_i c_i(\boldsymbol{\xi})\mathbf{A}_i, \tag{6.5}$$

where the functions $c_i(\boldsymbol{\xi})$ are known and the matrices $\mathbf{A}_i$ are constant. For FD, FV and nodal-basis FE discretizations, the coefficients $u_i(t;\boldsymbol{\xi})$ of $\boldsymbol{u}(t;\boldsymbol{\xi})$ correspond to the values of $u(\boldsymbol{x},t;\boldsymbol{\xi})$ at locations $\boldsymbol{x}^{(i)} \in \overline{\mathcal{D}}$, $i = 1,\ldots,d$, i.e.:

$$u_i(t;\boldsymbol{\xi}) = u(\boldsymbol{x}^{(i)},t;\boldsymbol{\xi}). \tag{6.6}$$

It is assumed to be the case here. A numerical solution of (6.4) yields the solution vector $\boldsymbol{u}(t;\boldsymbol{\xi}) = (u_1(t;\boldsymbol{\xi}),\ldots,u_d(t;\boldsymbol{\xi}))^T$ at times $\{t^{(i)}\}_{i=1}^m$. Each of the discrete solutions:

$$\boldsymbol{u}^{(i)}(\boldsymbol{\xi}) := \boldsymbol{u}(t^{(i)};\boldsymbol{\xi}) \in \mathbb{R}^d, \quad i = 1,\ldots,m, \tag{6.7}$$

is referred to as a *snapshot*.

For a fixed input $\boldsymbol{\xi} \in \mathcal{X}$, a Galerkin projection approximates the problem (6.4) in a proper (low dimensional) subspace $\mathcal{S}$ of $\mathbb{R}^d$. Let $\boldsymbol{v}_j(\boldsymbol{\xi}) \in \mathbb{R}^d$, $j = 1,\ldots,r$ be an orthonormal basis for $\mathcal{S}$ ($\dim(\mathcal{S}) = r \ll d$), where the notation makes explicit the dependence on the input. An approximation $\boldsymbol{u}_r(t;\boldsymbol{\xi}) \in \mathcal{S}$ of $\boldsymbol{u}$ in the space $\text{span}(\boldsymbol{v}_1(\boldsymbol{\xi}),\ldots,\boldsymbol{v}_r(\boldsymbol{\xi}))$ is assumed as:

$$\boldsymbol{u}_r(t;\boldsymbol{\xi}) = \sum_{j=1}^r a_j(t;\boldsymbol{\xi})\boldsymbol{v}_j(\boldsymbol{\xi}) = \mathbf{V}_r(\boldsymbol{\xi})\boldsymbol{a}(t;\boldsymbol{\xi}), \tag{6.8}$$

where:

$$\begin{aligned} \boldsymbol{a} &= (a_1(t;\boldsymbol{\xi}),\ldots,a_r(t;\boldsymbol{\xi}))^T, \\ \mathbf{V}_r(\boldsymbol{\xi}) &= [\boldsymbol{v}_1(\boldsymbol{\xi})\ldots\boldsymbol{v}_r(\boldsymbol{\xi})]. \end{aligned} \tag{6.9}$$

The Galerkin projection of equation (6.4) onto the basis vectors $\boldsymbol{v}_i(\boldsymbol{\xi})$, $i = 1, \ldots, r$, yields (replacing $\boldsymbol{u}$ with $\boldsymbol{u}_r$):

$$\begin{aligned} \dot{\boldsymbol{a}}(t;\boldsymbol{\xi}) &= \mathbf{A}_r(\boldsymbol{\xi})\boldsymbol{a}(t;\boldsymbol{\xi}) + \boldsymbol{f}_r\left(\boldsymbol{a}(t;\boldsymbol{\xi});\boldsymbol{\xi}\right), \\ \boldsymbol{a}(0;\boldsymbol{\xi}) &= \boldsymbol{a}_0(\boldsymbol{\xi}) := \mathbf{V}_r(\boldsymbol{\xi})^T\boldsymbol{u}_0(\boldsymbol{\xi}), \end{aligned} \tag{6.10}$$

where:

$$\begin{aligned} \mathbf{A}_r(\boldsymbol{\xi}) &:= \mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{A}(\boldsymbol{\xi})\mathbf{V}_r(\boldsymbol{\xi}), \\ \boldsymbol{f}_r\left(\boldsymbol{a}(t;\boldsymbol{\xi});\boldsymbol{\xi}\right) &:= \mathbf{V}_r(\boldsymbol{\xi})^T\boldsymbol{f}\left(\mathbf{V}_r(\boldsymbol{\xi})\boldsymbol{a}(t;\boldsymbol{\xi});\boldsymbol{\xi}\right). \end{aligned} \tag{6.11}$$

Equations (6.10) represent a system of r ODEs in time for the coefficients $a_i(t;\boldsymbol{\xi})$. The basic premise of POD, detailed in chapter 2, is the construction of a basis $\{\boldsymbol{v}_j(\boldsymbol{\xi})\}_{j=1}^r$ using the discrete snapshots $\{\boldsymbol{u}^{(i)}(\boldsymbol{\xi})\}_{i=1}^m$.

6.2 Basis emulation and modified DEIM

The ROM is valid for a fixed input $\boldsymbol{\xi} \in \mathcal{X}$; that is, for each input the snapshot matrix $\mathbf{X}(\boldsymbol{\xi})$ is obtained from the FOM and the basis $\mathbf{V}_r(\boldsymbol{\xi})$ is constructed according to section 6.12.6. To perform an analysis with respect to the parameters (e.g., uncertainty quantification), this procedure is computationally prohibitive. A global basis across the parameter space of interest [44, 45] can be constructed by using a design-of-experiment (such as a Latin hypercube or Sobol sequence [126]) to select values $\boldsymbol{\xi}^{(j)} \in \mathcal{X}$, $j = 1, \ldots, n$, in order to compute a set of snapshot matrices $\mathbf{X}(\boldsymbol{\xi}^{(j)})$. The POD basis vectors are then extracted from the global snapshot matrix:

$$[\mathbf{X}(\boldsymbol{\xi}^{(1)}), \ldots, \mathbf{X}(\boldsymbol{\xi}^{(n)})] \in \mathbb{R}^{d \times nm}, \tag{6.12}$$

usually after a SVD to avoid rank deficiency. A consequence of this approach is that the optimality of the POD method is violated. Furthermore, the range of validity of the global basis could be limited for highly nonlinear mappings between

the parameters and the outputs [48].

The computational cost of solving the reduced order system (6.10) also depends on the form of $\boldsymbol{f}(\cdot;\boldsymbol{\xi})$ in (6.4). When $\boldsymbol{f}(\cdot;\boldsymbol{\xi}) \in \mathbb{R}^d$ is a strong nonlinearity, the computational efficiency of the POD-Galerkin approach is compromised by the fact that evaluation of $\boldsymbol{f}_r$ in (6.10) has a computational complexity that depends on d. In DEIM [63], one collects snapshots of the nonlinearity $\{\boldsymbol{f}(\boldsymbol{u}(t^{(i)};\boldsymbol{\xi});\boldsymbol{\xi})\}_{i=1}^m$ and form the matrix:

$$\mathbf{F}(\boldsymbol{\xi}) = [\boldsymbol{f}(\boldsymbol{u}(t^{(1)};\boldsymbol{\xi}) \dots \boldsymbol{f}(\boldsymbol{u}(t^{(m)};\boldsymbol{\xi})]. \tag{6.13}$$

For a nonlinear $\boldsymbol{f} \in \mathbb{R}^d$, the DEIM method seeks a set of vectors $\boldsymbol{w}_i(\boldsymbol{\xi}) \in \mathbb{R}^d$, $i = 1, \dots, d$, such that the subspace $\text{spans}(\boldsymbol{w}_1(\boldsymbol{\xi}), \dots, \boldsymbol{w}_s(\boldsymbol{\xi})) \subset \mathbb{R}^d$ for some $s \ll d$ well approximates $\boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\xi});\boldsymbol{\xi})$ for an arbitrary t. That is, an approximation:

$$\boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\xi});\boldsymbol{\xi}) \approx \mathbf{W}(\boldsymbol{\xi})\boldsymbol{h}(t;\boldsymbol{\xi}), \tag{6.14}$$

where:

$$\begin{gathered} \mathbf{W}(\boldsymbol{\xi}) = [\boldsymbol{w}_1(\boldsymbol{\xi}) \dots \boldsymbol{w}_s(\boldsymbol{\xi})], \\ \boldsymbol{h}(t;\boldsymbol{\xi}) \in \mathbb{R}^s. \end{gathered} \tag{6.15}$$

The basis $\{\boldsymbol{w}_i(\boldsymbol{\xi})\}_{i=1}^d$ is obtained from a PCA on $\mathbf{F}(\boldsymbol{\xi})\mathbf{F}(\boldsymbol{\xi})^T$ or SVD of $\mathbf{F}(\boldsymbol{\xi})$, and is arranged according to the sizes of the corresponding eigenvalues (in descending order).

Since the system $\boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\xi});\boldsymbol{\xi}) = \mathbf{W}(\boldsymbol{\xi})\boldsymbol{h}(t;\boldsymbol{\xi})$ is overdetermined in $\boldsymbol{h}(t;\boldsymbol{\xi})$, the DEIM method selects s of the d equations to obtain an 'optimal' solution. Let us introduce the matrix $\mathbf{P} = [\boldsymbol{e}_{p_1} \dots \boldsymbol{e}_{p_s}] \in \mathbb{R}^{d\times s}$, where $\boldsymbol{e}_{p_i}$ is the standard Euclidean basis vector in $\mathbb{R}^d$ with nonzero entry located at the p_i-th coordinate. Then, assuming $\mathbf{P}^T\mathbf{W}(\boldsymbol{\xi})$ is nonsingular, one obtains:

$$\boldsymbol{h}(t;\boldsymbol{\xi}) \approx (\mathbf{P}^T\mathbf{W}(\boldsymbol{\xi}))^{-1}\mathbf{P}^T\boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\xi});\boldsymbol{\xi}), \tag{6.16}$$

so that:

$$\begin{aligned}
\boldsymbol{f}_r\left(\boldsymbol{a}(t;\boldsymbol{\xi});\boldsymbol{\xi}\right) &= \mathbf{V}_r(\boldsymbol{\xi})^T \boldsymbol{f}\left(\mathbf{V}_r(\boldsymbol{\xi})\boldsymbol{a}(t;\boldsymbol{\xi});\boldsymbol{\xi}\right) \\
&\approx \mathbf{V}_r(\boldsymbol{\xi})^T \mathbf{W}(\boldsymbol{\xi})\boldsymbol{h}(t;\boldsymbol{\xi}) \\
&= \mathbf{V}_r(\boldsymbol{\xi})^T \mathbf{W}(\boldsymbol{\xi})(\mathbf{P}^T\mathbf{W}(\boldsymbol{\xi}))^{-1}\mathbf{P}^T\boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\xi});\boldsymbol{\xi}) \\
&= \mathbf{V}_r(\boldsymbol{\xi})^T \mathbf{W}(\boldsymbol{\xi})(\mathbf{P}^T\mathbf{W}(\boldsymbol{\xi}))^{-1}\boldsymbol{f}(\mathbf{P}^T\boldsymbol{u}(t;\boldsymbol{\xi});\boldsymbol{\xi}),
\end{aligned} \tag{6.17}$$

assuming that the function $\boldsymbol{f}\left(\cdot;\boldsymbol{\xi}\right)$ acts pointwise. The indices $p_i \in \{1, 2, \ldots, d\}$, $i = 1, \ldots, s$ are specified by using an iterative algorithm to limit the growth of the standard Euclidean norm error in the approximation of $\boldsymbol{f}(\boldsymbol{u}(t;\boldsymbol{\xi});\boldsymbol{\xi})$ in the space Range($\mathbf{W}(\boldsymbol{\xi})$) [63].

In this chapter,a systematic and rigorous method are introduced to approximate local bases and the nonlinearity by first approximating the snapshots $\{\boldsymbol{u}(t^{(i)};\boldsymbol{\xi})\}_{i=1}^m$ and $\{\boldsymbol{f}(\boldsymbol{u}(t^{(i)};\boldsymbol{\xi});\boldsymbol{\xi})\}_{i=1}^m$ for an arbitrary input $\boldsymbol{\xi}$ using the methods described in chapters 3 to 5, using any of the regression and manifold learning methods. Details are not repeated here, except that the learning problem is made explicit below.

