SURROGATE MODELING FOR UNCERTAINTY QUANTIFICATION IN SYSTEMS CHARACTERIZED BY EXPENSIVE AND HIGH-DIMENSIONAL NUMERICAL SIMULATORS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Rohit K. Tripathy

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Ilias Bilionis, Chair

School of Mechanical Engineering

Dr. Marcial Gonzalez

School of Mechanical Engineering

Dr. Marisol Koslowski

School of Mechanical Engineering

Dr. Shirley Dyke

Lyles School of Civil Engineering

Approved by:

Dr. Nicole L. Key

School of Mechanical Engineering

ACKNOWLEDGMENTS

I would like to acknowledge several people who have contributed to my personal and professional growth during my time as a graduate student. First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Ilias Bilionis, for being a fantastic mentor. I am extremely grateful for his guidance and support over the years. I am thankful to all members, past and present, of the Predictive Science Lab - Nimish, Piyush, Sharmila, Parth, Alex, Vanessa, Murali, Salar, Sabareesh, Atharva, and others, for their friendship. I would also like to acknowledge all of the wonderful people outside of our group with whom I formed friendships and acquaintances during my time at Purdue.

I would also like to thank and acknowledge the many professional acquaintances I developed over the past few years including Julie, Emil, Vishwas and others from Argonne National Lab, and Rossen, Assaf, Fabien and others from JPMorgan. Each of these individuals have contributed to my professional and technical development during the brief time I worked with them.

Last, but certainly not the least, I would like to thank my girlfriend Alexandra. She's the best.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

ABSTRACT

Tripathy, Rohit Ph.D., Purdue University, May 2020. Surrogate Modeling for Uncertainty Quantification in Systems Characterized by Expensive and High-Dimensional Numerical Simulators. Major Professor: Ilias Bilionis, School of Mechanical Engineering.

Physical phenomena in nature are typically represented by complex systems of ordinary differential equations (ODEs) or partial differential equations (PDEs), modeling a wide range of spatio-temporal scales and multi-physics. The field of computational science has achieved indisputable success in advancing our understanding of the natural world - made possible through a combination of increasingly sophisticated mathematical models, numerical techniques and hardware resources. Furthermore, there has been a recent revolution in the data-driven sciences - spurred on by advances in the deep learning/stochastic optimization communities and the democratization of machine learning (ML) software.

With the ubiquity of use of computational models for analysis and prediction of physical systems, there has arisen a need for rigorously characterizing the effects of unknown variables in a system. Unfortunately, Uncertainty quantification (UQ) tasks such as model calibration, uncertainty propagation, and optimization under uncertainty, typically require several thousand evaluations of the underlying physical models. In order to deal with the high cost of the forward model, one typically resorts to the surrogate idea - replacing the true response surface with an approximation that is both accurate as well cheap (computationally speaking). However, state-of-art numerical systems are often characterized by a very large number of stochastic parameters - of the order of hundreds or thousands. The high cost of individual evaluations of the forward model, coupled with the limited real world computational budget one is constrained to work with, means that one is faced with the task of

constructing a surrogate model for a system with high input dimensionality and small dataset sizes. In other words, one faces the *curse of dimensionality*.

In this dissertation, we propose multiple ways of overcoming the *curse of dimensionality* when constructing surrogate models for high-dimensional numerical simulators. The core idea binding all of our proposed approach is simple - we try to discover special structure in the stochastic parameter which captures most of the variance of the output quantity of interest. Our strategies first identify such a low-rank structure, project the high-dimensional input onto it, and then link the projection to the output. If the dimensionality of the low dimensional structure is small enough, learning the map between this reduced input space to the output is a much easier task in comparison to the original surrogate modeling task.

1. INTRODUCTION TO UNCERTAINTY QUANTIFICATION

1.1 Background

Despite the indisputable successes of modern computational science and engineering, the increase in the predictive abilities of physics-based models has not been on a par with the advances in computer hardware. On one hand, we can now solve harder problems faster. On the other hand, however, the more realistic we make our models, the more parameters we have to worry about, in order to be able to describe boundary and initial conditions, material properties, geometric imperfections, constitutive laws, etc. Since it is typically impossible, or impractical, to accurately measure every single parameter of a complex computer code, we have to treat them as uncertain and model them using probability theory. Unfortunately, the field of uncertainty quantification (UQ) [1–4], which seeks to rigorously and objectively assess the impact of these uncertainties on model predictions, is not yet mature enough to deal with high-dimensional stochastic spaces.

The most straightforward UQ approaches are powered by Monte Carlo (MC) sampling [5, 6]. In fact, standard MC, as well as advanced variations, are routinely applied to the uncertainty propagation (UP) problem [7–9], model calibration [10,11], stochastic optimization [12–14], involving complex physical models. Despite the remarkable fact that MC methods convergence rate is independent of the number of stochastic dimensions, realistic problems typically require tens or hundreds of thousands of simulations. As stated by A. O'Hagan, this slow convergence is due to the fact that "Monte Carlo is fundamentally unsound" [15], in the sense that it fails to learn exploitable patterns from the collected data. Thus, MC is rarely ever useful in UQ tasks involving expensive computer codes.

1.2 Sources of uncertainty

Uncertainty within predictive systems (numerical simulators or physical experiments) arises from multiple sources. We broadly categorize the various sources of uncertainty as follows:

1. Experimental sources.
2. Numerical sources.
3. Model-form uncertainty.
4. Parametric uncertainty.

1.2.1 Experimental uncertainty

The advent of computational methods notwithstanding, physical experimentation retains an important within the engineering sciences. Physical experiments serve as the most reliable validation technique for the predictions of computational methods. Furthermore, physical experiments also serve as a source of generating high-fidelity data i.e. they can be treated as a black-box forward model from which an output quantity of interest can be derived. Thus, while using information produced as the outcome of a physical experiment, one must be cognizant of various sources of uncertainty. Uncertainties arise within experimental setup primarily through errors in calibration or geometric imperfections in measuring equipment. For instance, particle image velocimetry (PIV), the field concerned with visualization of velocity fields in fluid flows and development of models for complex flows, exhibit complex nonlinear relationships between input parameters and calibration errors. Rigorous characterization of these uncertainties is an active area of research within PIV [16–18]. Wind tunnel experimentation is often used to validate numerical aerodynamic computation and flow field visualization. Given manufacturing tolerances and calibration errors, quantification of experimental errors is the subject of much research [19].

1.2.2 Numerical uncertainty

Mathematical models of complex physical systems often emerge as systems of analytically intractable partial differential equations (PDEs). Making predictions about the complex system entails developing and implementing numerical methods in order to approximately solve these mathematical models. Computer implementation of approximate solvers are limited in their accuracy by the round-off precision errors [20]. Such errors can be mitigated through the use of higher precision datatypes (such as double precision floating point numbers). However, the use of higher precision numbers entails greater storage requirements and as such can be prohibitive for massive systems. The existence of precision errors, therefore, lead to a trade-off between acceptable accuracy levels from approximate solvers and it's associated storage costs.

Furthermore, approximate numerical solvers of PDEs work by discretizing an infinite-dimensional problem into a finite-dimensional computation which is tractable to a computer. The transformation from infinite to finite dimensions is characterized by grid spacing parameters which induce error in the accuracy of the predictions. While it is known that the accuracy of the solution improves with decreasing grid sizes, the computational cost of the solution grows with decreasing step sizes,leading to a trade-off. The uncertainty in the solution induced by discretization is not known exactly but is bounded as a function of a grid spacing parameter. The finite element (FE) community, for instance, is actively engaged in developing better bounds on the solution accuracy - a line of inquiry referred to as *aposteriori error analysis* [21–24].

1.2.3 Model-form uncertainty

Model-form uncertainty (or functional uncertainty) [25] refers to uncertainty arising out of the selection of specific mechanistic or phenomenological model to explain physical processes. The classic example of model-form uncertainty arises in the context of constitutive laws for problems in continuum mechanics and inter-atomic potentials used in molecular dynamics (MD) simulations. Computational models for

turbulent fluid flow problems require making choices on a 'turbulence model' - a phenomenological model that captures turbulent flow behavior. Inspite of their flaws, Reynolds-averaged Navier Stokes (RANS) models find widespread use as the go-to turbulence model in a variety of design, analysis and prediction tasks in fluid dynamical systems. More sophisticated turbulence models such as large eddy simulations (LES) or direct numerical simulations (DNS) often incur prohibitive costs that exceed available computational budgets. Toward this end, many recent works have proposed methodologies for the quantification of uncertainties induced by the choice of RANS models for turbulent fluid flow [26, 27] Exchange and correlation functionals used in MD simulations are often simplifications of the actual material physics and many researchers have proposed methodologies to quantify uncertainties induced by the selection of simpler constitutive models in MD [28, 29].

1.2.4 Parametric uncertainty

The most common source of uncertainty in engineering and scientific applications is *parametric uncertainty*. Parametric uncertainty is the lack of knowledge about input parameters to a computational or experimental model. In the common scenario of PDE systems model for a physical phenomenon, there are numerous parametric inputs needed for closure of the numerical simulation. Such inputs could include parameters describing material properties or boundary/initial conditions. They could also involve parameters that convey geometric information about the system.

In this thesis, we exclusively consider problems that involve parametric uncertainties. Methods dealing with *experimental*, *numerical* or *model-form* uncertainties are beyond the scope of this work and we refer readers to any of the aforementioned citations for methodologies dealing with these types of uncertainties. Furthermore, it is worth mentioning that it is a common practice to categorize uncertainty in an application problem into *aleatoric* and *epistemic* uncertainties. The precise definition

of each kind is the subject of much debate[1] among mathematicians, statisticians and philosophers. For this work, we define *epistemic uncertainty* to be uncertainty arising out of a lack of knowledge about a deterministic quantity and *aleatoric uncertainty* to be uncertainty arising out of true randomness. All the problems that we consider within this thesis fall strictly into the camp of epistemic uncertainty, i.e., we are interested in inferring strictly deterministic quantities about which we have insufficient information.

1.3 Classification of uncertainty quantification tasks

1.3.1 Forward problem

The forward problem or the *uncertainty propagation* problem [1, 30] is the simplest and most common category of UQ tasks. Roughly speaking, the forward problem entails estimating statistical properties of output quantities of interest from a computational model or experimental setup with stochastic inputs. Formally, given a probability measure, μ, on the space of input quantities $\mathcal{X}$, what is the induced statistics on the space of the outputs $\mathcal{Y}$ of a system idealized as a mathematical function F? Note that this is essentially a question of estimating probability densities under a change of variables defined by the system F.

1.3.2 Inverse problem

The *inverse problem* (also known as the *model calibration* or the *state estimation* or the *parameter identification* problem [1,10,30] is extremely common cutting across scientific disciplines. Stated simply, the *inverse problem* is the task of finding an estimate of the unknown inputs to a model given data on observations of the model outputs. It is easy to see that this is the reverse of the propagation problem from Sec. 1.3.1. More formally, the inverse problem is the task of estimating a parameter

[1] https://www.stat.berkeley.edu/ aldous/Real˙World/ale˙epi.html

$\boldsymbol{\theta}$ which is the input to a model F, from observations $\mathcal{D}$ of output quantities from the model F.

The classical formulation of the inverse problem [10] treats the task of estimating $\boldsymbol{\theta}$ as an optimization problem, constrained by a system of PDEs [31]. Standard methods of numerical convex optimization [32,33] such as Newton's method are applied toward this task. The more general, robust approach towards the model calibration task poses the problem of estimating $\boldsymbol{\theta}$ as a Bayesian inference task, i.e., estimating the posterior probability density $p(\boldsymbol{\theta}|\mathcal{D})$. Note that the classical formulation of generating a point estimate of $\boldsymbol{\theta}$ is a special case of the generic Bayesian approach to model calibration. A point estimate recovered from minimizing the least squares discrepancies between observations and model predictions can be thought of as approximating the posterior over $\boldsymbol{\theta}$ with a Dirac delta function centered over the mode of the posterior.

1.3.3 Sensitivity analysis

Sensitivity analysis is a category of tasks in UQ that seek to answer questions around the degree of variation a function/model/experiment exhibit with respect to any given input variable or a group of input variables. Broadly speaking, sensitivity analysis is classified as either *local* sensitivity analysis - variation of a function around a given location in it's input regime of interest, or *global* sensitivity analysis - the mean variation of the function across the entire domain.

Local sensitivity analysis is often performed by analyzing perturbations of the function under consideration, f, around the point of interest. In the absence of analytical gradient information, finite difference approximations are used to characterize the derivatives (total or partial) of f. Randomized approaches to local sensitivity analysis have also yielded accurate results; see, for instance, [34]. In recent times, the emergence of scalable, easy-to-use automatic differentiation computational libraries has been enabling computation of exact gradients of complex numerical simulations, proving extremely useful for local sensitivity analysis tasks.

1.3.4 Optimization under uncertainty

Many engineering tasks involve optimizing a quantity of interest subject to scientific/economic constraints such as physical laws, monetary/computational cost etc. For instance, a question of fundamental importance to aerospace design engineers is the optimal shape of an airfoil that minimizes drag and maximizes lift. The *optimization under uncertainty* problem can be thus defined - given a quantity of interest q, that needs to minimized, as a function of design parameters $\mathbf{x}$ and stochastic parameters ξ, what is our best estimate of the design parameters that achieve this objective? Mathematically, we seek a solution to the following stochastic optimization task:

$$\mathbf{x}^* = \underset{x}{argmin} \mathbb{E}_{\mathbf{x}} \left[q(\mathbf{x}, \xi) \right], \tag{1.1}$$

where, $\mathbb{E}$ is the expectation operator. While Eqn. (1.1) looks like a standard stochastic optimization task, we are often, in realistic engineering scenarios, constrained by:

1. high cost of querying q due to the need for solving computational intensive PDE systems,

2. Lack of access to information about the gradients of q with respect to the design or stochastic parameters, making classical stochastic optimization routines infeasible.

The high computational cost of evaluating q coupled with the lack of gradient information has led to the emergence of black-box global optimization based on Gaussian processes [35] as the state-of-the-art approach to tackling the task of design optimization under uncertainty.

1.4 The surrogate approach

The simplest approach for tackling uncertainty quantification tasks is the Monte Carlo (MC) method [5,6,36]. The basic idea of MC is that one can compute empirical

estimates of the statistics of some quantity of interest (QoI) by sampling averages. The MC method is guaranteed to converge in the limit of infinite samples. MC methods, and its advanced variants, are routinely applied to UQ tasks such as UP [37], inverse problems [38,39], model calibration [40] and stochastic optimization [41]. The computational time to convergence of MC methods is independent of the number of the stochastic dimensions. However, the number of samples needed by MC methods, to obtain convergent statistics is large, typically being of the order of hundreds of thousands or millions. This makes MC methods unsuitable for UQ tasks involving expensive computer codes.

We typically deal with expensive computer codes, by building a cheap-to-evaluate surrogate of the response surface. To do this, a set of locations in the uncertain parameter-space are carefully selected and the forward model is evaluated at these locations. This produces a set of independent observations of the model response. The total number of such simulations to be performed is determined by one's computational budget and desired accuracy. Because the surrogate model can be queried very cheaply, one can use it as a replacement of the original simulator and perform UQ tasks using MC techniques. Popular choices for surrogate models in the literature include, Gaussian processes [42–47], polynomial chaos expansions [48–52], radial basis functions [53, 54] and relevance vector machines [44, 55].

1.5 Surrogate models and the curse of dimensionality

Despite their success, traditional surrogate modeling methods have become intractable for problems in which the number of input stochastic dimensions is large. In order to construct a surrogate response surface for a multivariate function with a large number of uncertain parameters, one has to overcome the phenomenon known as the *curse of dimensionality*, a term coined by the mathematician Richard Bellman [56] in the context of dynamic programming. In the context of statistical sampling and machine learning the implication of the curse of dimensionality is that to sufficiently

explore a high dimensional space, one must visit an exponentially large number of points in that space. As a concrete example, suppose the task of approximating a surrogate model for a univariate function can be done by visiting 10 locations in the input space and evaluating the forward model at those input locations. For a bivariate function of similar lengthscale, one would need to visit roughly $10 \times 10 = 100$ points in the input space to maintain a similar level of accuracy of the constructed surrogate. Generalizing, a d-variate function requires visiting $\mathcal{O}(10^d)$ locations in the input space and evaluating the forward solver at those locations.

Generic UQ techniques are unable to deal with high stochastic dimensions. This is due to the reliance on the Euclidean distance to define input-space correlations which becomes uninformative as the dimensionality of the input space increases [57]. In other words, blindly attempting to learn generic high-dimensional functions is a futile task. Instead, research efforts are focused on methodologies that can identify and exploit some special structure of the response surface, which can be discovered from data.

1.6 Dimensionality reduction in the context of surrogate modeling

The simplest way to address the curse of dimensionality is to use a variable reduction method, e.g., sensitivity analysis [1, 58] or automatic relevance determination [59–61]. Such methods rank the input features in order of their ability to influence the quantity of interest, and, then, eliminate the ones that are unimportant. Of course, variable reduction methods are effective only when the dimensionality of the input is reasonable (not very high-dimensional) and when the input variables are, more or less, uncorrelated. The common case of functional inputs, e.g., flow through porous media requires the specification of the permeability and the porosity as functions of space, cannot be treated directly with variable reduction methods. In such problems one has to start with a dimensionality reduction of the functional input. For example, if the input uncertainty is described via a Gaussian random

field, dimensionality reduction can be achieved via a truncated Karhunen-Loève expansion (KLE) [62]. If the stochastic input model is to be built from data, one may use principal component analysis (PCA) [63], also known as empirical KLE, or even non-linear dimensionality reduction maps such as kernel PCA [64]. The end goal of dimensionality reduction techniques is the construction of a low dimensional set of uncorrelated features on which variable reduction methods, or alternative methods, may be applied. Note that even though the new features are lower dimensional than the original functional inputs, they are still high-dimensional for the purpose of learning the response surface.

A popular example of an exploitable feature of response surfaces that can be discovered from data is additivity. Additive response surfaces can be expressed as the sum of one-variable terms, two-variable terms, and so on, interpreted as interactions between combinations of input variables. Such representations are inspired from physics, e.g., the Coulomb potential of multiple charges, the Ising model of statistical mechanics. Naturally, this idea has been successfully applied to the problem of learning the energy of materials as a function of the atomic configuration. For example, in [65] the authors use this idea to learn the quantum mechanical energy of binary alloys on a fixed lattice by expressing it as the sum of interactions between clusters of atoms, a response surface with thousands of input variables. The approach has also been widely used by the computational chemistry community, where it is known as high-dimensional model representation (HDMR) [66–69]. The UQ community has been embracing and extending HDMR [70, 71], sometimes referring to it by the name functional analysis of variance (ANOVA) [72, 73]. It is possible to model additive response surfaces with a GP by choosing a suitable covariance function. The first such effort can be traced to [74] and has been recently revisited by [75–79]. By exploiting the additive structure of response surfaces one can potentially deal with a few hundred to a few thousand input dimensions. This is valid, of course, only under the assumption that the response surface does have an additive structure with a sufficiently low number of important terms.

Another example of an exploitable response surface feature is active subspaces (AS) [80]. An AS is a low-dimensional linear manifold of the input space characterized by maximal response variation. It aims at discovering orthogonal directions in the input space over which the response varies maximally, ranking them in terms of importance, and keeping only the most significant ones. Mathematically, an AS is described by an orthogonal matrix that projects the original inputs to this low-dimensional manifold. The classic framework for discovering the AS was laid down by Constantine [81–86]. One builds a positive-definite matrix that depends upon the gradients of the response surface. The most important eigenvectors of this matrix form the aforementioned projection matrix. The dimensionality of the AS is identified by looking for sharp changes in the eigenvalue spectrum, and retaining only the eigenvectors corresponding to the highest eigenvalues. Once the AS is established, one proceeds by: i) Projecting all the inputs to the AS; ii) Learning the map between the projections and the quantity of interest. The latter is known as the *link function*. The framework has been successfully applied to a variety of engineering problems [87–94].

One of the major drawbacks of classic AS methodology is that it relies on gradient information. Even though, in principle, it is possible to compute the gradients either by deriving the adjoint equations [95] or by using automatic differentiation [96], in many cases of interest this is not practical, since implementing any of these two approaches requires a significant amount of time for software development, validation and verification. This is an undesirable scenario when one deals with existing complex computer codes with decades of development history. The natural alternative of employing numerical differentiation is also not practical for high-dimensional input, especially when the underlying computer code is expensive to evaluate and/or when one has to perform the analysis using a restricted computational budget. The second major drawback of the classic AS methodology is its difficulty in dealing with relatively large observational noise, since that would require a unifying probabilistic framework. This drawback significantly limits the applicability of AS to important problems that include noise. For example, it cannot be used in conjunction with

high-dimensional experimental data, or response surfaces that depend on stochastic models e.g., molecular dynamics.

The ideas of AS methodologies are reminiscent of the partial least squares (PSL) [97] regression scheme, albeit it is obvious that the two have been developed independently stemming from different applications. AS applications focus on computer experiments, while PSL has been extensively used to model real experiments with high-dimensional inputs/outputs in the field of chemometrics [98–100]. PSL not only projects the input to a lower dimensional space using an orthogonal projection matrix, but, if required, it can do the same to a high-dimensional output. It connects the reduced input to the reduced output using a linear link function. All model parameters are identified by minimizing the sum of square errors. PSL does not require gradient information and, thus, addresses the first drawback of AS. Furthermore, it also addresses, to a certain extent, the second drawback, namely the inability of AS to cope with observational noise, albeit only if the noise level is known a priori or fitted to the data using cross validation. As all non-Bayesian techniques, PSL may suffer from overfitting and from the inability to produce robust predictive error bars. Another disadvantage of PSL is the assumption that the link map is linear, a fact that severely limits its applicability to the study of realistic computer experiments. The latter has been addressed by the locally weighted PSL [101], but at the expense of introducing an excessive amount of parameters.

1.7 Overview of upcoming chapters

Our goal throughout this thesis is to construct approximations of the form $f(\mathbf{x}) = g(h(\mathbf{x}))$, where the function $g(\cdot)$ acts on a low-dimensional submanifold of the high-dimensional space of the original inputs. The approximation task is, thus, decomposed into:

1. Specifying a suitable form for the function h which projects the high-dimensional stochastic parameters into a suitable low-dimensional manifold, and,

2. Finding the right link function g which maps the projected inputs into the space of the quantity of interest.

Toward this end, in Ch. 2, we focus on the developing a Bayesian surrogate for the map between the uncertain parameters and the quantity of interest, specifically through the framework of GP regression. Our choice of this approach is motivated by the following factors:

1. Given that we are working in the low-data regime, the Bayesian formalism, and GP regression by extension, offers a systematic approach to capture the epistemic uncertainty induced as a consequence of limited data. Indeed, GP regression has been a popular choice of surrogates within the context of limited data uncertainty quantification. See, for instance, [44–47, 47, 102–106].

2. The GP framework is particularly advantageous for 2^{nd} order tasks such as optimal experimental design. Since the posterior predictive distribution in the standard regression setting with Gaussian likelihood is known in closed form, it simplifies the task of quantifying the expected information gain from future simulations. See, for instance, [44,59,107–109] for applications of GP regression to tasks involving active learning and multi-objective global optimization within the context of problems characterized by uncertainty.

3. In the framework of GP regression, we have the ability to encode prior knowledge within the mean function and covariance kernel of the GP surrogate.

We seek an approximation where the projection function h is an orthogonal linear transformation, i.e., $h(\mathbf{x}) = \mathbf{W}^T\mathbf{x}$, $\mathbf{W}^T\mathbf{W} = I_d$. This form of intrinsic structure, known as the *active subspace* (AS) has recently emerged as a popular approach to dimensionality reduction for surrogate modeling [81]. In our approach, we develop a probabilistic approach for recovering the AS. The classical approach to recovering the AS of any given quantity of interest requires evaluations of it's gradient with respect to the stochastic parameters - a major drawback which limits it's applicability

to arbitrary computational programs. Our approach poses the task in a gradient-free manner. Specifically, the projection matrix of the AS is posed a hyperparameter within the covariance kernel of the GP surrogate. This approach seamlessly integrates the dimensionality reduction from the AS projection with the Bayesian GP surrogate. The orthogonal projection matrix is then learnt by the usual procedure of maximizing the marginal likelihood of the data. Toward this end, we develop a two-stage iterative approach for maximizing the likelihood. Our approach alternates between performing optimization over the Stiefel manifold with respect to the projection matrix and the optimization unconstrained by orthogonality requirements with respect to the remaining hyperparameters. This procedure is carried out until a suitable convergence criterion is met. Furthermore, we develop a rigorous for model selection, i.e., answering the question - what is the right dimensionality of the active subspace for the function under investigation? Our approach for model selection relies on a cheap approximation of the model evidence using the Bayesian information criterion [110]. We show, empirically, that our proposed approach finds the right AS dimensionality. Additionally we show results for a suite of experiments that test the performance of the proposed framework. We test the robustness of our approach to observation noise as well as dataset size. Our methodology is tested on a benchmark problem of an elliptic partial differential equation with stochastic spatially varying conductivity coefficient. Following traditional approaches, the uncertainty is modeled through a spectral expansion in a basis of standard normal variables, i.e., we resort to the KL expansion to obtain a finite dimensional representation of the infinite dimensional random field input. Our active subspace GP regression framework is then used to construct accurate surrogate maps between the coefficients of the truncated KL expansion and a scalar quantity of interest which is a functional of the PDE solution. Finally, we use our approach to study the effects of uncertainties in the propagation of *solitons* in a granular crystals system. We wrap up Ch. 1 with comments.

In Ch. 3, we propose a systematic approach for constructing surrogate models using *deep neural networks* (DNNs) [111–114]. Neural networks (NNs) (or artificial

neural networks (ANNs)) are a class of function approximators that have shot to prominence in recent years because of breakthrough successes achieved in numerous artificial intelligence (AI) tasks such as image classification [115–119] and autonomous driving [120, 121]. The idea of DNNs is not new. The reason for their increased usage and popularity in recent times is due to: 1. Advancements in computer hardware leading to widespread availability of graphics processing units (GPUs) for accelerated computation; 2. Advances in stochastic optimization including techniques such as *Adam* [122], *RMSprop* [123], *AdaGrad* [124], *AdaDelta* [125] etc.; 3. Regularization techniques such as *dropout* [126]; and, 4. Development of easy-to-use software libraries, such as *Tensorflow* [127], *PyTorch* [128] and *Theano* [129].

The basic idea of DNNs is that one can represent multivariate functions through a hierarchy of features of increasing complexity. The most typical example of a DNN is a feedforward multilayer perceptron (MLP). A highly attractive property of MLPs is that, under mild assumptions on the underlying function being approximated, they are universal approximators [130]. This means that any continuous function, regardless of its complexity, can be approximated with a neural network of just one layer with a sufficient number of hidden units. DNNs tackle the curse of dimensionality through a series of nonlinear projections of the input into exploitable latent spaces. In fact, PCA can be thought of as a special case of a DNN with no hidden layers such that the latent space is recovered through an orthogonal projection of the input.

The powerful nonlinear function approximation capabilities coupled with the scalability of DNNs to high dimensions offers a very promising direction of research for the UQ community, with the potential to significantly improve upon state-of-the-art capabilities. [131] use a 3 layer convolutional DNN to learn a map between input coefficients of SPDEs to a functional of the PDE solution. [132] use a Bayesian fully convolutional encoder-decoder network to solve an image-to-image regression task mapping the random input coefficient field of an elliptic SPDE to the corresponding solution. These papers offer encouraging results for challenging problems in UQ. However, they are only applicable to tasks where input parameters and output quan-

tities can be treated as images and they require a lot of cross validation to learn an optimal network structure. Specifically in the context of SPDEs, we are interested in learning a surrogate that can make predictions at spatial locations other than those included in the discretization of the underlying deterministic numerical solver.

The task of selecting the architecture of the network and values for hyperparameters such as regularization constants is a persistent problem in the application of DNNs. Under constraints of limited data, this task assumes added importance. In this work, we present a methodology based on MLPs where we parameterize our network in a way that lends it the interpretation of discovering a low dimensional nonlinear manifold that captures maximal variation of the model outputs. We think of this procedure as discovering a nonlinear active subspace. The *projection function*, which connects the high dimensional input, to the low dimensional manifold, is linked to the scalar model output through a linear transformation. We utilize a combination of Bayesian global optimization (BGO) [35] and grid search to select the best setting of the network hyperparameters and determine the appropriate structure.

In Sec. 3.1, we discuss the task of surrogate modeling. In Sec. 3.2 through Sec. 3.5, we discuss the process of constructing a DNN surrogate model, including the parameterization of the network architecture and the optimization of the network parameters. We conclude with a description of the procedure we use to select network hyperparameters in Sec. 3.7. In Sec. 3.8, we demonstrate our methodology on a SPDE with uncertain diffusion coefficient. A novelty of our work is that we do not make any assumption on the regularity and lengthscales of the uncertain diffusion. Specifically, we construct a surrogate of the SPDE solver which can accurately predict the response when tested with input random fields that may not be structurally similar (in terms of smoothness and lengthscales) to samples of the input in the training dataset. This deviates from the standard formulation of this problem in the UQ literature, where a specific covariance structure is imposed on the uncertain parameter. As a result, our problem is not amenable to the application of preliminary dimensionality reduction using the KLE, thereby making it far more challenging than

the traditional formulation of the problem. We wrap up Ch. 3 with concluding remarks in Sec. 3.10.

In Ch. 4, we discuss a methodology for recovering classical *linear* active subspaces within deep neural networks. We pose the active subspace projection matrix as the reparameterization of an unconstrained tall-and-skinny matrix. This makes the DNN amenable to standard stochastic gradient descent routines for training, while strictly enforcing the orthonormality of the columns of the active subspace projection matrix. The effectiveness of the approach is demonstrated on a suite of synthetic examples as well as a high-dimensional stochastic partial differential equation problem.

We conclude this thesis in Ch. 5 with a discussion on:

1. A review of the methodologies developed as part of this dissertation, as well as our key findings and lessons learned.

2. A discussion on some future directions of research on the subject of data-driven uncertainty quantification, with an emphasis on the recently popularized approach of integrating physics with machine learning.

2. LEARNING LOW-RANK STRUCTURE WITH GAUSSIAN PROCESS REGRESSION

[1] Suppose we have a computer code simulating a physical phenomenon. Mathematically, we represent this simulator as a function $f : \mathcal{X} \to \mathcal{Y}$. f accepts a vector of inputs $\boldsymbol{\xi} \in \mathcal{X} \subseteq \mathbb{R}^D$ where $\boldsymbol{\xi}$ could specify material properties, external loads, boundary conditions, initial conditions, etc. The output of the computer code is some scalar quantity of interest $y = f(\boldsymbol{\xi}) \in \mathcal{Y} \subseteq \mathbb{R}$. Typically, f depends on the solution of some PDE which depends on $\boldsymbol{\xi}$. Furthermore, f is unknown in closed form and information about it can only be obtained by querying the simulator at feasible values of $\boldsymbol{\xi}$. Finally, the dimensionality, D, of the input vector $\boldsymbol{\xi}$ is large, potentially of the order of hundreds or thousands. Given a finite number of evaluations of the simulator, the task of constructing a surrogate function, $\hat{f}$, for the true response surface f becomes computationally infeasible without resorting to dimensionality reduction.

Suppose the inputs $\boldsymbol{\xi}$ to the function f are not known exactly (a common scenario in numerous engineering tasks). We formalize our beliefs about $\boldsymbol{\xi}$ using a suitable probability distribution:

$$\boldsymbol{\xi} \sim p(\boldsymbol{\xi}). \tag{2.1}$$

Given our beliefs about $\boldsymbol{\xi}$, we wish to characterize the statistical properties of the output $f(\boldsymbol{\xi})$ such as the mean:

$$\mu_f = \int f(\boldsymbol{\xi}) p(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}, \tag{2.2}$$

[1] The contents of this chapter are reproduced, with permission, from the paper entitled "Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation" [43] published in the Journal of Computational Physics (2016).

the variance,

$$\sigma_f^2 = \int (f(\boldsymbol{\xi}) - \mu_f)^2 p(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}, \tag{2.3}$$

and the probability density,

$$p_f(y) = \int \delta(y - f(\boldsymbol{\xi})) p(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}. \tag{2.4}$$

This is formally known as the uncertainty propagation problem (UP).

The UP problem is particularly hard when obtaining information about $f(\cdot)$ is expensive. In such cases, we are necessarily restricted to a limited set of observations. Specifically, assume that we have queried the information source at N input points,

$$\mathbf{X} = \left\{ \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)} \right\}, \tag{2.5}$$

and that we have measured

$$\mathbf{y} = \left\{ y^{(1)}, \ldots, y^{(N)} \right\}. \tag{2.6}$$

We frame the UP problem as follows - what is our best estimate of the statistics of the QoI, conditional on observing a limited dataset $\mathcal{D}$? Our approach relies on reconstructing the map between the stochastic input parameters and the output quantities of interest from $\mathcal{D}$, by exploiting low-rank latent structure in the space of the uncertain parameters.

Given that the stochastic parameter space we are working in is high-dimensional, i.e., $D \gg 1$, naive methods for reconstructing the map $f(\cdot)$ from data will fail due to the *curse of dimensionality*. The standard approach to circumvent this problem is to look for *intrinsic* exploitable structure within $f(\cdot)$. *Ridge functions* [133] are a special class of multivariate functions which possess the following structure:

$$f(\mathbf{x}) \approx g(\mathbf{W}^T \mathbf{x}), \tag{2.7}$$

where, $\mathbf{W} \in \mathbb{R}^{D\times d}$ is a tall-and-skinny matrix. Intuitively, such the columns of the matrix $\mathbf{W}$ map the high-dimensional inputs to a low-dimensional manifold on which the function variability is captured. A special case of *ridge functions* are obtained by imposing the constraint that the matrix $\mathbf{W}$ is made up of orthonormal columns, i.e., $\mathbf{W}^T\mathbf{W} = I_d$, where, I_d is the identity matrix in $\mathbb{R}^d$. The vector space of projected inputs, $\mathbf{z} = \mathbf{W}^T\mathbf{x} \in \mathbb{R}^d$ is referred to as the active subspace and it represents the optimal linear low-dimensional embedding of a high-dimensional multivariate function. To summarize, we are looking for a function $f(\mathbf{x}) \approx \hat{f}(\mathbf{x}) = g(\mathbf{z}) = g(\mathbf{W}^T\mathbf{x})$, with the constraint $\mathbf{W} \in V_d(\mathbb{R}^D)$, where,

$$V_d(\mathbb{R}^D) := \left\{\mathbf{A} \in \mathbb{R}^{D\times d} : \mathbf{A}^T\mathbf{A} = \mathbf{I}_d\right\}, \tag{2.8}$$

is known as the *Stiefel manifold* of tall-and-skinny matrices with orthonormal columns. $g : \mathbb{R}^d \to \mathbb{R}$ is a potentially nonlinear function acting on d-dimensional space and we refer to it as the *link* function. Note that the structure of the approximation is unique upto arbitrary rotations and relabeling of the *active subspace* coordinate system. The working hypothesis in the theory of approximation of high input dimensional functions is that the approximation is easier to construct if the size of the embedding d is sufficiently smaller than the original vector space D, i.e., $d \ll D$.

The classical formulation of the active subspace requires evaluation of the gradients of the response $f(\cdot)$. Given that this is a highly restrictive assumption, our goal is to construct the active subspace approximation without leveraging gradient information. In this chapter we discuss a complete framework for:

1. Constructing the active subspace ridge function approximation using observations of the function input and output pairs (but not it's gradients), and,

2. Selecting the right dimensionality of the active subspace.

With an accurate construction of the approximation $\hat{f}$, one may utilize for performing any uncertainty quantification task via Monte Carlo methods.

2.1 Gaussian Processes for data-driven modeling

In this section we provide a brief summary of Gaussian process regression (GPR). We begin with a description of Gaussian processes (GP) as probability measures over function spaces (i.e. stochastic processes) and then discuss the application of GPs to the task of learn an unknown map between a set of inputs and outputs, potentially contaminated with noise.

2.1.1 Gaussian processes

Without loss of generality, a Gaussian process (GP) may be defined as a stochastic process $\{X_t\}_{t\geq 0}$ such that any finite collection of random variables, $\{X_{t_1}, X_{t_2}, \cdots, X_{t_k}\}, \forall k \in \mathbb{N}$, are joint distributed as a multivariate Gaussian. A GP is fully specified by it's mean, $m(\cdot)$ and covariance function, $k(\cdot,\cdot)$ i.e.:

$$\mathbb{E}[X_t] = m(t), \tag{2.9}$$

$$cov(X_t, X_{t'}) = \mathbb{E}[(X_t - \mathbb{E}[X_t])(X_{t'} - \mathbb{E}[X_{t'}])] = k(t,t'). \tag{2.10}$$

If the covariance of the GP at two locations, X_t and $X_{t'}$ depends only on the distance between the two locations, i.e., $k(t,t') = \tilde{k}(|t-t'|)$ then the corresponding GP is said to be *stationary*. Processes that do not satisfy this criterion are said to be *non-stationary*.

Let X_t be a GP indexed with $t \geq 0$, and specified by a mean function $m(t)$ and a covariance function $k(t,t')$. Let the process be observed at locations $\mathbf{X} = (X_{t_1}, X_{t_2}, \cdots, X_{t_T})$. Denote the set of indices of the observed locations as $\mathbf{T} = (1, 2, \cdots, T)$. The distribution of the vector $\mathbf{X}$ is given by $\mathbf{X} \sim \mathcal{N}(\mathbf{m}, \mathbf{K})$, where,

$$\mathbf{m} = \left(m(t_1), m(t_2), \cdots, m(t_T)\right), \tag{2.11}$$

$$\mathbf{K} = \left[k(t_i, t_j)\right]_{i,j\leq T}. \tag{2.12}$$

Consider any new location $t^* \in [0, \infty)$. Per the property of a GP, X_{t^*} and $\mathbf{X}$ are jointly distributed as:

$$\begin{pmatrix} X_{t^*} \\ \mathbf{X} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} X_{t^*} \\ \mathbf{X} \end{pmatrix} \Big| \begin{pmatrix} m(t^*) \\ \mathbf{m} \end{pmatrix}, \begin{pmatrix} k(t^*, t^*) & k(t^*, \mathbf{T}) \\ k(\mathbf{T}, t^*) & \mathbf{K} \end{pmatrix} \right), \tag{2.13}$$

where, $k(t^*, \mathbf{T})$ is the matrix of cross-covariances between the locations $X(t^*)$ and $\mathbf{X}$. Since the joint distribution of $\left(X^{t^*}, \mathbf{X}\right)$ is Gaussian and the marginals of X_{t^*} and $\mathbf{X}$ are also Gaussian, the conditional distributions of X_{t^*} is also Gaussian and given by:

$$X_{t^*}|\mathbf{X} \sim \mathcal{N}\left(X_{t^*}|m(t^*) + k(\mathbf{T}, t^*)^T \mathbf{K}^{-1}(\mathbf{X} - \mathbf{m}), k(t^*, t^*) - k(\mathbf{T}, t^*)^T \mathbf{K}^{-1} k(\mathbf{T}, t^*)\right). \tag{2.14}$$

The covariance function of a GP establishes the correlation in the GP at any two locations t and t'. A valid covariance function is:

1. symmetric, i.e., $k(t, t') = k(t', t)$,
2. positive-definite, i.e., for any real valued function, $c(t)$,

$$\iint c(t)k(t, t')c(t')\mathrm{d}t\mathrm{d}t' > 0. \tag{2.15}$$

By *Mercer's Theorem* [42], there exists, for every continuous, symmetric positive-definite kernel, an orthogonal basis, $\{\phi_i(t)\}_{i \in \mathbb{N}}$ consisting of the eigenfunctions of the linear operator $\mathcal{T}_K \varphi(t) = \int k(t, t')\varphi(t')\mathrm{d}t'$ such that:

$$k(t, t') = \sum_{i=1}^{\infty} \lambda_i \phi_i(t)\phi_i(t'), \tag{2.16}$$

where, $\lambda_i > 0$ is the eigenvalue associated with the eigenfunction ϕ_i. The sequence $\lambda_j, j = 1, 2, \ldots, \infty$, when arranged in descending order, converges to 0 and thus the sum expressed in Eqn. (2.16) can be truncated at a finite number of terms M, i.e., $k(t, t') \approx \sum_{i=1}^{M} \lambda_i \phi_i(t)\phi_i(t')$. Notice that the two-point covariance may also be expressed as $k(t, t') = \sum_{i=1}^{\infty} \tilde{\phi}_i(t)\tilde{\phi}_i(t')$ where $\tilde{\phi}_i(t) = \sqrt{\lambda_i}\phi_i(t)$. In other words, the

covariance between two locations in a GP is an inner product of an infinite dimensional basis function vector $\boldsymbol{\phi}(s) = (\phi_1(s), \phi_2(s), \dots)$, i.e., $k(t, t') = \langle \boldsymbol{\phi}(t), \boldsymbol{\phi}(t') \rangle$.

