UNIVERSITY OF NOTTINGHAM

SCHOOL OF MATHEMATICAL SCIENCES

Improving Gaussian Process emulation using calibration-aware dimension reduction

Valentin Breaz

A thesis submitted to the University of Nottingham for the degree of

DOCTOR OF PHILOSOPHY

SEPTEMBER 2022

Abstract

The methods studied in this thesis are motivated by problems in the field of computer experiments, which involves the statistical analysis of computer codes (or computer simulators) implementing mathematical models that represent an underlying real-world process. Examples of complex real-world applications considered in this thesis are from epidemiology, hydrology, aerodynamics, and petroleum engineering.

Gaussian Processes (GPs) are flexible probabilistic models used in a variety of settings, such as regression, classification, and optimization. We have focused on their application in computer experiments, where they are mostly used as an emulator (or surrogate model) to approximate and eventually replace complex and expensive simulators.

The power of GPs comes from their ability to excell in small data and low-dimensional regime, being able to return accurate approximations of the simulator, together with calibrated predictive uncertainties. However, for simulators with high-dimensional inputs, they suffer from the curse of dimensionality, i.e., an exponential number of samples (with respect to the input dimensionality) is needed for accurate emulation. Fortunately, many simulators used in practice exhibit a low-dimensional structure that can be exploited by dimension reduction methods.

This thesis is concerned with how to best find and exploit this low dimensional structure when building GP emulators. We have focused on the (widespread) case where there exists a linear subspace, and tried to find the optimal projection onto this space. Many methods exist for finding the optimal projection, including Principal Component Analysis (PCA), Sufficient Dimension Reduction (SDR), and Active Subspaces (AS). We present a thorough comparison between all these methods in the context of GP emulation, as well as providing links between them and advice on which methods are

likely to work well in each situation. We have considered a variety of real-world simulators, from areas such such as aerodynamics and epidemiology.

Furthermore, we have also focused on GP emulation and dimension reduction for Bayesian inverse problems. For inverse problems, i.e., finding x that solves $y = f(x) + \epsilon_{noise}$, where y is a collection of noisy measurements (observations) and f is our simulator function, we do not need to approximate f globally, but only in some regions. Two successful algorithms for Bayesian inversion (i.e, finding $p(x|y)$ for some prior $p(x)$) are Randomized Maximum Likelihood (RML) and Ensemble Kalman Inversion (EKI).

RML consists of optimizing a number of highly correlated objective functions that depend on the same simulator function. In this regard, we develop a high-dimensional Bayesian optimization (HD-BO) approach to solve the RML problem based on GPs with dimension reduction via random embeddings. By sharing data between the different objective functions (i.e., using the common simulator evaluations $\{x_i, f(x_i)\}_{i=1}^{N_{train}}$ shared by the GP training sets for all the objectives), we are able to implement RML at a greatly reduced computational cost compared to existing methods, allowing us to efficiently sample from the posterior distribution $p(x|y)$ of the Bayesian inverse problem. We demonstrate the benefits of this approach in comparison to alternative optimization methods on a variety of real-world problems, including medical and fluid dynamics applications.

EKI are a family of (particle-based) methods which try to recover the unknown parameter in the classical inverse problem sense (i.e., x which generated the observations via $y = f(x) + \epsilon_{noise}$), although they are motivated from a Bayesian perspective as an iterative update from the prior distribution $p(x)$ towards the posterior $p(x|y)$. Firstly, we have investigated a new parametrization for the initial particles sampled from the prior distribution, which was not previously considered for EKI. Secondly, we have proposed a potential improvement regarding the selection of (GP) training points $\{x_i, f(x_i)\}_{i=1}^{N_{train}}$ for GP emulation within EKI. We demonstrate the benefits of our methodology on a groundwater modelling inverse problem.

Acknowledgments

I would like to thank my supervisors Richard Wilkinson, Marco Iglesias, and Domenico Bau for their continuous support. I would also like to express my gratitude to our collaborator Yan Chen, and TotalEnergies for funding our project. Without all these people, writing the thesis would not have been possible.

I would also like to thank the people from the School of Mathematical Sciences (University of Nottingham) and the School of Mathematics and Statistics (University of Sheffield) for all their help and many interesting conversations.

Last, but not least, I would to thank my family and friends, many of which I was very fortunate to meet in Sheffield and Nottingham. I will always be grateful for this opportunity.

CONTENTS

1 INTRODUCTION

The methods studied in this thesis are motivated by problems in the field of computer experiments [139], which involves the statistical analysis of computer codes implementing mathematical models that represent an underlying real-world process. Examples of complex real-world applications considered in this thesis are from epidemiology [44, 105], hydrology [68], aerodynamics [107], and petroleum engineering [46]. Many of these applications use partial differential equations (PDEs) as mathematical models; see [48] for a theoretical introduction, and [97] for an introduction to computational methods and their implementation.

1.1 INVERSE PROBLEMS

Our main area is on inverse problems, where the goal is to recover some unknown parameters linked to a collection of observed measurements; the book [162] contains a comprehensive introduction, with both theoretical results and numerical methods. Usually, the measurements are a collection of real values, which we denote by $\mathcal{D} \in \mathbb{R}^m$, such as spatio-temporal observations of water flow rates in a geological field [68]. The unknown parameters are usually denoted by $x \in \mathbb{R}^D$, and can correspond to the unknown spatial permeability values at D locations in the aforementioned geological field.

Typically, the link between x and $\mathcal{D}$ is represented via a Gaussian generative model (or a Gaussian likelihood)

$$\mathcal{D}|x \sim \mathcal{N}_m(f(x), \Sigma_{\text{obs}}),$$

where $f(x) : \mathbb{R}^D \to \mathbb{R}^m$ is called the *simulator* or the *computer code*; as discussed in the previous paragraph, $f(x)$ is often the computational model for the deterministic solution to a PDE modelling an underlying real-world process (e.g., the physical law between permeabilities and water flow values). The covariance matrix Σ_{obs} describes the modelling and observational errors, which are commonly assumed to be uncorrelated and homoscedastic ($\Sigma_{\text{obs}} = \sigma^2 I_m$). The inverse problems monograph for petroleum engineering [125] states that the observational errors are usually larger than the modelling errors in practical applications, although in other areas such as climate modelling [138] or cosmology [176], the modelling errors might be especially problematic.

1.2 BAYESIAN INVERSE PROBLEMS

We focus on Bayesian inverse problems (also known as Bayesian calibration), where the goal is to sample from the posterior distribution

$$p(x|\mathcal{D}) = \frac{p(\mathcal{D}|x)p(x)}{p(\mathcal{D})} \quad \text{(Bayes' rule)}$$

of the unknown parameters $x \in \mathbb{R}^D$ given the observed data $\mathcal{D} \in \mathbb{R}^m$. The work [162] contains a comprehensive survey from a theoretical perspective, and gives references to various practical applications. Note that the parameter space can be potentially infinite dimensional, e.g., a function of location $x : [0, 1]^2 \to \mathbb{R}$ in case of a 2D geostatistical application [86], but Bayes' rule (or Bayes' theorem) can be adapted to functional spaces [41].

The distribution $p(\mathcal{D})$ is known as the marginal likelihood or the model evidence, and it is never computed in this work. The prior distribution $p(x)$ is a probabilistic representation of our prior uncertainty about the unknown parameters, and is often approximated as a Uniform or Gaussian distribution. While this might seem like a restrictive assumption, note that even if both the prior and the likelihood $p(\mathcal{D}|x)$ are Gaussian, the posterior is non-Gaussian unless the simulator function $f(x)$ is linear; the proof for this result is standard, and it can be found for example in the collection of lecture notes [140]. We only consider non-linear simulators in this work, although linear Bayesian inverse problems is also an active area of research (see [51] for a theoretical monograph, and [165] for a recent gravimetrical application).

Going back to our goal of sampling from the posterior distribution, we outline three families of methods that are used throughout the thesis. Firstly, Markov Chain Monte Carlo (MCMC) methods, where a Markov chain is simulated such that the samples generated are asymptotically exact draws from the posterior distribution; see [137] for a comprehensive introduction on computational methods, and [114] for a standard book on theoretical foundations.

Secondly, the Randomized Maximum Likelihood (RML) algorithm, which was introduced by [124] as an approximate posterior sampling methodology. The procedure is usually formulated for the case when both the prior and the likelihood are Gaussian; RML proceeds by first perturbing the data $\mathcal{D}_n \sim \mathcal{N}_m(\mathcal{D}, \Sigma_{\text{obs}})$ and the prior mean $\mu_n \sim \mathcal{N}_D(\mu, \Sigma)$, followed by maximizing the resulting un-normalized log-posterior density

$$O_n(x) := \log \mathcal{N}_m(f(x)|\mathcal{D}_n, \Sigma_{\text{obs}}) + \log \mathcal{N}_D(x|\mu_n, \Sigma) \tag{1}$$

with respect to these perturbations. The resulting solution $x_n^\star = \arg\max_x O_n(x)$ is an approximate sample from the posterior distribution; the sample is an exact draw from the posterior only when the simulator $f(x)$ is linear, as discussed in the seminal paper [124]. Nonetheless, the RML samples have shown good practical performance in petroleum engineering applications with various non-linear simulators [55, 104, 161]. For recent methodological advances that improve the accuracy of the RML samples with respect to the true posterior for highly nonlinear simulators, see [10, 123, 7].

Finally, we mention the particle-based Ensemble Kalman Inversion (EKI) algorithms. While these methods try to recover the unknown parameter in the classical inverse problem sense (i.e., x_{true} which generated the observations via $\mathcal{D} \sim \mathcal{N}_m(f(x_{true}), \Sigma_{\text{obs}}))$, they are motivated from a Bayesian perspective as an iterative Kalman filter type update from the prior distribution towards the posterior [87, 82]. See [140] for an introduction to Kalman filter methods and for a vanilla version of the EKI algorithm, [82] for a recent survey of new EKI versions, and [87] for a recent application in material sciences. An alternative line of work targets the full posterior distribution $p(x|\mathcal{D})$ in a methodology known as the Ensemble Kalman Sampler (EKS) [57], together with its alternative version [60] and the recent improvements presented in [136].

1.3 COMPUTATIONAL CHALLENGES

To successfully apply these methods, often requires a large amount of computation. As exemplified in the methods comparison paper for posterior sampling in petroleum engineering [46], the number of simulator evaluations $f(x)$ required is as follows: $\propto 10^6$ in case of MCMC, $\propto 10^4$ is case of RML, and $\propto 10^3$ in case of EKI, respectively. These numbers can be prohibitively large in practical applications such as climate models, where a standard computer can take several days to produce a single simulator evaluation $f(x)$ [81]. Nonetheless, the smaller number of simulations required in comparison with MCMC and RML makes EKI a very popular choice in practice. Apart from the areas of application mentioned already, EKI has been successfully used in medical imaging [82] and petroleum engineering [83].

One way in which MCMC and RML can potentially be accelerated is by using gradient evaluations of the simulator. One popular gradient-based MCMC method is Hamiltonian Monte Carlo (HMC), where the Markov chain evolves according to Hamiltonian dynamics. This method is often employed when the dimensionality of $x \in \mathbb{R}^D$ is high; see [14] for an introduction, and [89] for a recent application in Machine Learning, where posterior sampling of the parameter space in very high-dimensional Bayesian (Artificial) Neural Network models was achieved. RML can benefit from the famous gradient-descent optimization algorithm, or the Limited-Memory BFGS (L-BFGS) algorithm [102], which is a quasi-Newton method that uses estimates of the inverse Hessian (second derivative). L-BFGS was used for RML in the petroleum engineering experiments from [76]. However, gradient information is often unavailable for many simulators used in practice; although automatic differentiation methods are being developed [183, 110], many institutions may lack resources to re-write complex computer codes. As a result, we focus on gradient-free methods throughout this thesis.

1.4 gaussian process (gp) emulation

Another way in which we can accelerate MCMC, RML and EKI is to replace the simulator (or in certain cases, the likelihood or the log-likelihood [93]) by a computationally inexpensive approximation known as an emulator, surrogate model, or meta-model; we will use the term 'emulator' for the rest of the thesis. The emulators will be simple enough to enable them to be queried $\propto 10^6$ times in reasonable computational time, as often required for certain MCMC approaches.

While models that explicitly encode knowledge about the mathematical model of the simulator $f(x)$ when building an emulator are increasingly popular (e.g. the physics-informed (artificial) neural networks [133]), we have chosen a standard black-box approach, i.e. we only require access to N simulator evaluations $\{x_i, f(x_i)\}_{i=1}^N$ in order to build our emulator, and we do not (explicitly) exploit the intrinsic structure of $f(x)$.

Most of the thesis is devoted to the study of Gaussian Process (GPs) emulation models; for the seminal work on using GP emulators for computer simulators, see [94]. Occasionally, we will also discuss (artificial) neural network (NN) models; see [191] for the use of NN emulators in a geological application. Our main motivation is that GPs have been successfully used to accelerate MCMC [35], RML [78], and EKI [91]. Apart from these three inversion methods considered in the thesis, both GPs and NNs have been successfully used in various alternative procedures for solving Bayesian inverse problems (see [174, 138] for GPs, and [5, 73, 149] for NNs).

The goal of GP emulators is to provide a good probabilistic prediction for the unknown value $f(x_*)$ given a new input x_*, which lays outside the set $\{x_i, f(x_i)\}_{i=1}^N$ that was used to build the emulator (also known as the training set). The probabilistic prediction is in the form of a Gaussian distribution $\mathcal{N}_m(f_{\text{GP}}(x_*), \sigma^2_{\text{GP}}(x_*))$, where $f_{\text{GP}}(x_*)$ is our prediction for $f(x_*)$, and $\sigma^2_{\text{GP}}(x_*)$ represents the predictive uncertainty (in other words, the uncertainty about our approximation $f_{\text{GP}}(x_*) \approx f(x_*)$). The textbook [135] contains a comprehensive introduction, which also includes the use of GP approximations in Machine Learning applications. Note that the output dimensionality m ($f(x) \in \mathbb{R}^m$) can raise additional challenges for GP emulators, as the correlation between different outputs can be difficult to model; see [17] for a standard multi-output GP approach, and [138, 190] for recent methodologies designed for high-dimensional outputs. In this work, we avoid these considerations by either looking at simulators with a one-dimensional output ($f(x) \in \mathbb{R}$), emulating the log-likelihood instead of the simulator ($\log \mathcal{N}_m(f(x)|\mathcal{D}, \Sigma_{\text{obs}}) \in \mathbb{R}$), or by simply treating all outputs as independent (and thus building m standard GP emulators with a one-dimensional output).

The training points $\{x_i, f(x_i)\}_{i=1}^N$ are usually selected via a random design (i.e. the inputs $x_i \sim p(x)$ are sampled independently at random from the prior), or by using a space-filling design such as a Latin hypercube [139]; both of these are examples of fixed design methods. A different approach is to use a goal-oriented design for solving Bayesian inverse problems. In this context, the selection of training points is guided towards the high-posterior density regions, as we want out emulator to be particularly accurate in those regions. Indeed, the work [160] shows that a goal-oriented design that targets simulator queries $\{x_i, f(x_i)\}_{i=1}^N$ from the high-posterior density regions leads to a more accurate approximation of the log-likelihood compared with a Latin hypercube design. Alternative goal-oriented designs for Bayesian inverse problems can be found in [93, 170, 3]. We will use both fixed and goal-oriented designs throughout the thesis.

1.5 HIGH-DIMENSIONAL GP EMULATORS AND DIMENSION REDUCTION

GP emulators belong to the family of nonparametric regression methods, where an approximation $f_{\mathrm{GP}}(x) \approx f(x)$ built from $\{x_i, f(x_i)\}_{i=1}^N$ is desired, without assuming a parametric form for $f_{\mathrm{GP}}(x)$; see [172] for a standard textbook on nonparametric statistical methods. GPs depend on a collection of hyperparameters; these can be fixed to some arbitrary values, but are usually estimated using the training data.

From a theoretical perspective, GPs (as any other nonparametric regression method) are known to suffer from the so called 'curse of dimensionality'. This typically means that the number of training points required for an accurate approximation of the simulator function $f : \mathbb{R}^D \to \mathbb{R}^m$ increases exponentially with the input dimension D [158]. Note that in practice the situation is even more difficult; typically, we estimate the GP hyperparameters via a high-dimensional non-convex optimization problem. As demonstrated in [56, 151], the landscape of the optimization problem can also be multi-modal. This can potentially lead to selecting unsatisfactory hyperparameter values; indeed, [96] shows that the predictive uncertainty is underestimated in various numerical experiments. There are certain cases in which the curse of dimensionality can be theoretically alleviated. For example, when the input distribution $p(x)$ lies close to or exactly on a lower dimensional manifold, as discussed in [182, 77, 66]; these GP models will not be considered in the thesis.

Fortunately, as [31] pointed out, many simulators (or log-likelihoods) used in Bayesian inverse problems have a latent low-dimensional linear structure, i.e.

$$f(x) \approx g(A^T x) \tag{2}$$

for some low-dimensional link function $g : \mathbb{R}^d \to \mathbb{R}^m$ with $d \ll D$, and $A \in \mathbb{R}^{D \times d}$ is a semi-orthogonal matrix (i.e., $A^T A = I_d$) which gives the linear low-dimensional structure. There are various ways in which such a structure can be discovered, using the family of methods known as likelihood informed subspaces (see [38] for a recent survey). In this work, we will outline one particular method known as active subspaces (AS), which was introduced together with a GP emulation approach ($f(x) \approx f_{\mathrm{GP}}(A^T x)$) in [30]. Since then, active subspaces have been successfully used as a dimension reduction tool for GP emulation in many applications such as geophysics [108], engineering [134], COVID-19 models [181], and Ebola spread modelling [127].

The standard approach for constructing the active subspace is to use gradient evaluations of the simulator [30]. As mentioned in Section 1.3, we focus on gradient-free methods in this work, and thus we often consider the GP emulation performance with the active subspace method as a 'gold standard', which is used to benchmark the performance of alternative (gradient-free) dimension reduction methods; this approach was also pursued in [103]. We will consider various gradient-free methods which can be used to estimate the latent low-dimensional linear structure in (2), such as Sufficient Dimension Reduction (SDR); see [103] for a comparison between different SDR methods for GP emulation, which include the classical Sliced Inverse Regression (SIR) algorithm [101] and the modern gradient kernel dimension reduction (gKDR) algorithm [54]. Another approach considered is the family of embedding learning methods, which try to estimate the low-dimensional linear structure A together with the rest of GP hyperparameters during training, in order to achieve $f(x) \approx f_{\mathrm{GP}}(A^T x)$ [169, 62, 166, 144, 134, 63].

The methods presented above are *supervised* dimension reduction approaches, in the sense that they require simulator evaluations $\{f(x_i)\}_{i=1}^N$, or gradients of the simulator $\{\nabla f(x_i)\}_{i=1}^M$ in the case of active subspaces. We will also consider an alternative family of methods known as *unsupervised* dimension reduction; these methods do not require access to the simulator in order to perform dimension reduction, and thus are particularly desirable for expensive simulators. The most popular unsupervised dimension reduction method is Principal Component Analysis (PCA), which only uses the prior covariance $\mathbb{C}\text{ov}[X \sim p(x)]$ and is discussed at length in the dimension reduction monograph [18]. In the case of a Gaussian prior $p(x) \sim \mathcal{N}_D(\mu, \Sigma)$, given that the eigenvalues of Σ decay sufficiently fast and that $f(x)$ is Lipschitz continuous, [184] proves that the matrix $A \in \mathbb{R}^{D\times d}$ selected by PCA satisfies the property (2), for some low-dimensional link function $g : \mathbb{R}^d \to \mathbb{R}$ with $d \ll D$. Another unsupervised option is to use random low-dimensional linear projections of the inputs; [43] shows that one such method can be useful for GP emulation when the direct approach of using a high-dimensional GP emulator fails.

1.6 OUR CONTRIBUTION

In Chapter 2, we will formally introduce GP emulators, together with the dimension reduction methods PCA and AS. Also, we will present a case study based on an Elliptic PDE simulator, which was used to demonstrate the clear benefits of AS methods for GP emulation when an active subspace exists and can be found [30], as well as the potential advantages of PCA in comparison with the baseline high-dimensional GP emulator (with no dimension reduction) [103]. In addition to the existing results, we will present a collection of new experiments, where we discover a new direction of study on the advantages of PCA for GP emulation versus the high-dimensional baseline.

In Chapter 3, we will introduce the additional supervised and unsupervised dimension reduction methods considered in this thesis, such as sufficient dimension reduction (SDR) and various random projections. While we include some theoretical properties, the main goal of this chapter is to study the performance of these methods on a variety of synthetic and real-world simulators, from areas such such as aerodynamics [107] and epidemiology [105].

In Chapter 4, we will formally introduce Bayesian inverse problems, while focusing on the emulation based solution of replacing the log-likelihood with a GP emulator. We provide an embedding learning approach of exploiting the active subspace structure for log-likelihoods, coupled with an MCMC procedure for solving the resulting Bayesian inverse problems; this procedure is adapted from [31]. We present very promising results on a quadratic toy model, as well as on a more challenging high-dimensional Elliptic PDE.

In Chapter 5, we develop a high-dimensional Bayesian optimization (HD-BO) approach for approximate posterior sampling via Randomized Maximum Likelihood (RML). We will include a thorough introduction for both HD-BO and RML. By sharing data between the different objective functions (1), we are able to implement RML at a greatly reduced computational cost compared to existing methods. We demonstrate the benefits of this approach in comparison to alternative (gradient-free) optimization methods on a variety of synthetic and real-world problems, including medical [44, 105] and fluid dynamics applications [71]. Furthermore, we show that the samples produced by our method cover well the high-posterior density regions in all of the experiments.

In Chapter 6, we will introduce the Ensemble Kalman Inversion framework (EKI). We have developed an EKI approach based on a (Bayesian) Neural Network parametrization of Gaussian random field priors for spatial maps of parameters $x : [0, 1]^2 \to \mathbb{R}$ [143]. For this parametrization, we present results on a two-phase (oil-water) Darcy flow simulator regarding permeability inversion in petroleum engineering [46, 83]. Furthermore, we have investigated the performance of GP emulators within EKI by adapting a new goal-oriented design from History Matching [59], which is another area of inverse problems methods [175, 168]. In a groundwater modelling application, we show that our approach provides significant improvements over the existing methodology [91], which is based on a random design.

Finally, in Chapter 7, we present the final conclusions for every chapter, together with various directions for future work.

2

GAUSSIAN PROCESSES AND DIMENSION REDUCTION - INTRODUCTION AND NEW RESULTS ON AN ELLIPTIC PDE CASE STUDY

The purpose of this chapter is twofold. Firstly, it contains an introduction to Gaussian Processes (GPs), Principal Component Analysis (PCA), and Active Subspaces (AS); these notions are central for the remaining chapters of the thesis. Secondly, we present a case study, which was used to investigate these methods in the existing literature [30, 103]. We complement the existing results with our new findings.

2.1 GAUSSIAN PROCESSES

As discussed in Chapter 1, Gaussian Processes (GPs) are used throughout the thesis in the context of Bayesian inverse problems, where they replace a potentially expensive computer simulator $f : \mathbb{R}^D \to \mathbb{R}^m$ or the log-likelihood $L(x) := \log p(\mathcal{D}|x) = \log \mathcal{N}_m(\mathcal{D}|f(x), \Sigma_{\text{obs}}) \in \mathbb{R}$.

Definition 2.1 *([135], Definition 2.1) A Gaussian Process $f_{GP}(x)$ is a collection of random variables such that for any finite number N and any inputs $\{x_n\}_{n=1}^N$, the random variables $\{f_{GP}(x_n)\}_{n=1}^N$ have a joint Gaussian distribution. The process is completely specified by its mean function $m(x) := \mathbb{E}[f_{GP}(x)]$ and its covariance function $k(x, x') := \mathbb{C}\text{ov}[f_{GP}(x), f_{GP}(x')]$, and is usually written as $f_{GP}(x) \sim GP(m(x), k(x, x'))$.*

For each output $[f(x)]_j$ of the simulator $f(x) := ([f(x)]_1, \dots, [f(x)]_m)^T$, where $[f(x)]_j : \mathbb{R}^D \to \mathbb{R}$, we assign a Gaussian Process prior

$$[f_{\mathrm{GP}}(x)]_j \sim \mathrm{GP}(m_j(x), k_j(x, x')) \tag{3}$$

to encode our prior beliefs about the regularity of $[f(x)]_j$. Ideally, samples $g(x) \sim [f_{\mathrm{GP}}(x)]_j$ ($g(x) : \mathbb{R}^D \to \mathbb{R}$) from this GP prior will have similar properties to $[f(x)]_j$; examples of such regularity proprieties are continuity, smoothness, and periodicity. See Algorithm 1 in [92] for a concrete algorithm for producing GP prior samples. Conditioned on the training data $\mathcal{X}_j := \{x_i, [f(x_i)]_j\}_{i=1}^N$, the posterior $[f_{\mathrm{GP}}(x)]_j | \mathcal{X}_j$ is still a Gaussian Process, which represents our posterior beliefs about the simulator $[f(x)]_j$ across the entire input domain $\mathbb{R}^D$. According to Theorem 3.3 in [92]:

$$[f_{\mathrm{GP}}(x)]_j | \mathcal{X}_j \sim \mathrm{GP}(\bar{m}_j(x), \bar{k}_j(x, x')),$$

with the posterior mean $\bar{m}_j(x)$ and covariance function $\bar{k}_j(x, x')$ given by

$$\bar{m}_j(x) = m(x) + k_{xX} k_{XX}^{-1} (Y_j - m_X),$$

$$\bar{k}_j(x, x') = k(x, x') - k_{xX} k_{XX}^{-1} k_{Xx'},$$

where $Y_j := ([f(x_1)]_j, \dots, [f(x_N)]_j)^T$ denotes the training outputs and $m_X := (m_j(x_1), \dots, m_j(x_N))^T$ is the prior mean evaluated at the training inputs; k_{XX} is an $N \times N$ matrix with elements $(k(x_i, x_j))_{i=1, j=1}^N$, which represents the (symmetric) covariance function evaluated at the training inputs $X := \{x_i\}_{i=1}^N$, whereas $k_{xX} = k_{Xx}^T := (k(x, x_1), \dots, k(x, x_N))^T$ is the covariance function evaluated at any input x and the training inputs X.