6.3 Formulation and solution of the learning problem

For an arbitrary input $\boldsymbol{\xi}$, the PDE model is a mapping $\boldsymbol{\eta} : \mathcal{X} \to \mathcal{O} \subset \mathbb{R}^{md}$, defined as follows:

$$\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{\xi}) = \left(\boldsymbol{u}(t^{(1)};\boldsymbol{\xi})^T, \ldots, \boldsymbol{u}(t^{(m)};\boldsymbol{\xi})^T\right)^T, \tag{6.18}$$

i.e., a vectorial rearrangement of snapshots for the given value of $\boldsymbol{\xi}$. It is often desired to approximate the mapping $\boldsymbol{\eta}(\cdot)$ given training points $\boldsymbol{y}^{(j)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(j)}) \in \mathcal{O}$, with corresponding design points $\boldsymbol{\xi}^{(j)} \in \mathcal{X}$, $j = 1, \ldots, n$. As discussed in previous chapters, the main methods for dealing with such high dimensional (large md) output problems are those of Conti and O'Hagan [27] and Higdon *et al.* [1]. The method

of Higdon *et al.* was extended in the previous 3 chapters by replacing PCA with nonlinear methods. In this chapter, this method to learn the snapshots $\boldsymbol{y}$ for new parameter values $\boldsymbol{\xi}$ is employed. The same approach is also used for snapshots of the nonlinearity, $\{\boldsymbol{f}(\boldsymbol{u}(t^{(i)};\boldsymbol{\xi});\boldsymbol{\xi})\}_{i=1}^{m}$, similarly arranged as in (6.18). The default manifold learning method is kPCA.

Given training data $\boldsymbol{y}^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)}) \in \mathcal{O}$, $i = 1, \ldots, n$ at design points $\boldsymbol{\xi}^{(i)} \in \mathcal{X}$, kPCA defines a map $\boldsymbol{\phi} : \mathcal{O} \to \mathscr{F}$, where $\mathscr{F}$ is a feature space. It is equivalent (see section 4.1) to composite mappings:

$$\mathbb{z}_i(\cdot) := z_i(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \to \mathbb{R}, \quad i = 1, \ldots, n, \tag{6.19}$$

of the inputs $\boldsymbol{\xi}$ to a low number (q) of coefficients. For the data points, the coefficients $z_i(\boldsymbol{y}^{(j)})$ are defined as in Eq. (4.9). One can also define the map:

$$\boldsymbol{z}_q(\boldsymbol{\eta}(\cdot)) := (\mathbb{z}_1(\cdot), \ldots, \mathbb{z}_q(\cdot))^T : \mathcal{X} \to \mathbb{R}^q, \tag{6.20}$$

from the input space $\mathcal{X}$ to a low dimensional space $\mathbb{R}^q$. Note that q is used to represent the dimensionality of the low-dimensional space in this case, as opposed to r in previous chapters. Here r is reserved for the dimensionality of the approximating POD space.

Approximating $\boldsymbol{\eta} : \mathcal{X} \to \mathcal{M}$ given the training points $\{\boldsymbol{y}^{(j)}\}_{j=1}^{n}$ is thus replaced by the problem of approximating $\boldsymbol{z}_q(\boldsymbol{\eta}(\cdot))$. This is achieved as in chapter 4 by placing univariate GP priors indexed by $\boldsymbol{\xi}$ over the individual coefficients $\mathbb{z}_i(\cdot)$, assuming that they are mutually uncorrelated. The training data is obtained by applying kPCA to the training set; that is:

$$\boldsymbol{z}_q(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})) = \boldsymbol{z}_q(\boldsymbol{y}^{(j)}) = [\widetilde{\boldsymbol{\alpha}}_1 \ldots \widetilde{\boldsymbol{\alpha}}_q]^T \mathbf{H}(\boldsymbol{k}_j - \mathbf{K}\mathbf{1}), \quad j = 1, \ldots, n, \tag{6.21}$$

with terms defined as in Eq. (4.20).

An inverse mapping is again given by a weighted average of the training points: $\boldsymbol{y} = \sum_{j\in\mathcal{J}} \vartheta(\boldsymbol{y}^{(j)})\boldsymbol{y}^{(j)}$, with weights $\vartheta(\boldsymbol{y}^{(j)})$, as in Eq. (4.49), where $\mathcal{J} \subseteq \{1, 2, \ldots, n\}$, defines the neighbouring points used for the averaging. The weights are defined as in Eq. (6.22) in terms of the distances $d_{j,*}$, $j = 1, \ldots, n$, between $\boldsymbol{y}$ and $\boldsymbol{y}^{(j)}$:

$$\rho(\boldsymbol{y}^{(j)}) = \frac{\chi(\boldsymbol{y}, \boldsymbol{y}^{(j)})}{\sum_{i\in\mathcal{J}} \chi(\boldsymbol{y}, \boldsymbol{y}^{(i)})} = \frac{\chi(d_{i,*})}{\sum_{i\in\mathcal{J}} \chi(d_{i,*})}, \quad j \in \mathcal{J}, \tag{6.22}$$

where $\chi(\boldsymbol{y}, \boldsymbol{y}') = \exp(-||\boldsymbol{y} - \boldsymbol{y}'||^2)$. The distances $d_{i,*}$ are calculated as in section 4.4.1.

6.4 Main Algorithm

As already stated, the kPCA coefficients for the snapshots of the field variable and the nonlinearity are assumed to be realisations of independent GPs. GPE is therefore performed separately on each coefficient to approximate its value for a new input $\boldsymbol{\xi}$. This provides the new snapshots, which are then used for POD/DEIM. A pseudo code is presented below. For convenience, the terminology 'kGPE-POD' is used to denote the method of Algorithm 9 without the modified DEIM. With the modified DEIM, the terminology 'kGPE-POD-DEIM' is instead used as shorthand.

Algorithm 9 POD-ROM for parameterised PDEs using manifold learning and DEIM.

1: Obtain snapshots from FOM using DOE: $\boldsymbol{u}(t^{(j)};\boldsymbol{\xi}^{(i)})^T$, $i = 1,\ldots,n$, $j = 1,\ldots,m$.

2: *For nonlinear problems, collect snapshots*: $\boldsymbol{f}(\boldsymbol{u}(t^{(j)};\boldsymbol{\xi}^{(i)});\boldsymbol{\xi}^{(i)})$, $i = 1,\ldots,n$, $j = 1,\ldots,m$.

3: Set: $\boldsymbol{y}^{(i)} \leftarrow \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)}) \leftarrow (\boldsymbol{u}(t^{(1)};\boldsymbol{\xi}^{(i)})^T,\ldots,\boldsymbol{u}(t^{(m)};\boldsymbol{\xi}^{(i)})^T)^T$, $i = 1,\ldots,n$.

4: *For nonlinear problems, set*:

$$\boldsymbol{y}_f^{(i)} \leftarrow \boldsymbol{\eta}_f(\boldsymbol{\xi}^{(i)}) \leftarrow (\boldsymbol{f}(\boldsymbol{u}(t^{(1)};\boldsymbol{\xi}^{(i)});\boldsymbol{\xi}^{(i)})^T,\ldots,\boldsymbol{f}(\boldsymbol{u}(t^{(m)};\boldsymbol{\xi}^{(i)});\boldsymbol{\xi}^{(i)})^T)^T$$

$i = 1,\ldots,n$

5: Do kPCA for $\{\boldsymbol{y}^{(i)}\}_{i=1}^n \rightarrow \{(z_1(\boldsymbol{y}^{(i)}),\ldots,z_q(\boldsymbol{y}^{(i)}))^T\}_{i=1}^n$

6: *For nonlinear problems do kPCA for* $\{\boldsymbol{y}_f^{(i)}\}_{i=1}^n \rightarrow \{(z_1^f(\boldsymbol{y}_f^{(i)}),\ldots,z_q^f(\boldsymbol{y}_f^{(i)}))^T\}_{i=1}^n$

7: **for** $j \leftarrow 1$ to q **do**

$$\{\eta(\boldsymbol{\xi}^{(i)}) \leftarrow \mathbb{z}_j(\boldsymbol{\xi}^{(i)}) \leftarrow z_j(\boldsymbol{\eta}(\boldsymbol{\xi}^{(i)})) \leftarrow z_j(\boldsymbol{y}^{(i)})\}_{i=1}^n$$

$$\text{Scalar GPE: } \mathbb{z}_j(\boldsymbol{\xi}) \leftarrow \mathbb{E}[\eta(\boldsymbol{\xi})]$$

For nonlinear problems:

$$\{\eta_f(\boldsymbol{\xi}^{(i)}) \leftarrow \mathbb{z}_j^f(\boldsymbol{\xi}^{(i)}) \leftarrow z_j^f(\boldsymbol{\eta}_f(\boldsymbol{\xi}^{(i)})) \leftarrow z_j^f(\boldsymbol{y}_f^{(i)})\}_{i=1}^n$$

$$\text{Scalar GPE: } \mathbb{z}_j^f(\boldsymbol{\xi}) \leftarrow \mathbb{E}[\eta_f(\boldsymbol{\xi})]$$

8: **end for**

9: Inverse map: $\boldsymbol{\eta}(\boldsymbol{\xi}) \leftarrow \sum_{i=1}^{N_n}\left(\chi(d_{i,*})/\sum_{i=1}^{N_n}\chi(d_{i,*})\right)\boldsymbol{y}^{(i)}$

10: *For nonlinear problems*: $\boldsymbol{\eta}_f(\boldsymbol{\xi}) \leftarrow \sum_{i=1}^{N_n}\left(\chi(d_{i,*})/\sum_{i=1}^{N_n}\chi(d_{i,*})\right)\boldsymbol{y}_f^{(i)}$

11: Snapshots for input $\boldsymbol{\xi}$: $(\boldsymbol{u}(t^{(1)};\boldsymbol{\xi})^T,\ldots,\boldsymbol{u}(t^{(m)};\boldsymbol{\xi})^T)^T \leftarrow \boldsymbol{\eta}(\boldsymbol{\xi})$

12: *For nonlinear problems*: $(\boldsymbol{f}(\boldsymbol{u}(t^{(1)};\boldsymbol{\xi});\boldsymbol{\xi})^T,\ldots,\boldsymbol{f}(\boldsymbol{u}(t^{(m)};\boldsymbol{\xi});\boldsymbol{\xi})^T)^T \leftarrow \boldsymbol{\eta}_f(\boldsymbol{\xi})$

13: POD with $\{\boldsymbol{u}(t^{(i)};\boldsymbol{\xi})\}_{i=1}^m$

14: *For nonlinear problems: combine 13 with DEIM on* $\{\boldsymbol{f}(\boldsymbol{u}(t^{(i)};\boldsymbol{\xi});\boldsymbol{\xi})\}_{i=1}^m$

6.5 Results and discussion

6.5.1 2D contaminant transport

The transport of a contaminant governed by a convection-diffusion equation. is tested here. This model can be used, e.g., for real-time prediction or for quantifying uncertainty in the concentration to support decision making [46]. The problem is specified as follows:

$$\begin{gathered}\partial_t u + \boldsymbol{q} \cdot \nabla u - \mu \nabla^2 u = 0 \quad \boldsymbol{x} = (x_1, x_2) \in \mathcal{D} := [0,1] \times [0,1] \\ u = 0 \quad \boldsymbol{x} \in \partial\mathcal{D}, \quad u(\boldsymbol{x}, t) = u_0 \quad t = 0\end{gathered} \tag{6.23}$$

where $u(\boldsymbol{x}, t; \boldsymbol{\xi})$ denotes the contaminant concentration (mol m^{-3}), $\boldsymbol{q}$ is the fluid velocity (m s^{-1}) and μ is the contaminant diffusion coefficient (m^2 s^{-1}). The input $\boldsymbol{\xi}$ is defined below. The initial concentration is given by $u_0(\boldsymbol{x}) = (2\pi k_0)^{-1/2} \sum_{i=1}^{3} k_i \exp(-k_0(\boldsymbol{x} - \boldsymbol{x}_i)^T(\boldsymbol{x} - \boldsymbol{x}_i)/2)$, where $\boldsymbol{x}_1 = (0.2, 0.2)^T$, $\boldsymbol{x}_2 = (0.2, 0.8)^T$, $\boldsymbol{x}_3 = (0.8, 0.8)^T$, $k_0 = 0.01$, $k_1 = 1$, $k_2 = 2$ and $k_3 = 3$. The magnitude of the velocity field is inversely proportional to the distance from $\boldsymbol{x} = (\widehat{x}_1, \widehat{x}_2)^T$:

$$\boldsymbol{q}(\boldsymbol{x}) = \frac{a_1(x_1 - \widehat{x}_1)\boldsymbol{e}_1 + a_2(x_2 - \widehat{x}_2)\boldsymbol{e}_2}{(x_1 - \widehat{x}_1)^2 + (x_2 - \widehat{x}_2)^2} \tag{6.24}$$

where $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are unit vectors in the x_1 and x_2 directions, respectively, and $a_i \in \mathbb{R}$. To avoid the singularity at $\boldsymbol{x} = (\widehat{x}_1, \widehat{x}_2)^T$, the norm of velocity is set to zero at this location. It is also setted $a_1 = a_2 = 1$ and $\mu = 1$, and consider variations in the input $\boldsymbol{\xi} = (\widehat{x}_1, \widehat{x}_2)^T \in \mathcal{X} := [0,1] \times [0,1]$.

