High-dimensional-output surrogate models for uncertainty and sensitivity analyses

by

Vasileios Triantafyllidis

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

School of Engineering

February 2018

THE UNIVERSITY OF WARWICK

Contents

Acknowledgments

First and foremost I would like to offer my gratitude to my supervisor, Dr Akeel Shah, who supported me all these years during my Ph.D. He has supported me not only academically but emotionally through the rough road to finish this thesis. He helped me to understand about predictive modelling from the very basics to the very advanced level. He is more than a supervisor and I have been extremely lucky to have him to talk about everything concerning me during this period.

I would like to thank my colleague and very good friend, Wei Xing who was always available to give his insights and helped me a lot with the discussions we had about my research and the advices he provided me with.

Moreover, I want to thank my parents, Christos and Athina for their support not only the last four years but also for the whole duration of my studies. My brother, George, for being supportive and keeping me motivated. They were always there to listen my concerns and to provide me with the stability I needed.

I have also to thank all the friends I made during this long period being a Ph.D. student. Miltos, Karolos, Christoforos, Mike, Giannis just to mention a few of them because I will forget some of them which is not fair.

Declarations

This thesis is based on the following refereed publications:

1. V Triantafyllidis, W Xing, AA Shah, PB Nair, *Neural Network Emulation of Spatio-temporal Data Using Linear and Nonlinear Dimensionality Reduction* in Advanced Computer and Communication Engineering Technology pp 1015-1029, Lecture Notes in Electrical Engineering, 2016

2. A. A. Shah, W. W. Xing, V. Triantafyllidis, *Reduced-order modelling of parameter-dependent, linear and nonlinear dynamic partial differential equation models*, Volume 473, issue 2200 in Proceedings of the Royal Society A, 2017.

3. W. W. Xing, V. Triantafyllidis, A.A. Shah, P.B. Nair, 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 and A.A. Shah, *A probabilistic global sensitivity analysis framework using multivariate Gaussian process emulators with applications to lithium-ion batteries*, IOP Conference Series, accepted

Contributions to the papers

The ANN coding with the Bayesian regularisation was done by me, by adapting the manifold learning techniques from W. Xing's GPE code. The first draft of the paper was written by me and revised before submitted by my supervisor who also provided the main idea.

For the second paper my supervisor provided the idea and W. Xing coded the emulatior. I worked with W. Xing on the numerical methods proposed in this paper (finite volume by me and finite difference by W. Xing) before deciding to use my code over that of W. Xing, although both gave similarly accurate results. The first draft was written by me (POD and Galerkin section) and W. Xing (manifold learning). The final version of the paper was written by my supervisor.

In the third paper, the coding for the ANN was done by me while for the GP emulation W. Xing did the coding. For the fourth paper the idea to use a probabilistic framework with sensitivity analysis was my supervisor's. The coding for the the sensitivity analysis was done entirely by me, as well as generation of all results. I also wrote the first draft of the paper, which was finalised by my supervisor.

Abstract

Computational models that describe complex physical phenomena tend to be computationally expensive and time consuming. Partial differential equation (PDE) based models in particular produce spatio-temporal data sets in high dimensional output spaces. Repeated calls of computer models to perform tasks such as sensitivity analysis, uncertainty quantification and design optimization can become computationally infeasible as a result. While constructing an emulator is one solution to approximate the outcome of expensive computer models, it is not always capable of dealing with high-dimensional data sets. To deal with high-dimensional data, in this thesis emulation strategies (Gaussian processes (GPs), artificial neural networks (ANNs) and support vector machines (SVMs)) are combined with linear and non-linear dimensionality reduction techniques (kPCA, Isomap and diffusion maps) to develop efficient emulators. For sensitivity analysis (variance based), a probabilistic framework is developed to account for the emulator uncertainty and the method is extended to multivariate outputs, with a derivation of new semi-analytical results for performing rapid sensitivity analysis of univariate or multivariate outputs. The developed emulators are also used to extend reduced order models (ROMs) based on proper orthogonal decomposition to parameter-dependent PDEs, including an extension of the discrete empirical interpolation method for non-linear problems PDE systems.

Chapter 1

Introduction and overview of methodologies

In this chapter the problems under consideration in this thesis are motivated and introduced, and a review of the relevant literature is provided. An overview of the methodology then follows. Since the methods and techniques are diverse and numerous, full details of the methods used are provided in the relevant chapters. In the overview below, details of the chapter and section in which each method is used are given. At the end of this chapter, the novelty and contributions of this thesis are made clear and the remainder of the thesis is summarised.

Modelling is important in many scientific and engineering fields, as it can aid experiments to lower costs and timescales and/or it can provide fundamental insights when experimental observations are not possible, e.g., *ab-initio* methods and distributions of quantities such as an electric potential inside a device. Mathematical models of physical problems usually have the form of ordinary or partial differential equations with boundary and/or initial conditions. The models can be solved on a computer by using numerical methods such as finite difference, finite volume and finite element, to provide a discretised system. The computer model is also known as a *simulator* or high fidelity model.

Although a high fidelity model provides (in theory) accurate insights into the system under examination, it is often too computationally expensive for studies such as *uncertainty* and *sensitivity* analysis, *real time control* and *optimisation*, i.e., when thousands of repeated calls are necessary for different parameter/input choices. In sensitivity and uncertainty analysis, e.g., Monte Carlo sampling is used to extract statistics of an output from the model, i.e., the simulator is run a high number of times and mean values, variances and so on are estimated from the outputs. If each simulator run takes 10 minutes, 70 days are required for 1000 runs, which is at the lower end for extracting reliable Monte Carlo estimates (usually 10000 runs are recommended). Thus, even moderately expensive simulators can be too expensive and alternatives methods are required, such as fast mathematical *approximations* of the simulator called *surrogate models, emulators* or *metamodels*.

An emulator is an approximation of a high-fidelity computational model (simulator) that can be evaluated very cheaply [1, 2, 3, 4, 5]. There are roughly three different categories of emulators [6]. The simplest approach (*hierarchical*) to create an emulator is by simplifying the physical equations governing a system or the numerical solution. Another (*data-driven*) approach is by using supervised learning, where a dataset with inputs and the corresponding outputs is given to train the emulator to provide accurate estimates [3, 5, 7]. The last category is *reduced order models*, based on projection schemes.

Hierarchical or multi-fidelity surrogate models are based on a simplification of the original model, e.g., 2-d *vs.* 3-d by exploiting a symmetry, or on relaxing tolerances in the numerical solution, e.g., reducing the element order or making the grid coarser (successively refining the grid until an 'acceptable' accuracy is achieved). This leads in lowering the fidelity of the initial model [8, 9]. This category of surrogate model has been used extensively in optimization [10, 11, 12, 13]. The need for reduced resolution to make hierarchical models faster, and at the same time to capture the most important details of the original model has led to the development of multi-scale models. In these types of models usually the global equation is being solved in a coarser grid, while local problems are dealt with in finer grids. Jenny in [14] proposed a multiscale finite volume scheme where the domain is divided into smaller volumes. The centres of the volumes are then used to create a dual grid. Multiscale finite element and volume methods are not always more computationally feasible, but they offer better parallelization. Methods such as multigrid [15] and adaptive mesh refinement [16] allow interpolation techniques to be used on multiple scales at the same domain.

Data-driven surrogate models are based on machine learning algorithms such as artificial neural networks (ANNs) [17, 18], Gaussian process (GP) or 'Kriging' models [19, 20], support and relevance vector machines [21], and radial basis functions [22], applied to a data set provided by a simulator. These techniques are used to find the map between the inputs and the outputs of the model. In [23] a feedforward neural network was used to estimate the remaining life of a Lithium-Ion battery. Bicer et al. [24] used an ANN with different training methods to model a proton exchange membrane fuel cell. Sainath et al. [25] proposed a low rank factorization for deep learning in the last weight layer to reduce the number of parameters of a network trained with a high number of output targets. GP models offer several advantages over ANNs and other methods, especially for limited data, and for problems in which error estimates in the predictions are necessary or desirable. Extending GP models to multi-output problems is, however, not straightforward, in contrast to ANNs. This is discussed in detail in the next section.

The last category of the emulators is the Reduced Order Models (ROMs) which are based on reducing the dimensionality of a problem by Galerkin projection onto a lower dimensional space. This space is a subspace of the original output space, and the methods differ mainly in how this space is selected, i.e., the basis used for the low-dimensional subspace. The dimension of the subspace (number of basis vectors) is kept low to ensure effective dimensionality reduction. Examples include proper orthogonal decomposition (POD) [26] and modal analysis [27]. ROMs and surrogate models in general are becoming indispensable in many areas and applications such as optimization, predictive control uncertainty quantification and sensitivity analysis where real-time model evaluations are needed. In practice, a high-dimensional dataset produced by a high-fidelity model with many degrees of freedom, can be replaced by a low dimensional surrogate that contains most of the variance of the variables. Ghommem et al. [28] developed a combination of a generalized finite element method and mode decomposition to reduce the model dimensionality of flows in porous media. To achieve this they discritised the domain using a finite element method and then used POD and dynamic mode decomposition. Willcox et al. [29] proposed a model order reduction method based on Fourier series. The coefficients of the discrete Fourier were computed and used to form a ROM. Their technique provides accurate and stable results, although its main drawback is the limitation to linear problems. Lieberman [30] used a greedy algorithm to build a reduced order model for the parameters.

1.1 High dimensional problems

This thesis is focused on emulating outputs of very high dimensionality, produced by parametrized partial differential equation models. The dimension of the output is based on the spatial grid where the quantity of interest is evaluated. In many cases, when a fine grid is needed in a model, the degree of the output dimensionality can end being very high. The emulation of outputs having such a dimensionality is challenging due to the computational power needed.

One of the most common approaches is to use GP emulation in which the output of the model is treated as a GP [4, 31, 32, 33]. To extend the GP emulation for multi-outputs, Kennedy [19] used the output as an additional input parameter although, this can be used only for small grids for low number of training points [34]. A mutli-dimensional prior was placed over the outputs by Conti [35] under the assumption of seperability of the covariance structure leading to the linear model of coregionalization [36]. This approach is computationally feasible however it is based on an assumption that is not always true. A further development of this method can be found in [37, 38, 39].

Lawrence in [40] introduced an extension of PCA called dual probabilistic PCA, where the linear mappings can become nonlinear by using a kernel trick with GPs. This leads to Gaussian process latent variable models (GPLVMs), where instead of marginalising the latent variables for the parameter optimisation, the parameters are marginalized and the latent variables are optimised. In an alternative approach also based on latent variables, Alvarez and Lawrence in [41] suggested a convolved GP to deal with multi-output models. The main idea behind their approach is the expression of each output as a convolution between a smoothing kernel and a latent function which has been drawn from a Gaussian process. This means that the outputs could be expressed as jointly Gaussian processes and can be used to model multi-output problems. Damianou and Lawrence [42] proposed deep GP models where the layers of a deep network are modelled as GPs and fed to subsequent layers. Their approach leads to learning complex relationships in the data which can be used in unsupervised and semi-supervised learning.

Higdon et al. [43] proposed GP emulation based on dimensionality reduction (PCA) where the coefficients are independent and uncorrelated. An alternative scheme was used by Bayarri et al. in [44] based on wavelet decomposition. The above mentioned techniques are based on linear dimensionality reduction which will fail on output spaces with high curvatures as will happen with all linear techniques such as multidimensional scaling and canonical correlations analysis.

Another alternative is the use of reduced order modeling for emulating models of partial differential equations [45, 46, 47, 48] which is based on the projection of the initial system of partial differential equations to a reduced dimensional subspace. To achieve this, a numerical method such as finite volume or finite element has to be used to calculate the basis of the POD in order to get the snapshots of the simulator. Reduced order modeling provides an insight into the physical properties of the model under consideration and at the same time a methodological estimation of the errors [48, 49, 50].

Due to the limitation of the linear methods in high-dimensional output spaces with high curvatures, this thesis extends Higdon's method [43] by using manifold learning techniques to deal with this nonlinearities. Kernel PCA and diffusion maps are combined with GPs, neural networks and support vector machines to develop efficient emulators to overcome the aforementioned challenges

1.2 Machine learning

Machine learning merges elements from computer science, mathematics, statistics and even biology and neuroscience to detect patterns in data. The aim is to train a machine to learn from data and find hidden structures and patterns in it, in order to make predictions about unseen data and make decisions under uncertainty. Machine learning can be used in many fields such as speech recognition, computer vision and control robotics and in general in classification and regression tasks. The difference between those two is that the output of the former is categorical while for the latter is continuous. In this thesis machine learning algorithms were used for regression problems.

1.2.1 Types of learning

- Supervised learning: A training set D is provided which contains the inputs $\boldsymbol{x}_i$, $i = 1, ..., N$ and the corresponding targets $\boldsymbol{t}_i$. The algorithm learns to generalise and make predictions of unseen cases. The input $\boldsymbol{x}_i$ usually is a D dimensional vector of numbers and the targets $\boldsymbol{t}_i$ consist of real values (regression) or is categorical (classification).

- Unsupervised learning: In this type of learning the correct outputs are not provided, so the algorithm tries to find similarities (patterns) of the inputs $\boldsymbol{x}_i$ (categorization). This is not a well defined problem as there is no metric system i.e. loss function to compare the targets $\boldsymbol{t}$ from the data $D = \{\boldsymbol{x}_i, \boldsymbol{t}_i\}$ with the output of a model ($\boldsymbol{y}$). In unsupervised learning the aim can be seen as a density estimation goal, where a model is developed in the form of $p(\boldsymbol{x}_i|\boldsymbol{\theta})$ instead of $p(\boldsymbol{y}_i|\boldsymbol{x}, \boldsymbol{\theta})$, where p is the probability, $\boldsymbol{x}_i$ is the input, $\boldsymbol{y}_i$ is the output of the model and $\boldsymbol{\theta}$ is the hyperparameters. This means that unsupervised is unconditional density estimation while supervised is conditional [51].

- Semi-supervised or reinforcement learning: Lies in between supervised (labeled data) and unsupervised (unlabeled data) where a small amount of labeled data is used in conjunction with unlabeled could improve the accuracy of the model.

1.2.1.1 Neural Networks

Artificial neural networks or simply neural networks (NN) are inspired from the way the human brain works. The human brain can be seen as very complex, nonlinear computer that receives and process informations. It has the ability of organising its simple component structures known as neurons to perform many complicated tasks. A NN tries to mimic the human brain in identifying patterns, processing data and making predictions of unseen values. Due to this connection with the human brain they were firstly used in biological systems [52, 53, 54, 55]. NN is a network of interconnected neurons via weights that learn from the data. Neurons are the basic elements of an ANN as they provide local data processing. They are arranged in layers where the first layer is the input layer and the last is the output layer. In between, there is one or more hidden layers of neurons. In most NN such the one used in this thesis, the information is propagated only in one direction (feed-forward NN), meaning from the input to the output layer while there is no connetions between the neurons of the same layer. Each layer of neurons is acts as an input for the next layer of neurons. To be activated each neuron has an activation function. This function acts on a weighted combination of the inputs from each neuron to which the neuron in question is connected. The output becomes the input to neurons in subsequent layers. Each weight expresses information used by the net to solve a problem while each neuron has an activation function that is the received signal from the previous neuron.

Due to the flexibility of using different numbers of neurons and layers in ANN there are different architectures. In general, three main architectures can be identified. A single-layer feed-forward network is the simplest architecture of a NN that contains the input layer where data is fed and propagates to last layer which is the output. In between there is only one hidden layer of neurons. The second category is multilayer feed-forward networks where more than one hidden layer of neurons exist between the input and the output. Using more layers leads to a subcategory of neural networks known as deep NN [56, 57]. The concept in deep learning NN is the same as in multilayer-feedforward networks. The term hidden layers and neurons cites the fact of not being directly connected to the input or the output layer. By adding more layers into a NN it can extract higher order statistics from the input layer. The input layer provides the information to the second layer (first hidden layer) which based on the activation function of the neurons propagates the output of the layer to the next layer. The output layer provides the response of all the network to the activation pattern of the input layer. The last category is recurrent networks which are different than the previous two mentioned before. In the aforementioned categories the information from the input layer to the output is transferred only in one way, forward. In recurrent NN sthere is at least one feedback loop. It can be said that is this case neurons have their own internal memory on what it has been computed so far. So, neurons have as input not only the example they see at the time but also the example of the previous evaluation. The main issue of recurrent neural networks is that is not easy to train them. This issue comes from the vanishing and exploding gradient problem. While during the forward pass the information is passing through a "squishing" function (e.g. sigmoid) preventing the so called explosion of the gradient. Although, the backward pass is linear due to the gradient calculation of the forward pass. This leads to an issue on training RNN with multiple layers. This thesis deals with multilayer feedforward networks as they are faster and more accurate at least for the problems that are being solved here.

The main advantage of an ANN is that it can perform different tasks such as prediction and pattern recognition. It can be used in complex nonlinear regression and classification problems with the accuracy that is desired in each case.

1.2.1.2 Gaussian Processes

In Gaussian processes a prior distribution is placed over the function having the form of a GP that maps the input to the output space, hence the function is a random variable. At different input points the joint distribution of the function at the selected points is multivariate Gaussian [58]. A GP is described by its mean and covariance function. When modelling using GPs in the covariance function exist unknown hyper-parameters which provide the specification of the model and their value is computed through the GP emulation. In most cases the mean is chosen to be zero which can achieved by centering the data. This is the prior knowledge held about on the model without taking into consideration the dataset and it will update as the data are fed to the model. There are several covariance functions that could be used in GPs such as rational quadratic, Ornstein-Uhlenbeck and squared exponential which is also used in this thesis. A GP can also be seen through a Bayesian framework, where a prior distribution is placed over the function that maps the inputs and output and contains the prior belief of the model. Using the dataset better prediction can be made in order to find the posterior distribution. To calculate the posterior, the probability distribution of the data given the hyperparameters (likelihood) has to be evaluated first. Although, the values of the hyper-parameters are unknown and have to be estimated. The estimation of the hyper-parameters can be done by incorporating maximum likelihood estimation where a uniform distribution is assumed over the hyper-parameters and by applying Bayes' theorem to get the probability of the hyper-parameters given the target outputs. By maximizing the probability of the target given the hyperparameters an estimation of the hyperparameters can be obtained from *maximum a posteriori* (MAP) estimation methods.

1.2.1.3 Support Vector Machines

Support vector machines (SVM) is another supervised machine learning technique that is used for both regression and classification tasks. SVMs introduced by Vapnik [59] and later on Boser [60] applied the kernel trick to make them able to deal with non-linear problems. Having a dataset consisting of inputs and the corresponding output for each design point and assuming a linear function that maps the input and the output, SVM has as target to solve a convex optimization problem. The optization problem can be defined as: given an error precision, find a map of the inputs such that its deviation is not greater than the error for its training point and at the same time is as flat as possible, although, such a solution is not always feasible. To overcome this barrier a "soft margin" loss function can be employed which involves slack variables. The resulting formulation can be solved by constructing a Lagrangian function which leads to a dual optimization problem. Taking into account the Karush-Kuhn-Tucker (KKT) conditions the support vectors can be computed.

1.2.2 Dimensionality Reduction

In many scientific fields there is a dependance on dealing with high-dimensional data which raises the need of dimensionality reduction. These high-dimensional data in most cases can be reduced in a lower dimensional space where it is easier to visualize and analyze them without significant loss of the information contained of them. In most cases the data cannot be described by a linear function, hence linear DR techniques will fail. This led to the development of nonlinear techniques such as kPCA, Isomap and diffusion maps. In this section an overview of the techniques used in this thesis is provided, and a full explanation of each technique is given in the relevant chapters.

1.2.2.1 Linear Dimensionality Reduction

1.2.2.1.1 Principal component analysis: One of the most used linear dimensionality reduction techniques is the principal component analysis (PCA) and it is applied in many different areas such as the finance sector (risk management of interest rate derivatives portfolios), engineering (image processing, denoising) and neuroscience. It is effective especially when data lie on a linear subspace. Assuming a dataset of inputs and the corresponding outputs, the target of PCA is to find a low dimensional linear subspace of the ambient space, which is spanned by a basis vector in a way that as much variance of the original data is retained. This subspace is defined of mutually orthogonal axes which are also known as principal components. Having the empirical covariance matrix the eigenvectors of the data can be extracted from it. Using the eigenvectors a new basis can be computed on which the projections of the original data lie.

1.2.2.1.2 Multidimensional scaling Another linear dimensionality reduction technique similar to PCA is multidimensional scaling MDS. Classical MDS was first developed by Torgerson [61] as a method to capture the mapping of data points residing on a the Euclidean space. In [62] MDS was generalized to capture dissimilarities between the data points. The main idea behind MDS is having a dataset consisting of inputs and outputs to minimize a cost function that contains the Euclidian distances between all the training points. This optimization task is solved by using an eigenproblem. The resulting eigenvalues are adjusted in increasing order and the eigenvectors are centered. The first eigenvectors are used to project the data in the low dimensional space.

1.2.2.2 Manifold Learning/Non-linear Dimensionality Reduction

1.2.2.2.1 Kernel PCA The main idea behind kPCA is that instead of using PCA in the original space where the data may suffer from high curvatures, PCA can be performed in a high dimensional feature space where the mapped points reside on a linear space, although, mapping to a higher dimensional space leads to a increased computational cost. To overcome this barrier, the kernel trick has to be used meaning that the inner product of two points in the feature space is given by a kernel function. Assuming a dataset of inputs and the corresponding outputs instead of using the covariance function to perform the standard PCA, the eigenvalue problem is solved for a kernel matrix. The right selection of the kernel function depends on the prior knowledge of the given data.

1.2.2.2.2 Isomap Isomap is an extension of the MDS method that is able to handle nonlinearities in data with high curvatures. This can be achieved by using geodesic rather than euclidean distances. For neighbouring data points the Euclidean distances can be used while for data points that are far-away from each other the geodesic distances has to be approximated by a shortest path distance. In the original Isomap it can be calculated from Dijkstra's [63] and Floyd's [64] algorithms. The first step in Isomap is the determination of the neighbour points based on Euclidean distances. The simplest approach is the connection of each point with the rest of them that lie on a fixed radius or the connection of a number of the closest points. The next step is the construction of the dissimilarity matrix where the distances between points close to each other are Euclidean while for non-neighbouring points the distance is computed as the shortest path through neighbouring points. The final step is the application of the MDS method on the kernel matrix to obtain the representations of the data. A connection of kPCA and Isomap can be found in [65, 66]. Moreover, Choi et. al. [66] proposed a variant known as kernel Isomap to overcome the limitation of the non guaranteed positive definite kernel matrix where Kronecker delta function is introduced in the dissimilarity matrix.

1.2.2.2.3 Diffusion maps In diffusion maps the data points are mapped in a subset called diffusion space and from that the reduced order approximation can be calculated. The aim is to preserve the diffusion distances between the points of the original space to the diffusion space. In diffusion maps the data points are identified as nodes on a graph, while a positive definite kernel is constructed to form a Markov chain and also to define the edge weights. A diffusion process [67] has to be built on the graph and the kernel matrix has to be normalized to construct a degree matrix and then the diffusion matrix. The diffusion map maps the original data into another dimensional space by preserving the diffusion distances.

1.2.3 Reduced Order Modelling

Model reduction of linear and non parametric problems have been used for many years now, although, parametric order reduction is still a challenging area to be explored. Parametric order reduction focuses on systems where their governing equations hinge on a set of parameters. The aim is to create an accurate metamodel that describes the system behaviour for different values of parameters. Dynamical systems are the basic scheme in many scientific areas where modelling is needed to describe of complex systems such as fluid dynamics, signal processing, electrical and mechanical systems etc. These complex underlying phenomena are being modelled by using numerical methods and based on the details they include resulting in high dimensional complex models. Running simulations on these kind of high dimensional data usually is computationally expensive leading to the need of metamodels to evaluate fast and accurate the high resolution data to a lower dimensional space.

In most fields the problems under consideration are described by PDEs which in most cases are complex to be solved. Projection methods try to minimize this complexity by projecting these equation onto a lower dimensional space spanned by a set of basis vectors. There are several approaches for dimensionality reduction such as truncated balanced realization [68], Krylov subspace methods [69] and the most widely used method proper orthogonal decomposition (POD).

A reduced order model is basically a Galerkin projection for the original high dimensional space (produced by the simulator also known as full order model) to a lower dimensional space by forming a basis that retains most of the information. Applying Galerkin projection on PDE problems leads to the final form of an algebraic system for steady state models or in the case of dynamical models in an ODE system. The basis of the ROM can be calculated by balanced truncation[70], Krylov subspace method [71] and POD [72, 26], although, balanced truncation and Krylov method are efficient for use in linear problems or problems where non-linearities are not severe [73].

The main idea of POD is that the time response of a system given the inputs, restrain the main behaviour of the system. These outputs which also called snapshots have to be calculated first by using the full order model (simulator). The snapshots describe the system's state at a specific time and can be written as a matrix $W \in R^{N \times K}$ where N is the total number of snapshots and K is the total number of variables in each snapshot.

Performing POD in nonlinear parametric PDEs is challenging due to:

- The valid basis construction over the parameter space
- High dimensional spaces
- Sparse data use
- Efficient computation of reduced order system matrices and nonlinearities when using the emulator.

To overcome the nonlinearities and to embody the parametric reliance several solutions have been proposed:

- Use of a global basis capturing the whole parameter space
- Interpolation of the local basis
- Interpolation of the local system matrices

POD has been used to reduce the dimensionality in many linear systems or systems with affine parametric depences [74]. Nguyen and Peraire [75] mentioned the problem that arises when using the standard Galerkin projection method in high dimensional data due to the computation of the inner products required to evaluate the weak form of the nonlinearities (in a finite elements framework) and suggested the best-points interpolation method in order to find optimal interpolation points. Constructing ROMs for parametrised, nonlinear problems is still a challenging area and the following have to be considered: 1) the structure of a reliable basis for the whole parameter space which most of the times is high dimensional and 2) the computation of the reduced order matrices and non-linearities. There are different schemes to introduce parametric dependence in the model such as the use of a global basis over the parameter space, the interpolation of a local basis and for the construction of parametrized ROMs, the construction of local ROMs before the interpolation of these ROMs. In [76] Baur et. al. suggested a tangential interpolation method for the computation of the basis. A Moment-matching technique has be used for a single parameter case reduced order model in [77, 78] and extended for the two parameter case in [79]. Meier and Luenberger in [80] proposed an optimal selection strategy for expansion points and tangential direction. Gugerkin et. al. [81] improved the above mentioned technique by introducing the iterative rational Krylov algorithm that fixes the interpolation points and direction using successive substitution until an optimal point has been reached. Everson and Sirovich [82] proposed the gappy-POD for data that are not complete (missing elements from the dataset) and applied it on face recognition. The idea behind this extension of POD is in every iteration the dataset is checked for missing values. In this case an initial value is given based on point-wise mean of all snapshots at that point.

An empirical interpolation method (EIM) has been proposed by Barrault et.al. [83] and used for the approximation of parametric non-linear function by separable interpolants. Discretization schemes applied in EIM and used as an empirical interpolation operator [84, 85, 86] and later on as Discrete Empirical Interpolation (DEIM) [87, 88]. The main idea behind EIM and its discrete version is to replace the Galerkin projection with a computationaly efficient interpolation projection of the non-linear terms of a system. The Gauss-Newton approximation tensors (GNAT) technique was proposed by Carlberg in [89] which is a Petrov-Galerkin projection connected with residual minimization on spatio-temporal discretised PDEs and satisfies the consistency and discrete-optimality conditions.

1.2.4 Sensitivity Analysis

Lowering the time cost of simulations by identifying the most influential parameters and studying their effects is an effective precursor to tasks such as design optimization and uncertainty quantification. This process is referred to as *sensitivity analysis (SA)* [90, 91, 92].

SA methods can be categorized in different ways. In *quantitative* SA the influence of a parameter (usually referred to as *factor* after grouping together all parameters in the form of a vector called the *input*) is assigned (reproducibly) a number called a *sensitivity index* or *importance measure*. In *qualitative* SA, sensitivity is determined though inspection of the outputs, perhaps employing visualization tools such as scatter plots [93]. In *local* SA, the output variability is studied by perturbing an input around a nominal (base) value, while methods that attempt to measure the output variability across the entire input space are termed *global*. For small variations in the inputs local methods may be more computationally efficient. In many cases involving complex nonlinear models, however, local SA methods are inadequate.

Another way to categorize SA is based on the sampling method: one-at-a-time (OAT) and all-at-a-time (AAT). In all but the simplest of cases sensitivity indices must be approximated numerically based on a sample of the inputs (sampling-based SA), together with the corresponding outputs. In OAT sampling, output variations are measured by varying one input factor at a time, while in AAT sampling all input factors are varied using factorial or fractional factorial methods, which takes into account the joint influences of factors due to correlations, but comes at the cost of a high number of input samples in order to achieve accurate results. In local SA, OAT sampling is employed, while AAT is used in most global SA methods.

In this thesis two different global approaches of *SA* were used, the *elementary effect test* (EET) and *variance based sensitivity analysis* (VBSA). The main difference between EET and VBSA is that the former is based on the OAT sampling while the latter on the AAT. The simplest approach of SA is to estimate the derivative around a nominal point of the input space. This partial derivative can be approximated by using finite difference methods. Although, this method does not provide any insight of the dependance between different inputs. An extension of the local SA is to use multiple nominal points to perform global SA (elementary effect test) as proposed by Morris in [94]. The EET method is fully described in subsection 3.1.1. The second SA method used in this thesis is the variance based SA, which is based on the AAT sampling strategy. In VBSA the inputs are treated as stochastic variables leading to a probability distribution over the outputs. There are two sensitivity indices in VBSA as a measure of the contribution of each input and the interactions between all the inputs, the first order sensitivity index and the total effect index respectively. Details of the derivation of both sensitivity indices are given in subsection 3.1.2. The main advantage of using VBSA is that can be combined with the bayesian GP emulation to model high dimensional outputs embedded with probability distribution, which means that the statistics of the distribution can be extracted by using Monte Carlo sampling.

1.2.5 Design of experiments

For the techniques mentioned above, a crucial part that has to be taken into consideration is the design of experiments (DOE). The dataset that is used in order to build an emulator with high accuracy must contain adequate information for the inputs and the outputs. The dataset is created by using the simulator at different inputs, which is also known as design points, to get the corresponding outputs. To cover the whole input space and obtain the results at different points the design of experiments method has to be used [95]. The choice of the method for design of experiments is based on the problem under study and there is not a universal best method. In this thesis latin-hypercube and sobol sequence methods have been used. The former, proposed by McKay [96], is a statistical method that produces random samples from a multidimensional distribution while the latter belongs to quasi-random low-discrepancy sequences proposed by Sobol [97]. Both methods considered that they provide the best coverage of the input space based on [95]. Latin hypercube has been used in chapter 3 to select the design points for the SA techniques and Sobol's sequence was used to select the training points for the emulators used in each chapter.

1.2.6 Numerical Methods for Partial Differential Equations

As mentioned earlier, nowadays in many fields the models that have to be developed in order to describe a physical phenomenon are mostly nonlinear. This means that the governing equations of the system are nonlinear and described by partial differential equations. Moreover, PDEs are generally categorized based on the order of their highest derivative and also can be classified into three main categories, elliptic which arises from diffusion processes, parabolic which can be seen as a transition between elliptic and hyperbolic PDEs such as time dependent diffusion problems, and hyperbolic which involves discontinuities arising for example from shock waves. In most cases PDEs are difficult to solve based on only one universal method. The three main methods being used to solve non-linear partial differential equations are: the finite difference method, the finite volume method and the finite element method. All three methods are described in this subchapter. There are also spectral methods although they are beyond the scope of this thesis [98][99].

Finite difference, volume and element methods have some common steps for solving PDEs. The first step is to partition the spatial domain of the problem into smaller domains also know as the mesh. The next step is the approximation of partial differential equation by using spatio-temporal discretization. The final step is to solve the nonlinear or linear system of algebraic equations at the specific selected points of the second step. The main difference in the afore mentioned methods is that in finite difference the system equations are solved in their strong form (directly), while in finite volume and finite element the use of the weak form of the equations is needed, meaning that the equations are being integrated first. The advantages of using the weak form are the reduction of continuity requirements on the basis functions and the Neumann boundary conditions come naturally. Moreover, finite volume and finite elements give some freedom on constructing the mesh (e.g. shape of the elements in FEM). In this section, the numerical methods are briefly explained, more details of the numerical methods used in this thesis are fully described in the appendix A.

1.2.6.1 Finite Difference (FD)

Having a function of one spatial variable, the finite difference method considers that the partial derivative of any point on the grid can be approximated by the values of the variable at neighbour points. This can be done by using the Taylor's expansion on a selected grid point. Another option is to use polynomial interpolation or quadratic approximations of the function over the grid and then to differentiate. When dealing with continuous PDEs, a discretisation has to take place first of the spatial field, and then the approximation of the first order derivative can be approximated by the forward or backwards difference scheme using the Taylor's expansion. The use of the above mentioned schemes on each term of the PDE leads to a dynamical system. Next, the FD formulation is applied on the temporal derivative (time marching) to obtain the final approximations of the PDE. More details cane be found in section A1.1.1

1.2.6.2 Finite Volume (FVM)

The FVM is one of the most used numerical methods in fields such fluid dynamics, finance, image processing etc, due to its flexibility in creating unstructured mesh without the need of stabilization schemes while at the same time it offers good conservation of mass, momentum and energy. The first step in the FVM is the partition of the domain into a finite number of control volumes where their union forms the initial domain. These volumes form the mesh which can be regular or sometimes unstructured and define the boundaries of them. For each cell the governing equation is integrated to obtain an algebraic equation system. In the simplest approach, the integral is approximated for the central point of the cell. Then the system is solved to calculate the values of the output at the discrete point.

1.2.6.3 Finite Element Method (FEM)

In FEM the domain is divided in smaller subdomains that create a mesh. The initial step is the derivation of the so called weak form of the PDE, which has also to satisfy the condition of the original/strong form. To derive the weak form of the PDE, the strong form is multiplied by a test function and then it is integrated over the domain. The choice of the test function is based in a way that is being cancelled on the Dirichlet boundary. Next, the discretization of the domain into subdomains is taking part which called elements and usually are triangular. Connected to elements are nodes which could appear in the interior, the edges or even the vertices of the element. Each node is affiliated to a basis function (usually piecewise polynomials) which is nonzero if it is a part of the element otherwise is zero. Then, the dependent variable is approximated for each element concluding to a system of algebraic equations or ODEs.

1.2.7 Thesis Contribution

In this thesis ANNs, GPs and SVM for regression are combined with dimensionality reduction techniques to find the mapping of inputs and the output which lie on high dimensional spaces and applied in spatiotemporal data produced by a simulator (computer model). As can be seen in chapter 2, Neural Network models are multilateral and can learn quickly. In comparison to Gaussian Process Emulators, they can learn the various coefficients of the reduced order basis at the same time leading in learning multiple outputs and finding the correlation between them. In order to improve generalisation of the NN and avoid the use of cross-validation, a Bayesian regularization scheme was used meaning that a prior Gaussian distribution was placed over the weights during the training phase. More details can be found in subsection 2.1.4. The emulators are combined with manifold learning techniques such as Isomap, kernel principal component analysis (kPCA) and diffusion maps. The challenge in manifold learning is to find the inverse mapping also known as the pre-image problem. A novel and fast approach to the pre-image problem is introduced based on linear algebra avoiding the usual issues encountered such as local minima pitfalls and initial conditions.

In the reduced order modelling part, The POD method has been extended and applied in dynamic, parametric dependent, linear and nonlinear partial differential equations, in order to approximate the basis of new parameters values. In this extended POD method, which has the capability of being used with methods such empirical interpolation method (EIM), discrete empirical interpolation method (DEIM) greedy algorithms etc. to find the basis the method of snapshots was used in the place of the direct approximations of the system matrices. To efficiently approximate the snapshots, which usually lie in high dimensional spaces, the manifold learning techniques described in section 1.2.2.2 have been used. DEIM was used to deal with the nonlinearities for the approximation of snapshots at the desired location points (design of experiments) of the parameter space. The above described method is then applied to a 2D linear convection-diffusion problem approximated using finite volume (subsection 4.3.1), where the stochastic input is the velocity field. A second application of the method was performed on a nonlinear 1D Burger's equation descritised by the finite element method subsection 4.3.2.

When the underlying problem is represented by a system of parameterized steady-state or time-dependent partial differential equations (PDEs), the simulator outputs are *spatial* or *spatio-temporal fields* (e.g. velocity, pressure, temperature), which are functions of multiple input parameters. The outputs of interest could include one or more scalar/vector fields or the time evolution of a scalar quantity. For a single scalar quantity, the simulator can be represented as a function $\boldsymbol{\eta} : \mathcal{X} \to \mathbb{R}^d$, taking as inputs $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^l$ and generating outputs $\mathbf{y} \in \mathbb{R}^d$. The outputs are the vectorized field variable values at d discrete points in a spatial domain or time interval. Very few studies have focused on the emulation of such simulators, which 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.

1.2.8 Thesis Outline

In chapter 1 a literature review and an overview of the techniques used in this thesis have been presented. In chapter 2 an emulator based on ANN combined with linear and nonlinear dimensionality reduction techniques is presented in order to emulate high dimensional spatiotemporal data. The emulator was tested on a model of electromagnetic propagation. The results were also compared to the GPE approach. The network was trained using a bayesian approach based on bayesian regularization. The DR techniques used with the NN are the Isomap and kPCA which are fully explained in section 2.1.1.

In chapter 3, a GP emulator was used to perform a probabilistic sensitivity analysis of high dimensional output data with applications to lithium ion batteries. Computer aided design and analysis is an important tool in the development and testing of battery systems. Full battery models are highly complex, which impedes their application to tasks such as optimization and uncertainty analysis. Sensitivity analysis (SA) is an effective method for identifying the most important parameters in a model (lowering the computational burden in other tasks of design and analysis), but SA can also involve a prohibitive time cost, which has motivated the use of emulators. For high-dimensional output problems, emulators must themselves be computationally feasible. In this chapter a probabilistic framework for SA of high-dimensional-output battery models is developed using a Gaussian process emulator based on dimension reduction in the form of principal component analysis. This allows the performance of SA under uncertainty for multi-output problems, providing error bounds for the emulator-based prediction of sensitivity measures. It is shown how this can be achieved using Monte Carlo sampling or possibly by using semi-analytical expressions with highly efficient sampling. Moreover, SA can be performed for multivariate outputs by ranking the sensitivity measures related to the principal coefficients.

In chapter 4 a reduced order modelling approach is presented based on the POD technique which is applied on linear and nonlinear parameter dependent PDEs. An extension of the DEIM technique is explained in order to deal with nonlinearities. Two examples are provided for comparison to the classic global basis approach.

The Diffusion map is explained in subsection 5.2.1 and kernel PCA in subsection 2.1.1. In chapter 5 they are combined with a GPE to deal with high dimensional data. A new approximation of the inverse-mapping for diffusion maps and kPCA is also presented in section 5.5. In the same chapter two more emulation methods (ANN and SVM) are used to compare the results to Higdon's method.

1.3 Concluding remarks

In this chapter the motivation and problems under consideration were introduced. A definition for the simulator and the emulator was given and made the classification of its categories. Also, the main advantages of using an emulator for high dimensional data were outlined. Moreover, the main categories of machine learning (supervised, unsupervised and reinforcement learning) were presented. The emulators used in this thesis were also briefly discussed and are thoroughly introduced in its chapter. Linear and non-linear dimensionality reduction methods explained and the main idea of combining these techniques with the emulators was given.

Furthermore, the advantages of using sensitivity analysis techniques with emulators were introduced followed the need of design of experiments during the building stage when constructing an emulator. Also, the numerical methods used throughout this thesis for the discretization of partial differential equations were discussed.

Chapter 2

Neural Network based emulation of high dimensional spatio-temporal data combined with linear and non-linear dimensionality reduction techniques

In this chapter, linear and non linear dimensionality reduction techniques are combined with feed-forward neural networks to build an efficient emulator for high spatiotemporal data and applied on an electromagnetic application. An ANN has the advantage of being able to learn multiple coefficients of the reduced basis meaning that are able to capture efficiently correlation of the outputs, although their main drawback is suffering from overfitting. To address this issue, in this chapter the Bayesian Regularization (BR) learning technique has been used (i.e placing a prior over the weight vector) instead of using cross-validation.

This chapter is based on [100]: V Triantafyllidis, W Xing, AA Shah, PB Nair, *Neural Network Emulation of Spatio-temporal Data Using Linear and Nonlinear Dimensionality Reduction* *in Advanced Computer and Com-*

munication Engineering Technology pp 1015-1029, Lecture Notes in Electrical Engineering, 2016

Abbreviations

$\boldsymbol{\eta}$	representation of computer model as a function
$\boldsymbol{\xi}$	input vector
$\mathbf{y}$	output of the computer model
$\mathbf{w}_i$	PCA basis
$z_i^{(j)}$	uncorrelated coefficients of the basis
r	number of most dominant basis vectors
$\mathbf{D}$	dissimilarity matrix
δ_{ij}	Euclidean distances between points i and j
$\mathbf{K}$	Centred kernel matrix
$\boldsymbol{\Lambda}$	eigenvalues matrix
$\mathbf{V}$	eigenvectors matrix
$\mathbf{C}_{\mathcal{F}}$	sample covariance matrix in $\mathcal{F}$
E_D	network square error
α	inverse variance of the zero-mean Gaussian noise
β	inverse variance of the weights

$\mathbf{a}$	vector of network weights
$\mathcal{D}$	the dataset
$\mathcal{M}$	ANN model
N	total number of parameters in the model
m	number of training points
$\mathbf{H}^{MP}$	Hessian matrix
N_n	number of neighbour points
E_z	electric field in the transversal direction z
n	refractive index
k_0	free space wave number
$\widehat{\mathbf{n}}$	unit normal
b	width of rectangular sections of waveguide
ω	angular frequency of the incident wave
c	speed of light
f	frequency
θ	angle of incidence
ϵ	permittivity of air
μ	permeability of air
σ	conductivity of air

2.1 Emulation of spatio-temporal data sets

An emulator provides a probability distribution for the outputs of a computer model. The computer model is represented as a function $\boldsymbol{\eta} : \mathcal{X} \to \mathbb{R}^d$, taking as inputs (or parameters) $\boldsymbol{\xi} \in \mathcal{X} \subset \mathbb{R}^l$ and generating outputs $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi}) \in \mathcal{O} \subset \mathbb{R}^d$. The emulator is trained using m computer model outputs $\mathbf{y}^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)})$, referred to as *training points*, at selected *design points* $\boldsymbol{\xi}^{(i)} \in \mathcal{X} \subset \mathbb{R}^l$.

Spatio-temporal data sets from parametrized PDE models

Consider a generic parameterized nonlinear computer model (e.g., a system of parameterized partial differential equations (PDEs)) with input parameters $\boldsymbol{\xi} \in \mathbb{R}^l$ and outputs $y(\mathbf{x}, t; \boldsymbol{\xi})$ computed at different points in a spatial domain $\Omega \subset \mathbb{R}^2$. In this notation, $\mathbf{x}$ denotes the spatial variable and t represents time. The computer model is executed at design points $\boldsymbol{\xi}^{(k)}$, $k = 1, \ldots, m$. In steady-state problems this yields the values of $y(\mathbf{x}; \boldsymbol{\xi})$ at locations $\mathbf{x}_i$, $i = 1, \ldots, d$, on a spatial grid. These values $y_i^{(k)} := y^{(k)}(\mathbf{x}_i; \boldsymbol{\xi}^{(k)})$ can be vectorized as follows:

$$\mathbf{y}^{(k)} := (y_1^{(k)}, \ldots, y_d^{(k)})^T \in \mathbb{R}^d \tag{2.1}$$

For dynamic problems, $\mathbf{y}^{(k)}$ can be defined in a similar manner, with $d = d' \times N_t$, where d' is the number of spatial locations and N_t is the number of time steps. The method developed below can be applied to a single field of interest or to the emulation of multiple fields (PDE systems) as explained later.

Clearly, an ANN with d (as defined above) output neurons will not be computationally practical in many cases. For highly complex problems involving, e.g., interface tracking or phase change, the number of grid points required to fully resolve phenomena at all scales can lead to d values in excess of 10^6. To overcome this issue, DR (of the output space) is employed.

2.1.1 Dimensionality reduction

Linear methods for dimension reduction

PCA provides a basis $\mathbf{w}_i$, $i = 1, \ldots, d$, for $\mathbb{R}^d$ (and therefore $\mathcal{O}$) that is defined by the eigenvectors of the sample covariance matrix. For each input $\boldsymbol{\xi}^{(j)}$, the corresponding output has a unique representation $\mathbf{y}^{(j)} = \sum_{i=1}^{d} z_i^{(j)}\mathbf{w}_i$, in which the uncorrelated coefficients in this basis, $z_i^{(j)}$, are naturally ordered in a non-increasing manner with i. The data can be projected (potentially) onto a low-dimensional subspace of $\mathbb{R}^d$ by using the r most dominant basis vectors $\mathbf{w}_i$: $\mathbf{z}_r^{(j)} := (z_1^{(j)}, \ldots, z_r^{(j)})^T \in \mathbb{R}^r$. ANN is then performed on the input-output pairs $(\boldsymbol{\xi}^{(j)}, \mathbf{z}_r^{(j)})$, $j = 1, \ldots, m$, to yield a mean value for the random vector $\mathbf{z}_r = (z_1, \ldots, z_r)^T$ corresponding to a test input $\boldsymbol{\xi}$. The predicted output at the test input $\boldsymbol{\xi}$ is obtained as the linear combination

$$\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi}) \approx \sum_{i=1}^{r} z_i \mathbf{w}_i \tag{2.2}$$

or $\mathbf{y} = \mathbf{W}_r\mathbf{z}_r$, where $\mathbf{W}_r := [\mathbf{w}_1 \ldots \mathbf{w}_r]$.

Multidimensional scaling (MDS) is a mapping $\boldsymbol{\chi}_r : \mathcal{O} \to \mathcal{S}_r \subset \mathbb{R}^r$ that relates the Euclidean distances δ_{ij} between data points $\boldsymbol{\chi}_r(\mathbf{y}^{(i)})$ and $\boldsymbol{\chi}_r(\mathbf{y}^{(j)})$ in the mapped space $\mathcal{S}_r$ to 'dissimilarities' d_{ij} between $\mathbf{y}^{(i)}$ and $\mathbf{y}^{(j)}$ in data space $\mathcal{O}$. Let $\mathbf{D} := [d_{ij}]$ denote the dissimilarity matrix. Classical scaling [61] is an isometry in which dissimilarities are defined as Euclidean distances:

$$\delta_{ij} = d_{ij} = ||\mathbf{z}_r^{(i)} - \mathbf{z}_r^{(j)}||$$

for points $\mathbf{z}_r^{(i)}$ and $\mathbf{z}_r^{(j)}$ in $\mathcal{S}_r$, $i, j = 1, \ldots, m$. Let $\widehat{\mathbf{Z}}_d := [\mathbf{z}_d^{(1)}, \ldots, \mathbf{z}_d^{(m)}]^T$, or in centred form, $\mathbf{Z}_d = \mathbf{H}\widehat{\mathbf{Z}}_d$. It can be shown that

$$-(1/2)\mathbf{H}(\mathbf{D} \circ \mathbf{D})\mathbf{H} = \mathbf{Z}_d\mathbf{Z}_d^T = \mathbf{K} \tag{2.3}$$

where $\mathbf{K}$ is a centred kernel matrix and $\circ$ denotes a Hadamard product. Spectral decomposition yields $\mathbf{K} = \mathbf{V}_d\boldsymbol{\Lambda}_d\mathbf{V}_d^T$, where $\boldsymbol{\Lambda}_d := \mathrm{diag}(\lambda_1, \ldots \lambda_d) \in \mathbb{R}^{d\times d}$ and $\mathbf{V}_d := [\mathbf{v}_1, \ldots \mathbf{v}_d] \in \mathbb{R}^{m\times d}$. The non-zero eigenvalues λ_i, $i = 1, \ldots, d$, are arranged in a non-increasing order and the corresponding eigenvectors $\mathbf{v}_i \in \mathbb{R}^m$ are normalized. The data can be represented as $\mathbf{Z}_d = \mathbf{V}_d\boldsymbol{\Lambda}_d^{1/2} \in \mathbb{R}^{m\times d}$, and embedded in an r-dimensional linear subspace $\mathcal{S}_r$ of $\mathbb{R}^d$ by setting $\mathbf{V}_r := [\mathbf{v}_1, \ldots \mathbf{v}_r]$ and $\boldsymbol{\Lambda}_r := \mathrm{diag}(\lambda_1, \ldots \lambda_r)$ to yield

$$\mathbf{Z}_r = \mathbf{V}_r\boldsymbol{\Lambda}_r^{1/2} \tag{2.4}$$

The rows $\mathbf{z}_r^{(i)} \in \mathcal{S}_r$ of $\mathbf{Z}_r$ are the low-dimensional representations of the data points. MDS is equivalent to PCA (the coordinates are identical) when Euclidean distances are used. Both of these linear methods will fail when no linear subspace of $\mathbb{R}^d$ can accurately describe the output space $\mathcal{O}$. In such cases, nonlinear DR (or manifold learning) can be employed.