Given a new location $x_\star$ outside the training set, the GP posterior distribution $\mathcal{N}(\bar{m}_j(x_\star), \bar{k}_j(x_\star, x_\star))$ can be quickly computed and represents our fast probabilistic prediction for the expensive simulator evaluation $[f(x_\star)]_j$. Namely, the predictive mean $\bar{m}_j(x_\star)$ is our approximation for $[f(x_\star)]_j$, whereas the predictive standard deviation $\sqrt{\bar{k}_j(x_\star, x_\star)}$ quantifies the uncertainty associated with this approximation. By performing this prediction, we avoid the computational burden of evaluating $[f(x_\star)]_j$. Note that for large training sets $N \geq 10^4$ (not considered in this thesis), efficient methods for approximating $\bar{m}_j(x_\star)$ [177, 178] and $\bar{k}_j(x_\star, x_\star)$ [131] have been recently developed.

Throughout the thesis, the prior mean function $m(x)$ will always have a simple structure such as linear $m(x) = Ax + b$ or constant $m(x) = C$. The covariance function (also known as the kernel function) will always be the standard squared-exponential (or the Radial Basis Function (RBF) kernel)

$$\begin{aligned} k_j(x, x') :&= \mathbb{C}\text{ov}[[f_{\text{GP}}(x)]_j, [f_{\text{GP}}(x')]_j] \\ &= \sigma^2 \exp\Big(- \frac{||x - x'||_2^2}{2l^2} \Big), \end{aligned} \tag{4}$$

where $||x - x'||_2^2 := \sum_{i=1}^{D} (x_i - x_i')^2$, the hyperparameter σ^2 is the signal variance, and $l > 0$ is the lengthscale hyperparameter. Note that the correlation between points only depends on the distance $||x - x'||_2 / l$. Roughly speaking, the lengthscale l models how fast $[f(x)]_j$ varies with x, with σ^2 modelling the amplitude of these variations; if l is small, we expect a potentially large difference in the corresponding outputs $[f(x)]_j$ and $[f(x')]_j$ for nearby points x and x', whereas a large value of l corresponds to a slow variation between $[f(x)]_j$ and $[f(x')]_j$ for two distant points x and x'. The covariance function (4) also encodes an infinitely differentiable smoothness assumption for $[f(x)]_j$, i.e. the samples $g(x) \sim [f_{\text{GP}}(x)]_j$ are infinitely differentiable.

It is important to mention that we make these choices for $\big(m(x), k_j(x, x')\big)$ for the ease of exposition and comparison. Nonetheless, the methods presented in this thesis are not specific to these choices, and we could easily use other kernels and mean functions. See Chapter 4 in [135] for alternative covariance functions such as the Matérn family (which encode various degrees of smoothness), non-stationary covariance functions (e.g. where the lengthscale hyperparameter varies with x), or periodic covariance functions (which can be combined with non-periodic covariance functions in order to capture different trends in $[f(x)]_j$). Regarding the prior mean, a quadratic function was preferred in various works [30, 174].

In this work, the hyperparameters $\theta_C := \{C, \sigma, l\}$ (for a constant prior mean $m(x) = C$) or $\theta_{A,b} := \{A, b, \sigma, l\}$ (for a linear prior mean $m(x) = Ax + b$) are selected by maximizing the marginal likelihood

$$p(Y_j|\theta) \sim \mathcal{N}(Y_j|m_X, k_{XX}), \tag{5}$$

which is the probability density function (PDF) of the Gaussian distribution $\mathcal{N}(m_X, k_{XX})$ evaluated at the observed outputs $Y_j := \{[f(x_i)]_j\}_{i=1}^N$. As defined previously, $X := \{x_i\}_{i=1}^N$ are the training locations, m_X is the prior mean evaluated at X, and k_{XX} is the prior covariance matrix $(k(x_i, x_j))_{i=1,j=1}^N$. This approach is known as type II maximum likelihood (ML-II); we will discuss alternative methods for selecting the hyperparameters towards the end of this section. For simplicity, we write θ to represent both θ_C and $\theta_{A,b}$, while later in the thesis we will specify separately the prior mean structure considered. To expand on (5), we need to maximize

$$\log p(Y_j|\theta) = -\frac{1}{2}(Y_j - m_X)^T K_{XX}^{-1}(Y_j - m_X) - \frac{1}{2}\log\det(K_{XX}) - \frac{n}{2}\log 2\pi \tag{6}$$

with respect to θ, where the log marginal likelihood is preferred for numerical optimization. As discussed in Chapter 1, the optimization landscape (6) is a non-convex function of θ, and thus we need to resort to local optimization methods in order to find an approximate maximizer θ_{MLE}. The most common solution is to use gradient-based optimization methods such as the standard gradient descent algorithm, or the Limited-Memory BFGS (L-BFGS) algorithm [102], which is a quasi-Newton method that uses estimates of the inverse Hessian (second derivative) of (6); both of these algorithms will be employed throughout the thesis. The gradients of (6) with respect to the kernel hyperparameters σ and l can be found in the equation (5.9) of [135]:

$$\begin{aligned}\frac{\partial \log p(Y_j|\theta)}{\partial \theta_i} &= \frac{1}{2}(Y_j - m_X)^T K_{XX}^{-1}\frac{\partial K_{XX}}{\partial \theta_i}K_{XX}^{-1}(Y_j - m_X) - \frac{1}{2}\mathrm{tr}(K_{XX}^{-1}\frac{\partial K_{XX}}{\partial \theta_i}) \\ &= \frac{1}{2}\mathrm{tr}\Big((\alpha\alpha^T - K_{XX}^{-1})\frac{\partial K_{XX}}{\partial \theta_i}\Big),\end{aligned} \tag{7}$$

where $\theta_i \in \{\sigma, l\}$, $\alpha := K_{XX}^{-1} Y_j$, $\mathrm{tr}(\cdot)$ is the trace of a square matrix (i.e. the sum of the elements on the main diagonal, which is from the upper left to the lower right), and $\frac{\partial K_{XX}}{\partial \theta_i}$ is the matrix of elementwise derivatives. In order to derive (7) from (6), the matrix identities $\frac{\partial K_{XX}^{-1}}{\partial \theta_i} = -K_{XX}^{-1} \frac{\partial K_{XX}}{\partial \theta_i} K_{XX}^{-1}$ and $\frac{\partial}{\partial \theta_i} \log \det(K_{XX}) = \mathrm{tr}(K_{XX}^{-1} \frac{\partial K_{XX}}{\partial \theta_i})$ for the (invertible) positive definite symmetric matrix K_{XX} were used (see Appendix A.3.1 of [135]).

The unknown hyperparameters θ can also be treated as latent random variables, by assigning a (prior) probability distribution $\theta \sim p(\theta)$. Given the posterior distribution $p(\theta | Y_j)$ according to the marginal likelihood (5), the latent variables can be integrated out, in order to obtain the predictive distribution for some unknown value $[f(x_\star)]_j$. This predictive distribution is different from the standard GP posterior $\mathcal{N}(\bar{m}_j(x_\star), \bar{k}_j(x_\star, x_\star))$ obtained using the fixed set of hyperparameters θ_{MLE} which (approximately) maximize the log marginal likelihood. Indeed, the predictive distribution in the latent variable case is non-Gaussian, unless we fix θ_{MAP} to be the maximum a posteriori (MAP) estimate $\theta_{MAP} = \arg\max_\theta p(\theta | Y_j)$. The full posterior distribution $p(\theta | Y_j)$ cannot be analytically computed, although there exists conjugate priors for the prior mean and signal variance hyperparameters [122]. We thus need to resort to sampling methods [56, 58, 151], or alternative methods such as variational inference. In the latter case, $p(\theta | Y_j)$ is replaced by its best approximation from a parametric family of distributions (e.g. Gaussian [96]), or the non-parametric approximation provided by the Stein variational gradient descent (SVGD) method [129]. While computationally more demanding, latent variable approaches have shown improved performance compared with the ML-II procedure in various numerical experiments [58, 96, 151], particularly with regard to better calibrated predictive uncertainties.

Another alternative for selecting the GP hyperparameters θ is via cross-validation; see Chapter 5 in [135] for a thorough introduction. In the case of leave-one-out cross-validation (LOO-CV), we write $\mathcal{X}_j^{-i} := \mathcal{X}_j \backslash \{x_i, [f(x_i)]_j\}$

for the training set obtained by removing the observation $\{x_i, [f(x_i)]_j\}$. The goal is to maximize the average predictive log probability

$$\frac{1}{N}\sum_{i=1}^{N} \log \mathcal{N}([f(x_i)]_j | \bar{m}_j^{-i}(x_i), \bar{k}_j^{-i}(x_i, x_i)) \tag{8}$$

with respect to θ, where $\mathcal{N}(\bar{m}_j^{-i}(x_i), \bar{k}_j^{-i}(x_i, x_i))$ is the predictive distribution for $[f(x_i)]_j$ corresponding to the GP posterior $[f_{\mathrm{GP}}(x)]_j | \mathcal{X}_j^{-i}$. The objective function (8) is also known as the log-pseudo likelihood. The work [9] shows that LOO-CV outperforms ML-II in various numerical experiments, where the prior assumptions encoded by the covariance function $k_j(x, x')$ are misspecified for $[f(x)]_j$. For a recent consideration of multiple-fold cross validation, in which multiple observations are removed form the training set, see [69].

2.2 PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal Component Analysis (PCA) is a dimension reduction method which dates back to the beginning of the 20^{th} century [128], and since then it has been employed in numerous statistical and engineering applications. For examples where PCA has been used to reduce the input dimensionality in case of GP emulation, see [15, 103, 163].

Let us consider a high dimensional random vector $x \sim p_D(x)$, with the covariance matrix written as $\mathbb{C}\mathrm{ov}[x] \in \mathbb{R}^{D\times D}$. Since $\mathbb{C}\mathrm{ov}[x]$ is symmetric positive semidefinite, we can write its eigendecomposition as $\mathbb{C}\mathrm{ov}[x] := U\Lambda U^T$, where all its eigenvalues $\lambda_1 \geq \cdots \geq \lambda_D \geq 0$ are non-negative real numbers, and the corresponding eigenvectors $U := [u_1, \ldots, u_D]$ are orthonormal. In order to reduce dimensionality from D to any $d \leq D$, PCA extracts the d eigenvectors of $\mathbb{C}\mathrm{ov}[x]$ corresponding to the largest d eigenvalues. We write $U_{PCA} := [u_1, \ldots, u_d] \in \mathbb{R}^{D\times d}$ for the resulting dimension reduction matrix, whose columns are the eigenvectors mentioned above. In some cases, the covariance matrix is not available analytically, and we only have access to samples $\{x_i\}_{i=1}^N \sim p_D(x)$. In this context, PCA uses the empirical covariance matrix in the same way, i.e. by extracting the d dominant eigenvectors. We can write the emprical covariance matrix as $\frac{1}{N}ZZ^T$ (or $\frac{1}{(N-1)}ZZ^T$ for an unbiased estimate of the covariance), where $Z := [z_1, \ldots, z_N] \in \mathbb{R}^{D \times N}$ and $z_i = x_i - \frac{1}{N}\sum_{i=1}^N x_i$ for $i \in \{1, \ldots, N\}$. In case of a Gaussian distribution $x \sim \mathcal{N}_D(\mu, \Sigma)$, we can arrange the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D \geq 0$ of the covariance matrix Σ in non-increasing order, and define the amount of variance that is captured by the first j eigenvalues using the corresponding cumulative ratio, namely $E_j := \frac{\sum_{i=1}^j \lambda_i}{\sum_{i=1}^D \lambda_i}$. Consequently, one popular choice for the low dimensionality d is the lowest value $d \leq D$ such that $E_d \geq \delta$, where δ is a user-defined threshold [15, 28]. In other words, the first d eigenvalues explain at least 100δ percent of the prior variance. Indeed, using the eigendecomposition $\Sigma = U\Lambda U^T$, the total prior variance is $\sum_{i=1}^D \lambda_i$, according to the covariance matrix of $U^T x \sim \mathcal{N}_D(U^T\mu, \text{diag}(\lambda_1, \ldots, \lambda_D))$; consequently, the variance explained by PCA is $\sum_{i=1}^d \lambda_i$, according to the covariance matrix of $U_{PCA}^T x \sim \mathcal{N}_d(U_{PCA}^T\mu, \text{diag}(\lambda_1, \ldots, \lambda_d))$. The work [15] selects a threshold $\delta = 0.98$ for a GP emulation experiment, where the high-dimensionality $D = 31^2$ was reduced to $d = 12$ in a petroleum engineering application. While we focus on Gaussian prior distributions $x \sim \mathcal{N}_D(\mu, \Sigma)$ in this work, the same principle applies for any empirical sample $\{x_i\}_{i=1}^N \sim p_D(x)$ from a potentially non-Gaussian distribution $p_D(x)$; indeed, we can apply the same arguments to the eigendecomposition $\frac{1}{N}ZZ^T = \hat{U}\text{diag}(\hat{\lambda}_1, \ldots, \hat{\lambda}_D)\hat{U}^T$ of the empirical covariance matrix.

The main advantages of PCA can be summarized by the following results, which are discussed in great detail in [18]. Firstly, PCA extracts the d directions in which the projected data $U_{PCA}^T x$ has maximum variance; it also produces uncorrelated new variables $z = U_{PCA}^T x \in \mathbb{R}^d$. Secondly, PCA is useful not only in terms of dimension reduction, but also in terms of reconstructing x from $U_{PCA}^T x$. Indeed, in the empirical case where only samples $\{x_i\}_{i=1}^N \sim p_D(x)$ are available, U_{PCA} provides the optimal reconstruction error $\tilde{x}_i := U_{PCA}U_{PCA}^T(x_i - \tilde{\mu}) + \tilde{\mu} \approx x_i$ ($\tilde{\mu} := \frac{1}{N}\sum_{i=1}^N x_i$) in terms of minimum squared error $\sum_{i=1}^N ||\tilde{x}_i - x_i||^2$ over $d-$dimensional linear maps, according to the Eckart–Young-Mirsky theorem. Finally, in case of a Gaussian distribution $x \sim \mathcal{N}_D(\mu, \Sigma)$, U_{PCA} maximizes the mutual information between x and $z = U_{PCA}^T x$ over the space of $d-$dimensional orthogonal projections; the mutual information is the same as the differential entropy for this Gaussian case, and it can be written as $H(y) = \frac{1}{2}\log_2(e(2\pi)^d) + \frac{1}{2}\log_2 \det(\mathbb{C}\mathrm{ov}[y])$, where $y := U_{PCA}^T(x - \mu)$.

PCA will also be useful when we encounter prior distributions $p(x)$ in the form of Gaussian Random Fields (GRFs), whose samples $x \sim p(x)$ will be a function of location in two-dimensions, e.g. $x(s) : [0,1]^2 \to \mathbb{R}$. We can express $x(s)$ in terms of standard independent Gaussian random variables $z_i \sim \mathcal{N}(0,1)$ via

$$x(s) = \mu + \sum_{i=1}^{\infty} \sqrt{\lambda_i} u_i(s) z_i, \tag{9}$$

where μ, $\{\lambda_i\}_{i=1}^{\infty}$, and $\{u_i(s)\}_{i=1}^{\infty}$ are the mean, eigenvalues, and eigenfunctions, respectively, of the GRF. The eigenvalues and eigenfunctions are defined according to the covariance function of the GRF:

$$\int \mathbb{C}\mathrm{ov}[x(s), x(t)] u_i(s) ds = \lambda_i u_i(s).$$

According to the Mercer's theorem, we can write the covariance function as the infinite sum $\mathbb{C}\mathrm{ov}[x(s), x(t)] = \sum_{i=1}^{\infty} \lambda_i u_i(s) u_i(t)$.

The representation (9) is known as the Karhunen-Loève decomposition; in our computer experiments, we will use a finite-dimensional (truncated) version of the Karhunen-Loève decomposition for GRFs, together with a $D-$dimensional discretization of $[0,1]^2$ ($x \in \mathbb{R}^D$):

$$x = \mu + \sum_{i=1}^{Q} \sqrt{\lambda_i} u_i z_i, \tag{10}$$

which corresponds to a Gaussian prior distribution $p(x) \sim \mathcal{N}_D(\mu, \Sigma)$. The prior covariance Σ can be decomposed as $U\Lambda U^T$, where $U := [u_1, \ldots, u_Q] \in \mathbb{R}^{D \times Q}$ is the matrix of eigenvectors and $\Lambda := \mathrm{diag}(\lambda_1, \ldots, \lambda_Q) \in \mathbb{R}^{Q \times Q}$ is the diagonal matrix of corresponding eigenvalues $\lambda_1 \geq \cdots \geq \lambda_Q \geq 0$.

Using the decomposition (10), we can simply extract $U_{PCA} = [u_1, ..., u_d] \in \mathbb{R}^{D \times d}$ and write the PCA reconstruction $\tilde{x} := U_{PCA} U_{PCA}^T (x - \mu) + \mu \approx x$ corresponding to the dimension reduction $U_{PCA}^T x$ as:

$$\tilde{x} = \mu + \sum_{i=1}^{d} \sqrt{\lambda_i} u_i z_i. \tag{11}$$

The resulting covariance matrix of $\tilde{x} \sim \mathcal{N}_D(\mu, \Sigma_d)$ is $\Sigma_d := U_{PCA} \Lambda_{PCA} U_{PCA}^T$, with $\Lambda_{PCA} := \text{diag}(\lambda_1, \ldots, \lambda_d) \in \mathbb{R}^{d \times d}$. According to the Eckart–Young-Mirsky theorem, Σ_d is the best rank d approximation of the full covariance matrix Σ in the Frobenius distance $||\Sigma - \Sigma_d||_F := \sqrt{\sum_{i=1}^{D} \sum_{j=1}^{D} |(\Sigma)_{ij} - (\Sigma_d)_{ij}|^2}$, or the 2-norm distance, which can be written in terms of the largest eigenvalue as

$$||\Sigma - \Sigma_d||_2 := \sqrt{\lambda_{max}((\Sigma - \Sigma_d)^T (\Sigma - \Sigma_d))}.$$

As mentioned in Section 1.5, our goal is to use PCA for GP emulation, in order to break the curse of dimensionality that might occur when using the high-dimensional emulator with no dimension reduction (3), i.e. $[g_{\text{GP}}(x)]_j : \mathbb{R}^D \to \mathbb{R}$ for every output $[f(x)]_j$ of the simulator $f(x) := ([f(x)]_1, \ldots, [f(x)]_m)^T$. One option is to use $U_{PCA} \in \mathbb{R}^{D \times d}$ to build the $d-$dimensional Gaussian Process emulator

$$[g_{GP}(U_{PCA}^T x)]_j \sim \text{GP}(m_j(U_{PCA}^T x), k_j(U_{PCA}^T x, U_{PCA}^T x')). \tag{12}$$

Another option is provided by the decomposition (11), where we can use $z := (z_1, \ldots, z_d)^T \in \mathbb{R}^d$ as inputs for our GP:

$$[g_{GP}(z)]_j \sim \text{GP}(m_j(z), k_j(z, z')); \tag{13}$$

we can write the training sets as $\{U_{PCA}^T x_i, [f(x_i)]_j\}_{i=1}^N$ in the former case (12), and $\{z_i, [f(\tilde{x}_i)]_j\}_{i=1}^N$ in the latter case (13), where $\tilde{x}_i := \mu + \sum_{k=1}^d \sqrt{\lambda_k} u_k (z_i)_k$ are the reconstructions from (11). One advantage of the former construction is that given a collection of test inputs $\{x_i^\star\}_{i=1}^{N_{test}}$, we can return our fast probabilistic approximations for the expensive simulator via $[g_{\text{GP}}(U_{PCA}^T x_i^\star)]_j \approx [f(x_i^\star)]_j$. However, one advantage of the latter construction is that due to the lower-dimensionality $z_i \in \mathbb{R}^d$ compared with $x_i \in \mathbb{R}^D$, it might be easier to construct the training set, for example via a maximin latin hypercube design [163]. Indeed, we are mostly interested in using GP emulators $[g_{\mathrm{GP}}(z)]_j$ for Bayesian inverse problems, where we can easily use $\tilde{x} := \mu + \sum_{k=1}^{d} \sqrt{\lambda_k} u_k (z)_k$ to transform posterior samples $z := [(z)_1, \ldots, (z)_d]^T \sim p(z|\mathcal{D})$ into high-dimensional posterior samples $p(\tilde{x}|\mathcal{D})$. Note, however, that this approach can lead to problems in various inversion applications, as outlined for PCA applied to the output space $f(x) \in \mathbb{R}^m$ in [138].

We expect that the performance of the GP emulator will depend on the rate of decay of the eigenvalues of $\mathbb{C}\mathrm{ov}[x]$. Indeed, given that $[f(x)]_j$ is Lipschitz continuous (i.e. there exists $L \geq 0$ such that $|[f(x)]_j - [f(y)]_j| \leq L||x-y||_2$ for all $x, y \in \mathbb{R}^D$), [184] proves that if the eigenvalues of $\mathbb{C}\mathrm{ov}[x]$ decay sufficiently fast (see Equation 3.3 in [184]), then there exists a function $g : \mathbb{R}^d \to \mathbb{R}$ such that $[f(x)]_j \approx g(U_{PCA}^T x)$ with arbitrary accuracy ϵ, for $d = d(\epsilon) \ll D$ as a monotone decreasing function of ϵ.

2.3 ACTIVE SUBSPACES

As discussed in Section 1.5, Active Subspaces (AS) are a popular dimension reduction method for GP emulation. We will first introduce the standard gradient-based approach for constructing an active subspace, followed by a discussion about various gradient-free attempts which do not need access to the gradient $\nabla f(x)$ of the simulator function.

For every simulator output $[f(x)]_j$, we consider the expected outer product of the gradient $\nabla[f(x)]_j$ with respect to the prior distribution $p(x)$:

$$W = \int \nabla[f(x)]_j \nabla[f(x)]_j^T p(x) dx. \tag{14}$$

To outline some limitations of using W, for Bayesian inverse problems we can modify (14) by replacing the prior distribution $p(x)$ with an approximation $\tilde{p}(x|\mathcal{D})$ of the posterior distribution $p(x|\mathcal{D})$, in a methodology known as likelihood informed subspaces [37]. Although not considered in our work, this approach can lead to better results in various Bayesian inverse problems applications [185]. Furthermore, we could also treat all the outputs $f(x) = ([f(x)]_1, \dots, [f(x)]_m)^T$ jointly by using the recently developed multi-output active subspace methodology [184].

The AS method constructs the dimension reduction matrix $W_{AS} := [w_1, \dots, w_d] \in \mathbb{R}^{D\times d}$ by extracting the (orthonormal) eigenvectors $w_1, \dots, w_d$ corresponding to the largest d eigenvalues $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$ of the symmetric positive semidefinite matrix W (14). This approach is motivated by the existence of a low-dimensional approximation $[f(x)]_j \approx g(W_{AS}^T x)$, which satisfies the following error bound (Theorem 3.1, [30]):

$$\Big(\int \big([f(x)]_j - g(W_{AS}^T x)\big)^2 p(x)dx \Big)^{1/2} \leq C(\lambda_{d+1} + \cdots + \lambda_D)^{1/2}, \qquad (15)$$

where $\lambda_{d+1} \geq \cdots \geq \lambda_D$ are the trailing eigenvalues of W ($\lambda_{d+1} \leq \lambda_d$), and C is a constant that depends on $p(x)$; in particular, $C = 1$ if $p(x) \sim \mathcal{N}_D(0, I)$, as it can be seen from the proof of Theorem 3.1 in [30] and Corollary 3.2 in [23]. Note that if $\lambda_{d+1} = \cdots = \lambda_D = 0$, then the approximation becomes exact, as $[f(x)]_j = g(W_{AS}^T x)$ (almost surely with respect to $p(x)$).