The problem was discretized in space using a cell-centered finite volume method with $d = 2500$ square cells (control volumes). Central differencing was used for the diffusive term and a first-order upwind scheme for the convective term, defining the velocity values on a staggered grid. A fully implicit Euler method was used to solve the resulting semi-discrete linear problem with 100 equal time steps in

$t \in [0, T]$, $T = 0.2$ s. A total of 500 inputs $\boldsymbol{\xi}_j \in \mathcal{X}$, $j = 1, \ldots, 500$, were generated using a Sobol sequence [126]. For each input, the FOM was solved to yield solution vectors (snapshots) $\boldsymbol{u}_i(\boldsymbol{\xi}_j) \in \mathbb{R}^d$, $i = 1, \ldots, 100$, $j = 1, \ldots, 500$. The data points (vectorized snapshots) $\boldsymbol{y}_j = \boldsymbol{\eta}(\boldsymbol{\xi}_j)$, $j = 1, \ldots, 500$, were obtained using Eq. (6.18). $\mathcal{H} = L^2(\mathcal{D})$ is setted to define the POD basis and optimality. Of the 500 data points, $n_t = 300$ were reserved for testing. Training points were selected from the remaining 200 data points ($n \leq 200$).

A Gaussian kernel was used for kPCA. The free parameter s^2 was taken to be the average square distance between observations in the original space [110]: $s^2 = n^{-2} \sum_{i,j=1}^{n} ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2$. Polynomial, multi-quadratic and sigmoid kernels were also tested. The best performance was achieved with the sigmoid and Gaussian kernels. For the inverse mapping, $N_n = n$ was used (i.e., all training points). For the GP emulation, a squared exponential covariance function and a zero mean function (after centering the data) is used. The hyperparameters were found using a MLE (gradient descent). Errors in the predictions of the vectorised snapshots $\boldsymbol{y}_j$ were measured using a normalized error: $\epsilon = ||\boldsymbol{y}_j^p - \boldsymbol{y}_j||/||\boldsymbol{y}_j||$, where $\boldsymbol{y}_j^p$ denotes the prediction of the test point $\boldsymbol{y}_j = \boldsymbol{\eta}(\boldsymbol{\xi}_j)$, $j = 1, \ldots, n_t$, using steps 1a-6a of Algorithm 9. Errors in the predictions using kGPE-POD/kGPE-POD-DEIM at $\boldsymbol{\xi}_j$ were measured using a relative error ϵ_r:

$$\epsilon_r = \frac{1}{m} \sum_{i=1}^{m} \frac{||\boldsymbol{u}_i^p(\boldsymbol{\xi}_j) - \boldsymbol{u}_i(\boldsymbol{\xi}_j)||}{||\boldsymbol{u}_i(\boldsymbol{\xi}_j)||} \tag{6.25}$$

where $\boldsymbol{u}_i^p(\boldsymbol{\xi}_j)$ is the prediction (steps 1a-7a in Algorithm 9) of the test point (snapshot) $\boldsymbol{u}_i(\boldsymbol{\xi}_j)$.

One would naturally first examine the normalized errors ϵ in the predictions of the test data points $\boldsymbol{y}_j = \boldsymbol{\eta}(\boldsymbol{\xi}_j)$, $j = 1, \ldots, n_t$. Using $m = 10$ of the snapshots (selecting every 10), Fig. 6.1 shows Tukey box plots of ϵ for the $n_t = 300$ test cases as the manifold dimension q is increased, using $n = 80$ training points. Outliers

are plotted individually using a '+' symbol. It is noted that when predicting the training set in this case using $q = 10$ the maximum value of ϵ was around 10^{-11}, while the median was around 10^{-12}. As a comparison, the result for Isomap (replacing kPCA in Algorithm 9) is also included. The best results were obtained with kPCA, for which the errors converge after $q = 6$ dimensions (negligible further decrease). Diffusion maps were also tested and gave results similar to kPCA. The same pattern was observed at $n = 40$, 120 and 200 training points and also for all values of m up to 100. Based on the results, the approximating manifold dimension was set to $q = 10$ for all values of n and m (using kPCA).

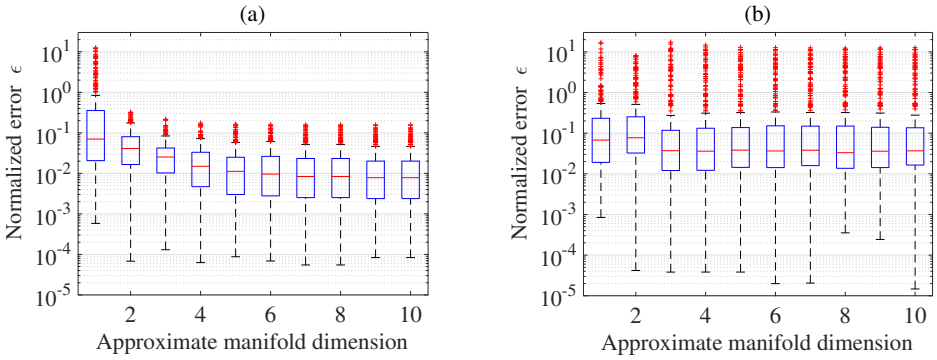

Figure 6.1: Tukey box plots of ϵ with increasing q for the contaminant transport model ($n_t = 300$, $n = 80$ and $m = 10$): (a) kPCA; (b) Isomap.

Fig. 6.2 compares kGPE-POD with a global basis method for increasing POD dimension r. In the global basis method the snapshot matrices comprising the global snapshot matrix corresponded to the $n = 80$ training points used for kGPE-POD. An SVD was performed on the global matrix before extracting the POD basis. For $n = 40$, the results were similar to the results depicted in Fig. 6.2, with a slight decrease in accuracy for both methods. Using $m = 10$ snapshots, the decrease in the relative errors ϵ_r in kGPE-POD is negligible for $r > 15$, while the global basis method continues to improve beyond $r = 50$. In principle, kGPE-POD uses the correct bases for the test parameter values. It is possible, therefore, that kGPE-POD would approach the true result for a smaller value of r compared to the global basis approach, which uses a single basis extracted from snapshots that do

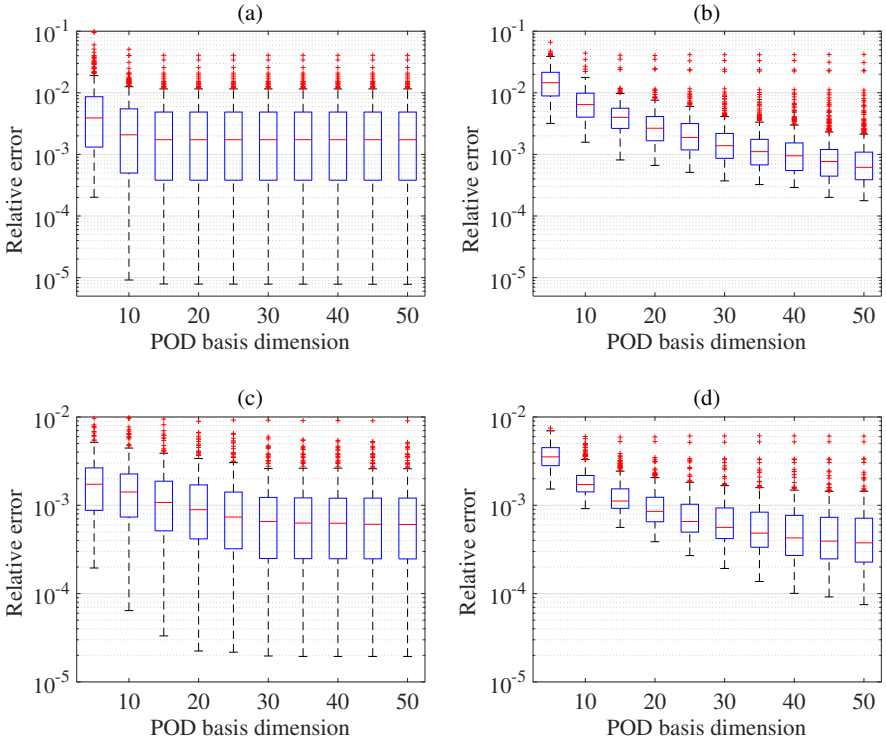

Figure 6.2: Tukey box plots of ϵ_r with increasing r for the contaminant transport model ($n_t = 300$ and $n = 80$). (a) kGPE-POD with $m = 10$; (b) global basis with $m = 10$; (c) kGPE-POD with $m = 100$; (d) global basis with $m = 100$.

not pertain to the test parameter values.

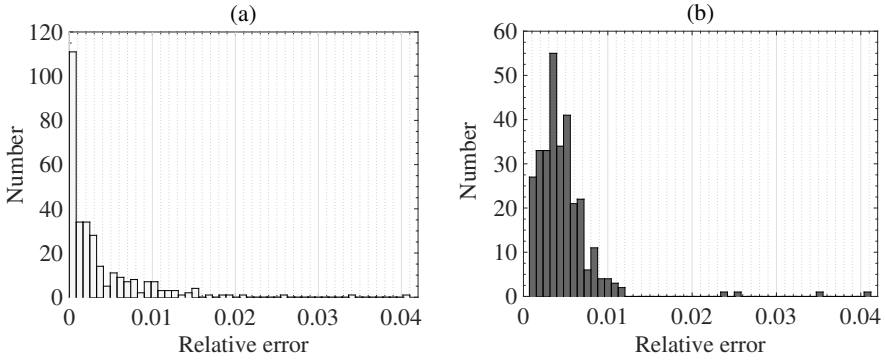

Figure 6.3: Histograms of ϵ_r corresponding to $m = 10$, $r = 15$ in Fig. 6.2, using: (a) kGPE-POD; and (b) a global basis.

For $m = 10$, kGPE-POD exhibits a minimum ϵ_r that is lower by more than an order of magnitude, while the maximum ϵ_r for both methods is roughly the same (0.04 for $r \geq 15$). At $r = 15$ in Figs. 6.2(a) and (b), the value of ϵ_r using kGPE-POD is lower than the minimum ϵ_r in the global basis method in 109 of the 300 test cases. For the global basis at $r = 15$, there are 131 cases with an error below the median (3.9×10^{-3}), while for kGPE-POD, 217 cases have errors below this value. kGPE-POD clearly exhibits a broader range of ϵ_r values, with a higher median for $r > 25$. Fig. 6.3 shows histograms of ϵ_r for the two methods in the case of $r = 15$, $m = 10$. The broader range of ϵ_r is due to the much lower minimum and to the presence of a greater number of cases with $\epsilon_r > 0.012$. The number of such cases (13) is, however, small. For $m = 100$ snapshots, both methods improve, with the global basis method exhibiting the greater improvement (e.g., the maximum ϵ_r is decreased by around an order of magnitude whereas for kGPE-POD the decrease is by a factor of 4 at $r = 15$). The global basis method has a lower median ϵ_r for $r \geq 20$, but also again a considerably higher minimum (more than an order of magnitude at $r = 25$). At $r = 30$, e.g., there are 77 cases in kGPE-POD with a lower ϵ_r than the minimum for the global basis.

To gain an indication of the actual quality of the predictions for different ϵ_r,