2.1.2 Gaussian process surrogate model

We discuss, in the section, how properties of Gaussian processes can be leveraged to learn arbitrary nonlinear functions from a dataset of observations. Let us denote the unknown function as $f(\cdot)$. Without loss of generality, f takes in a vector of inputs $\mathbf{x} \in \mathbb{R}^d$ and produces a scalar output, $y \in \mathbb{R}$. While we do not know the underlying structure of f, we have access to a dataset of observations of the outputs,

$$\mathbf{y} = \{y_1, y_2, \dots, y_N\}, \tag{2.17}$$

corresponding to a set of input observations,

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}. \tag{2.18}$$

GP regression proceeds as follows. As discussed in the previous section, a GP places a prior over a suitable space of functions. This prior captures our beliefs about the unknown function apriori through a suitable choice of the mean and the covariance function. Any finite sampling of the input space produces a joint distribution of random variables that is a multivariate normal whose mean vector and covariance function are governed by the choice of prior mean function and covariance kernel. By conditioning the prior information uses Bayes rule to combine these prior beliefs with observations. The result of this conditioning process is a *posterior* GP which is simultaneously compatible with our prior beliefs and measurements. Because the GP regression model is a Bayesian surrogate, it has the added benefit of providing estimates of uncertainty around the predicted quantities of interest. Furthermore, the predictive uncertainty can be easily segregated into the individual contributions

of *epistemic uncertainty* - uncertainty induced by limited measurements, and observational noise - uncertainty induced by noise in the measured data.

2.1.3 Statistical model

We now lay out the probabilistic model for modeling data $\mathcal{D}$. We begin by placing a Gaussian process prior on the unknown function f:

$$f(\mathbf{x}) \sim \mathcal{GP}(f(\mathbf{x})|m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{2.19}$$

Our prior beliefs about the properties of the unknown function f are captured in the specified mean and covariance functions $m(\cdot)$ and $k(\cdot, \cdot)$. For simplicity, we set the mean function $m(\mathbf{x}) = 0$. Let the covariance kernel $k(\mathbf{x}, \mathbf{x}')$ be specified by a vector of hyperparameters $\boldsymbol{\theta}$. As previously discussed, a valid covariance function is defined by a symmetric, positive-definite covariance operator and can be expressed as the inner product of a basis function vector applied to it's arguments x and x'. Covariance functions measure degrees of similarity between input locations. The most commonly used covariance function is the squared exponential (or radial basis) covariance function defined as follows:

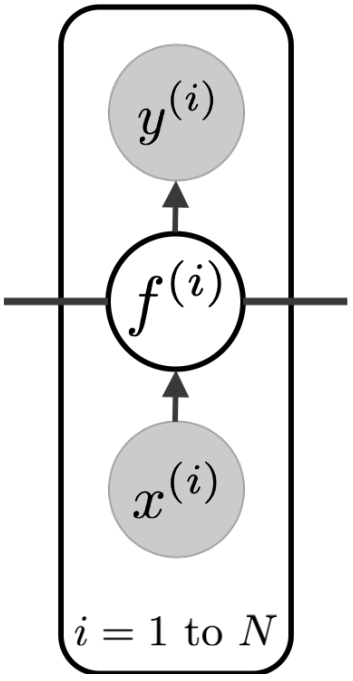


Figure 2.1. Probabilistic graphical model for GP regression.

$$k_{\text{SE}}(\mathbf{x}, \mathbf{x}') = s^2 \exp\left\{-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_i - x_i^{'})^2}{\ell_i^2}\right\}. \tag{2.20}$$

In this work, we will deal exclusively with the case where the output y is a real-valued scalar. As such, we pose the following model for the likelihood of the observed data:

$$y(\mathbf{x}) = f(\mathbf{x}) + \epsilon, \tag{2.21}$$

where, ϵ is additive measurement noise in the output data. Following conventional modeling choices, we endow ϵ with a 0-mean normal distribution with unknown noise variance, i.e, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. The Eqn. (2.21) captures our beliefs about the measurement process - the observation y is a noise corrupted version of the latent function f which is modeled via a Gaussian process prior. Let us denote the latent function values at input locations as $\mathbf{f} = (f_1, f_2, \ldots, f_N)$, where, $f_i = f(\mathbf{x}_i)$. Coupling this with the likelihood model in Eqn. (2.21), we get the following joint distribution over the observed data:

$$\mathbf{y} = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma_n^2 \mathbf{I}_N), \tag{2.22}$$

where, $\mathbf{I}_N$ is the $N \times N$ identity matrix.

The goal of inference is to make predictions about the latent function f at a new unseen input locations. Denote by $\mathbf{X}^* \in \mathbb{R}^{M \times D}$ a set of M new input locations and the corresponding latent function values as $\mathbf{f}^* \in \mathbb{R}^M$. By definition the latent function values at the training locations $\mathbf{X}$ and the test locations are distributed jointly as a multivariate Gaussian distribution whose mean vector and covariance matrix are fully specified by our choice of the mean function and covariance kernel for the GP prior. Conditioning the test latent function values $\mathbf{f}^*$ on the observed data, $\mathcal{D}$ induces the following posterior predictive distribution on the test outputs:

$$\mathbf{f}^* = \mathcal{N}\left(\mathbf{f}^* | \mathbf{K}_{MN}\left[\mathbf{K} + \sigma_n^2 \mathbf{I}_N\right]^{-1} \mathbf{y}, \mathbf{K}_{MM} - \mathbf{K}_{MN}\left[\mathbf{K} + \sigma_n^2 \mathbf{I}_N\right]^{-1} \mathbf{K}_{MN}^T\right), \tag{2.23}$$

where, $\mathbf{K}_{MN}$ are the cross-covariances between the training and test outputs and $\mathbf{K}_{MM}$ is the covariance matrix of the marginal distribution of the test output locations, i.e., $\mathbf{K}_{MN,ij} = k(\mathbf{x}_i, \mathbf{X}^*_{j,:})$ and $\mathbf{K}_{MM,ij} = k(\mathbf{X}^*_{i,:}, \mathbf{X}^*_{j,:})$.

2.1.4 Inference in Gaussian process regression

The task of hyperparameter inference in GP regression is, ideally, framed as a Bayesian inference problem. Denote, collectively, all the hyperparameters in the GP regression mean and covariance functions as $\boldsymbol{\theta}$. Let our prior beliefs about $\boldsymbol{\theta}$ be formalized into a probability distribution:

$$\boldsymbol{\theta} \sim p(\boldsymbol{\theta}). \tag{2.24}$$

The likelihood of the observed data, conditional on the hyperparameters is given by:

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{\mathbf{X}} + \sigma_n^2 I), \tag{2.25}$$

where, the distribution in Eqn. (2.26) is the *marginal likelihood* of the observed data, obtained by integrating out the latent function $\mathbf{f}$ from the likelihood of the measurement process, i.e.,

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y}, \mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) \mathrm{d}\mathbf{f}. \tag{2.26}$$

The likelihood noise, σ_n is also a hyperparameter and we ascribe to it, it's own prior $p(\sigma_n)$. In the regression setting (with Gaussian likelihood models), this marginalization can be carried out in closed form. For alternative likelihoods, a numerical approximation of the marginal likelihood needs to be derived. The posterior state of knowledge over the hyperparameters are then given by:

$$p(\boldsymbol{\theta}, \sigma_n) \propto p(\boldsymbol{\theta}) p(\sigma_n) p(\mathbf{y}|\mathbf{X}). \tag{2.27}$$

This joint distribution over $(\boldsymbol{\theta}, \sigma_n)$ is intractable analytically, and ideally, one must resort to approximate inference techniques such as Markov Chain Monte Carlo (MCMC) [134–137] or Variational Inference (VI) [138–141].

A simple and computationally efficient approach to characterizing the posterior $p(\boldsymbol{\theta}, \sigma_n|\mathcal{D})$ is approximate the full posterior distribution with a delta function centered at the mode of the posterior. If the posterior distribution is sharply peaked, such an approximation is reasonably good. Assuming a flat prior on the hyperparameters, i.e., $p(\boldsymbol{\theta}, \sigma_n) \propto 1$, our inference task reduces to the problem of finding a suitable local maxima of the marginal likelihood. For computational reasons, this task is framed as the minimization of the negative log marginal likelihood, i.e.,

$$\boldsymbol{\theta}^*, \sigma_n^* = \underset{\boldsymbol{\theta}, \sigma_n}{\operatorname{argmin}} - \log p(\mathbf{y}|\mathbf{X}). \tag{2.28}$$

The optimization task in Eqn. 2.28 can be efficiently performed using any second order optimization method. Unless otherwise stated, we adopt the L-BFGS [142] method to find local minima of the negative log marginal likelihood. Given that the objective function is highly non-convex, it is entirely possible to be trapped in bad local minima. To counter this possibility, our approach will be to be perform several repetitions of the optimization routine with different initial starting points in order to find the best possible *maximum likelihood estimate* (MLE).

2.2 Classical active subspace recovery and response approximation

Recall that our goal is to construct an approximation of the form $f(\mathbf{x}) \approx \hat{f}(\mathbf{x}) = g(\mathbf{W}^T\mathbf{x})$, where, $\mathbf{W}$ is a tall-and-skinny matrix of orthonormal columns. This approach was first formulated in [80] and extended and improved in [81]. The classical approach to recovering the manifold spanned by the columns of $\mathbf{W}$, the *active subspace* requires information about the gradients of the quantity of interest [81–84, 88, 89, 91]. Approximation, in the classical setting, follows two steps:

1. Get the matrix $\mathbf{W}$ from the spectral decomposition of an empirical centered covariance matrix of the gradient samples,

2. Construct the approximation $g : \mathbb{R}^d \to \mathbb{R}$, by fitting the projections of the inputs $\mathbf{X}$ onto the active subspace, to the measured outputs, $\mathbf{y}$.

This approach cannot deal with noisy measurements. For the sake of generality, we will work with the assumption that our measurements, $\mathbf{y}$ are noisy estimates of the true underlying function values, $\mathbf{f}$. The Bayesian formalism within which we are working, allows for a rigorous framework for dealing such observational noise.

Let the gradient of the quantity of interest be denoted as $\mathbf{g}$, and we collect a matrix of gradient samples at a finite number of input locations:

$$\mathbf{G} = \left\{\mathbf{g}^{(1)}, \ldots, \mathbf{g}^{(N)}\right\}, \tag{2.29}$$

where

$$\mathbf{g}^{(i)} = \nabla f\left(\mathbf{x}^{(i)}\right) = \left(\frac{\partial f(\cdot)}{\partial x_1}, \ldots, \frac{\partial f(\cdot)}{\partial x_D}\right)^T \in \mathbb{R}^D. \tag{2.30}$$

Let the input space be equipped with a probability measure ρ. Note that this may be different from the description of the input uncertainty $p(\mathbf{x})$. We define a $D \times D$ as follows:

$$\mathbf{C} := \int \mathbf{g}(\mathbf{x})\mathbf{g}(\mathbf{x})^T \rho(\mathbf{x}) \mathrm{d}\mathbf{x}, \tag{2.31}$$

where, $\mathbf{g}(\mathbf{x})$ is the gradient evaluation of the function at location $\mathbf{x}$. $\mathbf{C}$ is a real symmetric positive definite matrix and looks like the centered covariance of the gradient $\mathbf{g}$. This matrix admits a spectral decomposition of the form:

$$\mathbf{C} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \tag{2.32}$$

where,

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_D \end{bmatrix} \in \mathbb{R}^{D \times D}, \tag{2.33}$$

is a diagonal matrix of the eigenvalues of $\mathbf{C}$, arranged in decreasing order,i.e., $\lambda_i > \lambda_j \forall i < j$, and,

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_D \end{bmatrix} \in \mathbb{R}^{D \times D}, \tag{2.34}$$

is a matrix of eigenvectors of $\mathbf{C}$, such that $\mathbf{v}_i$ is the eigenvector corresponding to eigenvalue λ_i. Since $\mathbf{C}$ is a real symmetric matrix, the columns of the eigenvector matrix $\mathbf{V}$ form an orthonormal basis for the vector space $\mathbb{R}^D$. The classical approach relies on partitioning the matrices $\mathbf{\Lambda}$ and $\mathbf{V}$ as follows:

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_2 \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_2 \end{bmatrix}, \tag{2.35}$$

where $\mathbf{\Lambda}_1 = \text{diag}(\lambda_1, \cdots, \lambda_d) \in \mathbb{R}^{d \times d}$ are the d largest eigenvalues and $\mathbf{V}_1 = [\mathbf{v}_1\, \mathbf{v}_2 \cdots \mathbf{v}_d] \in \mathbb{R}^{D \times d}$ are their corresponding eigenvectors. The projection matrix is then set as:

$$\mathbf{W} = \mathbf{V}_1^T, \tag{2.36}$$

Since the matrix integral in Eqn. (2.31) cannot be carried out analytically, we resort to a MC approximation:

$$\hat{\mathbf{C}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{g}_i \mathbf{g}_i^T. \tag{2.37}$$

[81] quantifies the quality of the resulting approximation as follows. The i^{th} eigenvalue quantifies the total variation of the response f captured along the i^{th} eigenvector, i.e.,

$$\lambda_i = \int (\mathbf{v}_i^T \mathbf{g})^T \gamma(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{2.38}$$

Denote the projection along the first d eigenvectors $\mathbf{V}_1$ as $\mathbf{z} = \mathbf{V}_1^T \mathbf{x}$ and the projection along the last $D-d$ eigenvectors, $\mathbf{V}_2$ as $\tilde{\mathbf{z}} = \mathbf{V}_2^T \mathbf{x}$. The total variation of the response f captured in the subspace defined by $\mathbf{V}_1$ is then given by:

$$\int \nabla_{\mathbf{z}} f(\mathbf{x})^T \nabla_{\mathbf{z}} f(\mathbf{x}) \gamma(\mathbf{z}) \mathrm{d}\mathbf{z} = \sum_{i=1}^{d} \lambda_i, \tag{2.39}$$

and the total variation of the function captured in the subspace defined by $\mathbf{V}_2$ is given by:

$$\int \nabla_{\tilde{\mathbf{z}}} f(\mathbf{x})^T \nabla_{\tilde{\mathbf{z}}} f(\mathbf{x}) \gamma(\tilde{\mathbf{z}}) \mathrm{d}\tilde{\mathbf{z}} = \sum_{i=d+1}^{D} \lambda_i, \tag{2.40}$$

If the sum on the right hand side of Eqn. (2.39) is sufficiently small, the approximation $\hat{f}(\mathbf{x}) = g(\mathbf{W}^T\mathbf{x})$ is a sufficiently close to the true response f and may be deployed as a surrogate response. For further theoretical properties of the active subspace see [81].

The link function $g(\cdot)$ that maps the active subspace projections $\mathbf{z} = \mathbf{W}^T\mathbf{x}$ can be a parametric model such as a generalized linear model (GLM) or a deep neural network (DNN), or a non-parametric model such as GP regression.

2.3 Embedding active subspaces in Gaussian process models

The classic approach to active subspace based GP regression suffers primarily from two drawbacks - 1. It requires samples of the gradient of the quantity of interest and 2. It cannot seamlessly integrate measurement noise into the model. Toward this end, we develop a probabilistic approach to active subspace recovery that can overcome these limitations.

To recover the active subspace without gradient information, we propose a novel covariance function acting on the high-dimensional stochastic parameter space:

$$k_{\mathrm{AS}} : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}, \tag{2.41}$$

such that:

$$k_{\mathrm{AS}}(\mathbf{x}, \mathbf{x}') = k_0(\mathbf{z}, \mathbf{z}') = k_0(\mathbf{W}^T\mathbf{x}, \mathbf{W}^T\mathbf{x}'), \tag{2.42}$$

where k_0 is any standard kernel function acting on the low-dimensional space of projected inputs, $\mathbf{z} = \mathbf{W}^T\mathbf{x} \in \mathbb{R}^d$. The covariance function k_0 is characterized by it's own set of hyperparameters, $\boldsymbol{\phi}$. To summarize, the covariance function k_{AS} acting on the original high-dimensional input space, operates as follows - first the original

high-dimensional input pairs, $\mathbf{x}$ and $\mathbf{x}'$ are projected to the active subspace defined by the tall-and-skinny matrix $\mathbf{W}$, i.e. $\mathbf{z}$ and $\mathbf{z}'$ are computed. Then the covariance between the GP function values at locations $\mathbf{x}$ and $\mathbf{x}'$ is estimated as the covariance function output obtained by operating on the pair of low-dimensional embeddings, $\mathbf{z}$ and $\mathbf{z}'$. The covariance function k_{AS} is thus characterized by the hyperparameters of the base kernel k_0, $\boldsymbol{\phi}$ as well as the orthonormal projection matrix $\mathbf{W}$.

One may think of the GP regression model for the surrogate linking the high-dimensional inputs, $\mathbf{x}$ and the output y as just a standard GP regression model, albeit with an additional parameter, $\mathbf{W}$. Put another way, defining a GP regression with the covariance function of the form k_{AS} places a prior distribution over the Hilbert space of functions that assume the form - $f(\mathbf{x}) = g(\mathbf{W}^T\mathbf{x})$. The main challenge in the model, as it is has been setup, is how does one go about inference (fully Bayesian or maximum likelihood) while respecting the orthogonality constraints of the matrix $\mathbf{W}$, i.e., ensuring the search for a suitable $\mathbf{W}$ (or distribution over $\mathbf{W}$s) occurs over the right Stiefel manifold.

The challenge of conducting inference in the GP regression model with the covariance function k_{AS} as laid out in the previous paragraph is non-trivial and to deal with this, we will discuss our proposed two-step likelihood maximization approach. This approach is guaranteed to converge to a local maxima and with multiple restarts of the algorithms, can help produce 'good' local minima estimates.

2.3.1 Two-stage iterative negative marginal likelihood minimization

The negative log marginal likelihood minimization problem over the covariance function hyperparameters $\boldsymbol{\theta}$ and the noise variance σ_n^2, is constrained by the fact that the projection matrix $\mathbf{W}$ must satisfy $\mathbf{W} \in V_d\left(\mathbb{R}^D\right)$, i.e., it is a tall-and-skinny matrix of orthonormal columns. Our approach to dealing with this constrained optimization task is to devise an iterative two-stage method where we alternate the optimization over the projection matrix $\mathbf{W}$ and the remaining hyperparameters, $\boldsymbol{\phi}$ and σ_n. The

algorithm proceed by first keeping the hyperparameters, $\boldsymbol{\phi}$ and σ_n fixed while $\mathbf{W}$ is updated by taking a step in the direction of decreasing negative log marginal likelihood. Toward this end, we follow Alg. 2. In the second stage, the updated $\mathbf{W}$ is frozen at it's current value and the negative log marginal likelihood is minimized over the hyperparameters $\boldsymbol{\phi}$ and σ_n. The updates in the second stage follow the L-BFGS [142] update scheme. Collectively, our approach resembles a coordinate descent scheme. These two stages are performed alternatively until the convergence criterion (relative decrease in the objective function value) falls below a selected threshold. Empirically, it is found that the best results are obtained by fixing the number of iterations per stage to 1. The full two stage minimization routine is summarized in Alg. 1. Since the objective function is highly non-convex in it's arguments, we repeat the optimization process several times with different initial starting points chosen randomly and pick the best local minima. To initialize the projection matrix, we sample a random matrix of size $D \times d$ where each element is sampled i.i.d. from a standard Gaussian distribution and extracting the $\mathbf{Q}$ matrix from it's QR decomposition. It can be shown that this procedure yields uniform samples from the Stiefel manifold [143].

Algorithm 1 Algorithm to maximize the likelihood through a two-step iterative procedure

Require: Inputs $\mathbf{X}$, Outputs $\mathbf{y}$, maximum number of iterations maxitr, tolerance ϵ

1: Initialize orthogonal matrix $\mathbf{W}$ by sampling each element independently from $\mathcal{N}(0,1)$.
2: Initialize $\boldsymbol{\theta}$ and σ_n.
3: Initialize counter $i \leftarrow 1$
4: **while** $i <=$ maxitr **do**
5: Update $\mathbf{W}_i^* \leftarrow \underset{W}{argmin}\mathcal{L}(\mathbf{W}_{i-1}^*)$ using Alg. 2
6: Update $\boldsymbol{\theta}_i^*, \sigma_{n,\,i}^* \leftarrow \underset{\boldsymbol{\theta}}{argmin}\mathcal{L}(\mathbf{W}_i^*, \boldsymbol{\theta}_{i-1}^*, s_{n,i-1}^*)$ using the L-BFGS algorithm [142].
7: Update objective function $\mathcal{L}_i \leftarrow \mathcal{L}(\mathbf{W}_i^*, \boldsymbol{\theta}_i^*, s_{n,i}^*)$
8: **if** $\frac{\mathcal{L}_i - \mathcal{L}_{i-1}}{\mathcal{L}_{i-1}} < \epsilon$ **then**
9: break
10: **end if**
11: Update counter $i \leftarrow i + 1$
12: **end while**
13: **return** $\mathbf{W}_i^*, \boldsymbol{\theta}_i^*, s_{n,i}^*$

2.3.2 Maximizing the likelihood with respect to the projection matrix

The active subspace projection matrix $\mathbf{W}$ is a tall-and-skinny matrix consisting of orthonormal columns and is a member of the *Stiefel manifold*. Formally, the Stiefel manifold is defined as follows:

Definition. *$\mathcal{V}_p(\mathbb{R}^n)$ is said to be the Stiefel manifold of matrices with n rows and p columns ($n \geq p$), if:*

$$\mathcal{V}_p(\mathbb{R}^n) = \{\mathbf{W} \in \mathbb{R}^{n \times p} \; s.t. \; \mathbf{W}\mathbf{W}^T = \mathbf{I}_{p \times p}\}, \tag{2.43}$$

where, $\mathbf{I}_{p \times p}$ is the $p \times p$ identity matrix.

We seek to maximize the marginal likelihood of the Gaussian process regression model with respect to the projection matrix $\mathbf{W}$, i.e.,:

$$\mathbf{W}^* = \text{argmax}_{\mathbf{W} \in V_d(\mathbb{R}^D)} \mathcal{L}(\mathbf{W}), \tag{2.44}$$

where,

$$\mathcal{L}(\mathbf{W}) = \mathcal{L}(\mathbf{W}, \boldsymbol{\theta}, s_n; \mathbf{X}, \mathbf{y}). \tag{2.45}$$

To solve the optimization problem in Eqn. (2.44), we adopt a gradient descent method suitably modified to take into account the manifold's Riemannian structure.

The algorithm we use is a modified gradient descent algorithm. It differs from classical gradient descent in two ways:

1. We will use the Riemannian gradient of the function rather than the Euclidean gradient.

2. We will search over a curve that is not a geodesic (the equivalent of a straight line on a manifold).

We would like to emphasize that we do not view this problem as a constrained problem on $\mathbb{R}^{D \times d}$. Rather we take the domain of $\mathcal{L}$ to be $\mathcal{V}_d(\mathbb{R}^D)$. For a more detailed description of this algorithm see [144]. We provide a brief summary of the theory of our modified gradient descent procedure in the following sections.

2.3.3 Riemannian gradients

A manifold is a geometric object which is locally equivalent to Euclidean space. Formally, a set $M \subset \mathbb{R}^n$ is called a smooth k-dimensional manifold if for all $x \in M$ there exists a relatively open subset of M that can be mapped onto $\mathbb{R}^k$ by a smooth bijection whose inverse is also smooth. The collection of all vectors which are tangent to M at any particular point x form a k-dimensional vector space, $T_x M$, called the tangent space. A Riemannian metric, is a family of inner products $g_x = \langle \cdot, \cdot \rangle_x$ defined

on the tangents spaces such that $\langle \cdot, \cdot \rangle_x$ varies smoothly in x. The ordered pair (M, g) is referred to as a Riemannian manifold.

It is important to note that a given set M can have more than one Riemannian metric g. For the Stiefel manifold, the two common choices of g are the Euclidean metric $\langle Z_1, Z_2 \rangle_e = \text{tr}(Z_1^T Z_2)$ and the canonical metric

$$\langle Z_1, Z_2 \rangle_c = \text{tr}(Z_1^T P_X Z_2) = \text{tr}(Z_1^T (I - \frac{1}{2} X X^T) Z_2).$$

The canonical gradient is appealing because it gives equal weight to each of the coordinates of the tangent space. For $X \in \mathcal{M}_n^p$, there exists an $n \times (n-p)$ matrix $X_\perp$ such that $(X X_\perp)$ is an orthogonal $n \times n$ matrix. It can be shown that an arbitrary element of $T_X \mathcal{M}_n^p$ can be written as

$$Z = XA + X_\perp B$$

where A is $p \times p$, skew-symmetric matrix and B is a $(n-p) \times p$ matrix. Using this representation, one may check that

$$\langle Z, Z \rangle_c = \sum_{i>j} a_{i,j}^2 + \sum_{i,j} b_{i,j}^2.$$

(Note that a skew-symmetric matrix is uniquely determined by its entries above the diagonal.) On the other hand, the Euclidean metric would give each entry $a_{i,j}$, $i > j$, twice as much weight as each entry $b_{i,j}$. For this reason, we will use the canonical metric in our gradient descent algorithm.

Letting $G = \mathcal{D}\mathcal{F}(X) = \left(\frac{\partial \mathcal{F}}{\partial X_{i,j}} \right)_{i,j}$, we define the canonical gradient of $\mathcal{F}$ at point X to be the unique is the unique element of $T_X \mathcal{M}_n^p$ so that

$$\langle \nabla_c \mathcal{F}, \xi \rangle_c = G\xi, \quad \forall \xi \in T_X \mathcal{M}_n^p.$$

One may use the definition of $\langle \cdot, \cdot \rangle_c$ to check that $\nabla_c \mathcal{F} = G - XG^T X$. We will find it convenient to write $\nabla_c \mathcal{F} = AX$ where $A = GX^T - XG^T$.

2.3.4 Search Curves

Since classical gradient descent algorithm in Euclidean space minimizes the objective function over straight lines, it would be natural for us to minimize over geodesics, i.e. curves of minimal length between two points. However, exact geodesics are typically very expensive to compute. Therefore, instead we use a curve $Y(t)$ defined by

$$Y(t) = (I + \frac{t}{2}A)^{-1}(I - \frac{t}{2}A)X.$$

One may check that $Y(t)^T Y(t) = I$ for all t and that $Y'(0) = -\nabla_c \mathcal{F}(X) = -AX$. Therefore,

$$\frac{d}{dt}\mathcal{F}(Y(0)) = -\frac{1}{2}\|A\|_F^2$$

Thus, on a sufficiently small time interval, $F(Y(t))$ is decreasing.

2.3.5 Curvilinear search based on the Armijo-Wolfe conditions

Given that the curve $Y(t)$ is a descent path generated by the skew-symmetric matrix A, we are interested in performing curvilinear search along $Y(t)$ to generate new trial points for the modified descent algorithm. The ideal step-size t^* is given by the global minimizer of the univariate function $\phi(t) = \mathcal{F}(Y(t))$. In computing t^* we face a trade-off between the time needed in obtaining the global optimizer of $\phi(t)$ and a value of t which would produce substantial decrease in the value of the objective function $\mathcal{F}$. In general it is very expensive to identify the global minimizer of ϕ and instead we resort to inexact line search for an ideal step size. Typically, line search algorithms try out a set of candidate step sizes before stopping to accept a particular choice based on the fulfillment of certain conditions. A popular set of conditions for inexact line search are the so called Armijo-Wolfe conditions.

We define the Armijo-Wolfe conditions for our curvilinear search as follows:

$$\mathcal{F}(Y_k(t_k)) \leq \mathcal{F}(Y_k(0)) + \rho_1 t_k \mathcal{F}_t^{'}(Y_k(0)), \tag{2.46}$$

$$\mathcal{F}_t^{'}(Y_k(t_k)) \geq \rho_2 \mathcal{F}_t^{'}(Y_k(0)), \tag{2.47}$$

where $\mathcal{F}(Y_k(t_k))$ is the value of the objective function evaluated at a new candidate trial point $Y_k(t_k)$ generated by a taking a step of t_k along the search direction, $\mathcal{F}(Y_k(0))$ is the value of the objective function at the current trial point, $\mathcal{F}_t^{'}(Y_k(0))$ is the derivative of the objective function w.r.t. to the step size t evaluated at the current trial point, $\mathcal{F}_t^{'}(Y_k(t_k))$ is the derivative of the objective function w.r.t. to t evaluated at the candidate next trial point and $\rho_1, \rho_2 \in [0, 1]$ are known as the Armijo-Wolfe parameters.

The quantity $\mathcal{F}_t^{'}(Y(t))$ is defined as follows:

$$\mathcal{F}_t^{'}(Y(t)) = tr(G^T Y^{'}(t)),$$

where,

$$Y^{'}(t) = -(I + \frac{t}{2}A)^{-1} A \left(\frac{X + Y(t)}{2} \right)$$

The inequality (2.46) ensures that any suitable candidate for the next trial point has to sufficiently reduce the value of the objective function as it moves along the descent curve $Y(t)$. The inequality (2.47) ensures that unacceptably short step sizes are ruled out. Typically, $\rho_2 \in (\rho_1, 1]$.

2.3.6 Full algorithm for optimizing W

The initial guess for **W** is set by orthgonalizing a random matrix whose elements are drawn from a standard normal distribution. Note that the objective function depends on **W** as well as an additional set of hyperparameters, all of which are kept constant during the optimization over the Stiefel manifold. We also note that this

optimization task is challenging oweing, primarily, to the difficulty of maintaining orthogonality constraints. The method is also prone to getting trapped in local minima.

For a given point in the feasible set, $\mathbf{W}$, line search step size τ, and the gradient $\mathbf{G} := \nabla_{\mathbf{W}}\mathcal{L}(\mathbf{W})$, we define:

$$\mathbf{A} := \mathbf{G}\mathbf{W}^T - \mathbf{W}\mathbf{G}^T, \tag{2.48}$$

where, A is a skew-symmetric matrix and $\mathbf{G}$ is the Euclidean gradient. The gradient descent update is then defined as:

$$\mathbf{Y}(\eta) = \left(\mathbf{I} + \frac{\eta}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I} + \frac{\eta}{2}\mathbf{A}\right)\mathbf{W}\ , \tag{2.49}$$

This update scheme is also known as the *Cayley transformation*. It is easy to show that the above update preserves the orthogonality of the gradient descent trial points. Note that the gradient descent update for $\mathbf{W}$ depends on the step size η and, consequently, the marginal likelihood also depends on η:

$$\mathcal{L}(\mathbf{W}) = \mathcal{L}(\mathbf{W}, \boldsymbol{\theta}, \sigma_n, \eta; \mathbf{X}, \mathbf{y}) \tag{2.50}$$

The full iterative process is described in Alg. 2.

Algorithm 2 Optimization of the active subspace projection matrix $\mathbf{W}$

Require: Marginal likelihood $\mathcal{L}$, Euclidean gradient $\mathbf{G}$, step size τ, maximum number of iterations M, tolerance ϵ

Sample an initial guess $\mathbf{W}_0$ such $\mathbf{W}_{0,ij} \sim \mathcal{N}(0,1)$.
Orthogonalize the columns of the initial guess using the Singular Value Decomposition (SVD).
Update counter $i \leftarrow 1$
while $i \leq M$ **do**
    Evaluate $\mathcal{L}_i, \mathbf{G}_i$ at $\mathbf{W}_i$
    Compute skew-symmetric matrix $\mathbf{A}$ using Eqn. (2.48).
    Select step size η using the Brent algorithm [145] - $\eta^* \leftarrow \underset{\eta}{argmax}\mathcal{L}(\eta)$.
    Update $\mathbf{W}^* \leftarrow \mathbf{Y}(\tau^*)$
    Update $\mathcal{L}^* \leftarrow \mathcal{L}(\mathbf{W}^*)$
    if $\mathcal{L}^* - \mathcal{L}_i < \epsilon\mathcal{L}_i$ **then**
        break
    end if
    Update counter $i \leftarrow i + 1$
end while
return $\mathbf{W}^*$

2.3.7 Model selection - picking the right active subspace dimensionality

The active subspace is an intrinsic property of the underlying unknown true response - it is not contigent on the experimental design used for collecting data necessary to approximate the response. As such, if the active subspace exists, there is a correct dimensionality, d associated with it. The classical approach reveals the right active subspace dimensionality through the magnitudes of the eigenvalues of the empirical covariance matrix of the gradients. Unfortunately, our gradient-free does not have an easy lookup procedure to figure out the correct active subspace dimen-

sionality. In this section, we focus on the specific model selection task of picking the correct dimensionality of the active subspace in our gradient-free setting. Bayesian model selection proceeds as follows. Suppose one has a set of candidate models which we denote as $\mathcal{M}$. $\mathcal{M}$ may be finite (such as the current case) or it may be an infinite set of candidate models. A prior distribution over the set of candidate models is posed, say $p(\mathcal{M})$. The process of picking the right model is then posed as the Bayesian inference task of estimating the posterior distribution over $\mathcal{M}$, conditioned over the observed data, $(\mathbf{X}, \mathbf{y})$, i.e.,

$$p(\mathcal{M}|\mathbf{X}, \mathbf{y}) \propto p(\mathcal{M})p(\mathbf{y}|\mathbf{X}). \tag{2.51}$$

Note that the likelihood term $p(\mathbf{y}|\mathbf{X})$ is the marginal likelihood, i.e., it is obtained by integrating over the entire set of model parameters, say, β. The term $p(\mathbf{y}|\mathbf{X})$ is known as the *model evidence*, and is infact the normalization constant of the posterior distribution of the model parameters, β conditioned on the observed data. The model evidence is, in practice, analytically intractable and must be suitably approximated [**?**, **?**]. A crude approximation to the model evidence is the so-called Bayesian information criterion (BIC) [146]. The BIC score of our statistical model with a d-dimensional active subspace is defined as:

$$\mathrm{BIC}(d) = p(\mathbf{y}|\mathbf{X}) - \frac{1}{2} N_{\mathrm{param}} \log N, \tag{2.52}$$

where, N is the number of samples in the dataset and N_{param} is the number of parameters in the model, i.e., the total number of hyperparameters being estimated. Effectively, the BIC score penalizes model fit (first term in Eqn. 2.52) with model complexity.

2.3.8 A note on computational complexity

The computational cost of the gradient-free approach to AS discovery is dominated by the optimization over the search curves in the Stiefel manifold (Alg. 2). Each step of the search curve optimization requires the solution of a symmetric $N \times N$ and a generic $d \times d$ matrix. A complete analysis of the computational complexity of the overall algorithm is beyond the scope of this work. However, we can easily infer that the computational cost of our methodology is significantly greater than that of the classical approach to AS, whose main cost arises from a single SVD of a $N \times D$ matrix, and the maximization of the likelihood of a GP with a $N \times N$ covariance matrix. This additional computational cost is the price one has to pay for foregoing the computation of the gradients.

2.4 Numerical experiments

Our `Python` implementation of the classic approach to active subspace recovery (Sec. 2.2) as well our proposed gradient-free approach based on GP regression (Sec. 2.3) is available publicly at https://github.com/PredictiveScienceLab/py-aspgp. All the results from this section can be replicated by using the code from this repository. Note that in all of our numerical examples, we apply standard preprocessing techniques to the data such as *standardization*, i.e., subtracting the empirical mean and variance of the test dataset from the observed outputs. It goes without saying, that, the preprocessing is only applied to the training examples to prevent corruption the test dataset.

We begin this section, in Sec. 2.4.1, by setting up a series of experiments using synthetic examples that exhibit an active subspace with a known dimensionality d. Toward this end, we set our test functions as high-dimensional multivariate functions which have a quadratic link function and the original high-dimensional input space is connected to the active subspace via a known projection matrix $\mathbf{W}$. In our experiments with these synthetic functions, we first demonstrate that our approach is

capable of recovering the right active subspace and in doing so circumvents the main drawback of the classical approach, i.e., the unavailability of gradient information about the quantity of interest. Furthermore, we set up a series of experiments to demonstrate:

1. The robustness of our methodology to observational noise,

2. The consistency of our methodology with respect to dataset sizes.

Our next numerical example is a classic problem in uncertainty quantification - a 2D elliptic partial differential equation with an unknown spatially varying diffusion coefficient that is modeled with a Gaussian random field. Specifically, the version of this problem we apply our methodology to, has gradients that enable us to draw comparisons with the classical approach. In Sec. 2.4.2, we apply our technique to this benchmark UQ problem with 100 input dimensions arising from a spectral representation of the stochastic diffusion coefficient. The comparisons to the classical gradient-based approach show the high degree of agreement between our approach and the classical approach. We wrap this section on numerical examples, by applying our proposed methodology to real engineering uncertainty quantification problem - that of propagating uncertainty through a high-dimensional dynamical system model representing the propagation of solitary waves in 1D particle chains known as granular crystals (Sec. 2.4.3). Granular crystals are of deep interest for a variety of applications due to their excellent shock absorbing properties arising from the propagation of these solitary waves or *solitons*. Specifically, this particle chain system is characterized by uncertainties in it's material and geometric properties which influence the properties of solitons propagating through the chain. Given that we do not have access to the gradients of the quantities of interest with respect to the input stochastic parameters, this UQ problem is not amenable to the classical approach.

2.4.1 Synthetic function with known underlying structure

To study the properties of our proposed gradient-free methodology for recovering active subspaces and comparing our approach to the gradient-based classical approach we set up a synthetic function with known active subspace as follows. Denote the synthetic function as f and let f accept a vector of inputs $\mathbf{x} \in \mathbb{R}^D$. For the sake of simplicity we assume the response from this function is a scalar, i.e., $f : \mathbb{R}^D \to \mathbb{R}$. By construction, f exhibits intrinsic low-dimensional structure and can be expressed as:

$$f(\mathbf{x}) = g(\mathbf{z}) = g(\mathbf{W}^T\mathbf{x}), \tag{2.53}$$

where, $\mathbf{W} \in V_d(\mathbb{R}^D)$ is a tall-and-skinny matrix with orthonormal columns. The function $g : \mathbb{R}^d \to \mathbb{R}$ is known as the *link function* and we define it as a quadratic function in the d-dimensional active subspace:

$$g(\mathbf{z}) = \mathbf{z}^T\mathbf{A}\mathbf{z} + \mathbf{b}^T\mathbf{z} + c. \tag{2.54}$$

The parameters of the quadratic, $\mathbf{A} \in \mathbb{R}^{d\times d}, \mathbf{b} \in \mathbb{R}^d$ and $c \in \mathbb{R}$ are picked by random sampling such that all of their elements are i.i.d. standard Gaussian, i.e., $\mathbf{A}_{ij} \sim \mathcal{N}(0,1), \mathbf{b}_i \in \mathcal{N}(0,1), c \sim \mathcal{N}(0,1)$. To ensure reproducibility, we fix the seed of our data generating scripts. The gradients of f are, ofcourse, computable exactly:

$$\nabla f(\mathbf{x}) = \left(\mathbf{b} + 2\mathbf{x}^T\mathbf{W}\mathbf{A}\right)\mathbf{W}^T. \tag{2.55}$$

We set the true input dimensionality of the function to $D = 10$ and investigate cases where $d = 1, 2$. To test robustness to noise, artificial additive Gaussian noise is added to the observations and is controlled by a standard deviation parameter, σ_n. We first verify that our methodology is:

1. able to recover *an* active subspace,

2. able to recover the *right* active subspace,

3. robust to observation noise,

4. consistent to the size of the dataset.

Function with 1 dimensional active subspace

We first set the dimensionality of the active subspace to be $d = 1$. The projection matrix is sampled from a uniform distribution on the appropriate Stiefel manifold and the parameters of g are also sampled according to the procedures outlined in the previous section. We begin our experiment by sampling 140 observations and artificially adding Gaussian noise to it with a variance of 0.1. The scripts for generating the data can be found in the `GitHub` repository for this work. For validation we use a dataset with 60 samples. Fig. 2.2 shows a comparison of the link function obtained with our proposed methodology and the link function obtained using the classical approach. We note that the two approaches show very good agreement with each other. It is worth noting the parameterization of the active subspace obtained in the gradient-free approach is unique upto arbitrary rotations and relabeling of the coordinate system in the projected space. Fig. 2.3 shows a comparison of the individual components of the projection matrix obtained from the two approaches and a comparison of their respective predictive accuracy. We note that we obtain very good predictions from our gradient-free approach which match the predictive quality from the classical approach that utilizes gradient information.

Function with 2 dimensional active subspace

We now set the dimensionality of the active subspace to be $d = 2$. The projection matrix is, again, sampled from the Stiefel manifold and the parameters of g are sampled as i.i.d. Gaussian. As with the 1D case, we generate data by sampling 140 observations and artificially adding Gaussian noise to it with a variance of 0.1.