Nonlinear methods for dimension reduction

In contrast to MDS, Isomap uses geodesic distances for the dissimilarities between points on the manifold $\mathcal{O}$ [101]. Neighbourhood points on the manifold can be determined by using either of the following methods: (i) all points lying within an ϵ ball; or (ii) the n (neighbourhood number) closest points. A dissimilarity matrix $\mathbf{D} := [d_{ij}]$ is then constructed by: (i) using Euclidean distances between neighbours as the geodesic distances; (ii) for non-neighbouring points, using the shortest path distances through neighbouring points. Classical scaling on the kernel matrix $\mathbf{K} = -(1/2)\mathbf{H}(\mathbf{D} \circ \mathbf{D})\mathbf{H}$ subsequently yields a representation of the data in $\mathbb{R}^r$ (an r-dimensional parameterization of $\mathcal{O}$).

kPCA [102] maps high-dimensional data in a space $\mathcal{O}$ to a higher-dimensional feature space $\mathscr{F}$ *via* a mapping $\boldsymbol{\phi} : \mathcal{O} \to \mathscr{F}$, in which linear PCA is performed. In our case, the data consists of the training data $\mathbf{y}^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)}) \in \mathcal{O} \subset \mathbb{R}^d$, $i = 1, \dots, m$, 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:

$$\mathbf{C}_{\mathcal{F}}\mathbf{w} = \left(\frac{1}{m} \sum_{i=1}^{m} \widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(i)}) \left(\widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(i)}) \right)^T \right) \mathbf{w} = \lambda \mathbf{w}, \tag{2.5}$$

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

Equation (2.5) shows that the eigenvectors $\mathbf{w}$ are linear combinations of $\widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(i)})$, i.e., $\mathbf{w} = \sum_{i=1}^{m} \alpha_i \widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(i)})$. Using this expression in Eq. (2.5) and premultiplying by $\widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(i)})^T$ (noting that $\widetilde{\mathbf{K}}$ is positive semidefinite), yields the eigenvalue problem $\widetilde{\mathbf{K}}\boldsymbol{\alpha} = m\lambda\boldsymbol{\alpha}$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^T$. Once computed, the orthonormal $\boldsymbol{\alpha}_i$ are rescaled by $\boldsymbol{\alpha}_i \mapsto \boldsymbol{\alpha}_i/\sqrt{\lambda_i} = \widetilde{\boldsymbol{\alpha}}_i$. This defines orthonormal eigenvectors $\widetilde{\mathbf{w}}_i = \sum_{j=1}^{m} \widetilde{\alpha}_{ji} \widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(j)}),\ i = 1, \ldots, m$, where $\widetilde{\alpha}_{ji} = \alpha_{ji}/\sqrt{\lambda_i}$ and α_{ji} denote the j-th components of $\widetilde{\boldsymbol{\alpha}}_i$ and $\boldsymbol{\alpha}_i$, respectively. Strictly speaking, there are $\min(\dim \mathscr{F}, m)$ basis vectors $\widetilde{\mathbf{w}}_i$, but it is assumed for the purposes of illustration that $\dim \mathscr{F} > m$, without loss of generality. A mapped training point $\widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(j)})$ can be expressed in the basis $\{\widetilde{\mathbf{w}}_i\}_{i=1}^{m} \subset \mathcal{F}$ as $\widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(j)}) = \sum_{i=1}^{m} z_i(\mathbf{y}^{(j)})\widetilde{\mathbf{w}}_i$, where the i-th coefficient is calculated as follows:

$$\begin{aligned} z_i(\mathbf{y}^{(j)}) = \widetilde{\mathbf{w}}_i^T \widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(j)}) &= \sum_{l=1}^{m} \widetilde{\alpha}_{li} \widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(l)})^T \widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(j)}) \\ &= \sum_{l=1}^{m} \widetilde{\alpha}_{li} \widetilde{K}_{lj} = \widetilde{\boldsymbol{\alpha}}_i^T \widetilde{\mathbf{k}}_j = \widetilde{\boldsymbol{\alpha}}_i^T \mathbf{H}(\mathbf{k}_j - \mathbf{K}\mathbf{1}), \end{aligned} \tag{2.6}$$

for $i = 1, \ldots, m$, where $\mathbf{k}_j = (K_{1j}, \ldots, K_{mj})^T$ and $\widetilde{\mathbf{k}}_j = (\widetilde{K}_{1j}, \ldots, \widetilde{K}_{mj})^T$. It is therefore possible define $\mathbf{z}(\mathbf{y}^{(j)}) = (z_1(\mathbf{y}^{(j)}), \ldots, z_m(\mathbf{y}^{(j)}))^T$, where the $z_i(\mathbf{y}^{(j)})$, $i = 1, \ldots, m$, are given by Eq. (2.6).

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 $\widetilde{\mathbf{w}}_i$ (equal to λ_i) decreases as i increases and the coefficients in an expansion of a mapped training point in the basis $\{\widetilde{\mathbf{w}}_i\}_{i=1}^{m}$ are uncorrelated. The goal is to find an r-dimensional approximation of the points $\widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(j)})$, where ideally $r \ll m$. The reconstruction error [103] of the projection $\widetilde{\boldsymbol{\phi}}_r(\mathbf{y}^{(j)}) = \sum_{i=1}^{r} z_i(\mathbf{y}^{(j)})\widetilde{\mathbf{w}}_i$ of $\widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(j)})$ onto the subspace $\mathcal{F}_r = \mathrm{span}(\widetilde{\mathbf{w}}_1, \ldots, \widetilde{\mathbf{w}}_r)$ is given by $||\widetilde{\boldsymbol{\phi}}_r(\mathbf{y}^{(j)}) - \widetilde{\boldsymbol{\phi}}(\mathbf{y}^{(j)})||^2 = \sum_{i=r+1}^{m} \lambda_i^2$, where $||\cdot||$ is the standard Euclidean norm for $\dim \mathscr{F} < \infty$ or the $L^2(\mathcal{O})$ norm of (equivalence classes of) square integrable functions on $\mathcal{O}$ for $\dim \mathscr{F} = \infty$. The value of r is typically chosen according to a variance criterion (or modal energy) [103]: Select r such that $\sum_{i=1}^{r} \lambda_i / \sum_{i=1}^{m} \lambda_i > \varrho$ for some threshold ϱ.

Now a mapping $\widetilde{\boldsymbol{\phi}}_r : \mathcal{O} \to \mathcal{F}_r$ is defined as the orthogonal projection of $\widetilde{\boldsymbol{\phi}}(\cdot)$ onto $\{\widetilde{\mathbf{w}}_i\}_{i=1}^{r}$:

$$\widetilde{\boldsymbol{\phi}}_r(\mathbf{y}^{(j)}) = \sum_{i=1}^{r} z_i(\mathbf{y}^{(j)})\widetilde{\mathbf{w}}_i. \tag{2.7}$$

The notation $\mathbf{z}_r(\mathbf{y}^{(j)}) = (z_1(\mathbf{y}^{(j)}), \ldots, z_r(\mathbf{y}^{(j)}))^T$ is used, which, from Eq. (2.6), is given by $\mathbf{z}_r(\mathbf{y}^{(j)}) = [\widetilde{\boldsymbol{\alpha}}_1 \ldots, \widetilde{\boldsymbol{\alpha}}_r]^T \mathbf{H}(\mathbf{k}_j - \mathbf{K}\mathbf{1})$.

Remark 1. *It is assumed that the training data captures the structure of $\mathcal{O}$ sufficiently well to (implicitly) define a representative basis, $\widetilde{\mathbf{w}}_i$, $i = 1, \ldots, m$, for the image $\widetilde{\boldsymbol{\phi}}[\mathcal{O}] \subset \mathcal{F}$ of the entire space $\mathcal{O}$ under $\widetilde{\boldsymbol{\phi}}$. Equation (2.7) then yields a reduced-dimensional approximation $\widetilde{\boldsymbol{\phi}}_r(\mathbf{y}) = \sum_{i=1}^r z_i(\mathbf{y})\widetilde{\mathbf{w}}_i$ for an arbitrary $\mathbf{y} \in \mathcal{O}$. Equivalently, by the injectivity of $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$, and assuming that the feature map is injective, Eq. (2.7) defines a map $(\widetilde{\boldsymbol{\phi}}_r \circ \boldsymbol{\eta})(\cdot) = \widetilde{\boldsymbol{\phi}}_r(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \to \mathcal{F}_r$, i.e., directly from the entire permissible input space $\mathcal{X}$ to $\mathcal{F}_r$. The basis vectors are, however, unknown without an explicit form for $\boldsymbol{\phi}$. For an arbitrary input $\boldsymbol{\xi} \in \mathcal{X}$, the coefficients $z_i(\mathbf{y})$ define computable maps $\mathbb{z}_i(\cdot) = z_i(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \to \mathbb{R}$ and $\mathbf{z}_r(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \to \mathbb{R}^r$. Thus:*

$$\begin{aligned} \widetilde{\boldsymbol{\phi}}_r(\boldsymbol{\eta}(\boldsymbol{\xi})) &= \sum_{i=1}^{r} \mathbb{z}_i(\boldsymbol{\xi})\widetilde{\mathbf{w}}_i, \\ \mathbf{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi})) &= (\mathbb{z}_1(\boldsymbol{\xi}), \ldots, \mathbb{z}_r(\boldsymbol{\xi}))^T. \end{aligned} \tag{2.8}$$

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

2.1.2 Main algorithm

The emulation algorithm employing DR on the output space is now described in the pseudo code below, including for multiple spatio-temporal data sets. The last step relates to reconstruction of the predicted point in physical space (in $\mathcal{O} \subset \mathbb{R}^d$) and is described in the sequel.

Algorithm 1: ANN learning of spatio-temporal models using DR

---

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

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

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

4. Using $\mathbf{z}_r$ approximate the computer model output $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$ by solving the pre-image problem (see below).

2.1.3 Pre-image problem (inverse mapping)

When using PCA, the reconstruction of the point in physical space $\mathcal{O}$ is given by the linear combination (2.2). In Isomap and kPCA only the predicted coordinates, $z_1, \ldots, z_r$, of a point $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$ in the reduced space are available. In Isomap, the Euclidean distances between points in the reduced space are equal to geodesic distances $d_{i,*}$ between a predicted point $\mathbf{y}$ and points $\mathbf{y}^{(i)} \in \mathcal{O}$ in physical space. Local linear interpolation can be used to approximate the coordinates of $\mathbf{y}$ by using these geodesic distances as weights [104]:

$$\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi}) \approx \sum_{i=1}^{N_n} \frac{w_i}{\sum_{i=1}^{N_n} w_i} \mathbf{y}^{(i)} \tag{2.9}$$

where $w_i = 1/d_{i,*}$ and N_n is the number of neighbours selected for the reconstruction.

The same method can be used for kPCA. ANN on the first r kPCA coefficients (in the basis $\widetilde{\mathbf{f}}_j$) yields an approximation $\boldsymbol{\phi}(\mathbf{y})$ of the point $\mathbf{y}$ in feature space. The distance $\widetilde{d}_{i,*}$ between $\boldsymbol{\phi}(\mathbf{y}^{(i)})$ and $\boldsymbol{\phi}(\mathbf{y})$ in feature space is given by:

$$\widetilde{d}_{i,*}^2 = \boldsymbol{\phi}(\mathbf{y})^T \boldsymbol{\phi}(\mathbf{y}) + \boldsymbol{\phi}(\mathbf{y}^{(i)})^T \boldsymbol{\phi}(\mathbf{y}^{(i)}) - 2\boldsymbol{\phi}(\mathbf{y}^{(i)})^T \boldsymbol{\phi}(\mathbf{y}) \tag{2.10}$$

Substituting equation (2.7) into equation (2.10) for Gaussian kernel gives:

$$\widetilde{d}_{i,*}^2 = \boldsymbol{\tau}^T \Phi^T \Phi \boldsymbol{\tau} + k(i,i) - 2\boldsymbol{\tau}^T \Phi^T \boldsymbol{\phi}(\mathbf{y}^{(i)}) = \boldsymbol{\tau}^T K \boldsymbol{\tau} + 1 - 2\boldsymbol{\tau}^T \mathbf{k}_i \tag{2.11}$$

where

$$\begin{aligned} \boldsymbol{\tau} &= H\widetilde{U}_r \mathbf{z}_r^{(j)} + \mathbf{1} \\ \mathbf{k}_i &= (k(1,i), \ldots, k(m,i))^T \end{aligned} \tag{2.12}$$

For an isotropic kernel ($k(\mathbf{y}, \mathbf{y}^{(i)}) = k(||\mathbf{y} - \mathbf{y}^{(i)}||^2)$), the relationship $\widetilde{d}_{i,*}^2 = 2 - 2k(d_{i,*}^2)$ follows from equation (2.10). In the case of a Gaussian kernel:

$$d_{i,*}^2 = -2s^2 \log\left(2 - \widetilde{d}_{i,*}^2/2\right) \tag{2.13}$$

which is combined with (2.11) to yield $d_{i,*}$. For other kernel functions [105], similar relationships can be derived.

It should also be noted that it is also possible to reconstruct $\mathbf{y}$ from the predicted coefficients using a least-squares approximation. This method is, however, prone to instability, as is the fixed point algorithm of Mika et al. [106].

2.1.4 Bayesian Regularization

In order to improve generalization (and avoid cross-validation), Bayesian regularization [107] is used. A prior (zero-mean, Gaussian) distribution is placed on the network weights (for a fixed architecture), which leads to the minimization of $F(\mathbf{a}) = \beta E_D/2 + \alpha E_W/2$, where E_D is the network square error, α is the inverse variance of the zero-mean (assumed) Gaussian noise, β is the inverse variance of the weights, and $E_W = ||\mathbf{a}||^2$, where $\mathbf{a}$ is the vector of network weights. The posterior density of the weights is given by:

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

where $\mathcal{D} = \{\mathbf{y}^{(i)}\}_{i=1}^m$ is the data set, $\mathcal{M}$ indicates the ANN model, $P(\mathbf{a}|\alpha, \mathcal{M})$ is the prior density and $P(\mathcal{D}|\mathbf{a}, \beta, \mathcal{M})$ is the likelihood function. The optimal weights should maximize the posterior likelihood $P(\mathbf{a}|\mathcal{D}, \alpha, \beta, \mathcal{M})$. A uniform prior density $P(\alpha, \beta, \mathcal{M})$ for the parameters α, β gives:

$$P(\mathcal{D}|\alpha, \beta, \mathcal{M}) = \frac{\mathbf{Z}_F(\alpha, \beta)}{\mathbf{Z}_D(\beta)\mathbf{Z}_W(\alpha)} \frac{\exp(-\beta E_D - \alpha E_W)}{\exp(-F(\boldsymbol{a}))} = \frac{\mathbf{Z}_F(\alpha, \beta)}{\mathbf{Z}_D(\beta)\mathbf{Z}_W(\alpha)} \tag{2.15}$$

in which $\mathbf{Z}_D(\beta) = (\pi/\beta)^{(m/2)}$ and $\mathbf{Z}_W(\alpha) = (\pi/\alpha)^{(N/2)}$, where N is the total number of parameters in the model. The unknown normalization factor $\mathbf{Z}_F(\alpha, \beta)$ can be approximated in terms of the Hessian matrix $\mathbf{H}^{MP}$ of $F(\mathbf{a})$ by a quadratic Taylor series expansion of $F(\mathbf{a})$ around its minimum, at $\mathbf{a} = \mathbf{a}^{MP}$. Placing the result in (2.15) and differentiating yields

$$\alpha^{MP} = \frac{\gamma}{2E_W(\mathbf{a}^{MP})} \qquad \beta^{MP} = \frac{m - \gamma}{2E_D(\mathbf{a}^{MP})} \tag{2.16}$$

where $\gamma = N - 2\alpha^{MP}/\mathrm{tr}(\mathbf{H}^{MP})$. 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 [108] (implemented in the trainbr function in Matlab).

2.2 Results and discussion

Details of training and testing

In both examples below, the data set consisted of 500 points ($\mathbf{y}^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)})$), with inputs $\boldsymbol{\xi}^{(i)}$ selected using a Sobol sequence (uniform sampling) design-of-experiment (DOE). This was found to be adequate for the examples presented below. It must be noted that the DOE is, in general, a vital aspect of any emulation strategy. An appropriate sampling of the input space is paramount for generating training samples that are representative of the region output space $\mathcal{O}$ that is of interest. In the present case, the training samples must generate a basis (either in physical space or in a feature space) that accurately captures the output space. Since these issues are encountered in all emulation methods, they are not focus of on here.

400 points were reserved for testing and up to 100 points were used for training ($m \leq 100$). 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 DR 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. 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. The ANN architecture in all cases contained a single hidden layer and the ANN was trained using Bayesian regularization [107]. The number of neurons for each example was selected using sequential network construction [108]. In general, an arbitrary ANN architecture can be used within the framework.

Example 1: 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:

$$-\nabla^2 E_z - n^2 k_0^2 E_z = 0$$

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

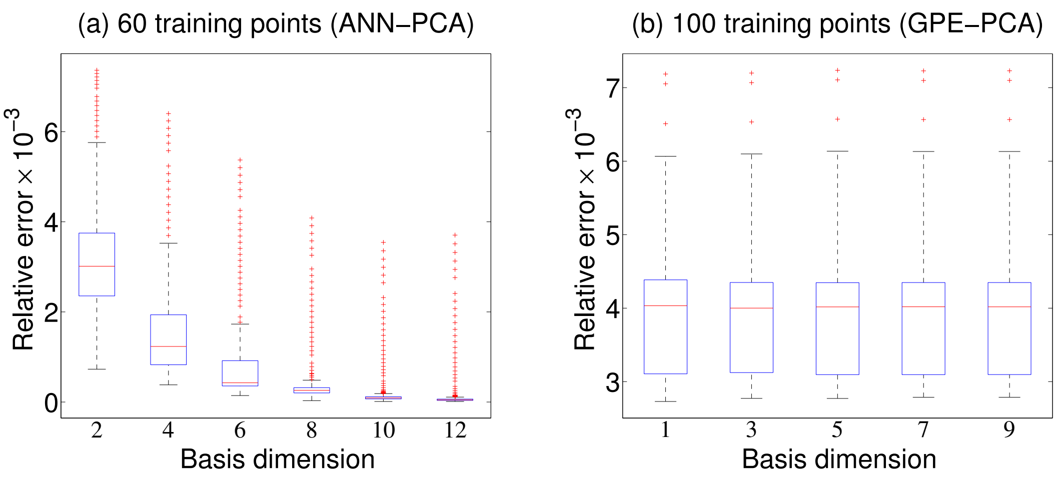


Figure 2.1: Boxplots of the relative errors for different numbers of components (r) using ANN with PCA ($M = 60$) and GPE with PCA ($M = 100$) in the waveguide example.

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: $\widehat{\mathbf{n}} \times \boldsymbol{E} = 0$, where $\widehat{\mathbf{n}}$ is the unit normal. The incident field has the form:

$$\boldsymbol{E} = (0, 0, \sin(\pi(b/2 - \xi)/b)) = \Re(\boldsymbol{E}e^{i\omega t}) \tag{2.17}$$

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 (ξ_l, η_j), $l, j = 1, \ldots, 100$, were vectorized (see equation 2.1) to give 500 data points $\mathbf{y}^{(i)}$ in $\mathbb{R}^d$. Up to 100 were used for training and the remainder for testing.

Results

All three dimension reduction methods 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 Figure 2.1 (a) for 60 training points. The other methods gave similar results, so to conserve space they are omitted. The standard method of Higdon et. al. [43] using a maximum likelihood estimate (MLE) for the hyperparameters failed to provide meaningful results, as demonstrated in Figure 2.1 (b) for $m = 100$. An example of the worst predictions for $m = 60$ using ANN with PCA ($r = 12$) is shown in Figure 2.2. The relative error is 2.3×10^{-3}.

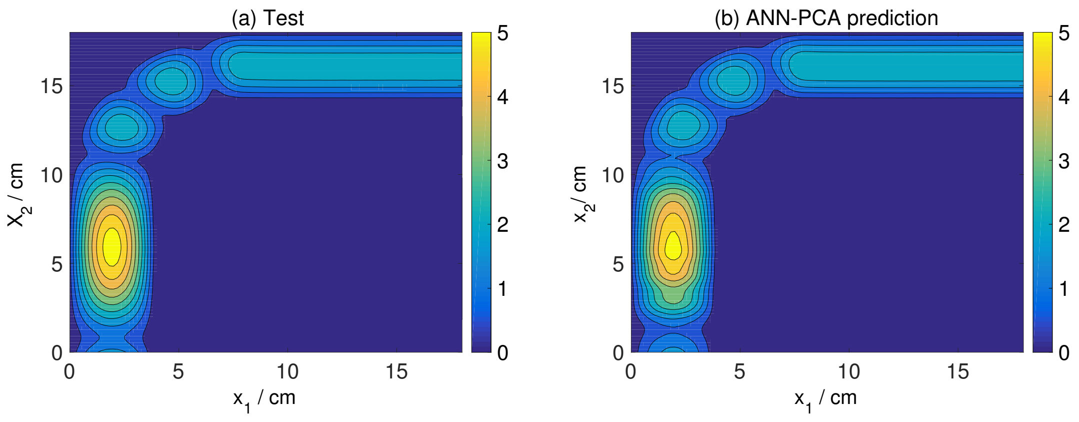


Figure 2.2: Representative examples of prediction using ANN with PCA ($r = 12$, $M = 60$) for the 2D waveguide.

Example 2: 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 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:

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

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:

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

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 $|\boldsymbol{E}(\xi, \eta)|$ for each $\boldsymbol{\xi}^{(i)}$ at locations (ξ_l, η_j), $l, j = 1, \ldots, 500$, were vectorized to form the data points $\mathbf{y}^{(i)} \in \mathbb{R}^d$ used for testing and training.

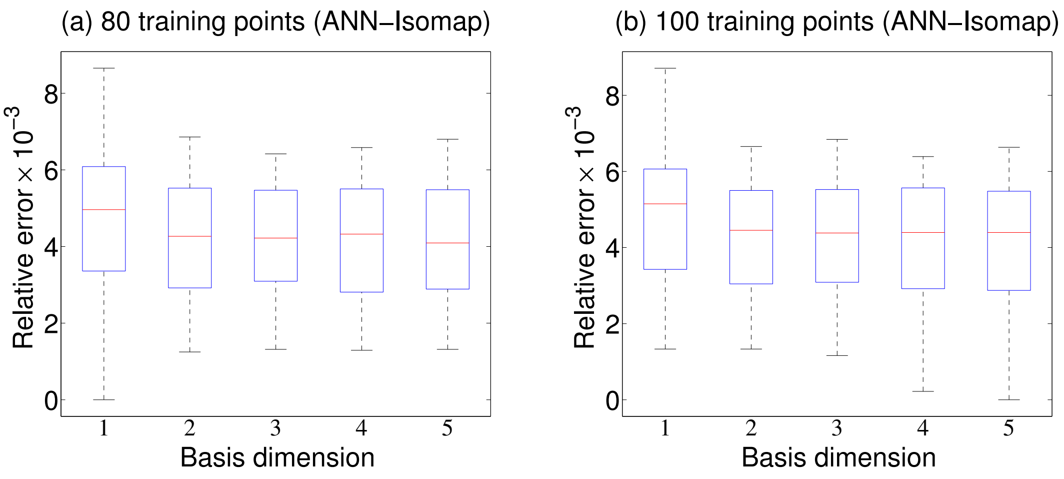


Figure 2.3: Boxplots of the relative errors for different numbers of components (r) using ANN with Isomap ($M = 80$ and 100) in the RCS example.

Results

PCA with both ANN and GPE (method of Higdon et al. [43]) 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 Figures 2.3 (a) and (b), up to 5 components (beyond which no improvements were visible). Figure 2.4 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}.

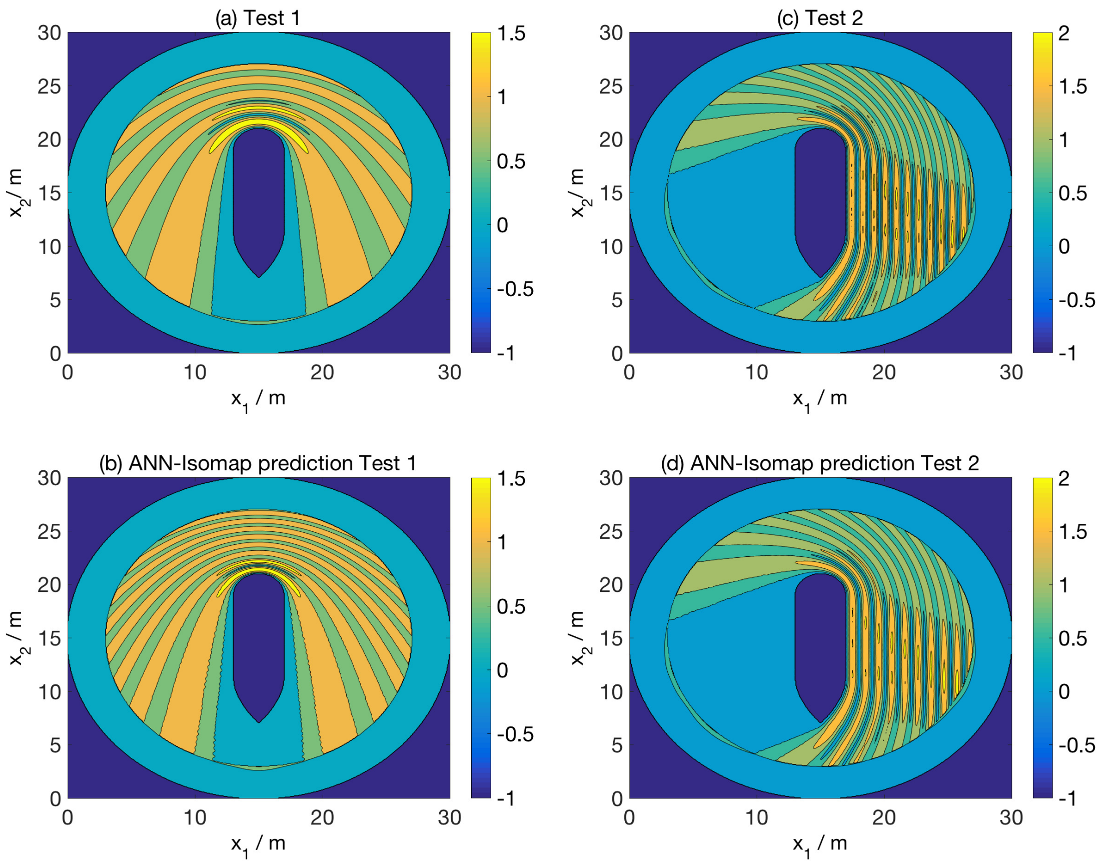


Figure 2.4: Representative examples of the predictions using ANN with Isomap ($r = 5$) and 100 training points in the RCS example.

2.3 Concluding Remarks

In this chapter an efficient approach for developing emulators based on ANN and dimensionality reduction techniques of the output space was presented. The training and prediction time was of the magnitude of a few minutes, which is a significant reduction in computational time in comparison to the approximation of of d outputs simultaneously. The developed emulator can be used for applications such as uncertainty quantification, design optimization, real-time control and inverse parameter estimation. In most cases standard linear dimensionality reduction techniques will fail to produce reasonable results, which was the motivation for developing the techniques used in this chapter. The results were compared to Higdon's method [43] to prove the necessity of non-linear dimensionality reduction techniques when dealing with high-dimensional complex data.

The drawback of the ANNs is the absence of an uncertainty measure (compared to GPE) as there are not placed any assumptions, although this has as a result more accurate predictions. This drawback can be solved by using Bayesian regularization discussed in this chapter. The developed emulator was applied on two different examples, a 2D h-bend waveguide and a 2D radar interaction with a boat (radar cross section), giving accurate results for all the methods tested.

Chapter 3

Gaussian process emulation for probabilistic global sensitivity analysis framework

Abbreviations

$\boldsymbol{\eta}$	representation of computer model as a function
$\boldsymbol{\xi}$	input vector
$\mathbf{y}$	output of the computer model
Q	scalar quantity of interest
$\bar{\boldsymbol{\xi}}$	nominal base point
$s_i(\cdot)$	local sensitivity measure
Δ_i	finite change
c_i	normalizing factor
p	number of levels in EET
μ_i	mean of EE_i
σ_i	standard deviation of EE_i
M	number of base points
$\mathbb{E}[\cdot$	$]expectationoperatorvarianceoperator$
$\mathbb{V}\mathrm{ar}(S_i)$	first order sensitivity index
S_{Ti}	total effect index

$\boldsymbol{\xi}_{\sim i}$	vector of all inputs factors ξ_i
$p(\cdot)$	probability density function
N	number of points in latin hypercube
$\mathbf{v}_i$	basis for $\mathbb{R}^d$
$w(\cdot)$	uncorrelated components of PCA
$\boldsymbol{\Sigma}$	symmetric and positive definite variance-covariance matrix
$\mathbf{y}_r$	reduced dimensional approximation
$c(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}_i)$	covariance function
$\boldsymbol{\theta}_i$	hyperparameters
$\mathbf{d}_i$	data
$p(\mathbf{d}_i \vert \boldsymbol{\theta}_i)$	likelihood of $\mathbf{d}_i$ given $\boldsymbol{\theta}_i$)
$\mathbb{E}_{\boldsymbol{\eta}_r}(\cdot)$	expected value of a quantity with respect to the distribution over $\boldsymbol{\eta}_r$
$\mathbb{V}\mathrm{ar}_{\boldsymbol{\eta}_r}(\cdot)$	variance of a quantity with respect to the distribution over $\boldsymbol{\eta}_r$
J	total number used in MC estimate of the expected value
$\mathcal{R}$	subset of $\mathbb{R}^L$
c_s^p	solid Li concentrations in the positive electrode
c_s^n	solid Li concentrations in the negative electrode
D_j^s	diffusion coefficient of Li in the active material
R_p	i particle radius
a	active surface area

i_2	current density in the electrolyte
κ_2	effective ionic conductivity (using a Bruggemann correction
T	temperature
F	Faraday's constant
R_U	universal gas constant
ϕ_2	electrolyte potential
f_A	mean molar activity coefficient of the electrolyte
c	lithium ion concentration
t_+^0	is the transference number of Li^+
κ_1	effective conductivity of the solid
ϕ_1	solid-phase potential
ϵ_j	volume fraction of electrolyte ($j = p$ for the positive electrode, $j = n$ for the negative electrode and
$j = s$	for the separator)
D_j	effective diffusion coefficient of the Li^+ through the electrolyte
ν_+	number of cations
α_a	charge transfer coefficients for the negative electrode
α_c	charge transfer coefficients for the positive electrode
c_t	total concentration of lithium
k_j	rate constant

η_j	overpotential at an electrode
U_j	equilibrium potential.
SOC_{in}	initial state of charge
R_p	particle diameter in the positive electrode
ϵ_p	porosity of positive electrode
m_t	number of points used for testing

In the context of electrochemical cell models, which are highly complex in their fullest forms, very little attention has been paid to SA. In the majority of cases, formal SA methods are not used; the model is simply run multiple times by varying factors OAT and inspecting the outputs, using *ad-hoc* measures or by employing visualization tools, e.g., [109, 110, 111] for proton exchange membrane fuel cell (PEMFC) models, and [112] for a lithium-ion battery model. Such methods are computationally inefficient and do not take into account interactions between factors. In a small number of studies more rigorous approaches have been used, but almost invariably with highly simplified models. In [113] SA was performed on a equivalent circuit model for a Li-ion battery based on the elementary effect test, and Laoun *et al.* [114] performed a variance-based sensitivity analysis using a simple algebraic PEMFC model.

Applying formal SA methods to complex battery and fuel cells models is computationally burdensome and often not feasible, particularly with brute force Monte Carlo (MC) approaches. In order to overcome this issue an emulator or meta-model can be used to replace the complex model. The emulator itself can be difficult to construct when outputs in high-dimensional spaces are required (e.g., a temperature or electric potential field). If the quantity of interest (QoI), which is derived from the output, is a scalar, an alternative is to use an emulator directly between the inputs and QoI. It may be the case, however, that there are multiple QoIs, in which case it would be ideal to emulate the output, especially when other tasks (e.g., optimization and uncertainty quantification) involving different quantities, including perhaps the original output, are to be performed subsequently.

To address these issues, an approach for SA of a nonlinear Li-ion battery model (full balances for charge and mass) is developed, by employing a Gaussian process emulator based on dimensionality reduction to approximate entire charge-discharge curves. Efficiencies are extracted from the curves in order to perform a SA using two global methods, namely the elementary effect test [94] and a variance-based method. Further more, estimates of the statistics of sensitivity measures in a variance-based approach [115] is derived, extending previous results for the scalar case considered by Oakley and O'Hagan [116] to linear functional QoIs derived from the multi-dimensional output.

3.1 Problem setup and sensitivity analysis

The three main aims of sensitivity analysis are [117]: (i) *factor ranking* (measure the contribution of each factor to variations in the QoI); (ii) *factor screening* to identify those factors that have a negligible effect on the QoI; and (iii) *mapping* to determine regions of the input space that produce extreme QoI values. The main aim determines which SA approach should be used. The procedure in SA is as follows: (i) define a model and identify parameters (factors) of interest as well as QoIs; (ii) assign probability density functions (pdfs) over the inputs or define intervals; (iii) use a sampling method to generate an input matrix in order to generate a matrix of QoIs; and (iv) use local or global methods to quantify the influences of factors on the QoI.

The model is often complex, and typically comprises a system of ordinary or partial differential equations, expressing, e.g., conservation laws. Examples of factors are operating conditions and transport parameters. QoIs could be one or more of the dependent variables or (more often) a scalar quantity derived from these outputs, e.g., the value of a field at a point in a domain or a linear functional such as a spatial average.

Suppose that the model output is $y = \eta(\mathbf{x};\boldsymbol{\xi}) \in \mathcal{F}$ for some function space $\mathcal{F}$, where $\mathbf{x}$ represents, e.g., space, time or space-time and $\boldsymbol{\xi}^T = (\xi_1, \xi_2, ..., \xi_k) \in \mathcal{X} \subset \mathbb{R}^k$ is a vector of input factors; that is, a spatial, temporal or spatio-temporal field, *parameterized* by inputs $\boldsymbol{\xi}$. The computer model (simulator), on the other hand, provides a finite-dimensional approximation of $\eta(\mathbf{x};\boldsymbol{\xi})$, e.g., at discrete times and/or spatial points in a grid (or in terms of a finite basis, e.g., in a finite-element solution). In any case, one may write the simulator output as a vector $\mathbf{y} \in \mathbb{R}^d$, in which the d components of $\mathbf{y}$ represent values of $\eta(\mathbf{x};\boldsymbol{\xi})$ at d different points in a spatial domain or d different times in a temporal grid. The simulator can therefore be considered as a mapping $\boldsymbol{\eta} : \mathcal{X} \to \mathcal{Y} \subset \mathbb{R}^d$ between a feasible input space $\mathcal{X} \subset \mathbb{R}^k$ and an output space $\mathcal{Y}$, i.e., $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$.

When Q is a scalar QoI that is derived from y *via* a *linear* functional $G : \mathcal{F} \to \mathcal{Q} \subset \mathbb{R}$, it can instead be considered as a mapping $F = (G \circ \eta)(\boldsymbol{\xi}) : \boldsymbol{\xi} \mapsto Q$ directly between $\mathcal{X}$ and $\mathcal{Q}$, i.e., $Q = F(\boldsymbol{\xi}) = G(\eta(\mathbf{x};\boldsymbol{\xi}))$. In reality, there is an approximation $q = f(\boldsymbol{\xi})$ of Q, where the functional $f : \mathcal{X} \to \mathcal{Q}$ is derived from a discrete linear functional $g : \mathcal{Y} \to \mathcal{Q}$ that acts on the simulator outputs $\mathbf{y}$, that is $f(\boldsymbol{\xi}) = (g \circ \boldsymbol{\eta})(\boldsymbol{\xi}) = g(\boldsymbol{\eta}(\boldsymbol{\xi}))$. The aim is to develop a SA framework in which the outputs $\mathbf{y}$ are estimated by an emulator. It is also wished to characterize the uncertainty in the sensitivity measures as a result of the uncertainty in $\mathbf{y}$. In the following two sections the SA methods used are described and the construction of the emulator, as well as methods for estimating the uncertainty in the sensitivity measures induced by the emulator. The battery model and SA results are then presented.

3.1.1 The elementary effect test

The simplest approaches to SA are (local) methods that perturb the input factors one at a time (OAT) and rely on visual inspection of the QoIs. A more systematic approach measures the sensitivity of the QoI by estimating the derivative $\partial f/\partial \xi_i$ around a nominal (base) point $\bar{\boldsymbol{\xi}}$, again OAT. Approximation methods are used to calculate the partial derivatives, e.g., one can form finite differences [118]:

$$\begin{aligned} s_i(\bar{\boldsymbol{\xi}}) &\equiv c_i \partial f/\partial \xi_i(\bar{\boldsymbol{\xi}}) \\ &\approx c_i \left[f(\bar{\xi}_1, \ldots, \bar{\xi}_i + \delta_i, \ldots, \bar{\xi}_k) - f(\bar{\xi}_1, \ldots, \bar{\xi}_i, \ldots, \bar{\xi}_k) \right] / \Delta_i \end{aligned} \tag{3.1}$$

where $s_i(\bar{\boldsymbol{\xi}})$ is used as a local sensitivity measure for ξ_i, Δ_i is a finite change in ξ_i and c_i is a normalizing factor to prevent scaling issues. To calculate all the s_i, $k+1$ model evaluations are required. Selecting the Δ_i is largely through trial and error; if they are too small s_i may not provide a useful guide of the effect on q if the model is highly nonlinear. The main weakness of OAT local sensitivity methods is that they provide no information on how the sensitivity to a given ξ_i depends on the values of the other factors.

In order to extend this method to a global analysis, in a way that is less computational expensive than AAT global methods, multiple points $\{\bar{\boldsymbol{\xi}}_j\}_{j=1}^r$ may be used rather than a single $\bar{\boldsymbol{\xi}}$, which leads to the class of *elementary effect methods*. There are several ways to achieve this [119], differing in terms of how the differences are calculated, and the methods for selecting the $\bar{\boldsymbol{\xi}}_j$ and Δ_i. The most well known approach is due to Morris [94]. Suppose that $\mathcal{X}$ is the unit hypercube, and each direction ξ_i is discretized into p levels (points).

The *elementary effect* of ξ_i at $\bar{\boldsymbol{\xi}}_j = (\bar{\xi}_{j,1}, \ldots, \bar{\xi}_{j,k})^T$ is:

$$EE_i^{(j)} \equiv \frac{1}{\Delta_i} \left[f(\bar{\xi}_{j,1}, \ldots, \bar{\xi}_{j,i} + \delta_i, \ldots, \bar{\xi}_{j,k}) - f(\bar{\xi}_{j,1}, \ldots, \bar{\xi}_{j,i}, \ldots, \bar{\xi}_{j,k}) \right], \quad (3.2)$$

where Δ_i is a value in $\{0, 1/(p-1), 2/(p-1) \ldots, 1\}$. Typically, $\Delta_i = \Delta$, $\forall i$. Morris [94] proposed two sensitivities measures for each ξ_i, namely the mean and the standard deviation of the finite distribution F_i over $\{EE_i^{(j)}\}_{j=1}^r$ (an improvement is to use $|EE_i^{(j)}|$, which ensures that in sample mean approximation negative values do not cancel positive values). The mean μ_i of EE_i provides information on the influence of ξ_i, while the standard deviation σ_i measures the degree of interaction with the other factors.

To calculate the statistics for each elementary effect, i.e., μ_i and σ_i, M base points $\{\bar{\boldsymbol{\xi}}_j\}_{j=1}^M$ can be chosen, then construct so-called *trajectories* in $\mathcal{X}$ of $k+1$ points $\bar{\boldsymbol{\xi}}_j \cup \{\boldsymbol{\xi}_{j,n}\}_{n=1}^k$ for each j. Setting $\boldsymbol{\xi}_{j,0} = \bar{\boldsymbol{\xi}}_j$, the trajectory point $\boldsymbol{\xi}_{j,n}$ is obtained by perturbing a randomly chosen factor of $\boldsymbol{\xi}_{j,n-1}$ by $\pm\Delta$ until all factors have been perturbed, but with the property that $\boldsymbol{\xi}_{j,n}$ and $\boldsymbol{\xi}_{j,n-1}$ differ in only one factor. The model is run at every point in each of the trajectories (a total of $M(k+1)$) to obtain $EE_{i,j}$, $i = 1, \ldots k$, $j = 1, \ldots M$, to yield:

$$\mu_i = \frac{1}{M} \sum_{j=1}^{M} EE_{i,j} \quad \text{and} \quad \sigma_i^2 = \frac{1}{M-1} \sum_{j=1}^{M} (EE_{i,j} - \mu_i)^2. \quad (3.3)$$

3.1.2 Variance based SA

A more sophisticated approach to SA, embedded in probability, involves treating the inputs as stochastic variables, which leads to a distribution over the QoI [115, 120, 118]. A variance based first-order effect of each input factor is given by $\mathbb{V}\mathrm{ar}_{\xi_i}(\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i])$, where $\mathbb{E}[\cdot]$ and $\mathbb{V}\mathrm{ar}(\cdot)$ denote expectation and variance operators with respect to the distribution over a subscripted random variable (or with respect to $p(\boldsymbol{\xi})$ if no subscript is present, i.e. $\mathbb{E}[\cdot] \equiv \mathbb{E}_{\boldsymbol{\xi}}[\cdot]$). The quantity $\boldsymbol{\xi}_{\sim i}$ is the vector of all inputs factors *excluding* ξ_i (and similarly for multiple indices). The *first order sensitivity index* (or main effect index) for the input ξ_i is defined as

$$S_i \equiv \mathbb{V}\mathrm{ar}_{\xi_i}(\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i])/\mathbb{V}\mathrm{ar}(q) \quad (3.4)$$

which measures the contribution of the main effect of ξ_i to the total QoI variance. Another measure of sensitivity, defined below, is the *total effect index*, which incorporates interactions between the factors ξ_i:

$$S_{T_i} \equiv \frac{\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[\mathbb{V}\mathrm{ar}_{\xi_i}(q|\boldsymbol{\xi}_{\sim i})]}{\mathbb{V}\mathrm{ar}(q)} = 1 - \frac{\mathbb{V}\mathrm{ar}_{\boldsymbol{\xi}_{\sim i}}(\mathbb{E}_{\xi_i}[q|\boldsymbol{\xi}_{\sim i}])}{\mathbb{V}\mathrm{ar}(q)}. \tag{3.5}$$

The variance-based SA framework can be couched in terms of the decomposition of the variance of q. Suppose $q = f(\boldsymbol{\xi}) \in L^2(\mathcal{X})$ (square integrable functions defined on $\mathcal{X}$) and $\mathcal{X}$ is (without loss of generality) a unit hypercube $\mathcal{X} = \{\boldsymbol{\xi}|0 \leq \xi_i \leq 1; i = 1, \ldots, k\}$. It is also assumed that the factors are independently and uniformly distributed within $\mathcal{X}$, which means that the probability density functions $p(\xi_{i_1}, \ldots, \xi_{i_l}) = \mathbf{1}_{[0,1]^l}$ for $\{i_1, \ldots, i_l\} \subset \{1, \ldots, k\}$, where $\mathbf{1}_A$ is the indicator function on a set A, and the expectation operators $\mathbb{E}_{\boldsymbol{\xi}_{\sim i_1 \ldots i_l}}[\cdot]$ are simple unweighted integrals over $\xi_{i_1}, \ldots, \xi_{i_l}$. The function $f(\boldsymbol{\xi})$ can be decomposed in the following way (Hoeffding decomposition) [121]:

$$f(\boldsymbol{\xi}) = f_0 + \sum_{i=1}^{k} f_i(\xi_i) + \sum_{i=1}^{k} \sum_{j=i+1}^{k} f_{ij}(\xi_i, \xi_j) + \ldots + f_{1\ldots k}(\xi_1, \ldots, \xi_k), \tag{3.6}$$

where f_0 is a constant, $f_i(\xi_i)$ (the main effect of ξ_i) is a function only of ξ_i, $f_{ij}(\xi_i, \xi_j)$ (the interaction) is a function only of ξ_i and ξ_j, and so on. The following condition is imposed [121]:

$$\int_0^1 f_{i_1 i_2 \ldots i_s}(\xi_{i_1}, \xi_{i_2}, \ldots, \xi_{i_s}) d\xi_{i_w} = 0, \tag{3.7}$$

for $1 \leq i_1 < i_2 < \ldots < i_s \leq k$ and $i_w \in \{i_1, i_2, \ldots, i_s\}$. One consequence of this condition and the decomposition is that the summands are orthogonal, that is:

$$\int_{\mathcal{X}} f_{i_1 i_2 \ldots i_m}(\xi_{i_1}, \xi_{i_2}, \ldots, \xi_{i_m}) f_{i'_1 i'_2 \ldots i'_n}(\xi_{i'_1}, \xi_{i'_2}, \ldots, \xi_{i'_n}) d\boldsymbol{\xi} = 0, \tag{3.8}$$

for $\{i_1, i_2, \ldots, i_m\} \neq \{i'_1, i'_2, \ldots, i'_n\}$. Other consequences are that $f_0 = \mathbb{E}[q] = \int_{\mathcal{X}} f(\boldsymbol{\xi}) d\boldsymbol{\xi}$, $f_i = \mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i] - f_0$, $f_{ij} = \mathbb{E}_{\boldsymbol{\xi}_{\sim ij}}[q|\xi_i, \xi_j] - f_i - f_j - f_0$, etc.. By squaring and integrating Eq. (3.6) and using the orthogonality property, one obtains a *decomposition of the total variance*: $V = \mathbb{V}\mathrm{ar}(q) = \int_{\mathcal{X}} f^2(\boldsymbol{\xi})d\boldsymbol{\xi} - f_0^2$:

$$V = \sum_{i=1}^{k} \mathbf{V}_i + \sum_{i=1}^{k} \sum_{j=i+1}^{k} \mathbf{V}_{ij} + \ldots + \mathbf{V}_{1\ldots k}, \tag{3.9}$$

in which

$$\begin{aligned} \mathbf{V}_i &= \mathbb{V}\mathrm{ar}_{\xi_i}(f_i(\xi_i)) = \mathbb{V}\mathrm{ar}_{\xi_i}(\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]) \\ \mathbf{V}_{ij} &= \mathbb{V}\mathrm{ar}_{\xi_i\xi_j}(f_{ij}(\xi_i, \xi_j)) \\ &= \mathbb{V}\mathrm{ar}_{\xi_i\xi_j}(\mathbb{E}_{\boldsymbol{\xi}_{\sim ij}}[q|\xi_i, \xi_j]) - \mathbb{V}\mathrm{ar}_{\xi_i}(\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]) - \mathbb{V}\mathrm{ar}_{\xi_j}(\mathbb{E}_{\boldsymbol{\xi}_{\sim j}}[q|\xi_j]) \end{aligned} \tag{3.10}$$

and so on. The terms $\mathbf{V}_{i_1\ldots i_l}$, $l \leq k$ are called partial variances and it is clear that the main effect indices S_i are simply the partial variances normalized by the total variance. It is also possible to define higher order sensitivity indices by normalizing the $\mathbf{V}_{i_1\ldots i_l}$, e.g. the second-order index $S_{ij} = \mathbf{V}_{ij}/V$, which measures the effect of interactions between ξ_i and ξ_j on q. It is also straightforward to show that the sensitivity indices sum to 1 due to the normalization by V: $\sum_{i=1}^{k} S_i + \sum_{i=1}^{k} \sum_{j=i+1}^{k} S_{ij} + \ldots + S_{1\ldots k} = 1$.

To compute the main and total indices the quasi MC method described in [122] is used. The first step is to generate a matrix $\mathbf{X} = [\xi_{i,j}]$, $i = 1, \ldots, 2k$, $j = 1, \ldots, N$, of N points in the $2k$ hypercube, using a low-discrepancy sequence such as a Latin hypercube. This is done according to the distribution $p(\boldsymbol{\xi})$ over the factors, e.g., $p(\boldsymbol{\xi}) = \mathbf{1}_{[0,1]^k}$ for independent, uniformly distributed factors $\xi_i \sim \mathcal{U}[0, 1]$. $\mathbf{X}$ is then partitioned into a matrix $\mathbf{A} \in \mathbb{R}^{N\times k}$ consisting of the first k columns and a matrix $\mathbf{B} \in \mathbb{R}^{N\times k}$ consisting of the remaining k columns. This provides two independent sets of N samples in the k hypercube. A third matrix $\mathbf{C}_i$ consists of the columns of matrix $\mathbf{B}$ except the i-th column, which is set to the i-th column of $\mathbf{A}$.