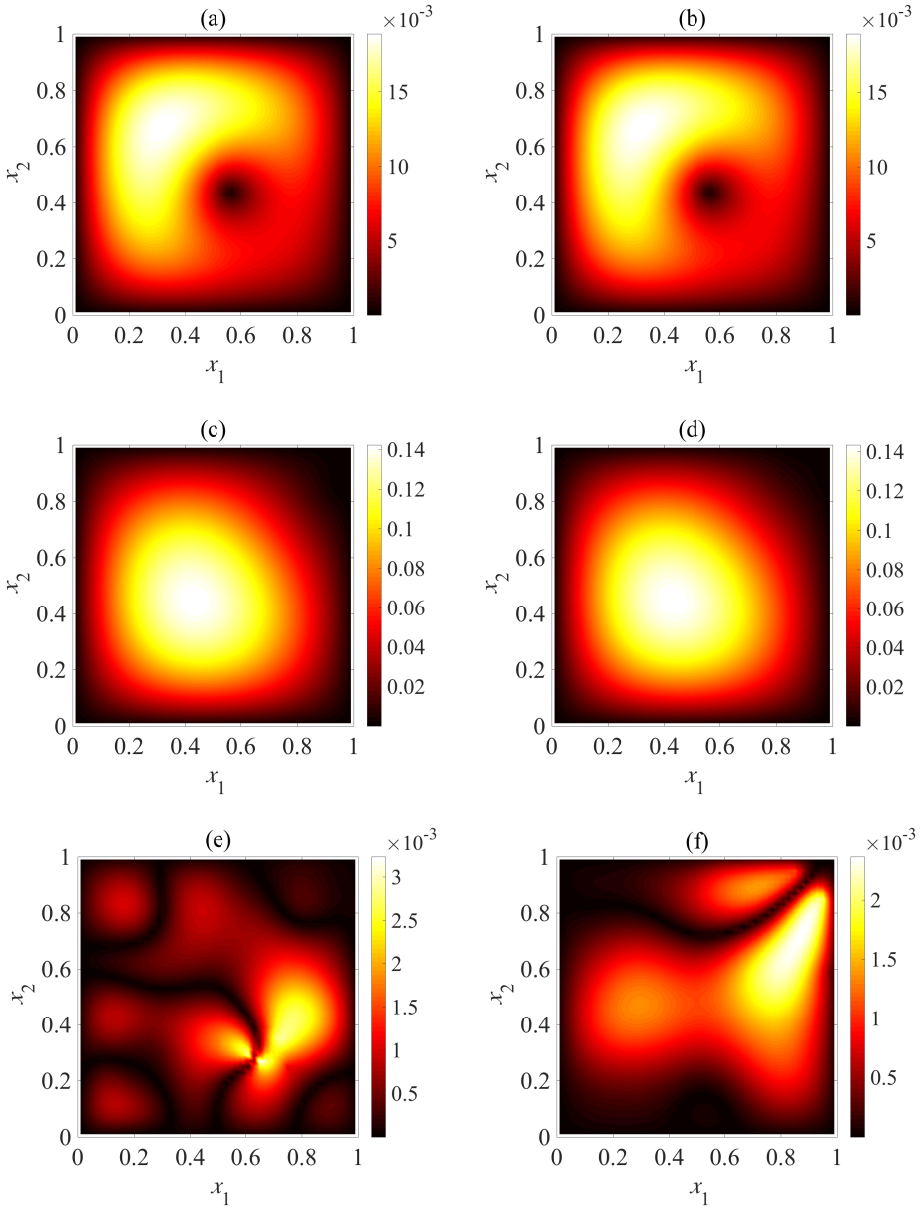

Figure 6.4: (a) The FOM and (b) the kGPE-POD prediction of the concentration field (mol m^{-3}) for the contaminant transport model at $\boldsymbol{\xi} = (0.7382, 0.4179)^T$ and $t = 0.02s$ ($\epsilon_r \approx 0.0021$). (c) The FOM and (d) the kGPE-POD predictions at $\boldsymbol{\xi} = (0.7539, 0.7461)^T$ and $t = 0.2s$ ($\epsilon_r \approx 0.0127$). In all cases $n = 80$, $m = 10$ and $q = 6$. (e) Absolute pointwise error for the case $\boldsymbol{\xi} = (0.7382, 0.4179)^T$ and (f) absolute pointwise error for $\boldsymbol{\xi} = (0.7539, 0.7461)^T$.

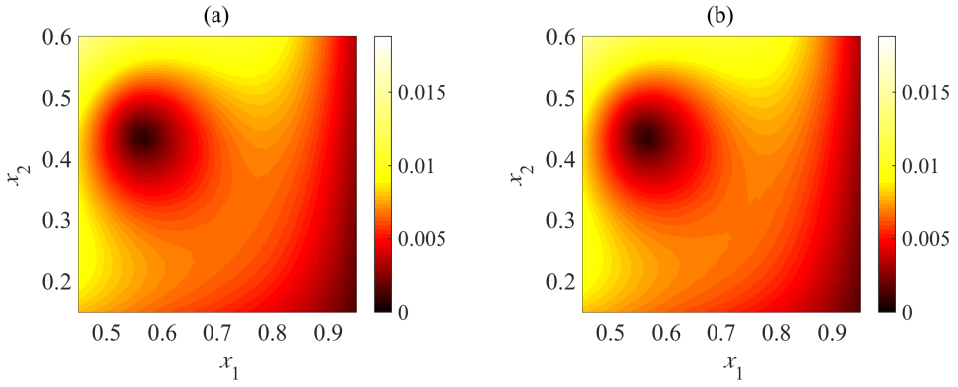

Figure 6.5: A close-up of (a) the kGPE-POD prediction and (b) the test corresponding to Figs. 6.4(a) and (b).

Fig. 6.4 compares the predicted kGPE-POD concentration fields in two test cases: (a) near the median ($\epsilon_r \approx 0.0021$) and near the upper whisker ($\epsilon_r \approx 0.0127$) at $r = 10$ in Fig. 6.2(a). The change in the profiles from one input to the other is well captured. Figs. 6.4(e) and (f) show the absolute pointwise errors for the two examples. It can be seen that there are localized regions of high error. For the first case ($\boldsymbol{\xi} = (0.7382, 0.4179)^T$), a comparison of the region of highest error (lower right quadrant) with the test is shown in Fig. 6.5, which clearly highlights the fine-scale differences leading to the error. The trends and general profile (and in most of the domain the actual concentration values) are nevertheless well captured even with a small value of r.

In order to assess the generalization accuracy more fully, a UQ problem is considered for the accumulated contaminant concentration $\bar{u}(\boldsymbol{x};\boldsymbol{\xi}) := \int_0^T u(\boldsymbol{x}, t;\boldsymbol{\xi})dt$ at the location $\boldsymbol{x}_c = (0.5, 0.5)^T$, by considering $\boldsymbol{\xi}$ to be a random vector distributed according to $p(\boldsymbol{\xi}) = \mathcal{N}(\boldsymbol{\mu}, \sigma^2\mathbf{I})$, where $\boldsymbol{\mu} = (0.5, 0.5)^T$ and $\sigma^2 = 0.1$. The distribution of $\bar{u}(\boldsymbol{x}_c;\boldsymbol{\xi})$ was estimated using Monte Carlo sampling with N_M samples $\boldsymbol{\xi}^i$ (this notation is to avoid confusion with the design points) drawn from $p(\boldsymbol{\xi})$. It is setted as $q = 6$, $n = 80$, $N_M = 3000$, and approximated $\bar{u}(\boldsymbol{x}_c;\boldsymbol{\xi})$ with a trapezoidal rule. Fig. 6.6 compares the histograms obtained from kGPE-POD, the global basis method and the FOM, using $m = 10$ snapshots. The FOM took 55.18

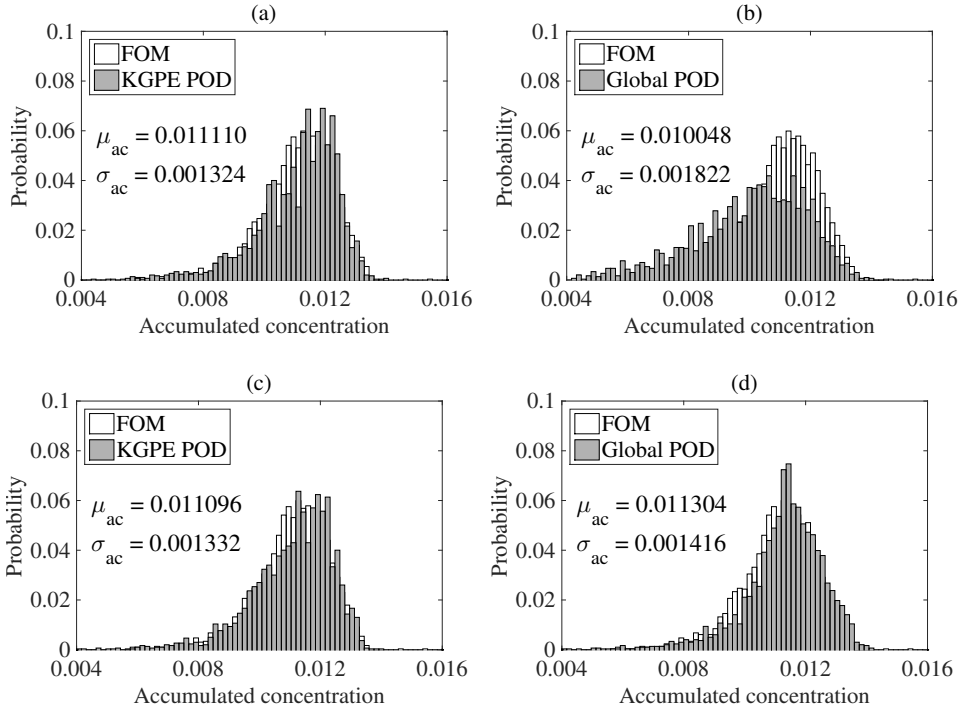

Figure 6.6: Estimated distribution of $\bar{u}(\boldsymbol{x}_c;\boldsymbol{\xi})$ from $N_M = 3000$ MC samples using $n = 80$ and $m = 10$: (a) kGPE-POD with $r = 10$; (b) global basis with $r = 10$; (c) kGPE-POD with $r = 50$; (d) global basis with $r = 50$.

h to complete and yielded $\mu_{ac} = 0.011087$ and $\sigma_{ac} = 0.001218$, obtained from $\mu_{ac} = (1/N_M)\sum_{i=1}^{N_M} \bar{u}(\boldsymbol{x};\boldsymbol{\xi}^i)$ and $\sigma_{ac}^2 = (N_M - 1)^{-1}\sum_{i=1}^{N_M}(\bar{u}(\boldsymbol{x};\boldsymbol{\xi}^i) - \mu_{ac})^2$. For $r = 10$, kGPE-POD exhibited reasonable accuracy with regards to μ_{ac} (within 0.2 %) and σ_{ac} (within 8.7 %), while the global basis method was inaccurate (50 % error in σ_{ac}). For $m = 10$, $r = 50$, both methods were accurate, with kGPE-POD still providing better estimates of μ_{ac} and σ_{ac}. For $m = 100$, the results are shown in Fig. 6.7. kGPE-POD was again more accurate for $r = 10$, while for $r = 30$, the two methods exhibited a similar level accuracy.

6.5.2 Burgers equation

A 1-D Burgers equation is considered here, with inputs $\boldsymbol{\xi}$ to be defined later:

$$\begin{aligned} &\partial_t u + \frac{1}{2}\partial_x(u^2) - \frac{1}{Re}\partial_{xx}u = g(x), \quad x \in \mathcal{D} := (0,1) \\ &u(0,t) = u(1,t) = 0, \quad u(x,0) = u_0(x) := \sin(k\pi x)e^{-(c_1 x + c_2)} \end{aligned} \tag{6.26}$$

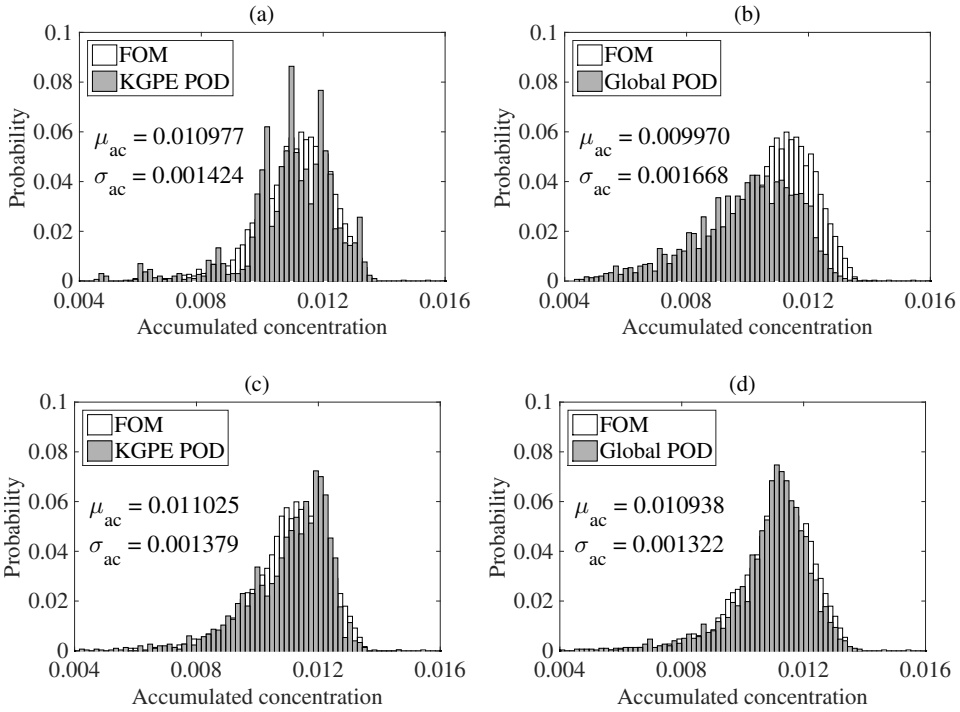

Figure 6.7: Estimated distribution of $\bar{u}(\boldsymbol{x}_c;\boldsymbol{\xi})$ from $N_M = 3000$ MC samples with $n = 80$ and $m = 100$: (a) kGPE-POD with $r = 10$; (b) global basis with $r = 10$; (c) kGPE-POD with $r = 30$; (d) global basis with $r = 30$.

where $u(x,t;\boldsymbol{\xi})$ is the flow velocity, $c_1, c_2 \in \mathbb{R}$, $k \in \mathbb{N}$, Re is the Reynold's number and $g(x)$ is a source term. One seeks a weak solution $u(x,t;\boldsymbol{\xi}) \in \mathcal{V} := H_0^1(\mathcal{D})$ satisfying:

$$(\partial_t u, v) + \frac{1}{2}(\partial_x(u^2), v) + \frac{1}{Re}a(u,v) = (g,v) \quad \forall v \in \mathcal{V} \tag{6.27}$$

where $a(\varphi_1, \varphi_2) := (\varphi_1', \varphi_2')$, $\varphi_1, \varphi_2 \in \mathcal{V}$, defines a bilinear functional, in which a prime denotes an ordinary derivative w.r.t. x. The interval $\overline{\mathcal{D}} = [0,1]$ is partitioned into $N+1$ equally sized subintervals $[x_i, x_{i+1}]$, where $x_i = (i-1)/(N+1)$, $i = 1, \ldots, d = N+2$. A standard piecewise linear basis $\{\psi_i(x)\}_{i=1}^d$ defines the approximating space $\mathcal{V}^h := \text{span}(\psi_1, \ldots, \psi_d) \subset \mathcal{V}$.