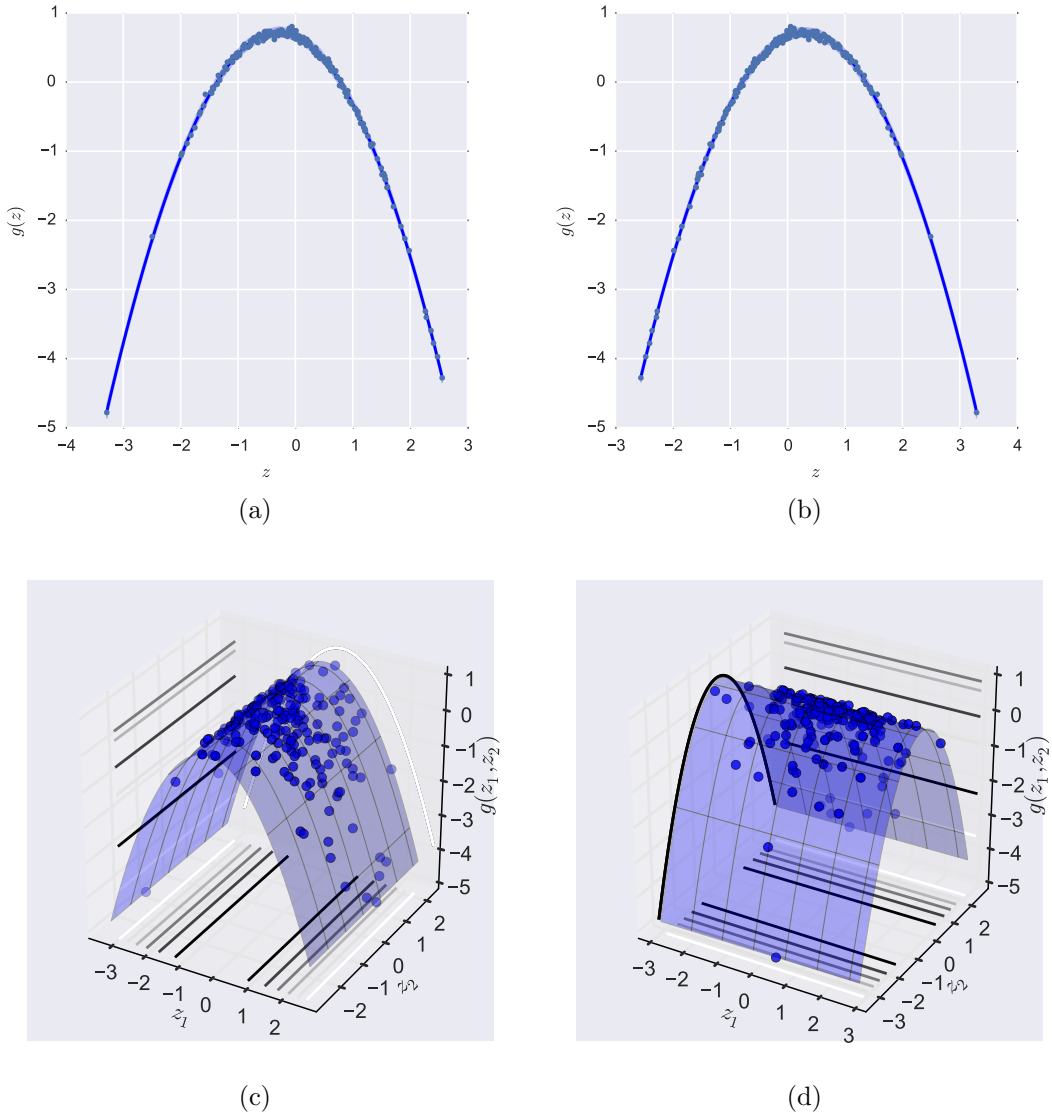


Figure 2.2. Results for one dimensional active subspace recovery in synthetic function. The left column shows a visualization of the link function obtained with the classical approach for AS recovery. The right column shows a visualization of the link function obtained with the proposed gradient-free GP approach for AS recovery. The scatter plots show the training data used to develop these constructions. Note the flatness of the function in one of the coordinate directions in the bottom panel - the function intrinsically exhibits a 1D AS.

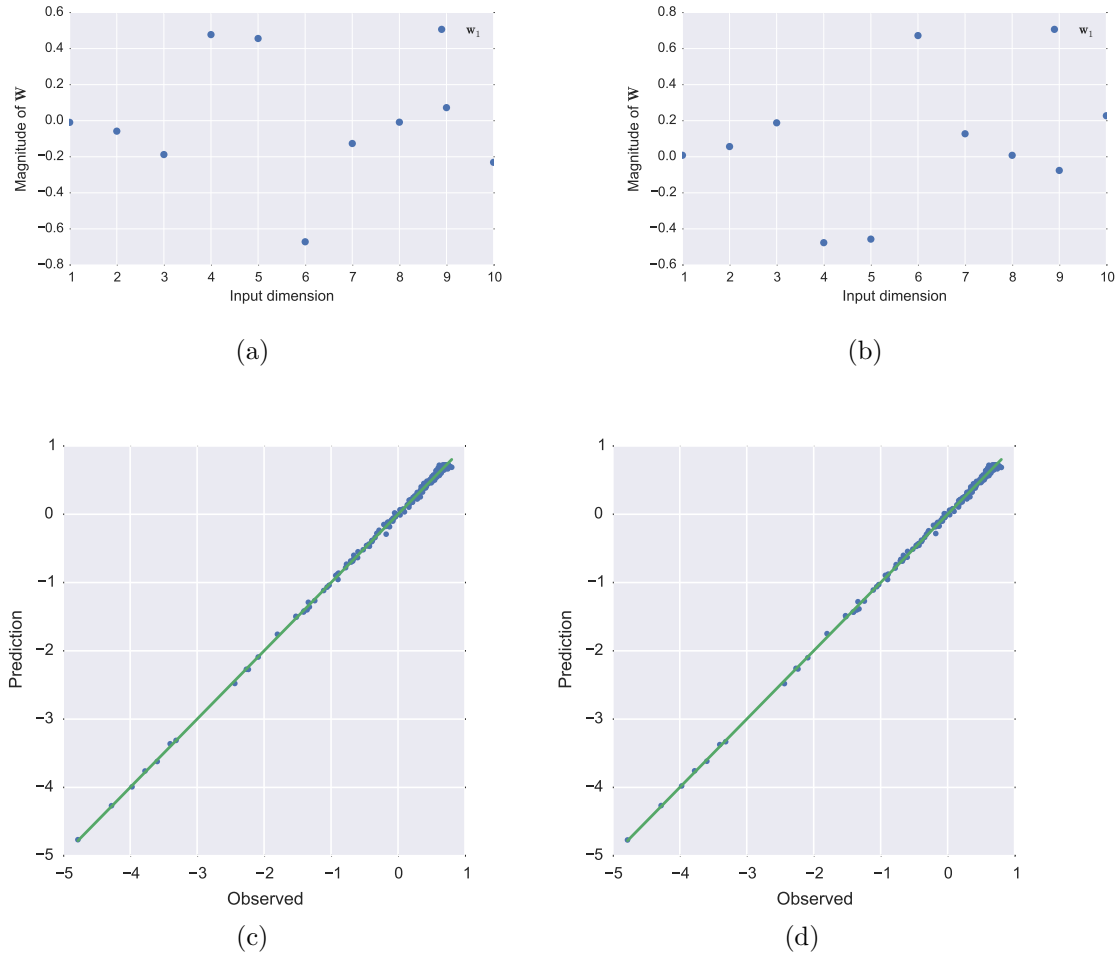


Figure 2.3. Results for one dimensional active subspace recovery in synthetic function. The left column shows the visualization of the components of the projection matrix (top) and the a comparison of the predicted and true outputs from the surrogate (bottom) obtained using the gradient-free approach. The right column shows the visualization of the components of the projection matrix (top) and the a comparison of the predicted and true outputs from the surrogate (bottom) obtained using the classical approach. Note that the components of the projection matrix obtained from the two approaches have the same magnitude but opposite signs.

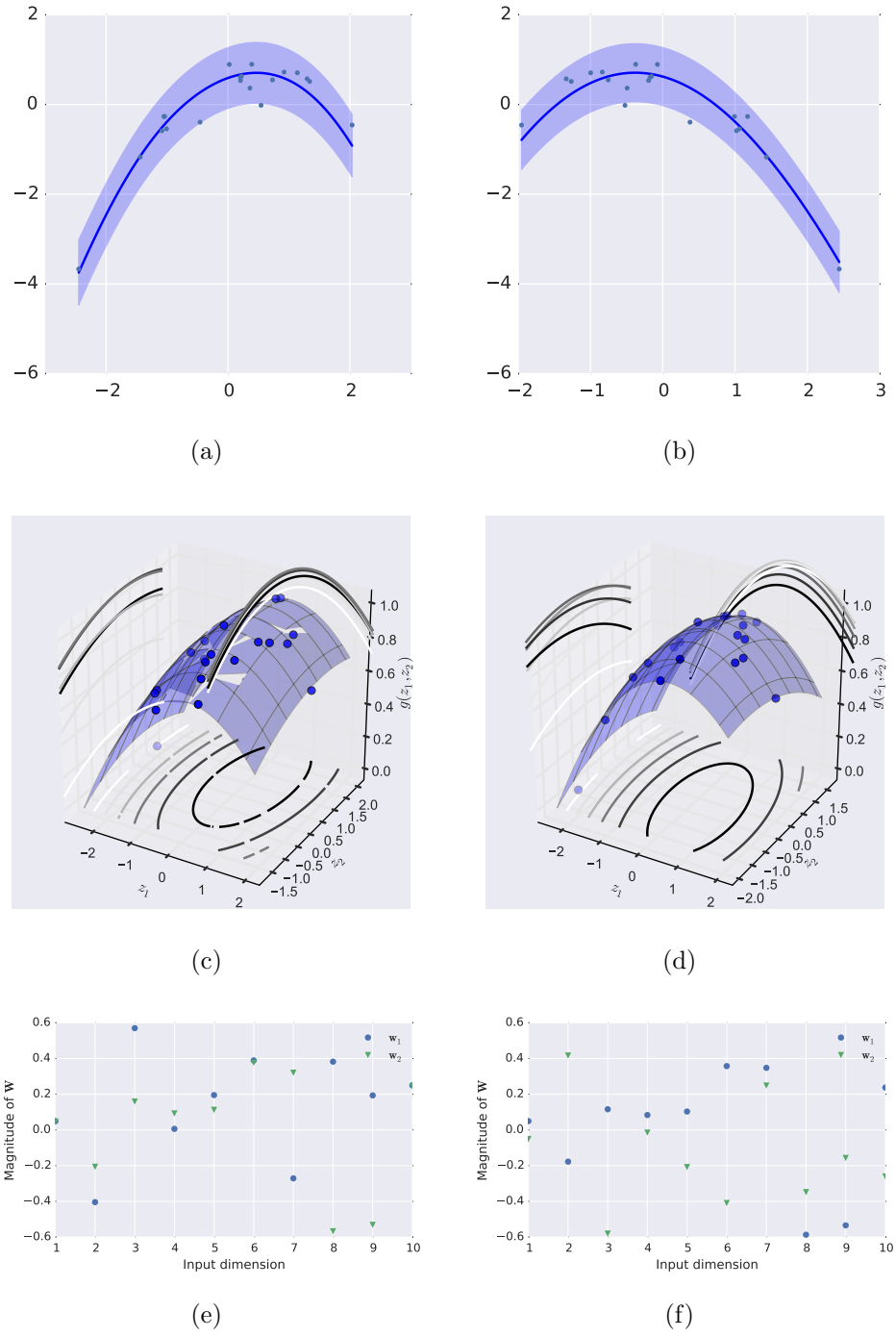
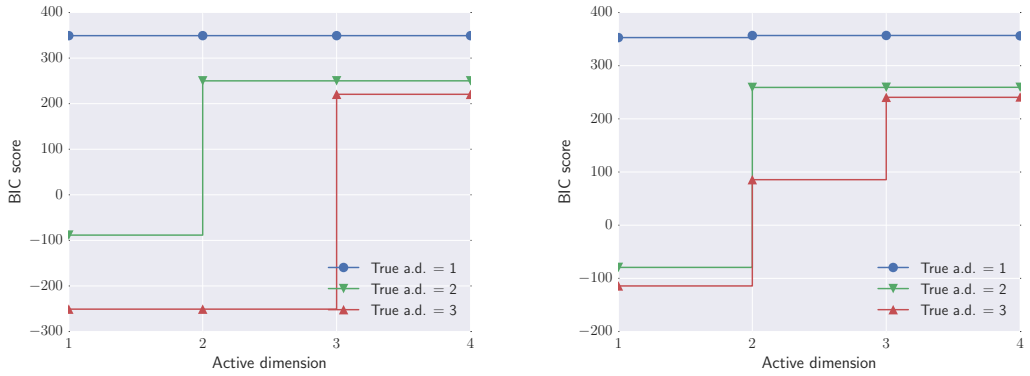


Figure 2.4. Results for two dimensional active subspace recovery in synthetic function. The left column shows a visualization of the link function obtained with the classical approach for AS recovery. The right column shows a visualization of the link function obtained with the proposed gradient-free GP approach for AS recovery. The scatter plots show the training data used to develop these constructions.



(a) (b)

Figure 2.5. Variation of the active subspace dimensionality with the Bayesian Information Criterion - the left plot shows the BIC vs d plot obtained from the gradient based approach to active subspace recovery while the right plot shows the BIC vs d plot obtained from the proposed gradient-free approach to active subspace recovery.

For validation we use a dataset with 60 samples. Fig. 2.4 shows a comparison of the link function obtained with our proposed methodology and the link function obtained using the classical approach. We note that the two approaches show very good agreement with each other. Once again, the representation of the active subspace obtained in the gradient-free approach is unique upto arbitrary rotations and relabeling of the coordinate system in the projected space.

Capturing the correct dimensionality of the active subspace

We now show, empirically, that our proposed approach, correctly (and automatically) recovers the correct dimensionality of the active subspace. Recall that the Bayesian information criterion is a crude approximation of the model evidence and equals the data likelihood penalized by model complexity as measured by the number of statistical parameters. Our hypothesis is that the BIC score will be maximized for the model with the correct active subspace dimensionality. Fig. 2.5 shows a variation of the BIC score with increasing active subspace dimensionality d, as obtained from the classical gradient-based and the proposed gradient-free approach to

active subspace recovery. We notice that the BIC score, for both methods, reaches a maximum for some d_{correct}, and then remains constant. Applying the rule that we pick the smallest active subspace dimensionality for the same BIC score, we observe the gradient-free approach recovers the same active subspace dimensionality as the classical approach.

Robustness and consistency of proposed methodology

As previously stated, the classical method for active subspace recovery is not robust to observation noise. In this section, we show that the the proposed gradient-free approach, which relies on a probabilistic formulation of the active subspace can work with, and is robust to, measurement noise. Furthermore, we also show that the proposed method is consistent, i.e., if we increase the number of the samples in the training dataset, the model performance improve. In setting up experiments to test these hypotheses, we introduce the following metric:

$$\epsilon_{\text{rel}} = \frac{\|\mathbf{W}_{\text{true}} - \mathbf{W}_{\text{proposed}}\|_F}{\|\mathbf{W}_{\text{true}}\|_F}. \tag{2.56}$$

The metric ϵ_{rel} in Eqn. (2.56) is the relative error in the measurement of the active subspace projection matrix from the gradient-free approach - $\|\cdot\|_F$ is the *Frobenius norm*, $\mathbf{W}_{\text{true}}$ is the true projection matrix obtained from the SVD of the gradient covariance matrix (classical approach) and $\mathbf{W}_{\text{proposed}}$ is the projection matrix obtained from the gradient-free approach.

The data we use to conduct these experiments come from input-output measurements of the synthetic function with low-dimensional quadratic structure by setting $D = 10$ and $d = 1$, i.e., the projection matrix is a column vector $\mathbf{W} \in \mathbb{R}^{10\times 1}$.

For our first experiment, we take a dataset of fixed size N and create several replicas of the output measurements. To each replica we add different levels of additive zero-mean Gaussian noise with variance σ_n^2. We then use each of these replicas along with the matrix of input variables and train separate models using the gradient-free

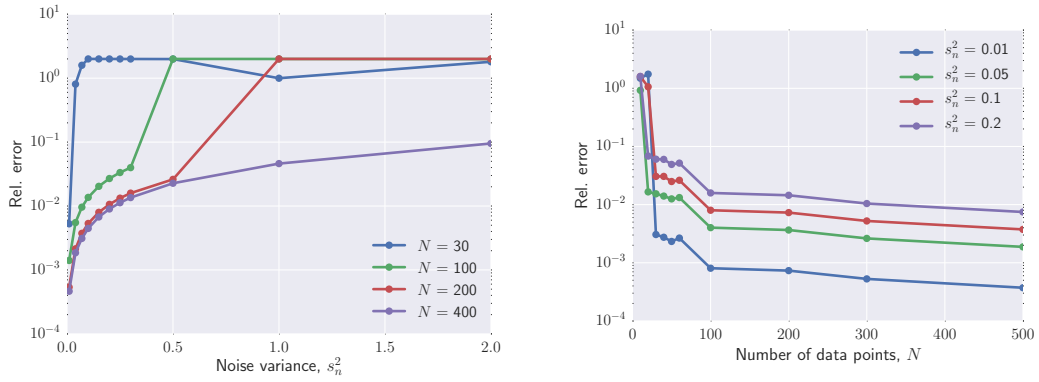
approach. For each model, corresponding to a dataset with a different noise level, we compute ϵ_{rel} and generate a plot of ϵ_{rel} as a function of σ_n^2. We repeat this procedure for several Ns. The results are shown in Fig. 2.6 (left plot). Notice how the relative error in the measurement of the projection matrix increases as a function of the noise variance. Also, datasets of higher N have lower ϵ_{rel} relative to datasets of lower N, for the same σ_n^2.

For our second experiment, we take several datasets of different Ns and corrupt their output measurements with noise of fixed σ_n^2. This is best done by taking one large dataset and taking appropriately sized subsets of the data. For instance, in our experiments, we generate one large dataset of size $N = 500$ and to create a dataset of size $N' < 500$, we take the first N' samples from the superset of size N. Since the original data is sampled randomly, randomness is preserved in the data subsets created this way. We now apply the gradient-free approach on these separate datasets of different sizes and note the relative error in the measured projection matrix. This experiment is repeated for several σ_n and a plot of the results are shown in Fig. 2.6 (right plot). Again, note how ϵ_{rel} decreases consistently on introducing more samples in the training dataset and also note how, for the same N, the relative error is lower for smaller σ_n.

2.4.2 Benchmark partial differential equation problem

We now tackle a classic problem in uncertainty quantification - the steady-state stochastic elliptic partial differential equation (PDE) in 2 dimensions with spatially varying diffusion coefficient. Consider the following PDE defined over the unit square $[0, 1]^2$:

$$\nabla \cdot (c(\mathbf{x}; \boldsymbol{\xi}) \nabla u(\mathbf{x}; \boldsymbol{\xi})) = 1, \ \mathbf{x} \in \Omega = [0, 1]^2, \tag{2.57}$$



(a) (b)

Figure 2.6. Robustness and consistency of the proposed gradient free approach to active subspace recovery - the left plot shows the variation in the relative error of the active subspace projection matrix as a function of the observation noise; the right plot shows the variation of the relative error in the projection matrix as a function of the number of samples in the training dataset.

subject to the following boundary conditions:

$$u(\mathbf{x}) \quad = 0, \ \mathbf{x} \in \Gamma_1, \tag{2.58}$$

$$\nabla u(\mathbf{x}) \cdot \mathbf{n} \quad = 0, \ \mathbf{x} \in \Gamma_2, \tag{2.59}$$

where, Γ_1 is the top, left and bottom sides of the domain and Γ_2 is the right side the domain. Γ_1 is equipped with homogeneous Dirichlet boundary conditions whereas Γ_2 is equipped with homogeneous Neumann boundary conditions. The diffusion coefficient is uncertain and spatially varying. Following the classic approach, the logarithm of a is modeled as a zero-mean Gaussian random field:

$$\log a(\mathbf{x}) \sim \mathrm{GP}(\log a | 0, k(\mathbf{x}, \mathbf{x}')), \tag{2.60}$$

where, the covariance function is set to be an exponential kernel with unit variance:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{\ell}\|\mathbf{x} - \mathbf{x}'\|_1\right). \tag{2.61}$$

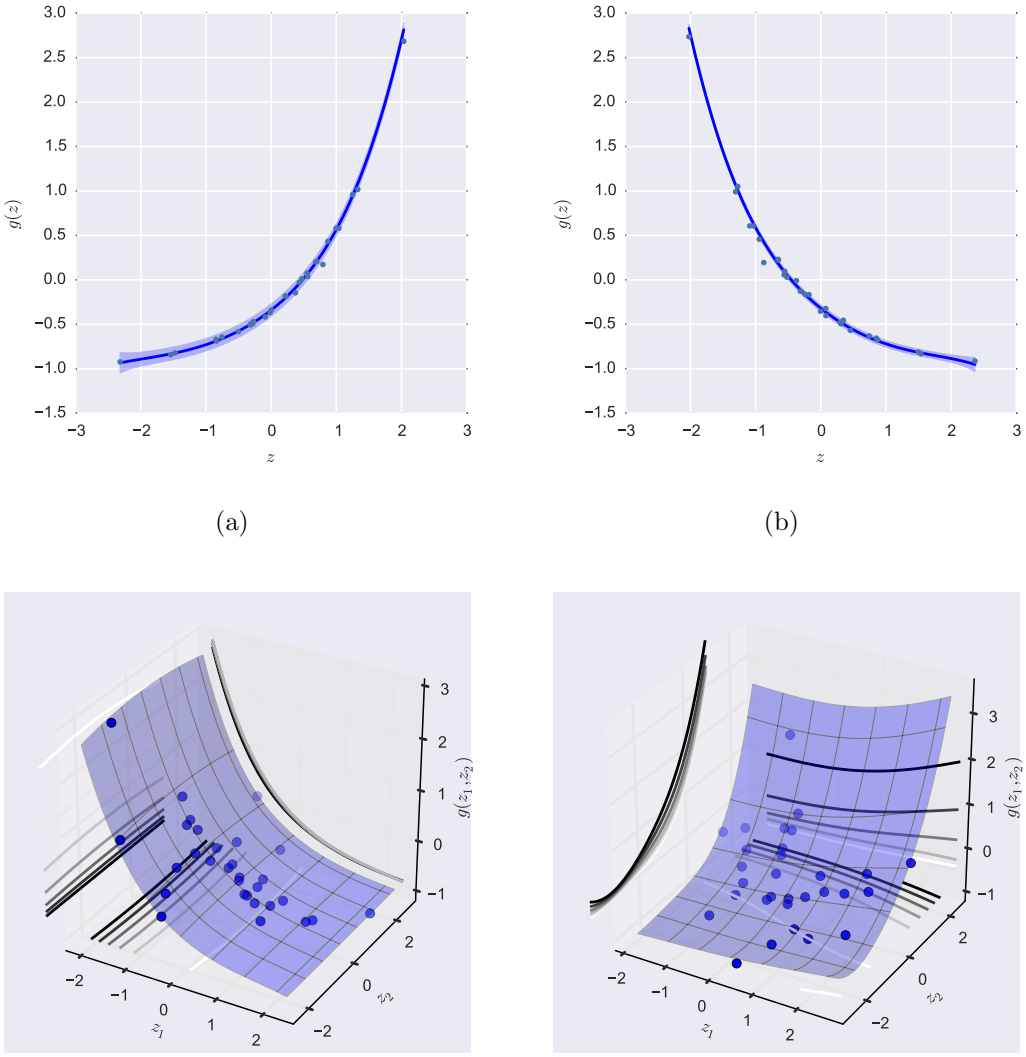
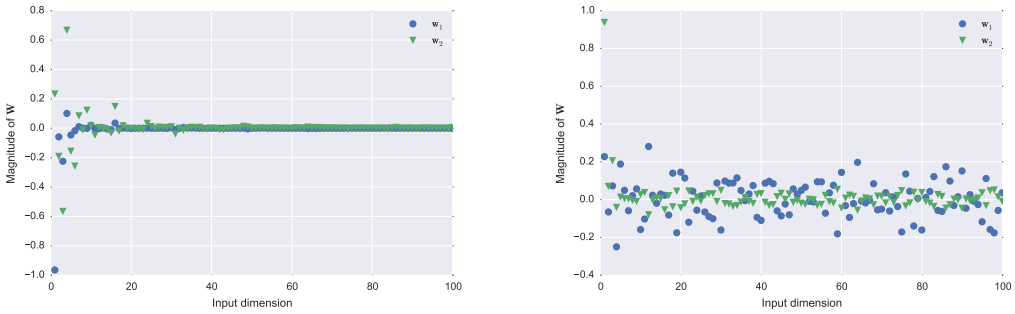


Figure 2.7. Results for the large lengthscale case for the stochastic PDE problem - the left column shows a plot of the link function obtained with the classic approach for $d = 1$ (top) and $d = 2$ (bottom) whereas the right column shows the link function obtained using the gradient-free approach for $d = 1$ (top) and $d = 2$ cases respectively.

ℓ is a lengthscale parameter that controls the spatial frequency of the uncertain coefficient a. Since $\log a$ is a infinite dimensional stochastic process, we produce a finite



(a) (b)

Figure 2.8. Results for the large lengthscale case for the stochastic PDE problem - the left plot shows the components of the active subspace projection matrix obtained with the classic approach while the right plot shows the components of the projection matrix estimated by the gradient-free approach.

dimensional representation by taking it's Karhunen-Loève expansion and truncating it at 100 terms:

$$\log a \approx \sum_{i=1}^{100} \sqrt{\lambda_i}\phi_i(\mathbf{x})\boldsymbol{\xi}_i, \tag{2.62}$$

where, $\boldsymbol{\xi}_i \sim \mathcal{N}(0,1)$ are i.i.d. standard normal random variables and $\{\lambda_i, \phi_i(\mathbf{x})\}$ are obtained through the solution of the eigenvalue problem:

$$\int k(\mathbf{x},\mathbf{x}')\phi_i(\mathbf{x}')\mathrm{d}\mathbf{x}' = \lambda_i\phi_i(\mathbf{x}). \tag{2.63}$$

The integral equation in Eqn. (2.63) cannot be solved analytically for our choice of covariance function and one must resort to a suitable numerical method to approximate the eigenfunctions and eigenvalues of the integral operator in Eqn. (2.63). Specifically, we resort to the Nyström approximation.

Denote all the $\boldsymbol{\xi}_i$s collectively as one vector $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_{100})^T \in \mathbb{R}^{100}$. The uncertainty in the diffusion coefficient is fully captured within the vector $\boldsymbol{\xi} \sim \mathcal{N}(0, I_{100})$. Given the uncertainty in the diffusion a, expressed through the stochastic process description of it's logarithm, we are interested in propagating this uncertainty to the

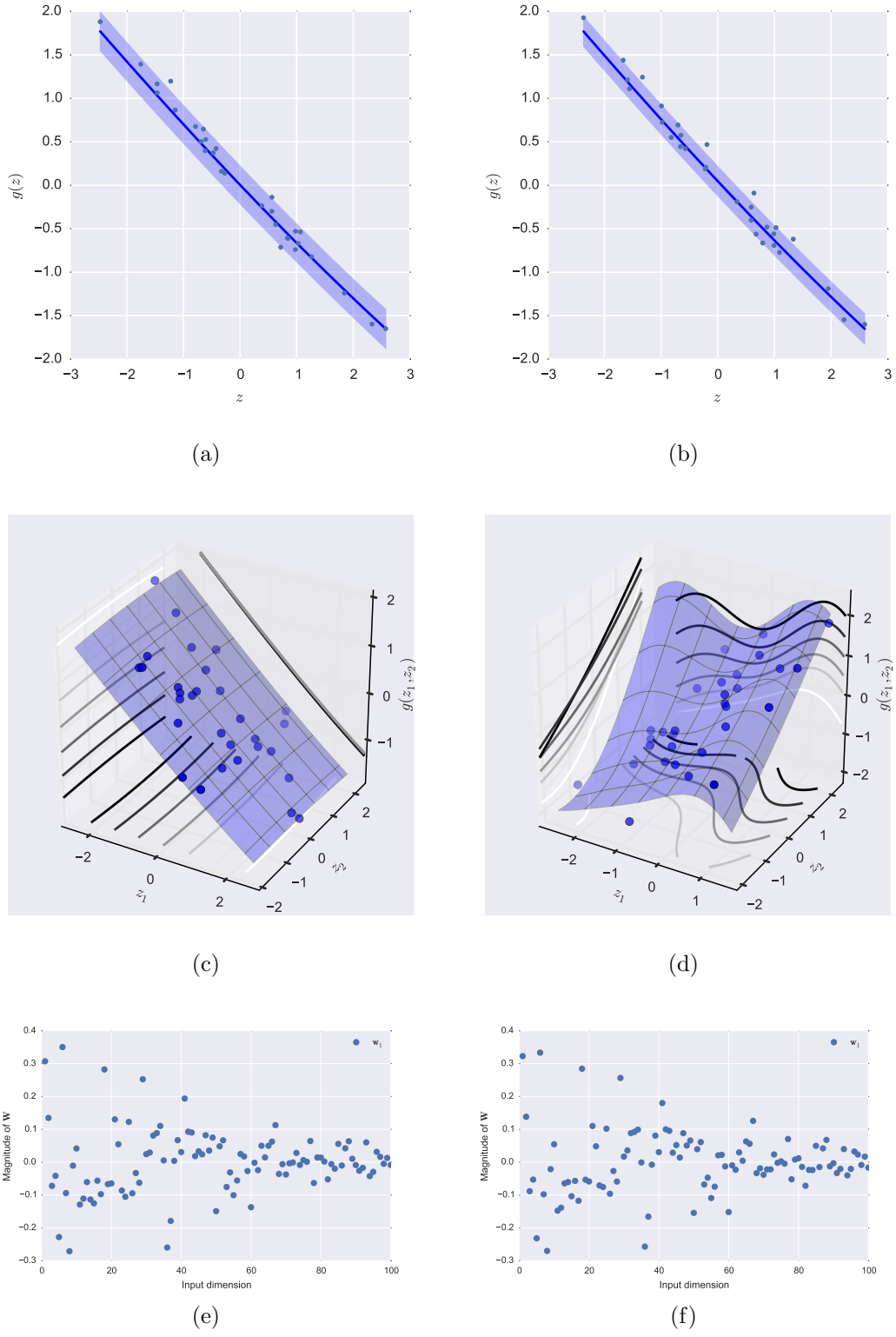
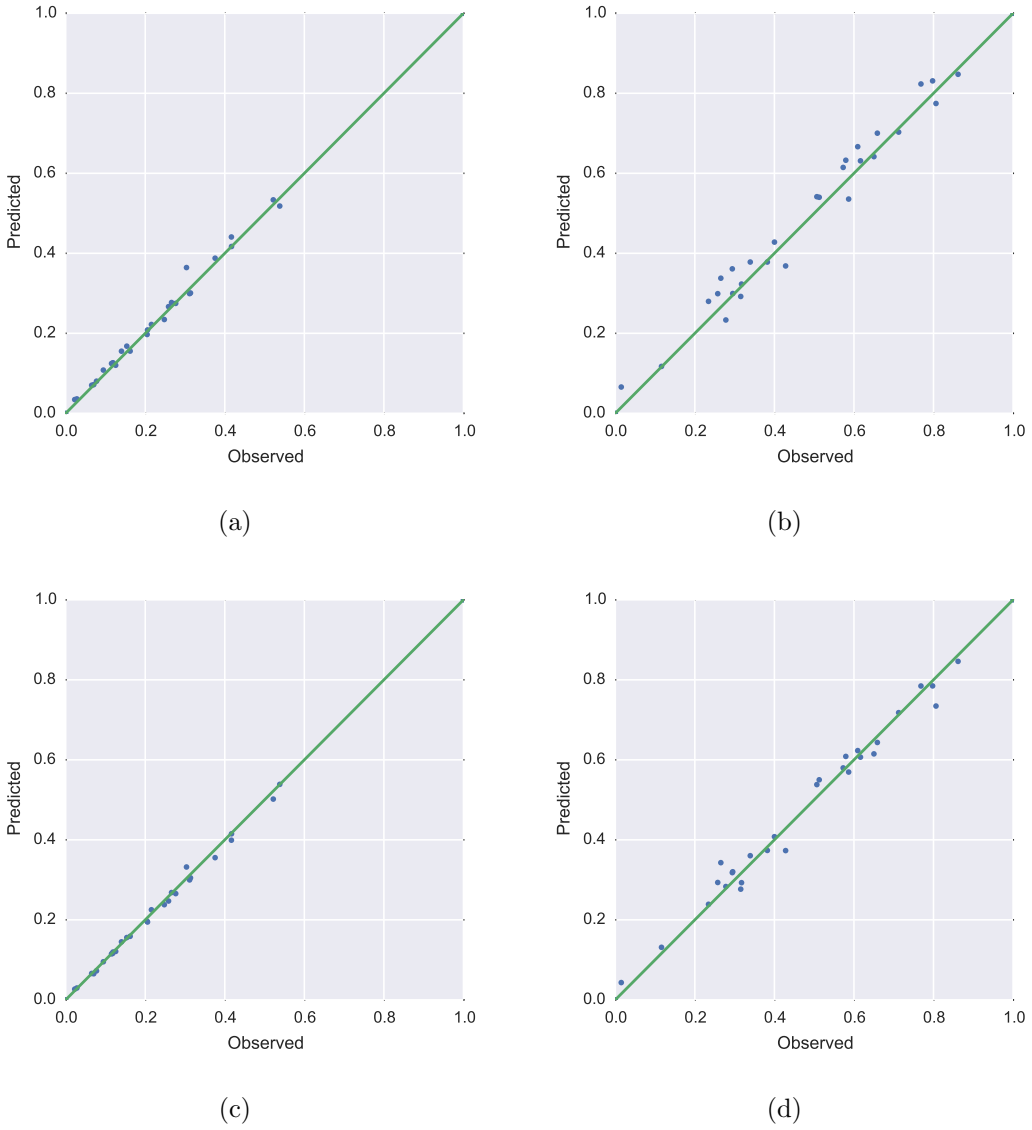


Figure 2.9. Results for the short lengthscale case for the stochastic PDE problem - the top row presents a comparison of the link functions obtained from the classic and gradient-free approaches for $d = 1$ while the bottom presents a comparison of the components of the projection matrix obtained from the classic and the gradient-free approaches.



(a) (b)

(c) (d)

Figure 2.10. Elliptic PDE. The dots correspond to true observed responses vs predicted ones for 30 validation inputs for the long ($\ell = 1$, left) and short ($\ell = 0.01$, right) correlation cases. Perfect predictions would fall on the green 45° line of each subplot. The top row corresponds to the gradient-free approach while the bottom row corresponds to the classic approach.

solution of the PDE and/or any functionals of the solution. Specifically, we consider the following quantity of interest:

$$q = \frac{1}{|\Gamma_2|} \int u(\mathbf{x}; \boldsymbol{\xi}) \mathrm{d}\mathbf{x}, \tag{2.64}$$

i.e., the scalar quantity of interest is a spatial average of the solution over the Neumann boundary. The PDE is solved numerical using a finite element solver and 300 samples of $\boldsymbol{\xi}$s and qs are generated, i.e. $\mathcal{D} = \left\{(\boldsymbol{\xi}^{(j)}, q^{(j)})\right\}_{j=1}^{300}$. Note that this a *small* dataset with *high* dimensional inputs. We conduct our experiments for 2 cases for the covariance lengthscale $\ell = 0.1$ and $\ell = 1$ which we will hereafter refer as the *short* and the *long* lengthscale cases respectively. The dataset is available publicly, and can be found at https://bitbucket.org/paulcon/active-subspace-methods-in-theory-and-practice/src and first appeared in Constantine's original paper introducing the classical approach to active subspaces [81]. The dataset is split into two parts - 270 samples for training and 30 samples for testing. The dataset source also contains gradients of q w.r.t. $\boldsymbol{\xi}$ thereby allowing us to draw comparisons of our approach with the classic approach.

The results of application of the classic and gradient-free approaches on the long lengthscale dataset are shown in Fig. 2.7 and Fig. 2.8. In Fig. 2.7, we show a comparison of the link functions obtained from the two approaches for one and two dimensional active subspaces. We observe that in both cases the link functions obtained through our approach matches the link function from the classic approach (upto rotation and relabeling of the axes ofcourse). In Fig. 2.8, we compare the active subspace projection matrix components obtained from both approaches and note good agreement.

Fig. 2.9 shows results for the case of the short lengthscale problem. We show that the link function obtained, for the case of $d = 1$ active dimensions, from both the classical as well as the proposed approach matches. In the same figure, we also show a side-by-side comparison of the components of the projection matrix obtained from the classical and gradient-free approach and note that the estimated gradient-free $\mathbf{W}$

matches the true $\mathbf{W}$ obtained from the SVD of the gradient covariance. We also present a comparison of the predicted outputs and true outputs from the test dataset for both lengthscale cases and both approaches - classical and gradient-free (see Fig. 2.10).

2.4.3 Propagation of geometric and material uncertainty in granular crystals

Granular crystals [147–152] which are a class of fabricated materials consisting of solid particles packed together in compact lattices have attracted significant attention because of their unique response to deforming forces. Specifically, granular crystals, under suitable external forcing, exhibit highly nonlinear dynamics, manifesting in the propagation of localized stress waves known as *solitons*. Mathematically, the response of granular crystal chains under external forcing is modeling through elastic Hertzian contact models [152]. The properties of the solitons propagating through granular crystal chains are a function of the material properties of the constituent particles (such as the elastic modulus) as well it's geometric properties (radii of the constituent particles).

We now setup the forward model for the soliton dynamics in an idealized 1D granular crystal chain. Suppose the chain consists of n particles whose displacements from the equilibrium is given by $\mathbf{r} = (r_1, r_2, \ldots, r_n)^T$. Let the radius and elastic modulus of the i^{th} particle be R_i and E_i. An external forcing is applied by striking the last particle with velocity v and the velocity v is itself uncertain. We define the vector of stochastic parameters as $\mathbf{x} = (R_1, \ldots, R_n, E_1, \ldots, E_n, v)^T$. The displacement r satisfies Newton's law, i.e., for the i^{th} particle, $m_i(\mathbf{x})\ddot{r}_i = f_i(\mathbf{x})$ with suitable initial conditions.

Let $r(t; \mathbf{x})$ be the displacement vector of the particles at time t as a function of the stochastic parameter vector $\mathbf{x}$. The force on each particle is, then, per Newton's laws, a function of the displacement r and the parameters $\mathbf{x}$. Suppose we observe the force

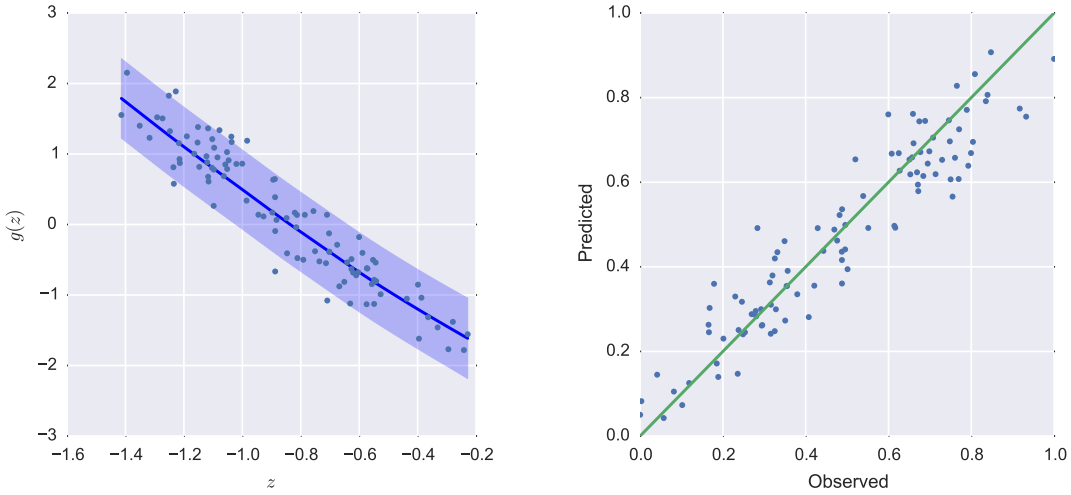
on all the particles at a discrete number of temporal points, $t_i, i = \{0, t_1, t_2, \ldots, t_{\text{final}}\}$. This is our output space and it's dimensionality is $n \times n_t$ where n_t is the number of time steps. We perform a preliminary dimensionality reduction by fitting the output to a solitary wave with particle dependent amplitude, frequency and velocity. Suppose, for the i^{th} particle, we denote these quantities as $A_i(\mathbf{x}), W_i(\mathbf{x})$ and $V_i(\mathbf{x})$. This reduces our output space to $n \times 3$ dimensions.

To this end, we will be observing the force on each particle as a function of time for a given set of parameters $\mathbf{x}$.