In practice, the integral (14) is usually analytically intractable, and thus we need to replace it by the Monte Carlo approximation

$$\hat{W} = \frac{1}{M} \sum_{m=1}^{M} \nabla[f(x_m)]_j \nabla[f(x_m)]_j^T \qquad (16)$$

for independent samples $x_m \sim p(x)$ (see [28] for a discussion about the practical choice $M = \alpha d \log(D)$ for $2 \leq \alpha \leq 10$ (Equation 4.1 [28]), which is motivated by the $\log(D)$ term in the upper bound for $||W - \hat{W}||_2$ (Corollary 3.6 [28])). In this case, the AS matrix becomes $\hat{W}_{AS} := [\hat{w}_1, \dots, \hat{w}_d]$, again obtained by extracting the d eigenvectors corresponding to the largest eigenvalues $\hat{\lambda}_1 \geq \cdots \geq \hat{\lambda}_d \geq 0$ of $\hat{W}$. Theorem 3.6 in [30] shows that we can find a different low-dimensional function $\hat{g} : \mathbb{R}^d \to \mathbb{R}$ with the following error bound on the approximation $[f(x)]_j \approx \hat{g}(\hat{W}_{AS}^T x)$:

$$\int ([f(x)]_j - \hat{g}(\hat{W}_{AS}^T x))^2 p(x)dx \leq C\Big(1 + \frac{1}{M}\Big)\Big(\epsilon\big(\sum_{i=1}^{d} \lambda_i\big)^{1/2} + \big(\sum_{i=d+1}^{D} \lambda_i\big)^{1/2}\Big)^2, \qquad (17)$$

where $\epsilon := ||W_{AS}W_{AS}^T - \hat{W}_{AS}\hat{W}_{AS}^T||_2$ is the estimated subspace error in the matrix 2-norm distance. The estimated subspace error ϵ is an important factor for choosing the low-dimensionality d in active subspace applications; indeed, Corollary 3.7 in [28] shows that for a standard choice of M (e.g. $M \propto \log(D)$, as discussed above),

$$\epsilon \leq \frac{4\lambda_1 \min(1, (\lambda_d - \lambda_{d+1})/(5\lambda_1))}{\lambda_d - \lambda_{d+1}} \tag{18}$$

with high probability, which indicates that ϵ is small when there is a large gap $\lambda_d - \lambda_{d+1}$ between the d^{th} largest eigenvalue of W and the next smaller one. As a result, we choose d to be the smallest positive integer $d \leq D$ for which there is a large gap $\hat{\lambda}_d - \hat{\lambda}_{d+1}$ between the corresponding eigenvalues of $\hat{W}$ (e.g. $\hat{\lambda}_d$ is at least one order of magnitude larger than $\hat{\lambda}_{d+1}$), under the constraint that the trailing eigenvalues $(\sum_{i=d+1}^{D} \hat{\lambda}_i)^{1/2}$ are small, so that a low-dimensional approximation $\hat{g}(\hat{W}_{AS}^T x) : \mathbb{R}^d \to \mathbb{R}$ of $[f(x)]_j$ exists according to (17). See Corollary 3.3 in [28] for theoretical guarantees on the quality of the approximation $\{\hat{\lambda}_i\}_{i=1}^D$ for $\{\lambda_i\}_{i=1}^D$.

The approximation $\hat{g}(\hat{W}_{AS}^T x)$ uses the simulator function $[f(x)]_j$ and thus it will not be considered in this work; instead, we will use GP emulators as low-dimensional approximations, built from training sets of the form $\{\hat{W}_{AS}^T x_i, [f(x_i)]_j\}_{i=1}^N$. However, we note that a much smaller number of simulator evaluations were needed to perform Bayesian inversion with $\hat{g}$ instead of $[f(x)]_j$ in the Markov Chain Monte Carlo (MCMC) procedure from [31]. We will also consider this setting in Chapter 4, but through the perspective of GP emulators.

The first use of $\hat{W}_{AS}^T x \in \mathbb{R}^d$ as dimension reduction for GP emulators

$$[g_{GP}(\hat{W}_{AS}^T x)]_j \sim \text{GP}(m_j(\hat{W}_{AS}^T x), k_j(\hat{W}_{AS}^T x, \hat{W}_{AS}^T x')) \tag{19}$$

with training sets of the form $\{\hat{W}_{AS}^T x_i, [f(x_i)]_j\}_{i=1}^N$ appears in the seminal work [30], where it was shown that this procedure can provide a better approximation for $[f(x)]_j$ compared to the standard GP baseline (3) with high-dimensional training inputs $\{x_i\}_{i=1}^N$ and same training outputs $\{[f(x_i)]_j\}_{i=1}^N$.

The computer experiment that involves an Elliptic Partial Differential Equation (PDE) simulator was also considered in [103], and will be revisited later in this chapter. Given that gradients of the simulator $\nabla[f(x)]_j$ might be unavailable in practice, the work [166] uses the GP training set $\{x_i, [f(x_i)]_j\}_{i=1}^N$ and replaces $\hat{W}_{AS}$ with the embedding $\hat{W}_{\mathrm{GP}} \in \mathbb{R}^{D\times d}$ that approximately maximizes the marginal likelihood (5) for

$$[g_{GP}(W_{GP}^T x)]_j \sim \mathrm{GP}(m_j(W_{GP}^T x), k_j(W_{GP}^T x, W_{GP}^T x')) \tag{20}$$

with respect to W_{GP}, i.e. by treating the dimension reduction matrix W_{GP} as an additional GP hyperparameter.

This approach has a long history in the Gaussian Process literature, which precedes the introduction of AS; the theoretical connection with AS was recently established by [181], and will be discussed shortly. The first attempt to treat the dimension reduction matrix W_{GP} as an additional GP hyperparameter is due to [169], where the embedding $\hat{W}_{\mathrm{GP}}$ is also learned together with the rest of GP hyperparameters by maximizing the marginal likelihood (5) for the GP (20). This approach has been extended to sparse GPs [154], sparse GPs with latent (probabilistic) hyperameters [164], and learning the embedding via active learning [62]. In the latter method, the training set $\{x_i, [f(x_i)]_j\}_{i=1}^N$ is constructed iteratively, such that every x_n is selected according to some criterion for improving the accuracy of the current estimate $\hat{W}_{\mathrm{GP}}^{(n)}$. Sparse GPs are methods designed for training sets with a large number of observations and are not considered in this thesis, since we focus on applications of small or medium size. Note that all GPs of the form (20) that treat the dimension reduction matrix W_{GP} as an additional GP hyperparameter use $\{x_i, [f(x_i)]_j\}_{i=1}^N$ as training sets.

The novelty in [166] compared with the traditional approach from [169] comes from the additional orthonormality constraint $\hat{W}_{\mathrm{GP}}^T \hat{W}_{\mathrm{GP}} = I_d$. In other words, the maximization of the marginal likelihood (5) for the GP (20) with respect to W_{GP} is performed over the space of semi-orthogonal matrices $\{B \in \mathbb{R}^{D\times d} : B^T B = I_d\}$ known as the Stiefel manifold; this mimics the orthonormal nature of the eigenvectors that form the (gradient-based) active subspace matrix $\hat{W}_{AS}$ ($\hat{W}_{AS}^T \hat{W}_{AS} = I_d$). The resulting embedding $\hat{W}_{\text{GP}}$ proved to be a good match for the gradient-based $\hat{W}_{AS}$ in a series of computer experiments, while the predictive uncertainty of the resulting GP was further improved by a latent variable treatment of the semi-orthogonal embedding W_{GP}, coupled with an orthogonal-MCMC method that samples from the posterior distribution $p(W_{\text{GP}} | \{x_i, [f(x_i)]_j\}_{i=1}^N)$ over the Stiefel manifold [63]. Alternatively, [134] suggests optimizing W_{GP} over the set of $d-$dimensional subspaces of $\mathbb{R}^D$ (also known as the Grassman manifold); in this way, we preserve the orthonormality constraint, while preventing the search over different orthonormal bases for the same subspace.

However, none of these works pointed out a theoretical connection between the gradient-based active subspace $\hat{W}_{AS}$ and the orthonormal embeddings $\hat{W}_{\text{GP}}$ that approximately maximize the marginal likelihood. In this regard, [181] shown that under the assumption that the simulator $[f(x)]_j$ is a sample from the GP prior (20), the Monte Carlo estimator $\hat{W}$ from (16) is a method-of-moments estimator for $\frac{\sigma^2}{l^2} W_{\text{GP}} W_{\text{GP}}^T$, regardless of whether W_{GP} is an orthonormal embedding [166, 134] or a traditional unstructured embedding [169]; recall from (4) that σ^2 is the signal variance and l is the lengthscale hyperparameter of the squared-exponential GP covariance function $k_j(W_{GP}^T x, W_{GP}^T x')$. Indeed, Corollary 1 in [181] shows that

$$\mathbb{E}_{\text{GP}}\Big[\frac{1}{M}\sum_{m=1}^{M} \nabla[g_{\text{GP}}(W_{\text{GP}}^T x_m)]_j \nabla[g_{\text{GP}}(W_{\text{GP}}^T x_m)]_j^T\Big] = \frac{\sigma^2}{l^2} W_{\text{GP}} W_{\text{GP}}^T$$

for any sample $\{x_m\}_{m=1}^M$. Note that $\hat{W} = \frac{1}{M}\sum_{m=1}^M \nabla[f(x_m)]_j \nabla[f(x_m)]_j^T$ for $[f(x)]_j \sim \text{GP}$ is simply a one-sample estimator for this expectation. In conclusion, by maximizing the marginal likelihood, we might be able to recover a good approximation to the gradient-based active subspace estimator $\hat{W}_{AS}$ (i.e. the dominant d eigenvectors of $\hat{W}$) regardless of the constraints used for W_{GP}, such as orthonormality or unstructured (no constraints).

We mention two additional orthonormal parmetrizations for W_{GP} in (20). Firstly, sufficient dimension reduction (SDR) methods [103], which have been originally proposed for recovering $A \in \mathbb{R}^{D \times d}$ in the case where there exists this exact low-dimensional linear structure $f(x) = g(A^T x)$ [72]. SDR methods will be discussed in detail in Chapter 3. Secondly, the Gaussian ridge functions methodology [144], where $\hat{W}_{\text{GP}}$ minimizes a GP prediction error over the Stiefel manifold on a separate test set $\{x_n^\star\}_{n=1}^{N_{test}}$, rather than maximizing the marginal likelihood on the training set. This method showed improved predictive performance compared with some alternative SDR methods on a real-world turbomachinery experiment [144]. While this approach is very interesting, in some applications we might not have access to enough budget to split the data $\{x_n, [f(x_n)]_j\}_{n=1}^{N_{budget}}$ in order to learn both the embedding $\hat{W}_{\text{GP}}$ (test data) and the rest of the GP hyperparameters (training data).

Finally, we mention [181], which shows that the active subspace can be estimated directly from a high-dimensional GP (3). While this result is important from both theoretical and practical point of view, it has been shown that this method can fall short in practice when the high-dimensional GP cannot accurately approximate the simulator function $[f(x)]_j$, and it was outperformed by various GP approaches with dimension reduction (20) and orthonormal paramterizations [63, 134].

2.4 elliptic partial differential equation (pde) case study

Following the computer experiments from [30, 103], we investigated the comparison between the predictive performance of GP emulators with PCA dimension reduction (12), active subspace (AS) dimension reduction (19), and the high-dimensional GP baseline with no dimension reduction (4), respectively. The computer experiments involve GP emulation of an elliptic Partial Differential Equation (PDE) solver as the simulator $f(x)$, which is a standard problem in geostatistical applications such as groundwater modelling [36] and petroleum engineering [78].

In this work, we treat the elliptic PDE simulator $f(x) : \mathbb{R}^D \to \mathbb{R}$ and its gradient $\nabla f(x) : \mathbb{R}^D \to \mathbb{R}^D$ from [30, 103] as black-box functions, since we will only focus on the prior distribution $p(x)$, simulator evaluations $\{x_i, f(x_i)\}_{i=1}^N$, and gradient evaluations $\{x_i, \nabla f(x_i)\}_{i=1}^M$ for $x_i \sim p(x)$. The evaluations $f(x)$ and $\nabla f(x)$ are performed using the implementation from [27]. For the precise mathematical formulation of $f(x)$ and $\nabla f(x)$, including details about the computational implementation, see Section 5.1 from [30]. Nonetheless, we mention that $x \in \mathbb{R}^D$ is the input of the simulator. From x, we generate the PDE (log-)coefficients $\log a_x$ via (21). The PDE solution corresponding to these coefficients is the function $v_x(s) : [0,1]^2 \to \mathbb{R}$ that satisfies

$$-\nabla_s \cdot (a_x \nabla_s v_x) = 1, \quad s \in [0,1]^2,$$

together with homogeneous Dirichlet boundary conditions (i.e, $v_x(s) \equiv 0$) on the left, top, and bottom of the spatial domain $[0,1]^2$; the right side of the spatial domain has a homogeneous Neumann boundary condition (i.e, $\nabla_s v_x(s) \equiv 0$). The resulting simulator output $f(x) \in \mathbb{R}$ is an approximation of the average PDE solution $\frac{1}{|\Gamma_2|} \int_{\Gamma_2} v_x(s) ds$ over the right boundary Γ_2 of the domain $[0,1]^2$, obtained using a numerical PDE solution $\hat{v}_x(s) : [0,1]^2 \to \mathbb{R}$ with a standard linear finite element method.

The PDE coefficients $a_x(s) : [0,1]^2 \to \mathbb{R}$ usually represent a physical quantity (e.g., permeability in a 2D field encoded as $[0,1]^2$), which is linked to the PDE solution $v_x(s) : [0,1]^2 \to \mathbb{R}$ (that represents another physical quantity, e.g., water volume is the 2D field); the PDE models the underlying physical law. To generate (physically) plausible PDE coefficients $a_x(s)$, we use the parametrization x (which also serves as input for our computer simulator). The prior distribution $p(x)$ is a standard multivariate Gaussian $x = [x_1, \ldots, x_D]^T \sim \mathcal{N}_D(0, I)$ with $D = 100$. The PDE coefficients a_x on a discretization $\mathcal{B}$ of $[0,1]^2$ of size $|\mathcal{B}|$ ($a_x \in \mathbb{R}^{|\mathcal{B}|}$) are obtained from the standard Gaussian random variables $x_i \sim \mathcal{N}(0,1)$ through a finite-dimensional Karhunen-Loève decomposition, which was introduced in Section 2.2:

$$\log a_x = \sum_{i=1}^{D} \sqrt{\lambda_i} u_i x_i. \tag{21}$$

We will shortly describe the eigenpairs $\{\lambda_i, u_i\}_{i=1}^{D}$. In this section, we fix $|\mathcal{B}| = D \times D = 10^4$, according to the implementation [27]. Since the simulator $f(x)$ depends on the eigenpairs $\{\lambda_i, u_i\}_{i=1}^{D}$, we explicitly include them as $f_{\lambda,u}(x)$. The eigenvectors $u_i \in \mathbb{R}^{|\mathcal{B}|}$ are the evaluations $u_i(b) : \mathcal{B} \to \mathbb{R}$ of the eigenfunctions $u_i(s) : [0,1]^2 \to \mathbb{R}$ for the infinite-dimensional Karhunen-Loève decomposition for the Gaussian random field $\log a_x(s) : [0,1]^2 \to \mathbb{R}$:

$$\log a_x(s) = \sum_{i=1}^{\infty} \sqrt{\lambda_i} u_i(s) x_i, \tag{22}$$

which is constructed through the extension $\{\lambda_i, u_i(s), x_i\}_{i=D+1}^{\infty}$. Following [30, 103], the eigenpairs $\{\lambda_i, u_i(s)\}_{i=1}^{\infty}$ are chosen so that

$$\mathbb{C}\text{ov}[\log a_x(s), \log a_x(t)] = \exp(-l^{-1}||s-t||_1); \tag{23}$$

this is known as an exponential covariance function with lengthscale l (the 1-norm distance for $s,t \in \mathbb{R}^2$ is defined as $||s-t||_1 := |s_1 - t_1| + |s_2 - t_2|$). One analytic construction for $\{\lambda_i, u_i(t)\}_{i=1}^{\infty}$ corresponding to this covariance function can be found in Example 1 from [19]. The intuition for the effect of the lengthscale l has been discussed in Section 2.1 for squared exponential covariance functions (4) of Gaussian Process priors for GP emulators; the same intuition applies here. However, samples $g(s) : [0,1]^2 \to \mathbb{R}$ corresponding to the covariance function (23) are no longer infinitely differentiable (in fact, they are not differentiable). Note that both a short lengthscale experiment ($l = 0.01$) and a long lengthscale experiment ($l = 1$) are considered in [30, 103]. We write $(\{\lambda_i^S, u_i^S\}_{i=1}^{D}, f_{\lambda^S,u^S}(x))$ and $(\{\lambda_i^L, u_i^L\}_{i=1}^{D}, f_{\lambda^L,u^L}(x))$ for the eigenpairs and the corresponding simulator used in the short lengthscale case and the long lengthscale case, respectively.

Recall that one goal was to perform dimension reduction from D to $d \leq D$ via AS and PCA; these methods were discussed in Section 2.3

and 2.2, respectively. Regarding active subspaces, we need to extract the d eigenvectors corresponding to the largest d eigenvalues of the AS matrix $\hat{W} = \frac{1}{M}\sum_{m=1}^{M} \nabla f_{\lambda,u}(x_m)\nabla f_{\lambda,u}(x_m)^T$ from independent prior samples $x_m \sim \mathcal{N}_D(0, I)$. For PCA, we need to extract the d eigenvectors $\{u_i\}_{i=1}^{d}$ corresponding to the largest eigenvalues $\{\lambda_i\}_{i=1}^{d}$. For the same set of inputs $\{x_i\}_{i=1}^{N}$, the GP training sets will be of the form $\{\hat{W}_{AS}^T x_i, f(x_i)_{\lambda,u}\}_{i=1}^{N}$ for the AS method with GP prior (19), and $\{U_{PCA}^T \log a_{x_i}, f(x_i)_{\lambda,u}\}_{i=1}^{N}$ for the PCA method with GP prior (12). Note that both training sets are generated from the same collection of independent prior samples $\{x_i\}_{i=1}^{N} \sim p(x)$ and corresponding simulator outputs $\{f(x_i)_{\lambda,u}\}_{i=1}^{N}$. $\mathcal{D}_N := \{x_i, f(x_i)_{\lambda,u}\}_{i=1}^{N}$ will be the training set for the high-dimensional GP baseline with no dimension reduction (4); we will use the notation NO-DR for this method. We consider a collection of $M = 300$ Monte Carlo samples to estimate the active subspace, $N = 300$ GP training points, and a further $N_\star = 300$ test points as independent prior samples $\{x_i^\star\}_{i=1}^{N_\star}$, which are used to assess the GP performance for approximating $f_{\lambda,u}(x)$ using the three methods AS, PCA, and NO-DR. The criterion for method comparison is the normalized predictive root mean squared error (N-RMSE), which we define using the different GP posteriors as

$$\sqrt{\frac{1}{N_\star}\sum_{i=1}^{N_\star}(\bar{m}(\hat{W}_{AS}^T x_i^\star) - f(x_i^\star)_{\lambda,u})^2/F} \quad \text{(AS)}, \tag{24}$$

$$\sqrt{\frac{1}{N_\star}\sum_{i=1}^{N_\star}(\bar{m}(U_{PCA}^T \log a_{x_i^\star}) - f(x_i^\star)_{\lambda,u})^2/F} \quad \text{(PCA)}, \tag{25}$$

$$\sqrt{\frac{1}{N_\star}\sum_{i=1}^{N_\star}(\bar{m}(x_i^\star) - f(x_i^\star)_{\lambda,u})^2/F} \quad \text{(NO-DR)}, \tag{26}$$

where $F := \max_{i \le N_\star} f(x_i^\star)_{\lambda,u} - \min_{i \le N_\star} f(x_i^\star)_{\lambda,u}$ is the normalizing factor. The experimental setup is in line with [103], except that we use a linear GP prior mean $m(x) = Ax + b$ only in the NO-DR case; for simplicity, we use a constant prior mean $m(U_{PCA}^T \log a_x) = C_1$ and $m(\hat{W}_{AS}^T x) = C_2$ for

PCA and AS, respectively. As discussed in Section 2.1, we optimize the GP hyperparameters by maximizing the marginal likelihood (5) using the LBFGS algorithm (NO-DR) or the gradient descent algorithm (AS, PCA).

The conclusions from [30, 103] can be summarized as follows; note that both AS and PCA use $d \leq 5$. The (gradient-based) AS method outperforms PCA and NO-DR on both cases considered, i.e. short lengthscale $f_{\lambda^S, u^S}(x)$ and long lengthscale $f_{\lambda^L, u^L}(x)$. PCA outperforms NO-DR for $f_{\lambda^L, u^L}(x)$, whereas NO-DR significantly outperforms PCA for $f_{\lambda^S, u^S}(x)$. The poor performance of PCA on $f_{\lambda^S, u^S}(x)$ is indicated by the very slow decay of the eigenvalues ($\lambda_1^S \approx \lambda_2^S \approx \cdots \approx \lambda_5^S$, Table 5.1, [30]), as the first five eigenvalues $\{\lambda_i^S\}_{i=1}^5$ corresponding to (21) explain only $\approx 6\%$ of the prior variance; we need to select $d = 90 \approx 100 = D$ to account for $\approx 92\%$ of the prior variance. The situation is very different for $f_{\lambda^L, u^L}(x)$, as the first eigenvalue λ_1^L is one order of mangnitude larger than λ_2^L (fast decay), and the five eigenvalues $\{\lambda_i^L\}_{i=1}^5$ (Table 5.1, [30]) explain $\approx 92\%$ of the prior variance. Note that the eigenvalues $\{\hat{\lambda}_i\}_{i=1}^D$ for the active subspace method $\hat{W}$ decay very fast for both $f_{\lambda^S, u^S}(x)$ and $f_{\lambda^L, u^L}(x)$, as there is a large gap $\hat{\lambda}_1 - \hat{\lambda}_2$ for $\hat{W}$ in both cases ($\hat{\lambda}_1$ is at least two orders of magnitude larger than $\hat{\lambda}_2$), with the trailing eigenvalues $\{\hat{\lambda}_i\}_{i=2}^D$ being small (Table 5.1, [30]). Therefore, the choice $d = 1$ for AS is in line with the criterion discussed in Section 2.3, whereas the choice $d \leq 5$ for PCA in the long lengthscale experiment $f_{\lambda^L, u^L}(x)$ is in line with the criterion from Section 2.2. The superior performance of AS compared to PCA as a dimension reduction method has also been observed outside the GP emulation literature; for computer experiments involving elliptic PDE simulators, see [184] for low-dimensional approximations for multi-output simulators, and [185] for a Bayesian inverse problem application. Proposition 3.1 in [184] includes an approximation theoretical result, which shows that the upper bound for the approximation error (15) achieved by AS in case of a Gaussian prior distribution $p(x) \sim \mathcal{N}_D(\mu, \Sigma)$ is always smaller than the upper bound achieved by an analogue PCA approximation.

In this work, we investigate the effect of the magnitude of the eigenvalues $\{\lambda_i\}_{i=1}^{D}$ in (21) with respect to the GP emulation performance for PCA, AS, and NO-DR. As discussed in Section 2.2, the rate of decay of these eigenvalues or the percentage of variance explained by the first d eigenvalues $\{\lambda_i\}_{i=1}^{d}$ has been considered previously in the literature [15, 184]. Here, we take a different approach; we keep the rate of decay fixed, while multiplying the eigenvalues by a constant factor $\{K\lambda_i\}_{i=1}^{D}$.

We motivate our approach through the following experiment. In Table 1, we present the long lengthscale setting from [30, 103]. In the first row, we use the original eigenpairs $\{\lambda_i^L, u_i^L\}_{i=1}^{D}$; the results are in line with [103], as expected. However, if we change the eigenpairs to $\{1/10 \cdot \lambda_i^L, u_i^L\}_{i=1}^{D}$, i.e. multiply each eigenvalue by a factor of $1/10$, while keeping the rate of decay and the eigenvectors fixed, NO-DR outperforms PCA (second row, Table 1). Note that AS is the best performing method at both magnitudes.

Table 1: GP emulation performance via N-RMSE criterion for AS (24), PCA (25), and NO-DR (26). Both PCA and AS use low-dimensionality $d = 1$. Best performing method is shown in bold; second best performing method is shown in italic.

Eigenpairs (21)	AS	PCA	NO-DR
$\{\lambda_i^L, u_i^L\}_{i=1}^{D}$	**0.01**	*0.05*	0.08
$\{0.1 \cdot \lambda_i^L, u_i^L\}_{i=1}^{D}$	**0.01**	0.06	*0.04*

We validate our observation using an additional set of eigenvalues $\lambda_i^F := 100 \cdot i^{-3}$ for $i \in \{1, \ldots, 100\}$, which has a faster rate of decay compared with the original long lengthscale eigenvalues λ^L (see Figure 1). While we expect that this will benefit PCA, the benefits for NO-DR are more difficult to predict. We couple λ^F with both sets of eigenvectors u^S (from the short lengthscale case) and u^L (from the long lengthscale case), and present the predictive performance in Table 2 and 3, respectively. The conclusions

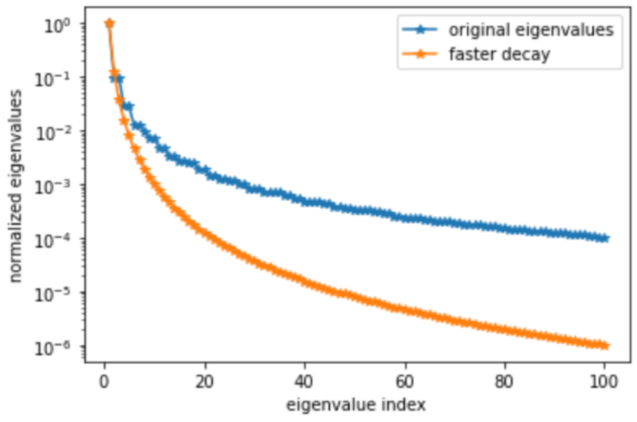


Figure 1: Original eigenvalues from the long lengthscale case (λ^L, blue stars) versus our proposed faster decay eigenvalues (λ^F, orange stars).

are in line with the experiment in Table 1; NO-DR outperforms PCA at smaller magnitudes λ^F, whereas PCA outperforms NO-DR at the larger magnitudes $100 \cdot \lambda^F$. AS is the best performing method in all cases, although it is matched by NO-DR in the smaller magnitudes experiments.

Table 2: GP emulation performance via N-RMSE criterion for AS (24), PCA (25), and NO-DR (26). Both PCA and AS use low-dimensionality $d = 1$. Best performing method is shown in bold; second best performing method is shown in italic.

Eigenpairs (21)	AS	PCA	NO-DR
$\{\lambda_i^F, u_i^S\}_{i=1}^D$	**0.01**	0.03	**0.01**
$\{100 \cdot \lambda_i^F, u_i^S\}_{i=1}^D$	**0.02**	*0.05*	0.07

For completeness, we present the (GP) lengthscales resulting from GP emulation (for AS and NO-DR) in Tabs. 4-6. We note that the GP lengthscale is constant for NO-DR throughout all the experiments; this is because without dimension reduction, only the hyperparameters corresponding to the linear GP prior mean are successfully optimized. Instead, with dimension reduction (i.e., AS), we can also successfully optimize the hyperparameters of the

Table 3: GP emulation performance via N-RMSE criterion for AS (24), PCA (25), and NO-DR (26). Both PCA and AS use low-dimensionality $d = 1$. Best performing method is shown in bold; second best performing method is shown in italic.