The FE approximation $u(x,t;\boldsymbol{\xi}) \approx u^h(x,t;\boldsymbol{\xi}) = \sum_{j=1}^d u_j(t;\boldsymbol{\xi})\psi_j(x)$ leads to the weak formulation: find $u = u^h(x,t;\boldsymbol{\xi}) \in \mathcal{V}^h$ such that (6.27) holds $\forall v = v^h(x) \in \mathcal{V}^h$. It is also convenient to make use of the *group (product) approximation* [149]: $u(x,t;\boldsymbol{\xi})^2 \approx \sum_{j=1}^d u_j(t;\boldsymbol{\xi})^2\psi_j(x) \in \mathcal{V}^h$. Setting $u = u^h$ and $v^h = \psi_j$ in (6.27) one obtains the semi-discrete problem:

$$\sum_{i=1}^d \dot{u}_i(t;\boldsymbol{\xi})(\psi_i, \psi_j) + \frac{1}{2}\sum_{i=1}^d u_i(t;\boldsymbol{\xi})^2(\psi_i', \psi_j) + \frac{1}{Re}\sum_{i=1}^d u_i(t;\boldsymbol{\xi})(\psi_i', \psi_j') = (g, \psi_j) \tag{6.28}$$

together with $\sum_{i=1}^d u_i(0;\boldsymbol{\xi})(\psi_i, \psi_j) = (u_0, \psi_j)$, $\forall j = 1, \ldots, d$. Defining the solution vector $\boldsymbol{u}(t;\boldsymbol{\xi}) = (u_1(t;\boldsymbol{\xi}), \ldots, u_d(t;\boldsymbol{\xi}))^T$, Eq. (6.28) and the initial condition lead to:

$$\mathbf{M}\dot{\boldsymbol{u}}(t;\boldsymbol{\xi}) + \boldsymbol{b}(\boldsymbol{u}(t;\boldsymbol{\xi})) + \frac{1}{Re}\mathbf{S}\boldsymbol{u}(t;\boldsymbol{\xi}) = \mathbf{g}, \quad \mathbf{M}\boldsymbol{u}(0;\boldsymbol{\xi}) = \boldsymbol{u}_0 \tag{6.29}$$

where the (i,j)-th elements of the mass and stiffness matrices $\mathbf{M}$ and $\mathbf{S}$ are given by (ψ_i, ψ_j) and $\left(\psi_i', \psi_j'\right)$, respectively, and the j-th components of $\boldsymbol{u}_0$ and $\mathbf{g}$ are (u_0, ψ_j) and (g, ψ_j), respectively, The nonlinear vector function $\boldsymbol{b}(\boldsymbol{u}(t;\boldsymbol{\xi}))$ arises from the second term in (6.28). a Runge-Kutta method with a variable time step is used to solve the semi-discrete problems in this example.

The coefficients $u_{i,j}(\boldsymbol{\xi})$, $j = 1, \ldots, d$, of the snapshots $\boldsymbol{u}_i(\boldsymbol{\xi}) = \boldsymbol{u}(t_i;\boldsymbol{\xi})$,

$i = 1, \ldots, m$, for an arbitrary value of $\boldsymbol{\xi}$ are the nodal coefficients in the FEM solution, and thus correspond to functions $\mathsf{u}_i(x, \boldsymbol{\xi}) := \sum_{j=1}^{d} u_{i,j}(\boldsymbol{\xi})\psi_j(x) \in \mathcal{V}^h$. For the definition of the POD basis, it is natural to chose the $L^2(\mathcal{D})$ norm for optimality; that is, $\mathcal{H} = L^2(\mathcal{D})$. A FE approximation of the POD basis functions $\{v_j^h(x;\boldsymbol{\xi})\}_{j=1}^d$ is given by $v_j^h(x;\boldsymbol{\xi}) = \sum_{i=1}^{d} v_{j,i}(\boldsymbol{\xi})\psi_i(x) \in \mathcal{V}^h$, $j = 1, \ldots, d$, in which the nodal coefficient $v_{j,i}(\boldsymbol{\xi})$ is the i-th component of the POD basis vector $\boldsymbol{v}_j(\boldsymbol{\xi})$, given by $\boldsymbol{v}_j(\boldsymbol{\xi}) = \mathbf{M}^{-1/2}\overline{\boldsymbol{v}}_j(\boldsymbol{\xi})$, where $\overline{\boldsymbol{v}}_j(\boldsymbol{\xi})$ is an eigenvector of $\mathbf{M}^{1/2}\mathbf{C}(\boldsymbol{\xi})\mathbf{M}^{1/2}$. Note that $L^2(\mathcal{D})$ orthogonality of the basis $\{v_j^h(x;\boldsymbol{\xi})\}_{j=1}^d$ is equivalent to orthogonality of the $\boldsymbol{v}_j(\boldsymbol{\xi})$ w.r.t. $\langle \boldsymbol{v}_j(\boldsymbol{\xi}), \boldsymbol{v}_i(\boldsymbol{\xi})\rangle_{\mathbf{M}} := \boldsymbol{v}_j(\boldsymbol{\xi})^T\mathbf{M}\boldsymbol{v}_i(\boldsymbol{\xi})$. The solution vector is then expanded as in Eq. (6.8): $\boldsymbol{u}(t;\boldsymbol{\xi}) \approx \mathbf{u}(t;\boldsymbol{\xi}) = \sum_{j=1}^{r} a_j(t;\boldsymbol{\xi})\boldsymbol{v}_j(\boldsymbol{\xi}) = \mathbf{V}_r(\boldsymbol{\xi})\boldsymbol{a}(t;\boldsymbol{\xi})$, leading to the reduced order model:

$$\begin{gathered}\dot{\boldsymbol{a}}(t;\boldsymbol{\xi}) + \mathbf{V}_r(\boldsymbol{\xi})^T\boldsymbol{b}\left(\mathbf{V}_r(\boldsymbol{\xi})\boldsymbol{a}(t;\boldsymbol{\xi})\right) + \frac{1}{Re}\mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{S}\mathbf{V}_r(\boldsymbol{\xi})\boldsymbol{a}(t;\boldsymbol{\xi}) = \mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{g} \\ \boldsymbol{a}(0;\boldsymbol{\xi}) = \boldsymbol{a}_0(\boldsymbol{\xi}) := \mathbf{V}_r(\boldsymbol{\xi})^T\boldsymbol{u}_0\end{gathered} \tag{6.30}$$

Another choice for optimality is $\mathcal{H} = H_1^0(\mathcal{D})$ with $a(\cdot,\cdot)$ as the inner product and associated semi-norm $|\varphi|_1 = a(\varphi,\varphi)^{1/2}$. The POD eigenvalue problem $\int_0^T a(u,v)udt = \lambda v$ leads to the eigenvalue problem $\mathbf{C}(\boldsymbol{\xi})^T\mathbf{S}\boldsymbol{v}_j(\boldsymbol{\xi}) = \lambda\boldsymbol{v}_j(\boldsymbol{\xi})$. The POD basis vectors are then given by $\boldsymbol{v}_j(\boldsymbol{\xi}) = \mathbf{S}^{-1/2}\overline{\boldsymbol{v}}_j(\boldsymbol{\xi})$, where $\overline{\boldsymbol{v}}_j(\boldsymbol{\xi})$ is an eigenvector of $\mathbf{S}^{1/2}\mathbf{C}(\boldsymbol{\xi})\mathbf{S}^{1/2}$, and are mutually orthogonal w.r.t. $\langle\cdot,\cdot\rangle_{\mathbf{S}}$. In the present example this approach gave almost identical results.

Case 1. In the first example, $g(x) \equiv 0$ and $k = 1$ are setted. The inputs were defined as $\boldsymbol{\xi} = (c_1, c_2, Re)^T \in \mathcal{X} = [2,5] \times [0.1,1] \times [10,1000]$. A total of 500 inputs $\boldsymbol{\xi}_j \in \mathcal{X}$ were selected using a Sobol sequence and numerical experiments were performed by solving the FOM model (6.29) with $d = 64$ nodes, for each $j = 1, \ldots, 500$, to obtain the solution vectors $\boldsymbol{u}(t_i;\boldsymbol{\xi}_j)$ and nonlinearity $\boldsymbol{b}(\boldsymbol{u}(t_i;\boldsymbol{\xi}_j))$ at times, $t_i = 0.25i$, $i = 1, \ldots, 40$ ($m = 40$). This yielded the data points (vectorized snapshots) $\boldsymbol{y}_j = \boldsymbol{\eta}(\boldsymbol{\xi}_j)$ and $\boldsymbol{y}_j^f = \boldsymbol{\eta}^f(\boldsymbol{\xi}_j)$, $j = 1, \ldots, 500$, according to Eq. (6.18). Of

the 500 data points, $n_t = 300$ were reserved for testing, and training points were selected from the remaining 200 points. The details of kPCA and GP emulation were as described in the previous example.

Analysis of the normalized errors ϵ for the n_t test cases with $n = 160$ training points showed convergence after $q = 8$ dimensions. Isomap gave similar results while Diffusion maps was inferior. $q = 9$ (kPCA) is used in the results presented below. Fig. 6.8(a) shows the results of kGPE-POD-DEIM for an in-

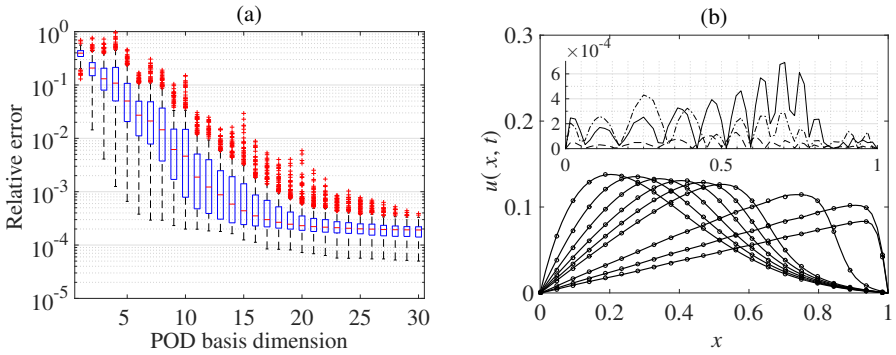

Figure 6.8: (a) Tukey box plots of ϵ_r with increasing r using kGPE-POD-DEIM for Burgers model case 1 ($n = 180$, $n_t = 300$ and $m = 15$). (b) Velocity profiles at $t = 0, 0.5, 1, 1.5, 2, 2.5, 5, 7.5, 10$ s simulated with the FOM (filled circles, every third node) and kGPE-POD-DEIM (solid lines) for a case with $\epsilon_r \approx 0.041$ at $r = 10$. The inset in Figure (b) shows the absolute pointwise error at $t = 2.5$ s (dashed), 5 s (solid) and 10 s (dashed dotted).

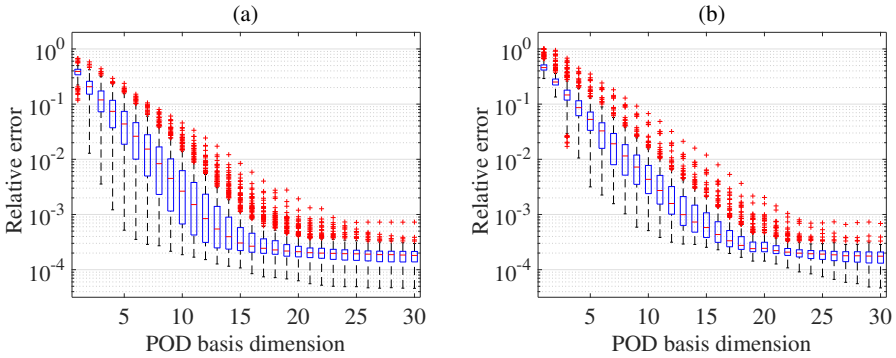

Figure 6.9: Tukey box plots of ϵ_r with increasing r for Burgers model case 1 ($n_t = 300$, $m = 40$ and $n = 180$): (a) kGPE-POD; (b) a global basis.

creasing r (with $s = r$). The relative errors converge after $r = 30$, i.e., further decreases are negligible. Fig. 6.8(b) compares the predicted velocity profiles at

$t = 0, 0.5, 1, 1.5, 2, 2.5, 5, 7.5, 10$ s from kGPE-POD-DEIM and the FOM for a point ($\epsilon_r \approx 0.041$) above the upper whisker at $r = 10$ in Fig. 6.8(a). The two sets of profiles are very close. The inset in Figure (b) shows the absolute pointwise error at $t = 2.5$, 5 and 10 s. Inspection of the full set of profiles showed that the error grew with time until the front developed, after which the error decayed. The highest absolute error was around 8.62×10^{-4} at $x = 0.703$, $t = 5.65s$, for which $u(x,t) \approx 0.103$ m s^{-1}. Thus, the maximum error was around 0.84 %.