The next step is to compute the QoI q by running the model at the selected inputs contained in the sample matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}_i$ to yields vectors $\mathbf{q_A} = f(\mathbf{A})$, $\mathbf{q_B} = f(\mathbf{B})$ and $\mathbf{q}_{\mathbf{C}_i} = f(\mathbf{C}_i)$ ($f(\mathbf{A})$ is used to denote vectorized q values from the set consisting of the rows of $\mathbf{A}$). The indices S_i and S_{T_i} are then calculated from:

$$\begin{aligned} S_i &= \frac{\mathbf{V}_i}{V} = \frac{\mathbb{V}\mathrm{ar}_{\xi_i}(\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i])}{\mathbb{V}\mathrm{ar}(q)} = \frac{\mathbf{q}_{\mathbf{A}}^T\mathbf{q}_{\mathbf{C}_i} - f_0^2}{\mathbf{q}_{\mathbf{A}}^T\mathbf{q}_{\mathbf{A}} - f_0^2} \\ &= \frac{(1/N)\sum_{j=1}^N q_{\mathbf{A},j}q_{\mathbf{C}_i,j} - f_0^2}{(1/N)\sum_{j=1}^N q_{\mathbf{A},j}^2 - f_0^2}, \\ S_{T_i} &= 1 - \frac{\mathbb{V}\mathrm{ar}_{\boldsymbol{\xi}_{\sim i}}(\mathbb{E}_{\xi_i}[q|\boldsymbol{\xi}_{\sim i}])}{\mathbb{V}\mathrm{ar}(q)} = \frac{\mathbf{q}_{\mathbf{B}}^T\mathbf{q}_{\mathbf{C}_i} - f_0^2}{\mathbf{q}_{\mathbf{A}}^T\mathbf{q}_{\mathbf{A}} - f_0^2} \\ &= \frac{(1/N)\sum_{j=1}^N q_{\mathbf{B},j}q_{\mathbf{C}_i,j} - f_0^2}{(1/N)\sum_{j=1}^N q_{\mathbf{A},j}^2 - f_0^2}, \end{aligned} \tag{3.11}$$

where $q_{\mathbf{A},j}$ is the j-th coordinate of $\mathbf{q}_{\mathbf{A}}$ (etc.) and $f_0 = (1/N)\sum_{j=1}^N q_{\mathbf{A},j}$ is the sample mean. This procedure is repeated for each $i = 1, \ldots, k$. The first of Eqs. (3.11) follows from the basic definition:

$$\mathbb{V}\mathrm{ar}_{\xi_i}(\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]) = \int_{[0,1]} \mathbb{E}^2_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]d\xi_i - \int_{[0,1]} (\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]d\xi_i)^2 \tag{3.12}$$

The second term on the r.h.s. is clearly $\mathbb{E}^2[q] = f_0^2$, while the first term can be written as:

$$\begin{aligned} &\int_{[0,1]} \mathbb{E}^2_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]d\xi_i = \\ &\int_{[0,1]^k}\int_{[0,1]^{k-1}} f(\xi_1, \ldots, \xi_i, \ldots, \xi_k) \times f(x_1', \ldots, \xi_i, \ldots, x_k')d\boldsymbol{\xi}d\boldsymbol{\xi}'_{\sim i}, \end{aligned} \tag{3.13}$$

i.e., the expectation over $\boldsymbol{\xi}$ and $\boldsymbol{\xi}'_{\sim i}$ of $f(\xi_1, \ldots, \xi_i, \ldots, \xi_k) \times f(x_1', \ldots, \xi_i, \ldots, x_k')$, which explains the MC estimate in Eq. (3.11). A similar explanation can be given for the S_{T_i} estimate.

The cost of this procedure is $2N$ runs of the model to generate the matrices $\mathbf{A}$ and $\mathbf{B}$, and an additional Nk runs to obtain the QoIs corresponding to the $\mathbf{C}_i$. This give a total of $N(k + 2)$, which is much lower than the cost of brute-force MC estimates of $\mathbb{V}\mathrm{ar}_{\xi_i}(\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i])$ and $\mathbb{V}\mathrm{ar}_{\boldsymbol{\xi}_{\sim i}}(\mathbb{E}_{\xi_i}[q|\boldsymbol{\xi}_{\sim i}])$. The former, e.g., would require $O(N)$ runs ($N \gg k$) to estimate the inner expectation for a fixed ξ_i, which it would be would need to repeat $O(N)$ times to estimate the outer variance, leading to $O(N^2)$ runs for each i. A number of alternatives to the estimates (3.11) have been proposed and details of these estimates can be found in [122].

3.2 Gaussian process emulation of the model outputs

Suppose the given *training points* $\{\mathbf{y}_j\}_{j=1}^m \subset \mathcal{Y}$ are values of $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$ at the *design points* $\{\boldsymbol{\xi}_j\}_{j=1}^m$. Without loss of generality the mean of the training points is taken to be zero. Suppose further that $\mathcal{Y}$ is a low-dimensional manifold embedded in $\mathbb{R}^d$. Specifically, it is assumed that $\mathcal{Y}$ is a linear subspace of $\mathbb{R}^d$ (trivial manifold). This is a perfectly rational assumption to make since the solver (based on physical laws) is deterministic and, therefore, the response manifold dimension will be limited by the dimensionality $k \ll d$ of the input space. A basis is possible to be approximated for the linear subspace $\mathcal{Y}$ by using principal component analysis (PCA) [103]. That is, a linear transformation $\mathbf{w}(\boldsymbol{\xi}) = \mathbf{V}^T\mathbf{y}$ of the training points can be found, in which $\mathbf{V} \in \mathbb{R}^{d\times d}$ has orthogonal columns $\mathbf{v}_i$ (a basis for $\mathbb{R}^d$) and the uncorrelated components $w_i(\boldsymbol{\xi})$ of $\mathbf{w}(\boldsymbol{\xi})$ have decreasing variance with i.

Let $\boldsymbol{\Sigma} = \mathbb{E}[\mathbf{y}\mathbf{y}^T]$ be the symmetric and positive definite variance-covariance matrix, i.e., covariances between coordinates of $\mathbf{y}$. The eigenvalue problem $\boldsymbol{\Sigma}\mathbf{v} = \lambda\mathbf{v}$ yields the $\mathbf{v}_i$ and corresponding positive eigenvalues $\lambda_1 > \cdots > \lambda_d$. The components of a point in this basis satisfy $\mathbb{V}\mathrm{ar}[w_i(\boldsymbol{\xi})] = \lambda_i$ and $\mathbb{E}[w_i(\boldsymbol{\xi})w_j(\boldsymbol{\xi})] = 0$ for $i \neq j$. Any point $\mathbf{y} \in \mathcal{Y}$ can be written in the form

$$\mathbf{y} = \mathbf{V}\mathbf{w}(\boldsymbol{\xi}) = \sum_{i=1}^{d} w_i(\boldsymbol{\xi})\mathbf{v}_i = \sum_{i=1}^{d}(\mathbf{v}_i^T\mathbf{y})\mathbf{v}_i \tag{3.14}$$

and an r-dimensional approximation $\mathbf{y}_r \in \mathcal{Y}_r = \mathrm{span}(\mathbf{v}_1, \ldots, \mathbf{v}_r)$ of $\mathbf{y}$ is given by $\mathbf{y}_r = \mathbf{V}_r\mathbf{w}(\boldsymbol{\xi}) = \sum_{i=1}^r w_i(\boldsymbol{\xi})\mathbf{v}_i$, where $\mathbf{V}_r = [\mathbf{v}_1 \ldots \mathbf{v}_r]$. It can be demonstrated [103] that $\mathbb{E}\left[\|\mathbf{y} - \mathbf{y}_r\|^2\right] = \sum_{i=r+1}^d \lambda_i$, from which a value of r can be selected based on a chosen tolerance.

In practice $\boldsymbol{\Sigma}$ is not known and must be approximated by the sample covariance matrix $\boldsymbol{\Sigma} = (1/m)\mathbf{Y}\mathbf{Y}^T$ where $\mathbf{Y} = [\mathbf{y}_1 \ldots \mathbf{y}_m]$. The PCA basis $\mathbf{v}_1, \ldots, \mathbf{v}_r$ extracted from the sample covariance matrix is strictly valid only for the training points. If there is a sufficient number of training points, however, this basis will provide a good approximation for all points in $\mathcal{Y}$.

The coefficients $w_i(\boldsymbol{\xi}) = \mathbf{v}_i^T\mathbf{y}$ of a point $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$ are assumed to be realizations of scalar, uncorrelated GPs, which means that they are also mutually independent. These coefficients can be emulated in a reduced-dimensional approximation $\mathbf{y}_r = \boldsymbol{\eta}_r(\boldsymbol{\xi}) \equiv \mathbf{V}_r\mathbf{w}_r(\boldsymbol{\xi}) \in \mathcal{Y}_r$ ($\mathbf{w}_r(\boldsymbol{\xi}) = (w_1(\boldsymbol{\xi}), \ldots, w_r(\boldsymbol{\xi}))^T$) of $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi}) = \mathbf{V}\mathbf{w}(\boldsymbol{\xi}) \in \mathcal{Y}$ [43]. The coefficients $w_i(\boldsymbol{\xi})$ are realizations of mutually independent GPs, so can be approximated separately for a chosen value of $\boldsymbol{\xi}$ using GP regression.

Focusing on $w_i(\boldsymbol{\xi})$ for some $i \in \{1, \ldots, r\}$, it is desired to approximate $w_i(\boldsymbol{\xi}) : \mathcal{X} \to \mathbb{R}$ given values of the function at design points $\{\boldsymbol{\xi}_j\}_{j=1}^m$. In GPR, a GP *prior distribution* indexed by $\boldsymbol{\xi} \in \mathcal{X}$ is placed over $w_i(\boldsymbol{\xi})$. For a fixed $\boldsymbol{\xi}$, $w_i(\boldsymbol{\xi})$ is a random variable, whereas $\{w_i(\boldsymbol{\xi})\}_{\boldsymbol{\xi}\in\mathcal{X}}$ is a realization of the GP (a deterministic function of $\boldsymbol{\xi}$). The joint distribution $p(w_i(\boldsymbol{\xi}_1), \ldots, w_i(\boldsymbol{\xi}_m))$ for an arbitrary finite collection of indices $\{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_m\}$ is a multivariate Gaussian. The GP prior is $w_i(\boldsymbol{\xi})|\boldsymbol{\theta}_i \sim \mathcal{GP}\left(0, c(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}_i)\right)$, i.e., $w(\boldsymbol{\xi})$ is distributed according to a GP with an identically zero mean function (without loss of generality the values of $w_i(\boldsymbol{\xi}_j)$ are centred using the mean $\overline{w}_i$, i.e., $w_i(\boldsymbol{\xi}_j) \mapsto w_i(\boldsymbol{\xi}_j) - \overline{w}_i$) and a covariance function $c(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}_i)$, given the values of *hyperparameters* $\boldsymbol{\theta}_i$. In this work, an anisotropic square-exponential covariance function is employed (for all i):

$$c(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}_i) = \theta_0 \exp\left\{-(\boldsymbol{\xi} - \boldsymbol{\xi}')^T \mathrm{diag}(\theta_{i,1}, \ldots, \theta_{i,k})(\boldsymbol{\xi} - \boldsymbol{\xi}')\right\}, \tag{3.15}$$

where $\boldsymbol{\theta}_i = (\theta_{i,0}, \ldots, \theta_{i,k})^T$, in which $\theta_{i,1}, \ldots, \theta_{i,k}$ are the inverse square correlation lengths. The hyperparameters are typically estimated as part of the GPR framework. The given data $\mathbf{d}_i \equiv (w_i(\boldsymbol{\xi}_1), \ldots, w_i(\boldsymbol{\xi}_m))^T$ (from a PCA) is used to update the prior (GP) belief to obtain a new *posterior* GP distribution. A distribution over value of $w_i(\boldsymbol{\xi})$ at each value of $\boldsymbol{\xi}$ is therefore obtained. The updated mean function is the expected value $\mathbb{E}[w_i(\boldsymbol{\xi})]$ across $\boldsymbol{\xi}$, while the updated covariance function yields predictive variances in these estimates, as a consequence of the finite data and assumed model.

The distribution of $\mathbf{d}_i$ given $\boldsymbol{\theta}_i$ (the likelihood) is $p(\mathbf{d}_i|\boldsymbol{\theta}_i) = \mathcal{N}(\mathbf{0}, \mathbf{C}_i)$, with covariance matrix $\mathbf{C}_i = [c(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j; \boldsymbol{\theta}_i)]_{i,j=1}^m$. The joint distribution over

$w_i(\boldsymbol{\xi})$ and $\mathbf{d}_i$ satisfies $p(w_i(\boldsymbol{\xi}), \mathbf{d}_i|\boldsymbol{\theta}_i) = \mathcal{N}(\mathbf{0}, \mathbf{C}'_i(\boldsymbol{\xi}))$, where:

$$\mathbf{C}'_i(\boldsymbol{\xi}) = \left[\begin{array}{c|c} \mathbf{C}_i & \mathbf{c}_i(\boldsymbol{\xi}) \\ \hline \mathbf{c}_i(\boldsymbol{\xi})^T & c(\boldsymbol{\xi}, \boldsymbol{\xi}; \boldsymbol{\theta}_i) \end{array}\right], \tag{3.16}$$

in which $\mathbf{c}_i(\boldsymbol{\xi}) = (c(\boldsymbol{\xi}_1, \boldsymbol{\xi}; \boldsymbol{\theta}_i), \ldots, c(\boldsymbol{\xi}_m, \boldsymbol{\xi}; \boldsymbol{\theta}_i))^T$. The predictive distribution at new inputs $\boldsymbol{\xi} \in \mathcal{X}$ is obtained from the joint distribution $p(w_i(\boldsymbol{\xi}), \mathbf{d}_i|\boldsymbol{\theta}_i)$ by conditioning on $\mathbf{d}_i$ [58]:

$$\begin{aligned} w_i(\boldsymbol{\xi})|\mathbf{d}_i, \boldsymbol{\theta}_i &\sim \mathcal{GP}\left(m'_i(\boldsymbol{\xi}), c'_i(\boldsymbol{\xi}, \boldsymbol{\xi}')\right), \\ m'_i(\boldsymbol{\xi}) &= \mathbf{c}_i(\boldsymbol{\xi})^T \mathbf{C}_i^{-1} \mathbf{d}_i + \overline{w}_i, \\ c'_i(\boldsymbol{\xi}, \boldsymbol{\xi}') &= c(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}_i) - \mathbf{c}_i(\boldsymbol{\xi})^T \mathbf{C}_i^{-1} \mathbf{c}(\boldsymbol{\xi}'), \end{aligned} \tag{3.17}$$

accounting for the mean $\overline{w}_i$ that was subtracted. The hyperparameters can be specified by point estimates [2, 123] such as the maximum log likelihood estimate (MLE):

$$\boldsymbol{\theta}_{i,MLE} = \arg\max_{\boldsymbol{\theta}_i} \left(-\ln|\mathbf{C}_i|/2 - \mathbf{d}_i^T \mathbf{C}_i^{-1} \mathbf{d}_i/2\right). \tag{3.18}$$

This procedure is repeated for each $i = 1, \ldots, r$ to obtain $\mathbf{w}_r(\boldsymbol{\xi}) = (w_1(\boldsymbol{\xi}), \ldots, w_r(\boldsymbol{\xi}))^T$. Using MLE estimates, $\mathbb{E}[\mathbf{w}_r(\boldsymbol{\xi})] = \mathbf{m}'(\boldsymbol{\xi}) \equiv (m'_1(\boldsymbol{\xi}), \ldots, m'_r(\boldsymbol{\xi}))^T$ is obtained. The predicted variance of each coefficient is $\mathbb{V}\mathrm{ar}(w_i(\boldsymbol{\xi})) = c'_i(\boldsymbol{\xi}, \boldsymbol{\xi})$. The model outputs are therefore distributed according to a multivariate GP with mean and cross-covariance matrix function as follows:

$$\begin{aligned} \mathbf{y}_r = \boldsymbol{\eta}_r(\boldsymbol{\xi}) &= \mathbf{V}_r \mathbf{w}_r(\boldsymbol{\xi}) \sim \mathcal{GP}\left(\mathbf{m}_{\mathbf{y}_r}, \mathbf{c}_{\mathbf{y}_r}(\boldsymbol{\xi}, \boldsymbol{\xi}')\right) \\ \mathbf{m}_r(\boldsymbol{\xi}) &= \mathbb{E}[\boldsymbol{\eta}_r(\boldsymbol{\xi})] = \mathbf{V}_r \mathbf{m}'(\boldsymbol{\xi}), \\ \mathbf{c}_r(\boldsymbol{\xi}, \boldsymbol{\xi}') &= \mathbb{C}\mathrm{ov}\left(\boldsymbol{\eta}_r(\boldsymbol{\xi}), \boldsymbol{\eta}_r(\boldsymbol{\xi}')\right) = \mathbf{V}_r \mathrm{diag}(c'_1(\boldsymbol{\xi}, \boldsymbol{\xi}'), \ldots, c'_r(\boldsymbol{\xi}, \boldsymbol{\xi}')) \mathbf{V}_r^T, \end{aligned} \tag{3.19}$$

by virtue of the fact that $\mathbb{C}\mathrm{ov}\left(w_i(\boldsymbol{\xi}), w_j(\boldsymbol{\xi})\right) = 0$ for $i \neq j$.

3.2.1 Probabilistic and multivariate sensitivity analysis

The advantage of using the emulation method described above is that it extends Bayesian GP modelling to multiple (including high-dimensional) outputs in a probabilistic manner, furnishing an explicit distribution over the output

(Eqs. (3.19)) and possibly over QoIs. This means that estimates of the statistics of sensitivity measures can be extracted(with respect to the emulator distribution) using full MC sampling. Take for example the main effect index $S_i = \mathbb{V}\mathrm{ar}_{\xi_i}(\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i])/\mathbb{V}\mathrm{ar}(q)$. The expected value and variance of a quantity with respect to the distribution over $\boldsymbol{\eta}_r$ are denoted $\mathbb{E}_{\boldsymbol{\eta}_r}[\cdot]$ and $\mathbb{V}\mathrm{ar}_{\boldsymbol{\eta}_r}(\cdot)$, respectively. Since $q = f(\boldsymbol{\xi}) = g(\boldsymbol{\eta}(\boldsymbol{\xi}))$, a MC estimate of $\mathbb{E}_{\boldsymbol{\eta}_r}[S_i]$ is given by:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\eta}_r}[S_i] &= \mathbb{E}_{\boldsymbol{\eta}_r}\left[\frac{\mathbb{V}\mathrm{ar}_{\xi_i}(\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[g(\boldsymbol{\eta}(\boldsymbol{\xi}))|\xi_i])}{\mathbb{V}\mathrm{ar}(g(\boldsymbol{\eta}(\boldsymbol{\xi}))}\right] \\
&= \mathbb{E}_{\boldsymbol{\eta}_r}\left[\frac{\mathbb{E}_{\xi_i}\left[\mathbb{E}^2_{\boldsymbol{\xi}_{\sim i}}[g(\boldsymbol{\eta}(\boldsymbol{\xi}))|\xi_i]\right] - \mathbb{E}^2\left[g(\boldsymbol{\eta}(\boldsymbol{\xi}))\right]}{\mathbb{E}\left[g(\boldsymbol{\eta}(\boldsymbol{\xi}))^2\right] - \mathbb{E}^2\left[g(\boldsymbol{\eta}(\boldsymbol{\xi}))\right]}\right] \\
&\approx \frac{1}{J}\sum_{j=1}^{J} \frac{N^{-3}\sum_{l=1}^{N}\left(\sum_{n=1}^{N} g(\boldsymbol{\eta}^{(j)}(\boldsymbol{\xi}_{\sim i}^{(n)}, \xi_i^{(l)}))\right)^2 - N^{-2}\left(\sum_{n=1}^{N} g(\boldsymbol{\eta}^{(j)}(\boldsymbol{\xi}^{(n)})\right)^2}{N^{-1}\sum_{n=1}^{N} g(\boldsymbol{\eta}^{(j)}(\boldsymbol{\xi}^{(n)})^2 - N^{-2}\left(\sum_{n=1}^{N} g(\boldsymbol{\eta}^{(j)}(\boldsymbol{\xi}^{(n)})\right)^2}
\end{aligned} \tag{3.20}$$

where $\boldsymbol{\eta}_r^{(j)}$ is drawn from $p(\boldsymbol{\eta}_r) = \mathcal{GP}\left(\mathbf{m}_r(\boldsymbol{\xi}), \mathbf{c}_r(\boldsymbol{\xi}, \boldsymbol{\xi}')\right)$ and the $\xi_i \sim \mathcal{U}[0,1]$ are independent. The notation $\boldsymbol{\eta}^{(j)}(\boldsymbol{\xi}_{\sim i}^{(n)}, \xi_i^{(l)})$ means that $\boldsymbol{\eta}^{(j)}(\boldsymbol{\xi})$ is evaluated at $\xi_i = \xi_i^{(l)}$, $\boldsymbol{\xi}_{\sim i} = \boldsymbol{\xi}_{\sim i}^{(n)}$ for some $i \in \{1, \dots, k\}$.

In this MC procedure $S_i \equiv S_i(\boldsymbol{\eta})$ is interpreted as a random function of the random vector $\boldsymbol{\eta}$ and samples $\boldsymbol{\eta}^{(j)}(\boldsymbol{\xi})$ (deterministic functions of $\boldsymbol{\xi}$) from $p(\boldsymbol{\eta}_r)$ is obtained to approximate the outer integral. Samples from $\boldsymbol{\xi}_{\sim i} \sim \mathcal{U}[0,1]^{k-1}$ and $\xi_i \sim \mathcal{U}[0,1]$ approximate the inner integrals of $g(\boldsymbol{\eta}(\boldsymbol{\xi})) = f(\boldsymbol{\xi})$, which is a random function of $\boldsymbol{\xi}$. In practice, the samples of $\boldsymbol{\xi}_{\sim i}$ and ξ_i are generated, and then samples from the distributions over $w_i(\boldsymbol{\xi})$, $i = 1, \dots, r$ taken using a Cholesky decomposition (see [124] for precise details) to obtain partial realizations (at the sampled values of $\boldsymbol{\xi}$) of $\mathbf{w}_r(\boldsymbol{\xi})$, from which (partial) realizations of $\boldsymbol{\eta}_r(\boldsymbol{\xi}) = \mathbf{V}_r\mathbf{w}_r(\boldsymbol{\xi})$ can be obtained. In fact, the last step is not necessary since it is possible to work directly with $\mathbf{w}_r(\boldsymbol{\xi})$ to obtain realizations of the QoI $q = g(\boldsymbol{\eta}(\boldsymbol{\xi}))$, i.e., $g(\boldsymbol{\eta}(\boldsymbol{\xi})) = g(\mathbf{V}_r\mathbf{w}_r(\boldsymbol{\xi}))$. $\mathbb{V}\mathrm{ar}_{\boldsymbol{\eta}_r}(S_i)$ is estimated in the same way and the same procedure can be used for S_{T_i} or indeed any other sensitivity measure.

In the scalar output case (the output being the QoI), Oakley and O'Hagan derived semi-analytical expressions for estimating the expectations $\mathbb{E}_{\boldsymbol{\eta}_r}[\cdot]$ and possibly variances $\mathbb{V}\mathrm{ar}_{\boldsymbol{\eta}_r}(\cdot)$ of several sensitivity measures using only a very small number of MC runs (e.g. $O(1)$ *vs.* $O(N)$ for $\mathbb{E}_{\boldsymbol{\eta}_r}[S_i]$ as required in full MC to estimate $\mathbb{V}\mathrm{ar}_{\xi_i}(\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[g(\boldsymbol{\eta}(\boldsymbol{\xi}))|\xi_i]))$ [116]. Equivalent semi-analytical expressions can be established for certain types of scalar QoIs derived from the multivariate output emulator used in this chapter, namely QoIs arising from a linear functional of the output, e.g., the value at a fixed point in space or time or a spatial/temporal average.

Consider a scalar linear functional QoI $Q = F(\boldsymbol{\xi}) = (G \circ \eta)(\boldsymbol{\xi})$ derived from the output of the model $y = \eta(\mathbf{x}; \boldsymbol{\xi}) \in \mathcal{F}$ *via* the linear functional $G : \mathcal{F} \to \mathbb{R}$. For example, considering $G(y) = \int_{\mathcal{R}} \eta(\mathbf{x}; \boldsymbol{\xi}) w(\mathbf{x}) d\mu(\mathbf{x})$, for some measure μ on a compact subset $\mathcal{R}$ of $\mathbb{R}^L$ representing space or time (provided $\eta(\mathbf{x}; \boldsymbol{\xi})$ and $w(\mathbf{x})$ are measurable, $\eta(\mathbf{x}; \boldsymbol{\xi})$ is integrable and $w(\mathbf{x})$ is bounded). To keep matters simple, set $w(\mathbf{x}) \equiv 1/\mu(\mathcal{R})$, use the Lebesgue measure and assume that $\eta(\mathbf{x}; \boldsymbol{\xi})$ is continuous; then the Riemann and Lebesgue integrals coincide and approximations to $G(y)$ by Riemann sums or Gauss quadratures will converge – Newton-Cotes formulae, on the other hand, are not guaranteed to converge even if $\eta(\mathbf{x}; \boldsymbol{\xi})$ is analytic in $\mathcal{R}$ [125].

In reality of course, discret output $\mathbf{y}_r = (y_1, \ldots, y_d)^T = \boldsymbol{\eta}_r(\boldsymbol{\xi})$ approximates $\eta(\mathbf{x}; \boldsymbol{\xi})$ at, say, points $\{\mathbf{x}_l\}_{l=1}^d \subset \mathcal{R}$. Correspondingly, a discrete approximation $g(\mathbf{y})$ of $G(y)$ is defined by a quadrature $g(\mathbf{y}_r) = \mu(\mathcal{R})^{-1} \sum_{j=1}^{d'} b_j y_{l_j}$, where $\{y_{l_j}\}_{j=1}^{d'} \subset \{y_l\}_{l=1}^d$ (approximating $\eta(\mathbf{x}_{l_j}; \boldsymbol{\xi})$, $j = 1, \ldots, d'$) is a subset of the coefficients of $\mathbf{y}_r$ and b_j are quadrature weights. If a Gauss quadrature is used, the points $\mathbf{x}_{l_j}$ are specified and must be included in the design $\{\mathbf{x}_l\}_{l=1}^d$. For ease of presentation, and without loss of generality, a mid-point Riemann sum is used, so that $f(\boldsymbol{\xi}) = g(\boldsymbol{\eta}_r(\boldsymbol{\xi})) = g(\mathbf{y}_r) = d^{-1} \sum_{l=1}^d y_l$.

Rather than a point estimate of $\mathbf{y}_r$ there is in fact a distribution over functions (3.19), which leads to distributions over $q = f(\boldsymbol{\xi})$ and therefore over the sensitivity measures, as a consequence of the uncertainty in the emulator output. The typical sensitivity measures employed are $S_i = \mathbf{V}_i/V = \mathbb{V}\mathrm{ar}_{\xi_i}(\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i])/\mathbb{V}\mathrm{ar}(q)$ and $S_{T_i} = 1 - \mathbb{V}\mathrm{ar}_{\boldsymbol{\xi}_{\sim i}}(\mathbb{E}_{\xi_i}[q|\boldsymbol{\xi}_{\sim i}])/\mathbb{V}\mathrm{ar}(q)$. Oakley and O'Hagan [116] also propose the main effects $f_i = \mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i] - f_0$ as useful graphical summaries of the influences of each variable. Here, semi-analytical estimates of the means and variances of these various quantities are derived, extending the analysis in [116] to multiple output problems.

Recalling relationship (3.19): $\mathbf{y}_r = \sum_{i=1}^r w_i(\boldsymbol{\xi}) \mathbf{v}_i$. Denoting the l-th component of $\mathbf{v}_j$ by v_j^l:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]\right] \\
&= \mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}\left[\frac{1}{d}\sum_{l=1}^{d} y_l\middle|\xi_i\right]\right] \\
&= \tfrac{1}{d}\mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}\left[\textstyle\sum_{l=1}^{d}\sum_{j=1}^{r} w_j(\boldsymbol{\xi})v_j^l\middle|\xi_i\right]\right] \\
&= \frac{1}{d}\mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}\left[\sum_{j=1}^{r} w_j(\boldsymbol{\xi})\sum_{l=1}^{d} v_j^l\middle|\xi_i\right]\right] \\
&= \tfrac{1}{d}\mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}\left[\textstyle\sum_{j=1}^{r} b_j w_j(\boldsymbol{\xi})\middle|\xi_i\right]\right] \\
&= \frac{1}{d}\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}\left[\sum_{j=1}^{r} b_j\mathbb{E}_{\boldsymbol{\eta}_r}[w_j(\boldsymbol{\xi})]\middle|\xi_i\right] \\
&= \tfrac{1}{d}\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}\left[\textstyle\sum_{j=1}^{r} b_j m'_j(\boldsymbol{\xi})\middle|\xi_i\right] \\
&= \frac{1}{d}\sum_{j=1}^{r} b_j\int_{[0,1]^{k-1}} m'_j(\boldsymbol{\xi})d\boldsymbol{\xi}_{\sim i} \\
&= \tfrac{1}{d}\textstyle\sum_{j=1}^{r} b_j T_j(\xi_i)
\end{aligned} \tag{3.21}$$

where $b_j = \sum_{l=1}^{d} v_j^l$ and the functions $T_j(\xi_i)$ are defined by the integrals in the last line. Similarly:

$$\mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}[q]\right] = \frac{1}{d}\sum_{j=1}^{r} b_j\int_{[0,1]^k} m'_j(\boldsymbol{\xi})d\boldsymbol{\xi} = \frac{1}{d}\sum_{j=1}^{r} b_j T'_j \tag{3.22}$$

where T'_j are now constants since the integration is over all input factors. Thus:

$$\mathbb{E}_{\boldsymbol{\eta}_r}[f_i] = \mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]\right] - \mathbb{E}_{\boldsymbol{\eta}_r}[\mathbb{E}[q]] = \frac{1}{d}\sum_{j=1}^{r} b_j\left(T_j(\xi_i) - T'_j\right) \tag{3.23}$$

The integrals in (3.21) and (3.22) can be approximated numerically at a very low computational cost. A slight abuse of notation is introduced and denoted by $\boldsymbol{\xi}_{\sim\mathcal{I}}$ the vector of factors excluding those corresponding to the index set

$\mathcal{I} \subset \{1, \ldots, k\}$ and denote by $\boldsymbol{\xi}_{\mathcal{I}}$, the subset of factors corresponding to $\mathcal{I}$.Thus:

$$\begin{aligned}
&\mathbb{C}\mathrm{ov}_{\boldsymbol{\eta}_r}\left(\mathbb{E}_{\boldsymbol{\xi}_{\sim\mathcal{I}}}[q|\boldsymbol{\xi}_{\mathcal{I}}], \mathbb{E}_{\boldsymbol{\xi}_{\sim\mathcal{J}}}[q|\boldsymbol{\xi}_{\mathcal{J}}]\right)\\
&= \mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim\mathcal{I}}}[q|\boldsymbol{\xi}_{\mathcal{I}}]\,\mathbb{E}_{\boldsymbol{\xi}_{\sim\mathcal{J}}}[q|\boldsymbol{\xi}_{\mathcal{J}}]\right] - \mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\boldsymbol{\xi}_{\mathcal{I}}]\right]\mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim\mathcal{J}}}[q|\boldsymbol{\xi}_{\mathcal{J}}]\right]\\
&= \int_{\boldsymbol{\xi}'_{\sim\mathcal{J}}}\int_{\boldsymbol{\xi}_{\sim\mathcal{I}}} \mathbb{E}_{\boldsymbol{\eta}_r}\left[f(\boldsymbol{\xi})f(\boldsymbol{\xi}')\right] d\boldsymbol{\xi}_{\sim\mathcal{I}}d\boldsymbol{\xi}'_{\sim\mathcal{J}} - \mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim\mathcal{I}}}[q|\boldsymbol{\xi}_{\mathcal{I}}]\right]\mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim\mathcal{J}}}[q|\boldsymbol{\xi}_{\mathcal{J}}]\right]\\
&= \int_{\boldsymbol{\xi}'_{\sim\mathcal{J}}}\int_{\boldsymbol{\xi}_{\sim\mathcal{I}}} \left\{\mathbb{C}\mathrm{ov}_{\boldsymbol{\eta}_r}\left(f(\boldsymbol{\xi}), f(\boldsymbol{\xi}')\right) - \mathbb{E}_{\boldsymbol{\eta}_r}\left[f(\boldsymbol{\xi})\right]\mathbb{E}_{\boldsymbol{\eta}_r}\left[f(\boldsymbol{\xi}')\right]\right\} d\boldsymbol{\xi}_{\sim\mathcal{I}}d\boldsymbol{\xi}'_{\sim\mathcal{J}}\\
&\qquad\qquad\qquad\qquad - \mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim\mathcal{I}}}[q|\boldsymbol{\xi}_{\mathcal{I}}]\right]\mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim\mathcal{J}}}[q|\boldsymbol{\xi}_{\mathcal{J}}]\right]\\
&= \int_{\boldsymbol{\xi}'_{\sim\mathcal{J}}}\int_{\boldsymbol{\xi}_{\sim\mathcal{I}}} \mathbb{C}\mathrm{ov}_{\boldsymbol{\eta}_r}\left(f(\boldsymbol{\xi}), f(\boldsymbol{\xi}')\right) d\boldsymbol{\xi}_{\sim\mathcal{I}}d\boldsymbol{\xi}'_{\sim\mathcal{J}}\\
&= \int_{\boldsymbol{\xi}'_{\sim\mathcal{J}}}\int_{\boldsymbol{\xi}_{\sim\mathcal{I}}} \mathbb{C}\mathrm{ov}_{\boldsymbol{\eta}_r}\left(\frac{1}{d}\sum\nolimits_{j=1}^{r} b_j w_j(\boldsymbol{\xi}), \frac{1}{d}\sum\nolimits_{p=1}^{r} b_p w_p(\boldsymbol{\xi}')\right) d\boldsymbol{\xi}_{\sim\mathcal{I}}d\boldsymbol{\xi}'_{\sim\mathcal{J}}\\
&= \int_{\boldsymbol{\xi}'_{\sim\mathcal{J}}}\int_{\boldsymbol{\xi}_{\sim\mathcal{I}}} \sum\nolimits_{j=1}^{r}\sum\nolimits_{p=1}^{r} b_j b_p \mathbb{C}\mathrm{ov}_{\boldsymbol{\eta}_r}\left(w_j(\boldsymbol{\xi}), w_p(\boldsymbol{\xi}')\right) d\boldsymbol{\xi}_{\sim\mathcal{I}}d\boldsymbol{\xi}'_{\sim\mathcal{J}}\\
&= \frac{1}{d^2}\int_{\boldsymbol{\xi}'_{\sim\mathcal{J}}}\int_{\boldsymbol{\xi}_{\sim\mathcal{I}}} \sum\nolimits_{j=1}^{r} b_j^2 c_j(\boldsymbol{\xi}, \boldsymbol{\xi}') d\boldsymbol{\xi}_{\sim\mathcal{I}}d\boldsymbol{\xi}'_{\sim\mathcal{J}} = U(\boldsymbol{\xi}_{\mathcal{I}}, \boldsymbol{\xi}_{\mathcal{J}})
\end{aligned} \tag{3.24}$$

in which the last step follows from the mutual independence of the w_i, and the posterior covariances $c_j(\boldsymbol{\xi}, \boldsymbol{\xi}')$ are given in Eqs. (3.17). Now, $\mathbb{V}\mathrm{ar}_{\boldsymbol{\eta}_r}(f_i) = \mathbb{E}_{\boldsymbol{\eta}_r}[\mathbb{E}^2_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]] - 2\mathbb{E}_{\boldsymbol{\eta}_r}[\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]\mathbb{E}[q]] - \mathbb{E}_{\boldsymbol{\eta}_r}[\mathbb{E}^2[q]] - \mathbb{E}^2_{\boldsymbol{\eta}_r}[f_i]$ in which the last term on the right hand side is already known. The first to third terms are calculated by using the definition of covariance and Eq. (3.24) with $(\mathcal{I}, \mathcal{J}) = (\{i\}, \{i\})$, $(\{i\}, \emptyset)$ and $(\emptyset, \emptyset)$, together with the previous expressions for $\mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]\right]$ and $\mathbb{E}_{\boldsymbol{\eta}_r}[\mathbb{E}[q]]$. The integrals in Eq. (3.24) are again cheap to evaluate. The means of all the f_i, evaluated separately for selected values of ξ_i in $[0, 1]$, can be combined on a single plot together with standard deviation bounds [116] to identify the strengths of influences of the factors.

Next the partial variances $\mathbf{V}_i$ are considered, noting that $\mathbb{E}_{\boldsymbol{\eta}_r}[\mathbf{V}_i] = \mathbb{E}_{\boldsymbol{\eta}_r}[\mathbb{V}\mathrm{ar}_{\xi_i}(\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i])] = \mathbb{E}_{\boldsymbol{\eta}_r}[\mathbb{E}_{\xi_i}[\mathbb{E}^2_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]] - \mathbb{E}^2_{\xi_i}[\mathbb{E}_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]]] = \mathbb{E}_{\boldsymbol{\eta}_r}[\mathbb{E}_{\xi_i}[\mathbb{E}^2_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]]] - \mathbb{E}_{\boldsymbol{\eta}_r}[\mathbb{E}^2[q]]$, the second term of which is already known. The first term is evaluated as follows:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\eta}_r}\left[\mathbb{E}_{\xi_i}[\mathbb{E}^2_{\boldsymbol{\xi}_{\sim i}}[q|\xi_i]]\right] = \mathbb{E}_{\boldsymbol{\eta}_r}\left[\int_{[0,1]}\left(\int_{[0,1]^{k-1}} f(\boldsymbol{\xi}^*)d\boldsymbol{\xi}^*_{\sim i}\int_{[0,1]^{k-1}} f(\boldsymbol{\xi})d\boldsymbol{\xi}_{\sim i}\right)d\xi_i\right]\\
&= \int_{[0,1]}\int_{[0,1]^{k-1}}\int_{[0,1]^{k-1}} \mathbb{E}_{\boldsymbol{\eta}_r}\left[f(\boldsymbol{\xi})f(\boldsymbol{\xi}^*)\right] d\boldsymbol{\xi}^*_{\sim i}d\boldsymbol{\xi}_{\sim i}d\xi_i\\
&= \int_{[0,1]}\int_{[0,1]^{k-1}}\int_{[0,1]^{k-1}} \{\mathbb{C}\mathrm{ov}_{\boldsymbol{\eta}_r}\left(f(\boldsymbol{\xi}), f(\boldsymbol{\xi}^*)\right) - \mathbb{E}_{\boldsymbol{\eta}_r}[f(\boldsymbol{\xi})]\mathbb{E}_{\boldsymbol{\eta}_r}[f(\boldsymbol{\xi}^*)]\} d\boldsymbol{\xi}^*_{\sim i}d\boldsymbol{\xi}_{\sim i}d\xi_i\\
&= \frac{1}{d^2}\sum_{j=1}^{r}\int_{[0,1]}\int_{[0,1]^{k-1}}\int_{[0,1]^{k-1}} b_j\left\{b_jc_j(\boldsymbol{\xi},\boldsymbol{\xi}^*) - \sum_{p=1}^{r} b_pw_j(\boldsymbol{\xi})w_p(\boldsymbol{\xi}^*)\right\} d\boldsymbol{\xi}^*_{\sim i}d\boldsymbol{\xi}_{\sim i}d\xi_i
\end{aligned} \tag{3.25}$$

in which $\boldsymbol{\xi}^* = (\xi_1^*, \ldots, \xi_i, \ldots, \xi_k^*)^T$. These integrals are readily and cheaply approximated numerically. A first order Taylor expansion yields $\mathbb{E}_{\boldsymbol{\eta}_r}[S_i] = \mathbb{E}_{\boldsymbol{\eta}_r}[\mathbf{V}_i/V] = \mathbb{E}_{\boldsymbol{\eta}_r}[\mathbf{V}_i]/\mathbb{E}_{\boldsymbol{\eta}_r}[V]$, from which the main effect indices can be approximated.

Another feature of this method is that it is possible to investigate the *sensitivity of multivariate outputs under uncertainty* to the inputs by *separately* (by virtue of their independence) ranking the sensitivity indices for the coefficients $w_i(\boldsymbol{\xi})$, $i = 1, \ldots, r$, using the procedures described above (i.e., $q = w_i(\boldsymbol{\xi})$). Since the contributions of these coefficients decay with i, one may only need to investigate the first one or two.

3.3 Li-ion battery model

A Li-ion battery comprised of a $LiMn_2O_4$ positive electrode and a graphite Li_xC_6 porous negative electrode is cinsidered. The electrolyte consists of a non-aqueous carbonate solvent mixture and a lithium salt $LiPF_6$ in a mixture of ethylene carbonate (EC) and diethyl carbonate (DEC) (1:1 ratio) dispersed in an inert polymer matrix to provide mechanical support. The domain is 1-d (direction x) and the positive and negative current collectors are located at $x = 0$ and $x = L$, respectively. The model is based on that of Newman *et al.* [126, 127], which includes mass and charge balances in the solid and liquid phases. The intercalation of Li in the solid phases of the electrodes is described by a mass balance with diffusion in a pseudo dimension R (into spherical solid particles). The solid Li concentrations in the positive and negative electrodes,

c_s^p and c_s^n respectively, are given by:

$$\frac{\partial c_j^s}{\partial t} = \frac{1}{R^2}\frac{\partial}{\partial r}\left(R^2 D_j^s \frac{\partial c_j^s}{\partial R}\right), \tag{3.26}$$

where D_j^s ($j = p$ for the positive electrode and $j = n$ for the negative electrode) is the diffusion coefficient of Li in the active material. The boundary conditions are $\partial c_j^s/\partial R|_{R=0} = 0$ and $-D_j^s \partial c_j^s/\partial R|_{R=R_p} = (1/aF)\partial i_2/\partial x$, where R_p is the particle radius, a is the specific active surface area and i_2 is the current density in the electrolyte, given by:

$$i_2 = -\kappa_2 \frac{\partial \phi_2}{\partial x} + \frac{\kappa_2 R_U T}{F}(1 - t_+^0)\left(1 + \frac{\partial \ln f_A}{\partial \ln c}\right)\frac{\partial \ln c}{\partial x}, \tag{3.27}$$

where κ_2 is the effective ionic conductivity (using a Bruggemann correction, i.e. multiply the free space value by the volume fraction of electrolyte raised to the power of 3/2), T is the temperature, F is Faraday's constant, R_U is the universal gas constant, ϕ_2 is the electrolyte potential, f_A is the mean molar activity coefficient of the electrolyte, c is the lithium ion (Li^+) concentration and t_+^0 is the transference number of Li^+. The solid phase current density i_1 is governed by Ohm's law: $i_1 = -\kappa_1 \partial \phi_1/\partial x$, where κ_1 is the effective conductivity of the solid (using a Bruggemann correction) and ϕ_1 is the solid-phase potential. Charge conservation demands that $i_1 + i_2 = I$, for a total current density I. The boundary conditions for the potentials (galvanostatic) are:

$$-\kappa_1 \frac{\partial \phi_1}{\partial x}\bigg|_{x=0} = -\kappa_1 \frac{\partial \phi_1}{\partial x}\bigg|_{x=L} = I, \quad -\kappa_2 \frac{\partial \phi_2}{\partial x}\bigg|_{x=0} = -\kappa_2 \frac{\partial \phi_2}{\partial x}\bigg|_{x=L} = 0, \tag{3.28}$$

while the electronic charge fluxes are zero and the ionic charge fluxes are continuous at the separator interfaces. The mass balance for Li^+ is:

$$\epsilon_j \frac{\partial c}{\partial t} = \frac{\partial}{\partial x}\left(\epsilon_j D_j \frac{\partial c}{\partial x}\right) - \frac{(1 - t_+^0)}{\nu_+ F}\frac{\partial i_2}{\partial x}, \tag{3.29}$$

where ϵ_j is the volume fraction of electrolyte ($j = p$ for the positive electrode, $j = n$ for the negative electrode and $j = s$ for the separator), D_j is the effective diffusion coefficient of the Li^+ through the electrolyte, and ν_+ is the number of cations into which a mole of electrolyte dissociates. The flux at both ends of the cell ($-\epsilon_j D_j \partial c/\partial x$ by virtue of the zero ionic charge flux) is set to zero. The current density is given by the Butler-Volmer equation:

$$\frac{\partial i_j}{\partial x} = -aFk_j(c)^{\alpha_{a,j}}(c_t - c_j^s)^{\alpha_{a,j}}(c_j^s)^{\alpha_{c,j}} \left\{ \exp\left(\frac{\alpha_a F \eta_j}{R_U T}\right) - \exp\left(\frac{\alpha_a F \eta_j}{R_U T}\right) \right\}, \tag{3.30}$$

where α_a and α_c are the charge transfer coefficients for the negative ($j = n$) and positive ($j = p$) electrode reactions, c_t is the total concentration of lithium, k_j is the rate constant for the relevant reaction and $\eta_j = \phi_1 - \phi_2 - U_j$ is the overpotential at the relevant electrode, in which U_j is the corresponding equilibrium potential.

3.4 Results and discussion

3.4.1 Emulator performance and selection

The Li-ion battery model was implemented in COMSOL Multiphysics[1]. A total of 500 simulations were performed by varying the initial state of charge SOC$_{in}$ (initial c_n^s divided by c_t), the particle diameter in the positive electrode R_p and the positive electrode porosity ϵ_p.

$\boldsymbol{\xi} = (\text{SOC}_{in}, R_p, \epsilon_p)^T \in \mathcal{X} \subset \mathbb{R}^k$ was set as the input, with the $k = 3$ factors given by the components. The inputs for the 500 simulations were selected using a Sobol sequence (pseudo-uniform). A current pulse $i(t)$ consisting of 12 s of 120 A discharge, followed by 12 s of relaxation (0 A load), and then 12 s of 120 A charge was simulated (galvanostatic operation). The output was taken to be the cell voltage $E_{cell}(t)$[V] (a function of time) at 0.5 s intervals, yielding a total of 73 values at $t = 0, 0.5, 1, \ldots, 35.5, 36$ s. These values were vectorized to form the outputs:

$$\mathbf{y}^T = \boldsymbol{\eta}(\boldsymbol{\xi}) = (E_{cell}(0), E_{cell}(0.5), \ldots, E_{cell}(35.5), E_{cell}(36)) \in \mathcal{Y} \subset \mathbb{R}^d \tag{3.31}$$

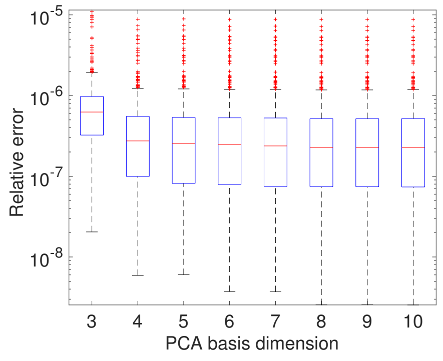
where $\boldsymbol{\eta}$ as before represents the simulator (battery model) and $d = 73$. The

[1]For details of the default parameter values and the implementation please refer to `https://www.comsol.com/models/batteries-and-fuel-cells-module` ('1D Lithium-Ion Battery Model for Internal Resistance and Voltage Loss Determination'). Last accessed 28 September 2017.

first 100 outputs (and corresponding inputs) were reserved for training the emulator and the remaining $m_t = 400$ were used for testing the emulator. The training data set is denoted $\{(\boldsymbol{\xi}_j, \mathbf{y}_j)\}_{j=1}^{m}$, as before, and the test data set is denoted $\{(\boldsymbol{\xi}_j^*, \mathbf{y}_j^*)\}_{j=1}^{m_t}$. The QoI in this example is the energy efficiency, defined as:

$$q = f(\boldsymbol{\xi}) = \frac{\int_{[d]} i(t)E_{cell}(t)dt}{\int_{[c]} i(t)E_{cell}(t)dt} \in \mathcal{Q} = [0,1] \tag{3.32}$$

in which $[d]$ ($[c]$) is the discharge (charge) time interval. q measures the energy recovered during discharge as a proportion of the energy used to charge the battery to an equivalent initial SOC.



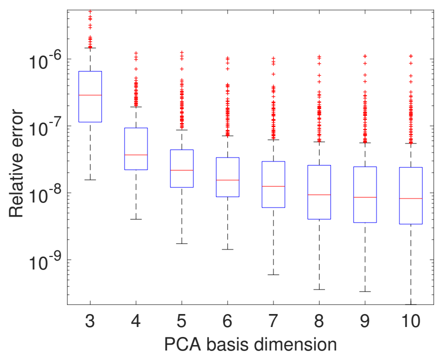


Figure 3.1: Boxplots of the emulator errors on the test set $\{(\boldsymbol{\xi}_j^*, \mathbf{y}_j^*)\}_{j=1}^{m_t}$ using the training set $\{(\boldsymbol{\xi}_j, \mathbf{y}_j)\}_{j=1}^{m}$ with $m = 50$ (left) and $m = 100$ (right).

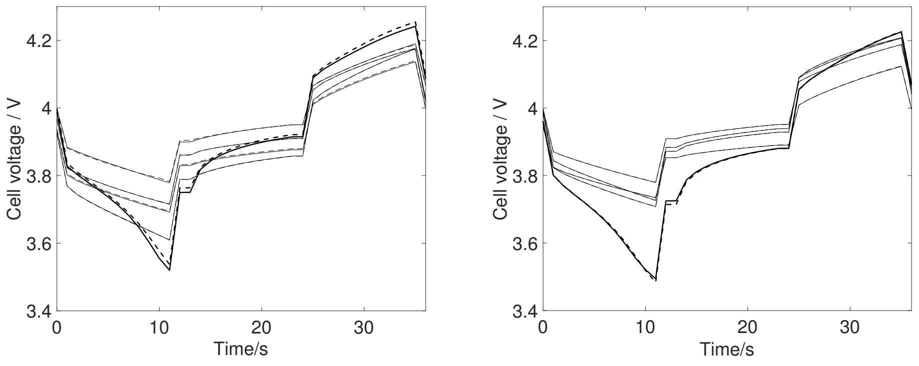


Figure 3.2: Example predictions $\overline{\mathbf{y}}^*_{r,j}$ (dashed lines) of the cell voltage during the discharge-charge cycle $\mathbf{y}^*_j$ (solid lines) using $r = 5$ for $m = 50$ (left) and $r = 10$ for $m = 100$ (right). The worst case predictions (highest ϵ^*) are the thick lines and 4 further examples are shown for each value of m.