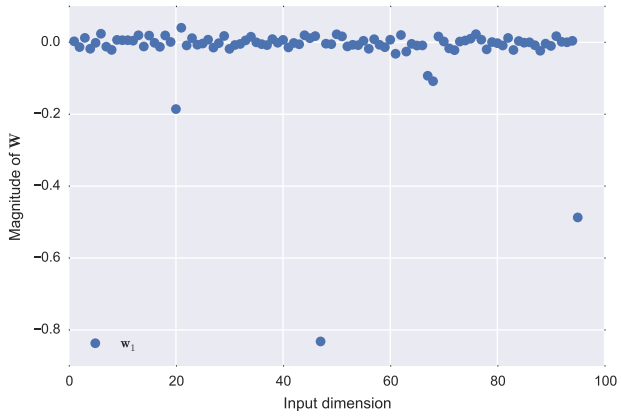
$$F_i\left(t; \mathbf{x}\right) \equiv F_i\left(\mathbf{q}\left(t; \mathbf{x}\right); \mathbf{x}\right). \tag{2.65}$$

That is, for each $\mathbf{x}$, we obtain, by integrating the equations of motion, the force at a finite number of timesteps, $0 = t_1 < \cdots < t_{n_t}$, say $\mathbf{F}(\mathbf{x}) := \{F_i(t_j; \mathbf{x}) : i = 1, \ldots, n_p, j = 1, \ldots, n_t\}$. The dimensionality of the output is given by the number of time steps times the number of particles. This is a very high-dimensional output and we first reduce it's dimensionality by fitting the output to a solitary wave whose properties-amplitude, wave width and velocity, are particle dependent. Let the properties of the soliton over the i-th particle be amplitude $a_i\left(\mathbf{x}\right)$, wave speed $v_i\left(\mathbf{x}\right)$ and width $w_i\left(\mathbf{x}\right)$. These are our quantities of interest. We treat each property corresponding to each particle as a separate scalar quantity of interest for apply our gradient-free surrogate model approach.

In our experiment, we have a chain with $n = 47$ particles. We will look at some of these quantities of interest - the properties of the soliton over particle numbers 20 and 30. Denote the quantities of interest as y_1 through y_6. 1000 simulations of the dynamical system are run with a Latin Hypercube (LHS) for the stochastic parameters, $\mathbf{x}$ and the resultant soliton properties are collected for surrogate modeling. In our experiments, we consider cases with and without inter particle gaps resulting in two different design matrices $\mathbf{X}_1 \in \mathbb{R}^{142}$ and $\mathbf{X}_2 \in \mathbb{R}^{95}$ respectively. To summarize,

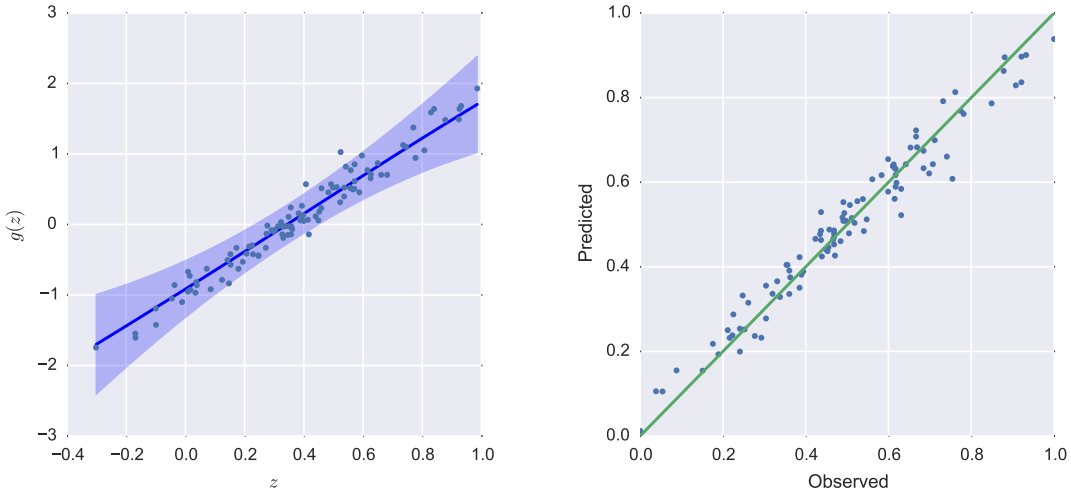


(a) (b)



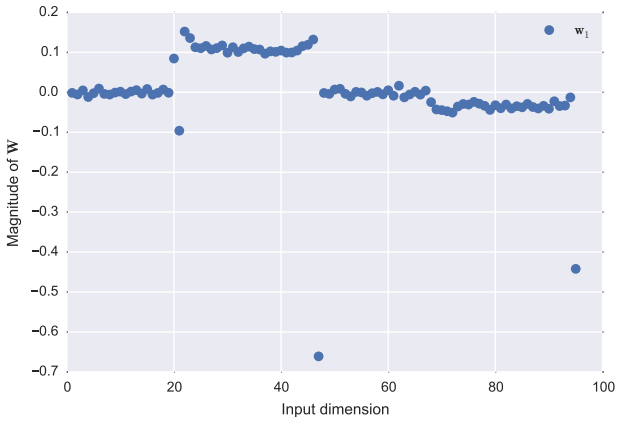
(c)

Figure 2.11. Results for the granular crystals without gaps when the soliton is over particle 20 - (a) link function for the amplitude of the soliton, (b) Comparison of the predicted and observed test amplitude values, (c) Components of the estimated projection matrix.



(a)

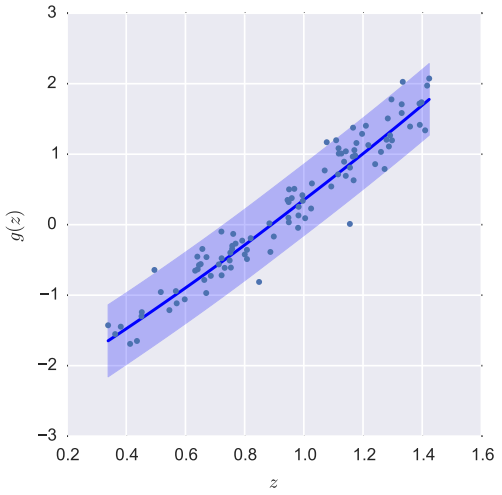
(b)



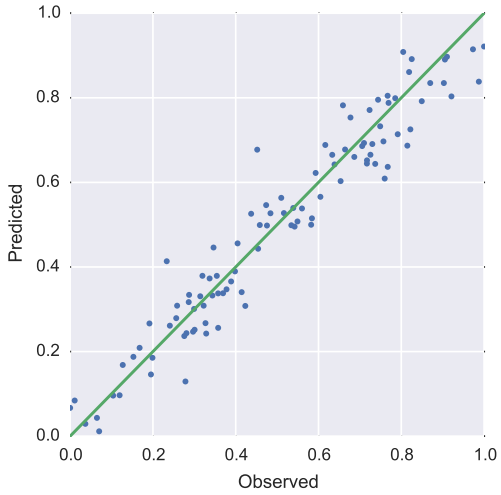
(c)

Figure 2.12. Results for the granular crystals without gaps when the soliton is over particle 20 - (a) link function for the time-of-flight of the soliton, (b) Comparison of the predicted and observed test time-of-flight values, (c) Components of the estimated projection matrix.

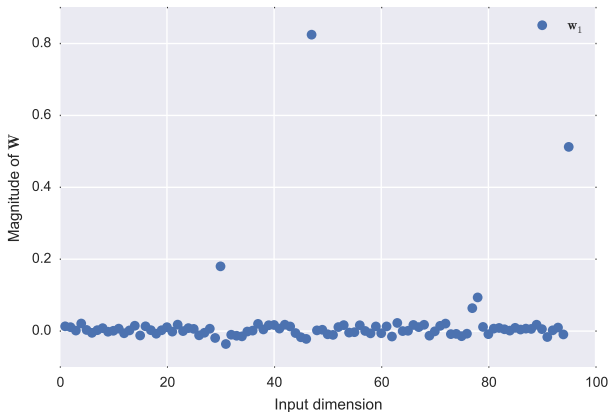
we have $2 \times 6 = 12$ datasets in total - one for each combination of input design and output quantity of interest. We construct separate surrogate models for each dataset.



(a)

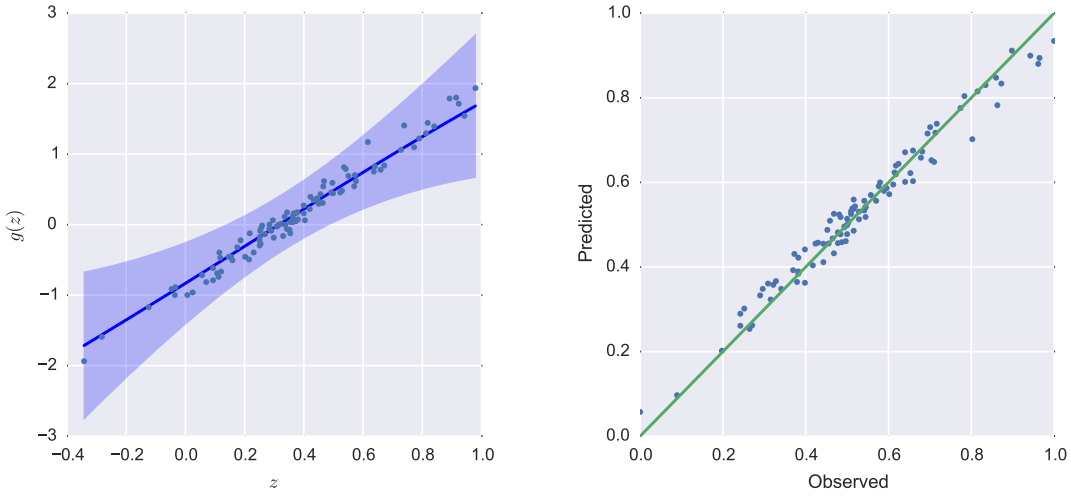


(b)

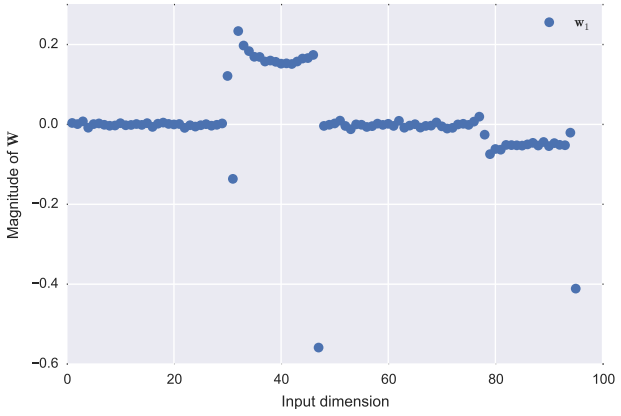


(c)

Figure 2.13. Results for the granular crystals without gaps when the soliton is over particle 30 - (a) link function for the amplitude of the soliton, (b) Comparison of the predicted and observed test amplitude values, (c) Components of the estimated projection matrix.

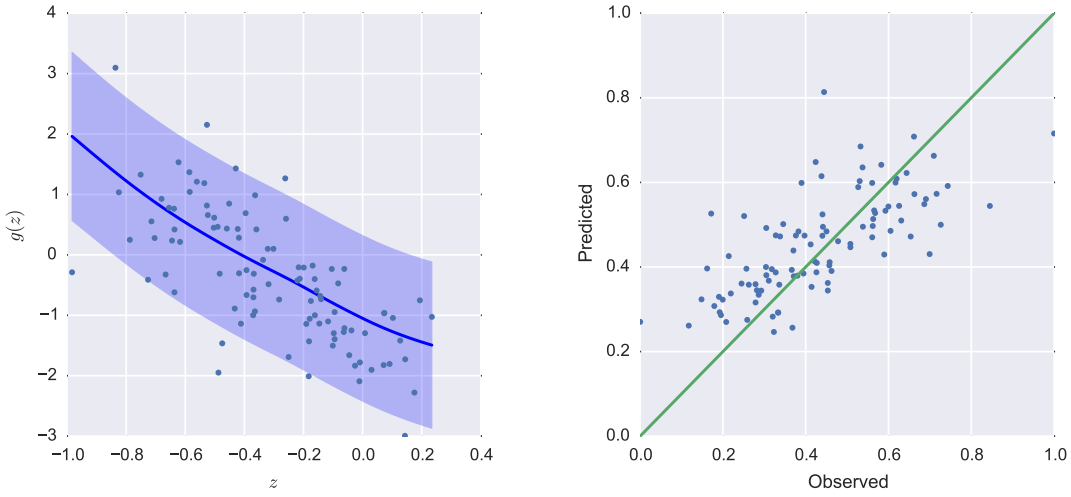


(a) (b)

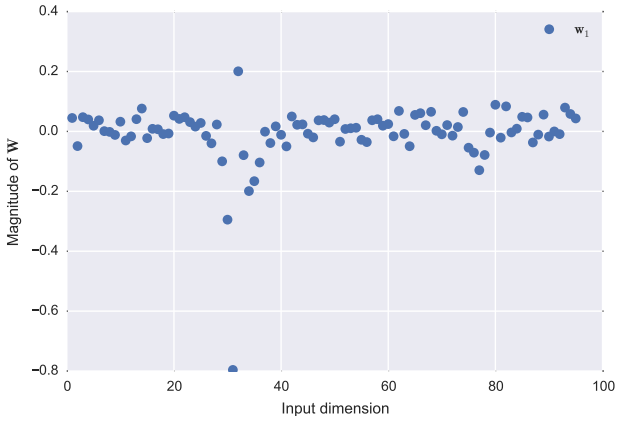


(c)

Figure 2.14. Results for the granular crystals without gaps when the soliton is over particle 30 - (a) link function for the time-of-flight of the soliton, (b) Comparison of the predicted and observed test time-of-flight values, (c) Components of the estimated projection matrix.

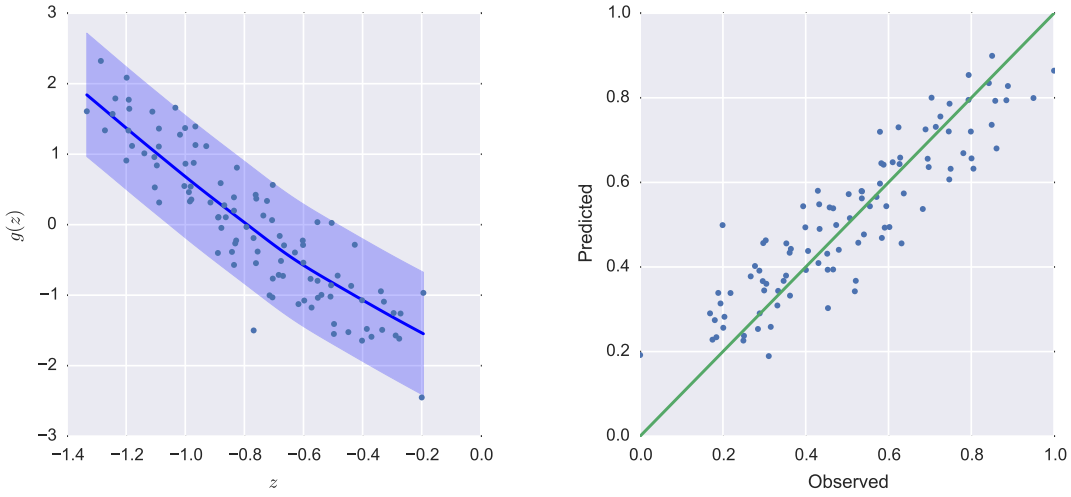


(a) (b)

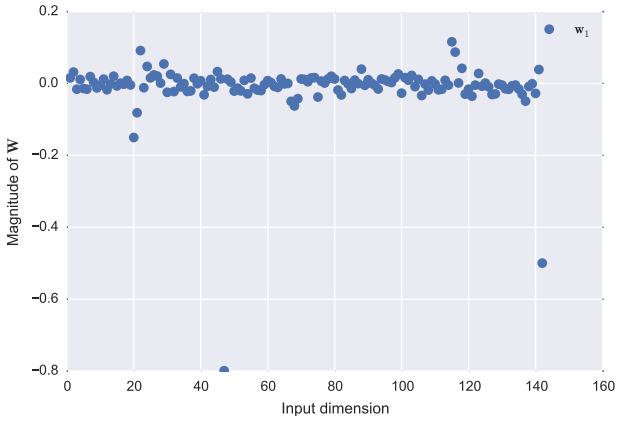


(c)

Figure 2.15. Results for the granular crystals without gaps when the soliton is over particle 30 - (a) link function for the width of the soliton, (b) Comparison of the predicted and observed test width values, (c) Components of the estimated projection matrix.

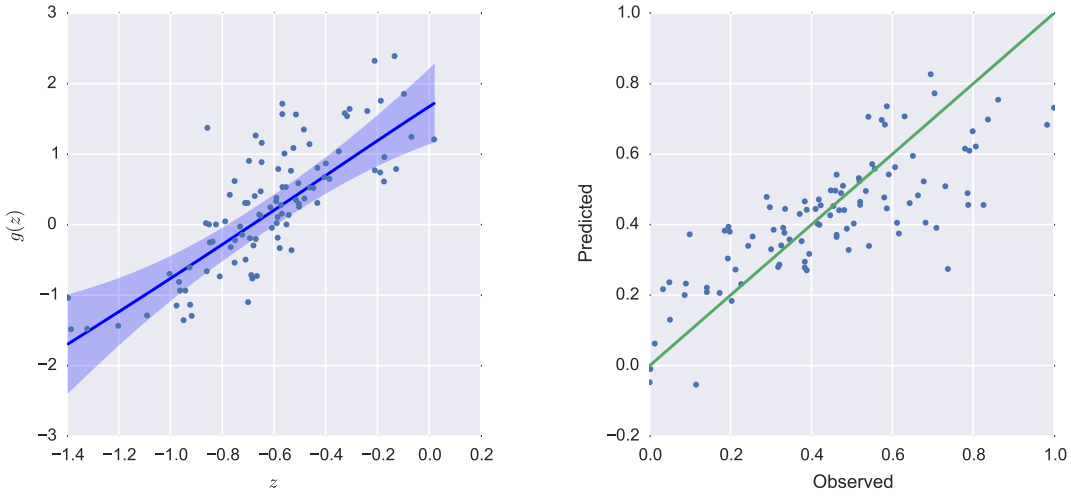


(a) (b)



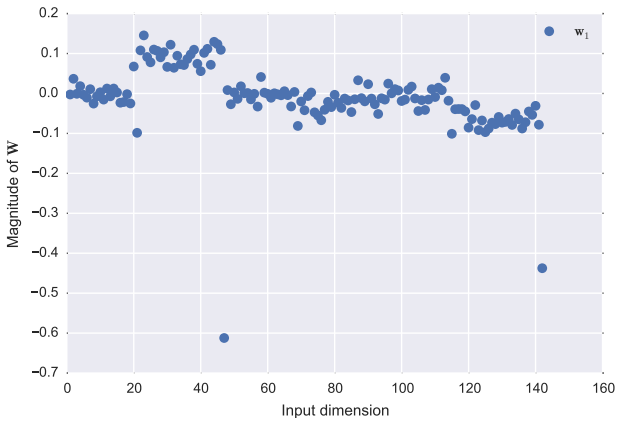
(c)

Figure 2.16. Results for the granular crystals with gaps when the soliton is over particle 20 - (a) link function for the amplitude of the soliton, (b) Comparison of the predicted and observed test amplitude values, (c) Components of the estimated projection matrix.



(a)

(b)



(c)

Figure 2.17. Results for the granular crystals with gaps when the soliton is over particle 20 - (a) link function for the time-of-flight of the soliton, (b) Comparison of the predicted and observed test time-of-flight values, (c) Components of the estimated projection matrix.

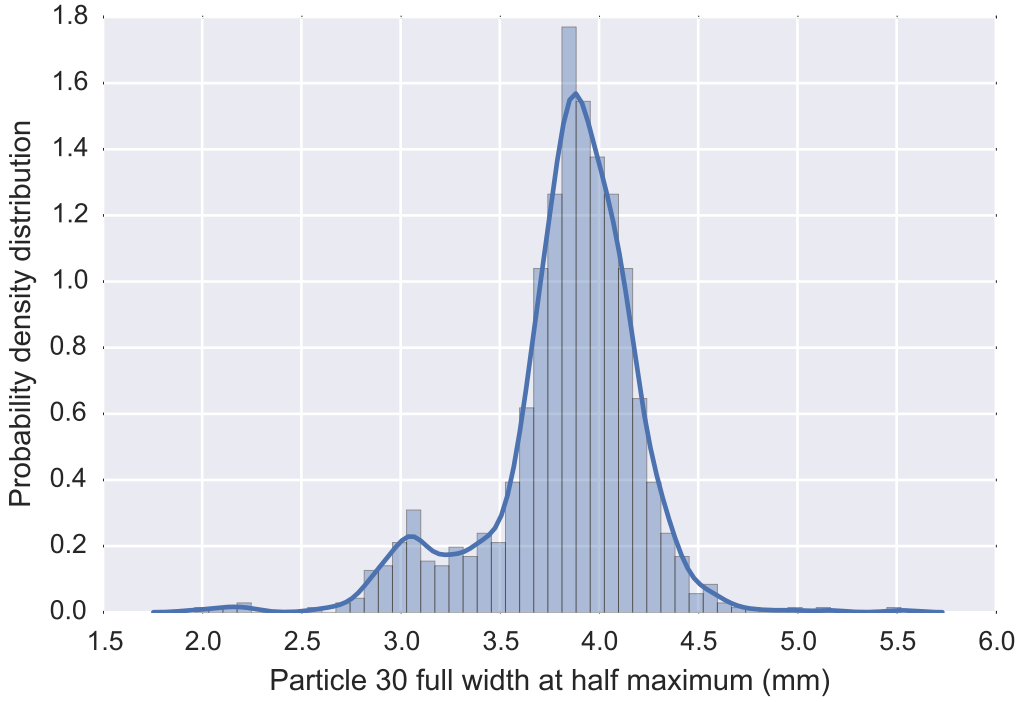
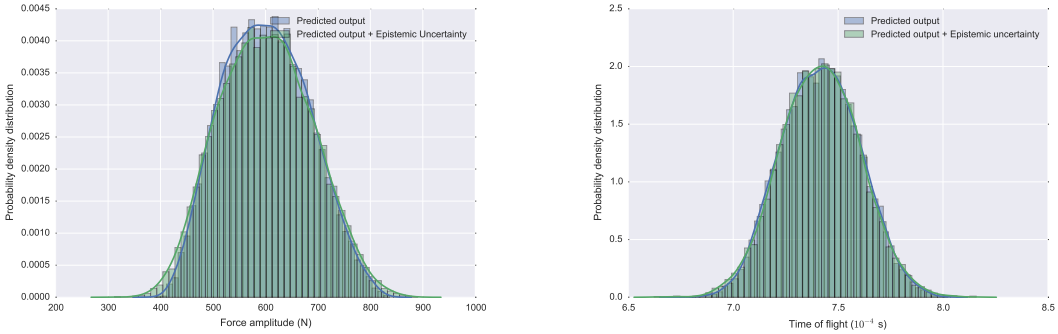
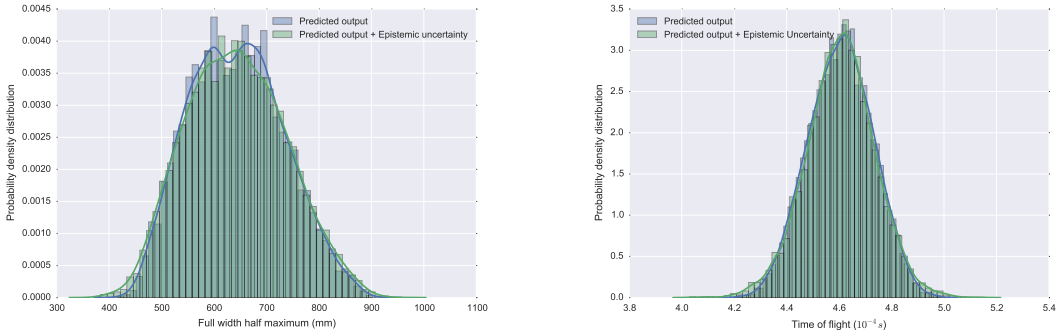


Figure 2.18. Histogram of the solition width over particle 30 for granular crystals with inter-particle gaps.



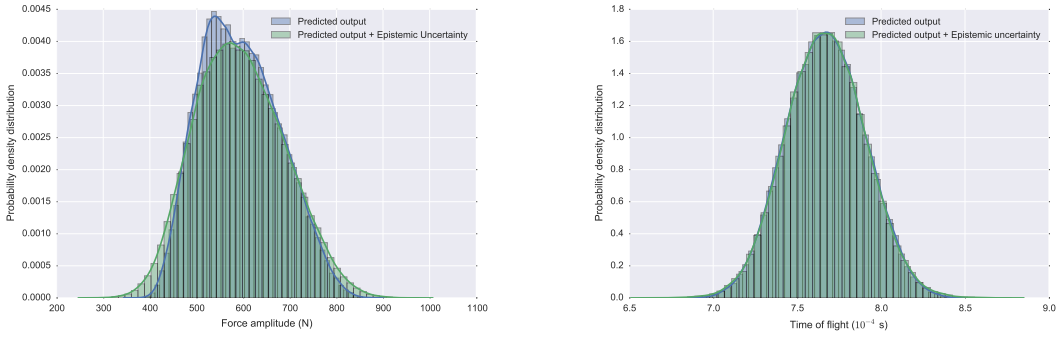
(a) (b)

Figure 2.19. Uncertainty propagation results for the granular crystals without gaps when the soliton is over particle 20 - (a) Marginal distribution of the soliton amplitude, (b) Marginal distribution of the soliton time-of-flight.



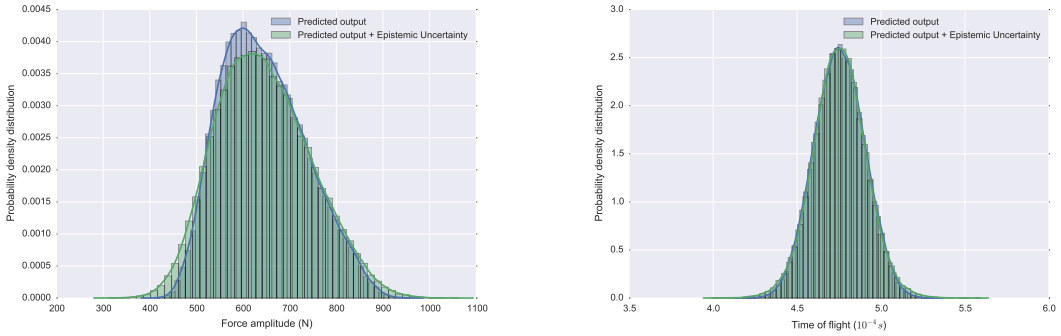
(a) (b)

Figure 2.20. Uncertainty propagation results for the granular crystals without gaps when the soliton is over particle 30 - (a) Marginal distribution of the soliton amplitude, (b) Marginal distribution of the soliton time-of-flight.



(a) (b)

Figure 2.21. Uncertainty propagation results for the granular crystals with gaps when the soliton is over particle 20 - (a) Marginal distribution of the soliton amplitude, (b) Marginal distribution of the soliton time-of-flight.



(a) (b)

Figure 2.22. Uncertainty propagation results for the granular crystals with gaps when the soliton is over particle 30 - (a) Marginal distribution of the soliton amplitude, (b) Marginal distribution of the soliton time-of-flight.

2.4.4 Uncertainty Propagation Results

For constructing our surrogates we use datasets of size 1000 and a further 100 samples are used for testing. We note that most of the variation of the quantities of interest under investigation resides on a 1 dimensional active subspace.

We show results for estimated one dimensional link function, projection matrix components and comparison of predicted test output with true test outputs for the following quantities of interest:

1. amplitude of the soliton over the 20^{th} particle with no interparticle gaps (Fig. 2.11),

2. time-of-flight of the soliton over the 20^{th} particle with no interparticle gaps (Fig. 2.12),

3. amplitude of the soliton over the 30^{th} particle with no interparticle gaps (Fig. 2.13),

4. time-of-flight of the soliton over the 30^{th} particle with no interparticle gaps (Fig. 2.14),

5. width of the soliton over the 30^{th} particle with no interparticle gaps (Fig. 2.15),

6. amplitude of the soliton over the 20^{th} particle with interparticle gaps (Fig. 2.16),

7. time-of-flight of the soliton over the 20^{th} particle with interparticle gaps (Fig. 2.17).

An interesting observation from these results is that the quantities of interest are practically independent of some of the inputs. This is deduced from observing the estimated components of the projection matrix - for any given quantity of interest, many of the projection matrix components are zero. For instance, the amplitude of the soliton is primarily affected the radius of the particles and the elastic modulus. In all cases, the BIC score does not change significantly from changing $d = 1$ to $d = 2$.

Having built our surrogate response, it is now easy to propagate the uncertainty in the input parameters through to the output. The input stochastic parameters are assigned uniform distributions within permissible bounds and we use the predictive mean of the GP surrogate corresponding to the quantity of interest in a MC sampling framework. We sample 10^5 realizations of the inputs and use the cheap surrogate to obtain samples of the output for each input realization. The resulting densities of the output quantities of interest are visualized through the histograms of their discrete samples and shown in Fig. 2.19, 2.20, 2.21 and 2.22.

2.5 Closing remarks

We developed an approach for recovering intrinsic low-dimensional structure, known as the *active subspace*, in functions mapping parameters of physical systems with physical quantities of interest. Specifically, our major contributions are circumventing the two main drawbacks of the classical approach:

1. We bypasses the need for samples of the gradients of the quantity of interest - a highly restrictive requirement of the classical approach.

2. We frame our approach in a probabilistic setting which allows us to seamlessly integrate observational noise.

Furthermore, we also develop a systematic approach for model selection, i.e., picking the correct dimensionality of the active subspace.

The work presented in this chapter presents a first step towards fully Bayesian active subspaces. Gaussian process surrogates provide principled estimates of the uncertainty in predictive distributions over test data. However, the approximate inference approach adopted in this chapter, does not account for epistemic uncertainty over the GP hyperparameters, in particular the projection matrix, induced as a result of limited data. Future developments of the work presented here should seek to quantify this uncertainty by extending the inference procedure to a fully Bayesian one. The stumbling block for any such development is the manifold constraint on the projection matrix. To efficiently perform Bayesian inference over the Stiefel manifold one might consider MCMC procedures obtained by modifying the proposal distribution to sample from the appropriate manifold [153]. To take the fully Bayesian approach a step further, one might also integrate the active subspace dimensionality into the statistical model as another latent variable and infer it using techniques such as the reversible jump MCMC [154].

3. HIGH DIMENSIONAL SURROGATE MODELING WITH DEEP NEURAL NETWORKS

[1] In Ch. 2, we presented a novel GP Regression based approach for constructing surrogate models for high-dimensional and expensive numerical simulators. Toward this end, we developed a novel covariance kernel which embeds an orthogonal transformation of the input data onto a low-dimensional linear manifold, formally referred to as the *Stiefel manifold*. The proposed approach was demonstrated on several synthetic examples, a benchmark elliptic PDE problem and a real engineering problem involving granular crystals. While the proposed approach did prove to be reasonably effective on problems with moderately high dimensions, there are still the following issues to contend with:

1. As we increase the dimensionality of the problem, the size of the projection matrix $\mathbf{W}$ also increases. This means that the number of tunable hyperparameters in the GP regressor also increases, making the maximum likelihood optimization more expensive. This poses a natural problem when the uncertain inputs are functional in nature (permeability fields for instance). This limitation is partially overcome through techniques such as the KL expansion. However, this requires making restrictive assumptions about the nature of the probabilistic structure of the underlying uncertainties (such as it's correlation model).

2. The cost of each optimization (or MCMC) step in GP regression is $\mathcal{O}(N^3)$, where N is the number of data samples. While numerous approaches based on 'pseudo-inputs' exists [156, 157] which reduce the computational burden to

[1] The contents of this chapter are reproduced, with permission, from the paper entitled "Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification" [155] published in the Journal of Computational Physics (2018).

$(O)(M^2N)$, where $M \ll N$, such methods have only shown to be effective on problems with a very small number of input dimensions.

With these issues in mind we lay-out a simpler surrogate model approach based on deep neural networks that is: a. scalable to very large uncertain parameter spaces, b. scalable to large dataset sizes without requiring expensive pseudo-input approximations.

3.1 Surrogate model structure

Recall that we are interested in solving the uncertainty propagation problem through a high-dimensional and expensive computer code denoted by f. This amounts to evaluating various intractable integrals with an expensive integrand over a high-dimensional state space. The easiest way to do so, would be to use the MC method. Unfortunately, as discussed earlier in Ch. 1, the MC method is computationally infeasible because of slow convergence in the number of forward model evaluations. Furthermore, information about f can only be obtained by querying the computer code at carefully selected design points. Say, we have N design points, which we collectively denote as $\mathbf{X}$:

$$\mathbf{X} = (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \cdots, \boldsymbol{\xi}_N). \tag{3.1}$$

$\mathbf{X}$ is an $N \times D$ matrix, with each row representing a single sample from $p(\boldsymbol{\xi})$. We evaluate the computer code for each of these N samples and obtain an estimate $y_i = f(\boldsymbol{\xi}_i)$ of the model output, $\forall i \in \{1, 2, \cdots, N\}$. We collectively represent all samples of the model output as,

$$\mathbf{y} = (y_1, y_2, \cdots, y_N). \tag{3.2}$$

$\mathbf{y}$ is an vector in $\mathbb{R}^N$, with each element of the vector representing a sample of the output. We denote the inputs and the outputs taken together as $\mathcal{D} = (\mathbf{X}, \mathbf{y})$. Thus, the task of building a surrogate model can be summarized as follows - given data $\mathcal{D}$

collected by querying the computer code at a finite number of input locations, we wish to build an approximation $\hat{f}$ of the true model f. We propose a form of the surrogate model $\hat{f}$, which projects the input data onto a nonlinear low dimensional manifold, i.e., $\hat{f} : \mathcal{X} \to \mathbb{R}$, such that,

$$\hat{f}(\boldsymbol{\xi}) = g(\boldsymbol{\zeta}) = g(h(\boldsymbol{\xi})), \tag{3.3}$$

where, $\boldsymbol{\zeta} \in \mathcal{Z} \subseteq \mathbb{R}^d$ is the projected input corresponding to the true input $\mathbf{x}$. We call the function $h : \mathcal{X} \to \mathcal{Z}$, the *projection function* and the function $g : \mathcal{Z} \to \mathcal{Y}$, the *link function*. The link function is a generic nonlinear function of the projected inputs, $\boldsymbol{\zeta}$. One immediately recognizes this structure as a generalization of the active subspace surrogate in [81] which expresses $\hat{f}$ as:

$$\hat{f} = g(\mathbf{W}^T \boldsymbol{\xi}). \tag{3.4}$$

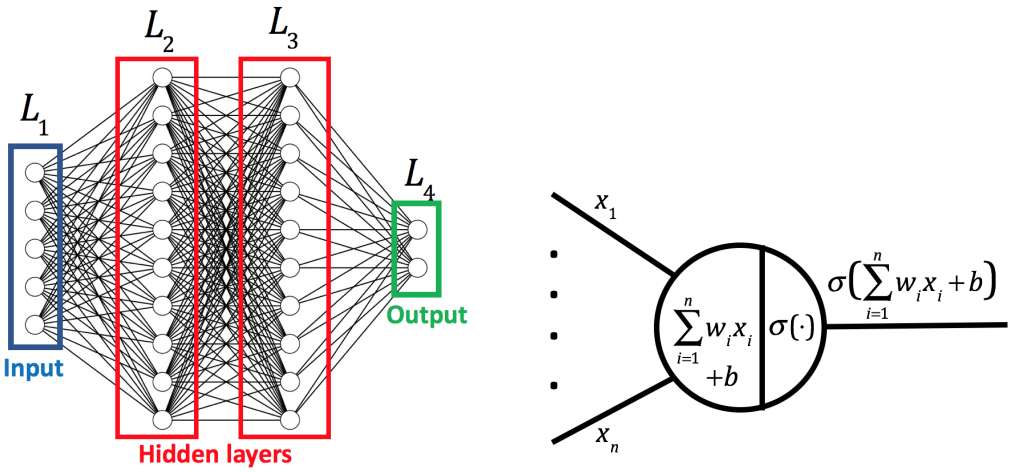
The proposed structure in Eq. 3.3 is capable of capturing directions which explain most of the variation in the model output. Alternatively, one also recognizes the above construction of the projection function as being the encoder section of neural network autoencoders[2] [112]. The complete structure is posed as a deep neural network (DNN) which we describe in the following section.

3.2 Structure of a feedforward Deep neural network

Neural networks (NN) are a powerful class of data-driven function approximation algorithms which represent information through a hierarchy of features. Each step in the hierarchy, beginning with the input, and ending with the final output, is known as a *layer*. Intermediate layers are known as *hidden layers*. By manipulating the number of hidden layers and the size of each hidden layer, one can learn functions of arbitrary complexity. The sizes of the input layer and output layer are fixed and determined

[2] An autoencoder is a kind of DNN used to recover a low dimensional embedding of a high dimensional space.

by the dimensionality of the input and output. Fig. 3.1(a) shows a schematic of a NN with 2 hidden layers. Each circle in the schematic of the NN represents the fundamental unit of computation in a NN, known as the *neuron*. A neuron accepts one or more inputs and produces an output by performing a linear transformation followed by an element-wise nonlinear transformation. A schematic of a single neuron and the computations taking place within it are shown in Fig. 3.1(b). We discuss the symbols in full detail in the subsequent paragraphs.



(a) (b)

Figure 3.1. 3.1(a)-Schematic of a neural network (NN). 3.1(b) - Schematic of a single neuron.

A DNN is simply a NN with a large number of hidden layers. The output of a layer is known as the *activation*. The activation from one layer of a DNN is used as the input to the next layer. The activation produced by the j^{th} hidden layer of the DNN is given by:

$$\mathbf{z}^{(j)} = \sigma(\mathbf{W}^{(j)}\mathbf{z}^{(j-1)} + \mathbf{b}^{(j)}),\ \forall j \in \{1, 2, \cdots, L\}, \tag{3.5}$$

where $\mathbf{W}^{(j)} \in \mathbb{R}^{d_j \times d_{j-1}}$, $\mathbf{b}^{(j)} \in \mathbb{R}^{d_j}$ and d_j is the number of neurons in the j^{th} hidden layer. Note that $\mathbf{z}^{(0)}$ is the input $\boldsymbol{\xi}$ and $d_0 = D$. σ is a nonlinear function applied element-wise on its arguments. Popular choices for σ include the *logistic* function,

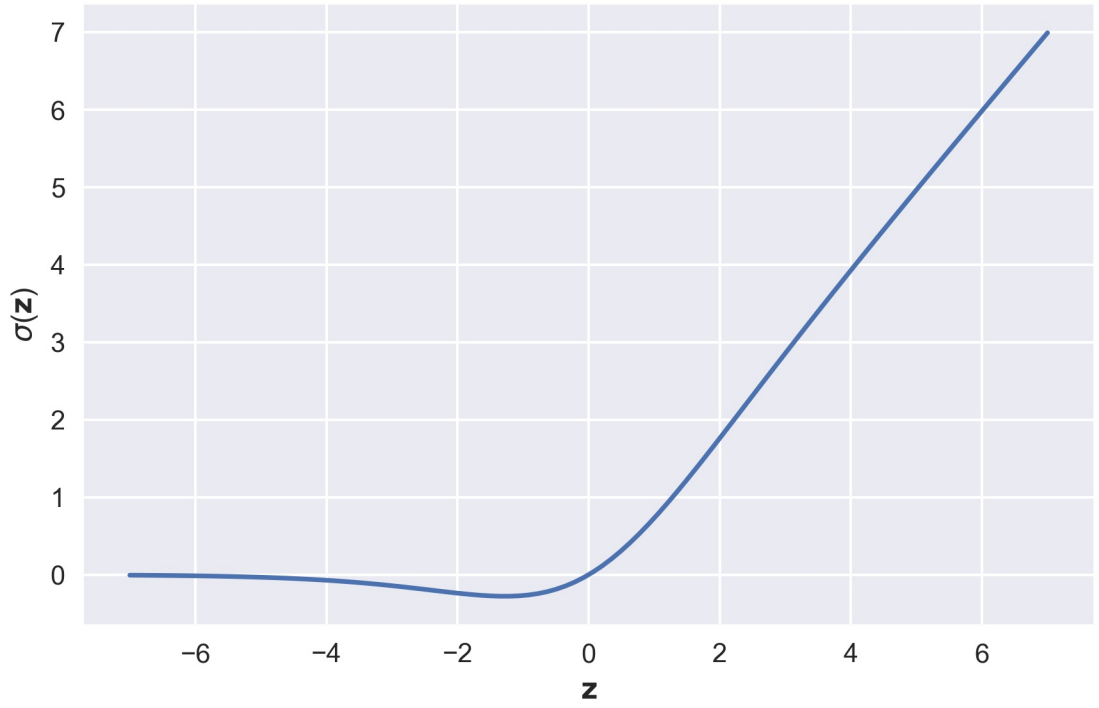


Figure 3.2. Swish activation with $\gamma = 1$.

the *hyperbolic tangent* function or the *rectified linear unit* (or ReLU) function [112]. The ReLU function has been utilized extensively in recent times, and has been shown to eliminate the need for an unsupervised pretraining phase while training deep architectures [158].

However, a recent result described in [159] demonstrates the superior performance of the *Swish* activation function defined as follows:

$$\sigma(\mathbf{z}) = \frac{\mathbf{z}}{1 + \exp(-\gamma \mathbf{z})}, \tag{3.6}$$

such that γ is either a constant or a hyperparameter to be learned from data. In this work, we use the Swish activation function with $\gamma = 1$.