Eigenpairs (21)	AS	PCA	NO-DR
$\{\lambda_i^F, u_i^L\}_{i=1}^D$	**0.01**	0.03	**0.01**
$\{100 \cdot \lambda_i^F, u_i^L\}_{i=1}^D$	**0.01**	*0.05*	0.06

covariance function, and thus predict almost perfectly on all the test sets (N-RMSE close to zero for all the experiments). A large magnitude ($\gg 1$) for the GP lengthscale often indicates an unimportant direction; however, the GP lengthscales resulting from AS do not fall into this category. Furthermore, the GP lengthscales themselves are not identifiable (only the ratio between GP lengthscales and outputscales, as seen in Theorem 3 from [188]); in our AS experiments, the GP outputscale can have a particularly low value (as low as 10^{-4}), due to the small variance and scale of the outputs.

Table 4: (GP) lengthscales of GP emulation for AS (24) and NO-DR (26).

Eigenpairs (21)	AS	NO-DR
$\{\lambda_i^L, u_i^L\}_{i=1}^D$	2.808	0.693
$\{0.1 \cdot \lambda_i^L, u_i^L\}_{i=1}^D$	7.186	0.693

To summarize, we have shown that the magnitude of the eigenvalues (21) is crucial for method comparison, as for a fixed rate of decay, the high-dimensional GP baseline NO-DR can outperform PCA at smaller magnitudes, while the opposite is true at larger magnitudes. We have demonstrated this for the original long lengthscale experiment $\{\lambda_i^L, u_i^L\}_{i=1}^D$, as well

Table 5: (GP) lengthscales of GP emulation for AS (24) and NO-DR (26).

Eigenpairs (21)	AS	NO-DR
$\{\lambda_i^F, u_i^S\}_{i=1}^D$	4.667	0.693
$\{100 \cdot \lambda_i^F, u_i^S\}_{i=1}^D$	4.750	0.693

Table 6: (GP) lengthscales of GP emulation for AS (24) and NO-DR (26).

Eigenpairs (21)	AS	NO-DR
$\{\lambda_i^F, u_i^L\}_{i=1}^D$	5.053	0.693
$\{100 \cdot \lambda_i^F, u_i^L\}_{i=1}^D$	2.087	0.693

as for a faster decaying set of eigenvalues $\{\lambda_i^F\}_{i=1}^D$ for both sets of eigenvectors $\{u_i^L\}_{i=1}^D$ and $\{u_i^S\}_{i=1}^D$ from [30, 103]. While some works only point out the percentage of the prior variance explained by the first d coefficients $\{\lambda_i\}_{i=1}^d$ when using PCA for GP emulation [15], our work outlines the additional importance of the magnitude of KL coefficients when considering PCA as an alternative option to the high-dimensional GP baseline. Finally, we associate the difficulties of NO-DR at larger magnitudes $\{K\lambda_i\}_{i=1}^D$ for $K \in \{10, 100\}$ with the significantly larger variance in the simulator outputs $f_{K\lambda,u}(x)$ due to the larger variance in the PDE coefficients $\log a_x$ (21).

3

DIMENSION REDUCTION - METHODS COMPARISON FOR GAUSSIAN PROCESS EMULATION

While various dimension reduction techniques for Gaussian Process (GP) emulation have been proposed in the literature, their strengths and weaknesses are not yet well understood. Although we will provide some motivation and introductory theoretical properties, our focus will be on analysing the performance of these methods on a variety of synthetic and real-world simulators. Among the families of methods considered are principal component analysis (PCA) and sufficient dimension reduction (SDR).

Note that all the simulators considered in this chapter have a one-dimensional output $f(x) : \mathbb{R}^D \to \mathbb{R}$, and thus we do not need to separately consider each simulator output $[f(x)]_j$ of $f(x) : \mathbb{R}^D \to \mathbb{R}^m$ ($j \in \{1, \ldots, m\}$), as we did in Chapter 2.

3.1 SUPERVISED DIMENSION REDUCTION

As discussed in Section 1.5, supervised dimension reduction methods require access to simulator evaluations $\{f(x_i)\}_{i=1}^N$ (gradient-free methods) or gradient evaluations $\{\nabla f(x_i)\}_{i=1}^M$ (gradient-based methods). Section 2.3 discusses both families of methods, and provides a detailed introduction of the gradient-based active subspace method for the GP (19), and the gradient-free approach of maximizing the marginal likelihood for the GP (20) with respect to the dimension reduction matrix $W_{\mathrm{GP}} \in \mathbb{R}^{D \times d}$.

3.1.1 *Sufficient Dimension Reduction (SDR)*

In this section, we will introduce the gradient-free family of methods known as Sufficient Dimension Reduction (SDR), which is another widely used way of performing supervised dimension reduction. Chapter 3 in [98] presents a variety of SDR methods, together with a collection of computer experiments. SDR has been used for GP emulation in applications such as tsunami modelling [103] and wind turbine modelling [192].

SDR methods involve searching for $d-$dimensional linear subspaces $A \in \mathbb{R}^{D\times d}$ that manage to explain all the variance in the outputs $y = f(x)$ that is caused by the inputs x, i.e. y is independent of x given $A^T x$:

$$y \perp\!\!\!\perp x | A^T x. \tag{27}$$

In line with the introduction from [70], if A satisfies (27) we call $S_{DRS} :=$ colspan(A) a dimension reduction subspace. The central subspace is the unique subspace of minimum dimensionality (denoted by $S_{y|x}$) which is contained in all other dimension reduction subspaces S_{DRS}:

$$S_{y|x} \subseteq S_{DRS}. \tag{28}$$

Finding the central subspace is the goal of all the SDR algorithms. As discussed in [65], the central subspace is also known as the effective dimension reduction space (EDR) space [101], or the effective subspace [53]. SDR methods have been extensively studied from different points of view. On one hand, there are a variety of theoretical considerations, such as the conditions that guarantee the existence of the central subspace (Lemma 1 and 2, [32]) to the theoretical equivalence between sufficient dimension reduction and ridge recovery (Theorem 2, [72]). Ridge functions are the class of functions $f : \mathbb{R}^D \to \mathbb{R}$ for which there exists a low-dimensional link function $g : \mathbb{R}^d \to \mathbb{R}$ such that $y = f(x) = g(A^T x)$ with $A \in \mathbb{R}^{D\times d}$; ridge recovery involves recovering the low-dimensional subspace A using $f(x)$. See [109] for a comprehensive theoretical survey of SDR methods, [130] for a monograph on ridge functions, and [29] for an application of active subspaces for ridge functions in the context of fluid dynamics.

On the other hand, SDR methods have been applied in various practical applications outside GP emulation, which are discussed in the textbook [100]; we outline magnetohydrodynamics for a simulator $f(x)$ that was also used in active subspace computer experiments [70], and image classification for hand-written digits [54]. The latter application outlines another feature of SDR methods; while we focus here on the setting where data is generated using a simulator function $\{x_i, y_i = f(x_i)\}_{i=1}^N$ for independent (and usually Gaussian) prior samples $x_i \sim p(x)$, SDR methods are generally applicable for problems of the form (27) with $y = g(f(x), \epsilon)$, i.e. for datasets of the form $\{x_i, y_i = g(f(x_i), \epsilon_i)\}_{i=1}^N$ arising from different prior distributions $p(x)$ (e.g. samples $x_i \sim p(x)$ belong to the space of natural images), regression functions $f(x)$, and noise models $g(\cdot, \epsilon)$ such as the standard additive noise model $y = g(f(x), \epsilon) = f(x) + \epsilon$ for $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.

Sliced Inverse Regression (SIR)

Sliced Inverse Regression (SIR) [101] is one of the oldest and most popular sufficient dimension reduction algorithms. Adopting the introduction from [70], we describe SIR by first looking at the $D \times D$ inverse regression matrix, which is the covariance of the $\mathbb{R}^D-$valued random vector $\mathbb{E}[x|y]$:

$$C_{IR} = \mathbb{C}\text{ov}[\mathbb{E}[x|y]]. \tag{29}$$

Under certain conditions, it is known that colspan$(C_{IR}) \subseteq S_{y|x}$, i.e. the inverse regression subspace is contained in the central subspace. One such condition under which the inclusion holds is when $p(x) \sim \mathcal{N}_D(\mu, \Sigma)$ (see Theorem 3.1 in [101]); this is indeed the case in many applications considered in this thesis.

SIR involves partitioning the output space $\{y_i = f(x_i)\}_{i=1}^N$ as

$$y_{min} = \tilde{y}_0 < \tilde{y}_1 < \cdots < \tilde{y}_{R-1} < \tilde{y}_R = y_{max} \tag{30}$$

for $y_{min} = \min_{1\leq i\leq N} y_i$ and $y_{max} = \max_{1\leq i\leq N} y_i$. We then map each observed output y_i to its corresponding partition $[\tilde{y}_{r-1}, \tilde{y}_r]$ via $h(y_i) = r$. In this way, we arrive at the sliced inverse regression population matrix

$$C_{SIR} = \mathbb{C}\text{ov}[\mathbb{E}[x|h(y)]]. \tag{31}$$

Under the same conditions that are required for colspan$(C_{IR}) \subseteq S_{y|x}$ to hold, we have that colspan$(C_{SIR}) \subseteq S_{h(y)|x}$ also holds. This leads to colspan$(C_{SIR}) \subseteq S_{h(y)|x} \subseteq S_{y|x}$, since for any transformation $g(y) : \mathbb{R} \to \mathbb{R}$, the resulting central subspace is contained in the original one $S_{g(y)|x} \subseteq S_{y|x}$, with equality holding when $g(y)$ is strictly monotonic [33]. In other words, colspan(C_{SIR}) is also contained in the central subspace.

After fixing the partitioning (30), the SIR algorithm proceeds by constructing the sample version $\hat{C}_{SIR}$ of the population matrix (31), using the sliced observations $\{x_i, h(y_i)\}_{i=1}^{N}$. The sample covariance is usually written as

$$\hat{C}_{SIR} = \frac{1}{N}\sum_{r=1}^{R} N_r \hat{\mu}_r \hat{\mu}_r^T,$$

where N_r is the number of predictors $\{x_i^{(r)}\}_{i=1}^{N_r}$ for which the corresponding outputs $y_i^{(r)} := f(x_i^{(r)})$ belong to the same partition $h(y_i^{(r)}) = r$, and $\hat{\mu}_r := \frac{1}{N_r}\sum_{i=1}^{N_r} x_i^{(r)}$ is the sample mean of these predictors. As discussed in [70], the performance of SIR in practice is relatively insensitive to the number of slices R; one suggestion for choosing the partitioning (30) is to ensure that each slice contains approximately the same number of samples, according to the computer experiments from the original source [101].

The dimension reduction matrix $\hat{W}_{SIR} := [\hat{w}_{SIR}^{(1)}, \ldots, \hat{w}_{SIR}^{(d)}] \in \mathbb{R}^{D\times d}$ is constructed by selecting the eigenvectors $\{\hat{w}_{SIR}^{(1)}, \ldots, \hat{w}_{SIR}^{(d)}\}$ corresponding to the largest eigenvalues $\hat{\lambda}_{SIR}^{(1)} \geq \cdots \geq \hat{\lambda}_{SIR}^{(d)} \geq 0$ of the sample covariance matrix $\hat{C}_{SIR}$. To motivate this choice, recall that we search for the central subspace $S_{y|x}$. We can write the population covariance matrix C_{SIR} as $C_{SIR} := W\text{diag}(\lambda_1, \ldots, \lambda_D)W^T$, for decreasing eigenvalues $\lambda_1 \geq \cdots \geq \lambda_D \geq 0$ and the corresponding orthogonal matrix of eigenvectors $W :=$

$[w_{SIR}^{(1)}, \ldots, w_{SIR}^{(D)}] \in \mathbb{R}^{D \times D}$. The eigenvectors of C_{SIR} corresponding to non-zero eigenvalues provide a basis for colspan$(C_{SIR}) \subseteq S_{y|x}$. As a result, if the tail eigenvalues $\lambda_{d+1} \geq \cdots \geq \lambda_D \geq 0$ of C_{SIR} are small, then $W_d := [w_{SIR}^{(1)}, \ldots, w_{SIR}^{(d)}] \in \mathbb{R}^{D \times d}$ approximates a basis for colspan(C_{SIR}). Therefore, SIR selects the d eigenvectors $\hat{W}_{SIR} := [\hat{w}_{SIR}^{(1)}, \ldots, \hat{w}_{SIR}^{(d)}] \in \mathbb{R}^{D \times d}$ corresponding to the largest d eigenvalues of $\hat{C}_{SIR}$, based on the following chain of approximations:

$$\text{colspan}(\hat{W}_{SIR}) \approx \text{colspan}(W_d) \approx \text{colspan}(C_{SIR}) \subseteq S_{y|x}, \tag{32}$$

In support of using the trailing eigenvalues $\hat{\lambda}_{SIR}^{(d+1)} \geq \cdots \geq \hat{\lambda}_{SIR}^{(D)}$ of $\hat{C}_{SIR}$ to detect when the corresponding eigenvalues $\lambda_{d+1} \geq \cdots \geq \lambda_D$ of C_{SIR} are small, Theorem 3.2 in [70] shows that for all $e \in \{1, \ldots, D\}$:

$$\mathbb{E}[(\hat{\lambda}_{SIR}^{(e)} - \lambda_e)^2] \to 0$$

as $N \to \infty$ with a fast rate of convergence $N_{r_{\min}}^{-1}$, where $N_{r_{\min}} := \min_{1 \leq r \leq R} N_r$ denotes the minimum number of samples per slice over all the R slices. The rate of convergence also supports choosing approximately the same number of samples in each slice (e.g. Section 6, [101]), as this maximizes $N_{r_{\min}}$.

In addition to checking that the trailing eigenvalues $\{\hat{\lambda}_{SIR}^{(i)}\}_{i=d+1}^{D}$ are small, we can use the gap $\hat{\lambda}_{SIR}^{(d)} - \hat{\lambda}_{SIR}^{(d+1)}$ between consecutive eigenvalues of $\hat{C}_{SIR}$ to guide the selection of the low-dimensionality d, as Theorem 3.3 in [70] shows that for sufficiently large N,

$$||\hat{W}_{SIR}\hat{W}_{SIR}^T - W_d W_d^T||_2 = \frac{1}{\lambda_d - \lambda_{d+1}} O(N_{r_{\min}}^{-1/2})$$

with high probability. Note that a similar result was obtained in the context of active subspaces (18). In conclusion, choosing d such that the gap $\hat{\lambda}_{SIR}^{(d)} - \hat{\lambda}_{SIR}^{(d+1)}$ is large and the trailing eigenvalues $\hat{\lambda}_{SIR}^{(d+1)} \geq \cdots \geq \hat{\lambda}_{SIR}^{(D)}$ are small is likely to be beneficial for small approximation errors in (32).

The first use of $\hat{W}_{SIR}^T x \in \mathbb{R}^d$ as dimension reduction for GP emulators

$$g_{GP}(\hat{W}_{SIR}^T x) \sim \text{GP}(m(\hat{W}_{SIR}^T x), k(\hat{W}_{SIR}^T x, \hat{W}_{SIR}^T x')) \tag{33}$$

with training sets of the form $\{\hat{W}_{SIR}^T x_i, f(x_i)\}_{i=1}^N$ appears in [103], for the Elliptic PDE experiment described in Section 2.4. This approach was shown to outperform the high-dimensional GP baseline with no dimension reduction (4) and the GP emulators with PCA dimension reduction (12) on both settings considered (i.e. GRF prior for the PDE coefficients (23) with short and long lengthscale, respectively). Note that the performance of the SIR-GP (33) was close to the gradient-based active subspace GP (19). Another recent application is on Bayesian Optimization, where a SIR-GP emulator was used to efficiently maximize the simulator (i.e. using a small number of simulator evaluations $\{\hat{W}_{SIR}^T x_i, f(x_i)\}_{i=1}^N$ selected by a suitable goal-oriented design) in computer experiments with input dimensionality up to $D \leq 20000$ [189]. We will revisit and formally introduce Bayesian Optimization in Chapter 5.

From a theoretical perspective, classical SDR methods such as SIR may struggle with non-Gaussian inputs or symmetric simulator functions. Indeed, a simple example from [18] shows that if $x := [x_1, \dots, x_D] \in \mathbb{R}^D$ and $x \sim \mathcal{N}_D(0, I)$, then for $A := [1, 0, \dots, 0] \in \mathbb{R}^{D\times 1}$ and a symmetric simulator $y = f(x) := (A^T x) = x_1^2$, we have that $\mathbb{E}[x_1|y] = 0$ and thus $\mathbb{E}[x|y] = 0_D$, i.e. SIR is not able to recover the dimension reduction matrix A. More recently, Kernel Sufficient Dimension Reduction methods (KDR) were introduced to overcome these limitations; see [53] for the seminal work on this topic.

Gradient Kernel Dimension Reduction (gKDR)

We consider gradient Kernel Dimension Reduction (gKDR) [54] as a state-of-the-art KDR method, achieving good practical performance, together with theoretical guarantees. Note that our presentation of gKDR is based on [54] and [103]; for a fully rigorous mathematical presentation, please consult the original sources.

To introduce gKDR, we need to start with a very short introduction to kernel methods; recall that 'kernel function' is another term for the covariance functions that are used for GPs, such as the squared-exponential kernel function (4).

Definition 3.1 *([92], Definition 2.3) A symmetric function $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is called a positive definite kernel (over $\mathbb{R}^D$) if for any integer $n \geq 1$, $(c_1, \ldots, c_n) \subset \mathbb{R}$ and $(x_1, \ldots, x_n) \subset \mathbb{R}^D$,*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0.$$

Definition 3.2 *([92], Definition 2.1) Let k be a positive definite kernel on $\mathbb{R}^D$. A Hilbert space $\mathcal{H}_k$ of real-valued functions on $\mathbb{R}^D$ equipped with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_k}$ is called a reproducing kernel Hilbert Space (RKHS) with reproducing kernel k, if the following two conditions are satisfied:*

- *For every $x \in \mathbb{R}^D$, we have $k(\cdot, x) \in \mathcal{H}_k$;*
- *For every $x \in \mathbb{R}^D$ and for all $f \in \mathcal{H}_k$, we have $f(x) = \langle f, k(\cdot, x) \rangle_{\mathcal{H}_k}$ (the reproducing property).*

According to the Moore-Aronszajn theorem [6], for any positive definite kernel $k(\cdot, \cdot) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$, there exists a unique associated RKHS $\mathcal{H}_k$. The function $\Phi(x) : \mathbb{R}^D \to \mathcal{H}_k$ such that $\Phi(x) := k(\cdot, x)$ is known as the (canonical) feature map. Note that the feature space $\mathcal{H}_k$ is often infinite-dimensional and constructing the feature map can be computationally intractable (e.g. for the squared-exponential kernel function (4) [64]). However, for kernel methods algorithms (e.g. the classical Support Vector Machines (SVM) for classification and regression [157]), we are only interested in the computationally tractable inner product $k(x, y) = \langle k(\cdot, y), k(\cdot, x) \rangle_{\mathcal{H}_k}$, which is known as the kernel trick. The inner product $\langle k(\cdot, y), k(\cdot, x) \rangle_{\mathcal{H}_k}$ is often used as a measure of similarity between features of x and y (see Remark 4 in [64]). For a comprehensive introduction to kernel methods, see the textbooks [141, 147, 157]. These methods have been extensively used in many areas of Machine Learning and Statistics, such as image classification [145] and hypothesis testing [74]. The relationship between kernel methods and Gaussian Processes is studied in the recent work [92].

Let $(x, y) \in \mathbb{R}^D \times \mathbb{R}$ be our random variables that generate the input/output data $\{x_i, y_i = f(x_i)\}_{i=1}^N$ and which further satisfy the SDR condition (27):

$$y \perp\!\!\!\perp x | A^T x$$

for some $A \in \mathbb{R}^{D \times d}$. Further, let $k_{\mathcal{X}}$ and $k_{\mathcal{Y}}$ be two kernels defined on the input and output space, respectively, and let $\mathcal{H}_{\mathcal{X}}$ and $\mathcal{H}_{\mathcal{Y}}$ be their corresponding RKHS. For the purpose of this work, and in line with the gKDR implementation from the Python package *mogp_emulator* [1] that we use, both kernels are the standard RBF kernel (or the squared-exponential covariance function) (4) with signal variance $\sigma^2 = 1$ and lengthscale $l > 0$. Note that the lengthscale can be different between the two kernels $k_{\mathcal{X}}$ and $k_{\mathcal{Y}}$.

Similar to the definition of the usual covariance matrix between two random vectors taking values in $\mathbb{R}^m$, but now with two random vectors $k_{\mathcal{X}}(\cdot, x) \in \mathcal{H}_{\mathcal{X}}$ and $k_{\mathcal{Y}}(\cdot, y) \in \mathcal{H}_{\mathcal{Y}}$ taking values in $\mathcal{H}_{\mathcal{X}}$ and $\mathcal{H}_{\mathcal{Y}}$, respectively, we can define the cross-covariance operator $C_{yx} : \mathcal{H}_{\mathcal{X}} \to \mathcal{H}_{\mathcal{Y}}$ such that

$$\langle h_2, C_{yx} h_1 \rangle_{\mathcal{H}_{\mathcal{Y}}} = \mathbb{E}_{yx}[h_2(y) h_1(x)]$$

holds for any $h_1 \in \mathcal{H}_{\mathcal{X}}$, $h_2 \in \mathcal{H}_{\mathcal{Y}}$. The covariance operator $C_{xx} : \mathcal{H}_{\mathcal{X}} \to \mathcal{H}_{\mathcal{X}}$ is defined similarly. While not necessarily the most intuitive, we use this definition as it is the most useful for our exposition. See Section 3.2 in [117] for alternative definitions, together with a discussion about cross-covariance operators on RKHSs.

Theorem 2 and Corollary 3 in [53] show that for $g \in \mathcal{H}_{\mathcal{Y}}$,

$$C_{xx} \mathbb{E}[g(y) | X = \cdot] = C_{xy} g$$

and

$$\mathbb{E}[g(y) | X = \cdot] = C_{xx}^{-1} C_{xy} g,$$

respectively. Note that these results require $\mathbb{E}[g(y) | X = \cdot] \in \mathcal{H}_{\mathcal{X}}$, together with the injectivity of C_{xx} (which guarantees the existence of the inverse operator $C_{xx}^{-1} C_{xx} f = C_{xx} C_{xx}^{-1} f = f$ for all $f \in \mathcal{H}_{\mathcal{X}}$); while the latter condition is satisfied under mild assumptions on $p(x)$ and $k(\cdot, x)$ [54] (including the squared-exponential kernel considered), the former condition is very difficult to check, as discussed in the theoretical monograph [95]. Nonetheless, using the general kernel property that

$$\frac{\partial f}{\partial x} = \langle f, \frac{\partial}{\partial x} k(\cdot, x) \rangle_{\mathcal{H}_\mathcal{X}}$$

holds for any $f \in \mathcal{H}_\mathcal{X}$ (Lemma 4.34, [157]), we have that

$$\frac{\partial}{\partial x} \mathbb{E}[g(y)|X = x] = \langle C_{xx}^{-1} C_{xy} g, \frac{\partial}{\partial x} k(\cdot, x) \rangle_{\mathcal{H}_\mathcal{X}} = \langle g, C_{yx} C_{xx}^{-1} \frac{\partial k_\mathcal{X}(\cdot, x)}{\partial x} \rangle_{\mathcal{H}_\mathcal{Y}}, \tag{34}$$

using the self-adjoint property $\langle C_{xx}^{-1} f, h \rangle_{\mathcal{H}_\mathcal{X}} = \langle f, C_{xx}^{-1} h \rangle_{\mathcal{H}_\mathcal{X}}$ (for all $f, h \in \mathcal{H}_\mathcal{X}$) of the inverse covariance operator C_{xx}^{-1}, and the adjoint property $\langle C_{xy} g, f \rangle_{\mathcal{H}_\mathcal{X}} = \langle g, C_{yx} f \rangle_{\mathcal{H}_\mathcal{Y}}$ (for all $f \in \mathcal{H}_\mathcal{X}$ and $g \in \mathcal{H}_\mathcal{Y}$) of the cross-covariance operator C_{yx} (see Section 3.2 in [117]).

Let us define $\Psi(x)$ for $x \in \mathbb{R}^D$ such that $\Psi(x) = \mathbb{E}[k_\mathcal{Y}(\cdot, y)|X = x] = \int k_\mathcal{Y}(\cdot, y) p(y|x) dy$. This is an example of a kernel mean embedding [12, 153], as $\Psi(x) \in \mathcal{H}_\mathcal{Y}$ for the squared-exponential (RBF) kernel $k_\mathcal{Y}(\cdot, y)$ considered here ($\Psi(x) = \int k_\mathcal{Y}(\cdot, y) p(y|x) dy$). If we plug $g = k_\mathcal{Y}(\cdot, y') \in \mathcal{H}_\mathcal{Y}$ into (34) for any $y' \in \mathbb{R}$, we have that

$$\frac{\partial \Psi(x)}{\partial x}(y') = \langle k(\cdot, y'), C_{yx} C_{xx}^{-1} \frac{\partial k_\mathcal{X}(\cdot, x)}{\partial x} \rangle_{\mathcal{H}_\mathcal{Y}} = C_{yx} C_{xx}^{-1} \frac{\partial k_\mathcal{X}(\cdot, x)}{\partial x}(y'), \tag{35}$$

using the reproducing kernel property $h(y') = \langle k_\mathcal{Y}(\cdot, y'), h \rangle_{\mathcal{H}_\mathcal{Y}}$ for all $h \in \mathcal{H}_\mathcal{Y}$. From now on, we will simply write (35) as $\frac{\partial \Psi(x)}{\partial x} = C_{yx} C_{xx}^{-1} \frac{\partial k_\mathcal{X}(\cdot, x)}{\partial x}$.