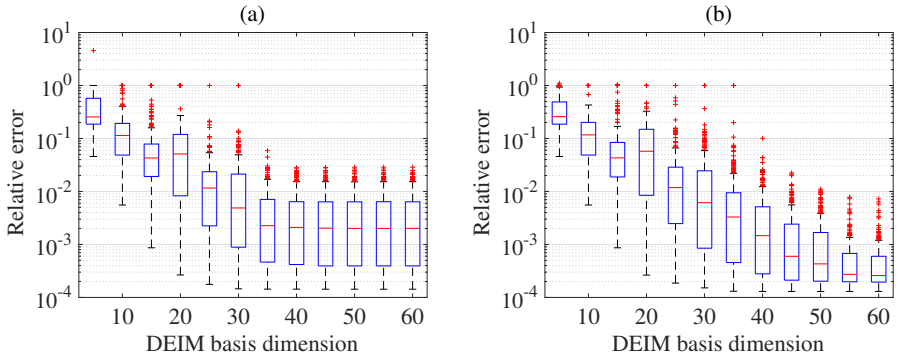

Figure 6.10: Tukey box plots of ϵ_r with increasing s for Burgers model case 2 ($n_t = 300$, $n = 180$ and $m = 200$) using kGPE-POD-DEIM with: (a) $r = 30$; and (b) $r = 50$.

With no approximation of the nonlinearity, a comparison between kGPE-POD and the global basis method exhibited trends similar to those seen in the previous example. For $m < 30$ and $n \leq 200$, kGPE-POD required fewer POD vectors to achieve a given level of accuracy; the lower bound for ϵ_r at $r = 10$ was one order of magnitude smaller for kGPE-POD. Both methods improved with increasing m, with the global basis method showing a greater improvement, especially in the lower bound for ϵ_r. For $m = 30$ and $n = 180$ the results are illustrated in Fig. 6.9, which shows that around $r = 28$ both methods exhibit similar levels of accuracy in terms of the maximum, minimum and median ϵ_r.

Case 2. In a second case, $g(x) = 0.02e^x$, $k = 3$ and $c_2 = 0.2$ are selected with inputs $\boldsymbol{\xi} = (c_1, Re)^T \in \mathcal{X} = [2,5] \times [10, 1000]$. As before, 500 inputs using a Sobol sequence are preselected and the FOM is run to generate data points, with $n_t = 300$

reserved for testing. In this case, $d = 128$ nodes is used and after inspection of the normalized errors ϵ $q = 9$ is setted. In contrast to the previous case, a large m ($m > 120$) was required for accurate results.

Fig. 6.10 shows the trends in the kGPE-POD-DEIM relative error ϵ_r on the $n_t = 300$ test points with increasing s for two values of r, using $n = 180$ and $m = 200$. For a fixed r, the errors decrease with an increasing s. For a fixed s, the errors were seen to decrease as r was increased up to a certain value. For higher values of r the solutions became less stable, with a corresponding increase in the error. This was more pronounced for small values of s. The optimal distribution of errors (in terms of the median, quartiles and extrema) was achieved for values of s between 5 and 10 higher than the value of r. Similar results for Burgers equation can be found in, e.g., [150, 151].

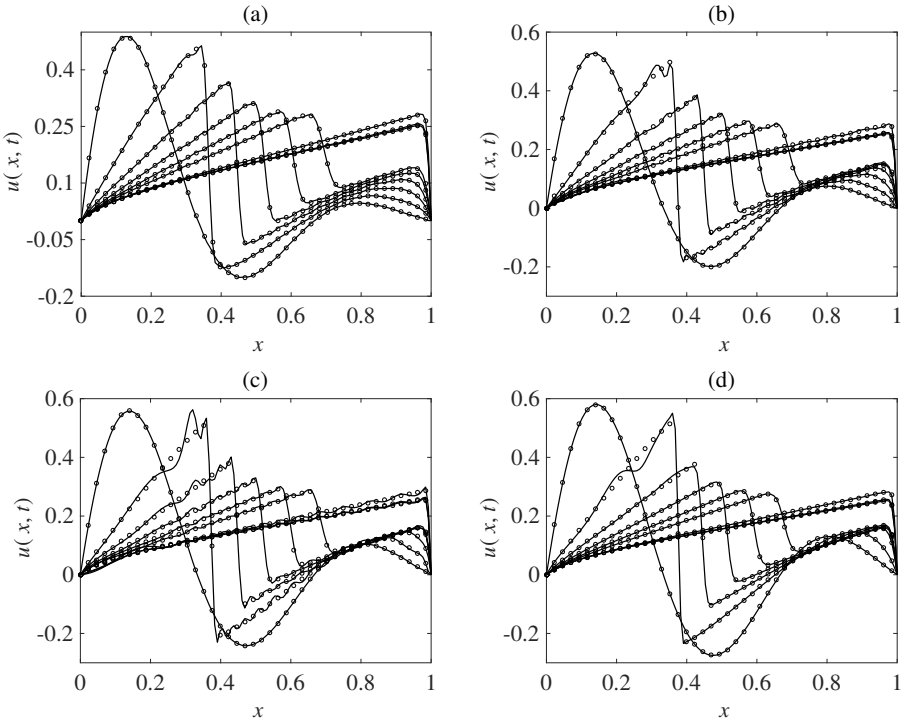

Figure 6.11: Velocity profiles predicted by the FOM (filled circles, every third node) and kGPE-POD-DEIM (solid lines) at $t = 0, 0.5, 1, 1.5, 2, 2.5, 5, 7.5, 10$ s for Burgers model case 2. (a) A point near the median ($\epsilon_r \approx 0.0022$) at $r = 30$, $s = 40$ in Figure 6.10(a); (b) a point near the upper whisker ($\epsilon_r \approx 0.0154$) at $r = 30$, $s = 40$; (c) point with the highest error ($\epsilon_r \approx 0.0282$) at $r = 30$, $s = 40$; (d) point with the highest error ($\epsilon_r \approx 0.0072$) at $r = 50$, $s = 55$ in Figure 6.10(b).

For $r = 30$ and $s = 40$, Figs. 6.11(a) and (b) compare the FOM and kGPE-POD-DEIM profiles at $t = 0, 0.5, 1, 1.5, 2, 2.5, 5, 7.5, 10$ s. The first of these corresponds to a point near the median of the relevant box plot in Fig. 6.10(a), while the second corresponds to a point near the upper whisker. Fig. 6.11(c) shows the point with the highest error using the same values of r and s. In this case, instability develops as the front forms but eventually settles. Using $r = 50$ and $s = 55$, the case with the highest error is shown in Fig. 6.11(d). In Fig. 6.11(d) It could be seen that the solutions at early times are more stable. The observed instability is a common feature of POD models [42, 49, 66]. Stabilization schemes, e.g., alternative inner products, post-processing steps and modification of the underlying model [49, 51, 86] can be incorporated within the framework developed in order to eliminate or minimize such problems.

6.6 Concluding remarks

In this chapter, a new POD method is introduced for building parameterized ROMs across potentially a broad window of parameter space for linear and nonlinear unsteady PDEs. Since the method introduced here is a general framework, a number of modifications could easily be made, e.g., changing the numerical method for the PDE, the emulator and the manifold learning method, according to different types of problems.

The manifold learning based GPE emulator could be treated as a general data-driven machine learning technique to interpolate properties other than the snapshots. For instance, one could employ it to learn the POD basis $\mathbf{V}_r(\boldsymbol{\xi})$ in Eq. (6.8) or the reduced-order system matrix $\mathbf{A}_r(\boldsymbol{\xi})$ in Eq. (6.10), both of which would reduce the computational time. Such approaches were, however, found to be unstable in our numerical experiments. Further investigations are required.

Compared to the FOM, our method shows huge computational saving (388

times faster than the FOM in the first example), while maintaining a high level of accuracy. Compared to a global POD method, our method takes extra computation effort on at each run to diagonalize the snapshot matrix. The actual influence of this is small (as the first example shows in the UQ) since most of the computational time is spent on solving the ROM.

An broad question is whether the direct data-driven approach, e.g., GPE or PCA-GPE, alone is better than kGPE-POD or kGPE-POD-DEIM. Such comparisons are not easy to find in the literature and are not of our focus here. Therefore there are no attempts to make a comparison here either. The best choice is likely to be problem dependent [152]. Researchers are no doubt also influenced by personal preference and the specific techniques with which they are most familiar, or which are most widely accepted in their community. The ROM approaches are inherently linear in the sense that they use a linear subspace of the output space to find approximate solutions. They are also less impressive in terms of computational savings. Data-driven approaches may not, on the other hand, be able to capture the nonlinear dynamical behaviour of some systems, in a way that the original model (or a reduced form) can.

Chapter 7

Conclusions and future work

In this thesis, our aim was to build efficient and accurate surrogate models for parameterized PDE models, where the outputs of interest are spatial and spatio-temporal fields. Applications include sensitivity analysis, optimization and uncertainty quantification, in which the simulator is too costly. A manifold learning based statistical emulation framework for emulating the discrete outputs in high-dimensional spaces is introduced. It can be seen as a direct extension of the method in [1] to cases in which the response surface generated by the data is too complex to be described accurately by a linear subspace. It is worth emphasizing that, although the method developed here may seem unsophisticated enough for real-world application where complicated system with complex PDEs are coupled with complicated boundary conditions, e.g. moving boundaries, and domains, the focus of this thesis is trying to explore and extend existing general surrogate model for complex problems. Considering that modelling a complicated system itself and coupling a surrogate modelling could easily goes beyond the scope of the thesis, such kinds of applications are not demonstrated. This, however, does not hinder the applications to such a problem.

In this thesis, various manifold learning techniques, namely, Isomap, kernel PCA and diffusion maps, and combined these with a number of emulation ap-

proaches for scalar outputs are demonstrated. Pre-image solutions, which map the predictive properties back to the physical space were placed in a general framework. For diffusion maps, a new and efficient method based on a spectral analysis of Markov operators is developed. As the experiments show, our methods show improvements in some cases compared to the use of PCA. The latter method [1] is, however, highly accurate in many cases. In such cases, it is advocated its use, given the relative simplicity and given the access it provides to predictive variances (estimates of the errors in the predictions) without resorting to Monte Carlo approximations. In cases where it fails, however, due to the limitations of PCA, our methods may provide answers.

There are several powerful approaches to manifold learning other than those used in this thesis, including Laplacian eigenmaps, local linear embedding (LLE) [86] and local tangent space alignment (LTSA) [131]. These methods may offer improved accuracy if solutions to the associated pre-image problems can be found. In tests thus far, LLE often suffers form instability within our framework. In contrast, LTSA shows huge advantages in terms of accuracy but is also unstable when using too high a number of dimensions for the reduced space. These issues require further research.

A major limitation is in the accuracy of the pre-image map. Locally linear (or similar) approaches are inherently limited by the distribution of, and distances between points. There has been little progress on this issue for some time and manifold learning methods would benefit greatly from a general method that is stable, accurate and scalable. This issue is considered in our work and would propose exploring several avenues, preferably of a probabilistic nature so that confidence bounds in the predictions of the pre-image are available.

One approach may be Gaussian Process Latent Variable Models (GPLVMs) [153], which are unsupervised methods used primarily for (nonlinear) dimensionality reduction. Our group is currently exploring the application of these methods using variational inference and sparse approaches (inducing variables) combined with

Markov Chain Monte Carlo to perform full Bayesian inference for PDE models (led by Charlie Gadd), based on prior work by Titsias and Lawrence [154, 155]. The framework is also worth exploring for standalone pre-image solutions, given that GPLVMs are generative models that p rovides a joint probability over the high and low dimensional points.

For steady-state problems, our methods are highly accurate and efficient. Solutions to a time dependent system are also discussed in this thesis by treating time as an additional parameter. This is similar to the 'time-input emulator' in [27]. To take the influence of the dynamics into account more fully requires further work, e.g., the iterative dynamic emulator introduced in [156].

The data-driven emulation methods is then implemented to develop a new approach to POD reduced order modelling of linear and nonlinear parameterized PDEs. This includes the extension of DEIM to parametric problems. The method was shown to be more accurate than a global basis approach (for linear cases), while of a similar computational costs. In theory, our method is applicable to a broader class of problems and broader ranges of parameter space.