Figure 3.1 shows Tukey boxplots of the relative errors on the test set $\{(\boldsymbol{\xi}^*_j, \mathbf{y}^*_j)\}_{j=1}^{m_t}$ using the training set $\{(\boldsymbol{\xi}_j, \mathbf{y}_j)\}_{j=1}^{m}$ with $m = 50$ and $m = 100$ training points for an increasing number of PCA basis dimensions r. The errors were defined as follows $\epsilon^* = ||\overline{\mathbf{y}}^*_{r,j} - \mathbf{y}^*_j||/||\mathbf{y}^*_j||$, in which $\overline{\mathbf{y}}^*_{r,j}$ is the mean GP prediction of $\mathbf{y}^*_j$ using Eq. (3.19). It can be seen that the emulator error decreases with increasing r, plateauing at $r = 5$ for $m = 50$ and $r = 10$ for $m = 100$. It is also evident that increasing the number of training points leads to more accurate predictions. For both values of m, example predictions of the cell voltage during the discharge-charge cycle are shown in Figure 3.2. The worst case predictions (highest ϵ^*) are shown, alongside 4 further examples for each value of m. It is clear that both values of m capture the trends well, and are quantitatively accurate, even in the worst case. For $m = 100$, the predictions are particularly accurate, so for the SA $m = 100$ and $r = 10$ were selected.

3.4.2 Sensitivity analysis

The SA was performed using the SAFE package developed by Pianosi *et al.* [128]. For the variance-based method a uniform distribution was placed on the factors and points $\mathbf{X} = [\xi_{i,j}]$, $i = 1, \ldots, 2k$, $j = 1, \ldots, N$ ($N = 5000$) in the $2k$

hypercube using a Latin hypercube design were sampled. The physical ranges were $0.1 \leq \epsilon_p \leq 0.4$, $0.5 \leq R_p[\mu\text{m}] \leq 2$ and $0.4 \leq \text{SOC}_{in} \leq 0.6$, and the factors were scaled to obtain $\mathcal{X} = [0,1]^3$. The sampled inputs were used to produce the three input matrices $\mathbf{A} \in \mathbb{R}^{N\times k}$, $\mathbf{B} \in \mathbb{R}^{N\times k}$ and $\mathbf{C}_i \in \mathbb{R}^{N\times k}$, $i = 1,\ldots,k$, from which the QoI values $\mathbf{q}_{\mathbf{A}} = f(\mathbf{A})$, $\mathbf{q}_{\mathbf{B}} = f(\mathbf{B})$ and $\mathbf{q}_{\mathbf{C}_i} = f(\mathbf{C}_i)$ were extracted. The S_i and S_{T_i} were then calculated from Eqs. (3.11).

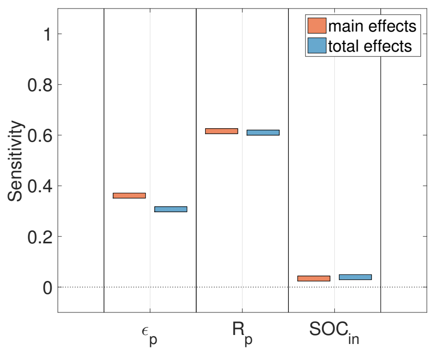


Figure 3.3: Box plots of the main and total effects for the energy efficiency.

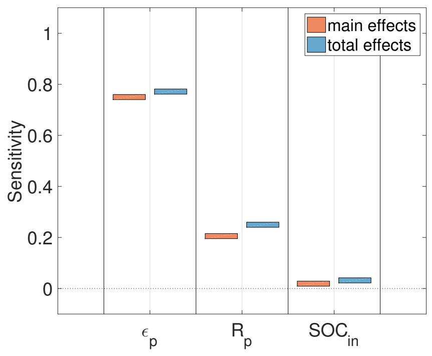


Figure 3.4: Box plots of the main and total effects for the cell voltage drop during discharge.

Figure 3.3 presents both the main and total effects for the three factors. As expected, the particle size and porosity are the most influential, while the initial SOC mainly affects the open-circuit potential (slight shift in the charge-discharge curve up or down) so has relatively little influence on q. The porosity determines the effective ionic conductivity (the volume fraction of electrolyte is ϵ_p) and since the ohmic loss is predominantly suffered in the ionic phase, ϵ_p has a major influence on the internal resistance. Moreover, the reaction rate depends upon the concentration (per unit volume of the electrode) of Li^+ according to the Butler-Volmer law (3.30), so a restricted supply of Li^+ in the positive electrode will lead to a large concentration overpotential for a fixed current (the overpotential in (3.30) must increase as c decreases in order to maintain a fixed left hand side, i.e, applied current density). The particle radius determines the level of mass transport resistance for the solid Li (which has to diffuse through the particle to react at $R = R_p$) as well as the specific surface area for reaction (smaller particles lead to higher specific areas). Thus, increasing the particle radius will lead to a higher concentration overpotential and, therefore, a deterioration in performance.

The ordering here is specifically for the energy efficiency (for a constant current charge-discharge cycle the energy efficiency is simply the average cell voltage during discharge divided by the average cell voltage during charge) so it is dangerous to draw too many conclusions. For another QoI, such as the voltage drop during discharge, ϵ_p has the greatest influence, followed by R_p and lastly SOC_{in}, as shown in Figure 3.4. The combined effect of an increased Ohmic drop and a higher concentration overpotential on the total polarization caused by a lower ϵ_p outweighs the effect of an increased concentration overpotential caused by a smaller R_p.

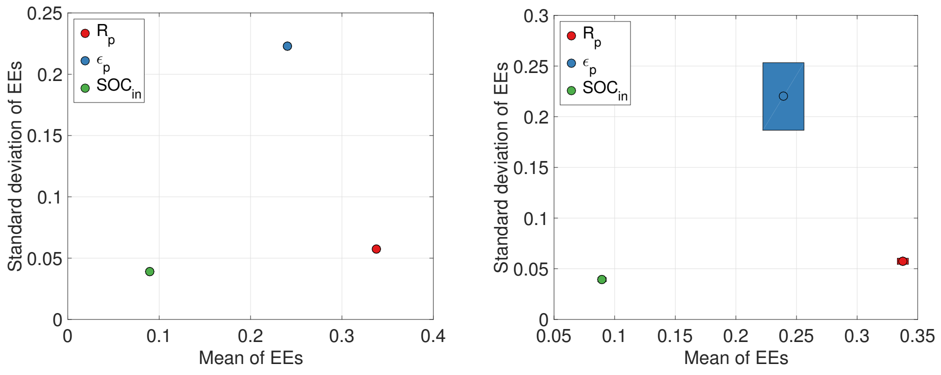


Figure 3.5: Means and standard deviations of the elementary effects with and without confidence intervals in the case of the energy efficiency.

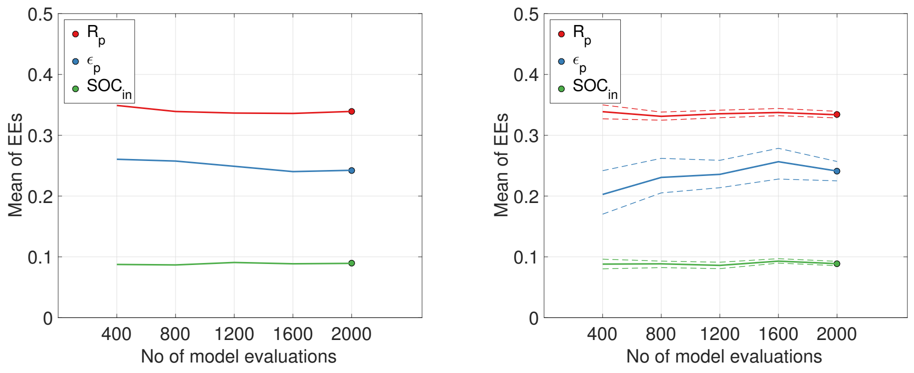


Figure 3.6: Convergence of the elementary effects for different numbers of model evaluations without confidence bounds (left) and with 95% confidence intervals (right) in the case of the energy efficiency.

The next results are for the elementary effect test (EET) on the same data set. A uniform distribution was selected for the three factors, which were again scaled to yield $\mathcal{X} = [0,1]^3$. A major difference between the variance-based method and the EET is the sampling strategy. The EET is highly efficient, requiring only $M(k+1)$ model evaluations *vs.* $N(k+2)$ to calculate the main effect indices; in the results above, $N(k+2) = 5000 \times (3+2) = 25000$, much higher than typical values of $M(k+1)$. Figure 3.5 shows the mean and standard deviation estimates using $M = 100$ trajectories $(100 \times (3+1) = 400$ model runs), both with and without confidence bounds. The confidence bounds were obtained using bootstrapping [129], which consists of re-sampling the base points with replacement to produce P copies of the trajectories and for each of the P copies to use the EET to estimate μ_i and σ_i. This provides empirical distributions over μ_i and σ_i from which means and confidence intervals (CIs) (such as the 95% CIs in Figure 3.5) can be estimated.

The ranking of the inputs is the same as in the variance-based method. The trends in the means of the μ_i both with and without confidence intervals are depicted in Figure 3.6 for increasing M. The means stabilize at around $M = 500$ (2000 model runs). There are small but noticeable fluctuations in the mean for ϵ_p around the value 0.25 even at much higher values of M but this behaviour is stable. The cause is the broad range of ϵ_p in comparison with the other factors and, therefore, the relatively small number of samples.

Moving to a higher value of M the confidence intervals shrink, suggesting greater accuracy in the predictions. The ranking, however, is accurate even for very low numbers of M, which shows that the EET is more efficient than the variance based method.

Further indication of this is provided in Figure 3.7, which shows the EET predictions for different M in the case of the voltage drop during charge. The results are again consistent with the variance-based method (there are similar fluctuations in the mean for ϵ_p). Although time cost is not an issue for the emulator, which provides extremely rapid predictions (on the order of a few seconds for 2000 predictions), in cases where a full simulator is used the much lower number of model runs for the EET represents an enormous advantage.

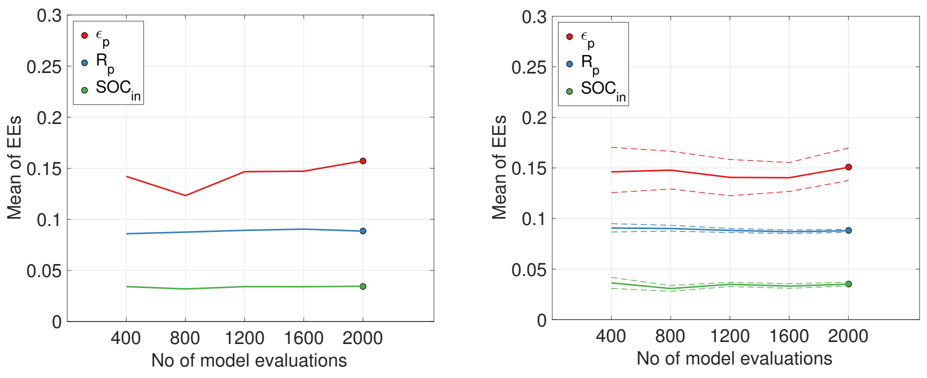


Figure 3.7: Convergence of the elementary effects for different numbers of model evaluations without confidence bounds (left) and with 95% confidence intervals (right) in the case of the voltage drop during discharge.

To investigate the *sensitivity of the charge-discharge curve*, the main and total effects of the PCA coefficients $w_i(\boldsymbol{\xi})$ are examined using the same Latin hypercube design and $N = 5000$ (e.g., η_E is replaced with $w_i(\boldsymbol{\xi})$, $i \in \{1, \ldots, r\}$). The results are depicted in Figure 3.8 for w_1 and w_2. Figure 3.9 shows an example (from the test set) of the contributions from the PCA eigenvectors ($w_i\mathbf{v}_i$, up to $i = 4$) towards the final (mean centred) voltage profile. The first two contributions can be seen to have by far the most influence. The sensitivities of the coefficients w_1 and w_2 are highest for ϵ_p and R_p, with roughly equal contributions from each, while higher-order coefficients (w_3 and w_4) were more heavily influenced by SOC_{in}.

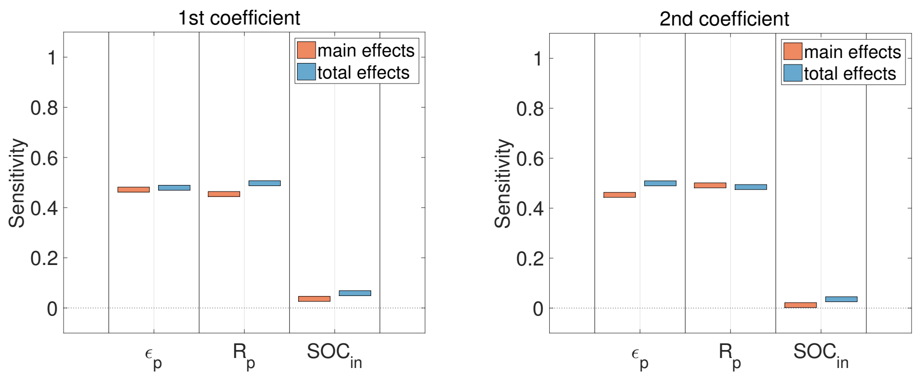


Figure 3.8: Main and total effects for the first two PCA coefficients (for the cell voltage curve).

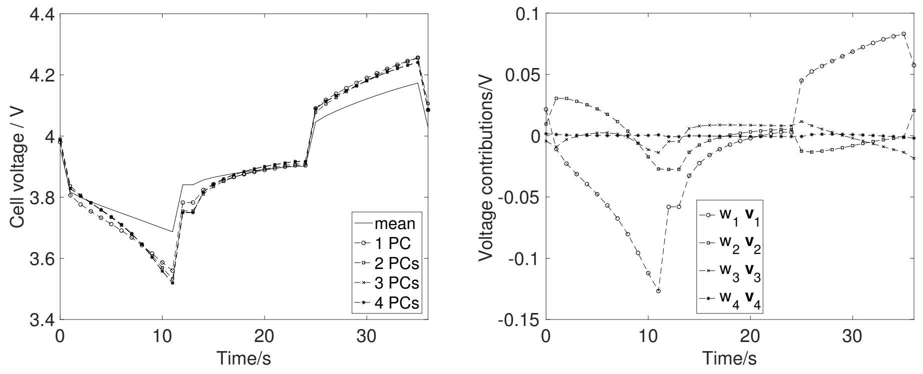


Figure 3.9: An example of the contributions from the PCA eigenvectors ($w_i\mathbf{v}_i$) towards the final (mean centred) voltage profile. In the left-hand figure, $w_i\mathbf{v}_i$ is successively added to the mean.

3.5 Concluding remarks

SA is often computationally unfeasible with complex computer models. In such cases emulators, of varying degrees of sophistication, can be employed. Quantifying the uncertainty in the emulator predictions is desirable, but this is only achievable for certain approaches. For multivariate outputs (especially in high dimensional spaces), SA under uncertainty is especially challenging, even when the QoI is a scalar.

In this chapter a GP emulation approach is proposed for performing SA under uncertainty when the model output is multivariate (possibly in a high-dimensional space). An example for a Li-ion battery is presented, revealing that the method is efficient and accurate. It is also able to perform a probabilistic SA on scalar QoIs and also on the output itself by ranking sensitivity measures for the random principal coefficients. Two different methods (EET and VBSA) are presented for achieving this aim, either through full MC sampling or by using semi-analytical expressions that are extensions of those derived by Oakley and O'Hagan [116]. The EET needs less number of model evaluations to provide accurate results as it belongs to OAT sampling method, while VBSA needs more number of model evaluations in order to extract higher order statistics. It also shown in this chapter that despite the more model evaluations VBSA usually needs, adapting the GPE method developed here this limitation can be overcome.

Chapter 4

Reduced order modeling of parameter dependent, linear and non linear dynamic partial differential equation models

In this chapter Proper Orthogonal Decomposition (POD) is described and used for the solution of dynamic parametrized PDEs (linear and non-linear). As mentioned in the introduction, POD is one of the most widely used reduced order method techniques for approximating the solutions of PDEs. The basic challenge of POD is the approximation of the basis for new unseen parameters along with dealing with nonlinear surfaces. To overcome the issue with nonlinearities the discrete empirical interpolation method (DEIM) is extended and described in detail in section 4.2. The outline of the chapter is as follows, ROMs for PDEs are presented at first following by the basic emulation strategy that has been used. Two examples are presented to compare the results with the mainstream global basis approach.

This chapter is based on the publication [130]: A. A. Shah, W. W. Xing, V. Triantafyllidis, *Reduced-order modelling of parameter-dependent, linear and nonlinear dynamic partial differential equation models*, Volume 473, issue 2200 in Proceedings of the Royal Society A, 2017

Abbreviations

$\boldsymbol{\eta}$	representation of computer model as a function
$\boldsymbol{\xi}$	input vector of parameters
$\mathbf{x}$	points in a regular domain
t	time
$\mathrm{u}(\cdot,\cdot,\cdot)$	dependent variable
$\mathcal{L}(\cdot)$	linear dependant parameter
$\mathcal{N}(\cdot)$	non- linear dependant parameter
$g(\cdot)$	source term
d	degrees of freedom
$\mathbf{u}^{(i)}(\cdot)$	snapshots of the original model
$u(\cdot)$	coefficients of $\mathbf{u}^{(i)}(\cdot)$
$\mathbf{v}_j(\cdot)$	orthonormal basis
$C(\cdot,\cdot,\cdot)$	spatial autocovariance matrix
λ	eigenvalues
r	number of most dominant eigenvectors

$w_i(\cdot)$	basis vectors for DEIM
$\boldsymbol{e}_{p_1}$	standard Euclidean basis vector
n	number of design points
$z_i(\cdot)$	coordinates of points in $\mathscr{F} composite mapping of \mathbb{z}_i(\cdot)$
$\mathbb{z}_i(\cdot)\mathbf{q}$	fluid velocity
μ	contaminant diffusion coefficient
$\boldsymbol{e}_i$	unit vector in the x_i direction
n	number of training points
n_t	number of testing points
N_n	number of points used for the inverse mapping
s^2	free parameter of the Gaussian kernel
ϵ	normalized error
m	number of snapshots
Re	Reynold's number
$\mathbf{M}$	mass matrix
$\mathbf{S}$	stiffness matrix

4.1 ROMs for parameterised dynamic PDEs using POD

4.1.1 Problem definition and Galerkin projection

Let $\mathbf{x} = (x_1, \ldots, x_L)$ denote a point in a bounded, regular domain $\mathcal{D} \subset \mathbb{R}^L$ ($L = 1, 2, 3$), 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 $u(\mathbf{x}, t; \boldsymbol{\xi})$:

$$\begin{aligned} \partial_t u + \mathcal{L}(\boldsymbol{\xi})u + \mathcal{N}(\boldsymbol{\xi})u = g(\mathbf{x}; \boldsymbol{\xi}) \quad (\mathbf{x}, t) \in \mathcal{D} \times (0, T] \\ u(\mathbf{x}, 0; \boldsymbol{\xi}) = u_0(\mathbf{x}; \boldsymbol{\xi}) \quad \mathbf{x} \in \mathcal{D} \end{aligned} \tag{4.1}$$

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 $g(\mathbf{x}; \boldsymbol{\xi})$ 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 functions with inner product $(v, v')_{L^2(\mathcal{D})} = \int_{\mathcal{D}} v(\mathbf{x})v'(\mathbf{x})d\mathbf{x}$. From hereon, the subscript in the notation for the inner product and norm in $L^2(\mathcal{D})$ is dropped. 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 $||u||_{L^2(0,T;\mathcal{H})} := \int_0^T ||u(\cdot, t; \boldsymbol{\xi})||_{\mathcal{H}}$. Then $u(\cdot, t; \boldsymbol{\xi}) \in \mathcal{H}$ for each $t \in (0, T)$. A spatial discretization of (4.1) leads to a system of ODEs in time:

$$\dot{\mathbf{u}}(t; \boldsymbol{\xi}) = \mathbf{A}(\boldsymbol{\xi})\mathbf{u}(t; \boldsymbol{\xi}) + \mathbf{f}(\mathbf{u}(t; \boldsymbol{\xi}); \boldsymbol{\xi}), \quad \mathbf{u}(0; \boldsymbol{\xi}) = \mathbf{u}_0(\boldsymbol{\xi}) \tag{4.2}$$

for a discrete state variable $\mathbf{u}(t; \boldsymbol{\xi}) = (u_1(t; \boldsymbol{\xi}), \ldots, u_d(t; \boldsymbol{\xi}))^T$, which is referred to as the *solution vector*. d is the number of grid points in a finite difference (FD) approximation, or 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 $\mathcal{L}(\boldsymbol{\xi})u$ and $\mathbf{f}(\mathbf{u}(t; \boldsymbol{\xi}); \boldsymbol{\xi}) \in \mathbb{R}^d$ arises from a combination of $\mathcal{N}(\boldsymbol{\xi})u$, $g(\mathbf{x}; \boldsymbol{\xi})$ and possibly the boundary conditions. The latter is nonlinear for $\mathcal{N}(\boldsymbol{\xi})u \neq 0$.

The precise relationship between $\mathbf{u}(t;\boldsymbol{\xi})$ and $u(\mathbf{x},t;\boldsymbol{\xi})$, the forms of $\mathbf{A}(\boldsymbol{\xi})$ and $\mathbf{f}(\mathbf{u};\boldsymbol{\xi})$, and the incorporation of boundary conditions depend on the method used. For a FD approximation, problem (4.1) is solved directly and the boundary conditions are incorporated in $\mathbf{f}(\mathbf{u};\boldsymbol{\xi})$. 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 $\mathbf{f}$ and/or the definition of $\mathcal{H}$. The form of $\mathbf{A}(\boldsymbol{\xi})$ is determined by the dependence of $\mathcal{L}(\boldsymbol{\xi})$ on $\boldsymbol{\xi}$. The simplest case is an affine form: $\mathbf{A}(\boldsymbol{\xi}) = \sum_i c_i(\boldsymbol{\xi})\mathbf{A}_i$, 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 $\mathbf{u}(t;\boldsymbol{\xi})$ correspond to the values of $u(\mathbf{x},t;\boldsymbol{\xi})$ at locations $\mathbf{x}^{(i)} \in \overline{\mathcal{D}}$, $i = 1,\ldots,d$, i.e., $u_i(t;\boldsymbol{\xi}) = u(\mathbf{x}^{(i)},t;\boldsymbol{\xi})$.This is assumed to be the case. A numerical solution of (4.2) yields the solution vector $\mathbf{u}^{(i)}(\boldsymbol{\xi}) := \mathbf{u}(t^{(i)};\boldsymbol{\xi})$ at times $\{t^{(i)}\}_{i=1}^m$. Each of the discrete solutions $\mathbf{u}^{(i)}(\boldsymbol{\xi}) \in \mathbb{R}^d$ is referred to as a *snapshot*.

For a fixed input $\boldsymbol{\xi} \in \mathcal{X}$, a Galerkin projection approximates the problem (4.2) in a proper (low-dimensional) subspace $\mathcal{S}$ of $\mathbb{R}^d$. Let $\mathbf{v}_j(\boldsymbol{\xi}) \in \mathbb{R}^d$, $j = 1,\ldots,r$, 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 $\mathbf{u}_r(t;\boldsymbol{\xi}) \in \mathcal{S}$ of $\mathbf{u}$ is assumed in the space $\mathrm{span}(\mathbf{v}_1(\boldsymbol{\xi}),\ldots,\mathbf{v}_r(\boldsymbol{\xi}))$:

$$\mathbf{u}_r(t;\boldsymbol{\xi}) = \sum_{j=1}^{r} a_j(t;\boldsymbol{\xi})\mathbf{v}_j(\boldsymbol{\xi}) = \mathbf{V}_r(\boldsymbol{\xi})\mathbf{a}(t;\boldsymbol{\xi}) \tag{4.3}$$

where $\mathbf{a} = (a_1(t;\boldsymbol{\xi}),\ldots,a_r(t;\boldsymbol{\xi}))^T$ and $\mathbf{V}_r(\boldsymbol{\xi}) = [\mathbf{v}_1(\boldsymbol{\xi})\ldots\mathbf{v}_r(\boldsymbol{\xi})]$. The Galerkin projection of equation (4.2) onto the basis vectors $\mathbf{v}_i(\boldsymbol{\xi})$, $i = 1,\ldots,r$, yields (replacing $\mathbf{u}$ with $\mathbf{u}_r$):

$$\dot{\mathbf{a}}(t;\boldsymbol{\xi}) = \mathbf{A}_r(\boldsymbol{\xi})\mathbf{a}(t;\boldsymbol{\xi}) + \mathbf{f}_r\left(\mathbf{a}(t;\boldsymbol{\xi});\boldsymbol{\xi}\right), \quad \mathbf{a}(0;\boldsymbol{\xi}) = \mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{u}_0(\boldsymbol{\xi}) \tag{4.4}$$

where $\mathbf{A}_r(\boldsymbol{\xi}) := \mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{A}(\boldsymbol{\xi})\mathbf{V}_r(\boldsymbol{\xi})$ and $\mathbf{f}_r\left(\mathbf{a}(t;\boldsymbol{\xi});\boldsymbol{\xi}\right) := \mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{f}\left(\mathbf{V}_r(\boldsymbol{\xi})\mathbf{a}(t;\boldsymbol{\xi});\boldsymbol{\xi}\right)$. Equations (4.4) represent a system of r ODEs in time for the coefficients $a_i(t;\boldsymbol{\xi})$. The basic premise of POD (outlined below) is the construction of a basis $\{\mathbf{v}_j(\boldsymbol{\xi})\}_{j=1}^r$ using the snapshots $\{\mathbf{u}^{(i)}(\boldsymbol{\xi})\}_{i=1}^m$.

4.1.2 Proper orthogonal decomposition

POD is presented in a number of ways (e.g., error minimization, 'variance' maximization) in the literature and often under different names. In this section a brief description of the motivation and practical (discrete) implementation is provided.

For a fixed $\boldsymbol{\xi} \in \mathcal{X}$, POD extracts an 'optimal' basis for a field $u(\mathbf{x}, t; \boldsymbol{\xi})$, $(\mathbf{x}, t) \in \mathcal{D} \times [0, T]$, given an ensemble of 'snapshots' $\{u(\mathbf{x}; t_j, \boldsymbol{\xi})\}_{j=1}^{m}$, $\mathbf{x} \in \mathcal{D}$. These are continuous equivalents of the discrete snapshots $\mathbf{u}_j(\boldsymbol{\xi})$. $u(\mathbf{x}, t; \boldsymbol{\xi})$ can be regarded as a realization of a stationary (w.r.t. t) random field indexed by $(\mathbf{x}, t)$ [72, 26, 131]. Applying Karhunen-Loéve (KL) theory [132] *for a fixed* t yields $u(\mathbf{x}, t; \boldsymbol{\xi}) = \lim_{M\to\infty} \sum_{i=1}^{M} a_i(t; \boldsymbol{\xi}) v_i(\mathbf{x}; \boldsymbol{\xi})$. The $v_i(\mathbf{x}; \boldsymbol{\xi})$ form an $L^2(\mathcal{D})$ orthonormal basis and are the eigenfunctions of an integral operator $\mathcal{C}$ with kernel given by the *spatial autocovariance function* $C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\xi})$, $\mathbf{x}, \mathbf{x}' \in \mathcal{D}$.

In practice, one must work within a finite-dimensional setting. Defining $\mathbf{U}(\boldsymbol{\xi}) := [\mathbf{u}_1(\boldsymbol{\xi}) \ldots \mathbf{u}_m(\boldsymbol{\xi})]$, the *spatial variance-covariance matrix* is given by $\mathbf{C}(\boldsymbol{\xi}) = \mathbf{U}(\boldsymbol{\xi})\mathbf{U}(\boldsymbol{\xi})^T \approx \mathbb{E}[\mathbf{u}(t; \boldsymbol{\xi})\mathbf{u}(t; \boldsymbol{\xi})^T]$. The continuous eigenvalue problem for $\mathcal{C}$ can be approximated numerically (non-uniquely) by a principal component analysis (PCA): $\mathbf{C}(\boldsymbol{\xi})\mathbf{v}_i(\boldsymbol{\xi}) = \lambda_i(\boldsymbol{\xi})\mathbf{v}_i(\boldsymbol{\xi})$ for eigenvectors $\mathbf{v}_i(\boldsymbol{\xi}) \in \mathbb{R}^d$ and eigenvalues $\lambda_i(\boldsymbol{\xi}) > 0$, $i = 1, \ldots, d$, arranged in decreasing order. The first r of these vectors define the space $\mathcal{S}(\boldsymbol{\xi})$. In certain cases it may be computationally convenient to use variants of POD/PCA to determine the $\mathbf{v}_i(\boldsymbol{\xi})$. In appendix B details of the method of snapshots and singular value decomposition are provided.

4.2 Basis emulation and DEIM extension

For each input/parameter $\boldsymbol{\xi}$ the snapshot matrix $\mathbf{U}(\boldsymbol{\xi})$ is obtained from the FOM and the basis $\mathbf{V}_r(\boldsymbol{\xi})$ is constructed according to section subsection 4.1.2. To perform an analysis w.r.t. the inputs, this procedure is computationally prohibitive. A global basis across the parameter space of interest [133] can be constructed by computing a set of snapshot matrices $\mathbf{U}(\boldsymbol{\xi}_j)$ for $\boldsymbol{\xi}_j \in \mathcal{X}$, $j = 1, \ldots, n$. The $\mathbf{v}_i(\boldsymbol{\xi})$ are extracted from a global snapshot matrix $[\mathbf{U}(\boldsymbol{\xi}_1), \ldots, \mathbf{U}(\boldsymbol{\xi}_n)] \in \mathbb{R}^{d\times nm}$ (usually after a SVD to avoid rank deficiency).

The global basis method uses information only from the "truth approximation", i.e., the FOM. The optimality of the POD method, on the other hand, is violated since the snapshots used to derive the basis do not pertain to the parameter value of interest (the particular dynamical system under consideration) during the online phase. Furthermore, the range of validity of the global basis could be limited for complex mappings between the parameters and the outputs [134]. Interpolation methods (and the method proposed here) violate the truth approximation in the sense that the snapshots or quantities derived therein are not obtained from the original model. In contrast to the global basis, however, these methods attempt to construct more accurate ROMs during the online phase. The main limitation is the accuracy of the interpolation or emulation, which depends on the data available and on the method itself. Moreover, it may not be possible to obtain sharp error bounds using such methods (in cases where the underlying PDE problem is amenable to a rigorous analysis).

Another problem associated with the standard POD-Galerkin approach is that the computational efficiency is compromised when $\mathbf{f}(\cdot;\boldsymbol{\xi}) \in \mathbb{R}^d$ is a strong nonlinearity, since the evaluation of $\mathbf{f}_r$ in Eq. (4.4) has a computational complexity that depends on d [135]. The DEIM [136] seeks a set of vectors $\mathbf{w}_i(\boldsymbol{\xi}) \in \mathbb{R}^d$, $i = 1, \ldots, d$, such that the subspace $\text{span}(\mathbf{w}_1(\boldsymbol{\xi}), \ldots, \mathbf{w}_s(\boldsymbol{\xi})) \subset \mathbb{R}^d$ for some $s \ll d$ well approximates $\mathbf{f}(\mathbf{u}(t;\boldsymbol{\xi});\boldsymbol{\xi})$ for an arbitrary t. That is, an approximation $\mathbf{f}(\mathbf{u}(t;\boldsymbol{\xi});\boldsymbol{\xi}) \approx \mathbf{W}(\boldsymbol{\xi})\mathbf{h}(t;\boldsymbol{\xi})$, where $\mathbf{W}(\boldsymbol{\xi}) = [\mathbf{w}_1(\boldsymbol{\xi}) \ldots \mathbf{w}_s(\boldsymbol{\xi})]$ and $\mathbf{h}(t;\boldsymbol{\xi}) \in \mathbb{R}^s$. The basis $\{\mathbf{w}_i(\boldsymbol{\xi})\}_{i=1}^d$ is constructed from snapshots of the nonlinearity $\{\mathbf{f}_i(\boldsymbol{\xi})\}_{i=1}^m$, where $\mathbf{f}_i(\boldsymbol{\xi}) = \mathbf{f}(\mathbf{u}_i(\boldsymbol{\xi});\boldsymbol{\xi})$, from which the matrix $\mathbf{F}(\boldsymbol{\xi}) = [\mathbf{f}_1(\boldsymbol{\xi}) \ldots \mathbf{f}_m(\boldsymbol{\xi})]$ is formed. A PCA on $\mathbf{F}(\boldsymbol{\xi})\mathbf{F}(\boldsymbol{\xi})^T$ or SVD of $\mathbf{F}(\boldsymbol{\xi})$ yields the $\{\mathbf{w}_i(\boldsymbol{\xi})\}_{i=1}^d$, arranged such that the corresponding eigenvalues decay with i.

Since the system $\mathbf{f}(\mathbf{u}(t;\boldsymbol{\xi});\boldsymbol{\xi}) = \mathbf{W}(\boldsymbol{\xi})\mathbf{h}(t;\boldsymbol{\xi})$ is overdetermined in $\mathbf{h}(t;\boldsymbol{\xi})$, the DEIM selects s of the d equations to obtain an 'optimal' solution. The matrix $\mathbf{P} = [\boldsymbol{e}_{p_1} \ldots \boldsymbol{e}_{p_s}] \in \mathbb{R}^{d \times s}$ is introduced, 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.

Assuming $\mathbf{P}^T\mathbf{W}(\boldsymbol{\xi})$ is nonsingular, one obtain:

$$\begin{aligned}\mathbf{f}_r\left(\mathbf{a}(t;\boldsymbol{\xi});\boldsymbol{\xi}\right) \approx \mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{W}(\boldsymbol{\xi})\mathbf{h}(t;\boldsymbol{\xi}) &= \mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{W}(\boldsymbol{\xi})(\mathbf{P}^T\mathbf{W}(\boldsymbol{\xi}))^{-1}\mathbf{P}^T\mathbf{f}(\mathbf{u}(t;\boldsymbol{\xi});\boldsymbol{\xi})\\ &= \mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{W}(\boldsymbol{\xi})(\mathbf{P}^T\mathbf{W}(\boldsymbol{\xi}))^{-1}\mathbf{f}(\mathbf{P}^T\mathbf{u}(t;\boldsymbol{\xi});\boldsymbol{\xi})\end{aligned} \tag{4.5}$$

assuming that the function $\mathbf{f}\left(\cdot;\boldsymbol{\xi}\right)$ acts pointwise. The indices $p_i \in \{1, 2, \ldots, d\}$, $i = 1, \ldots, s$ are specified by a greedy algorithm [136] that satisfies the following error bound (for a given s):

$$||\mathbf{f} - \widehat{\mathbf{f}}|| \leq ||(\mathbf{P}^T\mathbf{W}(\boldsymbol{\xi}))^{-1}||\,||(\mathbf{I} - \mathbf{W}(\boldsymbol{\xi})\mathbf{W}(\boldsymbol{\xi})^T)\mathbf{f}|| \tag{4.6}$$

where $||\cdot||$ is the standard Euclidean norm and $\widehat{\mathbf{f}} := \mathbf{W}(\boldsymbol{\xi})(\mathbf{P}^T\mathbf{W}(\boldsymbol{\xi}))^{-1}\mathbf{P}^T\mathbf{f}$ is the DEIM approximation of $\mathbf{f}$. This estimate is valid for a given t (considering $\mathbf{f}$ as a function of t) by virtue of the second factor on the r.h.s., which is the error in the best 2-norm approximation of $\mathbf{f}$ in Range($\mathbf{W}(\boldsymbol{\xi})$).

In this chapter, a systematic and rigorous method to approximate the local basis and the nonlinearity is introduced by first approximating the snapshots $\{\mathbf{u}_i(\boldsymbol{\xi})\}_{i=1}^m$ and $\{\mathbf{f}_i(\boldsymbol{\xi})\}_{i=1}^m$ for an arbitrary input $\boldsymbol{\xi}$ using Bayesian nonlinear regression. These snapshots lie in very high-dimensional spaces and thus a recently developed method is used that exploits manifold learning to yield a computationally feasible Gaussian process (GP) model. Below the components of this emulation method are described and subsequently explain how it can be used for a POD analysis of parameterized, dynamic problems.

4.2.1 Formulation and solution of the learning problem

For an arbitrary input $\boldsymbol{\xi}$, consider the mapping $\boldsymbol{\eta} : \mathcal{X} \rightarrow \mathcal{O} \subset \mathbb{R}^{md}$ defined below:

$$\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi}) = \left(\mathbf{u}_1(\boldsymbol{\xi})^T, \ldots, \mathbf{u}_m(\boldsymbol{\xi})^T\right)^T \in \mathbb{R}^{md} \tag{4.7}$$

i.e., a vectorial rearrangement of snapshots $\{\mathbf{u}_i(\boldsymbol{\xi})\}_{i=1}^m$ for the given value of $\boldsymbol{\xi}$. A similar map $\mathbf{y}^f = \boldsymbol{\eta}^f(\boldsymbol{\xi})$ for snapshots of the nonlinearity $\{\mathbf{f}_i(\boldsymbol{\xi})\}_{i=1}^m$ can be defined. The emulation procedure mirrors that described below for the snapshots $\{\mathbf{u}_i(\boldsymbol{\xi})\}_{i=1}^m$.

The aim is to approximate the mapping $\boldsymbol{\eta}(\cdot)$ given *training points* $\mathbf{y}_j =$

$\boldsymbol{\eta}(\boldsymbol{\xi}_j) \in \mathcal{O}$ (in a high dimensional space) for *design points* $\boldsymbol{\xi}_j \in \mathcal{X}$, $j = 1, \ldots, n$. One of the main methods for dealing with such high dimensional outputs is to define approximate outputs in an $q-$dimensional subset $\mathcal{O}_q \subset \mathcal{O}$ ($q \ll md$) using PCA and independently emulate the q coefficients of the points in $\mathcal{O}_q$ for new values of $\boldsymbol{\xi}$ [43]. Shah and co-workers [137, 138] extended the latter method by replacing PCA with manifold learning methods, making it applicable to a broader class of output spaces $\mathcal{O}$. In this chapter the method of [137, 138] is employed with kernel PCA (kPCA), which is outlined in subsection 2.1.1, together with an approximation of the inverse map. kPCA [102] defines a map $\boldsymbol{\phi}_q : \mathcal{O} \to \mathscr{F}_q$, where $\mathscr{F}_q$ is a q-dimensional feature space. The coordinates $z_i(\mathbf{y})$ of points $\boldsymbol{\phi}_q(\mathbf{y})$ in $\mathscr{F}_q$ define composite maps from the input space $\mathcal{X}$ to $\mathbb{R}$, i.e., $\mathbb{z}_i(\boldsymbol{\xi}) := z_i(\boldsymbol{\eta}(\boldsymbol{\xi}))$, $i = 1, \ldots, q$. Independent GP priors are placed over these maps, justified by the properties of kPCA.

The approximation of $\boldsymbol{\eta} : \mathcal{X} \to \mathcal{O}$ given the training points $\{\mathbf{y}_j\}_{j=1}^n$ is then substituted for independent approximations of the coefficients $\mathbb{z}_i(\boldsymbol{\xi})$, $i = 1, \ldots, q$, given training data $\{\mathbb{z}_i(\boldsymbol{\xi}_j) = z_i(\boldsymbol{\eta}(\boldsymbol{\xi}_j))\}_{j=1}^n$. The value of $\mathbb{z}_i(\boldsymbol{\xi})$ for a new input $\boldsymbol{\xi}$ is inferred from scalar GP emulation (outlined in section 3.2) as the mean of a posterior distribution. Given $\{\mathbb{z}_i(\boldsymbol{\xi})\}_{i=1}^q$, an approximation of the inverse $\boldsymbol{\phi}_q^{-1} : \mathscr{F}_q \to \mathcal{O}$ yields an approximation of $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi}) \in \mathcal{O}$, from which $\{\mathbf{u}_i(\boldsymbol{\xi})\}_{i=1}^m$ can be obtained using definition (4.7). GP emulation is exact at the training points if there are no (spurious) errors in the training data. In the present case, an error is introduced in the pre-image map so that the training snapshots will not be recovered exactly. This error, however, is negligible (section 4.3). I should be noted that the size of md is not a limitation for the manifold learning methods employed in this chapter, in which the eigenvalue problems are primarily dependent on the number of training points n.

4.2.2 Main Algorithm

Once the snapshots $\{\mathbf{u}_i(\boldsymbol{\xi})\}_{i=1}^m$ (and $\{\mathbf{f}_i(\boldsymbol{\xi})\}_{i=1}^m$ for nonlinear problems) are obtained using the procedure outlined in sections 4.1 and 4.2.2 for a new input $\boldsymbol{\xi}$, POD can be performed in the usual manner (with the extended DEIM for nonlinear problems). The entire procedure is outlined in Algorithm 1. It has to be mentioned that kPCA can be replaced with other manifold learning

Algorithm 1 kGPE-POD (steps 1a-7a) and kGPE-POD-DEIM (steps 1a-7a **and** 1b-7b).

```
1a: Snapshots from FOM:
    $\mathbf{u}_j(\boldsymbol{\xi}_i)^T,\ i=1,\dots,n,\ j=1,\dots,m$
2a: Set: $\mathbf{y}_i \leftarrow \boldsymbol{\eta}(\boldsymbol{\xi}_i)$
    $\leftarrow (\mathbf{u}_1(\boldsymbol{\xi}_i)^T,\dots,\mathbf{u}_m(\boldsymbol{\xi}_i)^T)^T, i=1,\dots,n$
3a: Do kPCA for $\{\mathbf{y}_i\}_{i=1}^n$
    $\to \{(z_1(\mathbf{y}_i),\dots,z_q(\mathbf{y}_i))^T\}_{i=1}^n$
4a: for $j \leftarrow 1$ to $q$ do
        $\{\eta(\boldsymbol{\xi}_i) \leftarrow \mathbb{z}_j(\boldsymbol{\xi}_i) \leftarrow z_j(\mathbf{y}_i)\}_{i=1}^n$
        Perform scalar GPE: $\mathbb{z}_j(\boldsymbol{\xi}) \leftarrow \mathbb{E}[\eta(\boldsymbol{\xi})]$
    end for
5a: Inverse map:
    $\boldsymbol{\eta}(\boldsymbol{\xi}) \leftarrow \sum_{j\in\mathcal{J}} \mathbf{y}_j \chi(d_{j,*})/\sum_{i\in\mathcal{J}} \chi(d_{i,*})$
6a: Snapshots for input $\boldsymbol{\xi}$:
    $(\mathbf{u}_1(\boldsymbol{\xi})^T,\dots,\mathbf{u}_m(\boldsymbol{\xi})^T)^T \leftarrow \boldsymbol{\eta}(\boldsymbol{\xi})$
7a: Perform POD with $\{\mathbf{u}_i(\boldsymbol{\xi})\}_{i=1}^m$
```

```
1b: Collect nonlinearity snapshots:
    $\mathbf{f}_j(\boldsymbol{\xi}_i),\ i=1,\dots,n,\ j=1,\dots,m$
2b: Set: $\mathbf{y}_i^f \leftarrow \boldsymbol{\eta}^f(\boldsymbol{\xi}_i)$
    $\leftarrow (\mathbf{f}_1(\boldsymbol{\xi}_i)^T,\dots,\mathbf{f}_m(\boldsymbol{\xi}_i)^T)^T,\ i=1,\dots,n$
3b: Do kPCA for $\{\mathbf{y}_i^f\}_{i=1}^n$
    $\to \{(z_1^f(\mathbf{y}_i^f),\dots,z_q^f(\mathbf{y}_i^f))^T\}_{i=1}^n$
4b: for $j \leftarrow 1$ to $q$ do
        $\{\eta^f(\boldsymbol{\xi}_i) \leftarrow \mathbb{z}_j^f(\boldsymbol{\xi}_i) \leftarrow z_j^f(\mathbf{y}_i^f)\}_{i=1}^n$
        Perform scalar GPE: $\mathbb{z}_j^f(\boldsymbol{\xi}) \leftarrow \mathbb{E}[\eta^f(\boldsymbol{\xi})]$
    end for
5b: Inverse map:
    $\boldsymbol{\eta}^f(\boldsymbol{\xi}) \leftarrow \sum_{j\in\mathcal{J}} \mathbf{y}_j^f \chi(d_{j,*})/\sum_{i\in\mathcal{J}} \chi(d_{i,*})$
6b: Snapshots for nonlinear term:
    $(\mathbf{f}_1(\boldsymbol{\xi})^T,\dots,\mathbf{f}_m(\boldsymbol{\xi})^T)^T \leftarrow \boldsymbol{\eta}^f(\boldsymbol{\xi})$
7b: Perform DEIM on $\{\mathbf{f}_i(\boldsymbol{\xi})\}_{i=1}^m$
```

methods, e.g., diffusion maps or Isomap [137, 138]. The terminology 'kGPE-POD' is introduced to denote the method of Algorithm 1 without the extended DEIM (i.e, steps 1a-7a alone). Similarly, the terminology 'kGPE-POD-DEIM' to denote the method of Algorithm 1 with the extended DEIM (steps 1a-7a **and** steps 1b-7b together) is used.

4.3 Applications, results and discussion

4.3.1 2D contaminant transport

The transport of a contaminant governed by a convection-diffusion equation is considered. This model can be used, e.g., for real-time prediction or for quantifying uncertainty in the concentration to support decision making [139]. The problem is specified as follows:

$$\begin{gathered}\partial_t u + \mathbf{q}\cdot\nabla u - \mu\nabla^2 u = 0 \quad \mathbf{x} = (x_1, x_2) \in \mathcal{D} := [0,1]\times[0,1] \\ u = 0 \quad \mathbf{x}\in\partial\mathcal{D}, \quad u(\mathbf{x},t) = u_0 \quad t = 0\end{gathered} \tag{4.8}$$

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

$$\mathbf{q}(\mathbf{x}) = \frac{a_1(x_1 - \widehat{x}_1)\boldsymbol{e}_1 + a_2(x_2 - \widehat{x}_2)\boldsymbol{e}_2}{(x_1 - \widehat{x}_1)^2 + (x_2 - \widehat{x}_2)^2} \tag{4.9}$$

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 $\mathbf{x} = (\widehat{x}_1, \widehat{x}_2)^T$, the norm of velocity is set to zero at this location. Also, $a_1 = a_2 = 1$, $\mu = 1$ and variations in the input $\boldsymbol{\xi} = (\widehat{x}_1, \widehat{x}_2)^T \in \mathcal{X} := [0, 1] \times [0, 1]$ are considered.

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 [97]. For each input, the FOM was solved to yield solution vectors (snapshots) $\mathbf{u}_i(\boldsymbol{\xi}_j) \in \mathbb{R}^d$, $i = 1, \ldots, 100$, $j = 1, \ldots, 500$. The data points (vectorized snapshots) $\mathbf{y}_j = \boldsymbol{\eta}(\boldsymbol{\xi}_j)$, $j = 1, \ldots, 500$, were obtained using Eq. (4.7). 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 [140]: $s^2 = n^{-2} \sum_{i,j=1}^{n} ||\mathbf{y}_i - \mathbf{y}_j||^2$. 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) were used. The hyperparameters were found using a MLE (gradient descent). Errors in the predictions of the vectorised snapshots $\mathbf{y}_j$ were measured using a normalized error: $\epsilon = ||\mathbf{y}_j^p - \mathbf{y}_j||/||\mathbf{y}_j||$, where $\mathbf{y}_j^p$ denotes the prediction of the test point $\mathbf{y}_j = \boldsymbol{\eta}(\boldsymbol{\xi}_j)$, $j = 1, \ldots, n_t$, using steps 1a-6a of Algorithm 1. Errors in the predictions using kGPE-POD/kGPE-POD-DEIM at $\boldsymbol{\xi}_j$ were measured using a relative error ϵ_r:

$$\epsilon_r = \frac{1}{m} \sum_{i=1}^{m} \frac{||\mathbf{u}_i^p(\boldsymbol{\xi}_j) - \mathbf{u}_i(\boldsymbol{\xi}_j)||}{||\mathbf{u}_i(\boldsymbol{\xi}_j)||} \tag{4.10}$$

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

First the normalized errors ϵ in the predictions of the test data points $\mathbf{y}_j = \boldsymbol{\eta}(\boldsymbol{\xi}_j)$, $j = 1, \ldots, n_t$ are examined. Using $m = 10$ of the snapshots (selecting every 10), Fig. 4.1 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 may be 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 1) 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).

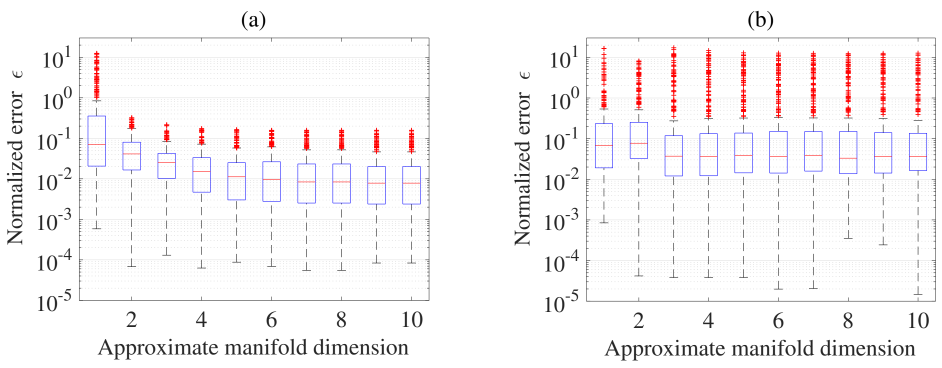


Figure 4.1: Tukey box plots of ϵ with increasing q for the contaminant transport model ($n_t = 300$, $n = 80$ and $m = 10$): (a) kPCA; (b) Isomap.