Note that we can use the SDR condition (27) to write

$$\Psi(x) = \int k_\mathcal{Y}(\cdot, y) p(y|x) dy = \int k_\mathcal{Y}(\cdot, y) p(y|A^T x) dy.$$

By exchanging the order of differentiation and integration, we have that

$$\frac{\partial \Psi(x)}{\partial x} = \int k_\mathcal{Y}(\cdot, y) \frac{\partial p(y|A^T x)}{\partial x} dy = A \int k_\mathcal{Y}(\cdot, y) \left. \frac{\partial p(y|z)}{\partial z} \right|_{z = A^T x} dy. \tag{36}$$

From (35) and (36) with $\int k_{\mathcal{Y}}(\cdot,y)\frac{\partial p(y|z)}{\partial z}\Big|_{z=A^T x} dy := \Xi(x) : \mathbb{R}^D \to \mathcal{H}_{\mathcal{Y}}$ and by taking the inner product $\langle \cdot,\cdot\rangle_{\mathcal{H}_{\mathcal{Y}}}$ in both equations, we have that

$$\begin{aligned} A^T\langle \Xi(x), \Xi(x)\rangle_{\mathcal{H}_{\mathcal{Y}}} A &= \left\langle C_{yx}C_{xx}^{-1}\frac{\partial k_{\mathcal{X}}(\cdot,x)}{\partial x}, C_{yx}C_{xx}^{-1}\frac{\partial k_{\mathcal{X}}(\cdot,x)}{\partial x}\right\rangle_{\mathcal{H}_{\mathcal{Y}}} \\ &= \left\langle \frac{\partial k_{\mathcal{X}}(\cdot,x)}{\partial x}, C_{xx}^{-1}C_{xy}C_{yx}C_{xx}^{-1}\frac{\partial k_{\mathcal{X}}(\cdot,x)}{\partial x}\right\rangle_{\mathcal{H}_{\mathcal{X}}} := M(x), \end{aligned} \tag{37}$$

using again the adjoint property of the covariance operators discussed in the context of (34). Therefore, the d eigenvectors corresponding to non-zero eigenvalues of the symmetric matrix $M(x) \in \mathbb{R}^{D\times D}$ are contained in the SDR subspace colspan(A) corresponding to $A \in \mathbb{R}^{D\times d}$.

Our goal is now to estimate $M(x)$ and its eigenspace using the observations $\{x_i, y_i = f(x_i)\}_{i=1}^N$. For this, we need to define the sample-based covariance operators. According to Equation 3 in [54], for every $f \in \mathcal{H}_{\mathcal{X}}$:

$$\hat{C}_{yx}^{(N)} f = \frac{1}{N}\sum_{i=1}^N k_{\mathcal{Y}}(\cdot,y_i)\langle k_{\mathcal{X}}(\cdot,x_i), f\rangle_{\mathcal{H}_{\mathcal{X}}} = \frac{1}{N}\sum_{i=1}^N f(x_i)k_{\mathcal{Y}}(\cdot,y_i),$$

and similarly for $\hat{C}_{xx}^{(N)}$ and $\hat{C}_{xy}^{(N)}$. These estimators are $\sqrt{N}-$consistent for the corresponding population-based covariance operators, according to Theorem 10 and Theorem 11 in [75]. Using these empirical covariance operators, we arrive at the following regularized version of the sample-based estimator $\hat{M}_N(x)$ for $M(x)$ in (37):

$$\begin{aligned} \hat{M}_N(x) &= \left\langle \frac{\partial k_{\mathcal{X}}(\cdot,x)}{\partial x}, (\hat{C}_{xx}^{(N)} + \epsilon_N I)^{-1}\hat{C}_{xy}^{(N)}\hat{C}_{yx}^{(N)}(\hat{C}_{xx}^{(N)} + \epsilon_N I)^{-1}\frac{\partial k_{\mathcal{X}}(\cdot,x)}{\partial x}\right\rangle_{\mathcal{H}_{\mathcal{X}}} \\ &= \nabla k_X(x)^T(G_X + N\epsilon_N I)^{-1}G_Y(G_X + N\epsilon_N I)^{-1}\nabla k_X(x), \end{aligned}$$

where G_X and G_Y are the kernel matrices $(G_X)_{ij} = k_{\mathcal{X}}(x_i, x_j)$ and $(G_{\mathcal{Y}})_{ij} = k_{\mathcal{Y}}(y_i, y_j)$, respectively, and $\nabla k_X(x) = \left(\frac{\partial k_{\mathcal{X}}(x_1,x)}{\partial x}, \ldots, \frac{\partial k_{\mathcal{X}}(x_N,x)}{\partial x}\right)$. The regularization parameter ϵ_N accounts for a potentially ill-conditioned kernel matrix G_X (e.g. when two columns are close to each other in case of nearby points x_i and x_j). Note that ϵ_N needs to converge to zero as $N \to \infty$ at a specific rate such that the convergence in probability $||\hat{M}_N(x) - M(x)||_F \to 0$ as

$N \to \infty$ is achieved at a rate between $N^{-1/4}$ and $N^{-1/3}$ (Theorem 2, [54]). In practice, ϵ_N can be tuned by the user. As an example, $\epsilon_N = 10^{-8}$ is the default setting from the *mogp_emulator* package.

Finally, recall from (37) that the eigenvectors of $M(x)$ corresponding to non-zero eigenvalues are contained in the SDR subspace colspan(A) for any $x \in \mathbb{R}^D$. Therefore, [54] suggests taking the average of $M(x_i)$ over all data points x_i, in order to arrive at the final gKDR estimator:

$$\begin{aligned}\tilde{M}_N &= \frac{1}{N}\sum_{i=1}^{N} \hat{M}_N(x_i) \\ &= \frac{1}{N}\sum_{i=1}^{N} \nabla k_X(x_i)^T (G_X + n\epsilon_N I)^{-1} G_Y (G_X + N\epsilon_N I)^{-1} \nabla k_X(x_i),\end{aligned}$$

which also shares similar theoretical guarantees with $\hat{M}_N(x)$ in terms of the convergence in probability $||\tilde{M}_N - \mathbb{E}[M(x)]||_F \to 0$ as $N \to \infty$ (Theorem 2, [54]). Similar to PCA (Section 2.2), AS (Section 2.3), and SIR (the other SDR method considered in this section), we consider the eigendecomposition of the positive semidefinite matrix $\tilde{M}_N = \tilde{W}\text{diag}(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_D)\tilde{W}^T$. The columns of the gKDR dimension reduction matrix $\tilde{W}_{gKDR} := [\tilde{w}_1, \ldots, \tilde{w}_d] \in \mathbb{R}^{D\times d}$ are the eigenvectors corresponding to the largest eigenvalues $\tilde{\lambda}_1 \geq \ldots \tilde{\lambda}_d \geq 0$.

If we write $W_{gKDR} \in \mathbb{R}^{D\times d}$ for the matrix whose columns are the dominant d eigenvectors of $M(x) := W\text{diag}(\lambda_1, \ldots, \lambda_D)W^T$, Proposition 1 in [103] shows that there exists a low-dimensional approximation $g(W_{gKDR}^T x) \approx f(x)$; the upper bound of this approximation $C(\sum_{i=d+1}^{D} b_i \lambda_i^2)^{1/2}$ is similar to the corresponding active subspace approximation (15), although the dependency on the trailing eigenvalues is more complicated due to the unknown weights b_i. In addition, Proposition 2 in [103] shows that the gKDR dimension reduction $\tilde{W}_{gKDR}$ also satisfies the low-dimensional approximation property $\tilde{g}(\tilde{W}_{gKDR}^T x) \approx f(x)$ for some $\tilde{g} : \mathbb{R}^d \to \mathbb{R}$; the upper bound

$$\frac{4}{\lambda_d - \lambda_{d+1}} N^{-1/3} \Big(\sum_{i=1}^{d} b_i \lambda_i^2\Big)^{1/2} + \Big(\sum_{i=d+1}^{D} b_i \lambda_i^2\Big)^{1/2}$$

resembles some similarities with the active subspace analogue (17) and (18). However, due to the unknown dependency on b_i, the work [103] does not use the eigengap $\lambda_d - \lambda_{d+1}$ to choose the low dimensionality d, and prefers for example a cross-validation approach (see Section 2.1), where d is chosen to maximize predictive performance of the resulting GP emulator:

$$g_{GP}(\tilde{W}_{gKDR}^T x) \sim \mathrm{GP}(m(\tilde{W}_{gKDR}^T x), k(\tilde{W}_{gKDR}^T x, \tilde{W}_{gKDR}^T x')), \tag{38}$$

which is constructed using the training set $\{\tilde{W}_{gKDR}^T x_i, f(x_i)\}_{i=1}^N$. As mentioned in the introduction of this section, the GP (38) has been used in applications such as tsunami modelling [103] and wind turbine modelling [192].

3.2 UNSUPERVISED DIMENSION REDUCTION

As mentioned in Section 1.5, unsupervised dimension reduction methods do not require access to simulator evaluations; this is particularly desirable in the small training data scenarios, where the supervised dimension reduction methods might not have enough data to accurately estimate a potentially existing low-dimensional structure (e.g. $A \in \mathbb{R}^{D\times d}$ such that $f(x) = g(A^T x)$ for some $g : \mathbb{R}^d \to \mathbb{R}$). Section 2.2 contains a detailed introduction to PCA, which is probably the most popular unsupervised linear dimension reduction method; one famous non-linear extension of PCA is known as kernel PCA [142], which is related to the brief introduction on kernel methods from Section 3.1. Given $x \sim p_D(x)$, Kernel PCA involves performing standard PCA in the infinite dimensional feature space $k(\cdot, x) \in \mathcal{H}_k$; see [142] for an application on feature extraction for natural images, [115] for an application in image denoising, and [146] for theoretical guarantees of kernel PCA in the empirical setting, where only samples $\{x_i\}_{i=1}^M$ are available. In this work, we will only focus on linear dimension reduction methods; as mentioned in Section 1.5, we are interested in computer simulators which have a latent low-dimensional linear structure (2).

We will now present a collection of additional unsupervised linear dimension reduction methods, which will be used in our computer experiments.

3.2.1 *Johnson-Lindenstrauss (JL)*

The Johnson-Lindenstrauss (JL) embedding $W_{JL} \in \mathbb{R}^{D\times d}$ was introduced in [90] as a classical dimension reduction method that tries to preserve the pairwise distances between a collection of high-dimensional points $\{x_i\}_{i=1}^N$ in the low dimensional space $\{W_{JL}^T x_i\}_{i=1}^N$. We will present a formal result, which will also be of interest for another dimension reduction method considered in Section 3.2.2.

Theorem 3.1 *([40], Theorem 2.1) For any $0 < \epsilon < 1$ and positive integer N, let $d \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1}\log N$ and $M \in \mathbb{R}^{D\times d}$ be a random matrix with independent standard Gaussian entries $M_{ij} \sim \mathcal{N}(0,1)$. For this setting, the random matrix $W_{JL} := \frac{1}{\sqrt{d}}M$ is a Johnson-Lindenstrauss transform, i.e. for any subset $V \subset \mathbb{R}^D$ with N elements ($|V| = N$), the following approximation*

$$(1-\epsilon)||x-x'||_2^2 \leq ||W_{JL}^T(x-x')||_2^2 \leq (1+\epsilon)||(x-x')||_2^2 \tag{39}$$

holds for every $x, x' \in V$ with high probability.

In other words, given sufficiently large d ($d \propto \log N$), JL succeeds in preserving the pairwise $||.||_2-$distances between any collection of high-dimensional points $\{x_i\}_{i=1}^N$ in the low dimensional space $\{W_{JL}^T x_i\}_{i=1}^N$. Note that the covariance function of the low-dimensional GP

$$g_{GP}(W_{JL}^T x) \sim \mathrm{GP}(m(W_{JL}^T x), k(W_{JL}^T x, W_{JL}^T x')) \tag{40}$$

is expressed as $k(W_{JL}^T x, W_{JL}^T x') = \sigma^2 \exp\left(\frac{-||W_{JL}^T(x-x')||_2^2}{2l^2}\right)$. Therefore, it only depends on the distances between points, and hence it might be able to accurately model the relationship between $f(x)$ and $f(x')$, while hopefully requiring less training data $\{x_i, f(x_i)\}_{i=1}^N$ compared to the high-dimensional GP with no dimension reduction (3). JL methods have been successfully applied in dimension reduction for kernel methods; see [26] for theoretical guarantees on the approximation

$$k(W_{JL}^T x, W_{JL}^T x') = (W_{JL}^T x)^T W_{JL}^T x'$$

of a (high-dimensional) dot-product kernel $k(x, x') = x^T x'$, together with various computer experiments with natural images $\{x_i\}_{i=1}^N$ as inputs.

3.2.2 *Count Sketch (CS)*

Count Sketch (CS) is another unsupervised dimension reduction method based on random embeddings, which was recently introduced in the context of Gaussian Processes in [118]. The dimension reduction matrix $W_{CS} \in \mathbb{R}^{D \times d}$ is a sparse matrix with only one non-zero element in each row; the non-zero element is sampled at random from $\{-1, +1\}$. As presented in [118] for $D = 5$ and $d = 3$, one example of $W_{CS}^T x = z$ is

$$\begin{pmatrix} 0 & 0 & 1 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} x_3 - x_4 \\ -x_1 \\ x_2 + x_5 \end{pmatrix} = z.$$

We see that each element of x appears exactly once in z, with the sign determined by the $\{-1, +1\}$ element from the projection matrix.

Similar to the SDR assumption (27), CS was introduced in GPs for the case where there exists a low-dimensional link function $f(x) = g(A^T x)$ for some $A \in \mathbb{R}^{D \times d_e}$ with $d_e \ll D$. Theorem 2 in [118] shows that if the random embedding $W_{CS} \in \mathbb{R}^{D \times d}$ satisfies the following version of (39):

$$(1 - \epsilon)||Ay||_2^2 \leq ||W_{CS}^T Ay||_2^2 \leq (1 + \epsilon)||Ay||_2^2$$

for all $y \in \mathbb{R}^d$ (i.e. W_{CS} is an $\epsilon-$subspace embedding for A), the resulting CS-GP emulator

$$g_{GP}(W_{CS}^T x) \sim \text{GP}(m(W_{CS}^T x), k(W_{CS}^T x, W_{CS}^T x')) \tag{41}$$

is a good approximation for the (optimal) low-dimensional GP emulator

$$g_{GP}(A^T x) \sim \text{GP}(m(A^T x), k(A^T x, A^T x')).$$

Note that we require a low-dimensionality $d = O(d_e^2/\epsilon^2)$ for W_{CS} to be an $\epsilon-$subspace embedding for the true d_e-dimensional subspace A, according to Theorem 3 in [120] (upper bound) and Theorem 16 in [119] (lower bound).

Since our goal is GP emulation $g_{\mathrm{GP}}(W_{CS}^T x) \approx f(x)$, the training set for the GP (41) consists, as usual, of pairs of the form $\{W_{CS}^T x_i, f(x_i)\}_{i=1}^N$. The work [118] instead focused on Bayesian Optimization (BO), which involves maximizing the simulator function $f(W_{CS}W_{CS}^T x)$ over the $d-$dimensional random subspace W_{CS} using the GP (41) and training sets $\{W_{CS}^T x_i, f(W_{CS}W_{CS}^T x_i)\}_{i=1}^N$. That is, an additional lift $W_{CS} : \mathbb{R}^d \to \mathbb{R}^D$ was applied to the inputs $W_{CS}^T x_i$ in order to generate the outputs. Nonetheless, we will check whether the remarkable practical performance of this method in BO applications such as parameter optimization for a $D = 100$-dimensional neural network [118] translates to our GP emulation experiments.

3.2.3 *Randomly Projected Additive GPs (RPA-GP)*

So far, we have only considered $g_{\mathrm{GP}}(W^T x)$ using a *single* random embedding $W \in \mathbb{R}^{D\times d}$. We end this section by considering a more flexible approach called 'Randomly Projected Additive GPs' (RPA-GP [43]), which uses an additive composition of J low-dimensional Gaussian random embeddings $W_j \in \mathbb{R}^{D\times d}$ with independent entries $(W_j)_{ik} \sim \mathcal{N}(0, 1/d)$ for $i \in \{1, \dots, D\}$ and $k \in \{1, \dots, d\}$. We write the resulting covariance function as:

$$k(x, x') = \sum_{j=1}^{J} k_j(W_j^T x, W_j^T x'), \tag{42}$$

where each kernel function k_j is the standard squared exponential covariance function (4). Note that this is a direct generalization of the Johnson-Lindenstrauss GP (40). It is shown in [43] that the kernel (42) converges to the high-dimensional inverse multiquadratic (IMQ) kernel

$$k_{IMQ}(x, x') := \frac{1}{\sqrt{1 + ||x - x'||^2}} \tag{43}$$

as $J \to \infty$ at a rate of $J^{-1/2}$. Furthermore, RPA-GP with a constant mean function $m(x) = C$ and covariance function (42) is able to match the predictive performance of the high-dimensional IMQ-GP (43) with only relatively few projections J of very low dimensionality d on a variety of real-world simulators. Note that in a couple of small training data experiments, RPA-GP outperformed the high-dimensional (no dimension reduction) IMQ-GP.

Finally, [43] also presents an extension of RPA-GP named 'Diverse Projected Additive GPs' (DP-GP), which uses an additional optimization algorithm that encourages the projections to be more diverse by trying to maximize the distance between them. The resulting DP-GP covariance function is written as:

$$k(x, x') = \sum_{j=1}^{J} k_j(U_j^T x, U_j^T x'), \tag{44}$$

where $\{U_1, \dots, U_J\}$ are the deterministic projections obtained via the Diverse Projections (DP) optimization algorithm (Equation 7, [43]). Note that while these projections are deterministic, they are still part of the unsupervised dimension reduction framework, as no simulator evaluations are used to construct them. The resulting DPA-GP with a constant mean function $m(x) = C$ and covariance function (44) was shown to significantly outperform the original RPA-GP (42) in various computer experiments [43].

3.3 COMPUTER EXPERIMENTS

We will look at a series of synthetic and real-data simulators $y = f(x)$, most of which were considered in [63]. In all of these experiments, we know that $f(x) \approx g(\hat{W}_{AS}^T x)$ for some low-dimensional function $g : \mathbb{R}^d \to \mathbb{R}$, where the matrix $\hat{W}_{AS} \in \mathbb{R}^{D \times d}$ has been extracted as in (16) using the gradient-based active subspace method.

We perform a method comparison between the predictive performance of the GP emulators resulting from all the supervised and unsupervised dimension reduction methods discussed in this chapter, including the methods introduced in Chapter 2. From this perspective, our work complements [103], which focused on sufficient dimension reduction (SDR) methods. The work [63] only covers gradient-free supervised dimension reduction methods that are motivated from an active subspace perspective; these methods were discussed in Section 2.3.

The synthetic experiments are based on a quadratic function of the form

$$z = A^T x \tag{45}$$

$$f(x) = z^T V z + bz + c + \epsilon. \tag{46}$$

As mentioned, the true low-dimensional embedding $A \in \mathbb{R}^{D\times d}$ is recovered by the AS method ($\hat{W}_{AS} \approx A$). To construct the quadratic coefficients $V \in \mathbb{R}^{d\times d}$, $b \in \mathbb{R}^d$ and $c \in \mathbb{R}$, we independently sample all their entries from the standard Normal distribution $\mathcal{N}(0,1)$. Also, we fix the noise level $\epsilon \sim \mathcal{N}(0, 25\cdot 10^{-4})$, which accounts for the variation of $f(x)$ in the inactive directions $z^\perp \in \mathbb{R}^{D-d}$. We consider eight scenarios, based on the input dimensionality D and the true low dimensionality d, i.e.

$$(D,d) \in \{(10,1),(10,2),(25,1),(25,2),(50,1),(50,2),(100,1),(100,2)\}. \tag{47}$$

In terms of the real-data simulators, four examples are considered:

- ONERA M6 aerodynamics simulator [107], where $y = f(x)$ the lift coefficient for a wing model; $D = 50, d = 1$ (i.e. $x \in \mathbb{R}^{50}$, $\hat{W}_{AS} \in \mathbb{R}^{50\times 1}$)
- HIV long-term model [105], where $y = f(x)$ is the cell count at time $t = 3400$; $D = 27, d = 1$
- Elliptic-PDE [30], which is the simulator $f_{\lambda^L, u^L}(x)$ considered in Section 2.4 for a Gaussian random field (GRF) prior with long lengthscale (23) for the PDE coefficients; $D = 100, d = 1$
- Hydrological flow PDE model (with initially saturated soil) [68], where $y = f(x)$ is the water (runoff) volume. The inputs $x \sim \mathcal{N}(0, I_D)$ gener-

ate a GRF prior of short lengthscale (21) for the PDE coefficients that describe the subsurface permeability; $D = 100, d = 1$.

All the synthetic and real-data simulator settings are identical to [63], except for the additional hydrological flow experiment. We will perform a comparative study in order to verify the ability of the GPs using various dimension reduction methods to approximate the different simulators $f(x)$. We expect that the gradient-based active subspace method $g_{\mathrm{GP}}(\hat{W}_{AS}^T x)$ (19) will perform best for all the simulators $f(x)$, since the existence of a low-dimensional approximation $f(x) \approx g(\hat{W}_{AS}^T x)$ was proved in the existing literature. The Github page [27] provides access to all the active subspaces $\hat{W}_{AS} \in \mathbb{R}^{D \times d}$ considered in this work. Furthermore, we will also include the high-dimensional (no dimension reduction) $g_{\mathrm{GP}}(x)$ (3) as a standard baseline in all the experiments.

Note that in all the experiments, the GP hyperparameters θ which we introduced in Section 2.1 are initialized by their default values in GPyTorch [61]; for simplicity, we choose a constant GP prior mean function. As discussed in Section 2.1, they are further optimized by maximizing the marginal likelihood (5) using the LBFGS algorithm or the gradient descent algorithm. When introducing the different dimension reduction methods $W \in \mathbb{R}^{D \times d}$, we have presented various criteria for choosing the low-dimensionality d; we have tried these values, together with various alternatives (e.g. $W \in \mathbb{R}^{D \times 2d}$, $W \in \mathbb{R}^{D \times 5d}$, or $W \in \mathbb{R}^{D \times 10d}$). However, we have found that the performance was relatively insensitive to this choice, and thus we have not included all of these results. For simplicity, one common feature for all the dimension reduction methods is the use of the low-dimensionality d which corresponds to the dimensionality chosen by the gradient-based AS method $\hat{W}_{AS} \in \mathbb{R}^{D \times d}$. The remaining experimental details can be summarized as follows:

- Sliced Inverse Regression (SIR): number of slices $R = 10$ as default in the Python package *sliced* [2]; each slice contains approximately the same number of samples, as motivated in Section 3.1.1.

- Gradient Kernel Dimension Reduction (gKDR): lengthscale parameters $\sigma_x = M_x$, $\sigma_y = M_y$ for the kernels $k_{\mathcal{X}}(\cdot, x)$ and $k_{\mathcal{Y}}(\cdot, y)$, respectively, where M_x and M_y are the median values of pairwise distances of the training inputs $\{x_i\}_{i=1}^N$ and outputs $\{y_i = f(x_i)\}_{i=1}^N$, respectively. This is the default choice in the Python package *mogp_emulator* [1].

- Randomly Projected Additive GPs (RPA-GP): As seen in the experiments from [43], we have chosen $J = 20$ one-dimensional projections $(W_j \in \mathbb{R}^{D\times 1}, U_j \in \mathbb{R}^{D\times 1})_{j=1}^J$ for both versions RPA-GP (42) and DPA-GP (44), respectively.

- Principal Component Analysis (PCA): Since for all the experiments we only have access to the datasets $\{x_i, f(x_i)\}_{i=1}^N$ of simulator evaluations, and not to the underlying distribution that generated the independent samples $x_i \sim p(x)$, we use the empirical covariance matrix of $\{x_i\}_{i=1}^N \sim p(x)$ to perform PCA.

For the GP (20), we use the term 'embedding learning' for the procedure of maximizing the marginal likelihood (5) with respect to (the embedding) $W_{\text{GP}} \in \mathbb{R}^{D\times d}$. The resulting solution $\hat{W}_{\text{GP}} \in \mathbb{R}^{D\times d}$ that (approximately) maximizes the marginal likelihood (5) is obtained using the gradient descent algorithm. We consider two approaches for embedding learning, as follows:

- Type II maximum likelihood embedding learning with the embedding initialized at random (ML-II-LE): For the GP (20), the embedding $W_{GP} \in \mathbb{R}^{D\times d}$ is implemented as a single-layer (linear) neural network as part of the Deep Kernel Learning [179] routine in GPyTorch [61], with its corresponding default random initialization.

- Type II maximum likelihood embedding learning with the embedding initialized via gradient Kernel Dimension Reduction (gKDR-LE): The embedding $W_{GP} \in \mathbb{R}^{D\times d}$ in (20) is initialized with the solution $\tilde{W}_{gKDR}$ of the gKDR method. Note that for a high-dimensional input space $D = 100$, due to the RAM memory limitations provided by the gKDR implementation from the Python package *mogp_emulator* [1], we replace gKDR-LE with SIR-LE, i.e. the embedding $W_{\text{GP}} \in \mathbb{R}^{D\times d}$ is initialized with the solution $\hat{W}_{SIR}$ of the SIR method.