Chapter 8

Appendix

8.1 Continuous state space diffusion maps

Full details of the following can be found in the references [135, 137, 138]. 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 $\mathcal{M}$ [107, 135, 137, 138], 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 [137, 138]:

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

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 [137, 138]:

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

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 $\mathbb{p}_s(\boldsymbol{y}',\boldsymbol{y}) = \mathsf{k}(\boldsymbol{y},\boldsymbol{y}')/\sqrt{\mathbb{d}(\boldsymbol{y}')}\sqrt{\mathbb{d}(\boldsymbol{y})}$, the following self-adjoint, compact operator $\mathcal{S}$ [137, 138] is obtained:

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

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, the expansion $\mathbb{p}_s(\boldsymbol{y},\boldsymbol{y}') = \sum_{i=1}^{\infty}\gamma_i s_i(\boldsymbol{y})s_i(\boldsymbol{y}')$ is obtained. 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, the expansion $\mathbb{p}(\boldsymbol{y},\boldsymbol{y}') = \sum_{i=1}^{\infty}\gamma_i r_i(\boldsymbol{y})l_i(\boldsymbol{y}')$ [138] is obtained. The t-step transition probabilities $p_t(\boldsymbol{y}^{(i)},\boldsymbol{y}^{(j)})$ defined in Eq. (4.28), 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. (4.28), 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 the general 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 [138]:

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

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, one can define the maps $\boldsymbol{\psi}_r^t : \mathcal{M} \to \mathcal{D}_r^{(t)} \subset \mathbb{R}^r$ as follows:

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

Bibliography

[1] D. Higdon, J. Gattiker, B. Williamsa, and M. Rightley, "Computer model calibration using high-dimensional output," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 570–583, 2008.

[2] M. Kennedy and A. O'Hagan, "Predicting the output from a complex computer code when fast approximations are available," *Biometrika*, vol. 87, pp. 1–13, 2000.

[3] T. Santner, B. Williams, and W. Notz, *The Design and Analysis of Computer Experiments*. Springer, 2003.

[4] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical science*, pp. 409–423, 1989.

[5] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 3, pp. 425–464, 2001.

[6] J. Oakley and A. O'Hagan, "Bayesian inference for the uncertainty distribution of computer model outputs," *Biometrika*, vol. 89, no. 4, pp. 769–784, 2002.

[7] A. Keane and P. Nair, *Computational Approaches for Aerospace Design*. John-Wiley and Sons, 2005.

[8] A. OHagan, "Bayesian analysis of computer code outputs: a tutorial," *Reliability Engineering & System Safety*, vol. 91, no. 10, pp. 1290–1300, 2006.

[9] M. Eldred and D. Dunlavy, "Formulations for surrogate-based optimization with data fit, multifidelity, and reduced-order models," in *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, number AIAA-2006-7117, Portsmouth, VA*, vol. 199, 2006.

[10] C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker, "Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments," *Journal of the American Statistical Association*, vol. 86, no. 416, pp. 953–963, 1991.

[11] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[12] M. Stein, "Large sample properties of simulations using latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.

[13] A. B. Owen, "A central limit theorem for latin hypercube sampling," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 541–551, 1992.

[14] J. C. Helton, "Uncertainty and sensitivity analysis techniques for use in performance assessment for radioactive waste disposal," *Reliability Engineering & System Safety*, vol. 42, no. 2, pp. 327–367, 1993.

[15] R. Aslett, R. J. Buck, S. G. Duvall, J. Sacks, and W. J. Welch, "Circuit optimization via sequential computer experiments: design of an output buffer," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 47, no. 1, pp. 31–48, 1998.

[16] A. Saltelli, K. Chan, E. M. Scott, *et al.*, *Sensitivity analysis*, vol. 1. Wiley New York, 2000.

[17] M. D. Morris, T. J. Mitchell, and D. Ylvisaker, "Bayesian design and analysis of computer experiments: use of derivatives in surface prediction," *Technometrics*, vol. 35, no. 3, pp. 243–255, 1993.

[18] R. L. Iman and W. Conover, "Small sample sensitivity analysis techniques for computer models. with an application to risk assessment," *Communications in statistics-theory and methods*, vol. 9, no. 17, pp. 1749–1842, 1980.

[19] A. O'Hagan, J. Bernardo, J. Berger, A. Dawid, A. Smith, *et al.*, "Uncertainty analysis and other inference tools for complex computer codes," 1998.

[20] C. Bishop, "Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn," 2007.

[21] D. Mackay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, UK, 2003.

[22] J. McFarland, S. Mahadevan, V. Romero, and L. Swiler, "Calibration and uncertainty analysis for computer simulations with multivariate output," *AIAA journal*, vol. 46, no. 5, pp. 1253–1265, 2008.

[23] M. C. Kennedy, C. W. Anderson, S. Conti, and A. OHagan, "Case studies in gaussian process modelling of computer codes," *Reliability Engineering & System Safety*, vol. 91, no. 10, pp. 1301–1309, 2006.

[24] J. Rougier, D. M. Sexton, J. M. Murphy, and D. Stainforth, "Analyzing the climate sensitivity of the hadsm3 climate model using ensembles from different but related experiments," *Journal of Climate*, vol. 22, no. 13, pp. 3540–3557, 2009.

[25] P. M. Tagade, B.-M. Jeong, and H.-L. Choi, "A gaussian process emulator approach for rapid contaminant characterization with an integrated multizone-cfd model," *Building and Environment*, vol. 70, pp. 232–244, 2013.

[26] L. Lee, K. Pringle, C. Reddington, G. Mann, P. Stier, D. Spracklen, J. Pierce, and K. Carslaw, "The magnitude and causes of uncertainty in global model simulations of cloud condensation nuclei," *Atmos. Chem. Phys*, vol. 13, pp. 8879–8914, 2013.

[27] S. Conti and A. OHagan, "Bayesian emulation of complex multi-output and dynamic computer models," *Journal of statistical planning and inference*, vol. 140, no. 3, pp. 640–651, 2010.

[28] H. Wackernagel, *Multivariate geostatistics*. Springer Science & Business Media, 2003.

[29] T. E. Fricker, J. E. Oakley, and N. M. Urban, "Multivariate gaussian process emulators with nonseparable covariance structures," *Technometrics*, vol. 55, no. 1, pp. 47–56, 2013.

[30] J. Rougier, "Efficient emulators for multivariate deterministic functions," *Journal of Computational and Graphical Statistics*, vol. 17, no. 4, pp. 827–843, 2008.

[31] B. Konomi, G. Karagiannis, A. Sarkar, X. Sun, and G. Lin, "Bayesian treed multivariate gaussian process with adaptive design: application to a carbon capture unit," *Technometrics*, vol. 56, no. 2, pp. 145–158, 2014.

[32] A. Gelfand, A. Schmidt, S.Banerjee, and C. Sirmans, "Nonstationary multivariate process modeling through spatially varying coregionalization," *TEST*, vol. 13, no. 2, pp. 263–312, 2004.

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

[34] B. C. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *Automatic Control, IEEE Transactions on*, vol. 26, no. 1, pp. 17–32, 1981.

[35] P. Feldmann and R. W. Freund, "Efficient linear circuit analysis by padé approximation via the lanczos process," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 14, no. 5, pp. 639–649, 1995.

[36] L. Sirovich, "Turbulence and the dynamics of coherent structures. part i: Coherent structures," *Quarterly of applied mathematics*, vol. 45, no. 3, pp. 561–571, 1987.

[37] G. Berkooz, P. Holmes, and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," *Annual review of fluid mechanics*, vol. 25, no. 1, pp. 539–575, 1993.

[38] P. Benner, S. Gugercin, and K. Willcox, "A survey of projection-based model reduction methods for parametric dynamical systems," *SIAM Review*, vol. 57, no. 4, pp. 483–531, 2015.

[39] T. Bui-Thanh, K. Willcox, and O. Ghattas, "Model reduction for large-scale systems with high-dimensional parametric input space," *SIAM Journal on Scientific Computing*, vol. 30, pp. 3270–3288, 2008.

[40] M. A. Grepl, Maday, Yvon, N. C. Nguyen, and A. T. Patera, "Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations," *ESAIM: M2AN*, vol. 41, no. 3, pp. 575–605, 2007.

[41] M. A. Grepl and A. T. Patera, "A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations," *ESAIM: M2AN*, vol. 39, no. 1, pp. 157–181, 2005.

[42] T. Bui-Thanh, K. Willcox, and O. Ghattas, "Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications," *AIAA Journal*, vol. 46, no. 10, pp. 2520–2529, 2008.

[43] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera, "An "empirical interpolation" method," *C. R. Acad. Sci. Paris Ser. I Math*, vol. 339, pp. 667–672, 2004.

[44] J. A. Taylor, "Dynamics of large scale structures in turbulent shear layers," *Dept. of Mechanical & Aeronautical Engineering, Clarkson University, NY, Rept. MAE-354*, 2001.

[45] J. Taylor and M. Glauser, "Towards practical flow sensing and control via pod and lse based low-dimensional tools," *Journal of Fluids Engineering*, vol. 126, no. 3, pp. 337–345, 2004.

[46] J. Degroote, J. Vierendeels, and K. Willcox, "Interpolation among reduced-order matrices to obtain parameterized models for design, optimization and probabilistic analysis," *International Journal for Numerical Methods in Fluids*, vol. 63, no. 2, pp. 207–230, 2010.

[47] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk, "Convergence rates for greedy algorithms in reduced basis methods," *SIAM Journal on Mathematical Analysis*, vol. 43, no. 3, pp. 1457–1472, 2011.

[48] T. Lieu, C. Farhat, and M. Lesoinne, "Reduced-order fluid/structure modeling of a complete aircraft configuration," *Computer methods in applied mechanics and engineering*, vol. 195, no. 41, pp. 5730–5742, 2006.

[49] D. Amsallem and C. Farhat, "Stabilization of projection-based reduced-order models," *International Journal for Numerical Methods in Engineering*, vol. 91, no. 4, pp. 358–377, 2012.

[50] D. Amsallem and C. Farhat, "An online method for interpolating linear parametric reduced-order models," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2169–2198, 2011.

[51] D. Amsallem, J. Cortial, K. Carlberg, and C. Farhat, "A method for interpolating on manifolds structural dynamics reduced-order models," *International Journal for Numerical Methods in Engineering*, vol. 80, no. 9, pp. 1241–1258, 2009.

[52] H. Panzer, J. Mohring, and R. Eid, "Parametric model order reduction by matrix interpolation," *at-Automatisierungstechnik*, vol. 58, no. 8, pp. 475–484, 2010.

[53] M. Geuss, H. Panzer, and B. Lohmann, "On parametric model order reduction by matrix interpolation," in *Control Conference (ECC), 2013 European*, pp. 3433–3438, July 2013.

[54] C. Lieberman, K. Willcox, and O. Ghattas, "Parameter and state model reduction for large-scale statistical inverse problems," *SIAM Journal on Scientific Computing*, vol. 32, no. 5, pp. 2523–2542, 2010.

[55] C. Himpe and M. Ohlberger, "Data-driven combined state and parameter reduction for inverse problems," *Advances in Computational Mathematics*, vol. 41, no. 5, pp. 1343–1364, 2015.

[56] A. Hay, J. T. Borggaard, and D. Pelletier, "Local improvements to reduced-order models using sensitivity analysis of the proper orthogonal decomposition," *Journal of Fluid Mechanics*, vol. 629, pp. 41–72, 2009.

[57] A. Hay, I. Akhtar, and J. T. Borggaard, "On the use of sensitivity analysis in model reduction to predict flows for varying inflow conditions," *International Journal for Numerical Methods in Fluids*, vol. 68, no. 1, pp. 122–134, 2012.

[58] Y. Chen, *Model order reduction for nonlinear systems.* PhD thesis, Massachusetts Institute of Technology, 1999.

[59] Z. Bai, "Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems," *Applied numerical mathematics*, vol. 43, no. 1, pp. 9–44, 2002.

[60] J. R. Phillips, "Projection-based approaches for model reduction of weakly nonlinear, time-varying systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, pp. 171–187, Feb 2003.

[61] P. Astrid, *Reduction of Process Simulation Models.* PhD thesis, Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands, 2004.

[62] P. Astrid, S. Weiland, K. Willcox, and T. Backx, "Missing point estimation in models described by proper orthogonal decomposition," *IEEE Transactions on Automatic Control*, vol. 53, no. 10, pp. 2237–2251, 2008.

[63] S. Chaturantabut and D. C. Sorensen, "Nonlinear model reduction via discrete empirical interpolation," *SIAM Journal on Scientific Computing*, vol. 32, no. 5, pp. 2737–2764, 2010.

[64] S. Chaturantabut and D. C. Sorensen, "A state space error estimate for pod-deim nonlinear model reduction," *SIAM Journal on Numerical Analysis*, vol. 50, no. 1, pp. 46–63, 2012.

[65] B. Peherstorfer, D. Butnaru, K. Willcox, and H.-J. Bungartz, "Localized dis-

crete empirical interpolation method," *SIAM Journal on Scientific Computing*, vol. 36, no. 1, pp. A168–A192, 2014.

[66] K. Carlberg, C. Farhat, J. Cortial, and D. Amsallem, "The {GNAT} method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows," *Journal of Computational Physics*, vol. 242, pp. 623 – 647, 2013.