Fig. 4.2 compares kGPE-POD with a global basis method for increasing POD dimension r. In the global basis method the snapshot matrices compris-

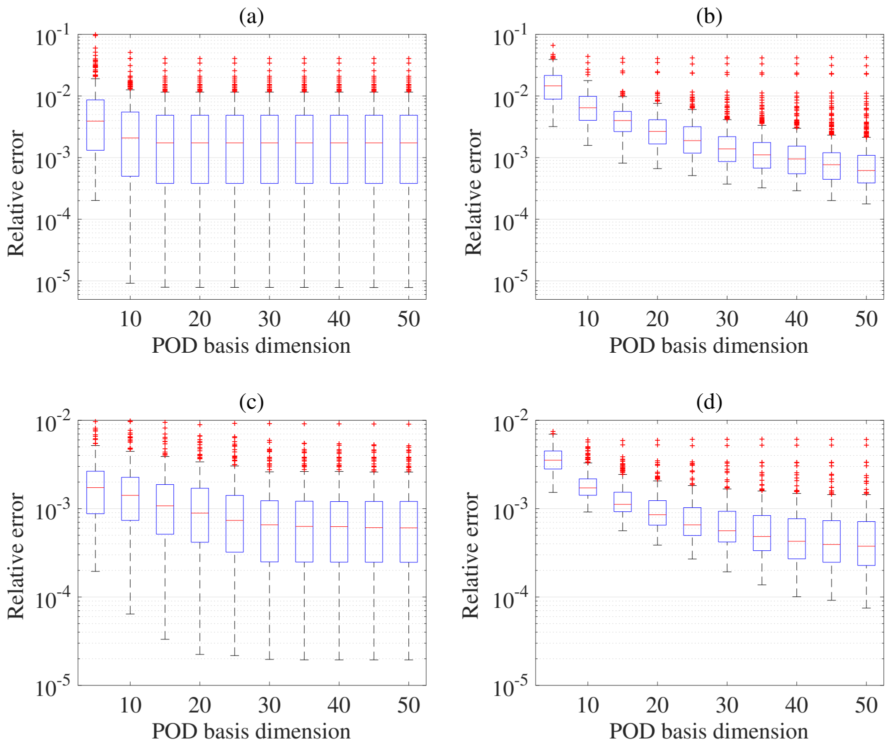


Figure 4.2: Tukey box plots of ϵ_r with increasing r for the contaminant transport model ($n_t = 300$ and $n = 80$). (a) kGPE-POD with $m = 10$; (b) global basis with $m = 10$; (c) kGPE-POD with $m = 100$; (d) global basis with $m = 100$.

ing 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. 4.2, 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 not pertain to the test parameter values.

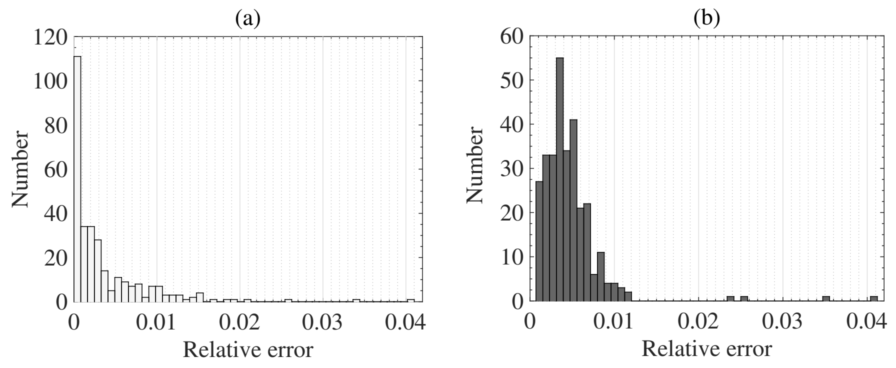


Figure 4.3: Histograms of ϵ_r corresponding to $m = 10$, $r = 15$ in Fig. 4.2, using: (a) kGPE-POD; and (b) a global basis.

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. 4.2(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. 4.3 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, Fig. 4.4 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. 4.2(a). The change in the profiles from one input to the other is well captured. Figs. 4.4(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. 4.5, 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 for the accumulated contaminant concentration $\bar{u}(\mathbf{x}; \boldsymbol{\xi}) := \int_0^T u(\mathbf{x}, t; \boldsymbol{\xi}) dt$ at the location $\mathbf{x}_c = (0.5, 0.5)^T$ was considered, 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 $\bar{u}(\mathbf{x}_c; \boldsymbol{\xi})$ 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 $p(\boldsymbol{\xi})$. The setting $q = 6$, $n = 80$, $N_M = 3000$ were used, and $\bar{u}(\mathbf{x}_c; \boldsymbol{\xi})$ approximated with a trapezoidal rule. Fig. 4.6 compares the histograms obtained from kGPE-POD, the global basis method and the FOM, using $m = 10$ snapshots. The FOM took 55.18 h to complete and yielded $\mu_{ac} = 0.011087$ and $\sigma_{ac} = 0.001218$, obtained from $\mu_{ac} = (1/N_M) \sum_{i=1}^{N_M} \bar{u}(\mathbf{x}; \boldsymbol{\xi}^i)$ and $\sigma_{ac}^2 = (N_M - 1)^{-1} \sum_{i=1}^{N_M} (\bar{u}(\mathbf{x}; \boldsymbol{\xi}^i) - \mu_{ac})^2$. 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. 4.7.

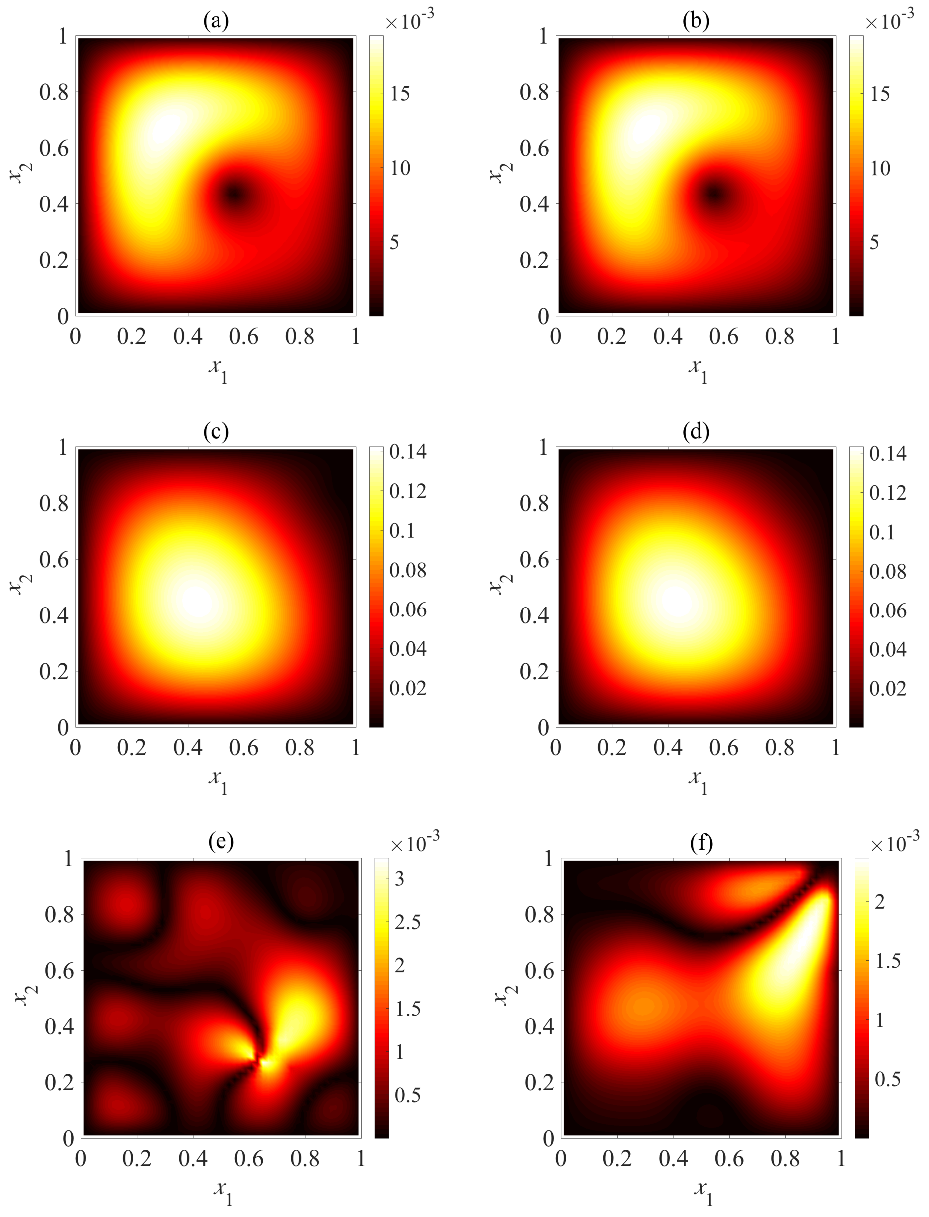


Figure 4.4: (a) The FOM and (b) the kGPE-POD prediction of the concentration field (mol m^{-3}) for the contaminant transport model at $\boldsymbol{\xi} = (0.7382, 0.4179)^T$ and $t = 0.02s$ ($\epsilon_r \approx 0.0021$). (c) The FOM and (d) the kGPE-POD predictions at $\boldsymbol{\xi} = (0.7539, 0.7461)^T$ 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 $\boldsymbol{\xi} = (0.7539, 0.7461)^T$.

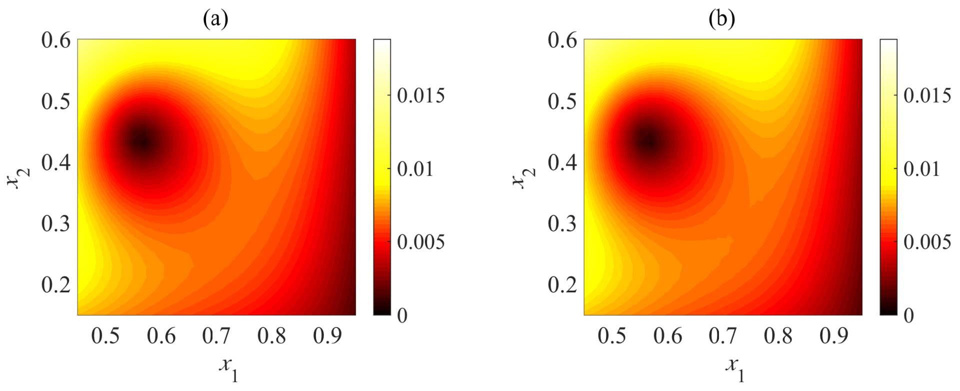


Figure 4.5: A close-up of (a) the kGPE-POD prediction and (b) the test corresponding to Figs. 4.4(a) and (b).

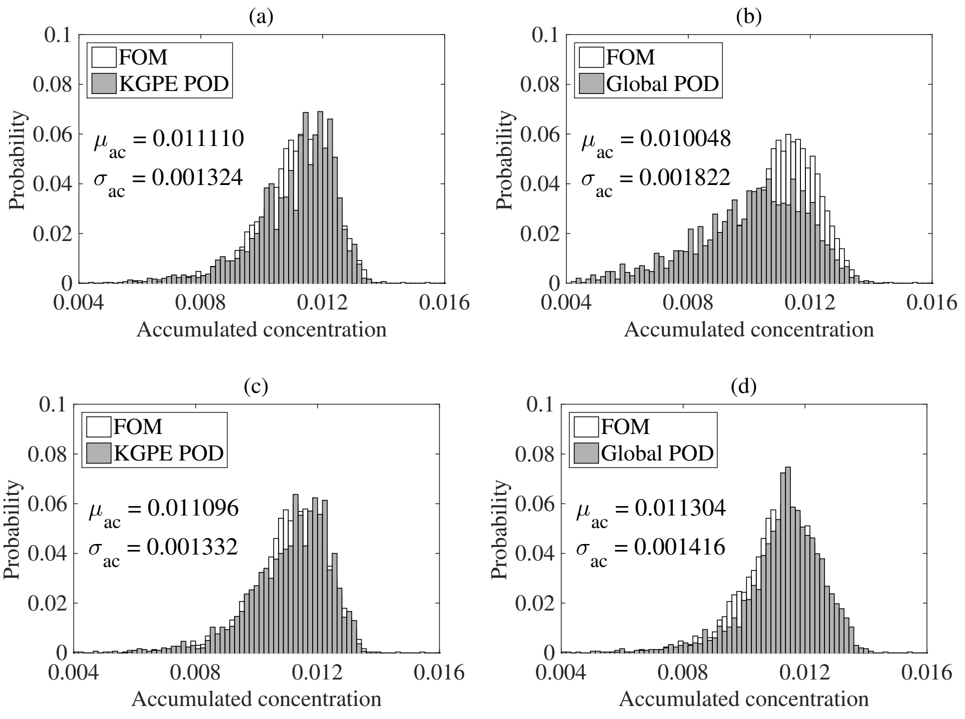


Figure 4.6: Estimated distribution of $\bar{u}(\mathbf{x}_c;\boldsymbol{\xi})$ from $N_M = 3000$ MC samples using $n = 80$ and $m = 10$: (a) kGPE-POD with $r = 10$; (b) global basis with $r = 10$; (c) kGPE-POD with $r = 50$; (d) global basis with $r = 50$.

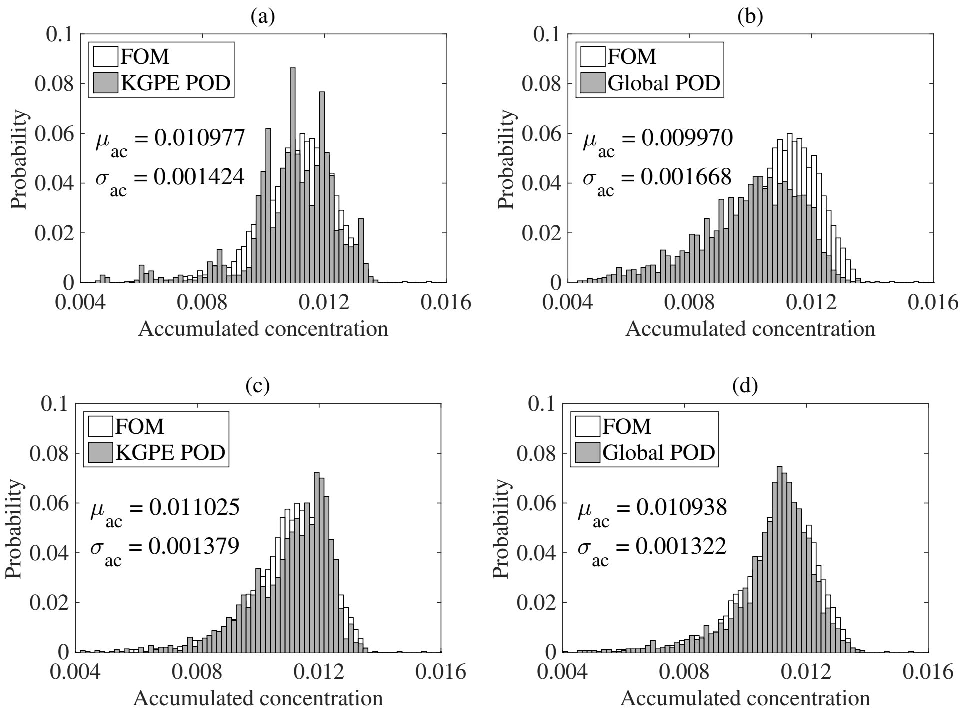


Figure 4.7: Estimated distribution of $\bar{u}(\mathbf{x}_c;\boldsymbol{\xi})$ from $N_M = 3000$ MC samples with $n = 80$ and $m = 100$: (a) kGPE-POD with $r = 10$; (b) global basis with $r = 10$; (c) kGPE-POD with $r = 30$; (d) global basis with $r = 30$.

kGPE-POD was again more accurate for $r = 10$, while for $r = 30$, the two methods exhibited a similar level accuracy.

4.3.2 Burgers equation

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

$$\begin{gathered}\partial_t u + \frac{1}{2}\partial_x(u^2) - \frac{1}{Re}\partial_{xx}u = g(x), \quad x \in \mathcal{D} := (0,1) \\ u(0,t) = u(1,t) = 0, \quad u(x,0) = u_0(x) := \sin(k\pi x)e^{-(c_1 x + c_2)}\end{gathered} \tag{4.11}$$

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. A weak solution $u(x,t;\boldsymbol{\xi}) \in \mathcal{V} := H_0^1(\mathcal{D})$ is sought satisfying:

$$(\partial_t u, v) + \frac{1}{2}(\partial_x(u^2), v) + \frac{1}{Re}a(u,v) = (g,v) \quad \forall v \in \mathcal{V} \tag{4.12}$$

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 (4.12) holds $\forall v = v^h(x) \in \mathcal{V}^h$. Use is also made of the *group (product) approximation* [141]: $u(x, t; \boldsymbol{\xi})^2 \approx \sum_{j=1}^d u_j(t; \boldsymbol{\xi})^2 \psi_j(x) \in \mathcal{V}^h$. Setting $u = u^h$ and $v^h = \psi_j$ in (4.12) gives the semi-discrete problem:

$$\sum_{i=1}^d \dot{u}_i(t; \boldsymbol{\xi})(\psi_i, \psi_j) + \frac{1}{2}\sum_{i=1}^d u_i(t; \boldsymbol{\xi})^2(\psi_i', \psi_j) + \frac{1}{Re}\sum_{i=1}^d u_i(t; \boldsymbol{\xi})(\psi_i', \psi_j') = (g, \psi_j) \tag{4.13}$$

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 $\mathbf{u}(t; \boldsymbol{\xi}) = (u_1(t; \boldsymbol{\xi}), \ldots, u_d(t; \boldsymbol{\xi}))^T$, Eq. (4.13) and the initial condition lead to:

$$\mathbf{M}\dot{\mathbf{u}}(t; \boldsymbol{\xi}) + \mathbf{b}(\mathbf{u}(t; \boldsymbol{\xi})) + \frac{1}{Re}\mathbf{S}\mathbf{u}(t; \boldsymbol{\xi}) = \mathbf{g}, \quad \mathbf{M}\mathbf{u}(0; \boldsymbol{\xi}) = \mathbf{u}_0 \tag{4.14}$$

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 $\mathbf{u}_0$ and $\mathbf{g}$ are (u_0, ψ_j) and (g, ψ_j), respectively, The nonlinear vector function $\mathbf{b}(\mathbf{u}(t; \boldsymbol{\xi}))$ arises from the second term in (4.13). A Runge-Kutta method with a variable time step was used to solve the semi-discrete problems in this example.

The coefficients $u_{i,j}(\boldsymbol{\xi})$, $j = 1, \ldots, d$, of the snapshots $\mathbf{u}_i(\boldsymbol{\xi}) = \mathbf{u}(t_i; \boldsymbol{\xi})$, $i = 1, \ldots, m$, 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 the $L^2(\mathcal{D})$ norm was chosen 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 $\mathbf{v}_j(\boldsymbol{\xi})$, given by $\mathbf{v}_j(\boldsymbol{\xi}) = \mathbf{M}^{-1/2}\overline{\mathbf{v}}_j(\boldsymbol{\xi})$, where $\overline{\mathbf{v}}_j(\boldsymbol{\xi})$ 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 $\mathbf{v}_j(\boldsymbol{\xi})$ w.r.t. $\langle \mathbf{v}_j(\boldsymbol{\xi}), \mathbf{v}_i(\boldsymbol{\xi})\rangle_{\mathbf{M}} := \mathbf{v}_j(\boldsymbol{\xi})^T\mathbf{M}\mathbf{v}_i(\boldsymbol{\xi})$.

The solution vector is then expanded as in Eq. (4.3): $\mathbf{u}(t;\boldsymbol{\xi}) \approx \mathbf{u}(t;\boldsymbol{\xi}) = \sum_{j=1}^{r} a_j(t;\boldsymbol{\xi})\mathbf{v}_j(\boldsymbol{\xi}) = \mathbf{V}_r(\boldsymbol{\xi})\mathbf{a}(t;\boldsymbol{\xi})$, leading to the reduced order model:

$$\begin{gathered}\dot{\mathbf{a}}(t;\boldsymbol{\xi}) + \mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{b}\left(\mathbf{V}_r(\boldsymbol{\xi})\mathbf{a}(t;\boldsymbol{\xi})\right) + \frac{1}{Re}\mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{S}\mathbf{V}_r(\boldsymbol{\xi})\mathbf{a}(t;\boldsymbol{\xi}) = \mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{g} \\ \mathbf{a}(0;\boldsymbol{\xi}) = \mathbf{a}_0(\boldsymbol{\xi}) := \mathbf{V}_r(\boldsymbol{\xi})^T\mathbf{u}_0\end{gathered} \tag{4.15}$$

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 $\int_0^T a(u,v)u dt = \lambda v$ leads to the eigenvalue problem $\mathbf{C}(\boldsymbol{\xi})^T\mathbf{S}\mathbf{v}_j(\boldsymbol{\xi}) = \lambda\mathbf{v}_j(\boldsymbol{\xi})$. The POD basis vectors are then given by $\mathbf{v}_j(\boldsymbol{\xi}) = \mathbf{S}^{-1/2}\overline{\mathbf{v}}_j(\boldsymbol{\xi})$, where $\overline{\mathbf{v}}_j(\boldsymbol{\xi})$ 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 t $g(x) \equiv 0$ and $k = 1$ were set. 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 (4.14) with $d = 64$ nodes, for each $j = 1,\ldots,500$, to obtain the solution vectors $\mathbf{u}(t_i;\boldsymbol{\xi}_j)$ and nonlinearity $\mathbf{b}(\mathbf{u}(t_i;\boldsymbol{\xi}_j))$ at times, $t_i = 0.25i$, $i = 1,\ldots,40$ ($m = 40$). This yielded the data points (vectorized snapshots) $\mathbf{y}_j = \boldsymbol{\eta}(\boldsymbol{\xi}_j)$ and $\mathbf{y}_j^f = \boldsymbol{\eta}^f(\boldsymbol{\xi}_j)$, $j = 1,\ldots,500$, according to Eq. (4.7). 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. A value of $q = 9$ (kPCA) was used in the results presented below. Fig. 4.8(a) shows the results of kGPE-POD-DEIM for an increasing r (with $s = r$). The relative errors converge after $r = 30$, i.e., further decreases are negligible. Fig. 4.8(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. 4.8(a). The two sets of profiles are very close. The inset in Figure (b) shows the absolute pointwise error at each point x in the 1-d

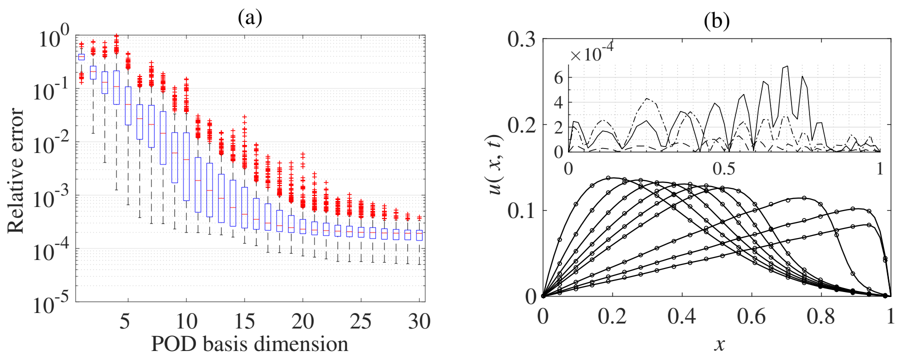


Figure 4.8: (a) Tukey box plots of ϵ_r with increasing r using kGPE-POD-DEIM for Burgers model case 1 ($n = 180$, $n_t = 300$ and $m = 15$). (b) Velocity profiles at $t = 0, 0.5, 1, 1.5, 2, 2.5, 5, 7.5, 10$ s simulated with the FOM (filled circles, every third node) and kGPE-POD-DEIM (solid lines) for 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).

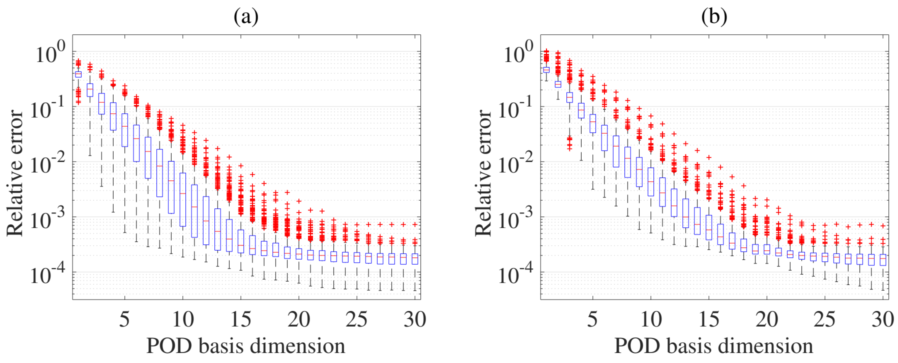


Figure 4.9: Tukey box plots of ϵ_r with increasing r for Burgers model case 1 ($n_t = 300$, $m = 40$ and $n = 180$): (a) kGPE-POD; (b) a global basis.

spatial domain. In other words, at each point x in the discretisation, the absolute (i.e., magnitude) difference between the full solution and the POD solution. This is shown for three different times (2.5 s, 5 s and 10 s) to show how the error changes along the x axis and also how it evolves with time. 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 %.

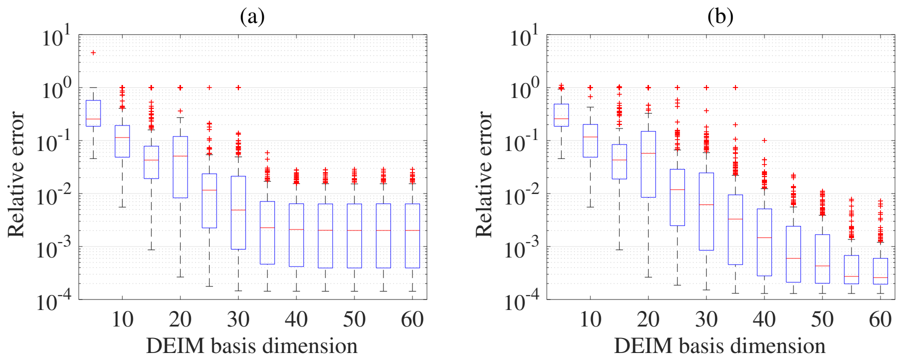


Figure 4.10: Tukey box plots of ϵ_r with increasing s for Burgers model case 2 ($n_t = 300$, $n = 180$ and $m = 200$) using kGPE-POD-DEIM with: (a) $r = 30$; and (b) $r = 50$.

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. 4.9, 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 $c_2 = 0.2$ were set, with inputs $\boldsymbol{\xi} = (c_1, Re)^T \in \mathcal{X} = [2,5] \times [10,1000]$. As before 500 inputs were selected using a Sobol sequence and the FOM ran to generate data points, with $n_t = 300$ reserved for testing. In this case $d = 128$ nodes were used and after inspection of the normalized errors ϵ $q = 9$ was set. In contrast to the previous case, a large m ($m > 120$) was required for accurate results.

Fig. 4.10 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., [142, 85].

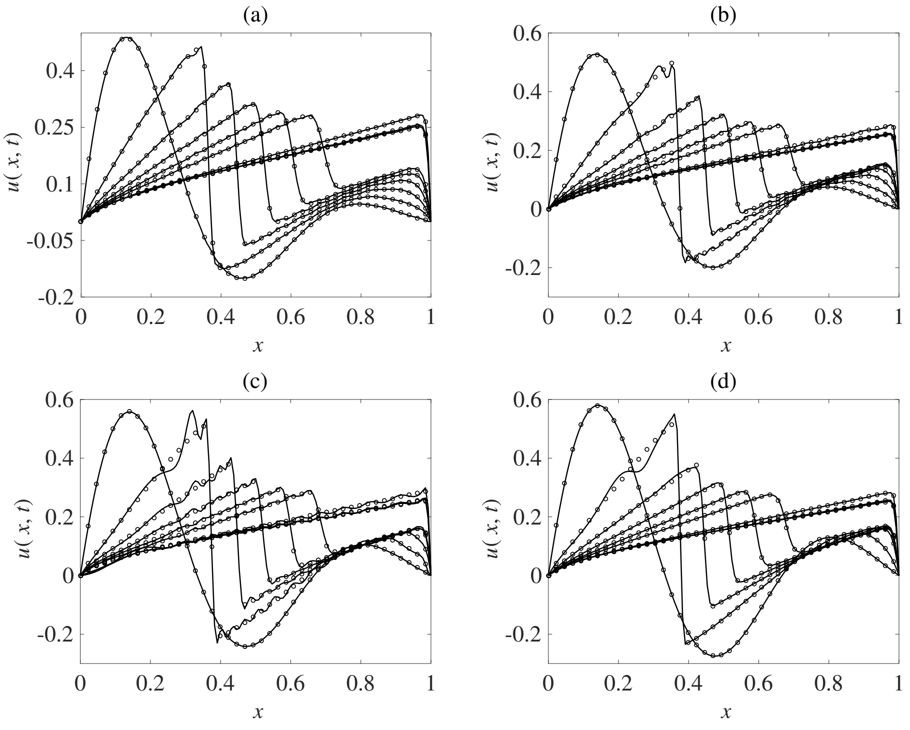


Figure 4.11: Velocity profiles predicted by the FOM (filled circles, every third node) and kGPE-POD-DEIM (solid lines) at $t = 0, 0.5, 1, 1.5, 2, 2.5, 5, 7.5, 10$ s for Burgers model case 2. (a) A point near the median ($\epsilon_r \approx 0.0022$) at $r = 30$, $s = 40$ in Figure 4.10(a); (b) a point 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 4.10(b).

For $r = 30$ and $s = 40$, Figs. 4.11(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.

4.10(a), while the second corresponds to a point near the upper whisker. Fig. 4.11(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. 4.11(d). In Fig. 4.11(d) it is seen that the solutions at early times are more stable. The observed instability is a common feature of POD models [143, 144, 145]. Stabilization schemes, e.g., alternative inner products, post-processing steps and modification of the underlying model [144, 146, 147] can be incorporated within the framework developed in this work in order to eliminate or minimize such problems.

4.4 Concluding remarks

The aim of the chapter was to develop efficient meta-models based on reduced order modeling techniques for parameterized PDEs where the quantities of interest are spatio-temporal fields. This new POD-ROM method uses bayesian inference to predict the new basis for new parameter values. In contrast to the global POD method, the new method proposed here is more computational expensive due to the diagonalization of the snapshot matrix, although, this cost is low as can be seen in the first example presented in this chapter. In comparison to the FOM the method proposed here is almost 380 faster for solving the desired quantity of interest (depending on the example), while at the same time remains accurate.

Moreover, in the examples considered here, a higher value of POD basis dimensions (r) is needed in order to perform similar to the POD-ROM method described earlier, especially, for lower values of snapshots. This means that the advantages of the global basis method as a meta-model become limited as the number of POD basis dimensions increases.

The method developed in this chapter can be seen as a general framework and alternations can be made to solve different problems. Someone can use different manifold learning technique and stabilization approaches. Furthermore, the developed Gaussian process based emulator could be used to the POD basis $\mathbf{V}_{\boldsymbol{r}}(\boldsymbol{\xi})$ or even the system matrix $\boldsymbol{A}_{\boldsymbol{r}}(\boldsymbol{\xi})$. Comparing the developed method of this chapter to the full order model has been proven that it reduces the computational cost.

Reduced order models use a linear subspace of the original output space and therefore are linear and struggle to find solutions for complex non-linear data, which in turn can affect the computational time they usually need to find a solution. In contrast, data driven methods may not be able to approximate the solution of non-linear models like the full order model can.

Chapter 5

GP Emulation with kernel PCA and Diffusion maps

Emulators can be used to find computational feasible approximations of computer models that are based on partial differential equations. Although, they may struggle to provide meaningful results when dealing with high-dimensional spaces or when the response surface is highly nonlinear. In this chapter a Gaussian process emulator is developed in conjunction with dimensionality reduction techniques i.e. kPCA and diffusion maps. In terms of diffusion maps, the main challenge is the computation of the inverse mapping. Etynier et al. [148] proposed a low rank-approximation for the inverse mapping while Gepshtein et al. [149] proposed an approach which is based on discrete approximation by using spectral relaxation. Here, an efficient approximation of the inverse map is proposed and tested.

This chapter is based on [138]: W. W. Xing, V. Triantafyllidis, A.A. Shah, P.B. Nair, N. Zabaras, *Manifold learning for the emulation of spatial fields from computational models* Journal of Computational Physics, vol. 326, pp. 666-690, 2016.

Abbreviations

$\boldsymbol{\eta}$	representation of computer model as a function
$\boldsymbol{\xi}$	input vector of parameters
$\mathbf{x}$	points in a regular domain
$\mathbf{y}$	output of the simulator
ζ_i, ζ_i^*	SVM slack variables
$\mathsf{k}(\cdot,\cdot)$	kernel
$\mathbf{D}$	degree matrix
$\mathbf{P}$	Markov matrix
$\mathbf{K}$	kernel matrix
s_i	eigenvectors
γ_i	eigenvalues
$\mathbf{p}_j^t$	probability mass function
$\mathbf{r}_i$	right eigenvectors
$\mathbf{l}_i$	left eigenvectors
$\boldsymbol{\psi}^t$	diffusion map

$m(\cdot)$	mean function of a GP
$\mathbb{c}(\cdot,\cdot)$	covariance function of a GP
$\boldsymbol{\theta}$	vector of hyperparameters
Φ	data matrix
$d_{i,*}$	distance measure
$\underline{\mathbf{K}}$	augmented kernel matrix
$\underline{\mathbf{D}}$	augmented degree matrix
$\underline{\mathbf{P}}$	augmented Markov matrix
m	number of training points
m_t	number of testing points
$\mathrm{u}(\cdot,\cdot,\cdot)$	dependent variable
$\mathbf{v}$	flow velocity
T	temperature
p	pressure
$\boldsymbol{g}$	gravitational acceleration
ρ	fluid density
ϵ	porosity of the medium
κ	permeability of the medium
ω	dynamic viscosity
c_e	coefficient of volumetric thermal expansion

$\overline{\lambda}$	volume averaged thermal conductivity of the fluid-solid mixture
C_p	heat capacity
r	number of dimension
Re	Reynold's number
$j_a(\eta_a)$	anode transfer current density
$j_c(\eta_c)$	cathode transfer current density
R_{agg}	radius of the agglomerate
D_{agg}	diffusion coefficient of the agglomerate
L_{act}	catalyst layer thickness
i_{0a}	exchange current density of the anode reaction
i_{0c}	exchange current densities of the cathode reaction
$C_{\mathrm{O_2},ref}$	reference concentration of oxygen
$C_{\mathrm{H_2},ref}$	reference concentration of hydrogen
$C_{\mathrm{O_2},agg}$	surface concentration of oxygen
$C_{\mathrm{H_2},agg}$	surface concentrations of hydrogen
T	temperature
F	Faraday's constant
R	universal gas constant
$\phi_e(\sigma_e)$	ionic conductivity
$\phi_s(\sigma_s)$	electronic conductivity

$E_{eq,a}$	anode equilibrium potential
$E_{eq,c}$	cathode equilibrium potential

5.1 Statement of the problem

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

The quantity or quantities of interest can include any or all of the u_i, or functions derived from the u_i. For the purposes of exposition, consider a single quantity of interest, denoted simply as $u(\mathbf{x}; \boldsymbol{\xi})$. The simulator provides values of $u(\mathbf{x}; \boldsymbol{\xi})$ at specified (fixed) locations, $\mathbf{x}^{(i)}$, $i = 1, \ldots, d$, 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: $\mathbf{y}^{(j)} = (u(\mathbf{x}^{(1)}; \boldsymbol{\xi}^{(j)}), \ldots, u(\mathbf{x}^{(d)}; \boldsymbol{\xi}^{(j)}))^T \in \mathbb{R}^d$. This process can be repeated for other spatial fields of interest to derive multiple vectorized outputs in $\mathbb{R}^d$. An example of the simultaneous emulation of multiple field outputs is given in Section 5.6. It is assumed for now that a single output $\mathbf{y}$ (derived from a single scalar field $u(\mathbf{x}; \boldsymbol{\xi})$) is the target for emulation.

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

To infer outputs of the simulator at new inputs, Conti and O'Hagan [35] 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.* [43] but in that case the outputs were a linear combination of PCA basis vectors with coefficients treated as independent univariate GPs indexed by $\boldsymbol{\xi}$. In this chapter, a similar approach is adopted but rather than using PCA coefficients, GP priors were placed over coefficients of a reduced-dimensional approximation of points in $\mathcal{M}$, obtained by manifold learning methods. It is assumed that $\mathcal{M}$ is a smooth manifold in $\mathbb{R}^d$. The high dimension d of the output space and the inability of linear dimensionality reduction methods such as PCA to capture complex response surfaces is the motivation for this approach.

5.1.1 Support Vector Machines

An other popular supervised learning technique is the Support Vector Machine (SVM) developed by Vapnik et. al. [59] which was extended by Boser [60] to non-linear tasks by introducing the kernel trick. Assuming a training dataset $D = \{y^i, \boldsymbol{\xi}^i\}_{i=1}^m$, where $y \in \mathbb{R}$ and $\boldsymbol{\xi} \in \mathcal{X} \subset \mathbb{R}^l$, in the epsilon-insensitive SVM regression the function $\eta(\boldsymbol{\xi})$,where $\mathbf{w} \in \mathcal{X}$ and $b \in \mathbb{R}$ has a linear form

$$\eta(\boldsymbol{\xi}) = \mathbf{w}^T\boldsymbol{\xi} + b \tag{5.1}$$

and the aim is to solve a convex optimization problem of the form:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y^{(i)} - \mathbf{w}^T\boldsymbol{\xi}^{(i)} - b \leq \varepsilon \\ & \mathbf{w}^T\boldsymbol{\xi}^{(\boldsymbol{i})} + b - y^{(i)} \leq \varepsilon \end{aligned} \tag{5.2}$$

in order for $\eta(\boldsymbol{\xi})$ to be as flat as possible and at the same time its deviation from $y^{(i)}$ to be less or equal to ε.

Cortes in [150] introduced the 'soft margins' for the SVM in which there are involved two slack variables ζ_i, ζ_i^*. As a result the Equation 5.2 can be rewritten in the form

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}(\zeta_i + \zeta_i^*) \\ \text{subject to} \quad & y^{(i)} - \mathbf{w}^T\boldsymbol{\xi}^{(i)} - b \leq \varepsilon + \zeta_i \\ & \mathbf{w}^T\boldsymbol{\xi}^{(\boldsymbol{i})} + b - y^{(i)} \leq \varepsilon + \zeta_i^* \\ & \zeta_i, \zeta_i^* \geq 0 \end{aligned} \tag{5.3}$$

where C is a constant positive term that controls the values outside the margin ε and also acts as regularization factor that prevents overfitting. In the linear *epsilon*-insensitive loss function the errors that lie within ε distance of the output are treated as zero, hence the loss function measures the values between the output y and the ε boundary:

$$L_\varepsilon = \begin{cases} 0, & \text{if } |y - \eta(\boldsymbol{\xi})| \le \varepsilon \\ |y - \eta(\boldsymbol{\xi})| - \varepsilon & \text{otherwise} \end{cases} \tag{5.4}$$

By using nonnegative Lagrange $\alpha_i, \alpha_i^*, \beta_i, \beta_i^*$ multipliers and forming a Lagrangian function Equation 5.3 can be solved as:

$$\begin{aligned} L = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}(\zeta_i, \zeta_i^*) - \sum_{i=1}^{m}\alpha_i(\varepsilon + \zeta_i - y^{(i)} + \mathbf{w}^T\boldsymbol{\xi} + b) \\ - \sum_{i=1}^{m}\alpha_i^*(\varepsilon + \zeta_i^* + y^{(i)} + \mathbf{w}^T\boldsymbol{\xi} + b) - \sum_{i=1}^{m}(\beta_i\zeta_i, \beta_i^a st\zeta_i^*) \end{aligned} \tag{5.5}$$

Taking the derivatives of the Lagrangian Equation 5.5 with respect to $\mathbf{w}, b, \zeta_i, \zeta_i^*$ and equating to zero results in:

$$\begin{aligned} \frac{\partial L}{\partial b} &= \sum_{i=1}^{m}(\alpha_i^* - \alpha_i) = 0 \\ \frac{\partial L}{\partial w} &= \mathbf{w} - \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)\boldsymbol{\xi}^{(i)} = 0 \\ \frac{\partial L}{\partial \zeta_i} &= C - \alpha - \beta_i \\ \frac{\partial L}{\partial \zeta_i^*} &= C - \alpha_i^* - \beta_i^* \end{aligned} \tag{5.6}$$

Which can also be written in the dual representation form as:

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(\boldsymbol{\xi}^{(i)})^T\boldsymbol{\xi}^{(j)} \\ & - \varepsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)y^{(i)} \\ \text{subject to} \quad & \sum_{i=1}^{m}(\alpha_i - \alpha_i^*) = 0 \\ & \alpha_i, \alpha_i^* \in [0, C] \end{aligned} \tag{5.7}$$

To obtain the optimal solutions the Karush-Kuhn-Tucker complementarity conditions must be taken into consideration:

$$\begin{aligned}
&\alpha_i(\varepsilon + \zeta_i - y^{(i)} + \mathbf{w}^T\boldsymbol{\xi} + b) = 0 \\
&\alpha_i^*(\varepsilon + \zeta_i + j^{(i)} - \mathbf{w}^T\boldsymbol{\xi} - b) = 0 \\
&(C - \alpha_i)\zeta_i = 0 \\
&(C - \alpha_i^*)\zeta_i^* = 0
\end{aligned} \tag{5.8}$$

After these conditions have been met, the computation of b is:

$$\begin{aligned}
b &= y^{(i)} - \mathbf{w}^T\boldsymbol{\xi}^{(i)} - \varepsilon \qquad \text{if} \quad \alpha_i \in (0, C) \\
b &= y^{(i)} - \mathbf{w}^T\boldsymbol{\xi}^{(i)} + \varepsilon \qquad \text{if} \quad \alpha_i^* \in (0, C)
\end{aligned} \tag{5.9}$$

The dual form of SVM involves the observation vectors in the form of inner product. By using the kernel trick [60][151] the original data could be mapped in a feature and compute the inner product making the use of a kernel function. This leads to be able for nonlinear problems to be solved in the feature space instead of the original space. The kernel trick is also used in this thesis for the kPCA and is presented in the following section.

5.2 Manifold learning methods

5.2.1 Diffusion maps

In diffusion maps, the training data $\mathbf{y}^{(i)} \in \mathcal{M} \subset \mathbb{R}^d$, $i = 1, \ldots, m$ are mapped to a subset of $\mathbb{R}^m$ called the *diffusion space* from which a reduced-dimensional approximation is subsequently obtained [152, 153]. The mapping embeds the data points in diffusion space by preserving a *diffusion distance* defined between the points in physical space. The data points $\mathbf{y}^{(i)}$ 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 $\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(m)}\}$ representing the training points. Edge weights are defined by a symmetric and positive definite kernel $\mathsf{k}(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})$ between the data points, e.g., the Gaussian kernel $\mathsf{k}(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \exp(-||\mathbf{y}^{(i)} - \mathbf{y}^{(j)}||^2/s^2)$. It is assumed that $\mathcal{G}$ is connected (otherwise the maps can be constructed separately on each connected component).

A diffusion process [67] on $\mathcal{G}$ is constructed by normalizing the connectivity (adjacency) matrix $\mathsf{K} = [\mathsf{K}_{ij}]$, where $\mathsf{K}_{ij} = \mathsf{k}(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})$. The degree matrix is defined as $\mathbf{D} = \operatorname{diag}(d_1, \ldots, d_m)$, where $d_i = \sum_j \mathsf{K}_{ij}$, and an $m \times m$ *diffusion matrix* is defined by $\mathbf{P} = \mathbf{D}^{-1}\mathsf{K}$. $\mathbf{P} = [P_{ij}]$ is a Markov matrix; the entry P_{ij} is considered to be a transition probability $p(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})$ from node $\mathbf{y}^{(i)}$ to $\mathbf{y}^{(j)}$ in a random walk on $\mathcal{G}$. The corresponding t step transition probability $p_t(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})$ (from $\mathbf{y}^{(i)}$ to $\mathbf{y}^{(j)}$ in $t \in \mathbb{N} = 1, 2, \ldots$ steps) is given by the (i, j)-th entry of $\mathbf{P}^t = \mathbf{P} \times \cdots \times \mathbf{P}$.

Since $\mathcal{G}$ is connected, $\mathbf{P}$ is ergodic and, therefore, possesses a unique stationary distribution $\boldsymbol{\pi}$ with entries $\pi_i = d_i / \sum_j d_j$ [152]. The symmetric matrix $\mathbf{P}' = \mathbf{D}^{-1/2}\mathsf{K}\mathbf{D}^{1/2}$ possesses the same eigenvalues γ'_i as $\mathbf{P}$. A spectral decomposition yields $\mathbf{P}' = \mathbf{S}\boldsymbol{\Gamma}'\mathbf{S}^T$, where the columns of $\mathbf{S}$ are the orthonormal eigenvectors $\mathbf{s}_i$, $i = 1, \ldots, m$, of $\mathbf{P}'$ and $\boldsymbol{\Gamma}' = \operatorname{diag}(\gamma'_1, \ldots, \gamma'_m)$. The eigenvalues are arranged such that $1 = \gamma'_1 > \cdots > \gamma'_m$ and the eigenvector $\mathbf{s}_1$ has entries $\sqrt{\pi_i}$ [154]. $\mathbf{P}$ has the spectral decomposition $\mathbf{P} = \mathbf{Q}\boldsymbol{\Gamma}'\mathbf{Q}^{-1}$, where $\mathbf{Q} = \mathbf{D}^{-1/2}\mathbf{S}$. The right and left eigenvectors of $\mathbf{P}$ are $\mathbf{r}_i = \mathbf{D}^{-1/2}\mathbf{s}_i$ and $\mathbf{l}_i = \mathbf{D}^{1/2}\mathbf{s}_i$, respectively. Therefore $\mathbf{l}_1 = \boldsymbol{\pi}\sqrt{\sum_j d_j}$ and $\mathbf{r}_1 = \mathbf{1}^T / \sqrt{\sum_j d_j}$. The right and left eigenvectors are bi-orthogonal, i.e., $\mathbf{l}_i^T \mathbf{r}_i = \delta_{ij}$, where δ_{ij} is the Kronecker delta. By the orthogonality of $\mathbf{S}$, $\mathbf{P}^t = \mathbf{Q}\boldsymbol{\Gamma}^t\mathbf{Q}^{-1}$, or $\mathbf{P}^t = \sum_{i=1}^m (\gamma'_i)^t \mathbf{r}_i \mathbf{l}_i^T$. The j-th row vector of $\mathbf{P}^t$, denoted $\mathbf{p}_j^t$, is:

$$\mathbf{p}_j^t = (p_t(\mathbf{y}^{(j)}, \mathbf{y}^{(1)}), \ldots, p_t(\mathbf{y}^{(j)}, \mathbf{y}^{(m)}))^T = \sum_{i=1}^{m} (\gamma'_i)^t r_{ji} \mathbf{l}_i, \tag{5.10}$$

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

A diffusion distance D_t (in physical space) is then defined as follows [152]:

$$D_t(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \left((\mathbf{p}_i^t - \mathbf{p}_j^t)^T \mathbf{D}^{-1} (\mathbf{p}_i^t - \mathbf{p}_j^t)\right)^{1/2}. \tag{5.11}$$

Now a family of diffusion maps $\boldsymbol{\psi}^t : \mathcal{M} \to \mathcal{D}^{(t)} \subset \mathbb{R}^m$ can be defined between the training points $\mathbf{y}^{(j)}$ and diffusion spaces $\mathcal{D}^{(t)}$ as follows [152, 153]:

$$\boldsymbol{\psi}^t(\mathbf{y}^{(j)}) = \left((\gamma_1')^t r_{j1}, \ldots, (\gamma_m')^t r_{jm}\right)^T. \tag{5.12}$$

The maps are indexed by the free parameter t. The coefficients of a mapped point $\mathbf{y}^{(j)}$ are the coefficients of $\mathbf{p}_j^t$ in the non-orthogonal basis $\{\mathbf{l}_i\}_{i=1}^m$. Diffusion maps embed the data points in $\mathcal{D}^{(t)}$ in the following sense [152, 153, 155]:

$$||\boldsymbol{\psi}^t(\mathbf{y}^{(i)}) - \boldsymbol{\psi}^t(\mathbf{y}^{(j)})|| = D_t(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}), \tag{5.13}$$

where $||\cdot||$ denotes the standard Euclidean norm. Equation (5.13) follows from the bi-orthogonality of the left and right eigenvectors. From Eq. (5.12) and the decay in the eigenvalues, mappings $\boldsymbol{\psi}_r^t(\mathbf{y}^{(j)}) : \mathcal{M} \to \mathcal{D}_r^{(t)} \subset \mathbb{R}^r$ are defined as follows:

$$\boldsymbol{\psi}_r^t(\mathbf{y}^{(j)}) = ((\gamma_1')^t r_{j1}, \ldots, (\gamma_r')^t r_{jr})^T, \tag{5.14}$$

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

In practice, the value of r is usually selected according to a criterion on the eigenvalues, e.g., as the largest index j such that $|(\gamma_j')^t| > \upsilon|(\gamma_2')^t|$ holds for a pre-selected precision υ [152]. 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. (5.10)). Algorithm 2 summarizes diffusion maps for data $\{\mathbf{y}^{(i)}\}_{i=1}^m$.