We trained the resulting GP emulators over five trials, where each trial consists of five training sets $\{x_i, y_i = f(x_i)\}_{i=1}^{N_{train}}$ of various sizes N_{train}. According to [63], we have chosen the training set sizes ranging from one to five times the input dimensionality, i.e $N_{train} \in \{D, 2D, 3D, 4D, 5D\}$. The performance of each GP was tested on a separate test set $\{x_i^\star, y_i^\star = f(x_i^\star)\}_{i=1}^{N_{test}}$. We compare the performance in terms of predictive accuracy via the test root mean squared error (RMSE), which we define as

$$\text{RMSE} := \sqrt{\frac{1}{N_\star}\sum_{i=1}^{N_\star}(\bar{m}(W^T x_i^\star) - y_i^\star)^2}$$

for any dimension reduction matrix $W \in \mathbb{R}^{D\times d}$, or

$$\text{RMSE} := \sqrt{\frac{1}{N_\star}\sum_{i=1}^{N_\star}(\bar{m}(x_i^\star) - y_i^\star)^2}$$

for the high-dimensional (NO-DR) GP baseline (3). In addition to the predictive accuracy, we also consider the predictive uncertainty quantification via the test negative predictive log-density (NPLD), which we define as

$$\text{NPLD} := -\sum_{i=1}^{N_\star}\log\mathcal{N}(y_i^\star|\bar{m}(W^T x_i^\star), \bar{k}(W^T x_i^\star, W^T x_i^\star))$$

for any dimension reduction matrix $W \in \mathbb{R}^{D\times d}$, or

$$\text{NPLD} := -\sum_{i=1}^{N_\star}\log\mathcal{N}(y_i^\star|\bar{m}(x_i^\star), \bar{k}(x_i^\star, x_i^\star))$$

for the high-dimensional GP baseline.

For any supervised dimension reduction method $W \in \mathbb{R}^{D\times d}$, we also present the First Subspace Angle (FSA) as a measure of discrepancy between the subspace spanned by the columns of W and gradient-based active subspace matrix $\hat{W}_{AS}$. FSA was used in [63] to check that W gets closer to $\hat{W}_{AS}$ as the training size N increases. For two semi-orthogonal matrices $M_1 \in \mathbb{R}^{D\times d}$ and $M_2 \in \mathbb{R}^{D\times d}$, let $M := (M_2, M_2^{\perp}) \in \mathbb{R}^{D\times D}$ ($M^T M = MM^T = I_D$). We can compute the FSA between the column spaces of M_1 and M_2 as

$$\text{FSA}(M_1, M_2) = ||M_1^T M_2^{\perp}||_F,$$

where $||\cdot||_F$ is the Frobenius norm defined in Section 2.2.

For each pair (D, d) from (47), the corresponding quadratic function experiment (46) can be described as follows. Out of a total of $N = 1000$ points $\{x_i, y_i = f(x_i)\}_{i=1}^N$ from [63], we have selected at random five trials, where each trial consists of five training sets of size $N_{train} = \{D, 2D, 3D, 4D, 5D\}$, and used the remaining points $N_{test} = \{N-D, N-2D, N-3D, N-4D, N-5D\}$ for testing. We plot the resulting test RMSE, test NPLD, and first subspace angle (FSA) versus the training set size for all the dimension reduction methods considered (lower values are better); the results are averaged over the five trials. Note that the plots use 'RP' and 'DP-ARD' to denote RPA-GP and DPA-GP, respectively; the FSA plots use 'true A' to denote the gradient-based active subspace $\hat{W}_{AS}$, whereas RMSE and NPLD plots use 'true embedding low dim GP' to denote $g_{\text{GP}}(\hat{W}_{AS}^T x)$ (19). These plots are presented in Figs. 2-25.

Our conclusions are summarized as follows:

- The unsupervised dimension reduction methods Count Sketch (CS), Johnson-Lindenstrauss (JL), and Principal Component Analysis (PCA), return virtually identical performance on all benchmarks. Note that although we do not have access to the distribution $\{x_i\}_{i=1}^N \sim p(x)$, we suspect that $x_i \sim \mathcal{N}_D(0, I)$, which would explain the poor performance of PCA.

- The other two unsupervised dimension reduction methods considered, i.e. Randomly Projected Additive GPs (RPA-GP, 42) and Diverse Projected Additive GPs (DPA-GP, 44) showed very unstable performance, failing to match the performance of the high-dimensional GP baseline (NO-DR) in almost all cases.

- For both $d = \{1, 2\}$, the performance gap between the gradient-based active subspace method $g_{\text{GP}}(\hat{W}_{AS}^T x)$ and the high-dimensional baseline $g_{\text{GP}}(x)$ increased with D, which further motivates the use of dimension reduction methods. Note that for $D = 100$, the high-dimensional baseline $g_{\text{GP}}(x)$ performs identically to the unsupervised dimension reduction methods CS, JL, and PCA.

- As a difference between the two different cases $d = 1$ and $d = 2$, the sufficient dimension reduction methods SIR and gKDR significantly outperformed the unsupervised ones (PCA, CS, JL) for $d = 2$, whereas for $d = 1$ the performance gap disappeared, as the methods performed relatively similarly.

- Gradient Kernel Dimension Reduction (gKDR) performed better than Sliced Inverse Regression (SIR) in terms of the First Subspace Angle (FSA) criterion for agreeing with the gradient-based AS embedding $\hat{W}_{AS}$. In terms of predictive performance, gKDR outperformed SIR for larger training data regime $N_{train} = 5D$ and $(D, d) \in \{(10, 1), (10, 2), (25, 1)\}$, cases where we can also notice a large advantage in terms of FSA. Regarding the small training regime $N_{train} = D$, SIR performs better, as gKDR tends to underestimate predictive uncertainty (i.e. large NPLD).

- Type II maximum likelihood embedding learning provided unstable performance in each experiment. In the small/medium training regime $N_{train} \leq 3D$, both ML-II-LE with random initialization and gKDR-LE/SIR-LE with gKDR or SIR initialization severely underestimate predictive uncertainty (high NPLD), which is probably an artefact of using maximum likelihood with small datasets in high dimensions for $W_{GP} \in \mathbb{R}^{D\times d}$ in (20). This is especially worrying if we want to use this embedding learning method for uncertainty based goal-oriented designs such as Bayesian Optimization (BO), which usually operates in the small data regime. Note that this problem can potentially be alleviated by treating $W_{GP} \in \mathbb{R}^{D\times d}$ in (20) as a latent variable, since [63] shows that predictive uncertainties can be better calibrated in this way.

- Note that for $(D, d) \in \{(10, 1), (10, 2), (25, 1)\}$ and $N_{train} = 5D$, where $\tilde{W}_{gKDR}$ approximates $\hat{W}_{AS}$ with reasonable accuracy according to the FSA criterion, gKDR-LE managed to stabilize training (i.e. all five training trials shown good predictive performance), and furthermore gKDR-LE performed almost on par with the gradient-based $g_{\mathrm{GP}}(\hat{W}_{AS}^T x)$ on all five trials. In general, gKDR-LE (or SIR-LE for $D = 100$) performed better than the randomly initialized ML-II-LE method, although training was not fully stabilized, i.e. there were still training sets among the five trials which lead to poor predictive performance.

- Nonetheless, gKDR-LE was the best performing dimension reduction method for

$$(D, d) \in \{(10, 1), (10, 2), (25, 1), (50, 1), (50, 2)\}$$

  and $N_{train} = 5D$, significantly outperforming the high-dimensional GP baseline, as well as both $g_{\mathrm{GP}}(\tilde{W}_{gKDR}^T x)$ with the fixed embedding $\tilde{W}_{gKDR}$ and $g_{\mathrm{GP}}(\hat{W}_{GP}^T x)$ with the embedding $\hat{W}_{GP}$ obtained via maximum likelihood with random initialization. Therefore, we can conclude that when we have enough data $\{(x_i, y_i)\}_{i=1}^{N_{train}} \to \tilde{W}_{gKDR}$ for gKDR to provide a reasonable approximation for $\hat{W}_{AS}$, initializing the maximum likelihood embedding learning procedure with $\tilde{W}_{gKDR}$ seems to be beneficial.

- It is important to point out that the embedding learning method with random initialization ML-II-LE uses a single starting point $W_{\text{GP}}^{(0)}$. If we restart training multiple times (up to 1000), the best solution $\hat{W}_{GP}$ outperforms the gKDR intialized solution gKDR-LE, at the cost of a much larger training time.

- Finally, note that the embedding learning procedure returns an unstructured matrix $\hat{W}_{GP} \in \mathbb{R}^{D\times d}$, as we use (unconstrained) gradient descent optimization to maximize the marginal likelihood for (20) with respect to W_{GP}. As discussed in Section 2.3, there are various works that use an orthogonality constraint $W_{\text{GP}}^T W_{\text{GP}} = I_d$ for maximum likelihood optimization, such as the Stiefel manifold constraint [166] or the Grassmann manifold constraint [134]. We have considered these approaches, and the results (not shown) are in line with the (unstructured) embedding learning approaches presented here.

Regarding the real-world simulators, the experimental setup is identical to the quadratic function, i.e. out of a total of N points $\{x_i, y_i = f(x_i)\}_{i=1}^N$ from [63] (or from [27] in case of the hydological experiment), we have selected at random five trials, where each trial consists of five training sets of size $N_{train} = \{D, 2D, 3D, 4D, 5D\}$, and used the remaining points $N_{test} = \{N-D, N-2D, N-3D, N-4D, N-5D\}$ for testing. We plot the resulting test RMSE, test NPLD, and first subspace angle (FSA) versus the training set size for all the dimension reduction methods considered (lower values are better); the results are averaged over the five trials. Note that the plots use 'RP' and 'DP-ARD' to denote RPA-GP and DPA-GP, respectively; the FSA plots use 'true A' to denote the gradient-based active subspace $\hat{W}_{AS}$, whereas RMSE and NPLD plots use 'true embedding low dim GP' to denote $g_{\text{GP}}(\hat{W}_{AS}^T x)$ (19). These plots are presented in Figs. 26-37.

We present our conclusions for each experiment as follows:

- (ONERA M6 aerodynamics simulator, $D = 50, d = 1$)

For this experiment, we see that the sufficient dimension reduction methods SIR and gKDR worked very well, performing better than the unsupervised dimension reduction methods and the high-dimensional GP baseline. Also, we note that SIR outperformed gKDR on all benchmarks (RMSE, NPLD, FSA) for $N_{train} \geq 100$, whereas gKDR performed better in the small data regime ($N_{train} = 50$). Maximum likelihood embedding learning with random initialization (ML-II-LE) provided stable performance and was the best performing dimension reduction method; gKDR initialization (gKDR-LE) did not bring any additional improvements. In particular, we note that for $N_{train} \geq 100$, ML-II-LE managed to outperform the gradient-based $g_{\mathrm{GP}}(\hat{W}_{AS}^T x)$.

- (HIV long-term model, $D = 27$, $d = 1$)

  As in the previous ONERA M6 simulator example, we see that the sufficient dimension reduction methods SIR and gKDR worked very well, performing better than the unsupervised dimension reduction methods and on par with the high-dimensional GP baseline. We also note that SIR slightly outperformed gKDR on all benchmarks (RMSE, NPLD, FSA) for $N_{train} \geq 90$, whereas gKDR performed better in the small data regime ($N_{train} = 30$). Randomly Projected Additive GPs (42) improve the high-dimensional baseline in terms of predictive uncertainty (NPLD), but falls short in terms of predictive accuracy (RMSE). Finally, we note that gKDR initialization stabilizes training for maximum likelihood embedding learning, as gKDR-LE achieves best performance, except for the gradient-based $g_{\mathrm{GP}}(\hat{W}_{AS}^T x)$, on all benchmarks for $N_{train} \geq 90$. In particular, we notice the significant improvement in performance with respect to the randomly initialized training routine ML-II-LE for $N_{train} < 5D$.

- (Elliptic-PDE, $D = 100$, $d = 1$)

  Firstly, we note that the high-dimensional GP baseline performed on par with PCA and the random embedding methods JL, CS. Secondly, we see that the sufficient dimension reduction method SIR worked very well, performing much better than the high-dimensional GP baseline. This was also noticed in the experiment performed in [103]. Maximum likelihood embedding learning (ML-II-LE) provided stable performance with random initialization and was the best performing dimension reduction method for $N_{train} \geq 200$; SIR initialization (SIR-LE) did not bring any additional improvements. In particular, we note that for $N_{train} \geq 200$, ML-II-LE managed to perform close to the gradient-based $g_{\mathrm{GP}}(\hat{W}_{AS}^T x)$. Finally, the Randomly Projected Additive GPs (RP, (42)) performed on par with the high-dimensional baseline, whereas the Diverse Projected version (DPA-GP, (44)) provided a significant improvement in performance for this example, and was the best dimension reduction method in the small data regime $N_{train} = 100$.

- (Hydrological flow model, $D = 100$, $d = 1$)

We did not test the Randomly Projected Additive GPs (RP, (42)) and the Diverse Projected version (DPA-GP, (44)) on this example due to the slow training time. Furthermore, in terms of unsupervised dimension reduction, we only present results for the random embedding Johnson-Lindenstrauss (JL), as Count Sketch (CS) and Principal Component Analysis (PCA) provided similar performance. Some conclusions are somewhat similar to the Elliptic PDE experiment considered above: the high-dimensional GP baseline performed on par with the random embedding JL, and the sufficient dimension reduction method SIR worked very well, performing much better than the high-dimensional GP baseline. However, maximum likelihood embedding learning with random initialization (ML-II-LE) provided unstable performance among different training trials, and we see that SIR performed better on average over all three benchmarks (RMSE, NPLD, FSA). SIR initialization (SIR-LE) improved some training runs, but on average SIR-LE was still inferior to SIR. We note that by restarting the unsuccessful training runs several times, we were able to stabilize ML-II-LE predictive performance, although without a significant improvement over standard SIR.

- Overall, the same conclusions from the quadratic function experiments hold here regarding multiple restarts (i.e. the best solution $\hat{W}_{GP}$ outperforms the gKDR intialized solution gKDR-LE, at the cost of a much larger training time), as well as the conclusions regrading orthogonally-constrained maximum likelihood optimization alternatives [166, 134] (i.e. the results (not shown) are in line with the (unstructured) embedding learning approaches presented here).

- We finish with a comment on the performance of PCA for all real-world simulators. Firstly, for the ONERA M6 simulator and the HIV long-term model, we suspect that the training inputs are uniformly distributed $x_i \sim \mathcal{U}_D([-1,1]^D)$, which would explain the poor performance of PCA. For the Elliptic PDE simulator, we saw in Section 2.4 and in [103] that PCA performs very well for this experiment. However, since we use the datasets from [63] that only provide $\{x_i, f(x_i)\}_{i=1}^N$, in the results presented here we suspect that we perform PCA for the coefficients $x_i \sim \mathcal{N}_D(0, I_D)$ instead of the corresponding PDE coefficients $\log a_{x_i}$ from (21). The same is true for the hydrological flow model, where the dataset is provided by [27]; nonetheless, this experiment uses a Gaussian random field with short lengthscale for the PDE coefficients. Recall from Section 2.4 that [103] outlined the poor performance of PCA in this setting, although for a different simulator.

- Another important point of discussion that was omitted from our experiments is the case $N < D$; this case is common for simulators which run on supercomputers. Nonetheless, the case $N = D$ (presented in all the experiments) helps us draw conclusions about the (worse) predictive performance for smaller training sets $N < D$. Unfortunately, all the supervised dimension reduction methods (i.e., ML-II-LE, SIR, and gKDR) return a poor performance for $N = D$ (and thus for $N < D$) for all experiments. The unsupervised dimension reduction methods also perform poorly at $N \leq D$ for all experiments, with the caveat that PCA can perform better under more suitable prior distributions (as discussed above); the high-dimensional GP baseline NO-DR also performs poorly. The only method that approximates well $f(x)$ in all the experiments for $N = D$ is AS (with the caveat that it requires access to gradient evaluations $\nabla f(x)$, which might be unavailable for simulators running on supercomputers); in some further tests with $N < D$, we have noticed that AS still performs very well. Finally, it is worth mentioning that the dimensionality in our experiments is not very high (since $D <= 100$ for all the experiments), and so in further work we will consider simulators with much higher dimensionality (e.g., $D \geq 10^4$). It is interesting to see if those further tests will also require $N \geq 2D$, or whether a smaller, but significant (e.g. $N = D/10$) number of training points would suffice; in any case, it is important to reduce the dimensionality of the problem as much as possible prior to performing GP emulation (e.g., for a PDE simulator, instead of using the high-dimensional discretization $\log a_x(s) : \mathbb{R}^D \to \mathbb{R}$ of the PDE coefficients as inputs for your emulator, use instead the lower-dimensional parametrization x arising from a Karhunen-Loève decomposition of $\log a_x(s)$, as in Section 2.4).

3.3.1 *Conclusion*

Our take home message is that supervised dimension reduction methods are beneficial when we have enough data. Furthermore, maximum likelihood embedding learning seems to be the best performing gradient-free method in the medium-large training data regime, with the caveat that it can actually be the worst performing method in the very small data regime. Our proposed sufficient dimension reduction initialization for embedding

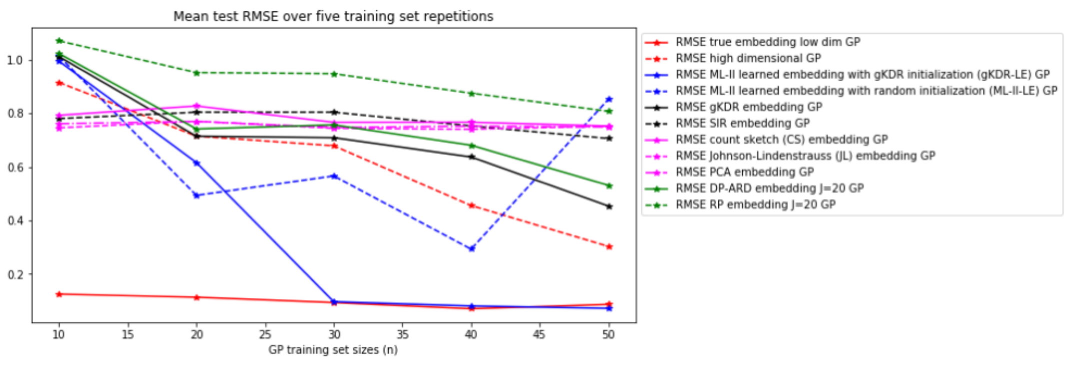


Figure 2: (Best viewed in colour) RMSE for quadratic function (46) with $D = 10, d = 1$

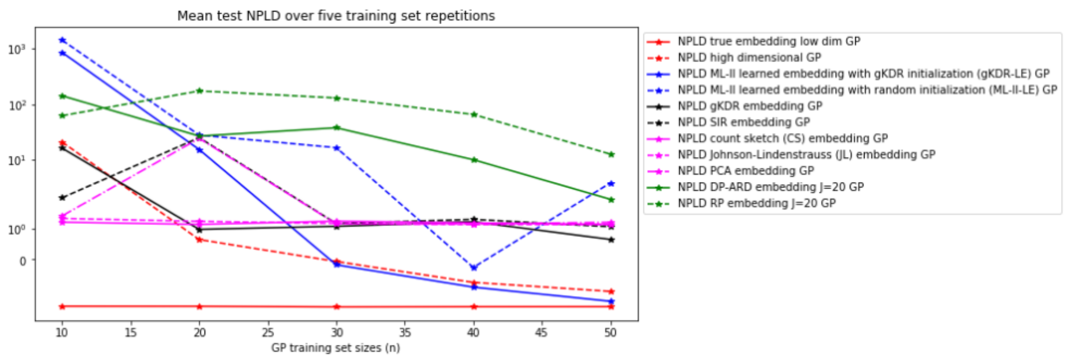


Figure 3: (Best viewed in colour) NPLD for quadratic function (46) with $D = 10, d = 1$

learning proved beneficial in many training instances, although it was outperformed by (a potentially large number of) multiple restarts with random initialization. Overall, the gradient-based active subspace is the best performing method, although gradients of the simulator are often unavailable in practical applications such as climate models [81]. For future work, we would like to investigate different prior distributions $p(x)$ for the inputs $\{x_i\}_{i=1}^N \sim p(x)$, so that the unsupervised dimension reduction methods may exhibit a more diverse and (potentially) better performance. Also, we will consider simulators with high-dimensional outputs, see e.g. [138, 190].

Following the observation that many computer simulators used in Uncertainty Quantification admit a low-dimensional linear structure, various dimension reduction methods have been proposed. We provide a review of

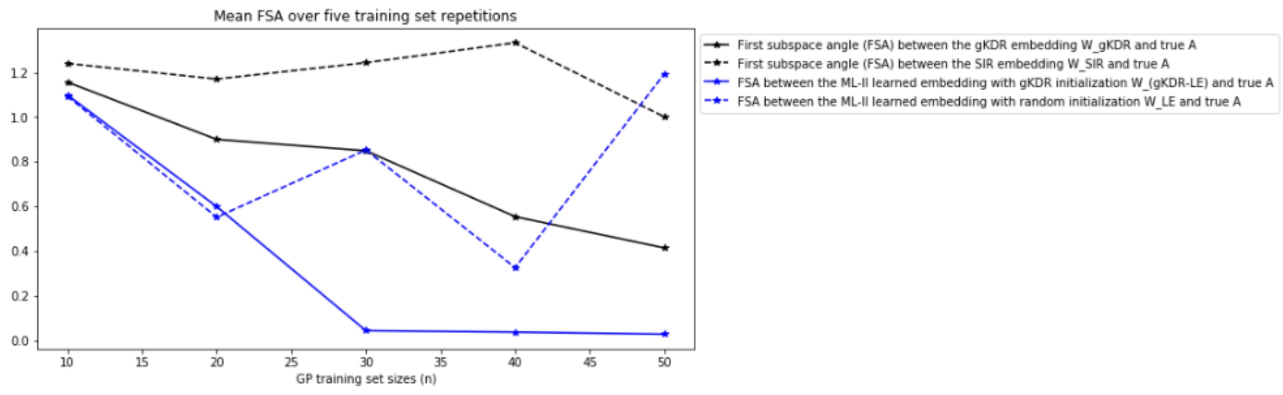


Figure 4: (Best viewed in colour) FSA for quadratic function (46) with $D = 10, d = 1$

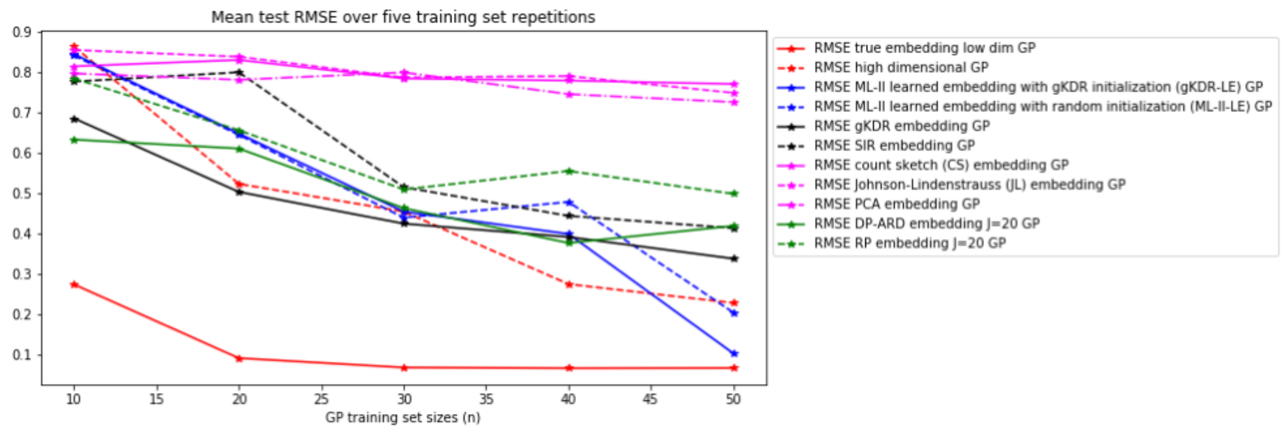


Figure 5: (Best viewed in colour) RMSE for quadratic function (46) with $D = 10, d = 2$

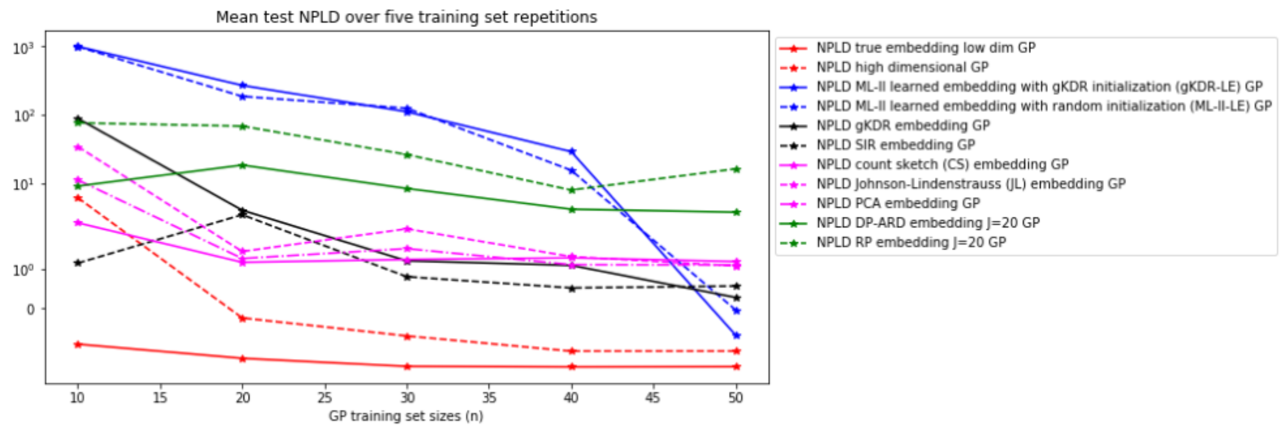


Figure 6: (Best viewed in colour) NPLD for quadratic function (46) with $D = 10, d = 2$