The quantities $\mathbf{W}^{(j)}$ and $\mathbf{b}^{(j)}$, $\forall j \in \{1, 2, \cdots, L+1\}$, are known as the *weights* and *biases* of the network, respectively. Collectively, they are known as the *parameters*

of the network, $\boldsymbol{\theta} = \{\mathbf{W}^{(j)}, \mathbf{b}^{(j)}\}_{j=1}^{L+1} \in \boldsymbol{\Theta}$. The weights and biases together, fully describe the structure of the network, known as the *network architecture*.

3.3 Training a deep neural network

As discussed in the previous section, $\hat{f}$ is a parameterized function with parameters $\boldsymbol{\theta}$. Estimating $\boldsymbol{\theta}$ reduces to the problem of minimizing a loss function $\mathcal{L}(\boldsymbol{\theta}; f)$, which captures the mismatch between f and $\hat{f}$. For regression tasks $\mathcal{L}$ is typically chosen to be the mean squared error. In practice, we do not have access to the function f; only a limited set of observations, $\mathcal{D}$. Suppose $\mathcal{D}$ is a dataset of N examples, with the i^{th} example denoted as $\mathcal{D}_i = (\boldsymbol{\xi}_i, y_i)$. The training problem is cast as minimizing the mismatch between a prediction $\hat{f}(\boldsymbol{\xi}_i)$ and the correct output, y_i.

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{arg\,min} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\boldsymbol{\theta}; y_i). \tag{3.7}$$

In practice, the averaging in Eq. 3.7 is performed over a small randomly sampled subset (or *mini-batch*) $\mathcal{D}_M \subset \mathcal{D}$, at each iteration of the optimization procedure.

3.4 Regularized loss function

Since DNNs are prone to overfitting [126], one resorts to penalizing the misfit function with an appropriate penalty term known as a regularizer. This ensures that the DNN generalizes better to unseen data. Popular choices for regularization include the scaled L_1 norm or L_2 norm of the weights [112]. Both the L_1 and L_2 norm penalties promote shrinkage in the parameter values, with the L_1 penalty additionally promoting sparsity in the network weights. The *elastic net* regularizer introduced in [160] is a mixture between the L_1 and L_2 regularizers and is known to combine the advantages of L_1 and L_2 penalties (See [160]). While typically the L_1 and L_2 parts in the elastic net are assigned different scaling factors, we share the scaling parameter λ (called the *regularization constant*). This choice is motivated by a need to reduce the

complexity of the model selection task. Over a set $\mathcal{D}_M$ consisting of M data samples the full loss function is expressed as,

$$\mathcal{L}(\boldsymbol{\theta}; \lambda, \mathcal{D}_M) = \frac{1}{M} \sum_{i=1}^{M} \|y_i - \hat{f}(\boldsymbol{\xi}_i, \boldsymbol{\theta})\|^2 + \lambda \sum_{i=1}^{L+1} \Big(\|\mathbf{W}^{(i)}\|_2^2 + \|\mathbf{W}^{(i)}\|_1 \Big). \tag{3.8}$$

From the point of view of constrained optimization, normed weight penalties limit model complexity by shrinking the feasible region of parameters, $\boldsymbol{\theta}$, to their corresponding norm balls. There is also a Bayesian justification for the use of normed penalties on the weights. $\boldsymbol{\theta}^*$ obtained by the minimization of the regularized loss function corresponds to the maximum a posteriori (MAP) estimate of $\boldsymbol{\theta}$, with the prior given by the chosen penalty. The L_1 and L_2 penalties correspond to a Laplace and Gaussian priors on the weights while the elastic net represents a compromise between the two. In unnormalized form, the elastic net regularizer with equi-scaling of the L_1 and L_2 parts, correspond to the following prior on the weights:

$$p(\mathbf{W}) \propto \exp(\lambda \|\mathbf{W}\|_2^2 + \lambda \|\mathbf{W}\|_1). \tag{3.9}$$

3.5 Gradient computation and optimization

A DNN $\hat{f}(\boldsymbol{\xi}; \boldsymbol{\theta})$ is a highly complicated function of the network parameters $\boldsymbol{\theta}$ because of the fact that it involves multiple layers of compositions of simpler functions. To perform gradient descent optimization one needs access to the gradients of the objective function. For training DNNs, this is achieved by utilizing the celebrated *backpropagation* algorithm [161]. In essence, the backpropagation algorithm is a recursive application of the standard chain rule. Unlike numerical differentiation schemes such as finite differences, backpropagation is exact.

Note that training a DNN with subsets (i.e. minibatches) of the full dataset $\mathcal{D}$, is a stochastic optimization problem with the objective function being the loss function described in Eq. (3.8). The most common way of solving this problem is via the stochastic gradient descent (SGD) [162] algorithm. As the name suggests, SGD is the

stochastic analogue of deterministic gradient descent. The SGD algorithm produces a converging sequence of updates of the optimization variables, by making appropriately sized steps in the direction of the negative gradient of the objective function. The key idea of the SGD method, is that it approximates the negative gradient of the objective function by averaging a finite set of objective function gradient samples. This is done by independently sampling a small subset of examples, $\mathcal{D}_M$, from the full training dataset, $\mathcal{D}$. The update scheme of the SGD method is:

$$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha_k \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \lambda, \mathcal{D}_M). \tag{3.10}$$

Note that the sampling of $\mathcal{D}_M$ is performed at every iteration of SGD. While the SGD algorithm is simple to implement, it is not guaranteed to perform well for complex high dimensional objective functions (as is typical for Eq. 3.7). While there are multiple variants of the SGD method that have demonstrated improvements over vanilla SGD, in this work, we solve Eq. (3.7) with the Adaptive Moments (ADAM) optimization algorithm [122]. The ADAM update scheme is as follows:

$$\mathbf{M}_k \leftarrow \beta_1 \mathbf{M}_{k-1} + (1 - \beta_1)\mathbf{G}_k, \tag{3.11}$$

$$\mathbf{V}_k \leftarrow \beta_2 \mathbf{V}_{k-1} + (1 - \beta_2)\mathbf{G}_k^2, \tag{3.12}$$

$$\widehat{\mathbf{M}}_k \leftarrow \frac{\mathbf{M}_k}{1 - \beta_1^k}, \tag{3.13}$$

$$\widehat{\mathbf{V}}_k \leftarrow \frac{\mathbf{V}_k}{1 - \beta_2^k}, \tag{3.14}$$

$$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha_k \frac{\widehat{\mathbf{M}}_k}{\sqrt{\widehat{\mathbf{V}}_k} + \eta}, \tag{3.15}$$

where, $\mathbf{G}_k = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \lambda, \mathcal{D}_M)$ is the estimate of the objective function gradient at iteration k and $\mathbf{M}_k$ and $\mathbf{V}_k$ are exponential moving average estimates of gradients and squared gradients respectively. $\mathbf{M}_0$ and $\mathbf{V}_0$ are set to 0 and the bias introduced by this initialization is corrected by computing $\widehat{\mathbf{M}}_k$ and $\widehat{\mathbf{V}}_k$. η is a suitably small number introduced to prevent 0 denominator. β_1 and β_2 are averaging parameters

which can be tuned. In practice, default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$, as suggested by [122] work well and we do not fiddle with these quantities.

3.6 Selecting network structure

Although various authors in the literature offer rules of thumb for selecting the number and size of DNN layers (such as those suggested in [163]), rigorous rules for the selection of these quantities do not exist. One typically resorts to extensive experimentation to arrive at a suitable network configuration. In the most naive case, the number and size of the hidden layers are hyperparameters selected using cross-validation. In this work, we are interested in learning a surrogate of the form described in Eq. (3.3). The function h accepts an input in a vector space of dimensions D and projects it to a vector space of dimension d, where $d << D$ (d is to be determined through our methodology). We parameterize this section of the DNN such that the widths of it's hidden layers decays exponentially. Specifically, the number of hidden units in the k^{th} hidden layer in this section is given by:

$$d_k = \lceil D \exp(\rho k) \rceil, \tag{3.16}$$

where, $\lceil a \rceil$ represents the ceiling (closest greater integer) of the number a. The parameter ρ is uniquely determined, conditional on the knowledge of d and the total number of layers, L, in h. It can be computed by substituting $k = L$ and $d_k = d$ in the expression $d_k = \lceil D \exp(\rho k) \rceil$. The link function g is formulated as a single layer MLP. The hidden layer in g is taken to have a width of $300d$. One could set this width to be anywhere between $100 - 500$ times the size of the encoding d. This would be an additional hyperparameter which we choose not to tune in the present work, to minimize our computational burden. The idea is that the subnetwork representing the link function g ought to have a sufficient number of hidden units to capture arbitrary nonlinearities. A visual representation of the DNN surrogate is shown in Fig. 3.3.

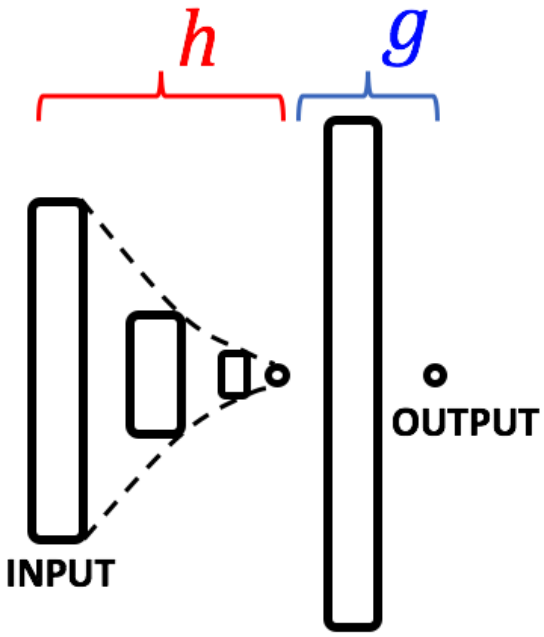


Figure 3.3. Visualization of the parameterized network structure with $L = 3$ and $d = 1$.

Note that no activation function is used at the output of the encoding subnetwork h, and the output of the link function subnetwork, g. The task of optimizing the network structure is then reduced to a task of cross validating over two integer quantities, L and d, a much simpler task than optimizing for the number of layers and sizes of the individual layers separately.

3.7 Combined global optimization and grid search for model selection

The stochastic optimization task stated in Eq. (3.7) is characterized by hyperparameters, weight decay λ, and the integer quantities L and d, which fully parameterize the structure of the network. We refer to structure parameters collectively, as, $\mathcal{S} = (L, d)$. Training a DNN involves, in addition to optimizing for $\boldsymbol{\theta}$, selection of appropriate values of hyperparameters. The naive approach to do this is to perform an intuition guided manual search. In this work, the task of model selection reduces to selecting 3 quantities - the discrete hyperparameters, L and d and the continuous hyperparameter, λ. To be systematic, we adopt a combined grid search and stochastic global optimization procedure. Specifically, we define a grid of values for L and d. Over each grid location of the structure parameters, we perform a Bayesian global optimization (BGO) [35, 164] for λ.

We split the dataset $\mathcal{D}$ into 3 parts - a training set, $\mathcal{D}_{\text{train}}$, a validation set, $\mathcal{D}_{\text{val}}$ and a test set, $\mathcal{D}_{\text{test}}$. We define a grid, $\mathcal{G}$ [3], of L and d values and seek to assign a score to each location on the grid. The optimal choice of the structure parameters, $\mathcal{S}$ would then be the grid location which minimizes the validation error:

$$\mathcal{R}(\mathcal{S}, \lambda) = \frac{1}{M_{\text{val}}} \sum_{i=1}^{M_{\text{val}}} (y_{\text{val},i} - \hat{f}_{\mathcal{S}}(\boldsymbol{\xi}_{\text{val},i}; \boldsymbol{\theta}^*_{\mathcal{S}}(\lambda)))^2, \tag{3.17}$$

[3] The use of the term 'grid' to denote the set of hyperparameters of a model is standard within machine learning literature. The reader, however, might be more familiar with the term 'lattice' to denote such a set.

where, $(\boldsymbol{\xi}_{\text{val},i}, y_{\text{val},i}) \in \mathcal{D}_{\text{val}}$, M_{val} is the size of the validation set, and $\hat{f}_{\mathcal{S}}$ is a DNN characterized by structure parameter, $\mathcal{S}$. $\boldsymbol{\theta}^*_{\mathcal{S}}(\lambda)$ is an estimate of the network parameters, $\boldsymbol{\theta}$ obtained by minimizing the loss function in Eq. (3.8), with the regularization constant set to λ and network structure parameter, $\mathcal{S}$.

The optimal choice of regularization constant λ, corresponding to structure parameter, $\mathcal{S}$, is:

$$\lambda^*_{\mathcal{S}} = \underset{\lambda}{arg\,min}\ \mathbb{E}[\mathcal{R}(\mathcal{S}, \lambda)]. \tag{3.18}$$

Eq. (3.18) is a stochastic global optimization problem characterized by a noisy objective function. BGO sequentially seeks out the global optimum of the objective function, $\mathcal{R}$, by iteratively updating a Gaussian process (GP) surrogate response surface for $\mathcal{R}(\lambda, \mathcal{S})$. During each iteration of BGO, a new pair of input-output observations is generated by maximizing an information acquisition function (IAF). The most popular choice of IAF is the expected improvement (EI) function. In closed form, the EI-IAF is given by:

$$\text{EI}(\lambda) = \begin{cases} (\mu(\lambda) - \mathcal{R}(\lambda^*, \mathcal{S}))\Phi(Z) + \sigma(\lambda)\phi(\lambda) \text{ if } \sigma(\lambda) > 0, \\ 0 \text{ if } \sigma(\lambda) \leq 0, \end{cases} \tag{3.19}$$

where, ϕ and Φ are the probability density function and the cumulative distribution function of the standard normal distribution. $Z = \frac{\mu(\lambda) - \mathcal{R}(\lambda^*, \mathcal{S})}{\sigma(\lambda)}$ where, $\mu(\lambda)$ is the predictive mean of the GP surrogate at λ, and $\sigma(\lambda)^2 = \sigma_{\text{GP}}(\lambda)^2 - \sigma_{\text{noise}}(\lambda)^2$, where $\sigma_{\text{GP}}(\lambda)^2$ is the predictive variance of the GP surrogate which captures the epistemic uncertainty induced due to the limited set of observations and $\sigma_{\text{noise}}(\lambda)^2$ is GP estimate of the observational noise induced by random initializations of the DNN weights and random splitting of the dataset into $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{test}}$ and $\mathcal{D}_{\text{val}}$. $\sigma(\lambda)^2$ is thus a filtered version of the predictive variance which is robust to observational noise. The BGO algorithm is summarized in Alg. 3. Note that the we maximize the negative of the validation error $\mathcal{R}$. In the present work, the GP surrogate is chosen to have zero prior mean and the Matern function with $\nu = \frac{3}{2}$ (See Eq. (4.17) of [42]) as the prior

covariance kernel. The expressions for the predictive mean and variance of the GP surrogate, i.e., $\mu(\cdot)$ and $\sigma(\cdot)^2$ can be found in Eq. (2.25) and (2.26) in [42].

Algorithm 3 Bayesian global optimization of validation error $\mathcal{R}(\lambda, \mathcal{S})$

Require: Training data, $\mathcal{D}_{\text{train}}$, validation data, $\mathcal{D}_{\text{val}}$, structure parameter, $\mathcal{S}$, number of initial observations, n_{init}, number of BGO iterations, $maxiter$, bounding box for λ, $\mathcal{B}$.

1: Initialize empty arrays, Λ_{BGO} and $\mathcal{R}_{\text{BGO}}$.
2: Use Latin Hypercube sampling (LHS) [165] to generate n_{init} samples of λ within the bounding box $\mathcal{B}$. Call it Λ_{init}.
3: **for** $\lambda_i \in \Lambda_{\text{init}}$ **do**
4: Minimize $\mathcal{L}$ from Eq. (3.8) with training data, $\mathcal{D}_{\text{train}}$ to obtain $\boldsymbol{\theta}_{\mathcal{S}}(\lambda_i)$.
5: Evaluate $\mathcal{R}_i$ = -$\mathcal{R}(\lambda_i; \mathcal{S})$.
6: Append λ_i and $\mathcal{R}_i$ to Λ_{BGO} and $\mathcal{R}_{\text{BGO}}$ respectively.
7: **end for**
8: Fit a GP surrogate linking Λ_{BGO} and $\mathcal{R}_{\text{BGO}}$.
9: **for** iter = 1 to $maxiter$ **do**
10: Get next sample of λ, $\lambda_{n_{\text{init}}+iter} = \underset{\lambda}{arg\,max} EI(\lambda)$.
11: Evaluate $\mathcal{R}_{n_{\text{init}}+iter} = -\mathcal{R}(\lambda_i; \mathcal{S})$.
12: Append $\lambda_{n_{\text{init}}+iter}$ and $\mathcal{R}_{n_{\text{init}}+iter}$ to Λ_{BGO} and $\mathcal{R}_{\text{BGO}}$ respectively.
13: **end for**
14: Get $index = \arg\max \mathcal{R}_{\text{BGO}}$.
15: **return** $\lambda^*_{\mathcal{S}} = \Lambda_{\text{BGO}}(index)$. Return λ corresponding to the highest observed negative validation error.

Finally, the optimal structure parameter, $\mathcal{S}^*$ is given by:

$$\mathcal{S}^* = \underset{S \in G}{arg\,min}\ \mathcal{R}(\mathcal{S}, \lambda^*_{\mathcal{S}}). \tag{3.20}$$

The full algorithm is summarized in Alg. 4. The estimation of $\boldsymbol{\theta}^*_{\mathcal{S}}$ for each individual $\mathcal{S}$ can be parallelized and the computational cost of the global optimization search for $\lambda^*_{\mathcal{S}}$ requires $maxiter$+n_{init} times the cost of a single run of the ADAM optimizer. The cost of a single iteration in the ADAM optimization grows linearly with the total number of weights in the network. To see this, recall that a single iteration comprises of a forward and backward pass through the network. Since the expressions for the gradients in each layer of the network are analytically available, evaluation of the gradients incur

a negligible cost in the overall execution of an optimization step. The computational cost of one iteration is, thus, dominated by the series of matrix multiplications in the forward propagation of the input and the backpropagation of the parameter gradients. The complexity of multiplying $m \times n$ and $n \times p$ matrices is given by $\mathcal{O}(mnp)$. Thus, the cost of executing a single forward-backward pass in a DNN with the proposed architecture is proportional to $M(300d(d+1) + \sum_{k=1}^{L} d_{k-1}d_k) = MN_{\text{weights}}$, where M is the size of the data mini-batch and N_{weights} is the total number of weights in the network. d_k is obtained from Eq. 3.16. Note that the network biases have a negligible effect on the computational cost of forward and backward propagation in the network. The effect of the input dimensionality D, and the structure parameters, L and d, on a single iteration of the ADAM optimization can be estimated by observing the growth of N_{weights} as a function of D, L and d, which is plotted in Fig. 3.4.

Algorithm 4 Full procedure for training DNN surrogate

Require: Data, $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, grid of L and h values, $\mathcal{G}$, parameters for Alg. 3, n_{iter}, *maxiter* and $\mathcal{B}$.

1: Split $\mathcal{D}$ into 3 parts - $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}$ and $\mathcal{D}_{\text{val}}$.
2: **for** $\mathcal{S} \in \mathcal{G}$ **do**
3: Set $\lambda^*_{\mathcal{S}} \leftarrow \underset{\lambda}{arg\,min}\ \mathbb{E}[\mathcal{R}(\mathcal{S}, \lambda)]$. Using Alg. 3.
4: Set $\boldsymbol{\theta}^*_{\mathcal{S},\lambda} \leftarrow \boldsymbol{\theta}^*_{\mathcal{S}}(\lambda^*_{\mathcal{S}})$.
5: Set $\mathcal{R}_{\mathcal{S}} \leftarrow \mathcal{R}(\mathcal{S}, \lambda^*_{\mathcal{S}}; \boldsymbol{\theta}^*_{\mathcal{S},\lambda})$. Estimate of the global minima of $\mathcal{R}$ for DNN with structure parameter $\mathcal{S}$.
6: **end for**
7: Set $\mathcal{S}^* \leftarrow \underset{S \in G}{arg\,min}\ \mathcal{R}_{\mathcal{S}}$. Get the structure parameter that minimizes the observed validation error.
8: **return** $\mathcal{S}^*, \lambda^*_{\mathcal{S}^*}, \boldsymbol{\theta}^*_{\mathcal{S}^*,\lambda^*_{\mathcal{S}^*}}$. Final DNN surrogate.

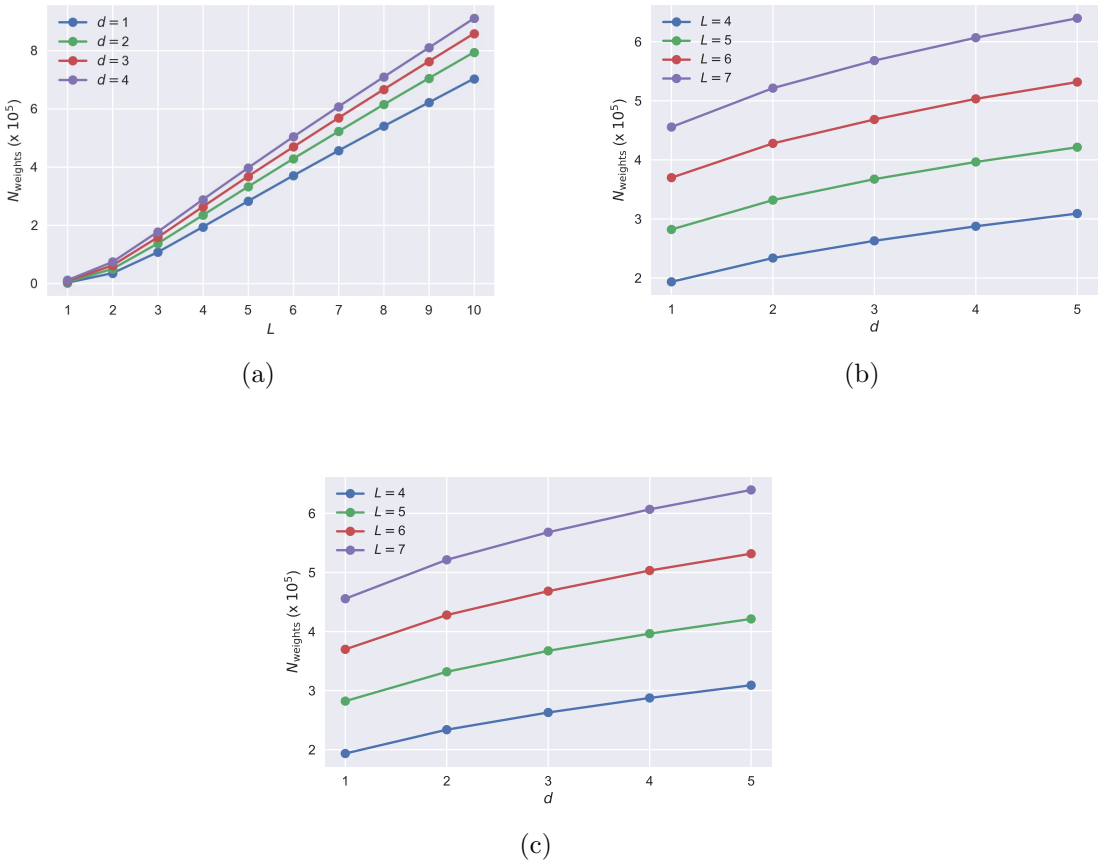


Figure 3.4. Growth of the number of network weights, N_{weights}, as a function of the input dimensionality D and structure parameters, L and d. 3.4(a) - Growth of N_{weights} as a function of L for various d, with D set to 1026. 3.4(b) - Growth of N_{weights} as a function of D for various L, with d set to 2. 3.4(c) - Growth of N_{weights} as a function of d for various L, with D set to 1026.

3.8 Numerical example

We consider the following benchmark elliptic PDE on the 2-d unit square domain:

$$-\nabla(a(\mathbf{x})\nabla u(\mathbf{x})) = 0,\ \forall \mathbf{x} \in [0,1]^2, \tag{3.21}$$

with boundary conditions:

$$u = 0,\ \forall x = 1, \tag{3.22}$$

$$u = 1,\ \forall x = 0, \tag{3.23}$$

$$\frac{\partial u}{\partial n} = 0,\ \forall y = 0 \text{ and } y = 1, \tag{3.24}$$

where, $\mathbf{x} = (x, y)$ are the physical coordinates in 2-d Euclidean space.

Eq. (3.21) is a model for 2-d steady-state diffusion processes. The quantity $a(\mathbf{x})$ is a spatially varying diffusion coefficient. The physical significance of the equation and all terms in it are derived from context. For instance, Eq. (3.21) could be an idealized model for single-phase groundwater flow in an aquifer [166], where a represents the transmissivity coefficient and the solution variable, u is the pressure.

It is often the case that the a is unknown throughout the PDE domain. The uncertainty in the diffusion coefficient is formalized by modeling a as a log normal random field, i.e.,

$$\log a(\mathbf{x}) \sim \mathrm{GP}(a(\mathbf{x})|m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{3.25}$$

where, $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ are the mean and covariance functions, respectively, of the Gaussian random field which models the logarithm of the diffusion coefficient $a(\mathbf{x})$. The mean function models beliefs about the generic trends of the diffusion field as a function of spatial location. For the sake of simplicity, we set $m(\mathbf{x}) = 0$ in this example. The covariance function k models beliefs about the regularity of the diffusion field and the lengthscales in which it varies. A popular choice for k is the exponential kernel:

$$k(\mathbf{x}, \mathbf{x}') = \exp\Big(- \sum_{i=1}^{2} \frac{|x_i - x_i'|}{\ell_i} \Big), \tag{3.26}$$

where ℓ_i represents the correlation length along the i^{th} spatial direction. The correlation lengths are typically assigned a fixed value. One then proceeds to use the truncated KLE to produce a reduced representation of the infinite-dimensional random field. The coefficients of the KLE are i.i.d. standard normal, and realizations of the diffusion field, a, can be generated easily, i.e., by sampling the KLE coefficients. For each realization of a, the corresponding solution of Eq. (3.21) is obtained. Any relevant quantity of interest, $q = \mathcal{Q}[u]$, is computed. Finally, one learns a surrogate response surface that maps the coefficients of the truncated KL expansion, using suitable learning algorithms such as GP regression.

We broaden the scope of the problem by removing the restrictions on the lengthscales, ℓ_i. The goal, in this example, is to construct a surrogate, which can accurately predict the solution, u, of the PDE, regardless of the lengthscales of the realization of a. Our approach, then, is to construct a surrogate which directly maps the discretized random field, to the numerical solution of the PDE.

3.8.1 Forward model

We solve the PDE using the finite volume method (FVM). The solver is implemented in the `Python` library `FiPy` [167]. The unit square domain is discretized into 32×32 finite volume cells. The input to the solver, $\hat{\mathbf{a}} \in \mathbb{R}^{32\times32}$ is the discretized version of a sample of the random diffusion a. The output of the solver, $\hat{\mathbf{u}} \in \mathbb{R}^{32\times32}$, is the numerical solution of the PDE corresponding to the realization $\hat{\mathbf{a}}$ of the diffusion field.

The model inputs, $\boldsymbol{\xi} = (\mathrm{vec}(\hat{\mathbf{a}}), \mathbf{x}) \in \mathbb{R}^{1026}$ are spatial coordinates appended to a flattened version of the discrete random field realization and the model outputs are the PDE solution at the FV cell centers, $u(\mathbf{x}, \hat{\mathbf{a}})$. Our goal is to learn a surrogate

response, $\hat{f} : \mathbb{R}^{1024} \times [0,1]^2 \to \mathbb{R}$, which maps the snapshot of the diffusion field and a particular coordinate in the unit square to the solution of the PDE at that location.

3.8.2 Data Generation

Intuitively, we would like to sample more realizations, of a, that have smaller lengthscales because one would observe the most variability in the solution corresponding to low lengthscale diffusion fields. Thus, instead of sampling lengthscale pairs uniformly from the unit square, we bias our sampling procedure to pick lengthscales that are smaller. Alg. 5 describes the procedure to select lengthscales to train the DNN surrogate. Note that a lower bound on ℓ_i is set by constraining the lengthscale to be larger the FV cell size, $h(= \frac{1}{32})$.

Algorithm 5 Sampling of lengthscale pairs.

Require: Number of lengthscale pairs, n, lower bound on lengthscale, h.

Initialize n-dimensional empty array $\mathbf{L}$.
Initialize counter, $c = 1$.
while $c \leq n$ **do**
  Sample $u = (u_1, u_2, u_3) \sim \mathcal{U}([0,1]^3)$. $\mathcal{U}(\mathcal{A})$ is the uniform distribution over the set $\mathcal{A}$.
  if $\exp(-u_1 - u_2) < u_3$ **then**
    Set $\ell_c = (\ell_{x,c}, \ell_{y,c}) = (h + u_1(1-h), h + u_2(1-h))$. Scale the sampled lengthscales to the range $[h, 1]$.
    Set $\mathbf{L}_c \leftarrow \ell_c$.
    Increment counter $c \leftarrow c + 1$.
  end if
end while
return $\mathbf{L}$.

We generate n different lengthscale pairs following the procedure in Alg. 5 to obtain a design of lengthscale pairs $\mathbf{L}$.

For each lengthscale pair, $(\ell_x, \ell_y) \in \mathbf{L}$, we solve the forward model N times by generating N realizations of the diffusion coefficient. In this example we set $n = 60$ and $N = 100$. Note that the choice of n and N are not fixed. One could potentially

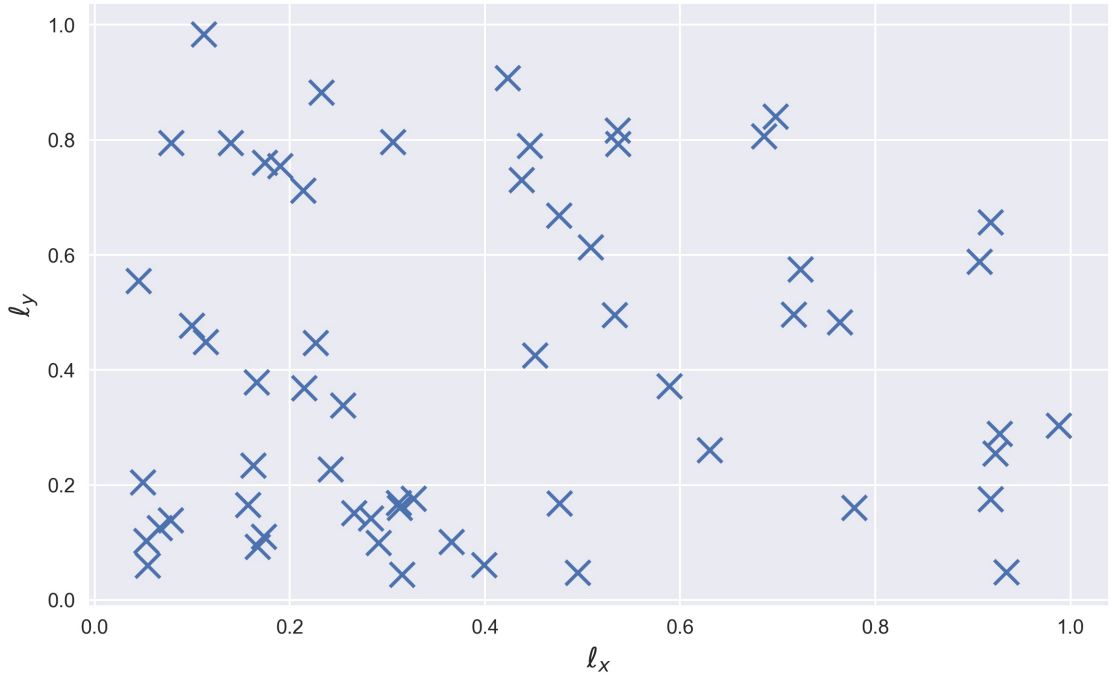


Figure 3.5. Visual representation of LHS design of lengthscale pairs. Each 'x' represents a sampled pair of lengthscales.

shrink the requisite size of the total dataset through a smart choice of n and N. Furthermore, one could even come up with adaptive strategies to select the number of evaluations per lengthscale pair. Exploration of these data generating strategies are, however, beyond the scope of the present work. A visual representation of $\mathbf{L}$ is shown in Fig. 3.5. Samples of the diffusion coefficient drawn from two different pairs of lengthscales are shown in Fig. 3.6 and 3.7.

3.8.3 Numerical settings

Dataset split

We generated our dataset, $\mathcal{D}$, of $n \times N = 60 \times 100 = 6000$ pairs of $\hat{a}$ and $\hat{\mathbf{u}}$. $\mathcal{D}$ is randomly shuffled and split into 3 parts - A set of 2000 training examples, $\mathcal{D}_{\text{train}}$, a set of 2000 validation examples, $\mathcal{D}_{\text{val}}$ and a set of 2000 test examples, $\mathcal{D}_{\text{test}}$. For

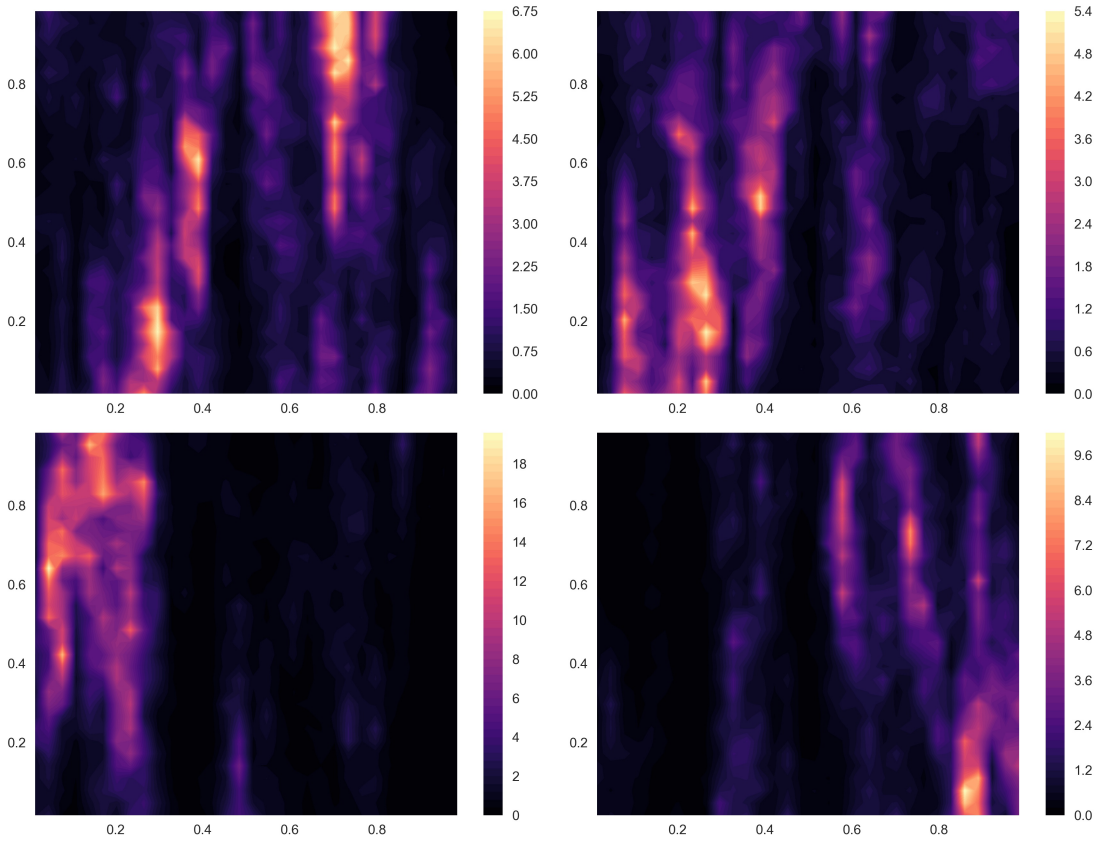


Figure 3.6. Samples of the random field $a(\mathbf{x})$ with lengthscales $\ell_x = 0.446$ and $\ell_y = 0.789$ along the x and y directions.

the purpose of constructing the surrogate, we work with the logarithm of both $\hat{\mathbf{a}}$ and $\hat{\mathbf{u}}$ during training [4]. Furthermore, $\hat{\mathbf{u}}$ in the training set is standardized along each dimension, i.e., scaled to have dimension-wise zero mean and unit variance. Necessary inversions of the transformations are performed during test time.

3.8.4 Model selection settings

For selection of $\lambda^*_{\mathcal{S}}$ using Alg. 3, we set the number of initial design points, $n_{\text{init}} = 5$. The number of BGO iterations, $maxiter = 10$ and the bounding box,

[4]Note that the use of the logarithmic transform is valid because the response $\hat{\mathbf{u}}_i > 0$, $\forall i$. Appropriate preprocessing of the data must be perform conditional on the properties of the data.

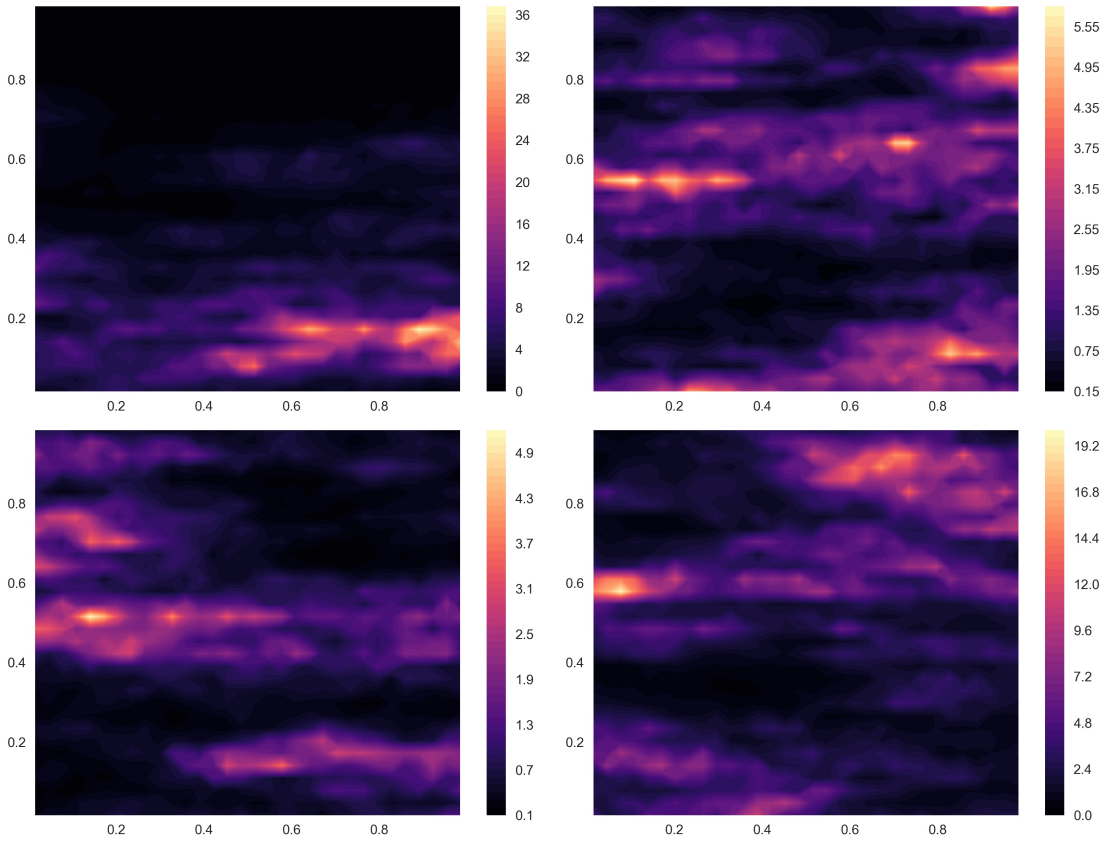


Figure 3.7. Samples of the random field $a(\mathbf{x})$ with lengthscales $l_x = 0.291$ and $l_y = 0.099$ along the x and y directions.

$\mathcal{B} = [10^{-10}, 10^{-3}]$. The grid of structure parameters is set to be $\mathcal{G} = \{3, 4, 5, 6, 7, 8, 9\} \times \{1, 2, 3, 4\}$.

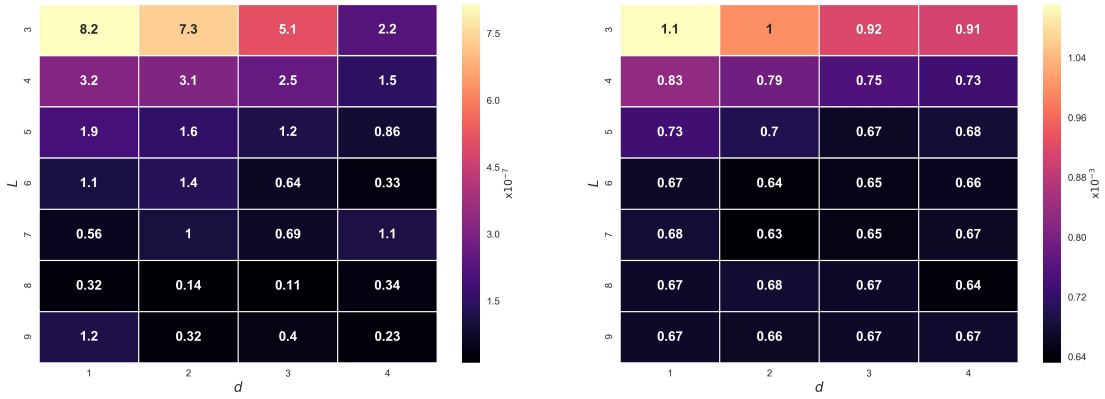
Network optimizer settings

We set the ADAM optimization learning rate α to be 1×10^{-3}. The optimizer is run for 45000 iterations and α is decreased by a factor of 0.1 every 15000 iterations. The batch size, M, is set to be 50. Default values of tunable parameters of the ADAM optimizer are used, as recommended in [122]. These settings are, by no means, universal. Refer to [163] for some practical guidelines on DNN hyperparameter selection.