In order to develop an inverse map approximation, diffusion maps are generalized to all points in $\mathcal{M}$ by taking the limit $m \to \infty$. In this limit, the random walk on the discrete graph using a Gaussian kernel converges to a discrete-time walk on the continuous state space $\mathcal{M}$ [152, 153, 155, 156]. Let μ be a probability measure on $\mathcal{M}$ defining the density of points. In the limit $m \to \infty$, a one-step transition kernel for the Markov chain on $\mathcal{M}$ can be defined by $\mathbb{p}(\mathbf{y}', \mathbf{y}) = \mathsf{k}(\mathbf{y}, \mathbf{y}')/\mathbb{d}(\mathbf{y}')$, from an arbitrary $\mathbf{y}' \in \mathcal{M}$ to an arbitrary $\mathbf{y} \in \mathcal{M}$, where $\mathbb{d}(\mathbf{y}') = \int_{\mathcal{M}} \mathsf{k}(\mathbf{y}, \mathbf{y}') d\mu(\mathbf{y})$. A corresponding forward transfer operator is defined by $\mathcal{L}\varphi(\mathbf{y}) = \int_{\mathcal{M}} \mathbb{p}(\mathbf{y}', \mathbf{y})\varphi(\mathbf{y}') d\mu(\mathbf{y}')$ for $\varphi(\mathbf{y}) \in L^2(\mathcal{M}, \mu)$. 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

Algorithm 2 Diffusion maps

```
1a: Form a kernel matrix $\mathsf{K}$ using a kernel function $\mathsf{k}(\cdot,\cdot)$.
        Degree of node $i$: $d_i \leftarrow \sum_j \mathsf{K}_{ij} \quad i = 1,\ldots,m$.
        Degree matrix: $\mathbf{D} \leftarrow \mathrm{diag}(d_1,\ldots,d_m)$.
        $\mathbf{P}' \leftarrow \mathbf{D}^{-1/2}\mathsf{K}\mathbf{D}^{1/2}$.
2a: Eigenvalue problem: $\mathbf{P}'\mathbf{s} = \gamma\mathbf{s} \to (\mathbf{s}_i, \gamma_i')$, $i = 1,\ldots,m$.
        $\mathbf{r}_i \leftarrow \mathbf{D}^{-1/2}\mathbf{s}_i \quad \text{and} \quad \mathbf{l}_i \leftarrow \mathbf{D}^{1/2}\mathbf{s}_i$.
3a: Select $r$ as the largest index $j$ such that $|(\gamma_j')^t| > \upsilon|(\gamma_2')^t|$ for a precision $\upsilon$:
        $\boldsymbol{\psi}_r^t(\mathbf{y}^{(j)}) \leftarrow ((\gamma_1')^t r_{j1},\ldots,(\gamma_r')^t r_{jr})^T \quad j = 1,\ldots,m$.
```

$\mathbb{p}_t(\mathbf{y},\mathbf{y}')$. A backward transfer operator $\mathcal{R}\varphi(\mathbf{y}) = \int_{\mathcal{M}} \mathbb{p}(\mathbf{y},\mathbf{y}')\varphi(\mathbf{y}')d\mu(\mathbf{y}')$ can be similarly defined , which is the analogue of multiplication of $\mathbf{P}$ from the right.

The kernel $\mathbb{p}_t(\mathbf{y},\mathbf{y}')$ admits the decomposition $\mathbb{p}_t(\mathbf{y},\mathbf{y}') = \sum_{i=1}^{\infty} \gamma_i^t r_i(\mathbf{y}) l_i(\mathbf{y}')$, where γ_i, $r_i(\mathbf{y})$ and $l_i(\mathbf{y})$ are the (common) eigenvalues and eigenfunctions of $\mathcal{L}$ and $\mathcal{R}$, respectively. They are, respectively, the continuous-space equivalents of γ_i', $\mathbf{r}_i$ and $\mathbf{l}_i$. Moreover $1 = \gamma_1 > \gamma_2 > \cdots$.

For a fixed $\mathbf{y} \in \mathcal{M}$, $\mathbb{p}_t(\mathbf{y},\mathbf{y}')$ is the continuous version (a probability density in $\mathbf{y}' \in \mathcal{M}$) of the probability mass function defined by Eq. (5.10); in the latter case, $\mathbf{y} = \mathbf{y}^{(j)}$ and $\mathbf{y}' \in \{\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(m)}\}$, i.e., the finite set of states accessible from $\mathbf{y}^{(j)}$. The j-th components of $\mathbf{r}_i$ and $\mathbf{l}_i$ are, respectively, approximations of $r_i(\mathbf{y}^{(j)})$ and $l_i(\mathbf{y}^{(j)})$ based on the training data. The diffusion distances between any two points $\mathbf{y},\mathbf{y}' \in \mathcal{M}$ are given by $D_t = ||\mathbb{p}_t(\mathbf{y},\mathbf{y}') - \mathbb{p}_t(\mathbf{y},\mathbf{y}')||_{1/\mathbb{d}}$, where $||\varphi||_{1/\mathbb{d}}^2 = \int_{\mathbf{y}'\in\mathcal{M}} |\varphi(\mathbf{y}')|^2/\mathbb{d}(\mathbf{y}')d\mu(\mathbf{y}')$ for functions $\{\varphi : ||\varphi||_{1/\mathbb{d}} < \infty\}$. In turn, diffusion maps $\boldsymbol{\psi}^t : \mathcal{M} \to \mathcal{D}^{(t)} \subset \ell^2$ are defined on the whole space $\mathcal{M}$ by $\boldsymbol{\psi}^t(\mathbf{y}) = (\gamma_1^t r_1(\mathbf{y}), \gamma_2^t r_2(\mathbf{y}),\ldots)$. Here, ℓ^2 denotes the space of sequences $\{(\mathrm{x}_1,\mathrm{x}_2\ldots) : \sum_{j=1}^{\infty} \mathrm{x}_j^2 < \infty\}$. Truncating the expansion of $\mathbb{p}_t$ at the first r terms leads to r-dimensional approximations of the diffusion maps $\boldsymbol{\psi}_r^t : \mathcal{M} \to \mathcal{D}_r^{(t)} \subset \mathbb{R}^r$, i.e., $\boldsymbol{\psi}_r^t(\mathbf{y}) = (\gamma_1^t r_1(\mathbf{y}),\ldots,\gamma_r^t r_r(\mathbf{y}))^T$.

Given an isotropic kernel $\mathsf{k}(\mathbf{y},\mathbf{y}')$, diffusion maps can be generalized by defining a family of anisotropic kernels $\mathsf{k}^{(\alpha)}(\mathbf{y},\mathbf{y}') = \mathsf{k}(\mathbf{y},\mathbf{y}')/(\mathbb{d}(\mathbf{y}')^{\alpha}\mathbb{d}(\mathbf{y})^{\alpha})$, for $\alpha \in \mathbb{R}$, and normalizing the resulting kernel to generalize $\mathbb{p}(\mathbf{y}',\mathbf{y})$ (or $\mathbf{P}$ in the discrete case) [152, 157, 158]. The standard algorithm described above corresponds to the limiting case of $\alpha = 0$ (isotropic kernel), and anisotropic kernels are not considered in this chapter due to the lack of inverse mappings for such special cases. In Section 5.5.2, a new inverse map for the isotropic case only is described.

Remark 2. *It is possible to instead consider the mappings* $(\boldsymbol{\psi}_r^t \circ \boldsymbol{\eta})(\cdot) = \boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \to \mathcal{D}_r^{(t)} \subset \mathbb{R}^r$ *that map* all *points in the input space to* $\mathcal{D}_r^{(t)}$*. The mapped point is given by the first* r *coordinates of the transition kernel* $\mathbb{p}_t(\mathbf{y}, \mathbf{y}')$ *(considering* $\mathbf{y}$ *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* $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$*. Composite functions* $\mathbb{r}_i(\cdot) = r_i(\boldsymbol{\eta}(\cdot)) : \mathcal{X} \to \mathbb{R}$ *are defined to obtain:*

$$\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi})) = (\gamma_1^t \mathbb{r}_1(\boldsymbol{\xi}), \ldots, \gamma_r^t \mathbb{r}_r(\boldsymbol{\xi}))^T \in \mathcal{D}_r^{(t)}. \tag{5.15}$$

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)* $\mathbf{l}_i$ *and* $\mathbf{r}_i$*.*

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* $\{\mathbf{y}^{(i)}\}_{i=1}^m$ *yields the new training points for emulation:* $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})) = \boldsymbol{\psi}_r(\mathbf{y}^{(j)}) = ((\gamma_1')^t r_{j1}, \ldots, (\gamma_r')^t r_{jr})^T$*,* $j = 1, \ldots, m$*, obtained from the empirical eigenfunctions and empirical eigenvalues.*

5.3 Emulation of coefficients in reduced-dimensional approximations

As explained in Remarks 1 and 2, rather than emulating the outputs in $\mathcal{M}$ directly, multivariate GP priors are placed over the reduced-dimensional representations in $\mathcal{F}_r$ or $\mathcal{D}_r^{(t)}$. In the actual approach described in Section 5.4, univariate GP priors are placed over the individual coefficients $\mathbb{r}_i(\cdot)$ or $\mathbb{z}_i(\cdot)$ and these coefficients are emulated separately. In this section, scalar GPE is therefore outlined.

A scalar valued simulator is a function $\eta : \mathcal{X} \to \mathbb{R}$ of inputs $\boldsymbol{\xi} \in \mathcal{X} \subset \mathbb{R}^l$. In univariate GPE, a GP prior indexed by $\boldsymbol{\xi} \in \mathcal{X}$ is placed over $\eta(\boldsymbol{\xi})$ and the emulator is trained using simulator outputs $\eta(\boldsymbol{\xi}^{(i)})$ at design points $\boldsymbol{\xi}^{(i)}$.

The notation $\mathbf{t} = (\eta(\boldsymbol{\xi}^{(1)}), \ldots, \eta(\boldsymbol{\xi}^{(m)}))^T$ is used. The prior is $\eta(\boldsymbol{\xi})|\boldsymbol{\theta}, \boldsymbol{\beta} \sim \mathcal{GP}\left(m(\boldsymbol{\xi}), \mathbb{c}(\boldsymbol{\xi}, \boldsymbol{\xi}')\right)$, where $\mathcal{GP}\left(m(\cdot), \mathbb{c}(\cdot, \cdot)\right)$ represents a GP with mean and covariance functions $m(\cdot)$ and $\mathbb{c}(\cdot, \cdot)$, respectively. The most common choices for the mean function are a linear function or a constant. In this work, $m \equiv 0$ was assumed by centering the data. $\boldsymbol{\theta}$ is a vector of hyperparameters (e.g., parameters in the covariance function) that are typically unknown *a priori*.

Remark 3. *A GP noise term can be added to the model, in which case $\eta(\boldsymbol{\xi})$ is a latent function while the simulator outputs are the observables: $t(\boldsymbol{\xi}) = \eta(\boldsymbol{\xi}) + \epsilon(\boldsymbol{\xi})$, in which $\epsilon(\boldsymbol{\xi}) \sim \mathcal{GP}(0, \sigma_n^2 \delta(\boldsymbol{\xi}, \boldsymbol{\xi}'))$, where $\delta(\cdot, \cdot)$ is the Kronecker delta. The noise can represent modelling or simulation errors or can be included for numerical stability. It can be included directly as an additional term in the covariance function $\mathbb{c}(\boldsymbol{\xi}, \boldsymbol{\xi}')$ (a so called 'jitter' or 'nugget' [159]), which leads to the same result for GP priors over the noise and latent function.*

A square exponential covariance function is used:

$$\mathbb{c}(\boldsymbol{\xi}, \boldsymbol{\xi}') = \theta_0 \exp\left(-(\boldsymbol{\xi} - \boldsymbol{\xi}')^T \mathrm{diag}(\theta_1, \ldots, \theta_l)(\boldsymbol{\xi} - \boldsymbol{\xi}')\right) + \sigma_n^2 \delta(\boldsymbol{\xi}, \boldsymbol{\xi}'), \tag{5.16}$$

where the last term is the jitter, and $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_l, \sigma_n^2)^T$. The parameters $\theta_1, \ldots, \theta_l$ are the inverse square correlation lengths. Alternatives to Eq. (5.16) include the Matérn class of functions and piecewise polynomials, which are also stationary [58].

The conditional predictive distribution at new inputs $\boldsymbol{\xi}$ is obtained in a straightforward manner from the joint distribution $p(\eta(\boldsymbol{\xi}), \mathbf{t}|\boldsymbol{\theta})$ [58]:

$$\begin{gathered} \eta(\cdot)|\mathbf{t}, \boldsymbol{\theta} \sim \mathcal{GP}\left(m'(\cdot; \boldsymbol{\theta}), \nu'(\cdot, \cdot; \boldsymbol{\theta})\right), \\ m'(\boldsymbol{\xi}; \boldsymbol{\theta}) = \boldsymbol{c}(\boldsymbol{\xi})^T \mathbf{C}^{-1} \mathbf{t} \quad \text{and} \quad \nu'(\boldsymbol{\xi}, \boldsymbol{\xi}'; \boldsymbol{\theta}) = \mathbb{c}(\boldsymbol{\xi}, \boldsymbol{\xi}') - \boldsymbol{c}(\boldsymbol{\xi})^T \mathbf{C}^{-1} \boldsymbol{c}(\boldsymbol{\xi}'), \end{gathered} \tag{5.17}$$

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

The hyperparameters $\boldsymbol{\theta}$ are unknown. Point estimates [19, 160] such as the maximum likelihood estimate (MLE) are employed in most cases; that is, the predictive distribution is given by Eq. (5.17) using the MLE estimate. The MLE is given by $\arg\max_{\boldsymbol{\theta}} \mathcal{R}(\boldsymbol{\theta})$, where $\mathcal{R}(\boldsymbol{\theta}) = \log p(\mathbf{t}|\boldsymbol{\theta})$ is the log likelihood:

$$\mathcal{R}(\boldsymbol{\theta}) = -\frac{1}{2} \ln |\mathbf{C}| - \frac{1}{2} \mathbf{t}^T \mathbf{C}^{-1} \mathbf{t} - \frac{m}{2} \ln(2\pi). \tag{5.18}$$

In a Bayesian inference approach, predictions at a new input $\boldsymbol{\xi}$ are made by integrating over $\boldsymbol{\theta}$ in the joint distribution of $\boldsymbol{\theta}$ and $\eta(\boldsymbol{\xi})$ given $\mathbf{t}$ (the posterior predictive distribution). The integral is analytically intractable but can be approximated using Monte Carlo integration, e.g., importance sampling, or Markov Chain Monte Carlo [161] to sample from the posterior over the hyperparameters $p(\boldsymbol{\theta}|\mathbf{t})$.

5.4 Multi-output emulation using manifold learning

The problem of emulating $\boldsymbol{\eta} : \mathcal{X} \to \mathcal{M}$ has been replaced with the problem of emulating the map $\mathbf{z}_r(\boldsymbol{\eta}(\cdot))$ defined by Eq. (2.8) or the map $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\cdot))$ defined by Eq. (5.15). Multivariate GP priors are placed over these maps, with training points for emulation given by Algorithms 1 and 2 for kPCA and diffusion maps, respectively. These multivariate GP priors take a particularly convenient form by assuming independence of the coordinates, as explained below.

The kPCA coefficients, $\mathbb{z}_i(\boldsymbol{\xi})$, $i = 1, \ldots, r$ are mutually uncorrelated; following Higdon *et al.* [43] (see also the wavelet decomposition approach in [44]) the approximation is therefore made that they arise from independent GPs. The diffusion map coefficients $\gamma_i \mathbb{r}_i(\boldsymbol{\xi})$, $i = 1, \ldots, r$, on the other hand, are not uncorrelated. As a simplification, however, the underlying GPs are treated as independent (see Remark 4). For both manifold learning methods, univariate GPE is then performed separately on each coefficient to approximate its value for a new input $\boldsymbol{\xi}$. The process is summarized below for each case, making clear the link between the notation of Sections 5.2 and 5.3.

1. **kPCA:** For a fixed $i = 1, \ldots, r$, $\eta(\boldsymbol{\xi}) = \mathbb{z}_i(\boldsymbol{\xi})$ is set. The training points are given by Eq. (2.6): $\eta(\boldsymbol{\xi}^{(j)}) = \mathbb{z}_i(\boldsymbol{\xi}^{(j)}) = \widetilde{\boldsymbol{\alpha}}_i^T \mathbf{H}(\mathbf{k}_j - \mathbf{K}\mathbf{1})$, $j = 1, \ldots, m$. Recall that $\mathbb{z}_i(\boldsymbol{\xi}^{(j)}) = z_i(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})) = z_i(\mathbf{y}^{(j)})$. The expected (mean) value at an input $\boldsymbol{\xi}$, given by Eq. (5.17), yields a prediction that is denoted $\mathbb{z}_i(\boldsymbol{\xi})$ (to avoid introducing new notation, there is no distinguish between $\mathbb{z}_i(\boldsymbol{\xi})$ and $\mathbb{E}[\mathbb{z}_i(\boldsymbol{\xi})]$). Set $\mathbf{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi})) = (\mathbb{z}_1(\boldsymbol{\xi}), \ldots, \mathbb{z}_r(\boldsymbol{\xi}))^T$. Again, this is the expected value $\mathbb{E}[\mathbf{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi}))]$.

2. **Diffusion maps:** For a fixed $i = 1, \ldots, r$, $\eta(\boldsymbol{\xi}) = \mathbb{r}_i(\boldsymbol{\xi})$ is set. The train-

ing points are given by Eq. (5.14): $\eta(\boldsymbol{\xi}^{(j)}) = \mathbb{r}_i(\boldsymbol{\xi}^{(j)}) = r_{ji}$, $j = 1, \ldots, m$. Recall that $\mathbb{r}_i(\boldsymbol{\xi}^{(j)}) = r_i(\boldsymbol{\eta}(\boldsymbol{\xi})^{(j)})) = r_i(\mathbf{y}^{(j)})$. For a new input $\boldsymbol{\xi}$, Eq. (5.17) 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 2 are used to compute the predicted value of $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi}))$.

Remark 4. *To take account of the correlations between the coefficients when using diffusion maps, the linear model of coregionalization (LMC) [36, 162] could be used to emulate the coefficients simultaneously. Alternatively, the GP model could be replaced by an artificial neural network (ANN). For moderately sized r, neither approach is computationally expensive. In this chapter, the approach of univariate GPs is compared with ANN using Bayesian regularization [107, 108].*

To complete the emulation, the inverse map must be approximated 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 5.5.3.

Remark 5. *The GPE framework furnishes predictive variances, given by Eq. (5.17). 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 $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi}) \in \mathcal{M}$. The inverse maps discussed below provide only the predictive means of the points $\mathbf{y}$. However, Monte Carlo (MC) estimates of higher-order statistics can be derived for a fixed input $\boldsymbol{\xi}$ by drawing samples from the posterior predictive Gaussian distribution (defined by Eq. (5.17)) over the coefficients $r_i(\mathbf{y}) = \mathbb{r}_i(\xi)$ or $z_i(\mathbf{y}) = \mathbb{z}_i(\boldsymbol{\xi})$ and using the deterministic inverse maps described below.*

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

The final step is to find approximations of the inverse mappings $\boldsymbol{\phi}_r^{-1}(\cdot) : \mathcal{F}_r \to \mathcal{M}$ and $(\boldsymbol{\psi}_r^t)^{-1}(\cdot) : \mathcal{D}_r^{(t)} \to \mathcal{M}$ for kPCA and diffusion maps, respectively. Note that these are the inverse mappings for the manifold learning methods (from the reduced dimensional space to physical space) and not the inverse mappings for the composite functions $\boldsymbol{\phi}_r(\boldsymbol{\eta}(\cdot))$ and $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\cdot))$. In practical terms (since the feature map is unknown), for kPCA the mapping $\mathbf{z}_r^{-1}(\cdot) : \mathbb{R}^r \to \mathcal{M}$, or $\mathbf{z}_r \mapsto \mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$ is sought. This can be achieved *via* a closed-form least-squares solution [163, 164]. 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.* [165] and other minimization routines.

For diffusion maps there has been little progress towards finding an inverse map approximation. Etyngier *et al.* [166] proposed an optimization procedure designed for 2-d shapes embedded in $\mathbb{R}^3$ (a closely related method can be found in [167]). 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 $\boldsymbol{\psi}_r^t(\mathbf{y}) = \boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi}))$ to be the mapped nearest $r+1$ neighbours of $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$ in $\mathcal{M}$. It then proceeds to minimize over the point $\mathbf{y}$ and its barycentric coordinates w.r.t. its $r+1$ closest neighbours. For large values of r and d, in particular for $d \gg m$, this procedure will be highly unstable and computationally expensive.

Given the reduced-dimensional representation $\mathbf{z}_r(\mathbf{y})$ or $\boldsymbol{\psi}_r^t(\mathbf{y})$ of an unknown point $\mathbf{y}$, a general method for finding the pre-image is to use a weighted average of N_n neighbouring (in some well defined sense) points of $\mathbf{y}$. 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 $\{\mathbf{y}^{(i)}\}_{i=1}^m$. The weighted average can be written as follows:

$$\mathbf{y} = \sum_{j \in \mathcal{J}} \vartheta(\mathbf{y}^{(j)})\mathbf{y}^{(j)}, \tag{5.19}$$

in which the weight $\vartheta(\mathbf{y}^{(j)})$ is associated with the data point $\mathbf{y}^{(j)}$, $j \in \mathcal{J}$, 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 $\mathbf{y}$ and the data points $\mathbf{y}^{(i)}$, $i = 1, \ldots, m$. The simplest approach, known as local linear interpolation [168, 104] is to take $\vartheta(\mathbf{y}^{(j)}) = d_{j,*}^{-1} / \sum_{j=1}^{m} d_{j,*}^{-1}$ and to select the index set $\mathcal{J}$ according to the N_n points of $\{\mathbf{y}^{(j)}\}_{i=1}^{m}$ with the largest values of $\vartheta(\mathbf{y}^{(j)})$. A generalization of this approach uses an isotropic kernel density $\chi(\mathbf{y}, \mathbf{y}') = \chi(||\mathbf{y} - \mathbf{y}'||)$ to weight the samples [169]:

$$\vartheta(\mathbf{y}^{(j)}) = \frac{\chi(\mathbf{y}, \mathbf{y}^{(j)})}{\sum_{i=1}^{m} \chi(\mathbf{y}, \mathbf{y}^{(i)})} = \frac{\chi(d_{j,*})}{\sum_{i=1}^{m} \chi(d_{i,*})}, \tag{5.20}$$

The particular form of kernel density used in this chapter is $\chi(\mathbf{y}, \mathbf{y}') = \exp(-||\mathbf{y} - \mathbf{y}'||^2)$, 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 $\mathbf{y}$ and the training points $\mathbf{y}^{(i)}$. 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 $\mathcal{M}$.

5.5.1 Kernel PCA

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

$$\begin{aligned} \boldsymbol{\phi}_r(\mathbf{y}) = \sum_{i=1}^{r} z_i \widetilde{\mathbf{w}}_i + \overline{\boldsymbol{\phi}} &= \sum_{i=1}^{r} z_i \Phi\mathbf{H}\widetilde{\boldsymbol{\alpha}}_i + \Phi\mathbf{1} \\ &= \Phi\left(\mathbf{H}[\widetilde{\boldsymbol{\alpha}}_1 \ldots, \widetilde{\boldsymbol{\alpha}}_r]\mathbf{z}_r + \mathbf{1}\right) = \Phi\boldsymbol{\tau}. \end{aligned} \tag{5.21}$$

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

$$\widetilde{d}_{i,*}^2 = \boldsymbol{\phi}(\mathbf{y})^T \boldsymbol{\phi}(\mathbf{y}) + \boldsymbol{\phi}(\mathbf{y}^{(i)})^T \boldsymbol{\phi}(\mathbf{y}^{(i)}) - 2\boldsymbol{\phi}(\mathbf{y})^T \boldsymbol{\phi}(\mathbf{y}^{(i)}). \tag{5.22}$$

Taking $\boldsymbol{\phi}(\mathbf{y}) \approx \boldsymbol{\phi}_r(\mathbf{y})$ and substituting Eq. (5.21) into Eq. (5.22) yields:

$$\widetilde{d}_{i,*}^2 \approx \boldsymbol{\tau}^T \mathbf{K} \boldsymbol{\tau} + \Bbbk(\mathbf{y}^{(i)}, \mathbf{y}^{(i)}) - 2\boldsymbol{\tau}^T \mathbf{k}_i, \tag{5.23}$$

with $\boldsymbol{\tau}$ defined as in Eq. (5.21). Note that $\Phi^T\Phi = \mathbf{K}$ and $\mathbf{k}_i = \Phi^T \boldsymbol{\phi}(\mathbf{y}^{(i)})$. For an isotropic kernel normalized such that $\Bbbk(\mathbf{y}', \mathbf{y}') = 1$, Eq. (5.22) gives $\widetilde{d}_{i,*}^2 = 2 - 2\Bbbk(\mathbf{y}^{(i)}, \mathbf{y})$, which, equating to the right hand side of Eq. (5.23), yields $\Bbbk(\mathbf{y}^{(i)}, \mathbf{y})$. For the Gaussian kernel, therefore, $d_{i,*}^2 = -s^2 \ln \Bbbk(\mathbf{y}^{(i)}, \mathbf{y})$ is obtained. Similar relationships exist for other commonly used kernel functions [170], e.g., the polynomial kernel $\Bbbk_n(\mathbf{y}, \mathbf{y}') = \left(\mathbf{y}^T\mathbf{y}' + c\right)^n$, $c \in \mathbb{R}$, $n = \mathbb{N}$. In combination with Eqs. (5.19) and (5.20), the values of $d_{i,*}$ yield an approximation of $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$.

5.5.2 Diffusion maps

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

$$\underline{\mathsf{K}} = \left[\begin{array}{c|c} \mathsf{K} & (\mathsf{k}(\mathbf{y}^{(1)}, \mathbf{y}), \dots, \mathsf{k}(\mathbf{y}^{(m)}, \mathbf{y}))^T \\ \hline (\mathsf{k}(\mathbf{y}^{(1)}, \mathbf{y}), \dots, \mathsf{k}(\mathbf{y}^{(m)}, \mathbf{y})) & \mathsf{k}(\mathbf{y}, \mathbf{y}) \end{array}\right]. \tag{5.24}$$

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

$$\underline{\mathbf{D}} = \left[\begin{array}{c|c} \widehat{\mathbf{D}} & 0 \\ \hline 0 & \mathsf{k}(\mathbf{y}, \mathbf{y}) + \sum_j \mathsf{k}(\mathbf{y}^{(j)}, \mathbf{y}) \end{array}\right], \tag{5.25}$$

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

$$\underline{\mathbf{P}} = \left[\begin{array}{c|c} \widehat{\mathbf{D}}^{-1}\mathbf{K} & \widehat{\mathbf{D}}^{-1}(\mathsf{k}(\mathbf{y}^{(1)}, \mathbf{y}), \ldots, \mathsf{k}(\mathbf{y}^{(m)}, \mathbf{y}))^T \\ \hline \dfrac{(\mathsf{k}(\mathbf{y}^{(1)}, \mathbf{y}), \ldots, \mathsf{k}(\mathbf{y}^{(m)}, \mathbf{y}))}{\mathsf{k}(\mathbf{y}, \mathbf{y}) + \sum_j \mathsf{k}(\mathbf{y}^{(j)}, \mathbf{y})} & \dfrac{\mathsf{k}(\mathbf{y}, \mathbf{y})}{\mathsf{k}(\mathbf{y}, \mathbf{y}) + \sum_j \mathsf{k}(\mathbf{y}^{(j)}, \mathbf{y})} \end{array} \right]. \tag{5.26}$$

The $(m+1)$-st row vector of $\underline{\mathbf{P}}$ is denoted $\underline{\mathbf{p}}_{m+1}$. The i-th entry in $\underline{\mathbf{p}}_{m+1}$ is the transition probability from $\mathbf{y}$ to $\mathbf{y}^{(i)}$, $i = 1, \ldots, m$ (the last entry is the transition probability from $\mathbf{y}$ to $\mathbf{y}$). From the discussion in Section 5.2.1, it is known that the i-th entry of $\underline{\mathbf{p}}_{m+1}$ approximates (based on the finite set $\{\mathbf{y}^{(i)}\}_{i=1}^m$) the value of the transition kernel $\mathbb{p}(\mathbf{y}, \mathbf{y}') = \sum_{j=1}^{\infty} \gamma_j r_j(\mathbf{y}) l_i(\mathbf{y}')$ with $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$ fixed, and with $\mathbf{y}' = \mathbf{y}^{(i)}$; the last entry is the value at $\mathbf{y}' = \mathbf{y}$. Thus:

$$\begin{aligned} \underline{\mathbf{p}}_{m+1} &\approx \sum_{j=1}^{\infty} \gamma_j r_j(\mathbf{y}) (l_j(\mathbf{y}^{(1)}), \ldots, l_j(\mathbf{y}^{(m)}), l_j(\mathbf{y}))^T \\ &\approx \sum_{j=1}^{r} \gamma_j r_j(\mathbf{y}) (l_j(\mathbf{y}^{(1)}), \ldots, l_j(\mathbf{y}^{(m)}), l_j(\mathbf{y}))^T, \end{aligned} \tag{5.27}$$

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

$$\boldsymbol{\psi}_r(\mathbf{y}) = (\gamma_1' \mathbb{r}_1(\boldsymbol{\xi}), \ldots, \gamma_r' \mathbb{r}_r(\boldsymbol{\xi}))^T = (\gamma_1' r_1(\mathbf{y}), \ldots, \gamma_r' r_r(\mathbf{y}))^T. \tag{5.28}$$

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

$$\underline{p}_{m+1,i} \approx \sum_{j=1}^{r} \gamma_j' \mathbb{r}_j(\boldsymbol{\xi}) l_{ij}, \quad i = 1, \ldots, m. \tag{5.29}$$

Equating this expression with that of the equivalent entry in Eq. (5.26), the following it is obtained:

$$\sum_{j=1}^{r} \gamma_j' \mathbb{r}_j(\boldsymbol{\xi}) l_{ij} = \frac{\mathsf{k}(\mathbf{y}^{(i)}, \mathbf{y})}{\mathsf{k}(\mathbf{y}, \mathbf{y}) + \sum_{j=1}^{m} \mathsf{k}(\mathbf{y}^{(j)}, \mathbf{y})}, \quad i = 1, \ldots, m. \tag{5.30}$$

For a Gaussian kernel $\mathsf{k}(\mathbf{y}, \mathbf{y}) = 1$, so solving the system of m equations above yields the unknown kernel values $\mathsf{k}(\mathbf{y}^{(i)}, \mathbf{y})$, $i = 1, \ldots, m$. The Euclidean distances $d_{i,*}$ are recovered from the kernel values. For a Gaussian kernel, $d_{i,*}^2 = -s^2 \ln \mathsf{k}(\mathbf{y}^{(i)}, \mathbf{y})$. In combination with Eqs. (5.19) and (5.20), these values of $d_{i,*}$ yield an approximation of $\mathbf{y} = \boldsymbol{\eta}(\boldsymbol{\xi})$.

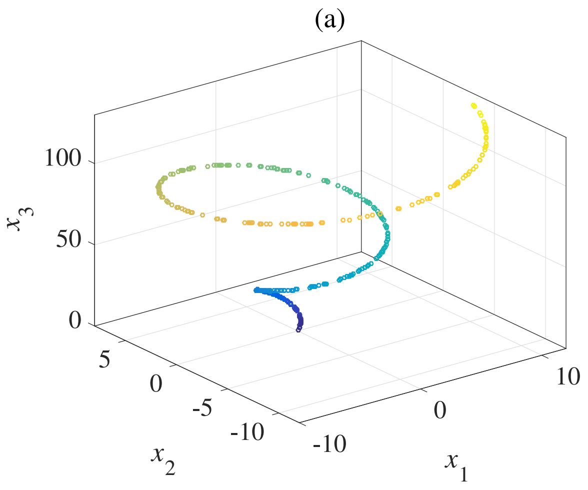
The results of this inverse map approximation on a conical spiral are illustrated in Fig. 5.1. A conical spiral is a 1-d manifold embedded in 3-d, and is defined by the following equations:

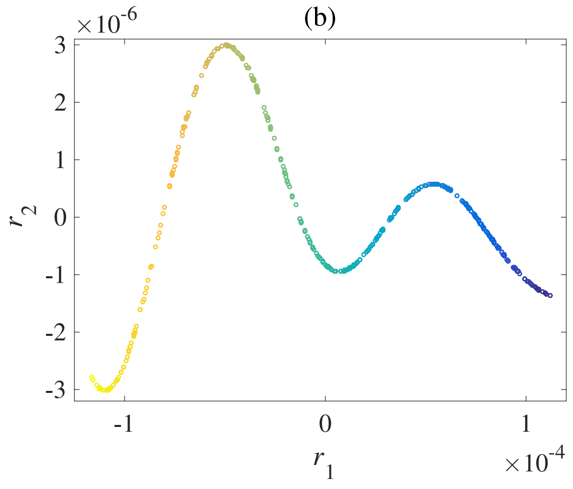
$$x_1 = 4\pi\mathsf{t}\cos(4\pi\mathsf{t}), \quad x_2 = 4\pi\mathsf{t}\sin(4\pi\mathsf{t}), \quad x_3 = 40\pi\mathsf{t}, \tag{5.31}$$

for a single variable $\mathsf{t} \in \mathbb{R}$. A total of 500 points were sampled from the spiral by sampling 500 values of t from a standard uniform distribution $\mathcal{U}(0, 1)$. Figure 5.1(a) shows the sampled points, Fig. 5.1(b) shows the 2-d ($r = 2$) approximation of the points using diffusion maps, and Fig. 5.1(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 were used [140], as detailed in Section 5.6.1. Similarly accurate results were obtained for other standard test sets, e.g., the swiss roll and a Gaussian surface.

5.5.3 Main algorithm

The proposed procedure for GPE of outputs in high-dimensional spaces is summarized in the pseudocode Algorithm 1, based on a Gaussian kernel for both kPCA and diffusion maps.





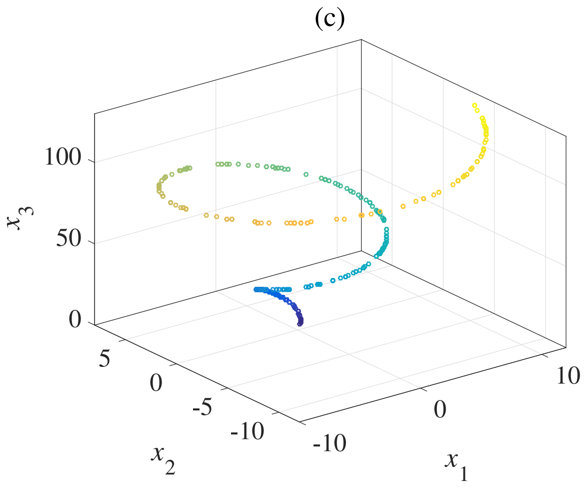


Figure 5.1: Illustration of the pre-image method for data lying on a conical spiral. 500 points were randomly sampled from the spiral, shown in Fig. (a). A 2-d approximation using diffusion maps is shown in Fig. (b). The reconstruction is illustrated in Fig. (c). Each point in Fig. (a) has a unique color, which is retained in Figs. (b) and (c).

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

```
1a: Manifold Learning for reduced-dimensional space approximation
        kPCA                                                          Diffusion maps
        Algorithm 1:                                                  Algorithm 2:
        $\{(z_1(\mathbf{y}^{(j)}),\dots,z_r(\mathbf{y}^{(j)}))^T\}_{j=1}^m$      $\{(\gamma'_1 r_{j1},\dots,\gamma'_r r_{jr})^T\}_{j=1}^m$
        $z_i(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})) \leftarrow z_i(\mathbf{y}^{(j)})$          $r_i(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)})) \leftarrow r_i(\mathbf{y}^{(j)}) \leftarrow r_{ji}$
2a: for $i \leftarrow 1$ to $r$ do
        kPCA                                                          Diffusion maps
        $\{\mathbb{z}_i(\boldsymbol{\xi}^{(j)}) \leftarrow z_i(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)}))\}_{j=1}^m$      $\{\mathbb{r}_i(\boldsymbol{\xi}^{(j)}) \leftarrow r_i(\boldsymbol{\eta}(\boldsymbol{\xi}^{(j)}))\}_{j=1}^m$
        $\{\eta(\boldsymbol{\xi}^{(j)}) \leftarrow \mathbb{z}_i(\boldsymbol{\xi}^{(j)})\}_{j=1}^m$      $\{\eta(\boldsymbol{\xi}^{(j)}) \leftarrow \mathbb{r}_i(\boldsymbol{\xi}^{(j)})\}_{j=1}^m$
        Scalar GPE: $\mathbb{z}_i(\boldsymbol{\xi}) \leftarrow \mathbb{E}[\eta(\boldsymbol{\xi})]$      Scalar GPE: $\mathbb{r}_i(\boldsymbol{\xi}) \leftarrow \mathbb{E}[\eta(\boldsymbol{\xi})]$
3a: end for
        kPCA                                                          Diffusion maps
        $\mathbf{z}_r(\boldsymbol{\eta}(\boldsymbol{\xi})) \leftarrow (\mathbb{z}_1(\boldsymbol{\xi}),\dots,\mathbb{z}_r(\boldsymbol{\xi}))^T$      $\boldsymbol{\psi}_r^t(\boldsymbol{\eta}(\boldsymbol{\xi})) \leftarrow (\gamma_1^t \mathbb{r}_i(\boldsymbol{\xi}),\dots,\gamma_r^t \mathbb{r}_r(\boldsymbol{\xi}))^T$
4a: Inverse map
        $\mathbf{y} \leftarrow \sum_{i=1}^{N_n} \left( \frac{\chi(d_{i,*})}{\sum_{i=1}^{N_n} \chi(d_{i,*})} \right) \mathbf{y}^{(i)}$
        kPCA                                                          Diffusion maps ($t = 1$)
        $\mathbb{k}(\mathbf{y}^{(i)},\mathbf{y}) \leftarrow \frac{1}{2}\left(1 - \boldsymbol{\tau}^T\mathbf{K}\boldsymbol{\tau} + 2\boldsymbol{\tau}^T\mathbf{k}_i\right)$      $\sum_{j=1}^{r} \gamma'_j \mathbb{r}_j(\boldsymbol{\xi}) l_{ij} \leftarrow \frac{\mathsf{k}(\mathbf{y}^{(i)},\mathbf{y})}{1+\sum_{j=1}^m \mathsf{k}(\mathbf{y}^{(j)},\mathbf{y})}$
        $d_{i,*} \leftarrow \sqrt{-s^2 \ln \mathbb{k}(\mathbf{y}^{(i)},\mathbf{y})}$      $d_{i,*} \leftarrow \sqrt{-s^2 \ln \mathsf{k}(\mathbf{y}^{(i)},\mathbf{y})}$
```

5.6 Results and discussion

In this section, three examples are considered. 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 [97] is a quasi-random sequence that is specifically designed to generate samples as uniformly as possible over the unit hypercube [122]. For each input $\boldsymbol{\xi}^{(i)}$, $i = 1, \dots, 500$, simulations were performed to yield data points $\mathbf{y}^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)}) \in \mathbb{R}^d$. Of the 500 data points, $m_t = 300$ were reserved for testing and the training points were selected from the remaining 200 ($m \leq 200$). The predicted value of $\mathbf{y}^{(i)}$ at a test input $\boldsymbol{\xi}^{(i)}$ using Algorithm 3 is denoted by $\mathbf{y}_p^{(i)} = \boldsymbol{\eta}(\boldsymbol{\xi}^{(i)})$. A relative error is defined as:

$$\text{Relative error} = \frac{||\mathbf{y}_p^{(i)} - \mathbf{y}^{(i)}||^2}{||\mathbf{y}^{(i)}||^2}, \tag{5.32}$$

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

5.6.1 Computational details

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

1. **kPCA.** A Gaussian kernel was used with the free parameter s^2 taken to be the average square distance between observations in the original space [140]: $s^2 = (1/m^2)\sum_{i,j=1}^{m} ||\mathbf{y}^{(i)} - \mathbf{y}^{(j)}||^2$. Polynomial and multi-quadratic kernels were also tested but found to be inferior. A sigmoid kernel was found to give similar results to those obtained with a Gaussian kernel. In the inverse mapping, all m points were employed for the reconstruction in physical space (inverse mapping).

2. **Diffusion maps.** A Gaussian kernel was used, in which the value of s^2 was determined as described above. Again, all m points were employed for the reconstruction. A value of $t = 1$ was used in the results presented below. Higher values of t did not lead to any significant changes.

3. **Gaussian Process Emulation**. The square exponential covariance function Eq. (5.16) 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.

5.6.2 Free convection in porous media

Subsurface flow in a porous medium can be modelled by Brinkman's equation (with a Boussinesq buoyancy term) and a thermal energy balance [171]:

$$\begin{gathered}-\left(\omega\kappa^{-1}\mathbf{v}+\nabla p\right)-\nabla\cdot\omega\epsilon^{-1}\left(\nabla\mathbf{v}+\nabla\mathbf{v}^{T}\right)=\rho\boldsymbol{g}c_e(T-T_c),\\ \rho C_p\mathbf{v}\cdot\nabla T-\nabla\cdot(\overline{\lambda}\nabla T)=0,\\ \nabla\cdot\mathbf{v}=0,\end{gathered}\tag{5.33}$$

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

A 2-d domain $(x_1, x_2) \in [0, 10] \times [0, 10]$ (in cm) is considered filled with water. The temperature boundary conditions are illustrated in Fig. 5.2. The temperature ranges from T_h to $T_c < T_h$ along the outer edges. The δ in Fig. 5.2 is a variable that goes from $\delta = 0$ at the surface $x_2 = 0$ (so that $T = T_h$ at this surface), to $\delta = 1$ at the cut-off (the dashed line at $x_2 = 0.1$cm on the right hand boundary), so at this cut-off $T = T_c$. In other words, temperature decreases linearly along the right hand boundary from $T = T_h$ at $x_2 = 0$, to $T = T_c$ at the cut-off point i.e $x_2 = 0.1$. Similarly for the left hand side boundary, the temperature goes from $T = T_h$ at $x_2 = 0$, to $T = T_c$ at $x_2 = 1$. Note that in this case the there is no cut-off, so delta is actually the same as x2 (both go from 0 to 1 along that segment of the boundary). To summarize, δ is generic variable that goes from 0 to 1 along a particular segment (the segment 0 to 0.1 on the right hand boundary and the segment 0 to 1 on the left hand boundary).

Buoyant flow is generated by the nonuniform temperature. No-slip conditions on all boundaries (with an arbitrary reference p) are assumed. The model was solved using the finite element method (FEM) with triangular elements and a quadratic Lagrange nodal basis. Details of the implementation and default parameter values can be found in [172].

Training and Testing. In this example, the input parameters were $\boldsymbol{\xi} =$

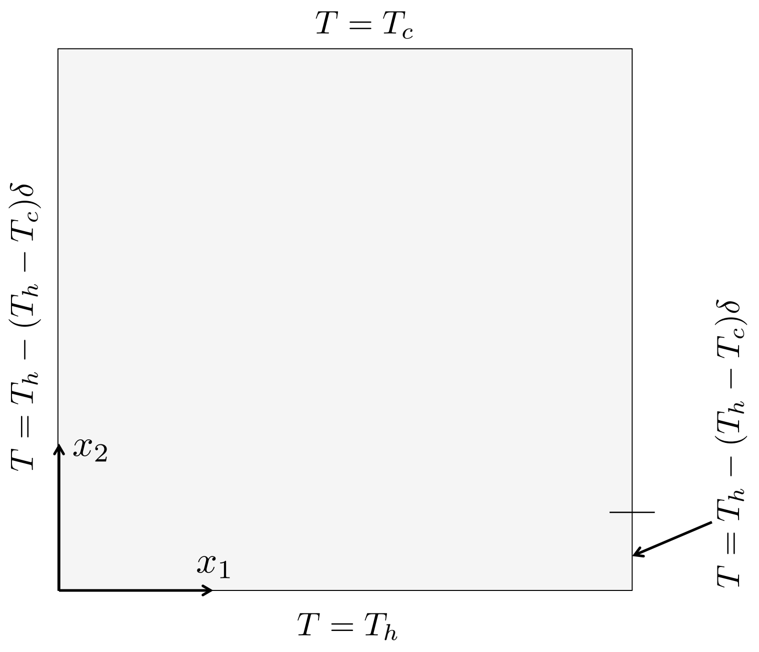


Figure 5.2: Temperature boundary conditions for the free-convection example. δ is a variable that represents the relative length of a boundary segment and goes from 0 to 1 along the segment as x_2 increases. The cut-off shown by the horizontal dash along $x_1 = 10$ cm is located at $x_2 = 1$ cm.

$(c_e[\mathrm{K}^{-1}], T_h[^{\mathrm{o}}\mathrm{C}])^T \in [10^{-11}, 10^{-8}] \times [40, 60]$. For each input $\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, 500$, the magnitude $|\mathbf{v}|$ of the velocity was recorded at each grid point on a regular 100×100 square spatial grid and the $d = 10^4$ values of $|\mathbf{v}|$ were vectorized to yield the data points $\mathbf{y}^{(i)} \in \mathbb{R}^d$, $i = 1, \ldots, 500$. In the notation of Section 5.1, $u(\mathbf{x}; \boldsymbol{\xi}) = |\mathbf{v}|$, $J = 1$, $l = 2$ and $d = 10^4$.

Results. Figure 5.3 shows Tukey box plots of the relative errors for the 300 test cases as the number of training points m and the approximate manifold dimension r are increased. For each box, the central line is the median, the lower and upper edges signify the first (Q_1) and third (Q_3) quartiles. The lower and upper lines (whiskers) define the errors within $1.5 \times (Q_3 - Q_1)$ of the first and third quartiles. All other points (considered outliers) are plotted individually using a '+' symbol. A decrease in the relative error for an increasing r is seen for both kPCA and diffusion maps. For both methods, the errors converge at around $r = 6$ dimensions. The median value of the error is marginally lower with kPCA, but it was found that the number of outliers was slightly higher using this method. For a high number of training points ($m \geq 80$), both methods provided accurate predictions and the differences in the errors were not significant. A comparison to Higdon's method [43] can be found in Figure 5.4, where the same problem is considered and equivalent boxplots are provided. The performance of both kPCA and diffusion maps is

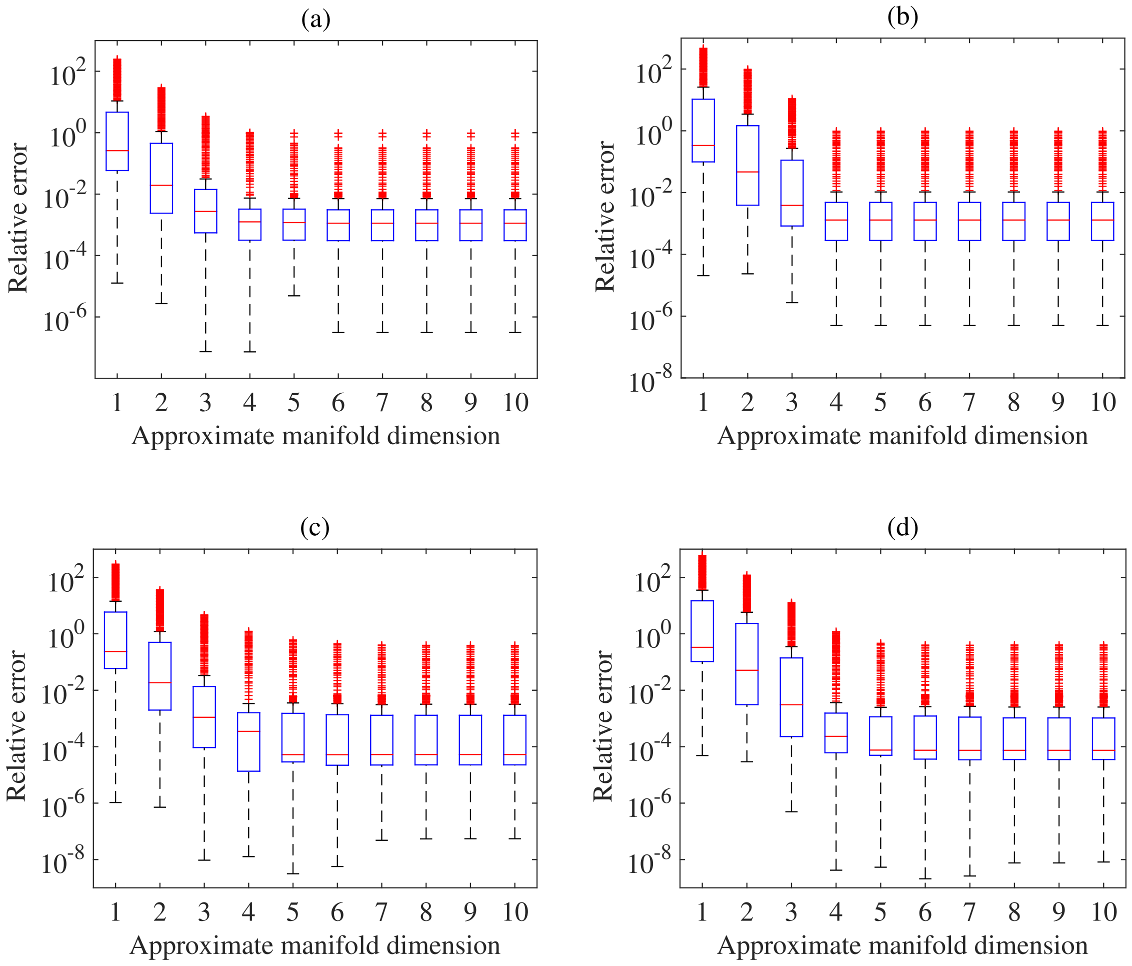


Figure 5.3: Tukey box plots of the relative error $||\mathbf{y}_p^{(i)} - \mathbf{y}^{(i)}||^2/||\mathbf{y}^{(i)}||^2$ in the free-convection example using Algorithm 3 with increasing approximate manifold dimension r on the 300 test points for: (a) kPCA with 40 training points; (b) diffusion maps with 40 training points; (c) kPCA with 120 training points; (d) diffusion maps with 120 training points.

far superior.