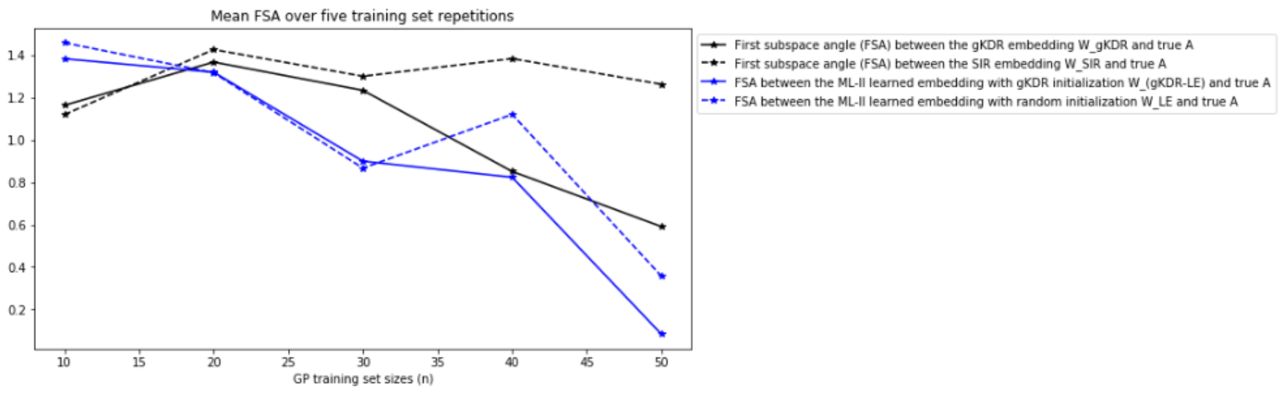


Figure 7: (Best viewed in colour) FSA for quadratic function (46) with $D = 10, d = 2$

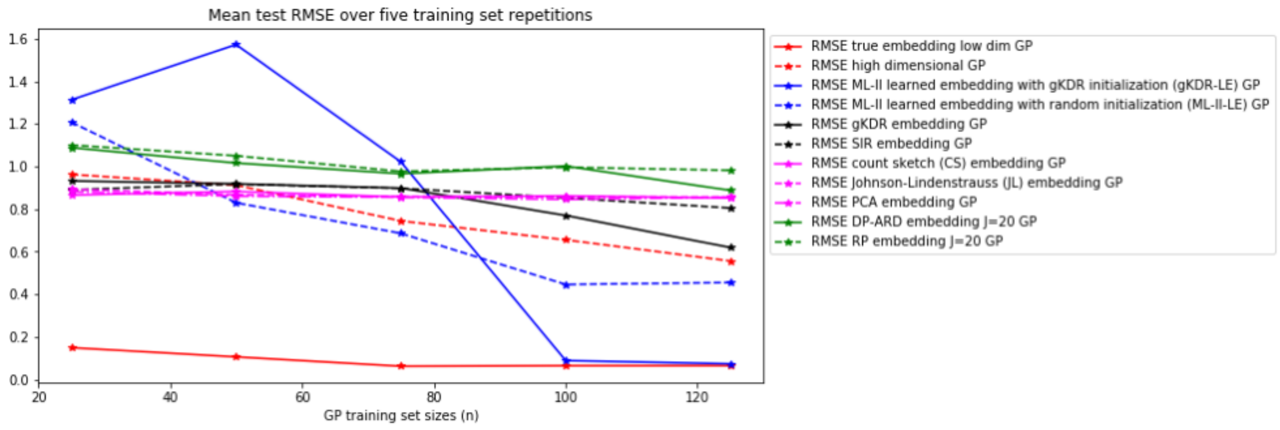


Figure 8: (Best viewed in colour) RMSE for quadratic function (46) with $D = 25, d = 1$

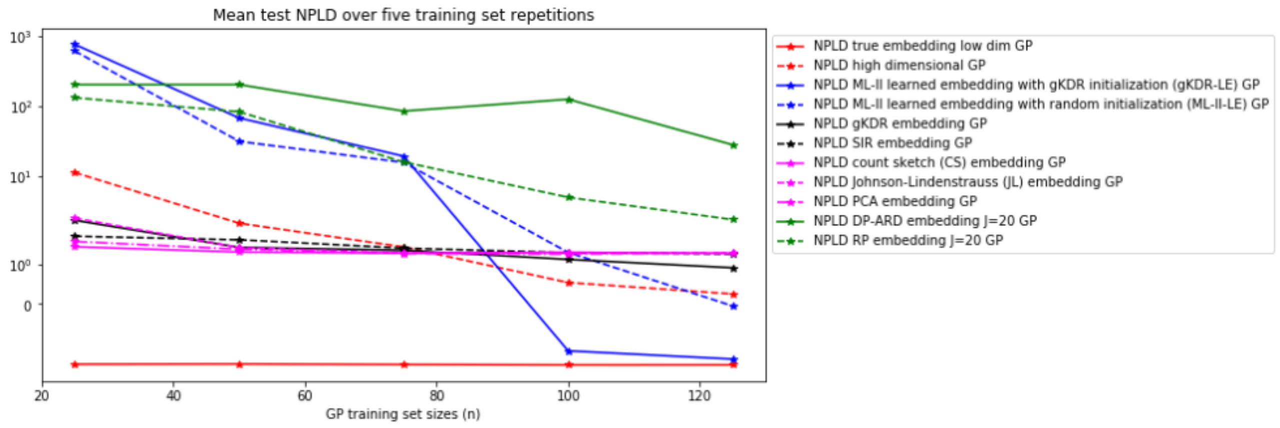


Figure 9: (Best viewed in colour) NPLD for quadratic function (46) with $D = 25, d = 1$

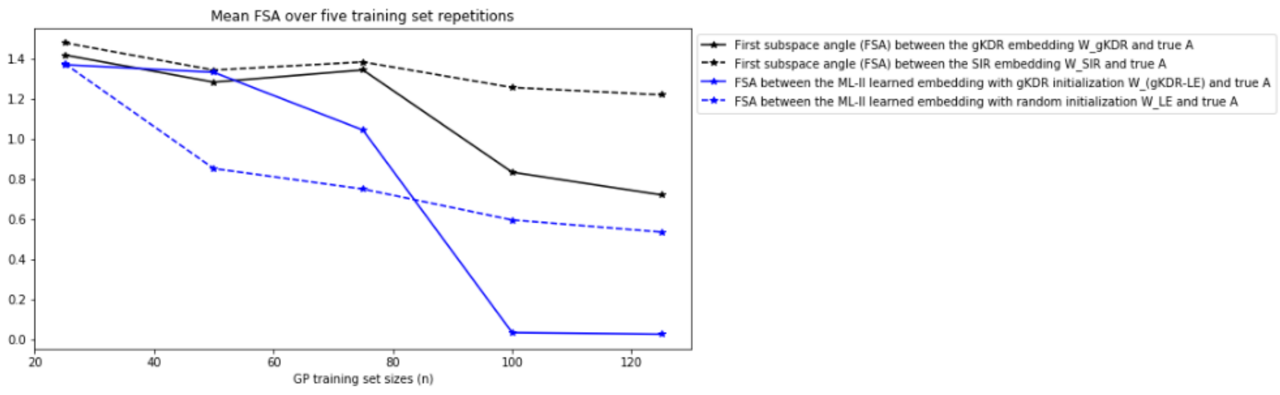


Figure 10: (Best viewed in colour) FSA for quadratic function (46) with $D = 25, d = 1$

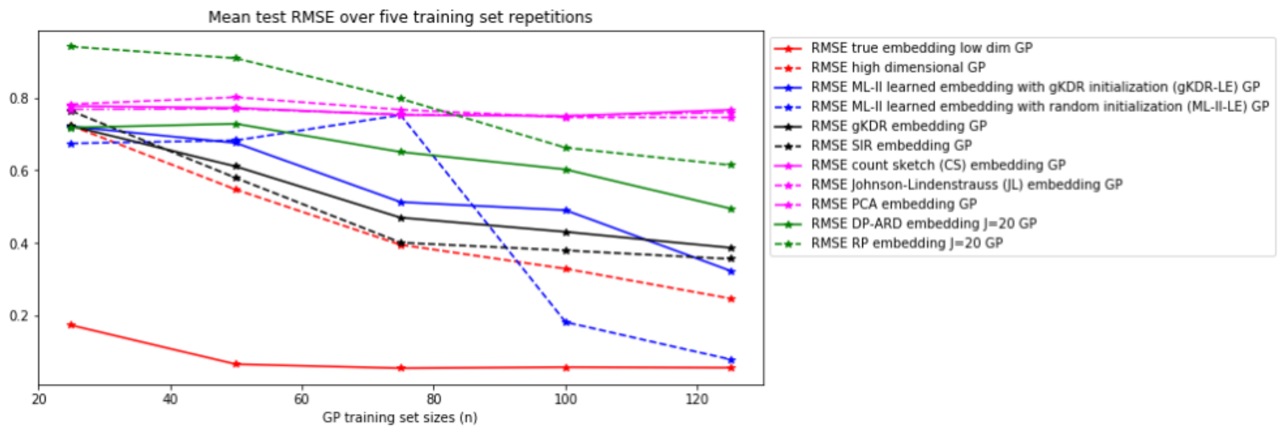


Figure 11: (Best viewed in colour) RMSE for quadratic function (46) with $D = 25, d = 2$

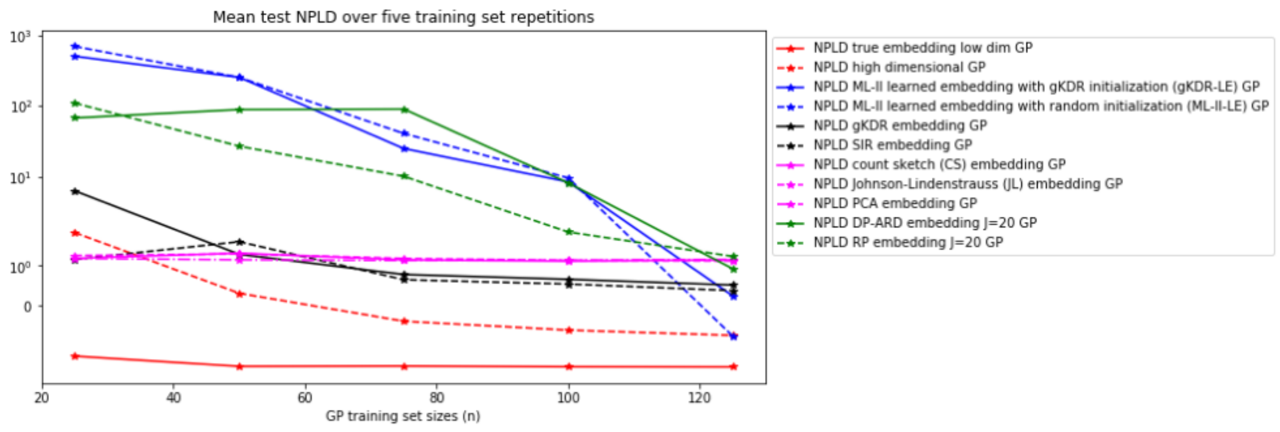


Figure 12: (Best viewed in colour) NPLD for quadratic function (46) with $D = 25, d = 2$

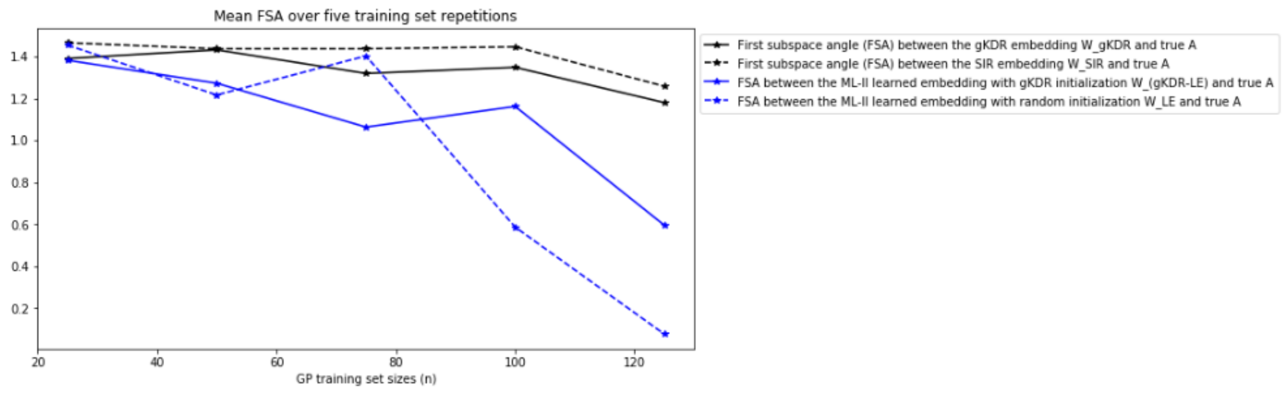


Figure 13: (Best viewed in colour) FSA for quadratic function (46) with $D = 25, d = 2$

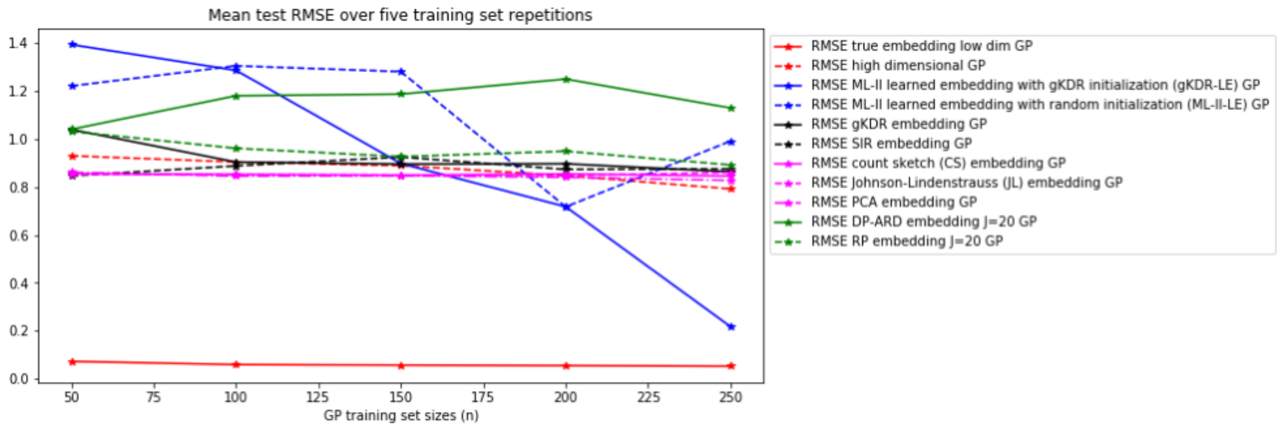


Figure 14: (Best viewed in colour) RMSE for quadratic function (46) with $D = 50, d = 1$

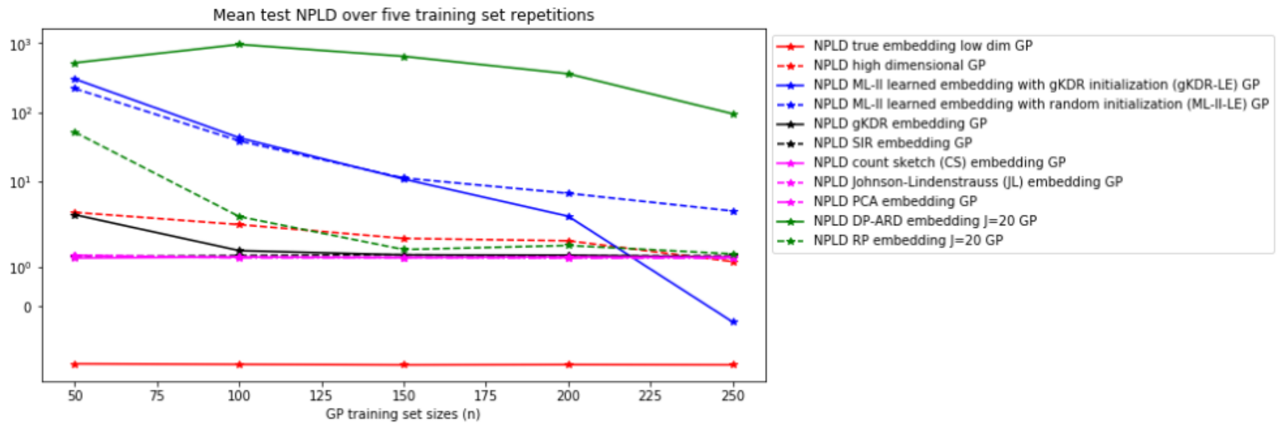


Figure 15: (Best viewed in colour) NPLD for quadratic function (46) with $D = 50, d = 1$

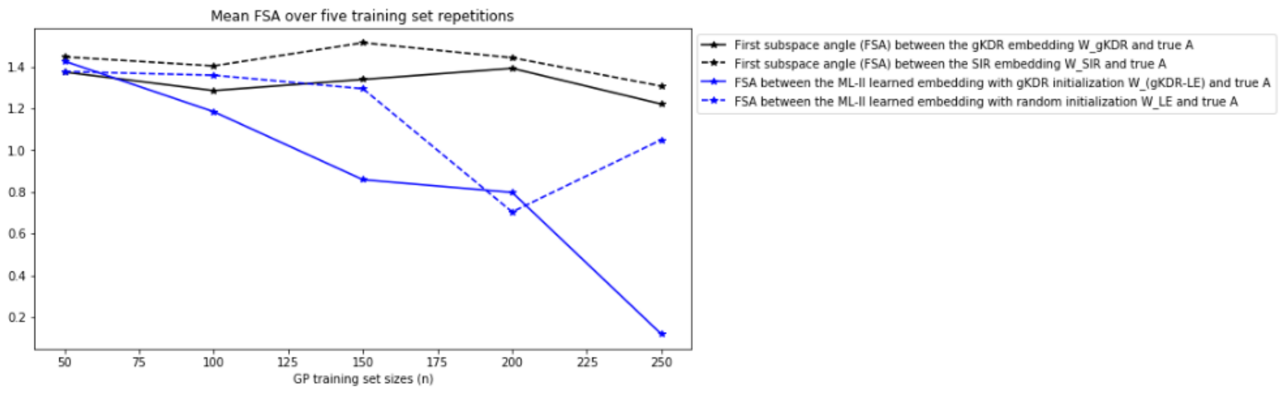


Figure 16: (Best viewed in colour) FSA for quadratic function (46) with $D = 50, d = 1$

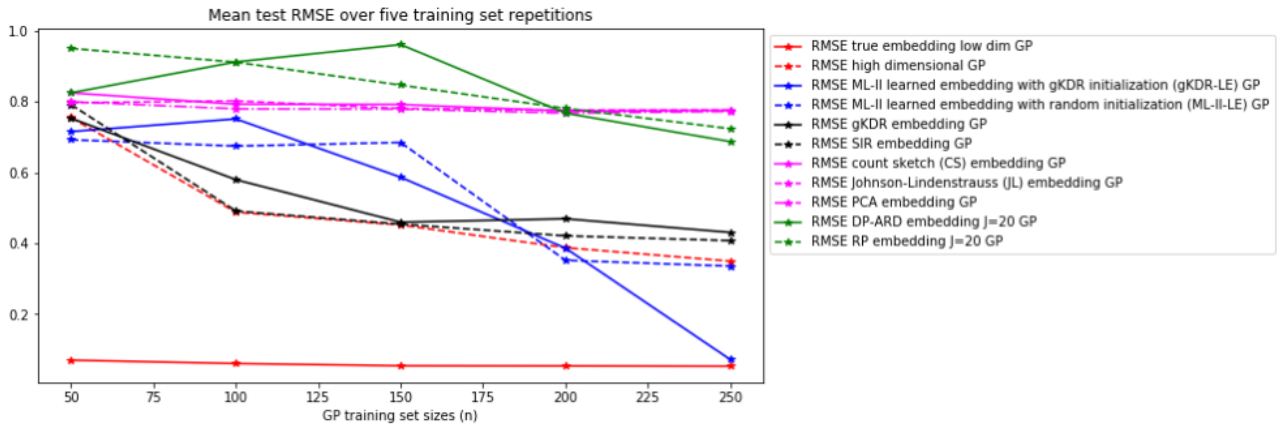


Figure 17: (Best viewed in colour) RMSE for quadratic function (46) with $D = 50, d = 2$

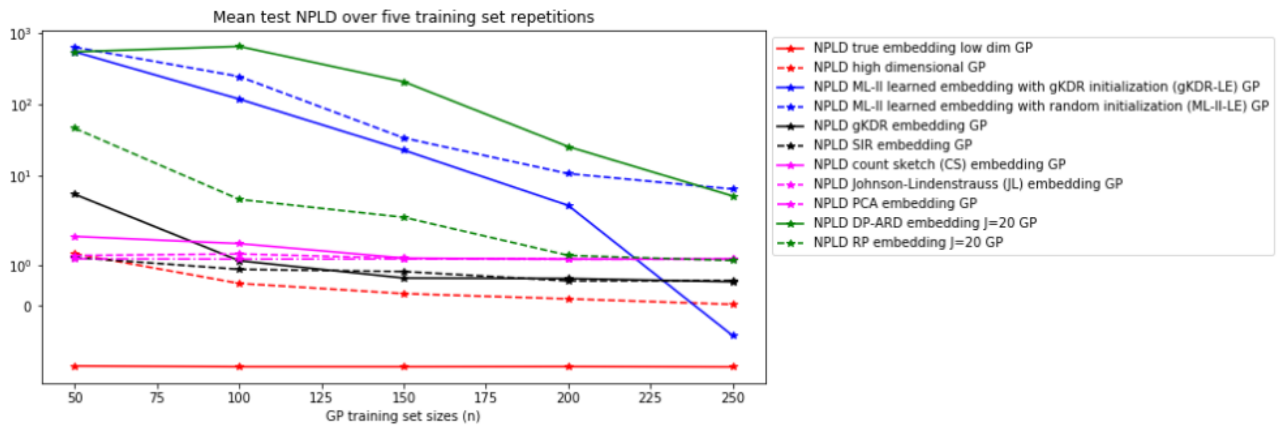


Figure 18: (Best viewed in colour) NPLD for quadratic function (46) with $D = 50, d = 2$

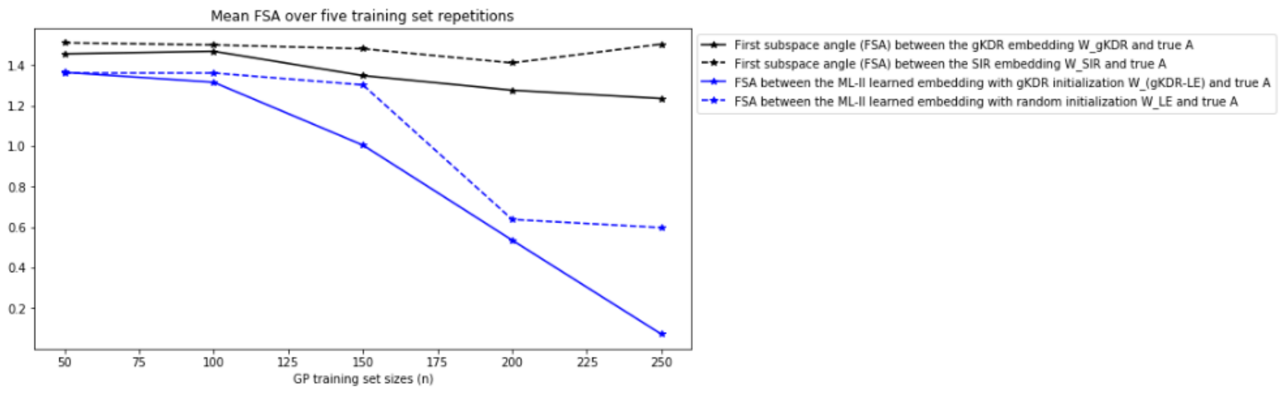


Figure 19: (Best viewed in colour) FSA for quadratic function (46) with $D = 50, d = 2$

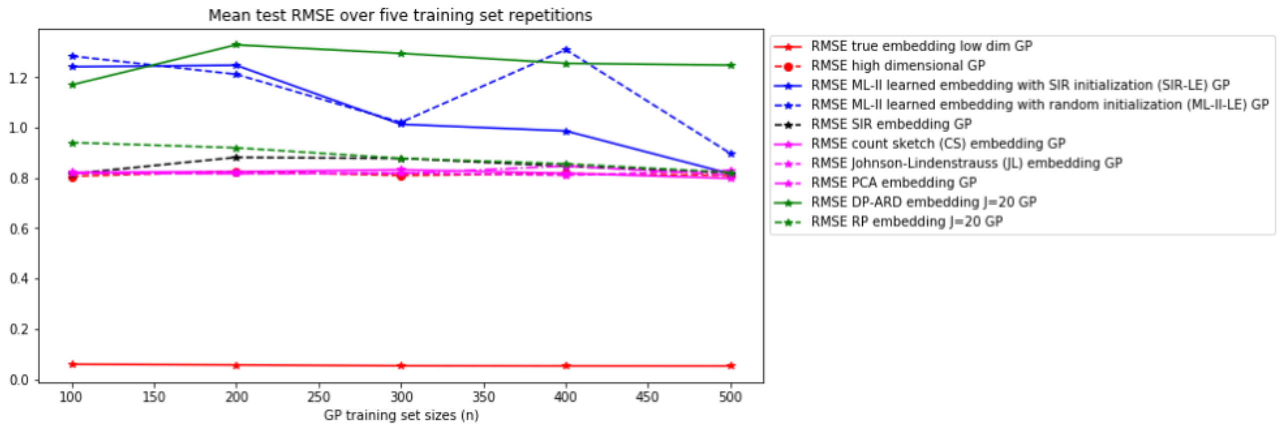


Figure 20: (Best viewed in colour) RMSE for quadratic function (46) with $D = 100, d = 1$