We use the `Python` library `tensorflow` to write scripts for training our DNN surrogates. For the purpose of reproducibility, the `NumPy` pseudo-random number generator seed is fixed. The code to replicate all the results are available at https://github.com/rohitkt10/deep-uq-paper.

3.8.5 Results

Fig. 3.8 shows a heatmap of λ and optimal validation error $\mathcal{R}_{\mathcal{S}}$ over the grid of the structure parameters. We observe that for the chosen grid, the optimal structure parameter is found to be $\mathcal{S}^* = (7, 2)$ and $\lambda^*_{(7,2)} = 1.043 \times 10^{-7}$. Fig. 3.9 shows GP surrogate response for $-\log \mathcal{R}$ as a function of $\log \lambda$, for $\mathcal{S} = (7, 2)$. Observe, from Fig. 3.9 that there is a dense clustering of the 'x' markers around the optimum, indicating the convergence of the sequential optimization process. Fig. 3.10 shows a scatterplot of $\boldsymbol{\zeta} = (z_1, z_2) = h(\boldsymbol{\xi})$, the low-dimensional embedding of the inputs, corresponding to 5 arbitrarily chosen input lengthscales from the test dataset, $\mathcal{D}_{\text{test}}$. Fig. 3.11 shows a visualization of the link function $g(\boldsymbol{\zeta})$ for 4 randomly selected examples from the test dataset. It is interesting to note that inputs sampled from the different lengthscale pairs gather around numerous non-segregated clusters in the low-dimensional manifold.



(a) (b)

Figure 3.8. 3.8(a) - Heatmap of $\lambda^*_{\mathcal{S}}$ over the grid $\mathcal{G}$. 3.8(b) - Heatmap of $\mathcal{R}_{\mathcal{S}}$ over the grid $\mathcal{G}$.

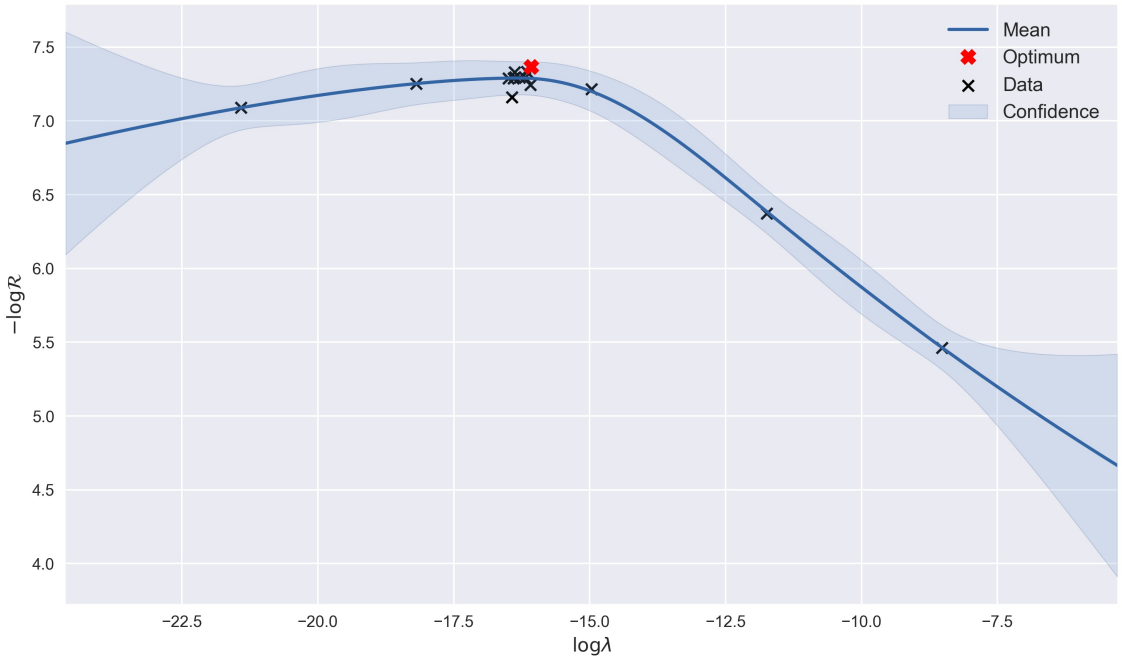


Figure 3.9. Gaussian process surrogate generated during BGO. We maximize the negative of the validation error $\mathcal{R}$ as a function of the logarithm of the regularization parameter, λ.

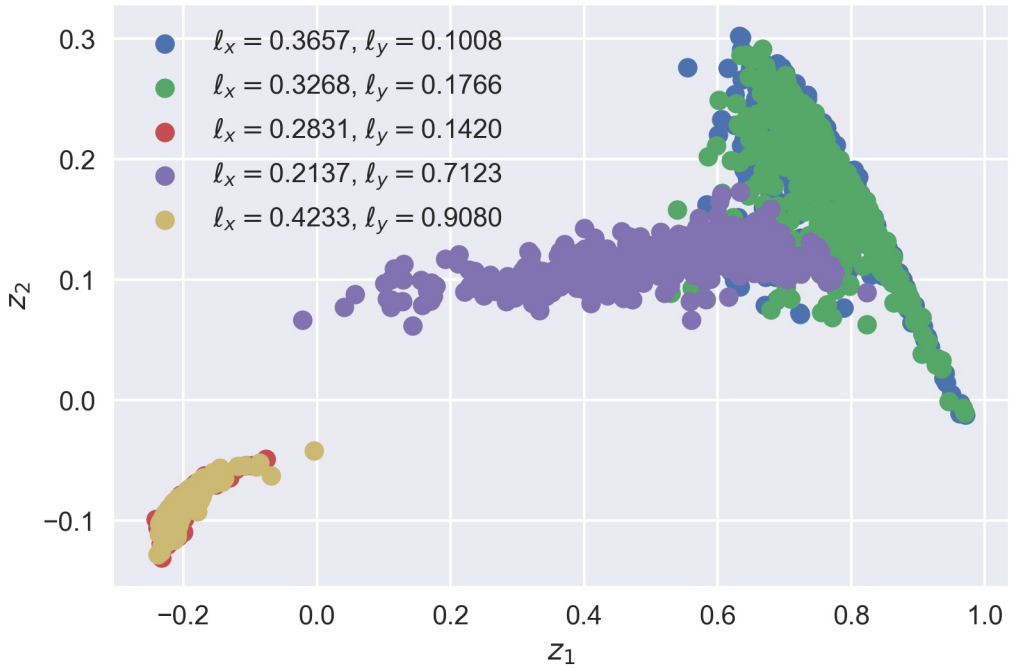


Figure 3.10. Scatter plot of low-dimensional embedding of the input diffusion fields from different lengthscales.

The quality of the DNN predictions are evaluated based on the following relative error metric:

$$\mathcal{E}(\hat{\mathbf{a}}) = \frac{\|\hat{\mathbf{u}}_{\text{DNN}} - \hat{\mathbf{u}}_{\text{FV}}\|_{\text{F}}}{\|\hat{\mathbf{u}}_{\text{FV}}\|_{\text{F}}}, \tag{3.27}$$

where, $\|\cdot\|_{\text{F}}$ is the Frobenius norm. $\hat{\mathbf{u}}_{\text{FV}}$ and $\hat{\mathbf{u}}_{\text{DNN}}$ are the FVM PDE solution and the DNN prediction of the PDE solution corresponding to the realization $\hat{\mathbf{a}}$ of the diffusion field. We also check the coefficient of determination, (also known as the R^2 score), which is defined as:

$$R^2 = 1 - \frac{\sum_{k=1}^{1024}(\hat{\mathbf{u}}_{\text{FV},k} - \hat{\mathbf{u}}_{\text{DNN},k})^2}{\sum_{k=1}^{1024}(\hat{\mathbf{u}}_{\text{FV},k} - \bar{\mathbf{u}}_{\text{FV},k})^2}, \tag{3.28}$$

where, k indexes all the FV cell centers, $\hat{\mathbf{u}}_{\text{FV},k}$ and $\hat{\mathbf{u}}_{\text{DNN},k}$ are the FV solution and DNN predicted solution at the k^{th} cell center respectively, and $\bar{\mathbf{u}}_{\text{FV}}$ is the mean of $\hat{\mathbf{u}}_{\text{FV}}$. Fig. 3.12 shows a comparison of the DNN predicted PDE solution solution

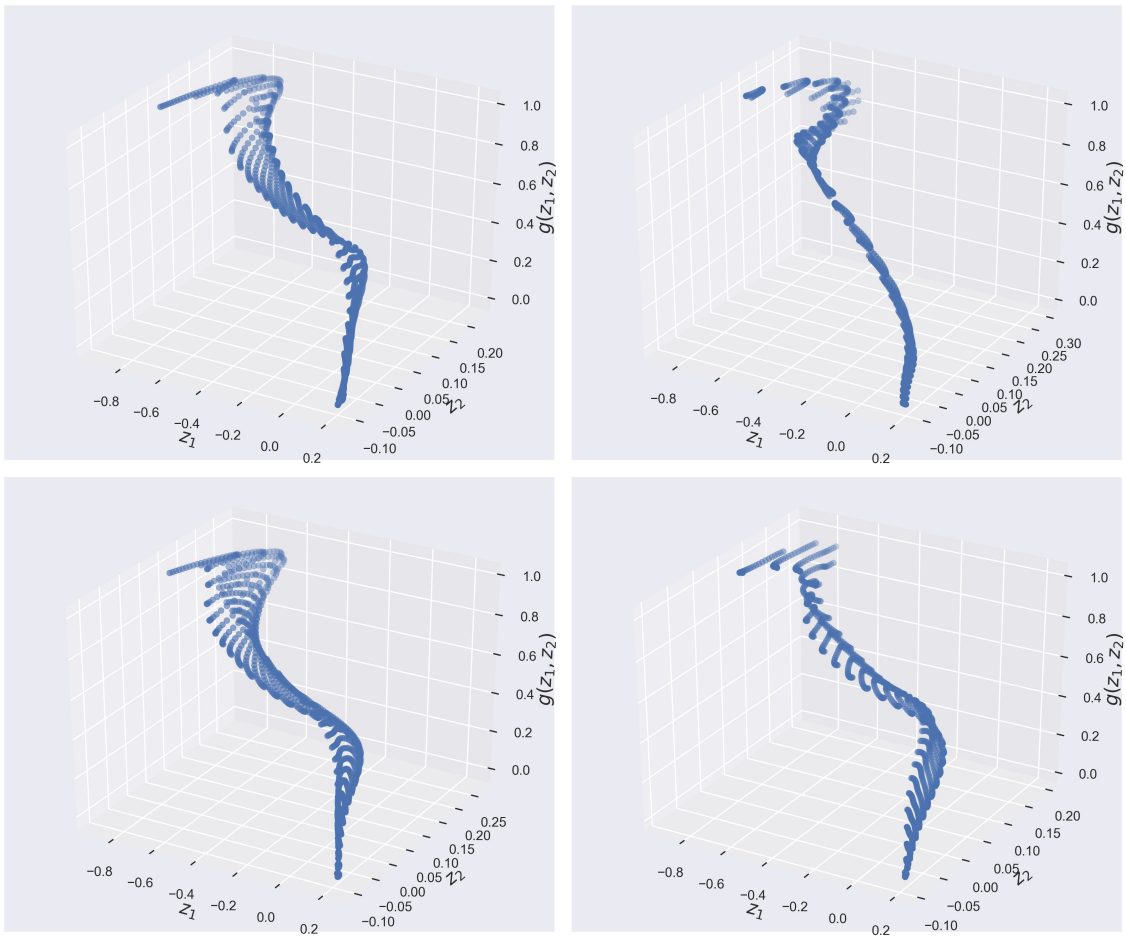


Figure 3.11. Link function, $g(\boldsymbol{\zeta})$ for 4 randomly selected examples from $\mathcal{D}_{\text{test}}$.

corresponding to a few randomly chosen realizations of the diffusion field from $\mathcal{D}_{\text{test}}$. We observe that the relative error as reported on the headers of the predicted fields in Fig. 3.12 are less than 0.1 and the R^2 scores close to 0.99, which implies that the predicted solution from the DNN matches the true very closely. We also note that the PDE solution predicted by the DNN is 'smoother' than the FV solution of the PDE. This effect gets more pronounced when the lengthscales of the input diffusion field are smaller. The smoothness is a consequence of regularizing the DNN loss function. Fig. 3.13 shows the histograms of $\mathcal{E}$ and R^2 scores for all samples in $\mathcal{D}_{\text{test}}$. Note that all testing of the predictive capacity of the network is done using the test set $\mathcal{D}_{\text{test}}$ because the $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$ have already been used during the training and model selection phase.

3.8.6 Predictions at arbitrary lengthscales

Having constructed a DNN surrogate for the FV solution of the PDE, we would like to test predictive accuracy for samples of $\hat{\mathbf{a}}$ with lengthscales which are not used to generate the dataset $\mathcal{D}$. A 10×10 uniform grid of lengthscales is generated in the domain $[h, 1]^2$, and for each lengthscale, 100 observations of the diffusion field and it's corresponding PDE solution are generated. The mean of the relative errors and mean of the R^2 scores for each lengthscale pair in this uniform grid is computed and shown in Fig. 3.14. We observe that even when the input field has lengthscales that do not match the lengthscales used for training, we are able to predict the solution with accuracy similar to that obtained during testing with samples from $\mathcal{D}_{\text{test}}$. This suggests that DNN surrogate is learning to map the 'picture' of the input field to the corresponding output. Note that the accuracy of the DNN decreases for diffusion fields with very fine lengthscales. This is consistent with the intuitive expectation that

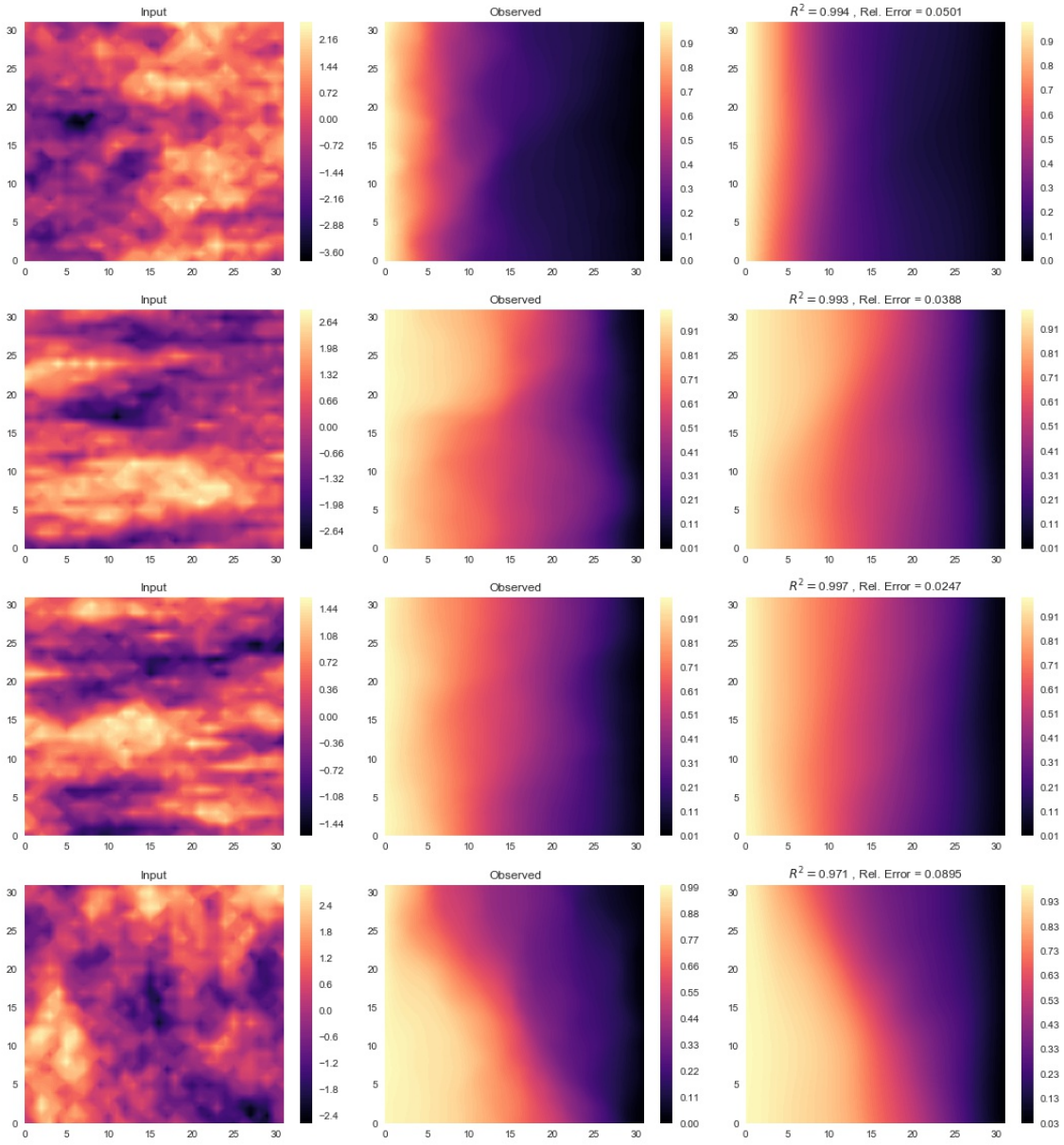
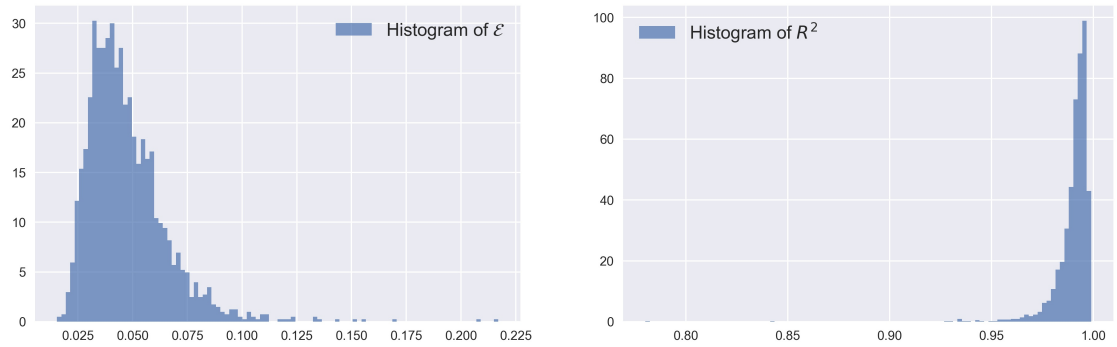


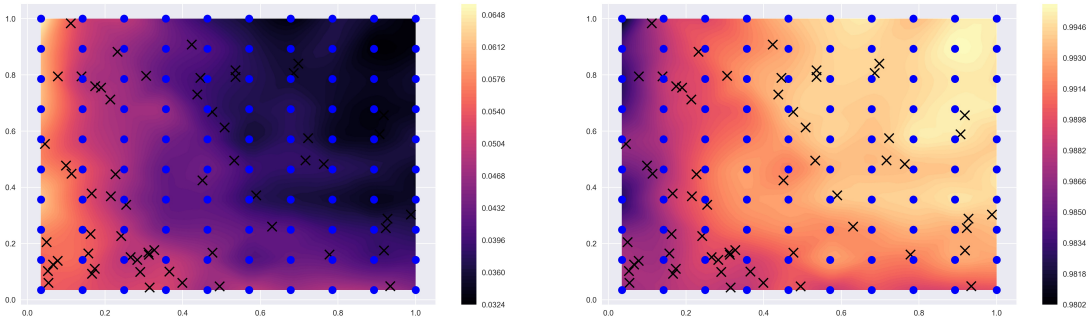
Figure 3.12. Comparisons of DNN prediction of the PDE solution to that correct solution for 4 randomly chosen test examples. The left column shows the logarithm of the input diffusion field, the middle column shows the FV solution of the PDE and the right column shows the solution of the PDE predicted by the DNN.

lesser the "variation" in the structure of the diffusion field, the easier it is characterize the function that maps the input to the solution.



(a) (b)

Figure 3.13. 3.13(a) - Histogram of relative errors, $\mathcal{E}$, for all examples in the test data set. 3.13(b) - Histogram of the R^2 scores for all examples in the test data set.



(a) (b)

Figure 3.14. 3.14(a) - Mean relative errors of the predicted solution corresponding to samples of a with arbitrary pairs of lengthscales not used in the DNN training. 3.14(b) - Mean R^2 scores of the predicted solutions corresponding to samples of a with arbitrary pairs of lengthscales not used in the DNN training. The 'x' markers correspond to lengthscales used in training the DNN and the solid dots correspond to lengthscales used to test the DNN surrogate.

3.8.7 Effect of dataset size

Intuitively, one would expect the accuracy of the DNN surrogate to improve as the size of the training dataset increases. To verify this proposition we construct a DNN surrogate when the training set has $N = 500, 750, 1000, 1250, 1500$ and 2000 samples. In each case, we follow the procedure outlined in Alg. 4 with the same model selection settings outlined in Sec. 3.8.4. Table 3.1 shows the optimal structure parameters corresponding to each N. It is worth noting that the size of the DNN is

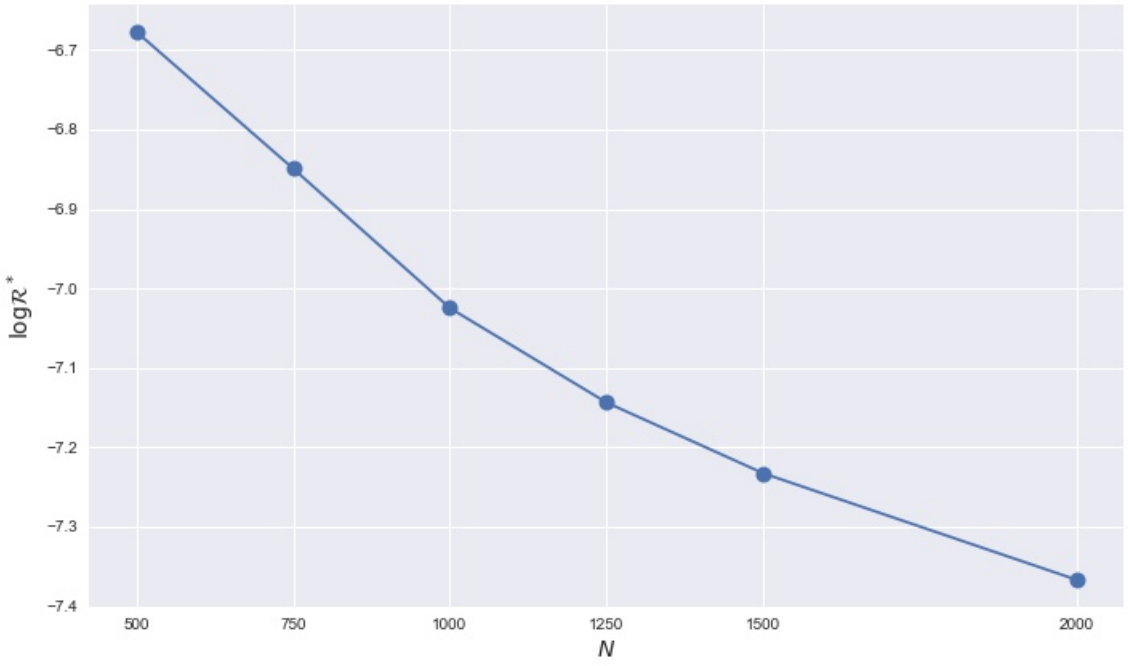


Figure 3.15. Variation of the log validation error, $\log \mathcal{R}$, corresponding to optimal structure estimated for different sizes of training datasets.

adaptive to the size of the training dataset. The decay of the optimal validation error with increasing training dataset size is visualized in Fig. 3.15.

Table 3.1. Optimal validation error, $\mathcal{R}^*$, and structure parameter, $\mathcal{S}^* = (L^*, d^*)$ corresponding to different sizes of the training dataset.

N	$\mathcal{R}^*_{\mathcal{S}^*}(\times 10^{-3})$	L^*	d^*
500	1.25965	8	4
750	1.06043	9	3
1000	0.88968	8	2
1250	0.78989	9	1
1500	0.72258	7	1
2000	0.63191	7	2

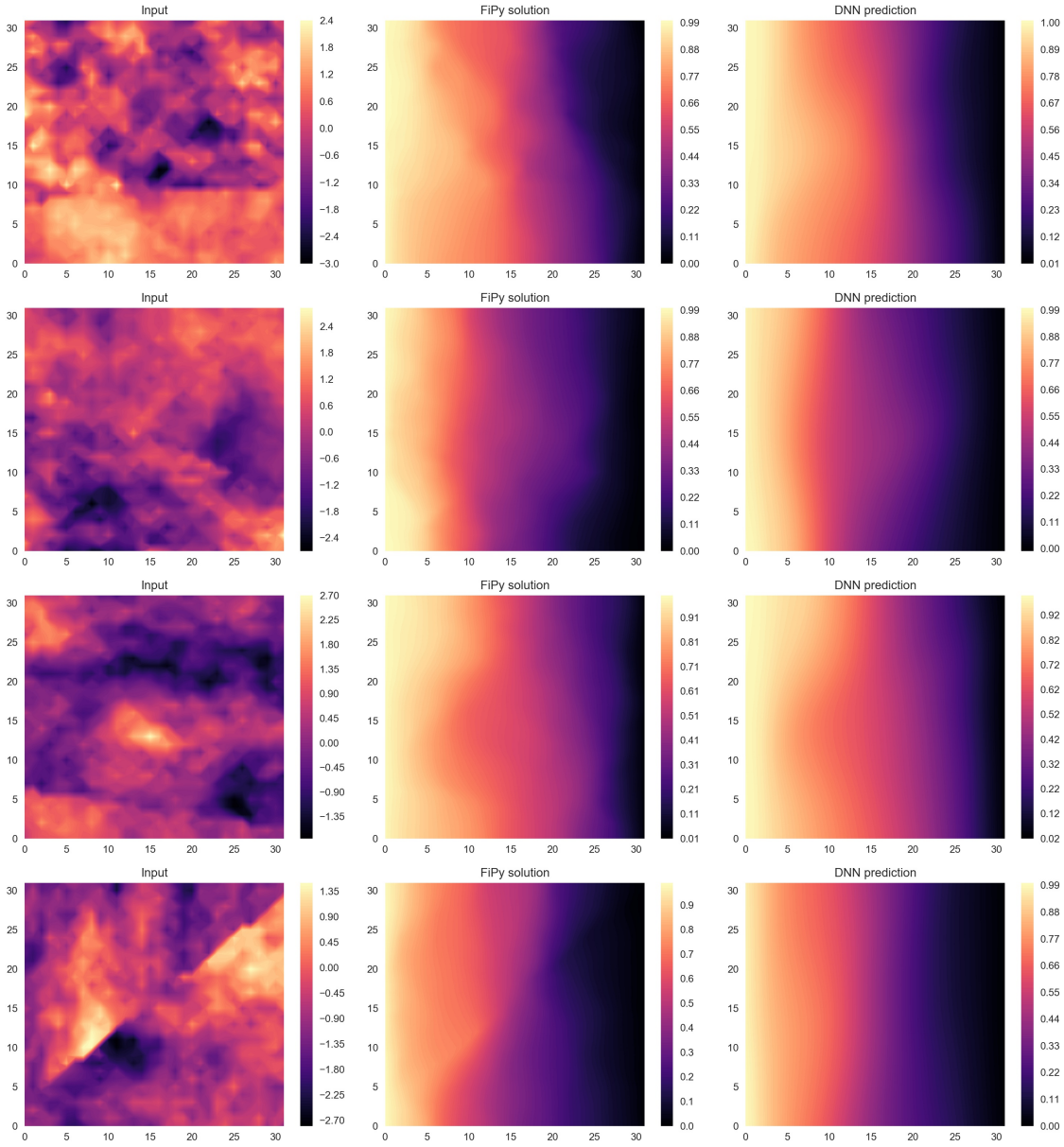


Figure 3.16. Comparisons of DNN prediction of the PDE solution to that correct solution for 4 randomly chosen stratified diffusion fields. The left column shows the logarithm of the input diffusion field, the middle column shows the FV solution of the PDE and the right column shows the solution of the PDE predicted by the DNN.

3.8.8 Predictions with stratified diffusion fields

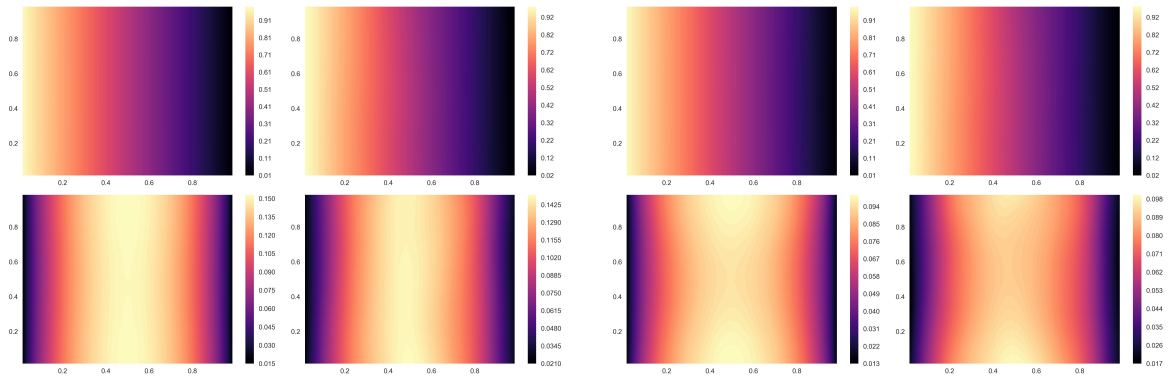
We now use the DNN surrogate to test it's predictive capability with stratified (or layered) diffusion fields. The construction of the stratified diffusion field is as follows - 1. Sample m_1 and m_2 independently from $\mathcal{U}([0,1])$; 2. Divide the input domain, $[0,1]^2$, into two regions, Ω_1 and Ω_2, separated by the line joining $\mathbf{x}_{m_1} = (0, m_1)$ and $\mathbf{x}_{m_2} = (1, m_2)$; 3. Set the log normal diffusion field as:

$$\log a(\mathbf{x}) \sim \mathrm{GP}(a|0, k_1(\mathbf{x}, \mathbf{x}'))\mathbb{I}_{\Omega_1}(\mathbf{x}) + \mathrm{GP}(a|0, k_2(\mathbf{x}, \mathbf{x}'))\mathbb{I}_{\Omega_2}(\mathbf{x}), \tag{3.29}$$

where, $\mathbb{I}_{\mathcal{A}}(\cdot)$ is the indicator function and k_1 and k_2 are exponential covariance kernels, each with their own unique lengthscale pair. We generate 100 such samples of stratified diffusion fields and use the DNN surrogate to predict the corresponding PDE solution. Fig. 3.16 shows a comparison of the PDE solution obtained from the FV solver and the PDE solution predicted by the DNN surrogate for 4 randomly chosen examples. Averaged over all 100 examples, we obtain a mean R^2 score of 0.99163, and a mean relative error, $\mathcal{E}$, of 0.04468. Qualitatively, however, it is clear that the DNN surrogate is unable to capture jumps in the solution caused at the stratification interface. This is consistent with our observation that the regularization of the DNN weights leads to predictions where sharp transitions are smoothed out. This might be alleviated, to some extent, by including in the training dataset examples with non-smooth solutions.

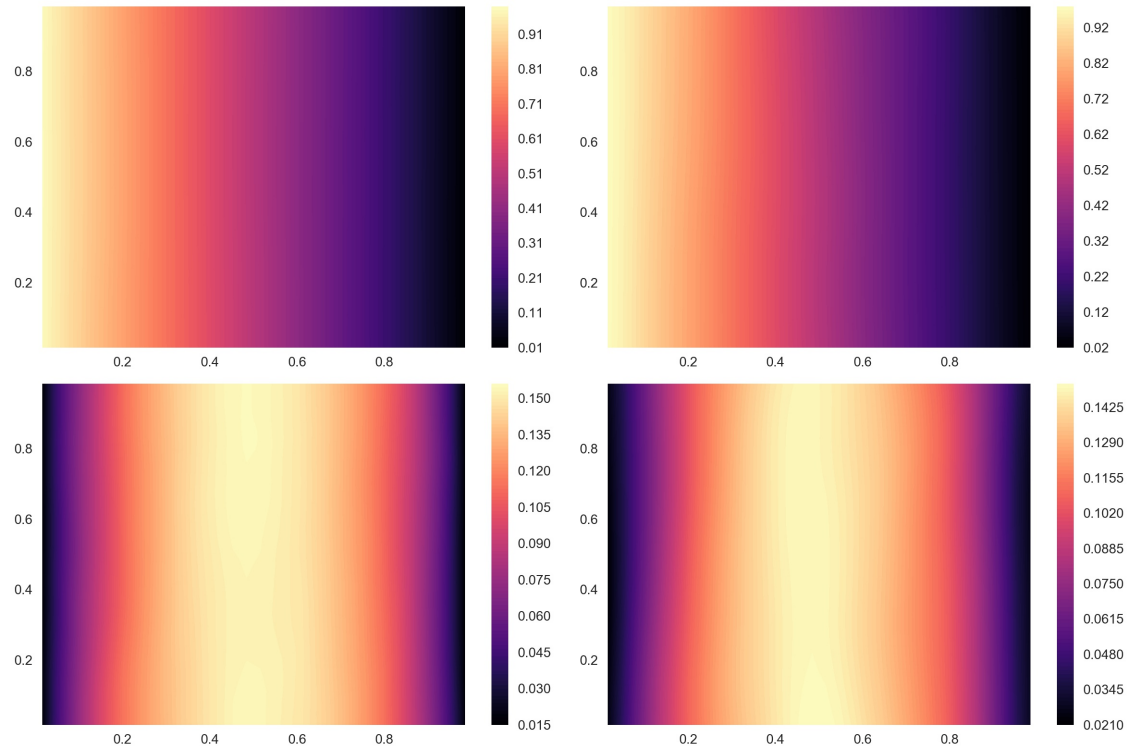
3.8.9 Uncertainty Propagation

Having constructed a DNN surrogate that maps the input diffusion coefficient of the PDE in Eq. 3.22 and verified its accuracy, we can use this surrogate to solve UP problems. This surrogate is generalizable for arbitrary choices of the lengthscale of the input diffusion field. Additionally, the DNN surrogate can also be used to



(a)

(b)



(c)

Figure 3.17. Mean and standard deviation of the PDE solution obtained by MC sampling of the DNN surrogate. In each sub figure the left column shows the MCS approximation and the right column shows the DNN approximation. The top half compares the mean of the solution and the bottom half compares the standard deviation. 3.17(a) - Case 1: $\ell_x = 0.1$ and $\ell_y = 0.5$. 3.17(b) - Case 2: $\ell_x = 0.05$ and $\ell_y = 0.15$. 3.17(c) - Case 3: $\ell_x \sim \text{TN}(0.1, 0.03, 0.07, 0.13)$ and $\ell_y \sim \text{TN}(0.5, 0.03, 0.47, 0.53)$.

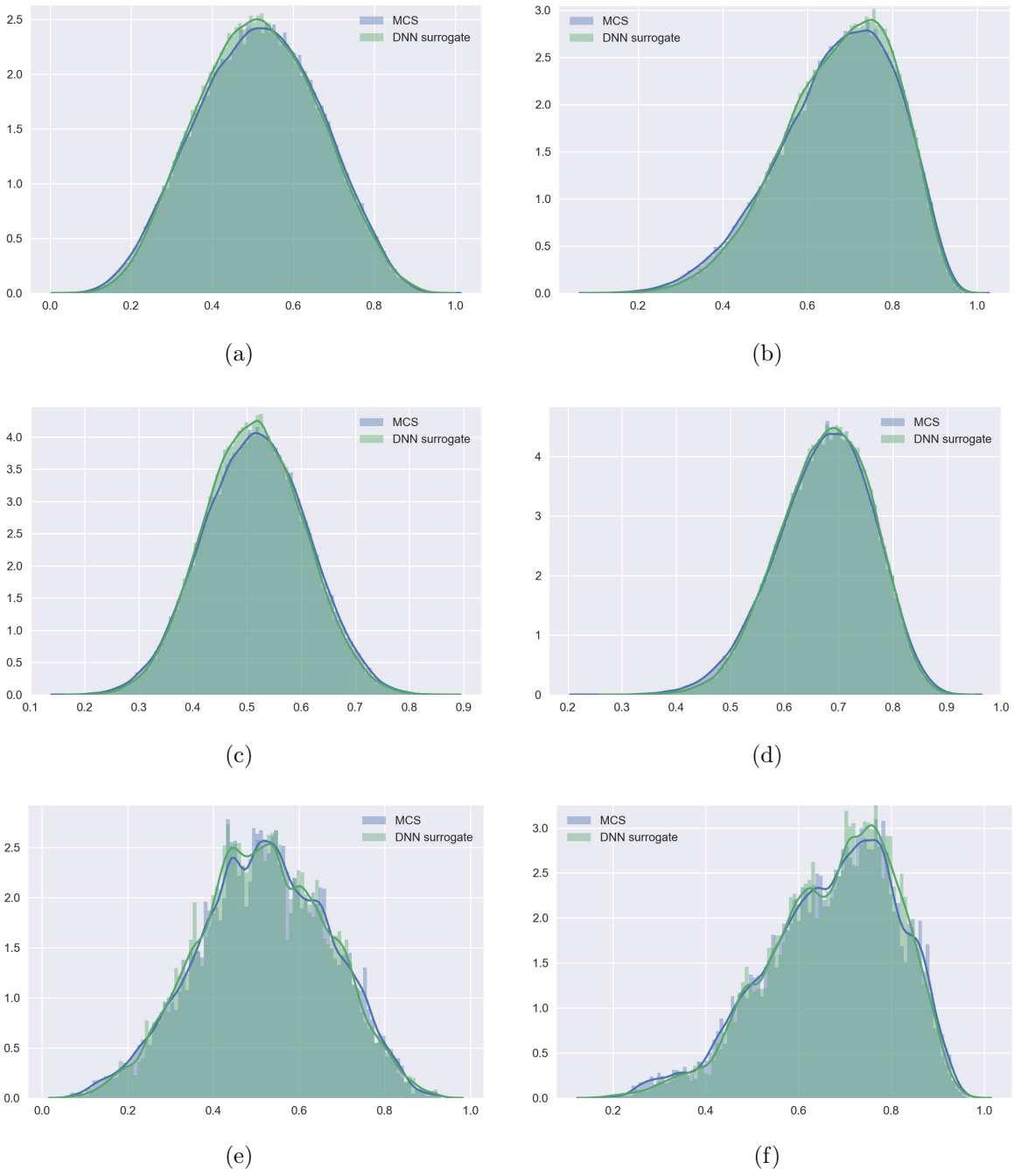


Figure 3.18. 3.18(a), 3.18(c) and 3.18(e) - Density of PDE solution at $\mathbf{x}_1$ for cases 1, 2 and 3 respectively. 3.18(b), 3.18(d) and 3.18(f)- Density of PDE solution at $\mathbf{x}_2$ for cases 1, 2 and 3 respectively.

propagate uncertainty for the case of uncertainty in the lengthscales. We solve the following 3 UP cases -

1. Case 1: Fixed lengthscales with $\ell_x = 0.1$ and $\ell_y = 0.5$.

2. Case 2: Fixed lengthscales with $\ell_x = 0.05$ and $\ell_y = 0.15$.

3. Case 3: Uncertain lengthscales with $\ell_x \sim \text{TN}(0.1, 0.03, 0.07, 0.13)$ and $\ell_y \sim \text{TN}(0.5, 0.03, 0.47, 0.53)$,

where, $\text{TN}(\mu, \sigma, a, b)$ is a truncated normal distribution with location and scale parameters, μ and σ, and support (a, b). In each case, we draw 10^5 samples from the corresponding input distribution and estimate the following output statistics from the DNN surrogate predictions:

1. Mean of $\hat{\mathbf{u}}$.

2. Variance of $\hat{\mathbf{u}}$.

3. Probability density of the PDE solution at $\mathbf{x}_1 = (0.484, 0.484)$ and $\mathbf{x}_2 = (0.328, 0.641)$.

The comparison between DNN surrogate approximation of the above quantities and their corresponding MCS approximations, for cases 1 and 2, are shown in Fig. 3.17 and Fig. 3.18. The relative error and R^2 scores between the DNN surrogate and the MCS approximations of the mean and standard deviations are shown in Tab. 3.2.