[67] L. S. Bastos and A. OHagan, "Diagnostics for gaussian process emulators," *Technometrics*, vol. 51, no. 4, pp. 425–438, 2009.

[68] L. Deng, D. Yu, *et al.*, "Deep learning: methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[69] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[70] F. D. Foresee and M. T. Hagan, "Gauss–Newton approximation to Bayesian learning," in *International Conference on Neural Networks, 1997*, vol. 3, pp. 1930–1935 vol.3, IEEE, Jun 1997.

[71] D. J. MacKay, "Bayesian interpolation," *Neural computation*, vol. 4, no. 3, pp. 415–447, 1992.

[72] J. Moody, *Prediction Risk and Architecture Selection for Neural Networks*, pp. 147–165. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994.

[73] V. Vapnik and A. Lerner, "Generalized portrait method for pattern recognition," *Automation and Remote Control*, vol. 24, no. 6, pp. 774–780, 1963.

[74] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152, ACM, ACM Press, 1992.

[75] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[76] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[77] H. W. Kuhn, "Nonlinear programming: a historical view," in *Traces and Emergence of Nonlinear Programming*, pp. 393–414, Springer, 2014.

[78] B. Schölkopf and A. Smola, "Support vector machines," *Encyclopedia of Biostatistics*, 1998.

[79] I. Jolliffe, *Principal component analysis*. Springer Series in Statistics, Wiley Online Library, 2005.

[80] W. S. Torgerson, "Multidimensional scaling: I. theory and method," *Psychometrika*, vol. 17, no. 4, pp. 401–419, 1952.

[81] I. Borg and P. J. Groenen, *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[82] T. F. Cox and M. A. Cox, *Multidimensional scaling*. CRC Press, 2000.

[83] T. Lin and H. Zha, "Riemannian manifold learning," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, pp. 796–809, May 2008.

[84] P. E. LJP and H. H. Van Den, "Dimensionality reduction: A comparative review," *Tech. Rrep*, 2007.

[85] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, 1998.

[86] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[87] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering.," in *NIPS*, vol. 14, pp. 585–591, 2001.

[88] D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[89] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[90] V. N. Vapnik and V. Vapnik, *Statistical learning theory*, vol. 1. Wiley New York, 1998.

[91] L. I. R. A. Aizerman, E. M. Braverman, "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.

[92] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[93] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[94] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[95] M. Balasubramanian and E. L. Schwartz, "The isomap algorithm and topological stability," *Science*, vol. 295, no. 5552, pp. 7–7, 2002.

[96] B. Fischl, M. I. Sereno, and A. M. Dale, "Cortical surface-based analysis: Ii: inflation, flattening, and a surface-based coordinate system," *Neuroimage*, vol. 9, no. 2, pp. 195–207, 1999.

[97] B. A. Wandell, S. Chial, and B. T. Backus, "Visualization and measurement of the cortical surface," *Journal of cognitive neuroscience*, vol. 12, no. 5, pp. 739–752, 2000.

[98] E. Wolfson and E. L. Schwartz, "Computing minimal distances on polyhedral surfaces," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, no. 9, pp. 1001–1005, 1989.

[99] O. C. Jenkins and M. J. Matarić, "A spatio-temporal extension to isomap nonlinear dimension reduction," in *Proceedings of the twenty-first international conference on Machine learning*, p. 56, ACM, 2004.

[100] H. Choi and S. Choi, "Kernel isomap," *Electronics letters*, vol. 40, no. 25, pp. 1612–1613, 2004.

[101] A. Saxena, A. Gupta, and A. Mukerjee, "Non-linear dimensionality reduction by locally linear isomaps," in *Neural Information Processing*, pp. 1038–1043, Springer, 2004.

[102] J. A. Lee and M. Verleysen, "Nonlinear dimensionality reduction of data manifolds with essential loops," *Neurocomputing*, vol. 67, pp. 29–53, 2005.

[103] B. Raytchev, I. Yoda, and K. Sakaue, "Head pose estimation by nonlinear manifold learning," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 4, pp. 462–466, IEEE, 2004.

[104] I. S. Lim, P. de Heras Ciechomski, S. Sarni, and D. Thalmann, "Planar arrangement of high-dimensional biomedical data sets by isomap coordinates," in *Computer-Based Medical Systems, 2003. Proceedings. 16th IEEE Symposium*, pp. 50–55, IEEE, 2003.

[105] M. Niskanen and O. Silvén, "Comparison of dimensionality reduction methods for wood surface inspection," in *Quality Control by Artificial Vision*, pp. 178–188, International Society for Optics and Photonics, 2003.

[106] F. R. Chung, *Spectral graph theory*, vol. 92. American Mathematical Soc., 1997.

[107] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied and computational harmonic analysis*, vol. 21, no. 1, pp. 5–30, 2006.

[108] J.-Y. Kwok and I.-H. Tsang, "The pre-image problem in kernel methods," *IEEE transactions on neural networks*, vol. 15, no. 6, pp. 1517–1525, 2004.

[109] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel pca and de-noising in feature spaces.," in *NIPS*, vol. 11, pp. 536–542, 1998.

[110] Y. Rathi, S. Dambreville, and A. Tannenbaum, "Statistical shape analysis using kernel pca," in *Electronic Imaging 2006*, vol. 6064, pp. 60641B–60641B, International Society for Optics and Photonics, 2006.

[111] C. K. Williams, "On a connection between kernel pca and metric multidimensional scaling," *Machine Learning*, vol. 46, no. 1-3, pp. 11–19, 2002.

[112] X. Ma and N. Zabaras, "Kernel principal component analysis for stochastic input model generation," *J. Comput. Phys.*, vol. 230, no. 19, p. 73117331, 2011.

[113] P. Etyngier, F. Ségonne, and R. Keriven, "Shape priors using manifold learning techniques," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8, IEEE, 2007.

[114] N. Thorstensen, F. Segonne, and R. Keriven, "Karcher means for shape and image denoising," in *International Conference on Scale Space (SSVM), Norway*, 2009.

[115] M. Cavazzuti, *Optimization Methods: From Theory to Design Scientific and Technological Aspects in Mechanics*. Springer Science & Business Media, 2012.

[116] A. Newman, "Model reduction via the karhunen?loeve expansion Part i: An

exposition.," *Technical Report T.R.96-32, University of Maryland, College Park, MD.*, 1996.

[117] E. Wong, *Stochastic Processes in Information and Dynamical Systems*. McGraw-Hill, 1971.

[118] W. Xing, A. A. Shah, and P. B. Nair, "Reduced dimensional gaussian process emulators of parametrized partial differential equations based on isomap," vol. 471, no. 2174, p. 20140697, 2015.

[119] L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, and D. D. Lee, "Spectral methods for dimensionality reduction," *Semisupervised learning*, pp. 293–308, 2006.

[120] L. J. van der Maaten, E. O. Postma, and H. J. van den Herik, "Dimensionality reduction: A comparative review," *Journal of Machine Learning Research*, vol. 10, no. 1-41, pp. 66–71, 2009.

[121] V. D. Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Advances in neural information processing systems*, pp. 705–712, 2002.

[122] C. K. Williams, "On a connection between kernel pca and metric multidimensional scaling," in *Advances in Neural Information Processing Systems 13*, pp. 675–681, MIT Press, 2001.

[123] J. Ham, D. Lee, S. Mika, and B. Schölkopf, "A kernel view of the dimensionality reduction of manifolds," in *Proceedings of the Twenty-first International Conference on Machine Learning, Banff, Canada*, pp. 369–376, ACM, 2004.

[124] F. Cailliez, "The analytical solution of the additive constant problem," *Psychometrika*, vol. 48, no. 2, pp. 305–308, 1983.

[125] B. Ganapathysubramanian and N. Zabaras, "A non-linear dimension reduction methodology for generating data-driven stochastic input models," *Journal of Computational Physics*, vol. 227, no. 13, pp. 6612–6637, 2008.

[126] I. M. Sobol, "Uniformly distributed sequences with an additional uniform property," *USSR Computational Mathematics and Mathematical Physics*, vol. 16, no. 5, pp. 236–242, 1976.

[127] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola, "Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index," *Computer Physics Communications*, vol. 181, pp. 259–270, 2010.

[128] M. D. Morris and T. J. Mitchell, "Exploratory designs for computational experiments," *Journal of statistical planning and inference*, vol. 43, no. 3, pp. 381–402, 1995.

[129] M. Hossain and M. Wilson, "Natural convection flow in a fluid-saturated porous medium enclosed by non-isothermal walls with heat generation," *Int. J. Therm. Sci.*, vol. 41, pp. 447–454, 2002.

[130] F. Wolff and R. Viskanta, "Solidification of a pure metal at a vertical wall in the presence of liquid superheat," *Int. J. Heat Mass Transfer*, vol. 31, pp. 1735–1744, 1988.

[131] Z. y. Zhang and H. y. Zha, "Principal manifolds and nonlinear dimensionality reduction via tangent space alignment," *Journal of Shanghai University (English Edition)*, vol. 8, no. 4, pp. 406–424, 2004.

[132] W. Xing, V. Triantafyllidis, A. Shah, P. Nair, and N. Zabaras, "Manifold learning for the emulation of spatial fields from computational models," *Journal of Computational Physics*, vol. 326, pp. 666 – 690, 2016.

[133] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Artificial Neural NetworksICANN'97*, pp. 583–588, Springer, 1997.

[134] D. Duvenaud, *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[135] R. R. Coifman, S. Lafon, a. B. Lee, M. Maggioni, B. Nadler, F. Warner, and 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*, vol. 102, pp. 7426–31, May 2005.

[136] R. Bellman, *Introduction to matrix analysis*. SIAM, 1997.

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

[138] B. Nadler, S. Lafon, R. Coifman, and I. Kevrekidis, "Diffusion maps, spectral clustering and reaction coordinates of dynamical systems," *Appl. Comput. Harmon. Anal.*, vol. 21, no. 1, pp. 113 – 127, 2006.

[139] D. Kushnir, A. Haddad, and R. R. Coifman, "Anisotropic diffusion on submanifolds with application to earth structure classification," *Applied and Computational Harmonic Analysis*, vol. 32, no. 2, pp. 280 – 294, 2012.

[140] R. Talmon and R. R. Coifman, "Empirical intrinsic geometry for nonlinear modeling and time series filtering," *Proceedings of the National Academy of Sciences*, vol. 110, no. 31, pp. 12535–12540, 2013.

[141] P. Arias, G. Randall, and G. Sapiro, "Connecting the out-of-sample and pre-

image problems in kernel methods," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2007.

[142] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[143] B. Seibold, "A compact and fast Matlab code solving the incompressible Navier-Stokes equations on rectangular domains, last accessed 29 January 2016," 2008.

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

[145] K. Broka and P. Ekdunge, "Modelling the pem fuel cell cathode," *Journal of Applied Electrochemistry*, vol. 27, no. 3, pp. 281–289, 1997.

[146] R. B. Bird, "Transport phenomena," *Applied Mechanics Reviews*, vol. 55, no. 1, pp. R1–R4, 2002.

[147] "COMSOL Multiphysics 5.0, Species Transport in the Gas Diffusion Layers of a PEM, last accessed 29 January 2016,"

[148] V. Triantafyllidis, W. Xing, A. Shah, and P. Nair, "Neural network emulation of spatio-temporal data using linear and nonlinear dimensionality reduction," in *Advanced Computer and Communication Engineering Technology*, pp. 1015–1029, Springer, 2016.

[149] I. Christie, D. F. Griffiths, A. R. Mitchell, and J. M. Sanz-Serna, "Product approximation for non-linear problems in the finite element method," *IMA Journal of Numerical Analysis*, vol. 1, no. 3, pp. 253–266, 1981.

[150] M. Drohmann, B. Haasdonk, and M. Ohlberger, "Reduced basis approximation for nonlinear parametrized evolution equations based on empirical op-

erator interpolation," *SIAM Journal on Scientific Computing*, vol. 34, no. 2, pp. A937–A969, 2012.

[151] M. M. Baumann, "Nonlinear model order reduction using pod/deim for optimal control of burgers equation," *Delft University of Technology, Netherlands*, 2013.

[152] S. Walton, O. Hassan, and K. Morgan, "Reduced order modelling for unsteady fluid flow using proper orthogonal decomposition and radial basis functions," *Applied Mathematical Modelling*, vol. 37, no. 20, pp. 8930–8945, 2013.

[153] N. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," *Advances in neural information processing systems*, vol. 16, no. 3, pp. 329–336, 2004.

[154] M. Titsias and N. Lawrence, "Bayesian gaussian process latent variable model.," in *AISTATS*, pp. 844–851, 2010.

[155] A. C. Damianou, M. K. Titsias, and N. D. Lawrence, "Variational inference for latent variables and uncertain inputs in gaussian processes," *J. Mach. Learn. Res.*, vol. 17, pp. 1425–1486, Jan. 2016.

[156] S. Conti, J. Gosling, J. Oakley, and A. O'hagan, "Gaussian process emulation of dynamic computer codes," *Biometrika*, p. ASP028, 2009.