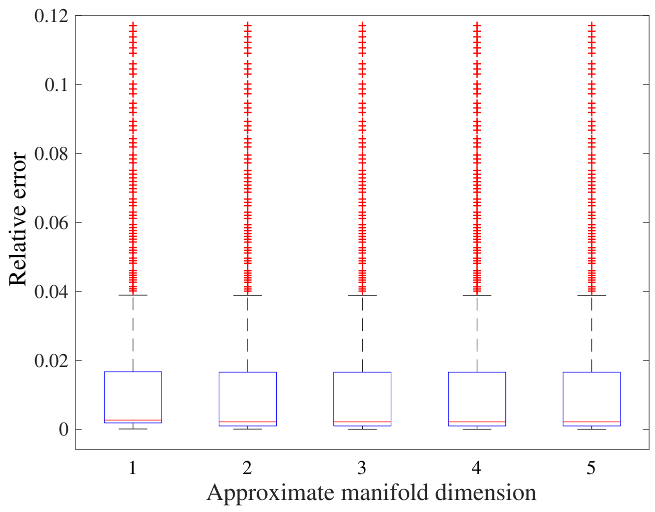


Figure 5.4: Boxplot of Higdon's method for free convection porous media using 40 training points.

Examples of the predictions are shown in Fig. 5.5 for 120 training points and $r = 5$. For both kPCA and diffusion maps, the error with respect to the first test example (Figs. 5.5(a)-(c)) lies around the median of the $r = 5$ boxplot in Fig. 5.3. The errors with respect to the second test example are close to the upper whiskers in the same boxplots. In both cases, Algorithm 3 with either kPCA or diffusion maps yields highly accurate predictions. An example of the outliers for both methods in the $r = 5$ boxplots in Figs. 5.3(c) and 5.3(d) is shown in Fig. 5.6. This figure demonstrates the worst level of prediction, which, nevertheless, captures the qualitative features of the velocity field and remains quantitatively accurate to a reasonable level.

Boxplots of the errors using an ANN and support vector machine regression (SVMR) for emulation of the coefficients, rather than GPE, are shown in Fig. 5.7 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 [107, 108] 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 hy-

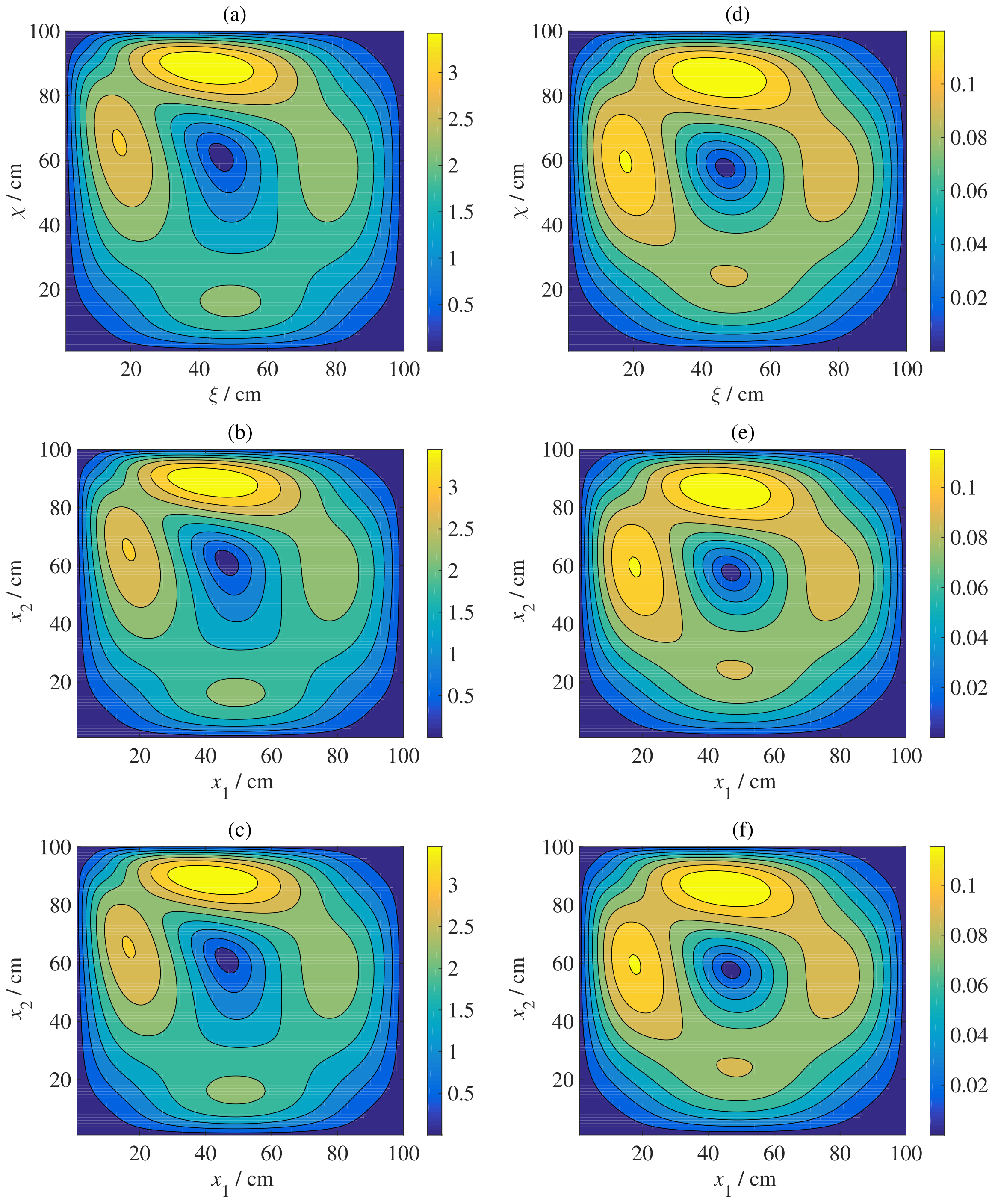


Figure 5.5: Predictions of the velocity field using 120 training points and $r = 5$ coefficients in the free-convection example. Figure (a) is the test point corresponding to $\boldsymbol{\xi} = (3.18 \times 10^{-9}[\mathrm{K}^{-1}], 56.7[^{\circ}\mathrm{C}])^T$, while Figs. (b) and (c) are the corresponding predictions using kPCA and diffusion maps, respectively. Figure (d) is the test point corresponding to $\boldsymbol{\xi} = (7 \times 10^{-11}[\mathrm{K}^{-1}], 46.7[^{\circ}\mathrm{C}])^T$, while Figs. (e) and (f) are the corresponding predictions using kPCA and diffusion maps, respectively.

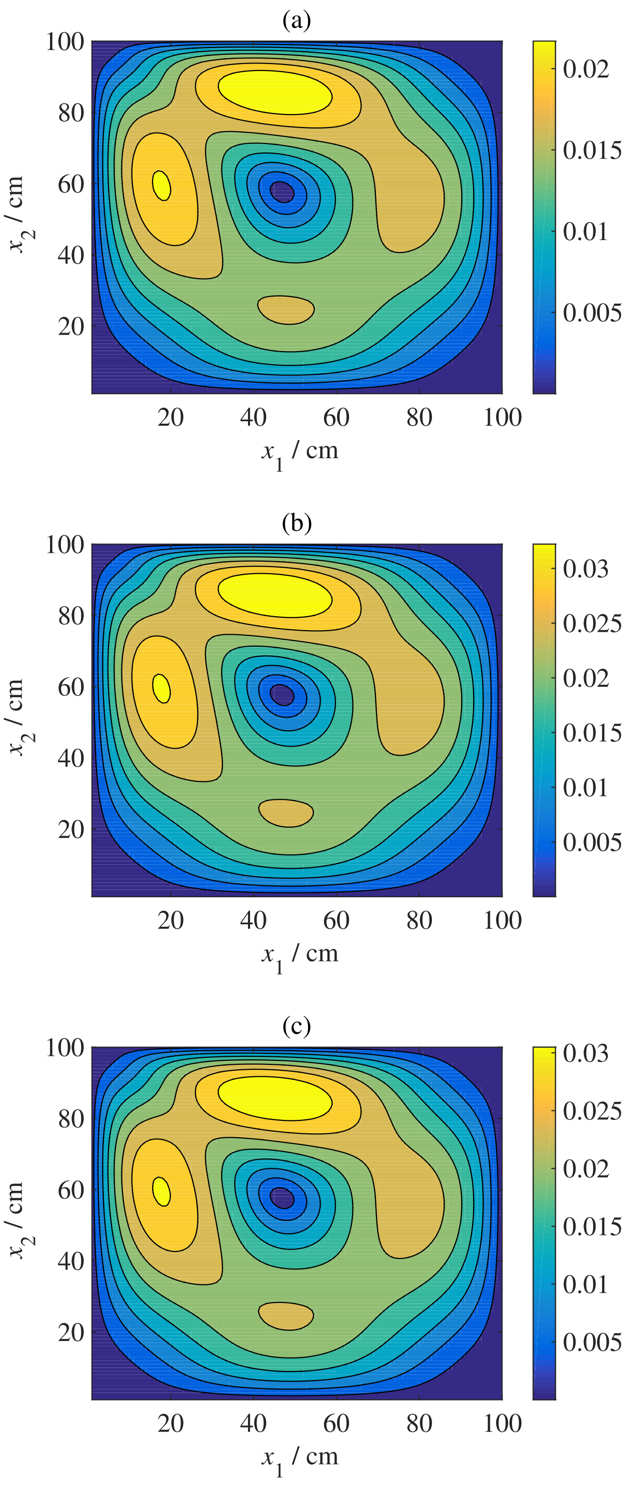


Figure 5.6: Predictions of the velocity field using 120 training points and $r = 5$ coefficients in the free-convection example in the case of an outlier. Figure (a) is the test point corresponding to $\boldsymbol{\xi} = (1\times10^{-9}[\mathrm{K}^{-1}], 40.7[^{\mathrm{o}}\mathrm{C}])^T$, while Figs. (b) and (c) are the predictions using kPCA and diffusion maps, respectively.

perparameters (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 [161]. A single hidden layer was employed and the number of neurons was selected using a sequential network construction [108]. For the SVMR, Gaussian and polynomial kernels were tested (with varying order), together with an L^1 loss function.

Comparing with Figs. 5.3(a) and (b), it may be seen 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. 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. 5.7 indicates that, at least in this example, GPE and ANN are superior.

5.6.3 Lid driven cavity

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

$$(\mathbf{v} \cdot \nabla)\mathbf{v} - Re^{-1}\nabla^2\mathbf{v} + \nabla p = 0, \quad \nabla \cdot \mathbf{v} = 0, \tag{5.34}$$

where $\mathbf{v} = (v_1, v_2)^T$ is the liquid velocity, p is the liquid pressure and Re is the Reynolds number. The boundary conditions are $\mathbf{v} = (v_1^0, 0)$ for $x_2 = 1$, where v_1^0 is the lid velocity, and $\mathbf{v} = 0$ on the other three boundaries. The model was solved using finite difference method on a staggered grid with implicit diffusion and a Chorin projection for the pressure [173].

Training and Testing. The Reynold's number and lid velocity were used as input parameters: $\boldsymbol{\xi} = (Re, v_1^0)^T \in [700, 1200] \times [0.01, 10]$. All other parameters were kept at the default values. For each input $\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, 500$, the pressure p and the component velocities v_1 and v_2 were recorded at each grid point on a regular 100×100 spatial grid. The $d/3 = 10^4$ values of each field variable were vectorized to yield vector outputs $\mathbf{y}_{v_1}^{(i)} \in \mathbb{R}^{d/3}$, $\mathbf{y}_{v_2}^{(i)} \in \mathbb{R}^{d/3}$

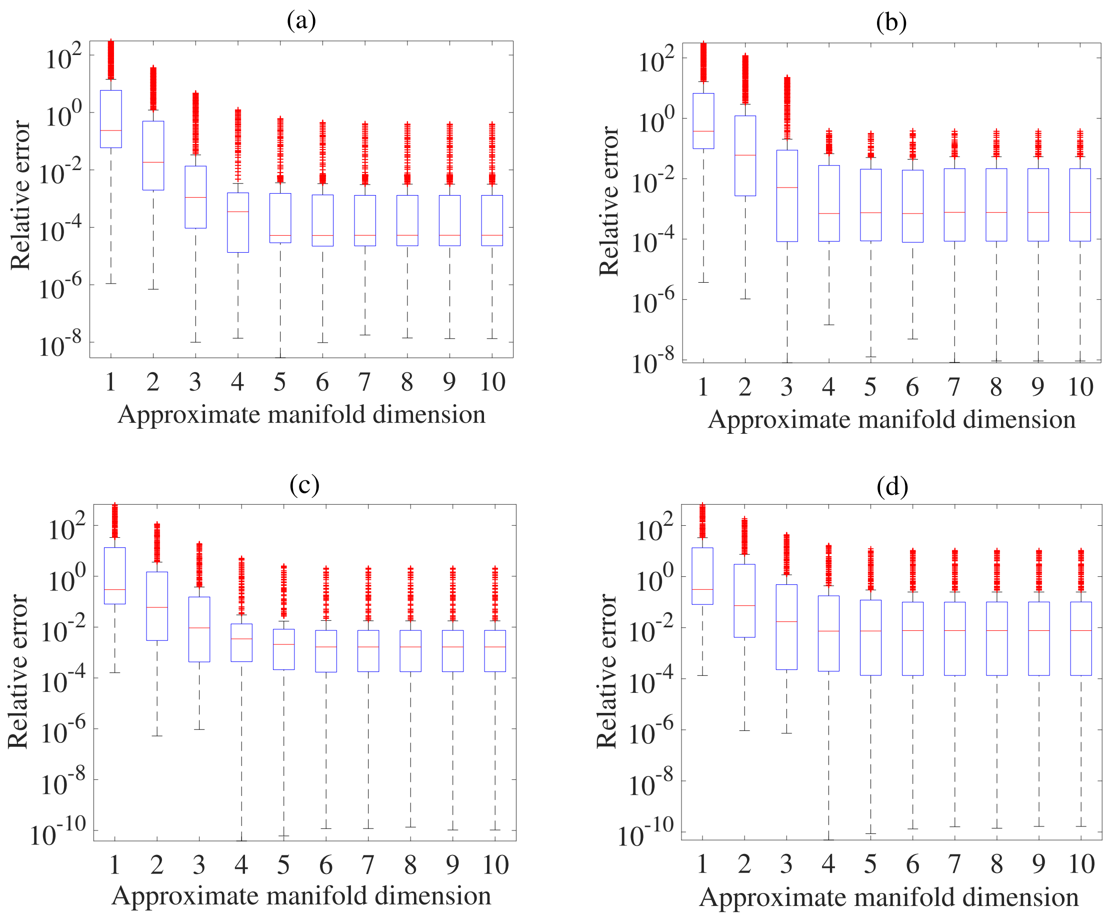


Figure 5.7: Tukey box plots of the relative error $||\mathbf{y}_p^{(i)} - \mathbf{y}^{(i)}||^2/||\mathbf{y}^{(i)}||^2$ in the free-convection example using Algorithm 3 with an ANN and SVMR for an increasing approximate manifold dimension r on the 300 test points. In both cases, 120 training points were used. (a) kPCA with ANN; (b) diffusion maps with ANN; (c) kPCA with SVMR; (d) diffusion maps with SVMR.

and $\mathbf{y}_p^{(i)} \in \mathbb{R}^{d/3}$. The three vectors were then combined into a single vector $\mathbf{y}^{(i)} = [\mathbf{y}_{v_1}^{(i)}\ \ \mathbf{y}_{v_2}^{(i)}\ \ \mathbf{y}_p^{(i)}] \in \mathbb{R}^d$ to account for the correlations between the fields. In the notation of Section 5.1, $J = 3$, $l = 2$ and $d = 3 \times 10^4$. This is a multiple field example discussed in Section 5.1, 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. 5.8 for an increasing r (approximate manifold dimension) and m. Around $r = 5$ is sufficient for both values of m using both methods. In this case, the differences between the methods was almost negligible, except that again there were fewer outliers for diffusion maps, particularly for low numbers of training points. Figure 5.9 shows the equivalent boxplots using Higdon's method [43]. 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. 5.10 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 black. 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. 5.10(a)-(c)) lies close to the median in the $r = 5$ boxplot in Fig. 5.8. The second test example corresponds to an outlier for both methods (relative error around 0.07). The results of Algorithm 3 remain accurate, especially for diffusion maps. The error in kPCA is primarily due to the prediction of the pressure field, in particular the maximum value in the top right corner. Nevertheless, the profile is well captured.

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

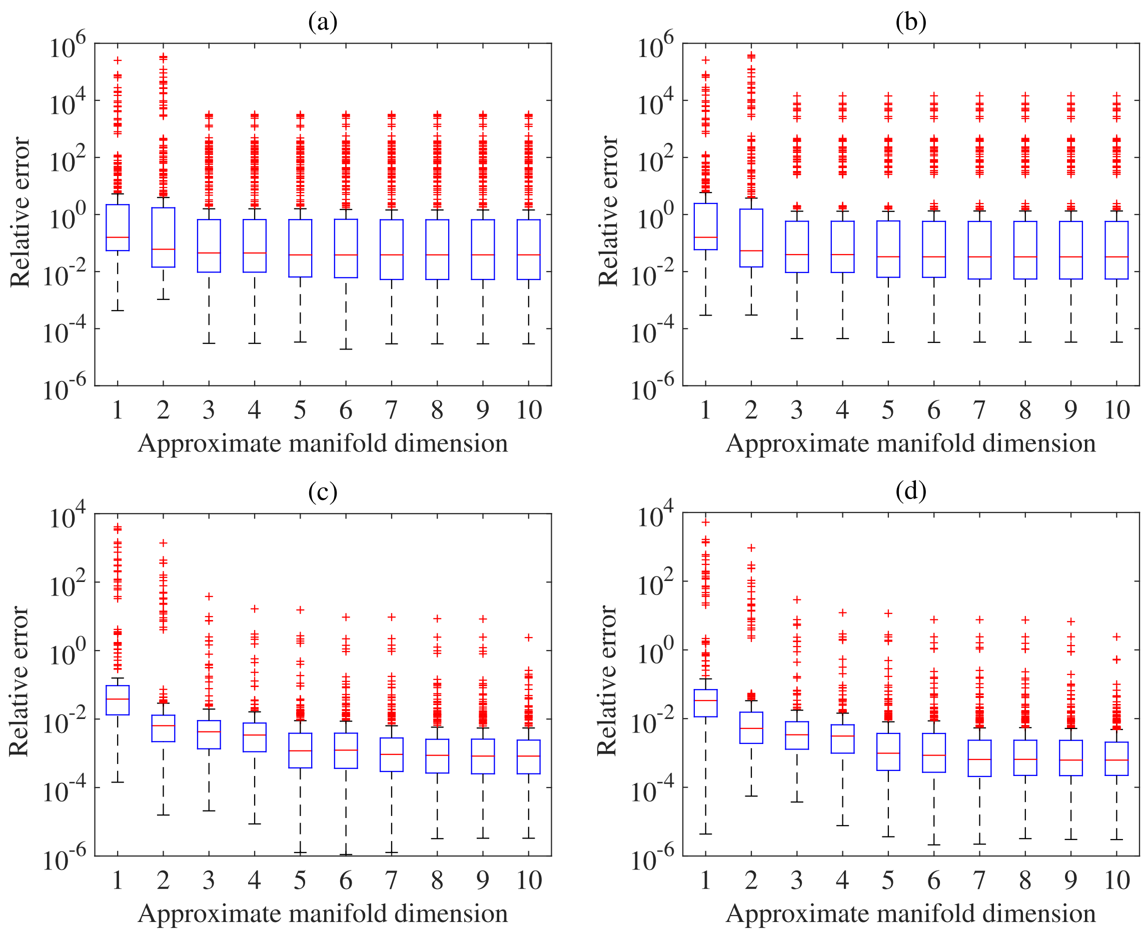


Figure 5.8: Tukey box plots of the relative error $||\mathbf{y}_p^{(i)} - \mathbf{y}^{(i)}||^2/||\mathbf{y}^{(i)}||^2$ in the lid-driven cavity example using Algorithm 3 with an increasing approximate manifold dimension r on the 300 test points for: (a) kPCA with 80 training points; (b) diffusion maps with 80 training points; (c) kPCA with 120 training points; (d) diffusion maps with 120 training points.

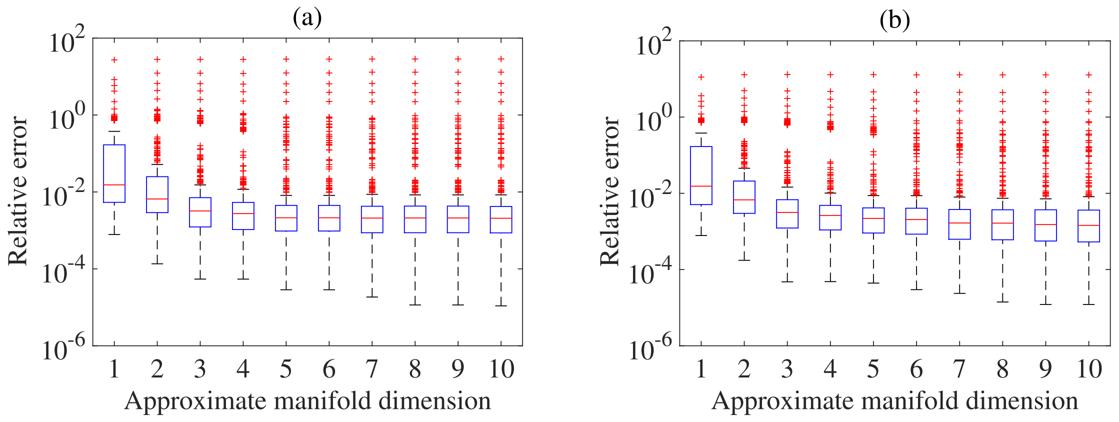


Figure 5.9: Tukey box plots of the relative error $||\mathbf{y}_p^{(i)} - \mathbf{y}^{(i)}||^2/||\mathbf{y}^{(i)}||^2$ in the lid-driven cavity example using Higdon's method [43] with an increasing approximate manifold dimension r on the 300 test points for: (a) 80 training points; (b) 120 training points.

conditions:

$$\begin{aligned}
v_1(x_1,1) &= 5c_1\sin(c_2\pi x_1)e^{-c_3x_1}, \quad v_2(x_1,1)=0,\\
v_1(x_1,0) &= 5c_4\sin(c_5\pi x_1)e^{-c_6x_1}, \quad v_2(x_1,0)=0,\\
v_2(1,x_2) &= 5c_7\sin(c_8\pi x_2)e^{-c_9x_2}, \quad v_1(1,x_2)=0,\\
v_2(0,x_2) &= 5c_{10}\sin(c_{11}\pi x_2)e^{-c_{12}x_2}, \quad v_1(0,x_2)=0,
\end{aligned} \tag{5.35}$$

for constants $c_1,\ldots,c_{12}$. The inputs were defined as $\boldsymbol{\xi} = (Re, c_1,\ldots,c_{12})^T \in [500,1000]\times(0,1)\times(0,1)\times\cdots\times(0,1)$. Inputs $\boldsymbol{\xi}^{(i)}$, $i=1,\ldots,1000$ were generated using a Sobol sequence and simulations were performed to yield 1000 data points. Of the 1000 data points, $m_t = 300$ were reserved for testing and the training points were selected from the remaining 700. Both kPCA and diffusion maps exhibited excellent performance, as illustrated in the boxplots in Fig. 5.11, 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. 5.12 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

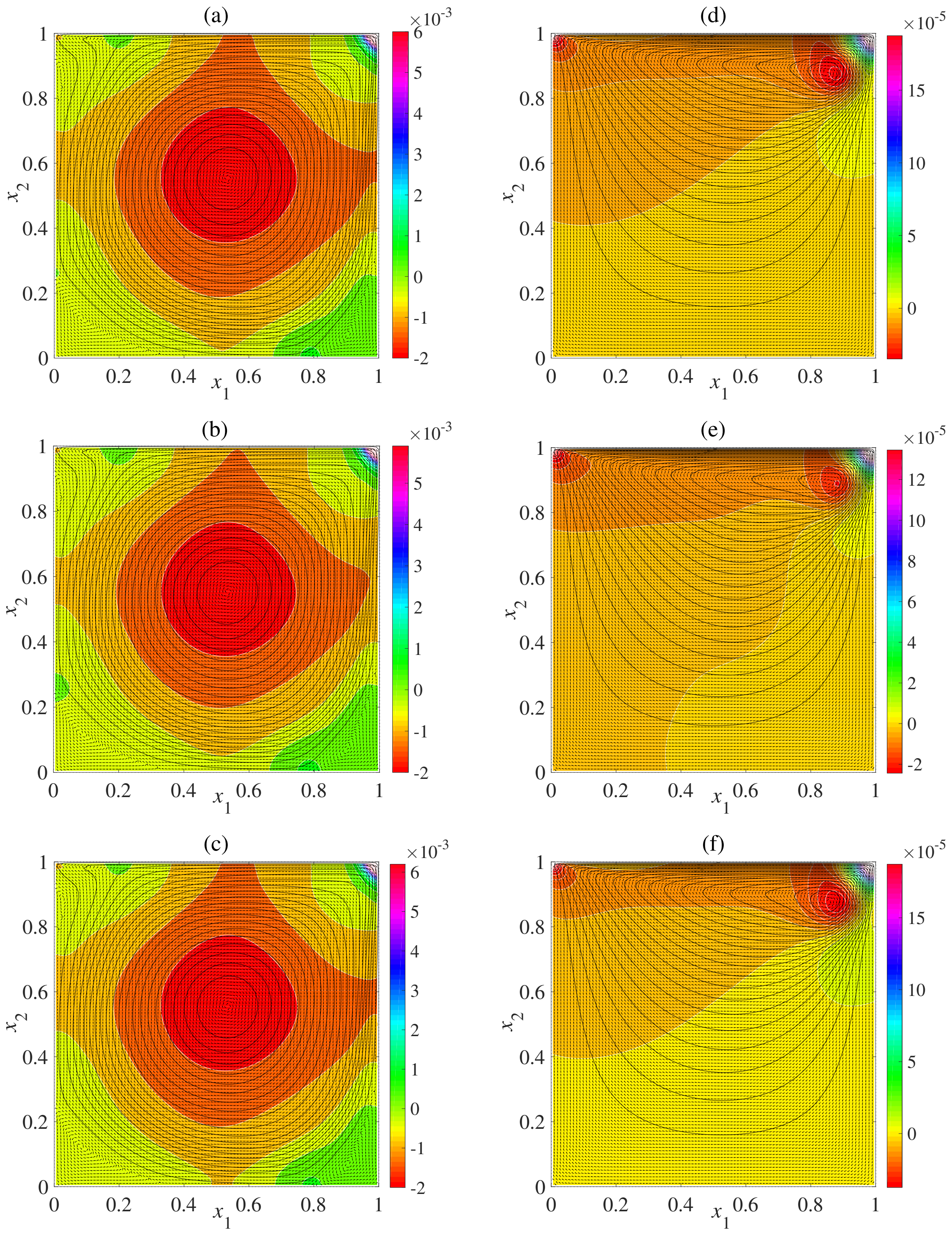


Figure 5.10: Predictions of the velocity field using 120 training points and $r = 5$ coefficients in the lid driven cavity example. Figure (a) is the test point corresponding to $\boldsymbol{\xi} = (874.8, 7.79)^T$, while Figs. (b) and (c) are the corresponding predictions using kPCA and diffusion maps, respectively. Figure (d) is the test point corresponding to $\boldsymbol{\xi} = (773.24, 0.77)^T$, while Figs. (e) and (f) are the corresponding predictions using kPCA and diffusion maps, respectively.

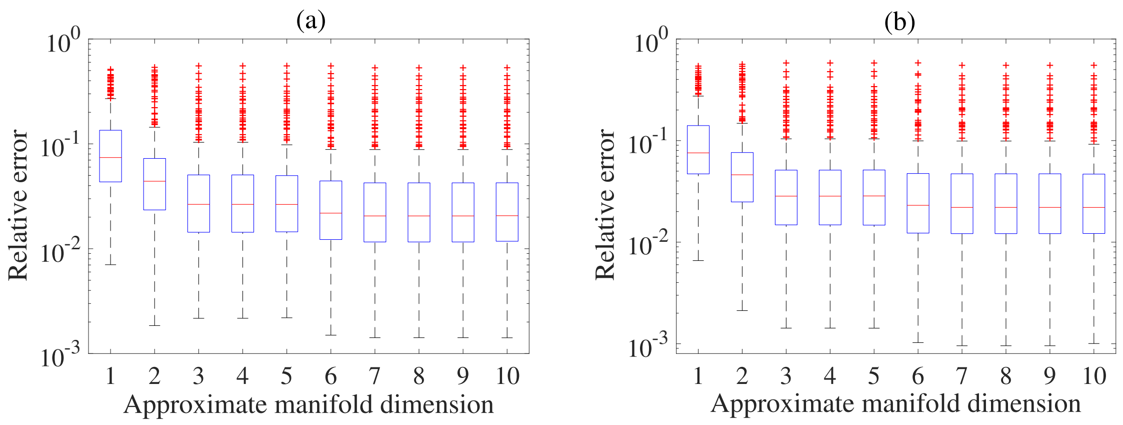


Figure 5.11: Tukey box plots of the relative error $||\mathbf{y}_p^{(i)} - \mathbf{y}^{(i)}||^2/||\mathbf{y}^{(i)}||^2$ in the lid-driven cavity example with boundary conditions as in Eq. (5.35). The trends are shown for an increasing approximate manifold dimension r using 600 training points and 300 test points for: (a) kPCA and (b) diffusion maps.

training points led to poor performance from all methods.

5.6.4 Hydrogen fuel cell model

In this example, a hydrogen/oxygen polymer electrolyte membrane (PEM) fuel cell model that incorporates species conservation was considered, charge conservation and a momentum balance in the porous layers. The 2-d domain includes the porous gas diffusion layers (GDLs), through which the species (oxygen, water and hydrogen) are transported from the channels to the reaction sites in the catalyst layers, which are adjacent to the PEM (Fig. 5.13).

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 [174]. The catalyst layer morphology is approximated as clusters (agglomerates) of carbon-supported platinum coated with the electrolyte. The transfer

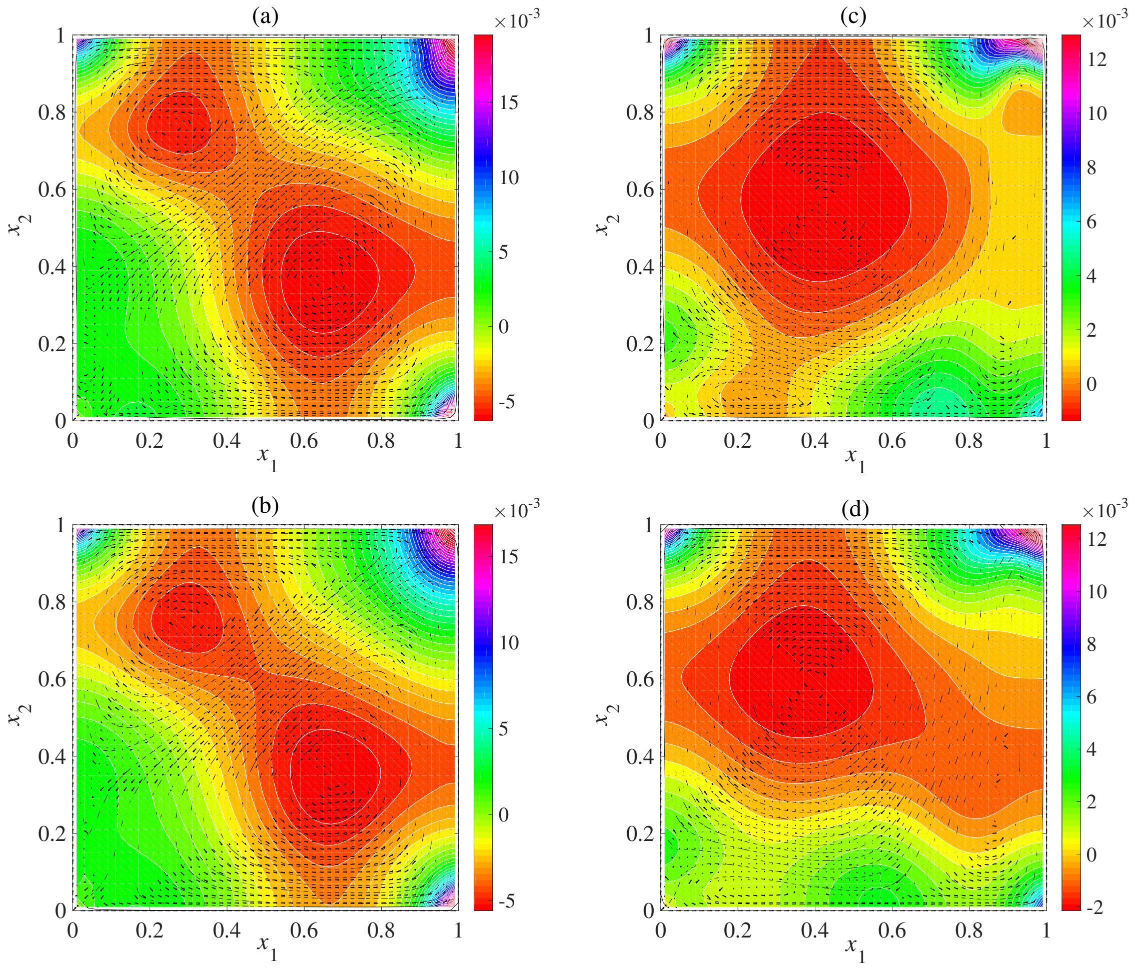


Figure 5.12: Predictions of the velocity and pressure fields using $m = 500$ training points and $r = 10$ coefficients in the lid driven cavity example with the boundary conditions of Eq. (5.35). Figure (a) is a test point and Fig. (b) is the corresponding prediction using kPCA, with a relative error of 0.0244. Figure (c) is a second test point and Fig. (d) is the corresponding prediction using kPCA, with a relative error of 0.2275.

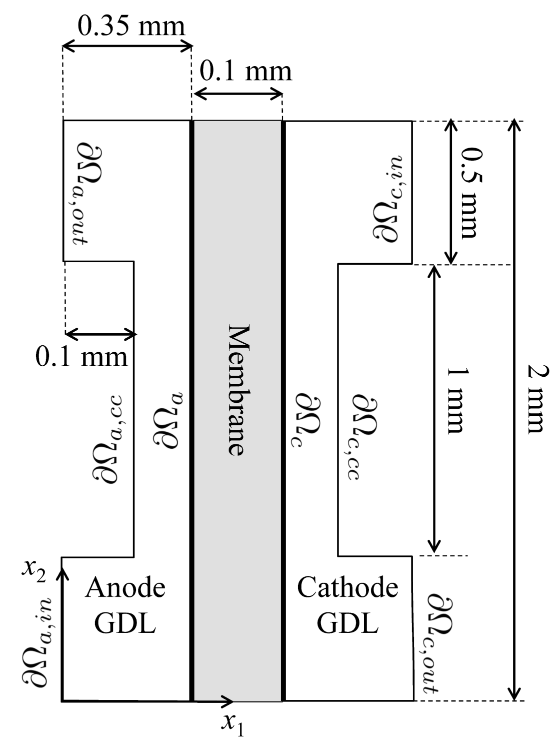


Figure 5.13: A schematic of the PEM fuel cell and the components that form the model domain.

current densities are expressed as follows [175]:

$$j_c = -\frac{12L_{act}FD_{agg}}{R_{agg}^2}C_{\mathrm{O}_2,agg}(1-\epsilon_{mac})(1-\lambda_c\coth\lambda_c),$$

$$j_a = -\frac{6L_{act}FD_{agg}}{R_{agg}^2}C_{\mathrm{H}_2,agg}\left(1-e^{-\frac{2F}{RT}\eta_a}\right)(1-\epsilon_{mac})(1-\lambda_a\coth\lambda_a), \tag{5.36}$$

$$\lambda_c = \sqrt{\frac{i_{0c}SR_{agg}^2}{4FC_{\mathrm{O}_2,ref}D_{agg}}e^{\frac{F}{2RT}\eta_c}} \quad \lambda_a = \sqrt{\frac{i_{0a}SR_{agg}^2}{2FC_{\mathrm{H}_2,ref}D_{agg}}},$$

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

is the universal gas constant. The reactants dissolve in the electrolyte at the agglomerate surfaces at a rate governed by Henry's law, so that:

$$C_{\mathrm{H}_2,agg} = pX_{\mathrm{H}_2}/K_{\mathrm{H}_2} \quad C_{\mathrm{O}_2,agg} = pX_{\mathrm{O}_2}/K_{\mathrm{O}_2}, \tag{5.37}$$

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

The charge balances are given by:

$$-\nabla \cdot (\sigma_e \nabla \phi_e) = 0 \quad \text{and} \quad -\nabla \cdot (\sigma_s \nabla \phi_s) = 0, \tag{5.38}$$

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

$$\eta_a = \phi_s - \phi_e - E_{eq,a} \quad \text{and} \quad \eta_c = \phi_s - \phi_e - E_{eq,c}, \tag{5.39}$$

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

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

$$\nabla \cdot (\rho \mathbf{v}) = 0, \quad \mathbf{v} = -k_p \omega^{-1} \nabla p, \tag{5.40}$$

where ω is the gas viscosity and k_p is the GDL permeability. The ideal gas law is used to determine the density: $\rho = (p/RT)\sum_i M_i X_i$, in which M_i is the molecular weight of species $i \in \{\mathrm{H}_2, \mathrm{O}_2, \mathrm{H}_2\mathrm{O}, \mathrm{N}_2\}$. The transport of species through the GDLs is governed by convection and multicomponent diffusion (Stefan-Maxwell) [176]. In the cathode, the species are $\mathcal{I}_1 = \{\mathrm{O}_2, \mathrm{H}_2\mathrm{O}, \mathrm{N}_2\}$ and in the anode the species are $\mathcal{I}_2 = \{\mathrm{H}_2, \mathrm{H}_2\mathrm{O}, \mathrm{N}_2\}$. The transport equations in the cathode are given by:

$$\begin{gathered} -\nabla \cdot \left\{ \rho Y_i \sum_{\substack{j \in \mathcal{I}_1 \\ j \neq i}} D_{i,j} \left(\nabla X_j + (X_j - Y_j) \nabla p / p \right) \right\} = -\rho \mathbf{v} \cdot \nabla Y_i, \\ Y_{\mathrm{N}_2} = 1 - Y_{\mathrm{O}_2} - Y_{\mathrm{H}_2\mathrm{O}}, \end{gathered} \tag{5.41}$$

for $i \in \{\mathrm{O}_2, \mathrm{H}_2\mathrm{O}\}$. Y_i is the mass fraction of species i and the $D_{i,j}$ are binary diffusivities [176]. Identical equations for species $\mathcal{I}_2$ are solved in the anode.

The boundary conditions for the potential impose a cell voltage $\mathbf{V}_{cell}$:

$$\begin{aligned} \phi_s &= 0 \quad \mathbf{x} \in \partial\Omega_{a,cc}, \\ \phi_s &= \mathbf{V}_{cell} \quad \mathbf{x} \in \partial\Omega_{c,cc}, \\ -\mathbf{n}\cdot\nabla\phi_s &= 0 \quad \text{otherwise}, \end{aligned} \tag{5.42}$$

where $\mathbf{n}$ 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 [174]:

$$\begin{aligned} -\mathbf{n}\cdot\mathbf{v} &= j_a \left(M_{\mathrm{H}_2}/2 + \lambda_{\mathrm{H_2O}} M_{\mathrm{H_2O}}\right)/(\rho F) \quad \mathbf{x} \in \partial\Omega_a, \\ -\mathbf{n}\cdot\mathbf{v} &= j_c \left(M_{\mathrm{O}_2}/2 + \left[1/2 + \lambda_{\mathrm{H_2O}}\right] M_{\mathrm{H_2O}}\right)/(\rho F) \quad \mathbf{x} \in \partial\Omega_c, \end{aligned} \tag{5.43}$$

where $\lambda_{\mathrm{H_2O}}$ is the water drag number [174]. At the other boundaries except the inlets and outlets $-\mathbf{n}\cdot(\rho\mathbf{v}) = 0$ is imposed. At the catalyst layer surfaces the mass fluxes of reactants are determined by Faraday's law:

$$\begin{aligned} -\mathbf{n}\cdot\mathbf{n}_{\mathrm{H}_2} &= M_{\mathrm{H}_2} j_a/(2F) \quad \mathbf{x} \in \partial\Omega_a, \\ -\mathbf{n}\cdot\mathbf{n}_{\mathrm{O}_2} &= M_{\mathrm{O}_2} j_c/(4F) \quad \mathbf{x} \in \partial\Omega_c, \\ -\mathbf{n}\cdot\mathbf{n}_{\mathrm{H_2O}} &= M_{\mathrm{H_2O}} j_c \left(1/2 + \lambda_{\mathrm{H_2O}}\right)/F \quad \mathbf{x} \in \partial\Omega_c, \end{aligned} \tag{5.44}$$

where $\mathbf{n}_i = -\rho Y_i \sum_{j\neq i} D_{i,j}(\nabla X_j + (X_j - Y_j)\nabla p/p) + \rho\mathbf{v}Y_i$ is the flux of species i. At all other boundaries except the inlets and outlets, $\mathbf{n}_i = 0$. 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 [177].

Training and Testing. The cell voltage $\mathbf{V}_{cell}$ and the membrane/electrolyte conductivity σ_e were used as input parameters: $\boldsymbol{\xi} = (\mathbf{V}_{cell}[\mathrm{V}], \sigma_e[\mathrm{S\ m^{-1}}])^T \in [0.2, 0.8] \times [1, 15]$. 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 $d = 4.5 \times 10^4$ values of $X_{\mathrm{H_2O}}$ were re-ordered into vector form to yield vectors $\mathbf{y}^{(i)} \in \mathbb{R}^d$. In the notation of Section 5.1, $u(\mathbf{x}; \boldsymbol{\xi}) = X_{\mathrm{H_2O}}$, $J = 1$, $l = 2$ and $d = 4.5 \times 10^4$.

Results. Figure 5.14 shows the Tukey box plots of the relative error for increasing r (approximate manifold dimension) and m. The results using both methods are highly accurate, particularly for $m = 120$ (in fact, $m = 80$ was found to give a similar level of performance). The performance with diffusion maps is better for $m = 60$, while the performance with kPCA is slightly superior with $m = 120$. Again there are more outliers in the box plots for kPCA. Examples of the predictions are shown in Fig. 5.15 for 120 training points and $r = 7$. In the first example (Figs. 5.15(a)-(c)), the error with respect to the test case lies close to the median in the $r = 7$ boxplot for kPCA (Fig. 5.14(c)), while for diffusion maps the error is near the upper whisker in the corresponding boxplot ($m = 120$, $r = 7$ in Fig. 5.14(d)). The second example (Figs. 5.15(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.

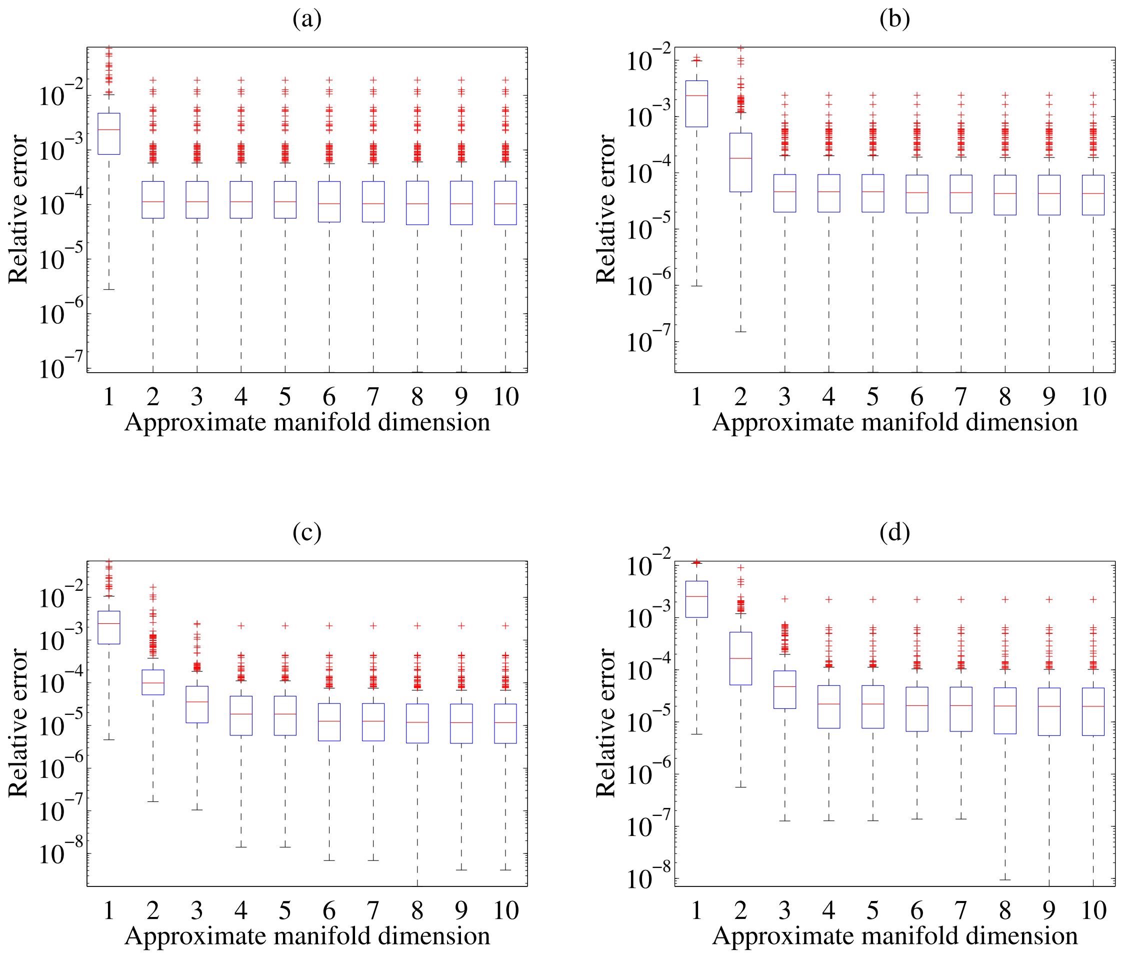


Figure 5.14: Tukey box plots of the relative error $||\mathbf{y}_p^{(i)} - \mathbf{y}^{(i)}||^2/||\mathbf{y}^{(i)}||^2$ in the PEM fuel cell example using Algorithm 3 with increasing approximate manifold dimension r on the 300 test points for: (a) kPCA with 40 training points; (b) diffusion maps with 40 training points; (c) kPCA with 120 training points; (d) diffusion maps with 120 training points.

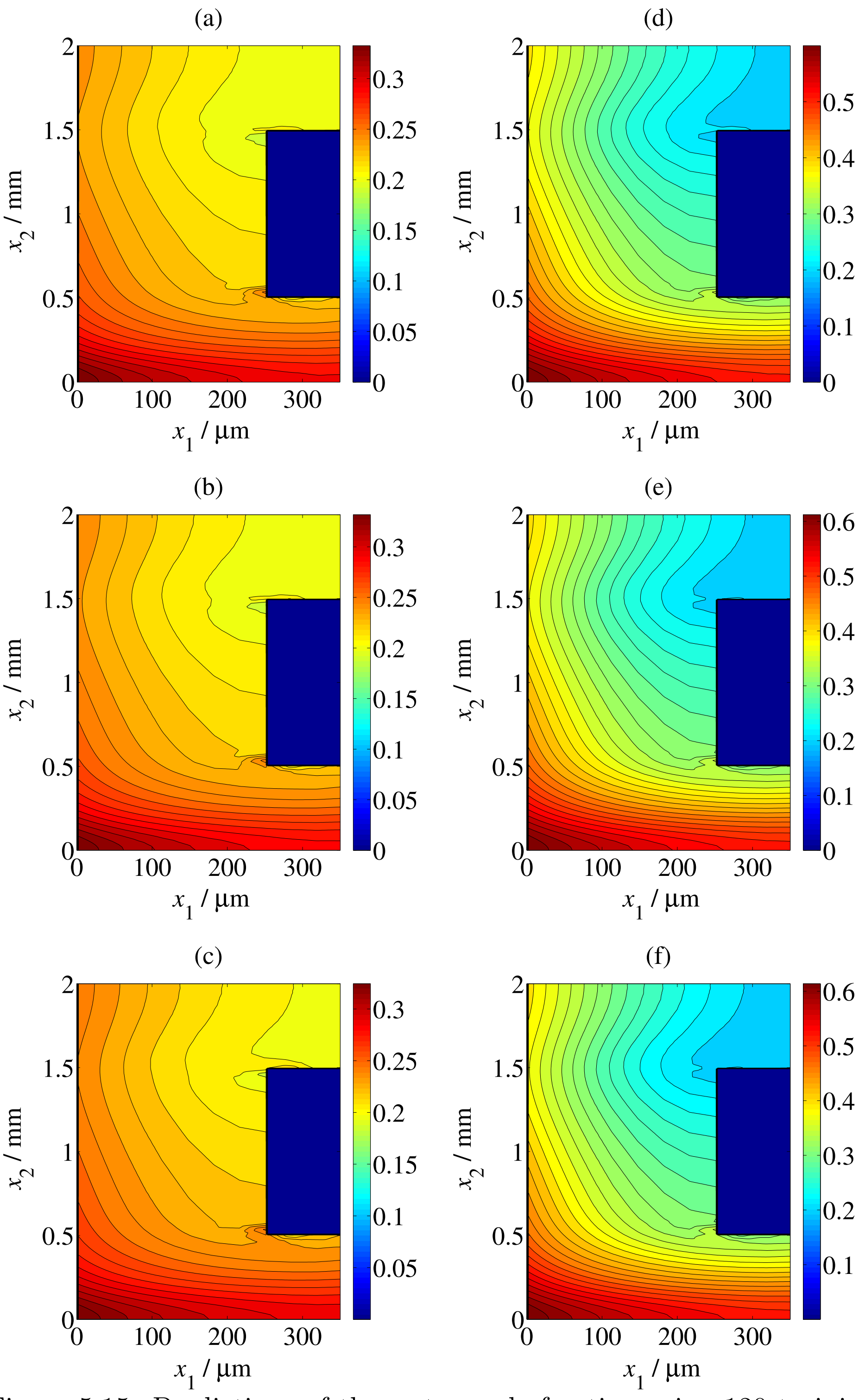


Figure 5.15: Predictions of the water mole fraction using 120 training points and $r = 7$ coefficients in the PEM fuel cell example. Figure (a) is the test point corresponding to $\boldsymbol{\xi} = (0.525[\mathrm{V}], 1.492[\mathrm{S\ m^{-1}}])^T$, while Figs. (b) and (c) are the corresponding predictions using kPCA and diffusion maps, respectively. Figure (d) is the test point corresponding to $\boldsymbol{\xi} = (0.301[\mathrm{V}], 9.039[\mathrm{S\ m^{-1}}])^T$ obtained using direct simulation, while Figs. (e) and (f) are the corresponding predictions using kPCA and diffusion maps, respectively.