We note that the mean PDE solution from the DNN surrogate matches that from the MCS sampling very closely in both cases. The error in the standard deviation, while reasonably low, is increased because of the tendency of the DNN to 'smooth out' the solution as discussed earlier. This is why we see a somewhat larger relative error for case 2, where the smaller lengthscales of the diffusion coefficient lead to PDE solutions that are inherently less smooth than the larger lengthscales of case 1.

Table 3.2. Relative error and R^2 scores in the mean and variance of the PDE solution for two different choices of spatial lengthscale pairs.

Case	Mean		Standard deviation	
	$\mathcal{E}$	R^2	$\mathcal{E}$	R^2
1	0.01174	0.99944	0.06565	0.96446
2	0.01080	0.99953	0.07035	0.95105
3	0.01187	0.99942	0.05941	0.97148

3.9 Multifidelity modeling

Many engineering problems are simulated, not by a single simulator, rather by a suite of simulators of varying accuracy. Typically, a simulator of high accuracy requires a greater time for a single evaluation, whereas, a simulator of lower accuracy has a smaller computational cost. A typical example of such a scenario can be found in the field of computational fluid dynamics (CFD), where the higher accuracy turbulence closure models carry a greater computational burden. The natural question one might ask here is - how does one construct a surrogate when one can gather information from a variety of sources of varying fidelity? It's worth keeping in mind that the accuracy of a surrogate is only as good as the quality of the solver which produces data. How then, does one systematically utilize a small amount of high fidelity information to improve the quality of a low fidelity surrogate? More generally, if one has a suite of simulators, and a fixed computational budget, how does one fuse information from all of these sources, and in what amount, to construct a surrogate with optimal accuracy? Such questions lie at the heart of the field known as *multifidelity uncertainty quantification* [168]. This area of exploration is not new. The seminal work on this topic is the celebrated autoregressive Gaussian process framework of [169]. This model was extended into the decoupled recursive co-kriging framework of [170]. Finally, [171] generalized the linear autoregressive model of [170] into a nonlinear framework. The common thread tying all frameworks together is that - a. they rely upon GPs to construct surrogates, and b. they do not consider ways to tackle the curse of dimensionality in the input parameter space and as such are applied primar-

ily to problems with a small number (< 10) of input stochastic dimensions. In this section we present a simple extension to the DNN surrogate structure from Sec. 3.1 to incorporate data from multiple sources.

3.9.1 Multifidelity DNN structure

Suppose we have a suite of computer codes of varying fidelities and we index them by s, i.e. $f_s(\mathbf{x}), s = 1, 2, \ldots, m$, where f_1 represents the cheapest and lowest fidelity computer code and f_m represents the highest fidelity, but most expensive, simulator. We assume that there exists some underlying correlation between the various simulators, i.e., the output from any given model is predictive of the output from any other model for the same vector of inputs, $\mathbf{x}$.

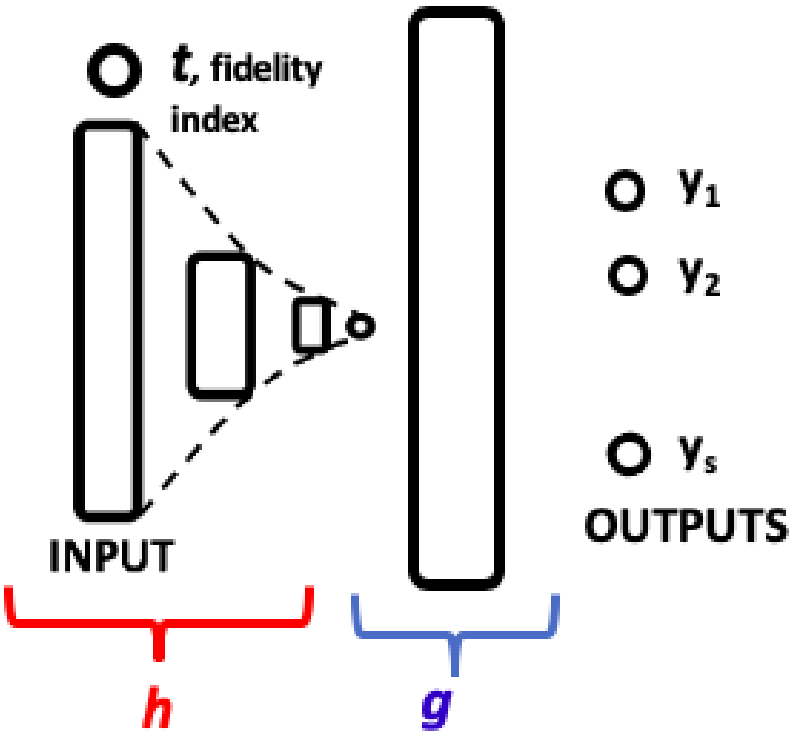


Figure 3.19. Extension of proposed architecture to the multifidelity case.

Suppose we have, at our disposal, an experimental design of inputs to be run at varying fidelity levels. Denote the dataset of observations collected by running

the computer code at fidelity level i, as $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_i}$. The complete dataset consisting of data samples generated at every fidelity level is then, denoted as, $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_s\}$. To handle the multifidelity case, we make a few simple changes to the structure of the DNN. First, we retain the overall shape. That is, the overall architecture is a composition of two nonlinear functions, $f(\mathbf{x}) = g(h(\mathbf{x}))$, where, g is a neural network link function with 1 hidden layer and h is a nonlinear projection function whose architecture is governed according to the rule defined in Eqn. (3.16). In addition to the stochastic parameters, $\mathbf{x}$, we add a new input parameter t which indexes the fidelity level of the simulator. Finally, we expand the output space to produce s outputs - each corresponding to the computer code from a specific fidelity level. For a visual representation of the architecture, see Fig. 3.19. To summarize, the modification to the original architecture does the following - given an input-output pair $(\mathbf{x}, y)$ and a fidelity level, the network should produce a vector of predictions, $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_s$ for the scalar quantity of interest for *all* fidelity levels. Finally, we modify our loss function to accommodate these changes in the architecture. If the input fidelity level is t, where $t \in \{1, 2, \ldots, s\}$, we need to compare the output label with the t^{th} component of the output vector.

3.9.2 Example - stochastic elliptic PDE with bi-fidelity data

Consider the elliptic PDE problem in two dimensions:

$$\nabla \left(a(\mathbf{x}) \nabla u(\mathbf{x}) \right) = 0, \tag{3.30}$$

where, $\mathbf{x} = (x_1, x_2) \in \Omega = [0, 1]^2$.

The PDE is equipped with boundary conditions -

$$\begin{aligned} u &= 0, \forall x_1 = 1, \\ u &= 1, \forall x_1 = 0, \\ \frac{\partial u}{\partial n} &= 0, \forall x_2 = 1, \end{aligned} \tag{3.31}$$

As usual, we model the diffusion with as a lognormal Gaussian random field with zero mean:

$$\log a(\mathbf{x}) \sim GP(0, k(\mathbf{x}, \mathbf{x}')), \tag{3.32}$$

where, k is set to an exponential kernel with lengthscales ℓ_1 and ℓ_2 corresponding to coordinates x_1 and x_2. We set $\ell_1 = \ell_2 = 0.3$.

To obtain a finite dimensional representation of the Gaussian random field, we resort to the Karhunen-Loeve expansion:

$$\log a(\mathbf{x}) = \sum_{i=1}^{M} \sqrt{\lambda_i}\phi_i(\mathbf{x})\xi_i, \tag{3.33}$$

where, $\xi_i \sim \mathcal{N}(0, 1)$ are i.i.d. Gaussian. $M = 350$ terms are needed to retain 95% of the variance of the random field.

We set up the bi-fidelity problem as follows. We consider a fine discretization(64×64) and a coarse discretization (32×32) of the input domain. The FEM solver is simulated $N_{low} = 900$ times for the coarse mesh and $N_{high} = 300$ times for fine mesh. We use Algorithm 4 to train the network with these $N_{low} + N_{high} = 1200$ data samples from two fidelity levels. Separately, we generate datasets comprising of *only* high-fidelity data samples, with varying dataset sizes and follow our standard training procedure.

Fig. 3.20 shows the decay of the test set mean squared error obtained from models trained with datasets comprising purely high-fidelity data. The errors obtained from these monofidelity datasets are compared to the test error obtained from the bifidelity dataset consisting of N_{high} and N_{low} high fidelity and low fidelity data. We observe that by augmenting a small number of high-fidelity runs (N_{high}) with a large number of cheaper low fidelity runs (N_{low}), we obtain a model that is as accurate as a model obtained from training with a dataset with 800 high fidelity samples.

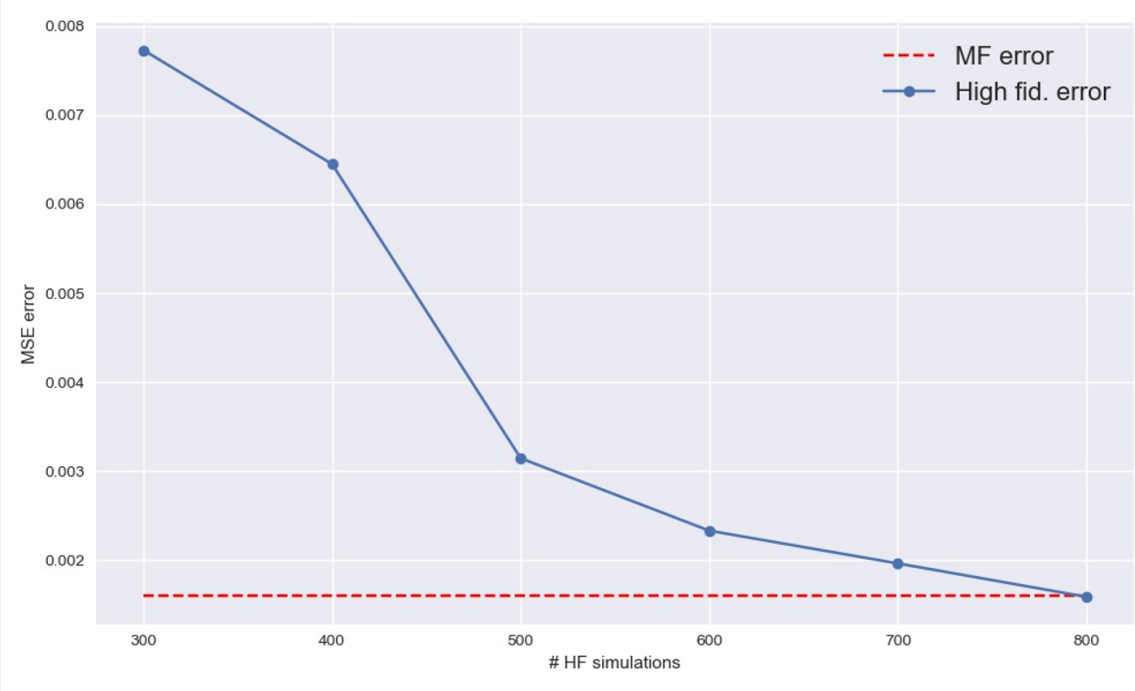


Figure 3.20. Comparison of the test dataset mean squared error (MSE) obtained from a purely high-fidelity dataset of varying dataset sizes with the MSE from a bifidelity dataset comprising data from runs of fine-grid and coarse-grid simulations.

3.10 Closing remarks

We propose a methodology for learning DNN surrogate models for uncertainty quantification based on a parameterization of the DNN structure, such that the DNN is a composition of an encoder and one-layer MLP. Our parameterization lends the DNN surrogate the interpretation of recovering a nonlinear active subspace. We use a combination of grid search and BGO to select model hyperparameters, namely, the number of hidden layers, L, the width of the AS, h, and the weight decay regularization constant, λ. We demonstrate our methodology with a UP problem in elliptic SPDE with uncertain diffusion coefficient, and learn a surrogate which maps a 'picture' of the discretized version of the coefficient to the PDE solution. Furthermore, we demonstrated that the DNN surrogate can effectively predict the solution of the PDE, even for diffusion fields with lengthscales that are not used for training the network.

This work is an early step towards using deep learning to create surrogate models for high dimensional UQ tasks. UQ for state-of-the-art computational simulators are notoriously difficult because of the prohibitive time span for individual simulations. One can extend the methodology proposed in this work to a Bayesian treatment of DNNs [172], i.e., imposing a prior on the weights of the NN and using approximate inference techniques such as variational inference [140, 173] to estimate the posterior distribution over the weights. Additionally, the Bayesian approach would allow one to better quantify the epistemic uncertainty induced by limited data.

DNNs are also naturally suited for tasks for multilevel/multifidelity UQ [168, 169, 171]. For instance, fully convolutional networks do not impose constraints on input dimensionality and can be trained on data obtained from several simulators at varying levels of fidelity. The hierarchical representation of information with a deep network can be used to learn correlations between heterogeneous information sources.

4. GRADIENT-FREE ACTIVE SUBSPACE RECOVERY IN DEEP NEURAL NETWORKS

[1] Many quantities of interest in the engineering sciences are both expensive to obtain and functions of a large number of input parameters. This makes construction of surrogate models of these quantities of interest, for various UQ tasks, infeasible, through naive approaches. Fortunately, such functions often exhibit some underlying low-dimensional structure. Such structure might be in the form of a low dimensional manifold or a decomposition of the function into an additive structure over several low-dimensional functions. A key focus of this thesis has been on exploring approaches to recover *active subspaces* - a specific kind of low-rank structure that is intrinsic to a high-dimensional multivariate function. We saw, in Ch. 2, that the active subspace representation can be embedded into a Gaussian process covariance kernel, lifting the requirement of obtaining gradients of the function with respect to the input. This was a significant relaxation over the classical approach to recovering active subspaces. In Ch. 3, we tackled the problem of finding a nonlinear version of the active subspace by posing the task in the framework of deep neural networks. The use of DNNs allowed us to tackle UQ tasks that were much more ambitious in scope compared to the GP approach laid out in Ch. 2. This is due to the inherent scalability advantages of a parametric representation (such as DNN) over a nonparametric representation such as GP. In this chapter, we present a methodology that applies the flexibility and scalability of DNNs to the task of recovering the classical version of the active subspace without access to gradient information. The key obstacle to our goal in this approach is that the standard stochastic gradient descent (SGD) approach to training

[1]The contents of this chapter are reproduced, with permission, from the paper entitled "Deep Active Subspaces: A Scalable Method for High-Dimensional Uncertainty Propagation" [174] published in the Proceedings of the ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (2019).

DNNs do not respect the necessary orthogonality constraints on the active subspace projection matrix. To enforce these constraints, we rely on a reparameterization of the weights of the first layer of the DNN, which makes our network amenable to standard SGD routines for training while retaining AS projection orthogonality.

4.0.1 Formal problem description

Consider a physical system modeled with a (potentially complex, coupled) system of partial differential equations. The PDE(s) is solved numerically using a black-box computer code, which we denote as f. f may be thought of as a multivariate function which accepts a vector of inputs $\boldsymbol{\xi} \in \boldsymbol{\Xi} \subset \mathbb{R}^D$ and produces a scalar quantity of interest (QoI) $f(\boldsymbol{\xi}) \in \mathcal{Y} \subset \mathbb{R}$. Information about f may be obtained through querying the solver at suitable input design locations $\boldsymbol{\xi}$. We allow for the possibility that our measurement from the computer code may be noisy, i.e., $y = f(\boldsymbol{\xi}) + \epsilon$, where ϵ is a random variable (the measurement noise might arise as a consequence of quasi-random stochasticity or chaotic behavior). Given this setup, the uncertainty propagation (UP) task is summarized as follows. Given a formal description of the uncertainty in the input parameters, $\boldsymbol{\xi} \sim p(\boldsymbol{\xi})$, we would like to estimate the statistical properties of the QoI. This includes, the probability density,

$$p(f) = \int \delta(f - f(\boldsymbol{\xi}))p(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}, \tag{4.1}$$

and measures of central tendency such as the mean:

$$\mu_f = \int f(\boldsymbol{\xi})p(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}, \tag{4.2}$$

and variance:

$$\sigma_f^2 = \int (f(\boldsymbol{\xi}) - \mu_f)^2 p(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}, \tag{4.3}$$

where, $\delta(\cdot)$ in Eqn. (4.1) refers to the Dirac δ-function.

As already discussed in the introduction, the standard MC method is infeasible when there is a large computational cost associated with querying f and one must resort to the surrogate approach - replacing the true simulator f with an accurate, cheap-to-evaluate approximation, $\hat{f}$. To do this, one queries f at a set of N carefully selected design locations $\mathbf{X} = (\boldsymbol{\xi}^i)_{i=1}^N$, resulting in a corresponding set of measurements, $\mathbf{y} = (y^i)_{i=1}^N$. We refer to the observed data, collectively, as $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$. Although the task of careful selection of the input design locations are a subject of a great deal of research, an in-depth discussion of this topic is beyond the scope of the present work. Here we simply assume that we are given $\mathcal{D}$.

4.0.2 A review of active subspaces

The fact that we are working in a high-dimensional regime $D(\gg 1)$ makes the task of constructing an accurate surrogate model with limited data ($N \approx \mathcal{O}(D)$) practically infeasible because of the curse of dimensionality. To circumvent this, one seeks to exploit low-rank structure within the true response f and methods for doing so are broadly categorized as 'dimensionality reduction' techniques. In this work, we focus on the case where the response admits the following structure:

$$f(\boldsymbol{\xi}) = g(\boldsymbol{\zeta}) = g(\mathbf{W}^T\boldsymbol{\xi}), \tag{4.4}$$

where, $\mathbf{W} \in \mathbb{R}^{D\times d}$ is a tall-and-skinny matrix of orthogonal columns which projects the high-dimensional input $\boldsymbol{\xi}$ to $\boldsymbol{\zeta}$ lying in a d-dimensional subspace such that $d \ll D$. In particular, $\mathbf{W}$ is constrained to be an element of the set:

$$\mathcal{V}_d\left(\mathbb{R}^D\right) = \left\{\mathbf{A} \in \mathbb{R}^{D\times d} : \mathbf{A}^T\mathbf{A} = \mathbf{I}_d\right\}. \tag{4.5}$$

$\mathcal{V}_d\left(\mathbb{R}^D\right)$ is known as the *Stiefel manifold* with $\mathbf{I}_d$ being the identity matrix in $\mathbb{R}^{d\times d}$ and $g : \mathbb{R}^d \to \mathbb{R}$ is known as the *link function*. The structure posited in Eqn. (4.4) takes on physical meaning where the columns of $\mathbf{W}$ correspond to directions of the input space

most sensitive to variation in the response f. The dimensionality reduction induced by the introduction of this structure, significantly simplifies the task of learning an accurate surrogate model.

Classical approach to active subspace recovery

The classical approach to recovering the *active subspace*, introduced in [81], proceeds as follows. Let the gradient of the QoI w.r.t. the input be denoted as $\nabla_{\boldsymbol{\xi}} f = \left(\frac{\partial f}{\partial \boldsymbol{\xi}_1}, \frac{\partial f}{\partial \boldsymbol{\xi}_2}, \cdots, \frac{\partial f}{\partial \boldsymbol{\xi}_D}\right) \in \mathbb{R}^D$. Given a probability distribution ρ endowed upon the input space, we define the symmetric positive semi-definite matrix,

$$\mathbf{C} = \int (\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}))(\nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}))^T \rho(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}, \tag{4.6}$$

which admits the spectral decomposition $\mathbf{C} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$, where $\boldsymbol{\Lambda}$ is a diagonal matrix of eigenvalues ordered by magnitude. Separating the d largest eigenvalues from the rest, we can write the matrix of eigenvectors, $\mathbf{V}$, as:

$$\mathbf{V} = \left[\mathbf{V}_1, \mathbf{V}_2\right], \tag{4.7}$$

where, $\mathbf{V}_1 \in \mathbb{R}^{D \times d}$ is a matrix consisting of the eigenvectors corresponding to the d largest eigenvalues and $\mathbf{V}_2 \in \mathbb{R}^{D \times (D-d)}$ is composed of the remaining eigenvectors. The active subspace projection matrix, then, is simply, $\mathbf{W} = \mathbf{V}_1$. Since the integral in Eqn. (4.6) is intractable analytically (due to the black-box nature of f), one only has access to discrete samples of the gradient at input locations $\boldsymbol{\xi}$ sampled from the distribution ρ. Given a dataset of S gradient evaluations, $\mathbf{g}^{(i)} = \nabla_{\boldsymbol{\xi}} f(\boldsymbol{\xi}^{(i)}), i = 1, 2, \cdots, S$, where the $\boldsymbol{\xi}^{(i)}$s are sampled iid from ρ, an approximation to the matrix $\mathbf{C}$ may be constructed as:

$$\mathbf{C}_S = \frac{1}{S} \sum_{i=1}^{S} \mathbf{g}^{(i)} \mathbf{g}^{(i),T}. \tag{4.8}$$

One may think of the approximation $\mathbf{C}_S$ as an empirical covariance matrix. After recovering the projection matrix $\mathbf{W}$ through the above procedure, one can obtain projected inputs, using $\mathbf{z} = \mathbf{W}^T\mathbf{x}$ and using a suitable technique such as Kriging to learn the link function $g(\cdot)$.

Recovering active subspaces without gradients within Gaussian processes

As discussed in Sec. 4.0.2, the classic approach to AS recovery requires the evaluation of an empirical covariance matrix from samples of the gradient $\nabla_{\boldsymbol{\xi}} f$. Obtaining gradient samples in challenging in practice. In some cases (such as simple dynamical systems), one might have access to an adjoint solver which can compute the gradients of the QoI wrt the input parameters [**?**]. In other cases, the gradients can be approximated through finite differences (FD). Note that a single first-order FD gradient evaluation requires 2 expensive forward model runs. Lastly, one might even approximate gradients through approximate global models for the data [90]. In general, the black-box nature of the response as well the associated cost of FD gradients means that one simply does not have access to $\nabla_{\boldsymbol{\xi}} f$ and therefore cannot construct $\mathbf{W}$ through the classical approach. To alleviate this limitation, [43] introduced a methodology for constructing surrogate models without requiring gradient information. The gradient-free approach relies on two key ideas:

1. In GPR, prior knowledge about the underlying function can be encoded in a principled manner through the mean and the covariance functions of the GP. Thus, a new covariance kernel may be defined where the AS projection matrix $\mathbf{W}$ is simply a hyperparameter and learned through available data, $\mathcal{D}$. Formally, the prior knowledge about the active subspace structure described in Eqn. (4.4) is expressed through a GP kernel which takes on the form:

$$k_{\text{AS}}(\mathbf{x}, \mathbf{x}') = k_{\text{base}}(\mathbf{z}, \mathbf{z}') = k_{\text{base}}(\mathbf{W}^T\mathbf{x}, \mathbf{W}^T\mathbf{x}'), \tag{4.9}$$

where, $k_{\text{base}}(\cdot,\cdot)$ is any standard kernel (such as the *Matern* or *Radial basis function (RBF)* kernels) which expresses prior knowledge about the regularity properties of the link function $g(\cdot)$. Once the active subspace kernel has been defined, inference in GPR proceeds through the maximization of the log marginal likelihood of the data wrt the kernel hyperparameters i.e.,

$$\mathbf{W}^*, \mathcal{H}^*, \sigma_n^* = \underset{W,H}{argmax} \log p(\mathbf{y}|\mathbf{X}, \mathbf{W}, \mathcal{H}, \sigma_n), \tag{4.10}$$

where, $\mathcal{H}$ is the set of all hyperparameters of the base kernel k_{base}, and σ_n is the standard deviation of the likelihood noise.

2. While it is easy to enforce positivity constraints on the hyperparameters $(\mathcal{H}, \sigma_n) = \boldsymbol{\phi}$, the optimization task in Eqn. (4.10) is made challenging because of the fact that it is non-trivial to enforce the orthogonality constraints on the projection matrix $\mathbf{W}$. In order to do so, the complete methodology of [43] relies on a coordinate-ascent scheme to iteratively optimize over the variables $\boldsymbol{\phi}$ while keeping $\mathbf{W}$ constant and vice versa. The optimization steps over $\boldsymbol{\phi}$ proceed via standard second-order techniques for unconstrained optimization, such as the L-BFGS method [142]. The optimization steps over the projection matrix $\mathbf{W}$ utilize an adapted version of gradient-ascent on the Stiefel manifold described in [144].

4.0.3 Active subspace recovery in neural networks

The methodology introduced by [43] lifts the gradient requirement of the classical approach to AS recovery by subsuming the AS projection matrix into the covariance kernel of a GP. While the methodology is sound and experimentally shown to recover the true AS, it suffers from two major drawbacks -

1. It is not agnostic to the choice of the surrogate model. Note that the gradient-free method described in Sec. 4.0.2 necessitates a GP surrogate by construction.

Inspite of the elegance of GPR, arising out of the principled framework it offers for incorporating prior knowledge, quantifying epistemic uncertainty and performing model selection, its standard formulation scales poorly due to the $\mathcal{O}(N^3)$ inversion of the (potentially dense) covariance matrix required at each optimization step. While sparse GPR [156, 157] partially alleviates this poor scaling through the introduction of $M(\ll N)$ inducing variables or 'pseudo-inputs', the task of selecting or optimizing for the inducing input locations is non-trivial.

2. The proposed solution for optimizing over the projection matrix $\mathbf{W}$, while respecting orthogonality constraints, is itself non-trivial, introduces Dd additional hyperparameters into the covariance kernel, and is prone to getting trapped in local stationary points [43].

We propose, here, a much simpler approach that is:

1. Is agnostic to the choice of the link function approximator,

2. Is trivial to implement.

Specifically, we express $\mathbf{W}$ as:

$$\mathbf{W} = h(\mathbf{Q}), \tag{4.11}$$

where, $\mathbf{Q} \in \mathbb{R}^{D\times d}$ lies on the standard Euclidean space, and $h : \mathbb{R}^{D\times d} \to \mathbb{R}^{D\times d}$ orthogonalizes the columns of $\mathbf{Q}$. Specifically, we chose h to be the celebrated *Gram-Schmidt (GS) orthonormalization* procedure [**?**]. The GS process may be summarized as follows. Given an unconstrained matrix $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_d] \in \mathbb{R}^{D\times d}$, where the $\mathbf{q}_i$s are the columns of $\mathbf{Q}$, we apply the transformation,

$$\mathbf{w}_i = \mathbf{w}_{i-1} - \left(\frac{\mathbf{w}_{i-1}^T\mathbf{q}_i}{\mathbf{w}_{i-1}^T\mathbf{w}_{i-1}}\right)\mathbf{w}_{i-1},\ i = 2, 3, \cdots, d, \tag{4.12}$$

with $\mathbf{w}_1 = \mathbf{q}_1$. The projection matrix $\mathbf{W}$ is then assembled by normalizing the $\mathbf{w}_i$s, i.e., $\mathbf{W} = \left[\frac{\mathbf{w}_1}{\|\mathbf{w}_1\|_2}, \frac{\mathbf{w}_2}{\|\mathbf{w}_2\|_2}, \cdots, \frac{\mathbf{w}_d}{\|\mathbf{w}_d\|_2}\right]$.

Now one only needs to care about the the Euclidean matrix $\mathbf{Q}$, and optimize it to the available data. Noting that the transformation specified by Eqn. (4.12) is fully differentiable (as it composed entirely of differentiable mathematical operations), one may simply define a routine implementing the GS process using a backpropagation-capable library (such as `TensorFlow` or `PyTorch`) to obtain exact gradients of any QoI wrt $\mathbf{Q}$.

Since the projection matrix $\mathbf{W}$ has been reparameterized without any concern for the specific structure of the link function, g, we are free to pick any suitable class of function approximator for g. In this work, we define g to be a deep neural networks (DNN) [112], a class of highly flexible nonlinear function approximators with satisfy universal approximation properties. Formally, a L-layered DNN representation for g is defined as:

$$g(\mathbf{z}) = f_{L+1} \circ f_L \circ \cdots \circ f_1(\mathbf{z}), \tag{4.13}$$

where, $f_i(\mathbf{z}_{i-1}) = h_i(\mathbf{W}_i^T \mathbf{z}_{i-1} + \mathbf{b}_i)$, with $\mathbf{W}_i \in \mathbb{R}^{d_i \times d_{i-1}}, \mathbf{b}_i \in \mathbb{R}^{d_i}, \mathbf{z}_i = \mathbb{R}^{d_i}, \mathbf{z}_0 = \mathbf{z}, \mathbf{z}_{L+1} = g(\mathbf{z})$, and $h_i(\cdot)$ is a suitable nonlinear function applied elementwise on its argument. h_{L+1} is set to be the identity function (since we are dealing with unconstrained real-valued outputs) and the other h_is are set as the hyperbolic tangent function, a standard choice in the literature. The matrices $\mathbf{W}_i$s and the vectors $\mathbf{b}_i$s are called the 'weights' and 'biases' of the DNN and here we denote all of them collectively as $\boldsymbol{\theta} = \{\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}_{L+1}, \mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_{L+1}\}$. The full surrogate, is there expressed as:

$$\hat{f}(\boldsymbol{\xi}; \boldsymbol{\theta}) = g(h(\mathbf{Q})^T \boldsymbol{\xi}; \boldsymbol{\theta}), \tag{4.14}$$

where the unknown parameters $(\boldsymbol{\theta}, \mathbf{Q})$ can be optimized through standard gradient-descent techniques. In this work, we use the famous Adaptive Moments (ADAM) optimization method [122].

4.0.4 Synthetic example with known active subspace

Let $f : \mathbb{R}^D \to \mathbb{R}$ such that $f(\boldsymbol{\xi}) = g(\boldsymbol{\zeta}) = g(\mathbf{W}^T\boldsymbol{\xi})$ where $\mathbf{W} \in \mathcal{V}_d\left(\mathbb{R}^D\right)$. Define $g : \mathbb{R}^d \to \mathbb{R}$ as a quadratic function in $\mathbb{R}^d$:

$$g(\boldsymbol{\zeta}) = \mathbf{W}^T\boldsymbol{\xi} = \alpha + \boldsymbol{\beta}^T\boldsymbol{\zeta} + \boldsymbol{\zeta}^T\boldsymbol{\Gamma}\boldsymbol{\zeta}. \tag{4.15}$$

The gradients of f are given by

$$\nabla f(\boldsymbol{\xi}) = \left(\beta + 2\boldsymbol{\xi}^T\mathbf{W}\boldsymbol{\Gamma}\right)\mathbf{W}^T. \tag{4.16}$$

For this pedagogical example, we set $D = 20$ and test our approach on two cases with true AS dimensionality, $d = 1$ and 2. The data for inputs $\boldsymbol{\xi}$, α, β and $\boldsymbol{\Gamma}$ are generated by sampling standard Gaussians of appropriate shapes. The matrix $\mathbf{W}$ is generated by performing the QR decomposition on a similarly generated matrix in $\mathbb{R}^{D\times d}$. The random seed is fixed for reproducibility. The output data $\mathbf{y}$ used for training is standardized, i.e. it is scaled to have 0 mean and unit variance.

Case 1: 1 dimensional active subspace

We begin testing our proposed gradient-free approach on a synthetic function which has an AS of dimensionality $d = 1$. To train the AS DNN, we use $N = 50$ input-output observations. Furthermore, the output data is corrupted with zero mean Gaussian noise of standard deviation 1×10^{-2}. The link function is approximated with a 2 layer DNN of 50 units per hidden layer and L_2 regularization with a weight decay constant of 1×10^{-4} is used to prevent overfitting. The ADAM optimizer is set to perform 3×10^4 iterations with a base learning rate of 1×10^{-3} dropped by a factor of 10 every 10^4 iterations. In Fig. 4.1 we visually compare the true AS for this case and the AS discovered by the DNN. We note that they are very close, indicating that our approach has found the correct AS upto a rotation of the y axis. Fig. 4.1 also shows a comparison of the AS DNN predicted response with the true

response from a test dataset of 500 observations. Qualitatively, the predictions match the observations very closely. Quantitatively, we achieve a root mean square error of 0.039689 on the test dataset. Note that we pursue no further optimization of our DNN structure.

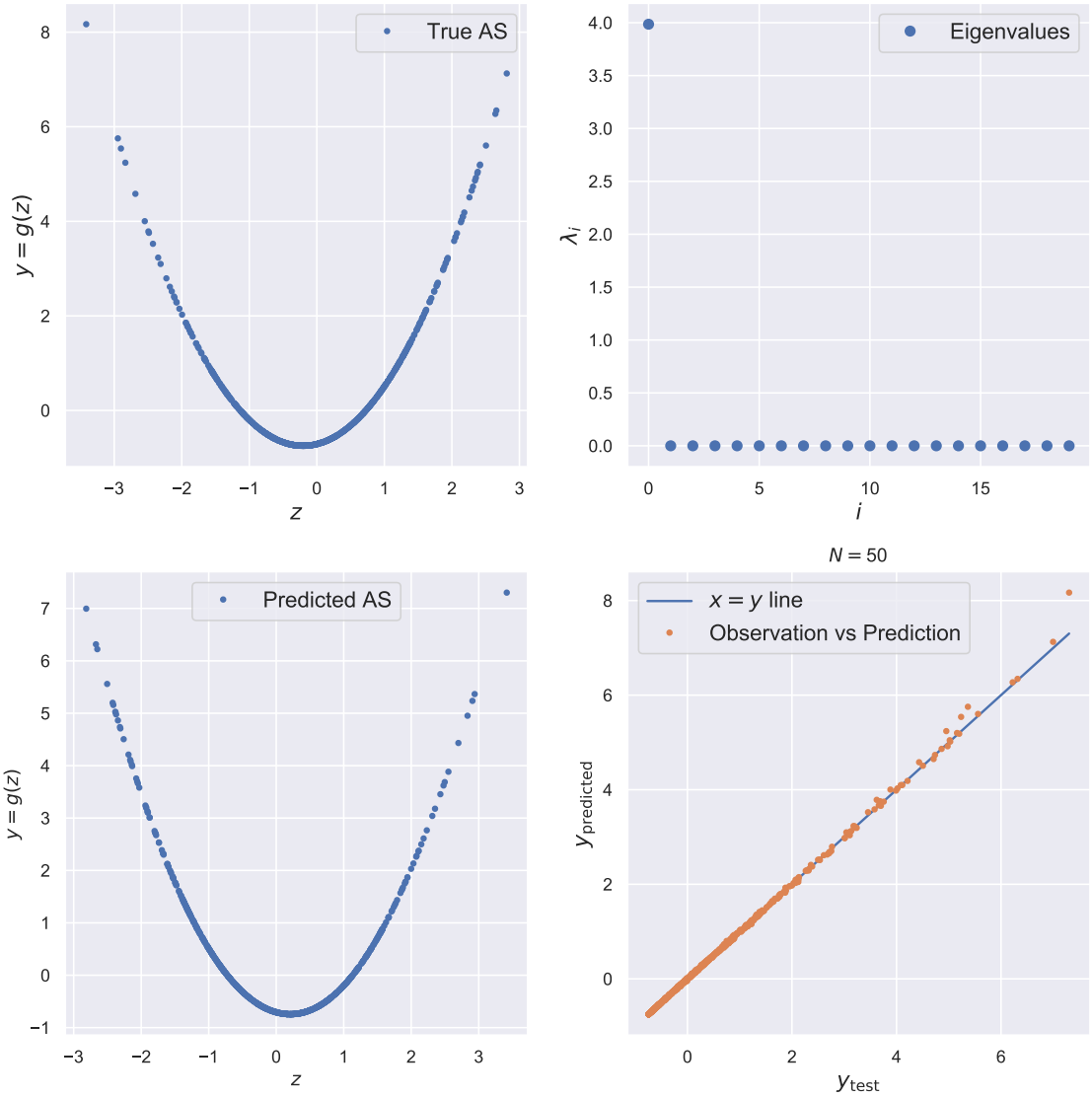


Figure 4.1. Synthetic function with $D = 20$ input dimensions admitting an $d = 1$ dimensional active subspace. Top left - True link function of f. Bottom left - Link function predicted by DNN. Top right - Spectral decomposition of the empirical covariance of the gradients. Bottom right - Comparison of predicted output and correct output on the test dataset.

4.0.5 Case 2: 2 dimensional active subspace

We now test our proposed on a synthetic function with AS dimensionality, $d = 2$. We use $N = 100$ input-output observations for training and corrupt the output data with zero mean Gaussian noise of standard deviation 1×10^{-2}. We retain all other experimental settings from Sec. 4.0.4. A comparison of the true AS and the predicted AS shown in Fig. 4.2 reveals that we recover the low-dimensional quadratic response upto arbitrary rotations of the coordinate system. We compare the predicted response of the AS DNN and true outputs from a test dataset of 500 observations. Again, we obtain excellent qualitative agreement as seen in Fig. 4.2 and quantitatively, we obtain a root mean squared error of 0.028748 between the predicted and true outputs.

4.0.6 Benchmark elliptic PDE example

Consider the following stochastic elliptic partial differential equation defined on the unit square in $\mathbb{R}^2$:

$$\nabla \cdot (a(\mathbf{s})\nabla u(\mathbf{s})) = 1, \mathbf{s} \in \Omega = [0,1]^2, \tag{4.17}$$

with boundary conditions:

$$u(\mathbf{s}) = 0, \mathbf{s} \in \Gamma_u, \tag{4.18}$$

$$\nabla u(\mathbf{s}) \cdot \mathbf{n} = 0, \mathbf{s} \in \Gamma_n, \tag{4.19}$$

where, Γ_u is the top, bottom and left boundaries and Γ_n denotes the right boundary of Ω. The diffusion coefficient a (or conductivity field) is a spatially-varying uncertain

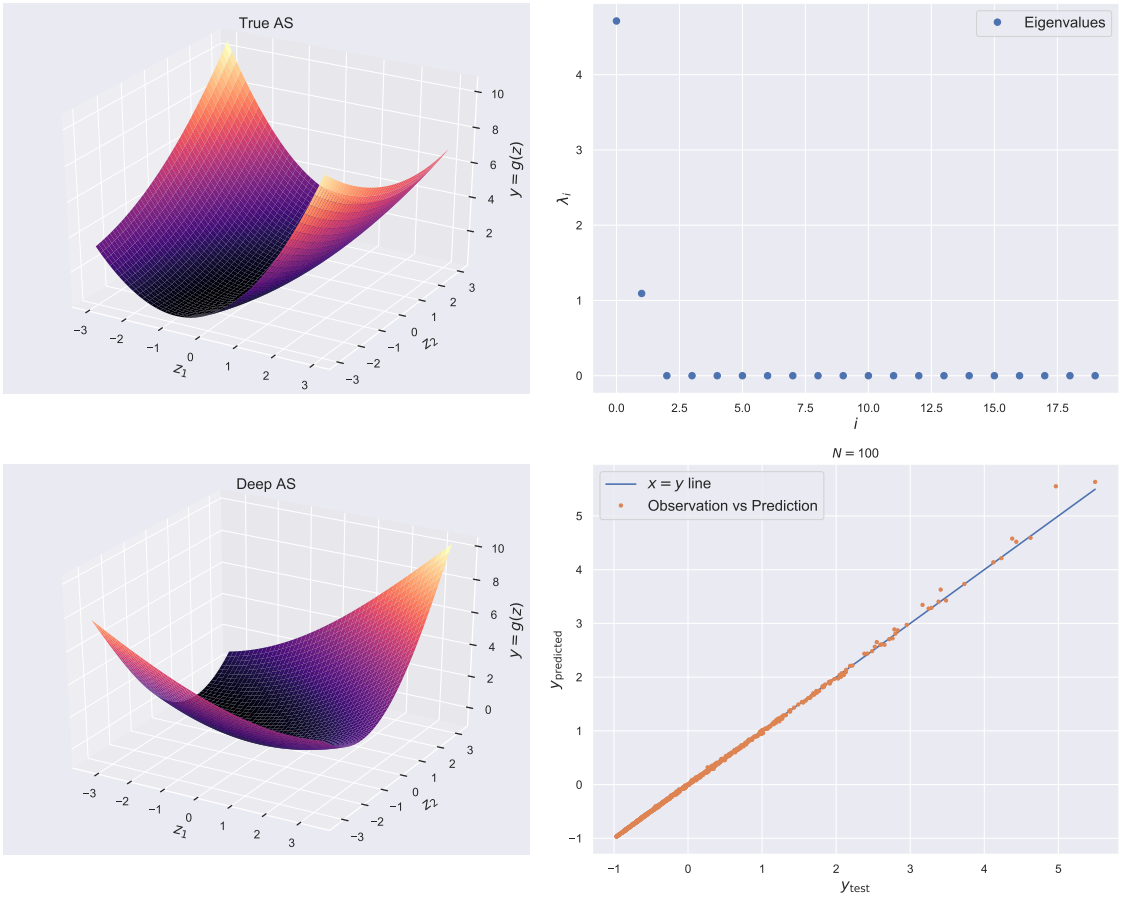


Figure 4.2. Synthetic function with $D = 20$ input dimensions admitting an $d = 2$ dimensional active subspace. Top left - True link function of f. Bottom left - Link function predicted by DNN. Top right - Spectral decomposition of the empirical covariance of the gradients. Bottom right - Comparison of predicted output and correct output on the test dataset.

input, and its logarithm is modeled as a 0-mean Gaussian random field, i.e., $\log a(\mathbf{s}) \sim GP(a|0, k(\mathbf{s}, \mathbf{s}'))$, where, the covariance function k is defined as follows:

$$k(\mathbf{s}, \mathbf{s}') = \exp\left(-\frac{\sum_{i=1}^{2}|\mathbf{s}_i - \mathbf{s}_i'|}{\ell}\right), \tag{4.20}$$

with ℓ being the correlation length. This formalization of the uncertainty around $a(\mathbf{s})$ makes it a stochastic process - an infinite dimensional quantity. We use the truncated

KL expansion to perform a preliminary dimensionality reduction by expressing the logarithm of the field as:

$$\log a(\mathbf{s}) = \sum_{i=1}^{100} \sqrt{\lambda_i}\varphi_i(\mathbf{s})x_i, \tag{4.21}$$

where, the λ_is and the φ_is are the eigenvalues and eigenfunctions of the correlation function, numerically obtained using the *Nyström approximation* [47], and the x_is are uncorrelated, standard normal random variables. Denote all the x_is collectively as $\mathbf{x} = (x_1, x_2, \cdots, x_{100}) \sim \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I}_{100})$. We are interested in the following scalar QoI:

$$q(\mathbf{x}) = \mathcal{F}[u(\mathbf{s};\mathbf{x})] = \frac{1}{|\Gamma_2|}\int u(\mathbf{s};\mathbf{x})\mathrm{d}\mathbf{s}. \tag{4.22}$$

Given a realization of the random variable, $\mathbf{x} = (x_1, x_2, \cdots, x_{100})$, one can generate a realization of the QoI, q, whose statistics one wishes to estimate. We have, at our disposal, a dataset of 300 realizations of the random variable $\mathbf{x}$ and the corresponding solution q. With this dataset, we construct a surrogate that maps $\mathbf{x}$ to q, i.e., $\hat{f} : \mathbb{R}^{100} \to \mathbb{R}$. We will consider two cases of the correlation length ℓ - a short correlation length of $\ell = 0.01$ and a long correlation length of $\ell = 1$ and attempt to recover as AS with $d = 1$. We randomly shuffle and split the data into a training set of 250 samples, and test on the remaining 50 samples. The output data is standardized to have zero mean and unit variance for numerical stability. The dataset for this example, and the code to generate it, can be found here: https://github.com/paulcon/as-data-sets/tree/master/Elliptic_PDE. Once again, we set our approximation of the link function to be a DNN with 2 hidden layers and 50 units per layer. All other experimental settings from Sec. 4.0.4 are retained. Lastly, for this example, samples of the QoI gradients are available and we use this to compare our results with the results obtained from classic AS. For the case of the classic AS, we use GPR as the link function.

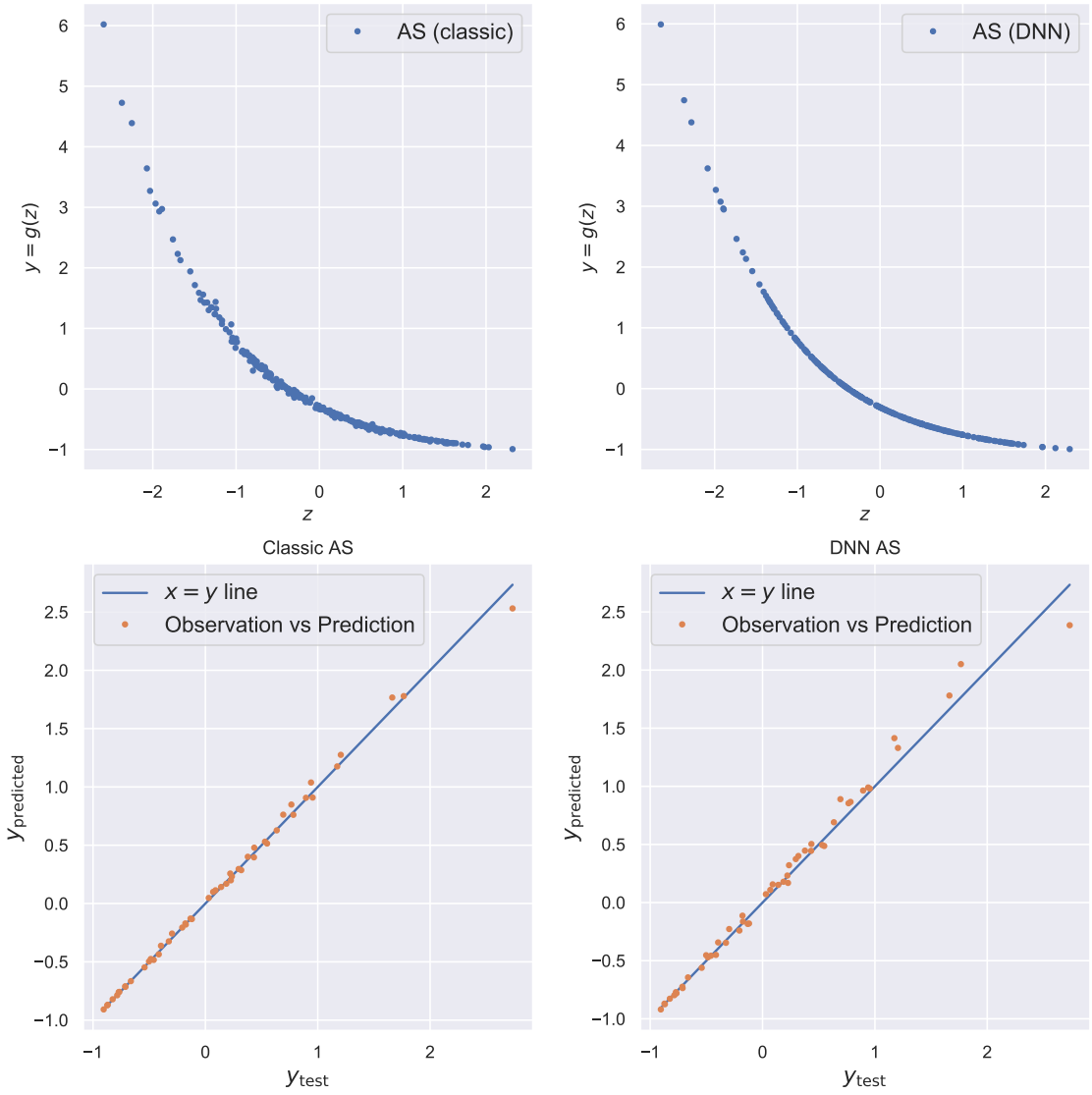


Figure 4.3. Stochastic elliptic PDE with $\ell = 1$ - The plots on the top visualize the 1d link function recovered by our gradient-free DNN AS approach and the classic AS approach. The bottom plots compare the output predictions vs observations on the test dataset for the DNN AS and the classic AS approaches.

Fig. 4.3 shows results comparing the deep AS approach and the classic AS approach for the $\ell = 1$ case. Fig. 4.4 shows the same for the case of $\ell = 0.01$. We observe that there is good qualitative agreement between the AS recovered by gradient-free deep AS approach and the gradient-based classic approach. This serves to verify the fact that our approach does indeed recover the correct AS. Table 4.1 shows a com-

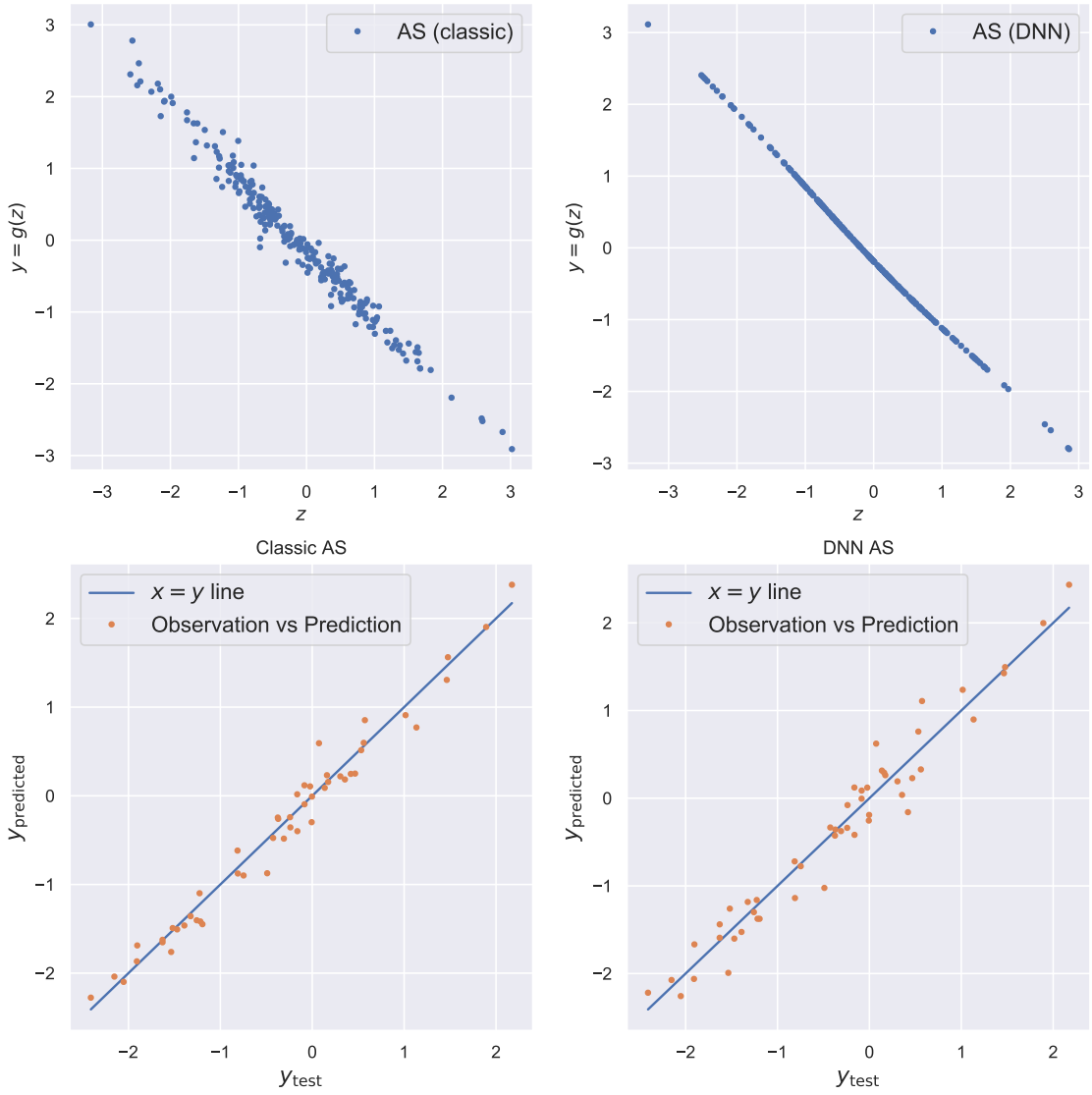


Figure 4.4. Stochastic elliptic PDE with $\ell = 0.01$ - The plots on the top visualize the 1d link function recovered by our gradient-free DNN AS approach and the classic AS approach. The bottom plots compare the output predictions vs observations on the test dataset for the DNN AS and the classic AS approaches.

parison of the RMSE error in the prediction of the outputs from the test dataset, for both ℓ cases. We observe that inspite of the fact that we do not use the information about the gradients of QoI, our gradient-free deep AS approach is are able to achieve RMSE comparable to the classic AS. Once again, we emphasize that we do not pursue optimization of the modeling choices involved in the DNN approximation of the link

function.

Table 4.1. Root mean square error (RMSE) on test dataset predictions from classic AS and deep AS response surfaces.

Approach	$\ell = 1$	$\ell = 0.01$
Classic AS	0.04378	0.17276
Deep AS	0.09241	0.23626

An interesting observation that emerges from the comparison of the classic and deep AS approaches for the short correlation length in Fig. 4.4 is that inspite of recovering a one-dimensional AS, the test data predictions from both approaches show a discrepancy from their true values. Since the QoI is generated from a deterministic computer code, we cannot explain this deviation as 'noise'. Rather, this suggests that a linear dimensionality reduction is sub-optimal and one might wish to recover a nonlinear generalization of the active subspace, such as the one discussed in [155]. An investigation into this shortcoming is beyond the scope of the present work. Finally, one may note that both the classic and the deep AS approaches perform much better on the $\ell = 1$ case, relative to the $\ell = 0.01$ case. This is unsurprising, considering that it becomes much more difficult to capture the uncertainty of the diffusion coefficient as its lengthscale reduces.

5. FUTURE WORK AND CONCLUSIONS

We conclude this thesis with a brief review of the work done so far in Sec. 5.1 followed by a discussion on open questions and future directions in Sec. 5.2.

5.1 Dissertation summary

The standard workflow for a team of computational scientists is divided, broadly, into two stages - (i) pose a mathematical model (typically a complex system of ODEs/PDEs) to describe the dynamics of the system under investigation, and, (ii) employ the most sophisticated numerical method available at their disposal, subject to a computational budget. Typical numerical solvers are characterized by a large number of input parameters - initial/boundary conditions (IC/BCs), material properties etc. It is fairly typical for many of these parameters, in a realistic problem, to be uncertain, i.e., we have incomplete information about an input to a computational solver, and this uncertainty is characterized quantitatively through a suitable probability distribution. 'Uncertainty quantification', then, can involve one or more of several categories of tasks such as the forward (or propagation) problem, the inverse (or the model calibration) problem, sensitivity analysis, optimization under uncertainty etc., as discussed in Ch. 1. In practice, much of UQ amounts to complex high-dimensional density estimation tasks. While classical Monte Carlo (MC) approaches are very well-suited for high-dimensional density estimation, they require far too many evaluations of the computational solver to be feasible under a realistic simulation budget. Thus, one has to resort to constructing a surrogate of the computational budget which can make accurate predictions of the quantity of the interest while being cheap to evaluate. Under the high-dimensional setting, the curse of dimensionality provides an insurmountable obstacle to the task of constructing

accurate surrogates under limited simulation budgets. Our work in this dissertation has focused on developed machine learning based approaches to mitigate the curse of dimensionality by exploiting the underlying structure of the quantity of interest. Towards this end we developed approaches to construct Gaussian process and DNN surrogate models that encode both linear and nonlinear dimensionality reduction and have applied our proposed approaches to various challenging problems in high-dimensional surrogate modeling with reasonable success.

5.2 Open questions and future work

The work performed as part of this dissertation explore ideas around discovering intrinsic structure in high-dimensional functions to make uncertainty quantification feasible in such cases. Results, while encouraging, are a small step towards the goal of accurate and scalable data-driven uncertainty quantification. A holistic program for data-driven uncertainty quantification needs to incorporate:

1. Underlying intrinsic structure in quantities of interest,
2. Be amenable to incorporating data sourced from multiple simulators and/or experiments,
3. Any available information around the physics such as governing PDEs, invariances, symmetries etc., and,
4. Exploit group structure and latent structure in dynamical systems.

In this thesis we have tackled some aspects of items 1 and 2 from the (non-exhaustive) list above. Future work on the subject of high-dimensional UQ needs to focus on the question of how to incorporate physics into machine learning. This is a burgeoning area of research, popularized fairly recently, and is colloquially known as *physics-informed machine learning* (PIML) which we discuss in some further detail in Sec. 5.2.1. Furthermore, while the present work is agnostic to time-dependence in the

underlying model, there are properties (such as flow map semigroup structure [175] and Koopman operators [176]) exhibited in dynamical systems, specifically, which can be exploited to produce better predictive models. We elaborate upon these ideas in Sec. 5.2.2.

5.2.1 Physics-informed machine learning

To construct a computationally inexpensive surrogate model connecting uncertain input parameters (material properties/constitutive relations) to quantities of interest (solution field variables), one can develop a more data efficient methodology by leveraging physical information. For instance, a DNN surrogate such as the ones used in Ch. 3 can be regularized by the addition of additional penalty terms in the loss function which (softly) enforce the constraint the surrogate must satisfy the governing physics. Such constraints impose inductive biases on the DNN whose regularizing effect can dramatically reduce data-requirements (or even completely eliminate it). Consider, for instance, the task of propagating uncertainty through a stochastic PDE. The associated physics can be incorporated through the corresponding variational principle. A physical process governed by the PDE may be expressed as $\mathcal{A}[u(\mathbf{x})] = 0$, where, $\mathcal{A}[\cdot]$ is some nonlinear PDE operator. Given an appropriate Lagrangian density function $L(u, x)$, the solution of the aforementioned PDE can be estimate through the minimization of the energy functional - $I[u] = \int L(u, \nabla u, \mathbf{x})\mathrm{d}\mathbf{x}$ [177]. This is achieved by setting the first variation of the energy functional to zero, i.e., $\delta I = 0$. Doing so leads to the celebrated *Euler-Lagrange equations* [178]. A special case of this idea is the *Dirichlet principle* [177] - a variational principle for the solution of Laplace's equation. For discrete mechanical systems, this is referred to as the *principle of least action* [179], where the Lagrangian is defined as the sum of the kinetic and potential energy of the system as a function of the system's generalized coordinates. To account for the stochastic parameters in the problem, the energy functional is modified by introducing an additional integral over the uncertain parameters

- $I[u(\mathbf{x};\omega)] = \int_\Omega \int_\mathcal{X} L(\nabla u, u, \mathbf{x}, \omega)\mathrm{d}\mathbf{x}\mathrm{d}\omega$. Indeed, this is the approach we take in our recent work on UQ in stochastic elliptic PDEs [180] where we do not use *any* data obtained from runs of a computational simulator.

5.2.2 Group-theoretic and latent structure in high-dimensional stochastic dynamical systems

It is well-known that dynamical systems exhibit a 1-parameter Lie group symmetry in the time variable. For any given dynamical system there exists a group transformation operator g_t which evolves the state of the system, $\mathbf{z}$, from it's initial condition ($\mathbf{z}_0$) to it's state at time t ($\mathbf{z}_t$). Following the theory of Lie groups, g_t may be expressed as $g_t(\cdot) = \exp(t\mathbf{v})(\cdot)$, where $\mathbf{v} = \langle \boldsymbol{\xi}, \nabla_\mathbf{z} \rangle$ is the *infinitesimal generator* and $\boldsymbol{\xi}$ is a suitable vector field. Using a truncated approximation of the exponential map and a DNN representation of $\boldsymbol{\xi}$, one might attempt to learn the infinitesimal generator, either from experimental data on the state variables or through available physics (PDEs).

Long range simulation/forecasting in high-dimensional nonlinear dynamical systems is extremely challenging. A standard approach for making system amenable to prediction is to exploit intrinsic linear structure. See, for instance, recent works in [181–183]. A topic worthy of detailed exploration is the application of generative models to discover intrinsic low-dimensional latent structure of high-dimensional chaotic dynamical systems, such that the dynamics in the intrinsic space is linear and amenable to long range forecasting and prediction.

REFERENCES

REFERENCES

[1] Ralph C Smith. *Uncertainty quantification: theory, implementation, and applications*, volume 12. Siam, 2013.

[2] Xiaoping Du and Wei Chen. Efficient uncertainty analysis methods for multidisciplinary robust design. *AIAA journal*, 40(3):545–552, 2002.

[3] Francesco D'auria, Nenad Debrecin, and Giorgio Maria Galassi. Outline of the uncertainty methodology based on accuracy extrapolation. *Nuclear technology*, 109(1):21–38, 1995.

[4] William L Oberkampf, Jon C Helton, Cliff A Joslyn, Steven F Wojtkiewicz, and Scott Ferson. Challenge problems: uncertainty in system response given uncertain parameters. *Reliability Engineering & System Safety*, 85(1-3):11–19, 2004.

[5] Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

[6] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[7] William J Morokoff and Russel E Caflisch. Quasi-monte carlo integration. *Journal of computational physics*, 122(2):218–230, 1995.

[8] Andrea Barth, Christoph Schwab, and Nathaniel Zollinger. Multi-level monte carlo finite element method for elliptic pdes with stochastic coefficients. *Numerische Mathematik*, 119(1):123–161, 2011.

[9] Frances Y Kuo, Christoph Schwab, and Ian H Sloan. Quasi-monte carlo finite element methods for a class of elliptic partial differential equations with random coefficients. *SIAM Journal on Numerical Analysis*, 50(6):3351–3374, 2012.

[10] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation*, volume 89. siam, 2005.

[11] Ilias Bilionis, Beth A Drewniak, and Emil M Constantinescu. Crop physiology calibration in the clm. *Geoscientific Model Development*, 8(4):1071–1083, 2015.

[12] James C Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005.

[13] Ilias Bilionis and Phaedon-Stelios Koutsourelakis. Free energy computations by minimization of kullback–leibler divergence: An efficient adaptive biasing potential method for sparse representations. *Journal of Computational Physics*, 231(9):3849–3870, 2012.

[14] Ilias Bilionis and Nicholas Zabaras. A stochastic optimization approach to coarse-graining using a relative-entropy framework. *The Journal of chemical physics*, 138(4):044313, 2013.

[15] Anthony O'Hagan. Monte carlo is fundamentally unsound. *The Statistician*, pages 247–249, 1987.

[16] Bernhard Wieneke. Piv uncertainty quantification from correlation statistics. *Measurement Science and Technology*, 26(7):074002, 2015.

[17] Andrea Sciacchitano, Bernhard Wieneke, and Fulvio Scarano. Piv uncertainty quantification by image matching. *Measurement Science and Technology*, 24(4):045302, 2013.

[18] Aaron Boomsma, Sayantan Bhattacharya, Dan Troolin, Stamatios Pothos, and Pavlos Vlachos. A comparative experimental evaluation of uncertainty estimation methods for two-component piv. *Measurement Science and Technology*, 27(9):094006, 2016.

[19] D Boon, Richard Dwight, JJ Sterenborg, and Hester Bijl. Reducing uncertainties in a wind-tunnel experiment using bayesian updating. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*, page 1856, 2012.

[20] Uri M Ascher and Chen Greif. *A first course on numerical methods*, volume 7. Siam, 2011.

[21] DW Kelly, JP De SR Gago, OC Zienkiewicz, and I Babuska. A posteriori error analysis and adaptive processes in the finite element method: Part i—error analysis. *International journal for numerical methods in engineering*, 19(11):1593–1619, 1983.

[22] Michael B Giles and Endre Süli. Adjoint methods for pdes: a posteriori error analysis and postprocessing by duality. *Acta numerica*, 11:145–236, 2002.

[23] Ivo Babuska and Werner C Rheinboldt. A posteriori error analysis of finite element solutions for one-dimensional problems. *SIAM Journal on Numerical Analysis*, 18(3):565–589, 1981.

[24] Mark Ainsworth and J Tinsley Oden. *A posteriori error estimation in finite element analysis*, volume 37. John Wiley & Sons, 2011.

[25] Inseok Park, Hemanth K Amarchinta, and Ramana V Grandhi. A bayesian approach for quantification of model uncertainty. *Reliability Engineering & System Safety*, 95(7):777–785, 2010.

[26] Heng Xiao, J-L Wu, J-X Wang, Rui Sun, and CJ Roy. Quantifying and reducing model-form uncertainties in reynolds-averaged navier–stokes simulations: A data-driven, physics-informed bayesian approach. *Journal of Computational Physics*, 324:115–136, 2016.

[27] Nicholas Geneva and Nicholas Zabaras. Quantifying model form uncertainty in reynolds-averaged turbulence models with bayesian deep neural networks. *Journal of Computational Physics*, 383:125–147, 2019.

[28] Samuel Temple Reeve and Alejandro Strachan. Error correction in multi-fidelity molecular dynamics simulations using functional uncertainty quantification. *Journal of Computational Physics*, 334:207–220, 2017.

[29] Alejandro Strachan, Sankaran Mahadevan, Vadiraj Hombal, and Lin Sun. Functional derivatives for uncertainty quantification and error estimation and reduction via optimal high-fidelity simulations. *Modelling and Simulation in Materials Science and Engineering*, 21(6):065009, 2013.

[30] Timothy John Sullivan. *Introduction to uncertainty quantification*, volume 63. Springer, 2015.

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

[32] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[33] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[34] James C Spall. A one-measurement form of simultaneous perturbation stochastic approximation. *Automatica*, 33(1):109–112, 1997.

[35] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[36] Christopher Z Mooney. *Monte carlo simulation*, volume 116. Sage publications, 1997.

[37] Piero Baraldi and Enrico Zio. A combined monte carlo and possibilistic approach to uncertainty propagation in event tree analysis. *Risk Analysis*, 28(5):1309–1326, 2008.

[38] Klaus Mosegaard and Albert Tarantola. Monte carlo sampling of solutions to inverse problems. *Journal of Geophysical Research: Solid Earth*, 100(B7):12431–12447, 1995.

[39] Klaus Mosegaard and Malcolm Sambridge. Monte carlo analysis of inverse problems. *Inverse Problems*, 18(3):R29, 2002.

[40] Ilias Bilionis, Beth A Drewniak, and Emil M Constantinescu. Crop physiology calibration in the clm. *Geoscientific Model Development*, 8(4):1071–1083, 2015.

[41] James C Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005.

[42] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

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

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

[45] Ilias Bilionis, Nicholas Zabaras, Bledar A Konomi, and Guang Lin. Multi-output separable gaussian process: Towards an efficient, fully bayesian paradigm for uncertainty quantification. *Journal of Computational Physics*, 241:212–239, 2013.

[46] Peng Chen, Nicholas Zabaras, and Ilias Bilionis. Uncertainty propagation using infinite mixture of gaussian processes and variational bayesian inference. *Journal of Computational Physics*, 284:291–333, 2015.

[47] Ilias Bilionis and Nicholas Zabaras. Bayesian uncertainty propagation using gaussian processes. *Handbook of Uncertainty Quantification*, pages 1–45, 2016.

[48] Dongbin Xiu and Jan S Hesthaven. High-order collocation methods for differential equations with random inputs. *SIAM Journal on Scientific Computing*, 27(3):1118–1139, 2005.

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

[50] Dongbin Xiu. Efficient collocational approach for parametric uncertainty analysis. *Commun. Comput. Phys*, 2(2):293–309, 2007.

[51] Habib N Najm. Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics. *Annual review of fluid mechanics*, 41:35–52, 2009.

[52] Youssef M Marzouk, Habib N Najm, and Larry A Rahn. Stochastic spectral methods for efficient bayesian solution of inverse problems. *Journal of Computational Physics*, 224(2):560–586, 2007.

[53] Silvia Volpi, Matteo Diez, Nicholas J Gaul, Hyeongjin Song, Umberto Iemma, KK Choi, Emilio F Campana, and Frederick Stern. Development and validation of a dynamic metamodel based on stochastic radial basis functions and uncertainty quantification. *Structural and Multidisciplinary Optimization*, 51(2):347–368, 2015.

[54] GJA Loeven, JAS Witteveen, and H Bijl. A probabilistic radial basis function approach for uncertainty quantification. In *Proceedings of the NATO RTO-MP-AVT-147 Computational Uncertainty in Military Vehicle design symposium*, 2007.

[55] Panagiotis Tsilifis, Iason Papaioannou, Daniel Straub, and Fabio Nobile. Sparse polynomial chaos expansions using variational relevance vector machines. *arXiv preprint arXiv:1912.11029*, 2019.

[56] Eamonn Keogh and Abdullah Mueen. Curse of dimensionality. In *Encyclopedia of Machine Learning*, pages 257–258. Springer, 2011.

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

[58] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[59] David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

[60] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[61] Radford M Neal. Assessing relevance determination methods using delve. *Nato Asi Series F Computer And Systems Sciences*, 168:97–132, 1998.

[62] Roger G Ghanem and Pol D Spanos. Stochastic finite element method: Response statistics. In *Stochastic Finite Elements: A Spectral Approach*, pages 101–119. Springer, 1991.

[63] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

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

[65] Jesper Kristensen, Ilias Bilionis, and Nicholas Zabaras. Adaptive simulation selection for the discovery of the ground state line of binary alloys with a limited computational budget. In *Recent Progress and Modern Challenges in Applied Mathematics, Modeling and Computational Science*, pages 185–211. Springer, 2017.

[66] Herschel Rabitz and Ömer F Aliş. General foundations of high-dimensional model representations. *Journal of Mathematical Chemistry*, 25(2-3):197–233, 1999.

[67] Ömer F Alış and Herschel Rabitz. Efficient implementation of high dimensional model representations. *Journal of Mathematical Chemistry*, 29(2):127–142, 2001.

[68] Genyuan Li, Carey Rosenthal, and Herschel Rabitz. High dimensional model representations. *The Journal of Physical Chemistry A*, 105(33):7765–7777, 2001.

[69] Genyuan Li, Sheng-Wei Wang, Carey Rosenthal, and Herschel Rabitz. High dimensional model representations generated from low dimensional data samples. i. mp-cut-hdmr. *Journal of Mathematical Chemistry*, 30(1):1–30, 2001.

[70] Rajib Chowdhury, BN Rao, and A Meher Prasad. High-dimensional model representation for structural reliability analysis. *Communications in Numerical Methods in Engineering*, 25(4):301–337, 2009.

[71] Xiang Ma and Nicholas Zabaras. An adaptive high-dimensional stochastic model representation technique for the solution of stochastic partial differential equations. *Journal of Computational Physics*, 229(10):3884–3915, 2010.

[72] Jia Wei, Guang Lin, Lijian Jiang, and Yalchin Efendiev. Analysis of variance-based mixed multiscale finite element method and applications in stochastic two-phase flows. *International Journal for Uncertainty Quantification*, 4(6), 2014.

[73] Zhiwen Zhang, Xin Hu, Thomas Y Hou, Guang Lin, and Mike Yan. An adaptive anova-based data-driven stochastic method for elliptic pdes with random coefficient. *Communications in Computational Physics*, 16(2):571–598, 2014.

[74] Tony A Plate. Accuracy versus interpretability in flexible modeling: Implementing a tradeoff using gaussian process models. *Behaviormetrika*, 26(1):29–50, 1999.

[75] Cari G Kaufman, Stephan R Sain, et al. Bayesian functional {ANOVA} modeling using gaussian process prior distributions. *Bayesian Analysis*, 5(1):123–149, 2010.

[76] Nicolas Durrande, David Ginsbourger, Olivier Roustant, and Laurent Carraro. Additive covariance kernels for high-dimensional gaussian process modeling. *arXiv preprint arXiv:1111.6233*, 2011.

[77] David K Duvenaud, Hannes Nickisch, and Carl E Rasmussen. Additive gaussian processes. In *Advances in neural information processing systems*, pages 226–234, 2011.

[78] Elad Gilboa, Yunus Saatçi, and John P Cunningham. Scaling multidimensional inference for structured gaussian processes. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):424–436, 2015.

[79] XuanLong Nguyen and Alan E Gelfand. Bayesian nonparametric modeling for functional analysis of variance. *Annals of the Institute of Statistical Mathematics*, 66(3):495–526, 2014.

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

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

[82] Paul G Constantine. A quick-and-dirty check for a one-dimensional active subspace. *arXiv preprint arXiv:1402.3838*, 2014.

[83] Paul Constantine and David Gleich. Computing active subspaces with monte carlo. *arXiv preprint arXiv:1408.0545*, 2014.

[84] Paul G Constantine, Armin Eftekhari, and Michael B Wakin. Computing active subspaces efficiently with gradient sketching. *arXiv preprint arXiv:1506.04190*, 2015.

[85] Paul G Constantine. *Active subspaces: Emerging ideas for dimension reduction in parameter studies*, volume 2. SIAM, 2015.

[86] Paul G Constantine, Carson Kent, and Tan Bui-Thanh. Accelerating markov chain monte carlo with active subspaces. *SIAM Journal on Scientific Computing*, 38(5):A2779–A2805, 2016.

[87] Zachary J Grey and Paul G Constantine. Active subspaces of airfoil shape parameterizations. *AIAA Journal*, 56(5):2003–2017, 2018.

[88] Trent W Lukaczyk, Paul Constantine, Francisco Palacios, and Juan J Alonso. Active subspaces for shape optimization. In *10th AIAA Multidisciplinary Design Optimization Conference*, page 1171, 2014.

[89] Paul G Constantine, Brian Zaharatos, and Mark Campanelli. Discovering an active subspace in a single-diode solar cell model. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 8(5-6):264–273, 2015.

[90] Jennifer L Jefferson, James M Gilbert, Paul G Constantine, and Reed M Maxwell. Active subspaces for sensitivity analysis and dimension reduction of an integrated hydrologic model. *Computers & Geosciences*, 83:127–138, 2015.

[91] Paul G Constantine, Michael Emory, Johan Larsson, and Gianluca Iaccarino. Exploiting active subspaces to quantify uncertainty in the numerical simulation of the hyshot ii scramjet. *Journal of Computational Physics*, 302:1–20, 2015.

[92] Carsten Othmer, Trent W Lukaczyk, Paul Constantine, and Juan J Alonso. On active subspaces in car aerodynamics. In *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 4294, 2016.

[93] Zachary del Rosario, Paul Constantine, and Gianluca Iaccarino. Developing design insight through active subspaces. In *19th AIAA Non-Deterministic Approaches Conference*, page 1090, 2017.

[94] Andrew Glaws, Paul G Constantine, John N Shadid, and Timothy M Wildey. Dimension reduction in magnetohydrodynamics power generation models: Dimensional analysis and active subspaces. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 10(5):312–325, 2017.

[95] R-E Plessix. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, 167(2):495–503, 2006.

[96] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*, volume 105. Siam, 2008.

[97] Paul Geladi and Bruce R Kowalski. Partial least-squares regression: a tutorial. *Analytica chimica acta*, 185:1–17, 1986.

[98] Svante Wold, Michael Sjöström, and Lennart Eriksson. Pls-regression: a basic tool of chemometrics. *Chemometrics and intelligent laboratory systems*, 58(2):109–130, 2001.

[99] Richard G Brereton. *Chemometrics: data analysis for the laboratory and chemical plant*. John Wiley & Sons, 2003.

[100] Ana P Ferreira and Mike Tobyn. Multivariate analysis in the pharmaceutical industry: enabling process understanding and improvement in the pat and qbd era. *Pharmaceutical development and technology*, 20(5):513–527, 2015.

[101] Sanghong Kim, Manabu Kano, Hiroshi Nakagawa, and Shinji Hasebe. Estimation of active pharmaceutical ingredients content using locally weighted partial least squares and statistical wavelength selection. *International journal of pharmaceutics*, 421(2):269–274, 2011.

[102] Anthony O'Hagan. Bayes–hermite quadrature. *Journal of statistical planning and inference*, 29(3):245–260, 1991.

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

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

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

[106] Ilias Bilionis and Nicholas Zabaras. Multidimensional adaptive relevance vector machines for uncertainty quantification. *SIAM Journal on Scientific Computing*, 34(6):B881–B908, 2012.

[107] Prasanna Balaprakash, Robert B Gramacy, and Stefan M Wild. Active-learning-based surrogate models for empirical performance tuning. In *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, pages 1–8. IEEE, 2013.

[108] Donald R Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383, 2001.

[109] Michael Emmerich, Nicola Beume, and Boris Naujoks. An emo algorithm using the hypervolume measure as selection criterion. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 62–76. Springer, 2005.

[110] Kevin P Murphy. Machine learning: A probabilistic perspective. adaptive computation and machine learning, 2012.

[111] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[112] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[113] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[114] Simon S Haykin, Simon S Haykin, Simon S Haykin, and Simon S Haykin. *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, NJ, USA:, 2009.

[115] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[116] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[117] Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[118] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[119] Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *International conference on artificial intelligence and statistics*, 2016.

[120] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015.

[121] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[122] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[123] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[124] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[125] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[126] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[127] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[128] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch, 2017.

[129] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7, 2010.

[130] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[131] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artificial neural networks. *arXiv preprint arXiv:1707.03351*, 2017.

[132] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification. *arXiv preprint arXiv:1801.06879*, 2018.

[133] Allan Pinkus. *Ridge functions*, volume 205. Cambridge University Press, 2015.

[134] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[135] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[136] Heikki Haario, Marko Laine, Antonietta Mira, and Eero Saksman. Dram: efficient adaptive mcmc. *Statistics and computing*, 16(4):339–354, 2006.

[137] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[138] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[139] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[140] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822, 2014.

[141] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[142] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[143] Robb J Muirhead. *Aspects of multivariate statistical theory*, volume 197. John Wiley & Sons, 2009.

[144] Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

[145] Richard P Brent. *Algorithms for minimization without derivatives*. Courier Corporation, 2013.

[146] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[147] AN Lazaridi and VF Nesterenko. Observation of a new type of solitary waves in a one-dimensional granular medium. *Journal of Applied Mechanics and Technical Physics*, 26(3):405–408, 1985.

[148] VF Nesterenko. Propagation of nonlinear compression pulses in granular media. *Journal of Applied Mechanics and Technical Physics*, 24(5):733–743, 1983.

[149] Vitali Nesterenko. *Dynamics of heterogeneous materials*. Springer Science & Business Media, 2013.

[150] Christophe Coste, Eric Falcon, and Stephan Fauve. Solitary waves in a chain of beads under hertz contact. *Physical review E*, 56(5):6104, 1997.

[151] Upendra Harbola, Alexandre Rosas, Massimiliano Esposito, and Katja Lindenberg. Pulse propagation in tapered granular chains: An analytic study. *Physical Review E*, 80(3):031303, 2009.

[152] Joseph John II Lydon. *Nonlinear Effects in Granular Crystals with Broken Periodicity*. PhD thesis, California Institute of Technology, 2015.

[153] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[154] Peter J Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[155] Rohit K Tripathy and Ilias Bilionis. Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of computational physics*, 375:565–588, 2018.

[156] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006.

[157] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

[158] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[159] Prajit Ramachandran, Barret Zoph, and Quoc Le. Searching for activation functions. 2017.

[160] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[161] Yves Chauvin and David E Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology Press, 1995.

[162] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[163] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[164] Piyush Pandita, Ilias Bilionis, and Jitesh Panchal. Extending expected improvement for high-dimensional stochastic optimization of expensive black-box functions. *Journal of Mechanical Design*, 138(11):111412, 2016.

[165] Ronald L Iman. Latin hypercube sampling. *Encyclopedia of quantitative risk analysis and assessment*, 2008.

[166] Weixuan Li, Guang Lin, and Bing Li. Inverse regression-based uncertainty quantification algorithms for high-dimensional models: Theory and practice. *Journal of Computational Physics*, 321:259–278, 2016.

[167] Jonathan E Guyer, Daniel Wheeler, and James A Warren. Fipy: partial differential equations with python. *Computing in Science & Engineering*, 11(3), 2009.

[168] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *Preprint*, pages 1–57, 2016.

[169] Marc C Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

[170] Loic Le Gratiet. Bayesian analysis of hierarchical multifidelity codes. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):244–269, 2013.

[171] Paris Perdikaris, Maziar Raissi, Andreas Damianou, ND Lawrence, and George Em Karniadakis. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. In *Proc. R. Soc. A*, volume 473, page 20160751. The Royal Society, 2017.

[172] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[173] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

[174] Rohit Tripathy and Ilias Bilionis. Deep active subspaces: A scalable method for high-dimensional uncertainty propagation. In *ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection.

[175] Klaus-Jochen Engel and Rainer Nagel. *One-parameter semigroups for linear evolution equations*, volume 194. Springer Science & Business Media, 1999.

[176] Anastasiya Salova, Jeffrey Emenheiser, Adam Rupe, James P Crutchfield, and Raissa M D'Souza. Koopman operator and its approximations for systems with symmetries. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(9):093128, 2019.

[177] Lawrence C. Evans. *Partial differential equations*. American Mathematical Society, 2010.

[178] Izrail Moiseevitch Gelfand, Richard A Silverman, et al. *Calculus of variations*. Courier Corporation, 2000.

[179] Lev Davidovitch Landau and EM Lifshift. Mechanics. 2. 1969.

[180] Sharmila Karumuri, Rohit Tripathy, Ilias Bilionis, and Jitesh Panchal. Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks. *Journal of Computational Physics*, 404:109120, 2020.

[181] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

[182] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pages 4832–4839. IEEE, 2019.

[183] Craig Gin, Bethany Lusch, Steven L Brunton, and J Nathan Kutz. Deep learning models for global coordinate transformations that linearize pdes. *arXiv preprint arXiv:1911.02710*, 2019.

VITA

VITA

Rohit Kaushal Tripathy was born in Sunabeda, Odisha (India) in 1992. After graduating from high school in 2010, he entered the undergraduate program in Mechanical Engineering at VIT University in Vellore, India. In 2014, upon completion of his undergraduate degree, Rohit began a masters degree in the School of Mechanical Engineering at Purdue University, during which time he began working as a research assistant in the Predictive Science Lab (PSL) led by Prof. Ilias Bilionis. Rohit continued his work at PSL as a doctoral student beginning in January 2016. During his time at the PSL, Rohit worked on the development of data driven methods for uncertainty quantification. While a PhD student, he spent a summer as an intern at the Mathematics and Computer Science (MCS) division of Argonne National Laboratory and two summers working as an intern in Quantitative Research (QR) groups in JPMorgan & Chase. Rohit is graduating from the doctoral program at Purdue in May 2020 and is expected to begin a postdoctoral appointment at Cold Spring Harbor Laboratory with Prof. Peter Koo immediately after.