5.7 Concluding Remarks

An emulator can be used to find an approximation of complicated computer models. There are cases though that an emulator will fail to provide accurate results, especially when dealing with high-dimensional or extremely non-linear data. Here, a Gaussian Process emulator has been developed for high dimensional output space. To achieve this non linear dimensionality reduction techniques (kPCA, Diffusion map) were used. Both methods have their challenges of finding a valid basis approximate the solution.

Also, GP replaced by SVM for regression and ANNs to compare the results of different emulators for both non-linear dimensionality reduction techniques. The results showed that for the examples considered here ANN and GPs are superior as they were able to capture the behaviour of the systems and give more accurate results.

In both techniques the inverse map from the low dimensional to the physical space were implemented. For kPCA there are several techniques for the approximation of the inverse map which are accurate and stable. The challenges arise for the pre-image problem of diffusion maps. The methods proposed until now in the literature are based on Delaunay triangulation which are highly unstable and computationally expensive. An other approach to the pre-image problem is to take the weighted average of N_n neighbouring points. In this chapter, a new distance measure is presented which is accurate and stable for high dimensional output spaces as it can be seen from the examples.

Moreover, there are several non-linear dimensionality reduction techniques that tested and potentially could replace those carried out in this chapter such as Laplacian Eigenmaps but their results are not presented as they are preliminary and further tests have to be done to assure they are accurate.

Chapter 6

Conclusion and further work

The target of this thesis was the construction of efficient emulators for parameter dependent partial differential equations, where the quantities of interest are spatio-temporal fields. Emulators can help to avoid repeated calls of the simulator for computationally expensive applications such as sensitivity and uncertainty analysis, optimization and inverse parameter estimation.

Although, in some cases the emulator can fail to give accurate and reliable results when the problem under consideration is of high dimensionality or its response surface is highly non linear. To overcome this issue, in this thesis nonlinear statistical emulation strategies has been developed.

Neural networks have seen a lot of attention the last years and have been used in many scientific fields. In this thesis swallow and deep neural networks were used and applied on electromagnetic fields giving accurate and stable results. The activation function used for its neuron was the softmax although its main limitation is that in some cases can lead to vanishing the gradient. For the problems of this thesis that was not the case, although newly developed activation function such as rectified linear units and scaled exponential linear unit can be used to test their accuracy, stability and the improvements could be offer.

The limitation of the rectified linear unit activation function is that could lead to a known issue, the ”explosion” of the gradient Scaled exponential linear unit from the other hand seems promising but more testing has to be made to validate its accuracy. Also, more research can be done in the deep architecture of the network, meaning more experiments on the number of hidden layers and the trade off between the accuracy and the computational time of the problems under consideration.

Non-linear dimensionality techniques (kernel PCA and diffusion map) were introduced in conjunction with emulation strategies such as NN, GPs and SVM for regression. For the diffusion map part, a new solution to the pre-image problem was presented based on a new metric measure to map from the feature space back to the physical space.

An extension of this work could be the development of a general framework for the pre-image solution applicable to most non-linear dimensionality reduction techniques. From the results presented throughout the thesis it has been proven that the method is more accurate and stable for nonlinear response surfaces than Higdon's method [43]. Although, Higdon's method can be quite useful in linear tasks due to the insight it provides to the predictive variances without the need of using Monte Carlo techniques.

In addition to the techniques presented in this thesis, there are manifold learning techniques that have the potential to solve non-linear tasks such as Laplacian eigenmaps and local linear embeddings. The usual issue of those techniques is the the mapping from the feature space to the physical space, which it could be solved by the pre-image solution presented in this thesis. A promising dimensionality reduction technique worth testing with the general emulation framework developed in this thesis is the t-distributed stochastic neighbour embedding (t-SNE), as it be used for visualization of high dimensional data. The objective function of t-SNE is minimized using a gradient descent optimization that is initiated randomly.

A promising approach is the unsupervised Gaussian Process Latent Variable Models (GPLVM) that could be used for nonlinear dimensionality reduction technique as soon as a solution can be found for the scalability issues when the number of parameters is very high.

Another alternative to deal with high dimensional datasets is to perform sensitivity analysis to do the screening and ranking of inactive variables. Two different methods tested here, namely the elementary effect test and the variance base sensitivity analysis, where both gave accurate results. Using a stochastic emulator such as GP it is possible to perform SA under uncertainty of high-dimensional multi-output problems. More SA techniques (regression analysis, variogram analysis of response surfaces and Fourier amplitude sensitivity test) can be tested with the GPE framework developed here to check their performance in terms of accuracy, stability and computational power that they need.

Finally, a reduced order model was developed in this thesis based on an extension of proper orthogonal decomposition to linear and nonlinear response surfaces by expanding the Discrete Empirical Interpolation Method to dynamic, parameter dependent problems. The method developed here, overcomes the main challenges of the accuracy and approximation of new parameter values of the POD bases, and in addition can handle nonlinear response surfaces. Compared to the global basis POD the extension proposed found from the results to be more accurate and needing almost the same computational time. An extension of this work could be the use of different nonlinear techniques or machine learning algorithms and their application on broader class of disciplines and problems as it would be useful to examine more their accuracy and stability.

Appendices

Appendix A: Numerical Methods

A1.1 Numerical Methods

In appendix A the three different numerical methods used in this thesis are presented. The numerical methods used for discretization are finite difference, finite volume and finite elements.

A1.1.1 Finite Difference Method

Finite Difference Method is the simplest from the three numerical methods that are described in this subchapter in therms of coding and mathematical background needed. The key idea is to replace the derivatives of the governing equations with finite differences which leads to an algebraic system to be solved. Consider the heat (parabolic) equation which has the form $u(x,t)$ such that:

$$u_t = u_{xx}, \qquad \forall (t,x) \in (0,t_F) \times (0,1) \tag{A1}$$

$$u(0,t) = u(1,t) = 0 \qquad \forall t \in (0,t_F) \tag{A2}$$

$$u(x,0) = u_0(x), \qquad \forall x \in [0,1] \tag{A3}$$

where t_F is the final time step of the model and $(0,1)$ is the space domain. The first step in using the finite difference method is the discretization of the domain in a uniform grid:

$$t_n = n\Delta t, \Delta t = \frac{t_f}{N}, \qquad n = 1, \dots, N \tag{A4}$$

$$x_j = j\Delta x, \Delta x = \frac{1}{j}, \qquad j = 1, \ldots, J \tag{A5}$$

Finite difference can be implemented both implicitly and explicitly. The difference between these two forms is that in the explicit scheme the output (e.g. t_{n+1}) depends on itself so there is need for a recursive computation, while in order to solve the explicit schemes an iterative method (for nonlinear problems) or a matrix-inverse (for linear) has to used. The explicit also known as forward scheme takes the form:

$$u_t(x_j, t_n) \approx \frac{u_j^{n+1} - u_j^n}{\Delta t} \tag{A6}$$

$$u_{xx}(x_j, t_n) \approx \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2} \tag{A7}$$

Substituting these two equation into equation Equation A1 the explicit scheme becomes:

$$u_j^{n+1} = u_j^n + \frac{\Delta t}{\Delta x}(u_{j+1}^n - 2u_j^n + u_{j-1}^n), \qquad 1 \leq j \leq J, \qquad 0 \leq n \leq N-1 \tag{A8}$$

The boundary and initial conditions can be calculated as:

$$u_0^n = u_J^n = 0, \qquad 0 \leq n \leq N-1 \tag{A9}$$

$$u_j^0 = u_0(j\Delta x), \qquad 0 \leq j \leq J \tag{A10}$$

The approximated solution u_j^{n+1} can be acquired by marching in time (explicit scheme).

A similar procedure can be used in the implicit scheme where:

$$u_t(x_j, t_n) \approx \frac{(u_j^n) - u_j^{n-1}}{|deltat} \tag{A11}$$

$$u_{xx}(x_j, t_n) \approx \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2} \tag{A12}$$

are being substituted in A1 leads to the implicit scheme:

$$-\frac{\Delta t}{(\Delta x)^2}u_{j-1}^n + (1 + 2\frac{\Delta t}{(\Delta x)^2})u_j^n - \frac{\Delta t}{(\Delta x)^2}u_{j+1}^n = u_j^{n-1} \qquad 1 \leq j \leq J-1 \tag{A13}$$

Both schemes, the explicit and implicit, can be generalized in the so called θ-method which has the form:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{\theta D_+^x D_-^x u_j^{n+1} + (1-\theta) D_+^x D_-^x u_j^n}{(\Delta x)^2} \qquad 0 \leq \theta \leq 1,\ 1 \leq j \leq J-1 \tag{A14}$$

For $\theta = 0$ the Equation A14 takes the form of the explicit method, when $\theta = 1$ takes the form of the implicit method and if $\theta = \frac{1}{2}$ corresponds to the implicit second order Crank-Nicolson scheme.

A1.1.2 Finite Volume Method

A numerical method that has been used in many research fields such as Computational Fluid Dynamics, electromagnetics etc. is the Finite Volume Method. It has the advantage of being able to be applied on complex and unstructured meshes and at the same time is more robust (numerically) than Finite Element Method. The first step in FVM method is the discretization of the domain into smaller called control volumes or cells which they form a grid. The boundaries of of each cell are defined by the grid while the computational node lies in the centre of the cell. During the next step, the governing algebraic equations have to be derived for each control volume. By doing this, an algebraic equation system is formed. The final step is the solution of the algebraic system in order to find the solution of the dependent variables at each volume. Compared to FDM described before, provides good conservation of mass, energy and momentum by using integration on the mesh and can handle complicated domains with ease. Consider the general transport equation

$$\frac{\delta(\rho\phi)}{\delta t} + div(\rho u \phi) = div(\Gamma\ grad\ \phi) + S \tag{A15}$$

where $\frac{\delta(\rho\phi)}{\delta t}$ is the unsteady term, $div(\rho u \phi)$ is the convection term, $div(\Gamma grad \phi)$ is the diffusion term, S is the source term, Γ is the diffusion coefficient and ϕ is the dependent variable (i.e mass, temperature etc.) In the 1-D case the governing equation of diffusion is

$$\frac{d}{dx}(\Gamma \frac{d\phi}{dx}) + S = 0 \tag{A16}$$

As mentioned earlier, the first step is the grid generation. Let P be the general nodal point, W and E are the nodes west and east of P respectively, w and e are the west and east faces of the control volume. For the discretization the 1-D steady state diffusion can be written as:

$$\oint_{\Delta V} \frac{d}{dx}(\Gamma \frac{d\phi}{dx})dV + \int_{\Delta V} S dV = (\Gamma A \frac{d\phi}{dx})(e) - (\Gamma A \frac{d\phi}{dx})(w) + S' \Delta V = 0 \quad \text{(A17)}$$

where A is the segmentation area of the cell, ΔV is the volume of each cell, s' is the average source. To calculate the diffusion coefficients and gradients of the west and east cell node linear approximation can be used:

$$\Gamma_w = \frac{\Gamma_W + \Gamma_P}{2} \qquad \Gamma_e = \frac{\Gamma_P + \Gamma_E}{2} \quad \text{(A18)}$$

while the diffusion fluxes are:

$$\left(\Gamma A \frac{d\phi}{dx}\right)_e = \Gamma_e A_e \left(\frac{\phi_E - \phi_P}{\delta x_{PE}}\right) \quad \text{(A19)}$$

$$\left(\Gamma A \frac{d\phi}{dx}\right)_w = \Gamma_w A_w \left(\frac{\phi_P - \phi_W}{\delta x_{WP}}\right) \quad \text{(A20)}$$

Which after rearrangement gives the equation:

$$\left(\frac{\Gamma_e}{\delta x_{PE}} A_e + \frac{\Gamma_w}{\delta x_{WP}} A_w - S_p\right)\phi_P = \left(\frac{\Gamma_w}{\delta x_{WP}} A_w\right)\phi_W + \left(\frac{\Gamma_e}{\delta x_{PE}} A_e\right)\phi_E + S_u \quad \text{(A21)}$$

Identifying the coefficients ϕ_W and ϕ_E as α_w and α_E respectively the discretised convection/diffusion equation can be written as:

$$\alpha_P \phi_P = \alpha_W \phi_W + \alpha_E \phi_E + S_u \quad \text{(A22)}$$

where

$$\begin{aligned} \alpha_W &= \frac{\Gamma_w}{\delta x_{WP}} A_w \\ \alpha_E &= \frac{\Gamma_e}{\delta x_{PE}} A_e \\ \alpha_P &= \alpha_W + \alpha_E - S_P \end{aligned} \quad \text{(A23)}$$

The 1-D convection diffusion problem discussed earlier, can be extended in 2-D where in addition to the west and east nodes of the central node P now there are also the south S and north N. The steady state diffusion equation is:

$$\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right)+S=0 \tag{A24}$$

which after discretization takes the form:

$$\alpha_P\phi_P=\alpha_W\phi_W+\alpha_E\phi_E+\alpha_S\phi_S+\alpha_n\phi_N+S_u \tag{A25}$$

where

$$\begin{aligned}\alpha_W&=\frac{\Gamma_w}{\delta x_{WP}}A_w\\ \alpha_E&=\frac{\Gamma_e}{\delta x_{PE}}A_e\\ \alpha_S&=\frac{\Gamma_s}{\delta y_{SP}}A_s\\ \alpha_N&=\frac{\Gamma_n}{\delta y_{PN}}A_n\\ \alpha_P&=\alpha_W+\alpha_E-+\alpha_S+\alpha_N-S_P\end{aligned} \tag{A26}$$

The FVM method described above in based on the central differencing scheme. Although it is a second order method, it is stable for Peclet number below 2 ($PE\leq 2$). The Peclet number is a measure of the relative strengths of convection and diffusion and is defined as:

$$Pe=\frac{\rho u}{\Gamma/\delta x} \tag{A27}$$

with δx being the length of the control volume. This instability for high velocity flows had as a result the need for the development of schemes such as the upwind differencing and hybrid schemes. The upwind differencing scheme introduces the direction of the flows [178]. Spalding in [179] combined central and upwind differencing and proposed the hybrid differencing scheme. The main idea behind it is to use the second-order central differencing while $Pe\leq 2$ and upwind which is good in cases where the transportiveness has to be satisfied. The implementation of the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) can be found in chapter 4.

A1.1.3 Finite Element Method

Finite Element Method (FEM) is another numerical method for solving partial Differential Equations (PDEs). Due to its capability of dealing with complicated mashes it has been used primarily is solid mechanics[180] but its use is not limited to that, it has also been used in other fields such as biomedicine [181], electromagnetics [182] etc. Basic functions cannot provide analytical solutions for PDEs when dealing with complex geometries. FEM discretizes the original domain Ω into smaller subdomains called elements and can have different shapes (triangular, rectangular, polygons). The elements form a mesh and its vertex called node which also can appear in on the edges or the interiors.

Consider the following steady state equation describing the temperature distribution on a heat sink.

$$\nabla(-\lambda\nabla T) = g(T, \boldsymbol{x}) \text{ in } \Omega \tag{A28}$$

where Ω is the original domain of the problem under consideration, λ is the thermal conductivity, T is the temperature with boundary conditions:

$$\begin{aligned} T &= T_0 \text{ in } \partial\Omega_1 \\ -(\lambda\nabla T)\boldsymbol{n} &= h_T(T - T_{amp}) \text{ in } \partial\Omega_2 \\ -(\lambda\nabla T) &= 0 \text{ in } \partial\Omega_3 \end{aligned} \tag{A29}$$

where h_T is the heat coefficient and n is the outward unit normal vector.

The first step in FEM is the multiplication of both sides of Equation A28 with a test function ξ and then to integrate the equation to get its weak form:

$$\int_\Omega \nabla(-\lambda\nabla T)\xi dV = \int_\Omega g\xi dV \tag{A30}$$

The test function is chosen from a function space that is able to expunge the Dirichlet boundary condition. By using Green's identity and integrating by parts Equation A30 becomes

$$\int_\Omega \lambda\nabla T\nabla\xi dV + \int_{\partial\Omega}(-\lambda\nabla T)\boldsymbol{n}\xi dS = \int_\Omega g\xi dV \tag{A31}$$

The weak formulation unbends the continuity requirements of the strong formulation such as the number of the partial derivatives and leads to the use of easy to construct polynomials e.g Lagrange polynomials. Having the weak formulation the next step is the discretization of Equation A31 such that $T = T_h$ which can be written as a combination of basis function $\phi_i,\ i, \ldots, N$

$$T_h(\boldsymbol{x}) = \sum_i T_i \phi_i(\boldsymbol{x}) \tag{A32}$$

which leads to the discretised equivalent of Equation A31:

$$\sum_i T_i \int_\Omega \lambda \nabla\phi_i \nabla\phi_j dV + \sum_i \int_{\partial\Omega} (-\lambda T_i \nabla\phi_i)\boldsymbol{n}\phi_j dS = \int_\Omega g\Big(\sum_i T_i\phi_i\Big)\phi_j dV \tag{A33}$$

After the discretization and having the boundary conditions prescribed the Equation A33 can be written in a matrix form as:

$$\boldsymbol{AT_h} = \boldsymbol{b} \tag{A34}$$

where $\boldsymbol{T_h}$ is the dependent variable, and $\boldsymbol{A}$ is the coefficient matrix also known as stiffness matrix.

Appendix B: Variants of POD

The *method of snapshots* is an indirect application of POD suitable for problems in which $m \ll d$. A temporal autocovariance function $K(t,t';\boldsymbol{\xi}) = \int_{\mathcal{D}} u(\mathbf{x},t;\boldsymbol{\xi})u(\mathbf{x},t';\boldsymbol{\xi})d\mathbf{x}$ is defined, with associated operator:

$$\mathcal{K}a_i(t;\boldsymbol{\xi}) := \int_0^T K(t,t';\boldsymbol{\xi})a_i(t';\boldsymbol{\xi})dt'$$

The orthogonal eigenfunctions $a_i(t;\boldsymbol{\xi})$ of $\mathcal{K}$ are the POD coefficients and the eigenvalues are identical to those of $\mathcal{C}$. Using $\mathbb{E}[a_i(t;\boldsymbol{\xi})a_j(t;\boldsymbol{\xi})] = \lambda'_i(\boldsymbol{\xi})\delta_{ij}$, the POD modes are given by $v_i(\mathbf{x};\boldsymbol{\xi}) = (1/\lambda'_i(\boldsymbol{\xi}))\int_0^T u(\mathbf{x},t;\boldsymbol{\xi})a_i(t;\boldsymbol{\xi})dt$. The discrete form (in space and time) of the eigenvalue problem is $\mathbf{X}(\boldsymbol{\xi})^T\mathbf{X}(\boldsymbol{\xi})\mathbf{a}_i(\boldsymbol{\xi}) = \lambda_i\mathbf{a}_i(\boldsymbol{\xi})$, where $\mathbf{K}(\boldsymbol{\xi}) := \mathbf{X}(\boldsymbol{\xi})^T\mathbf{X}(\boldsymbol{\xi})$ is a kernel matrix with entries $K_{ij} = \mathbf{u}^{(i)}(\boldsymbol{\xi})^T\mathbf{u}^{(j)}(\boldsymbol{\xi})$, i.e., the space-discrete form of $K(t^{(i)},t^{(j)};\boldsymbol{\xi})$. The eigendecomposition is $\mathbf{K}(\boldsymbol{\xi}) = \mathbf{A}(\boldsymbol{\xi})\boldsymbol{\Lambda}(\boldsymbol{\xi})\mathbf{A}(\boldsymbol{\xi})^T$, where $\boldsymbol{\Lambda}(\boldsymbol{\xi}) = \mathrm{diag}(\lambda_1(\boldsymbol{\xi}),\ldots,\lambda_m(\boldsymbol{\xi}))$ and the columns of $\mathbf{A}(\boldsymbol{\xi})$ are given by the $\mathbf{a}_i(\boldsymbol{\xi})$. The j-th component $a_{i,j}(\boldsymbol{\xi})$ of $\mathbf{a}_i(\boldsymbol{\xi})$ approximates $a_i(t^{(j)};\boldsymbol{\xi})$ yielding the discrete-time approximation $v_i(\mathbf{x};\boldsymbol{\xi}) = (1/\lambda_i(\boldsymbol{\xi}))\sum_{j=1}^m u(\mathbf{x},t^{(j)};\boldsymbol{\xi})a_{i,j}(\boldsymbol{\xi})$, i.e., a linear combination of the snapshots. In the fully-discrete case, using the normalization $\mathbf{a}_i(\boldsymbol{\xi}) \mapsto \mathbf{a}'_i(\boldsymbol{\xi})/\sqrt{\lambda_i(\boldsymbol{\xi})}$, $\mathbf{v}_i(\boldsymbol{\xi}) = \mathbf{X}(\boldsymbol{\xi})\mathbf{a}'_i(\boldsymbol{\xi})/\sqrt{\lambda_i(\boldsymbol{\xi})}$ is obtained.

These relationships are also evident from the singular value decomposition (SVD) of $\mathbf{X}(\boldsymbol{\xi})$, that is $\mathbf{X}(\boldsymbol{\xi}) = \mathbf{A}'(\boldsymbol{\xi})\boldsymbol{\Lambda}(\boldsymbol{\xi})^{1/2}\mathbf{V}(\boldsymbol{\xi})^T$, where the columns of $\mathbf{V}(\boldsymbol{\xi})$ are given by the $\mathbf{v}_i(\boldsymbol{\xi})$ and the columns of $\mathbf{A}'(\boldsymbol{\xi})$ are given by the $\mathbf{a}'_i(\boldsymbol{\xi})$. In this context, the columns of $\mathbf{A}'(\boldsymbol{\xi})$ and $\mathbf{V}(\boldsymbol{\xi})$, given respectively by the eigenvectors of $\mathbf{K}(\boldsymbol{\xi})$ and $\mathbf{C}(\boldsymbol{\xi})$, are referred to as left and right singular vectors. It is straightforward to show that $\mathbf{v}_i(\boldsymbol{\xi}) = k\mathbf{X}(\boldsymbol{\xi})\mathbf{a}'_i(\boldsymbol{\xi})$ for $k \in \mathbb{R}$. Thus, the earlier relationship is recovered by taking $k = 1/\sqrt{\lambda_i(\boldsymbol{\xi})}$ to normalise the $\mathbf{v}_i(\boldsymbol{\xi})$.

References

[1] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical science*, pp. 409–423, 1989.

[2] M. Kennedy and A. O'Hagan, "Bayesian calibration of computer codes," *J. R. Stat. Soc. Ser. B Stat. Methodol.*, vol. 63, pp. 425–464, 2001.

[3] T. Santner, B. Williams, and W. Notz, *The Design and Analysis of Computer Experiments.* Springer, 2003.

[4] 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.

[5] A. Keane and P. Nair, *Computational Approaches for Aerospace Design.* John-Wiley and Sons, 2005.

[6] 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.

[7] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning.* MIT Press, Cambridge MA, USA, 2006.

[8] B. Ganapathysubramanian and N. Zabaras, "Sparse grid collocation schemes for stochastic natural convection problems," *Journal of Computational Physics*, vol. 225, no. 1, pp. 652–685, 2007.

[9] G. Sangalli, “Capturing small scales in elliptic problems using a residual-free bubbles finite element method,” *Multiscale Modeling & Simulation*, vol. 1, no. 3, pp. 485–503, 2003.

[10] N. M. Alexandrov, R. M. Lewis, C. R. Gumbert, L. L. Green, and P. A. Newman, “Approximation and model management in aerodynamic optimization with variable-fidelity models,” *Journal of Aircraft*, vol. 38, no. 6, pp. 1093–1101, 2001.

[11] A. March and K. Willcox, “Provably convergent multifidelity optimization algorithm not requiring high-fidelity derivatives,” *AIAA journal*, vol. 50, no. 5, pp. 1079–1089, 2012.

[12] R. M. Lewis and S. G. Nash, “Model problems for the multigrid optimization of systems governed by differential equations,” *SIAM Journal on Scientific Computing*, vol. 26, no. 6, pp. 1811–1837, 2005.

[13] S. Arridge, J. Kaipio, V. Kolehmainen, M. Schweiger, E. Somersalo, T. Tarvainen, and M. Vauhkonen, “Approximation errors and model reduction with an application in optical diffusion tomography,” *Inverse Problems*, vol. 22, no. 1, p. 175, 2006.

[14] P. Jenny, S. Lee, and H. A. Tchelepi, “Multi-scale finite-volume method for elliptic problems in subsurface flow simulation,” *Journal of Computational Physics*, vol. 187, no. 1, pp. 47–67, 2003.

[15] P. Bastian and V. Reichenberger, *Multigrid for higher order discontinuous Galerkin finite elements applied to groundwater flow.* Universität Heidelberg. Interdisziplinäres Zentrum für Wissenschaftliches Rechnen [IWR], 2000.

[16] M. Mansour and A. Spink, “Grid refinement in cartesian coordinates for groundwater flow models using the divergence theorem and taylor’s series,” *Ground water*, vol. 51, no. 1, pp. 66–75, 2013.

[17] G. Kourakos and A. Mantoglou, “Pumping optimization of coastal aquifers based on evolutionary algorithms and surrogate modular

neural network models," *Advances in water resources*, vol. 32, no. 4, pp. 507–521, 2009.

[18] S. Yan and B. Minsker, "Optimal groundwater remediation design using an adaptive neural network genetic algorithm," *Water Resources Research*, vol. 42, no. 5, 2006.

[19] 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.

[20] D. A. Bau and A. S. Mayer, "Stochastic management of pump-and-treat strategies using surrogate functions," *Advances in water resources*, vol. 29, no. 12, pp. 1901–1917, 2006.

[21] H. Yoon, S.-C. Jun, Y. Hyun, G.-O. Bae, and K.-K. Lee, "A comparative study of artificial neural networks and support vector machines for predicting groundwater levels in a coastal aquifer," *Journal of Hydrology*, vol. 396, no. 1, pp. 128–138, 2011.

[22] S. M. Wild, R. G. Regis, and C. A. Shoemaker, "Orbit: Optimization by radial basis function interpolation in trust-regions," *SIAM Journal on Scientific Computing*, vol. 30, no. 6, pp. 3197–3219, 2008.

[23] J. Wu, C. Zhang, and Z. Chen, "An online method for lithium-ion battery remaining useful life estimation using importance sampling and neural networks," *Applied Energy*, vol. 173, no. Supplement C, pp. 134 – 140, 2016.

[24] Y. Bicer, I. Dincer, and M. Aydin, "Maximizing performance of fuel cell using artificial neural network approach for smart grid applications," *Energy*, vol. 116, pp. 1205–1217, 2016.

[25] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 6655–6659, IEEE, 2013.

[26] 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.

[27] W. Lv and A. Henry, “Direct calculation of modal contributions to thermal conductivity via green–kubo modal analysis,” *New Journal of Physics*, vol. 18, no. 1, p. 013028, 2016.

[28] M. Ghommem, M. Presho, V. M. Calo, and Y. Efendiev, “Mode decomposition methods for flows in high-contrast porous media. global–local approach,” *Journal of computational physics*, vol. 253, pp. 226–238, 2013.

[29] K. Willcox and A. Megretski, “Fourier series for accurate, stable, reduced-order models in large-scale linear applications,” *SIAM Journal on Scientific Computing*, vol. 26, no. 3, pp. 944–962, 2005.

[30] 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.

[31] M. C. Kennedy, C. W. Anderson, S. Conti, and A. O’Hagan, “Case studies in Gaussian process modelling of computer codes,” *Reliability Engineering & System Safety*, vol. 91, no. 10, pp. 1301–1309, 2006.

[32] 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.

[33] 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.

[34] J. McFarland, S. Mahadevan, V. Romero, and L. Swiler, “Calibration and uncertainty analysis for computer simulations with multivariate output,” *AIAA Journal*, vol. 46, pp. 1253–1265., 2008.

[35] S. Conti and A. O'Hagan, "Bayesian emulation of complex multi-output and dynamic computer models," *Journal of Statistical Planning and Inference*, vol. 140, no. 3, pp. 640–651, 2010.

[36] H. Wackernagel, *Multivariate Geostatistics: An Introduction with Applications.* Springer, Berlin, 1995.

[37] 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, pp. 145–158, 2014.

[38] 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.

[39] J. Rougier, "Efficient emulators for multivariate deterministic functions," *J. Computational and Graphical Statistics*, vol. 17, pp. 827–843, 2008.

[40] N. Lawrence, "Probabilistic non-linear principal component analysis with gaussian process latent variable models," *Journal of Machine Learning Research*, vol. 6, pp. 1783–1816, 2005.

[41] M. Alvarez and N. D. Lawrence, "Sparse convolved gaussian processes for multi-output regression," in *Advances in neural information processing systems*, pp. 57–64, 2009.

[42] A. Damianou and N. Lawrence, "Deep gaussian processes," in *Artificial Intelligence and Statistics*, pp. 207–215, 2013.

[43] 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.

[44] 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.

[45] 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.

[46] S. Deparis and G. Rozza, "Reduced basis method for multi-parameter-dependent steady navier–stokes equations: Applications to natural convection in a cavity," *Journal of Computational Physics*, vol. 228, no. 12, pp. 4359 – 4378, 2009.

[47] M. D. Gunzburger, J. S. Peterson, and J. N. Shadid, "Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, pp. 1030–1047, 2007.

[48] D. J. Knezevic, N.-C. Nguyen, and A. T. Patera, "Reduced basis approximation and a posteriori error estimation for the parametrized unsteady Boussinesq equations," *Mathematical Models and Methods in Applied Sciences*, vol. 21, no. 07, pp. 1415–1442, 2011.

[49] N. Nguyen, "A posteriori error estimation and basis adaptivity for reduced-basis approximation of nonaffine-parametrized linear elliptic partial differential equations," *Journal of Computational Physics*, vol. 227, no. 2, pp. 983 – 1006, 2007.

[50] A. Quarteroni, G. Rozza, and A. Manzoni, "Certified reduced basis approximation for parametrized partial differential equations and applications," *Journal of Mathematics in Industry*, vol. 1, no. 1, pp. 1–49, 2011.

[51] K. P. Murphy, *Machine learning: a probabilistic perspective.* MIT press, 2012.

[52] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[53] B. Widrow and M. E. Hoff, "Adaptive switching circuits," tech. rep., STANFORD UNIV CA STANFORD ELECTRONICS LABS, 1960.

[54] F. Rosenblatt, “Principles of neurodynamics,” 1962.

[55] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representation by back propagation,” *Parallel distributed processing: exploration in the microstructure of cognition*, vol. 1, 1986.

[56] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural networks*, vol. 61, pp. 85–117, 2015.

[57] Y. Bengio *et al.*, “Learning deep architectures for ai,” *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

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

[59] V. Vapnik and A. Lerner, “Generalized portrait method for pattern recognition,” *Automation and Remote Control*, vol. 24, no. 6, pp. 774–780, 1963.

[60] B. Boser, I. Guyon, and V. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152, ACM Press, 1992.

[61] W. S. Torgerson, “Multidimensional scaling: I. theory and method,” *Psychometrika*, vol. 17, no. 4, pp. 401–419, 1952.

[62] I. Borg and P. J. Groenen, *Modern multidimensional scaling: Theory and applications.* Springer Science & Business Media, 2005.

[63] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[64] R. W. Floyd, “Algorithm 97: shortest path,” *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[65] J. Ham, D. 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*, ICML ’04, (New York, NY, USA), pp. 369–376, ACM, 2004.

[66] H. Choi and S. Choi, “Kernel isomap,” *Electronics letters*, vol. 40, no. 25, pp. 1612–1613, 2004.

[67] F. R. Chung, *Spectral Graph Theory*, vol. 92. American Mathematical Soc., 1997.

[68] K. Zhou, J. C. Doyle, K. Glover, *et al.*, *Robust and optimal control*, vol. 40. Prentice hall New Jersey, 1996.

[69] Y. Saad, “Krylov subspace methods for solving large unsymmetric linear systems,” *Mathematics of computation*, vol. 37, no. 155, pp. 105–126, 1981.

[70] B. Moore, “Principal component analysis in linear systems: Controllability, observability, and model reduction,” *IEEE transactions on automatic control*, vol. 26, no. 1, pp. 17–32, 1981.

[71] Y. Saad, *Iterative methods for sparse linear systems.* SIAM, 2003.

[72] L. Sirovich, “Turbulence and the dynamics of coherent structures. i. coherent structures,” *Quarterly of applied mathematics*, vol. 45, no. 3, pp. 561–571, 1987.

[73] M. Condon and R. Ivanov, “Empirical balanced truncation of nonlinear systems,” *Journal of Nonlinear Science*, vol. 14, no. 5, pp. 405–414, 2004.

[74] B. Lohmann and R. Eid, “Efficient order reduction of parametric and nonlinear models by superposition of locally reduced models,” in *Methoden und Anwendungen der Regelungstechnik. Erlangen-Münchener Workshops*, pp. 27–36, 2007.

[75] N. Nguyen and J. Peraire, “An efficient reduced-order modeling approach for non-linear parametrized partial differential equations,” *International Journal for Numerical Methods in Engineering*, vol. 76, no. 1, pp. 27–55, 2008.

[76] U. Baur, C. Beattie, P. Benner, and S. Gugercin, “Interpolatory projection methods for parameterized model reduction,” *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2489–2518, 2011.

[77] L. Feng and P. Benner, “Arobust algorithm for parametric model order reduction,” *PAMM*, vol. 7, no. 1, pp. 1021501–1021502, 2007.

[78] P. Gunupudi, R. Khazaka, and M. Nakhla, “Analysis of transmission line circuits using multidimensional model reduction techniques,” *IEEE Transactions on Advanced Packaging*, vol. 25, no. 2, pp. 174–180, 2002.

[79] R. Eid, B. Salimbahrami, B. Lohmann, E. B. Rudnyi, and J. G. Korvink, “Parametric order reduction of proportionally damped second-order systems,” *Sensors and Materials, Tokyo*, vol. 19, no. 3, pp. 149–164, 2007.

[80] L. Meier and D. Luenberger, “Approximation of linear constant systems,” *IEEE Transactions on Automatic Control*, vol. 12, no. 5, pp. 585–588, 1967.

[81] S. Gugercin, A. C. Antoulas, and C. Beattie, “H_2 model reduction for large-scale linear dynamical systems,” *SIAM journal on matrix analysis and applications*, vol. 30, no. 2, pp. 609–638, 2008.

[82] R. Everson and L. Sirovich, “Karhunen–loeve procedure for gappy data,” *JOSA A*, vol. 12, no. 8, pp. 1657–1664, 1995.

[83] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera, “An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations,” *Comptes Rendus Mathematique*, vol. 339, no. 9, pp. 667 – 672, 2004.

[84] B. Haasdonk and M. Ohlberger, “Reduced basis method for explicit finite volume approximations of nonlinear conservation laws,” in *Proc. 12th International Conference on Hyperbolic Problems: Theory, Numerics, Application*, 2008.

[85] M. Drohmann, B. Haasdonk, and M. Ohlberger, “Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation,” *SIAM Journal on Scientific Computing*, vol. 34, no. 2, pp. A937–A969, 2012.

[86] M. Ohlberger and G. Rozza, "A reduced basis method for evolution schemes with parameter-dependent explicit operators," *Electronic Transactions on Numerical Analysis*, vol. 32, pp. 145–161, 2008.

[87] S. Chaturantabut and D. C. Sorensen, "Discrete empirical interpolation method for nonlinear model reduction," *Technical Report: CAAM, Rice University*, 2009.

[88] 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.

[89] K. Carlberg, C. Bou-Mosleh, and C. Farhat, "Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations," *International Journal for Numerical Methods in Engineering*, vol. 86, no. 2, pp. 155–181, 2011.

[90] E. Borgonovo and E. Plischke, "Sensitivity analysis: a review of recent advances," *European Journal of Operational Research*, vol. 248, no. 3, pp. 869–887, 2016.

[91] H. Christopher Frey and S. R. Patil, "Identification and review of sensitivity analysis methods," *Risk analysis*, vol. 22, no. 3, pp. 553–578, 2002.

[92] M. Saisana, A. Saltelli, and S. Tarantola, "Uncertainty and sensitivity analysis techniques as tools for the quality assessment of composite indicators," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 168, no. 2, pp. 307–323, 2005.

[93] K. Beven, "Prophecy, reality and uncertainty in distributed hydrological modelling," *Advances in water resources*, vol. 16, no. 1, pp. 41–51, 1993.

[94] M. D. Morris, "Factorial sampling plans for preliminary computational experiments," *Technometrics*, vol. 33, no. 2, pp. 161–174, 1991.

[95] M. Cavazzuti, *Optimization methods: from theory to design scientific and technological aspects in mechanics.* Springer Science & Business Media, 2012.

[96] 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.

[97] I. Sobol, "Uniformly distributed sequences with an additional uniform property," *USSR Computational Mathematics and Mathematical Physics*, vol. 16, no. 5, pp. 236 – 242, 1976.

[98] M. Y. Hussaini, C. L. Streett, and T. A. Zang, "Spectral methods for partial differential equations," 1983.

[99] A. Bueno-Orovio, V. M. Pérez-García, and F. H. Fenton, "Spectral methods for partial differential equations in irregular domains: the spectral smoothed boundary method," *SIAM Journal on Scientific Computing*, vol. 28, no. 3, pp. 886–900, 2006.

[100] 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.

[101] 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.

[102] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, pp. 1299–1319, July 1998.

[103] I. Jolliffe, *Principal Component Analysis.* Springer Series in Statistics, Springer, 2002.

[104] 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.

[105] 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, 2002.

[106] S. Mika, B. Scholkopf, A. Smola, K.-R. Muller, M. Scholz, and G. Ratsch, "Kernel PCA and de-noising in feature spaces," *Advances in neural information processing systems II*, vol. 11, pp. 536–542, 1999.

[107] 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, Jun 1997.

[108] J. Moody, *Prediction Risk and Architecture Selection for Neural Networks*, pp. 147–165. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994.

[109] J. M. Correa, F. A. Farret, V. A. Popov, and M. G. Simoes, "Sensitivity analysis of the modeling parameters used in simulation of proton exchange membrane fuel cells," *IEEE Transactions on energy conversion*, vol. 20, no. 1, pp. 211–218, 2005.

[110] D. Zhao, F. Gao, P. Massonnat, M. Dou, and A. Miraoui, "Parameter sensitivity analysis and local temperature distribution effect for a pemfc system," *IEEE Transactions on Energy Conversion*, vol. 30, no. 3, pp. 1008–1018, 2015.

[111] E. Testa, P. Maggiore, L. Pace, and M. D. L. Dalla Vedova, "Sensitivity analysis for a pem fuel cell model aimed to optimization," *WSEAS Transactions on Power Systems*, vol. 10, pp. 171–179, 2015.

[112] D. Zhou, K. Zhang, A. Ravey, F. Gao, and A. Miraoui, "Parameter sensitivity analysis for fractional-order modeling of lithium-ion batteries," *Energies*, vol. 9, no. 3, p. 123, 2016.

[113] S. Zhao and D. A. Howey, "Global sensitivity analysis of battery equivalent circuit model parameters," in *Vehicle Power and Propulsion Conference (VPPC), 2016 IEEE*, pp. 1–4, IEEE, 2016.

[114] B. Laoun, M. W. Naceur, A. Khellaf, and A. M. Kannan, "Global sensitivity analysis of proton exchange membrane fuel cell model," *International Journal of Hydrogen Energy*, vol. 41, no. 22, pp. 9521–9528, 2016.

[115] I. M. Sobol, "Sensitivity estimates for nonlinear mathematical models," *Mathematical Modelling and Computational Experiments*, vol. 1, no. 4, pp. 407–414, 1993.

[116] J. E. Oakley and A. O'Hagan, "Probabilistic sensitivity analysis of complex models: a bayesian approach," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 66, no. 3, pp. 751–769, 2004.

[117] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global sensitivity analysis: the primer.* John Wiley & Sons, 2008.

[118] F. Pianosi, K. Beven, J. Freer, J. W. Hall, J. Rougier, D. B. Stephenson, and T. Wagener, "Sensitivity analysis of environmental models: A systematic review with practical workflow," *Environmental Modelling & Software*, vol. 79, pp. 214–232, 2016.

[119] J. Norton, "An introduction to sensitivity assessment of simulation models," *Environmental Modelling & Software*, vol. 69, pp. 166–174, 2015.

[120] R. Cukier, C. Fortuin, K. E. Shuler, A. Petschek, and J. Schaibly, "Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. i theory," *The Journal of chemical physics*, vol. 59, no. 8, pp. 3873–3878, 1973.

[121] A. Saltelli and I. M. Sobol', "Sensitivity analysis for nonlinear mathematical models: numerical experience," *Matematicheskoe Modelirovanie*, vol. 7, no. 11, pp. 16–28, 1995.

[122] 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, no. 2, pp. 259–270, 2010.

[123] L. Bastos and A. O'Hagan, "Diagnostics for gaussian process emulators," *Technometrics*, vol. 51, pp. 425–438, 2009.

[124] A. A. Shah, "Surrogate modeling for spatially distributed fuel cell models with applications to uncertainty quantification," *Journal of Electrochemical Energy Conversion and Storage*, vol. 14, no. 1, p. 011006, 2017.

[125] V. I. Krylov and A. H. Stroud, *Approximate calculation of integrals.* Dover Books on Mathematics, Mineola, NY: Dover, 2006.

[126] M. Doyle, J. Newman, A. S. Gozdz, C. N. Schmutz, and J.-M. Tarascon, "Comparison of modeling predictions with experimental data from plastic lithium ion cells," *Journal of the Electrochemical Society*, vol. 143, no. 6, pp. 1890–1903, 1996.

[127] A. Nyman, T. G. Zavalis, R. Elger, M. Behm, and G. Lindbergh, "Analysis of the polarization in a li-ion battery cell by numerical simulations," *Journal of The Electrochemical Society*, vol. 157, no. 11, pp. A1236–A1246, 2010.

[128] F. Pianosi, F. Sarrazin, and T. Wagener, "A matlab toolbox for global sensitivity analysis," *Environmental Modelling & Software*, vol. 70, pp. 80–85, 2015.

[129] B. Efron, "Bootstrap methods: another look at the jackknife," in *Breakthroughs in statistics*, pp. 569–593, Springer, 1992.

[130] A. Shah, W. Xing, and V. Triantafyllidis, "Reduced-order modelling of parameter-dependent, linear and nonlinear dynamic partial differential

equation models," in *Proc. R. Soc. A*, vol. 473, p. 20160809, The Royal Society, 2017.

[131] A. J. Newman, "Model reduction via the karhunen-loeve expansion part i: An exposition," tech. rep., 1996.

[132] E. Wong, "Stochastic processes in information and dynamical systems," 1971.

[133] 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.

[134] 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.

[135] M. Rathinam and L. R. Petzold, "A new look at proper orthogonal decomposition," *SIAM Journal on Numerical Analysis*, vol. 41, no. 5, pp. 1893–1925, 2003.

[136] 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.

[137] W. Xing, A. A. Shah, and P. B. Nair, "Reduced dimensional Gaussian process emulators of parametrized partial differential equations based on Isomap," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 471, no. 2174, 2014.

[138] 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.

[139] 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.

[140] Y. Rathi, S. Dambreville, and A. Tannenbaum, "Statistical shape analysis using kernel PCA," vol. 6064, pp. 60641B–60641B–8, 2006.

[141] 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.

[142] M. M. Baumann, "Nonlinear model order reduction using pod/deim for optimal control of burgers equation," *Delft University of Technology, Netherlands*, 2013.

[143] 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.

[144] I. Kalashnikova, B. van Bloemen Waanders, S. Arunajatesan, and M. Barone, "Stabilization of projection-based reduced order models for linear time-invariant systems via optimization-based eigenvalue reassignment," *Computer Methods in Applied Mechanics and Engineering*, vol. 272, pp. 251–270, 2014.

[145] T. Bui-Thanh, K. Willcox, O. Ghattas, and B. van Bloemen Waanders, "Goal-oriented, model-constrained optimization for reduction of large-scale systems," *Journal of Computational Physics*, vol. 224, no. 2, pp. 880–896, 2007.

[146] 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.

[147] C. W. Rowley, T. Colonius, and R. M. Murray, "Model reduction for compressible flows using pod and galerkin projection," *Physica D: Nonlinear Phenomena*, vol. 189, no. 1-2, pp. 115–129, 2004.

[148] P. Etyngier, F. Segonne, 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.

[149] S. Gepshtein and Y. Keller, "Image completion by diffusion maps and spectral relaxation," *IEEE Transactions on Image Processing*, vol. 22, no. 8, pp. 2983–2994, 2013.

[150] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[151] B. Schölkopf and A. Smola, "Support vector machines," *Encyclopedia of Biostatistics*, 2005.

[152] D. Donoho, C. Chui, R. R. Coifman, and S. Lafon, "Special Issue: Diffusion Maps and Wavelets Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 5 – 30, 2006.

[153] 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, no. 21, pp. 7426–7431, 2005.

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

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

[156] 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.

[157] D. Kushnir, A. Haddad, and R. R. Coifman, "Anisotropic diffusion on sub-manifolds with application to earth structure classification,"

Applied and Computational Harmonic Analysis, vol. 32, no. 2, pp. 280 – 294, 2012.

[158] 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.

[159] I. Andrianakis and P. G. Challenor, “The effect of the nugget on Gaussian process emulators of computer models,” *Computational Statistics & Data Analysis*, vol. 56, no. 12, pp. 4215 – 4228, 2012.

[160] L. S. Bastos and A. O’Hagan, “Diagnostics for Gaussian process emulators,” *Technometrics*, vol. 51, no. 4, pp. 425–438, 2009.

[161] C. Bishop, “Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn,” 2007.

[162] A. E. Gelfand, A. M. Schmidt, S. Banerjee, and C. F. Sirmans, “Nonstationary multivariate process modeling through spatially varying coregionalization,” *Test*, vol. 13, no. 2, pp. 263–312, 2004.

[163] 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.

[164] J. T. Y. Kwok and I. W. H. Tsang, “The pre-image problem in kernel methods,” *IEEE Transactions on Neural Networks*, vol. 15, pp. 1517–1525, Nov 2004.

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

[166] P. Etyngier, F. Ségonne, and R. Keriven, “Shape priors using manifold learning techniques,” in *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*, pp. 1–8, 2007.

[167] N. Thorstensen, F. Segonne, and R. Keriven, *Pre-image as Karcher Mean Using Diffusion Maps: Application to Shape and Image Denoising*, pp. 721–732. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[168] X. Ma and N. Zabaras, "Kernel principal component analysis for stochastic input model generation," *Journal of Computational Physics*, vol. 230, no. 19, pp. 7311 – 7331, 2011.

[169] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[170] C. K. Williams, "On a connection between kernel PCA and metric multidimensional scaling," *Machine Learning*, vol. 46, no. 1, pp. 11–19, 2002.

[171] M. Hossain and M. Wilson, "Natural convection flow in a fluid–saturated porous medium enclosed by non-isothermal walls with heat generation," *International Journal of Thermal Sciences*, vol. 41, no. 5, pp. 447 – 454, 2002.

[172] "COMSOL Multiphysics 5.0, Free Convection in Porous Media, last accessed 29 January 2016,"

[173] B. Seibold, "A compact and fast Matlab code solving the incompressible Navier-Stokes equations on rectangular domains, last accessed 29 January 2016," 2008.

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

[175] K. Broka and P. Ekdunge, "Modelling the PEM fuel cell cathode," *Journal of Applied Electrochemistry*, vol. 27, no. 3, pp. 281–289, 1997.

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

[177] "COMSOL Multiphysics 5.0, Species Transport in the Gas Diffusion Layers of a PEM, last accessed 29 January 2016,"

[178] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics: the finite volume method.* Pearson Education, 2007.

[179] M. Maja, P. Tosco, and M. Vanni, "A one-dimensional model of gas-diffusion electrodes for O_2 reduction," *J. Electrochem. Soc.*, vol. 148, no. 12, pp. A1368–A1376, 2001.

[180] O. C. Zienkiewicz and R. L. Taylor, *The finite element method: solid mechanics*, vol. 2. Butterworth-heinemann, 2000.

[181] B. van Rietbergen, H. Weinans, R. Huiskes, and A. Odgaard, "A new method to determine trabecular bone elastic properties and loading using micromechanical finite-element models," *Journal of biomechanics*, vol. 28, no. 1, pp. 69–81, 1995.

[182] J.-M. Jin, *The finite element method in electromagnetics.* John Wiley & Sons, 2015.