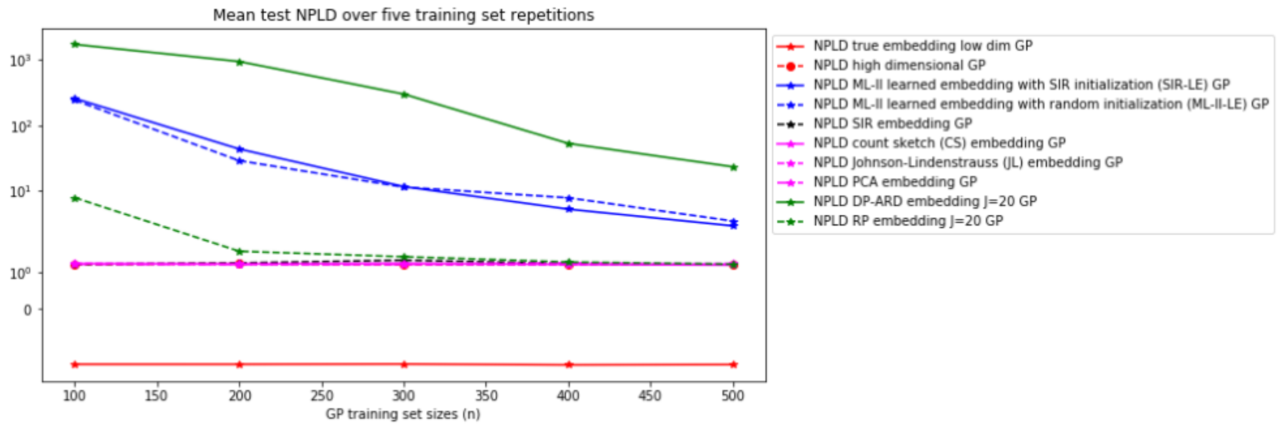


Figure 21: (Best viewed in colour) NPLD for quadratic function (46) with $D = 100, d = 1$

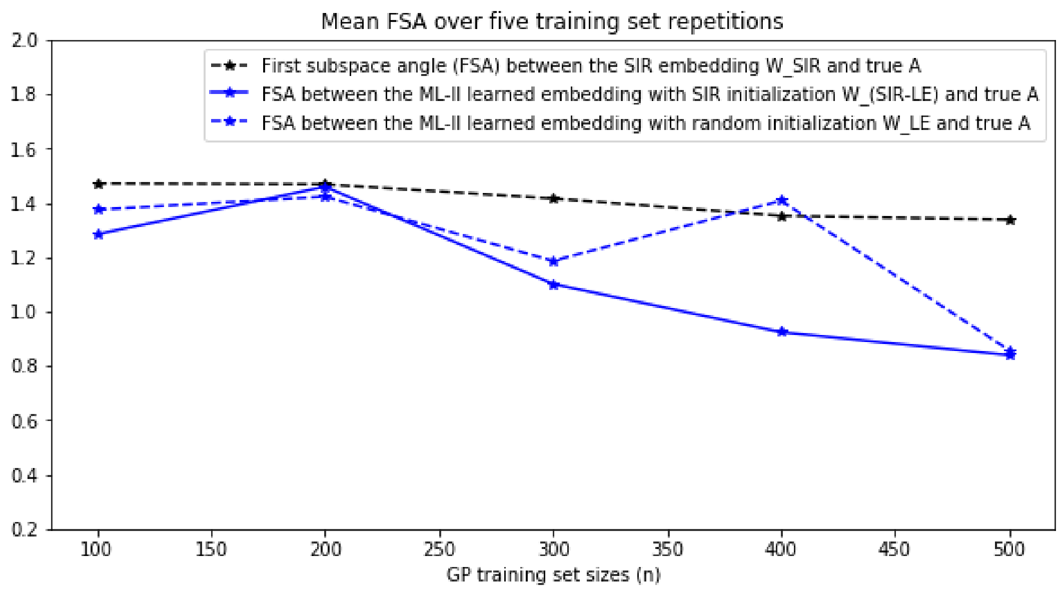


Figure 22: (Best viewed in colour) FSA for quadratic function (46) with $D = 100, d = 1$

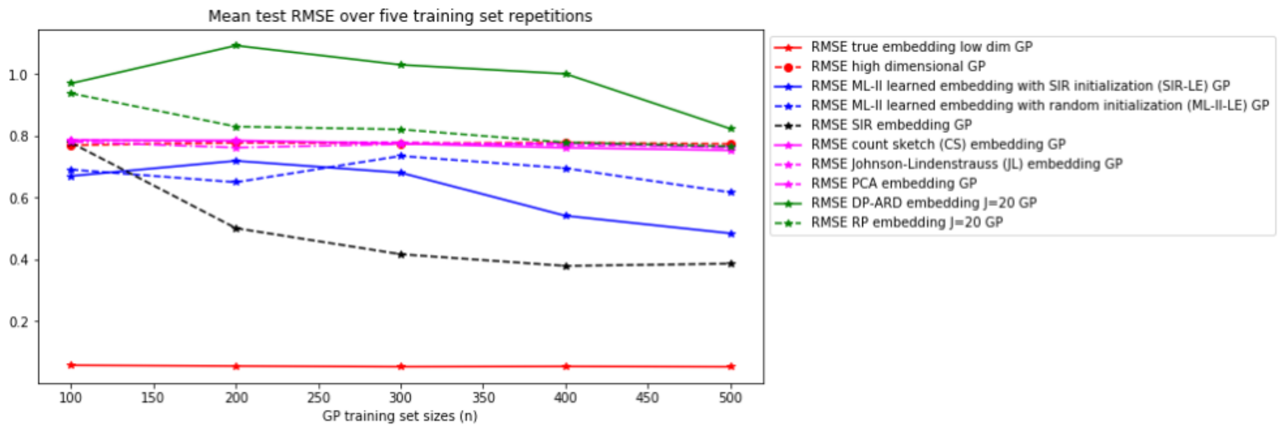


Figure 23: (Best viewed in colour) RMSE for quadratic function (46) with $D = 100, d = 2$

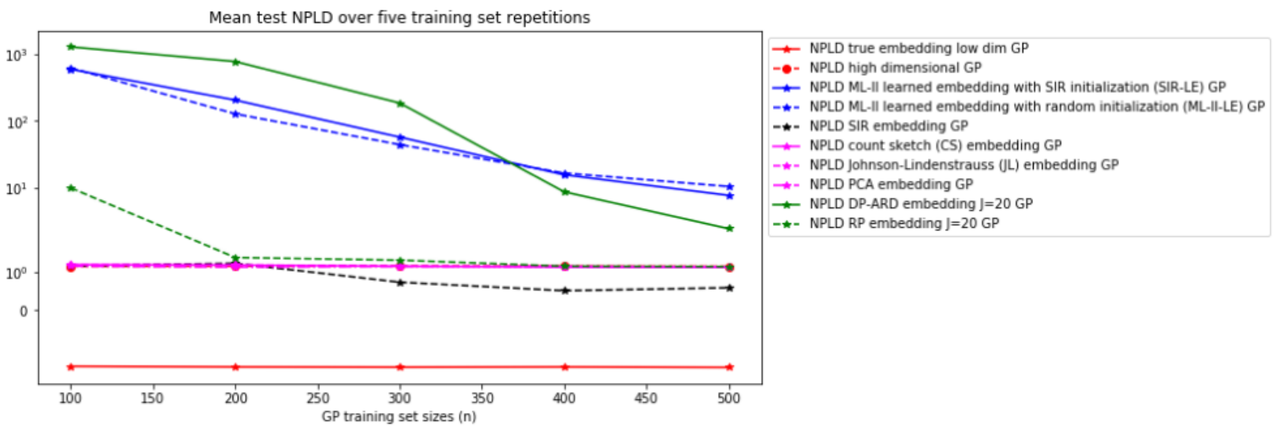


Figure 24: (Best viewed in colour) NPLD for quadratic function (46) with $D = 100, d = 2$

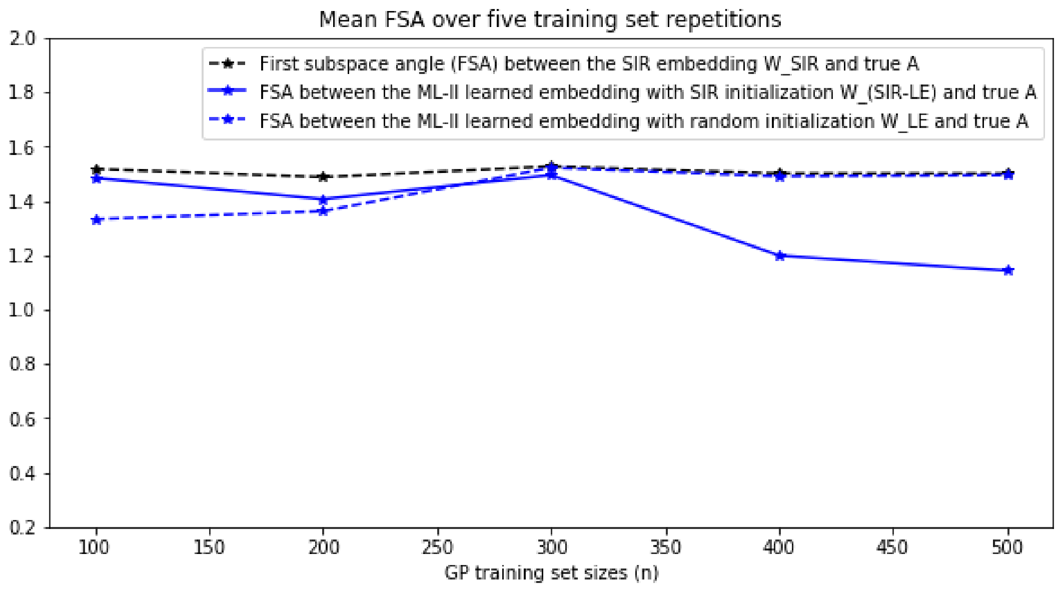


Figure 25: (Best viewed in colour) FSA for quadratic function (46) with $D = 100, d = 2$

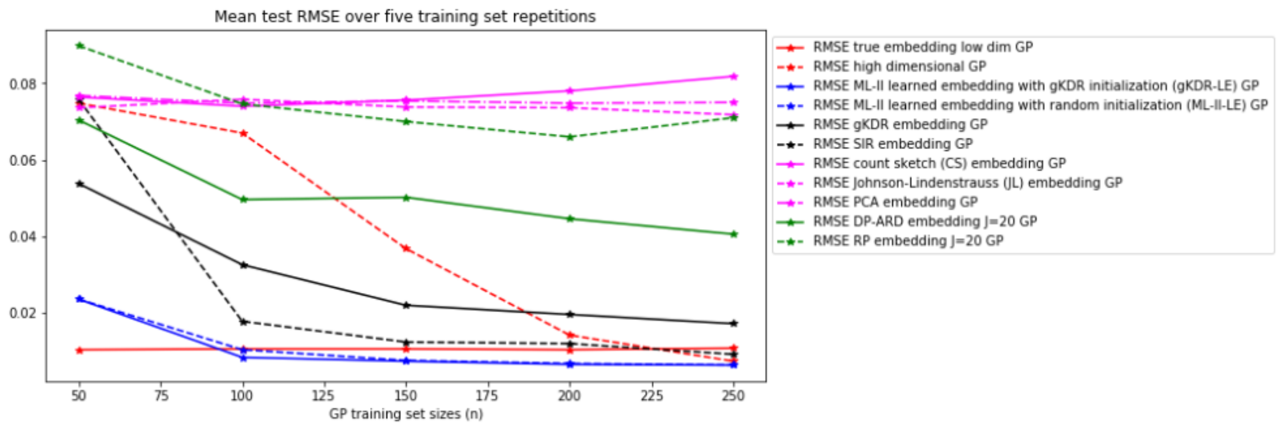


Figure 26: (Best viewed in colour) RMSE for the ONERA M6 simulator with $D = 50, d = 1$

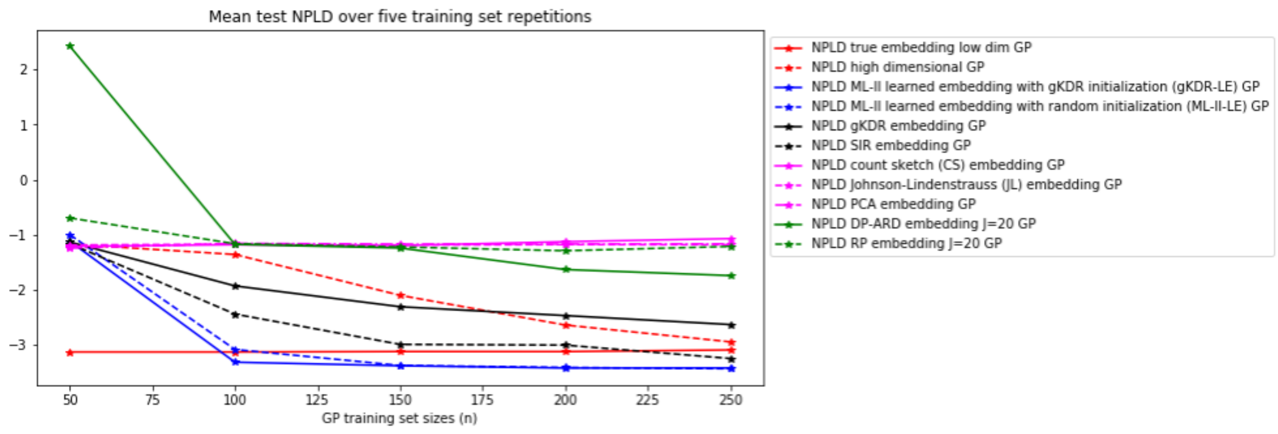


Figure 27: (Best viewed in colour) NPLD for the ONERA M6 simulator with $D = 50, d = 1$

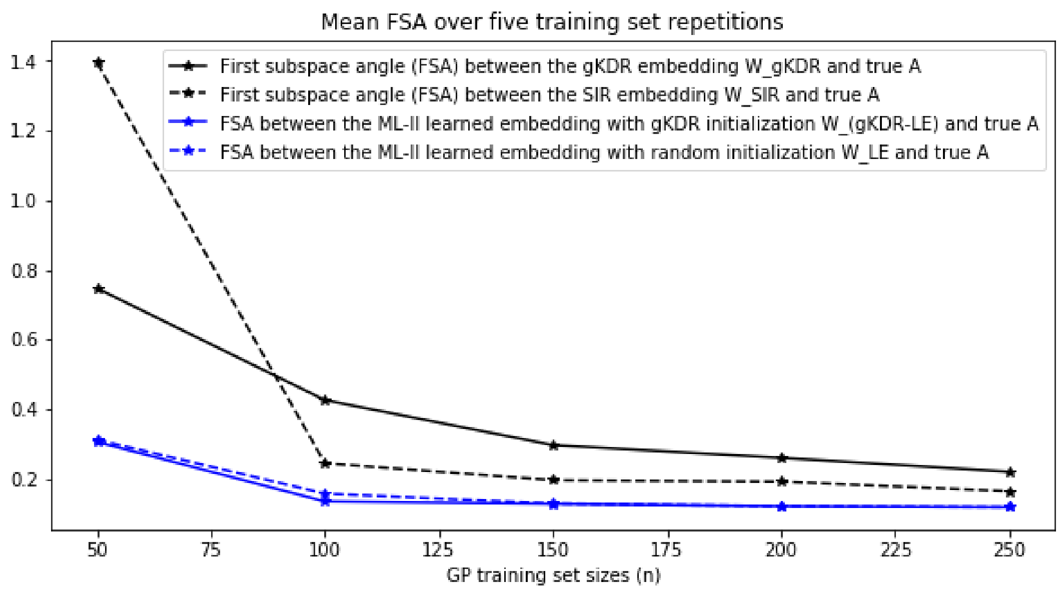


Figure 28: (Best viewed in colour) FSA for the ONERA M6 simulator with $D = 50, d = 1$

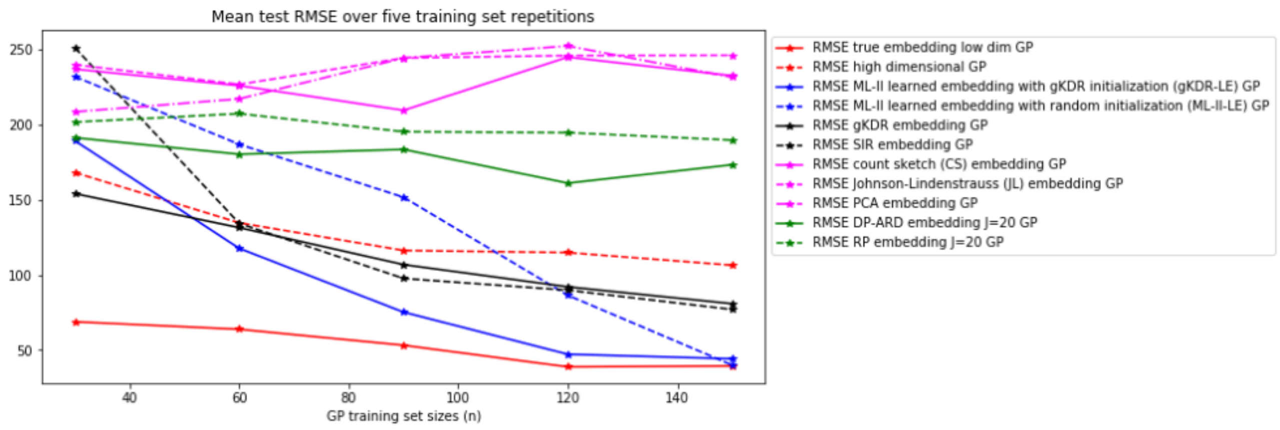


Figure 29: (Best viewed in colour) RMSE for the HIV long term model with $D = 27, d = 1$

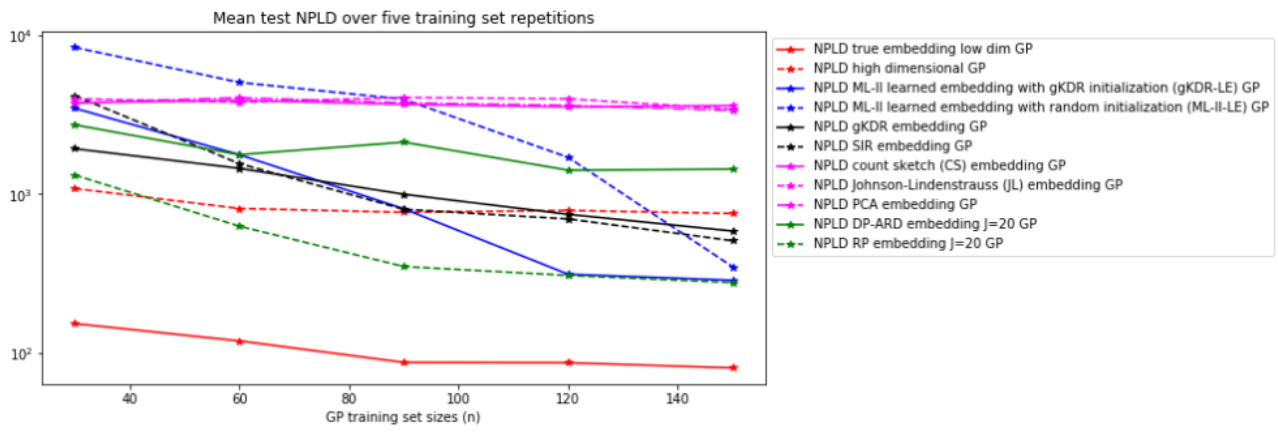


Figure 30: (Best viewed in colour) NPLD for the HIV long term model with $D = 27, d = 1$

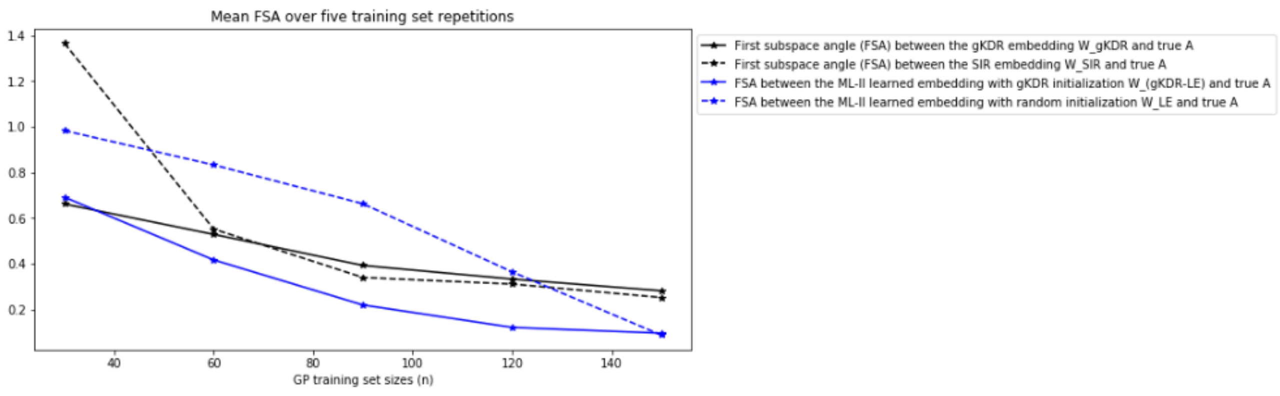


Figure 31: (Best viewed in colour) FSA for the HIV long term model with $D = 27, d = 1$

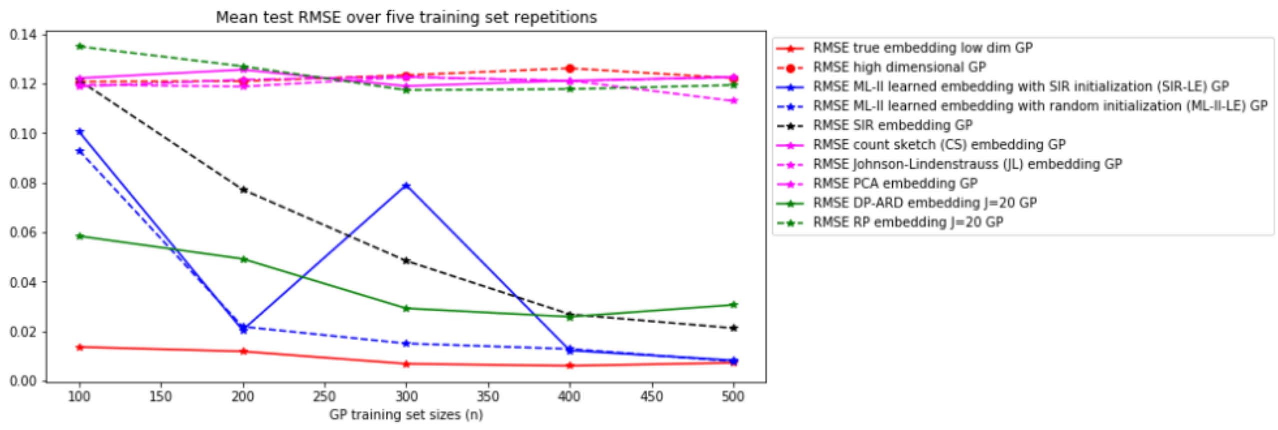


Figure 32: (Best viewed in colour) RMSE for the Elliptic PDE model with $D = 100, d = 1$

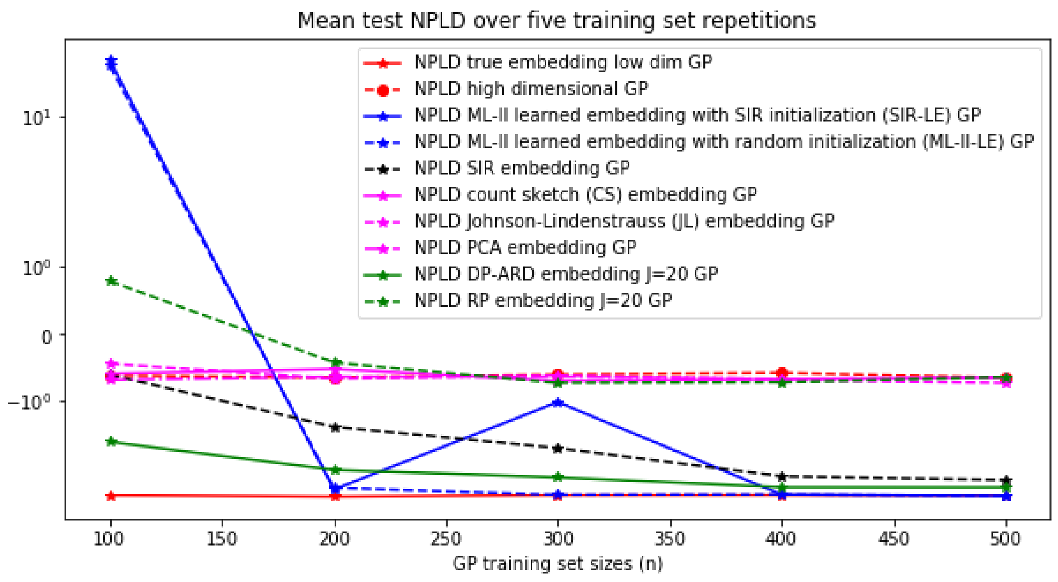


Figure 33: (Best viewed in colour) NPLD for the Elliptic PDE model with $D = 100, d = 1$

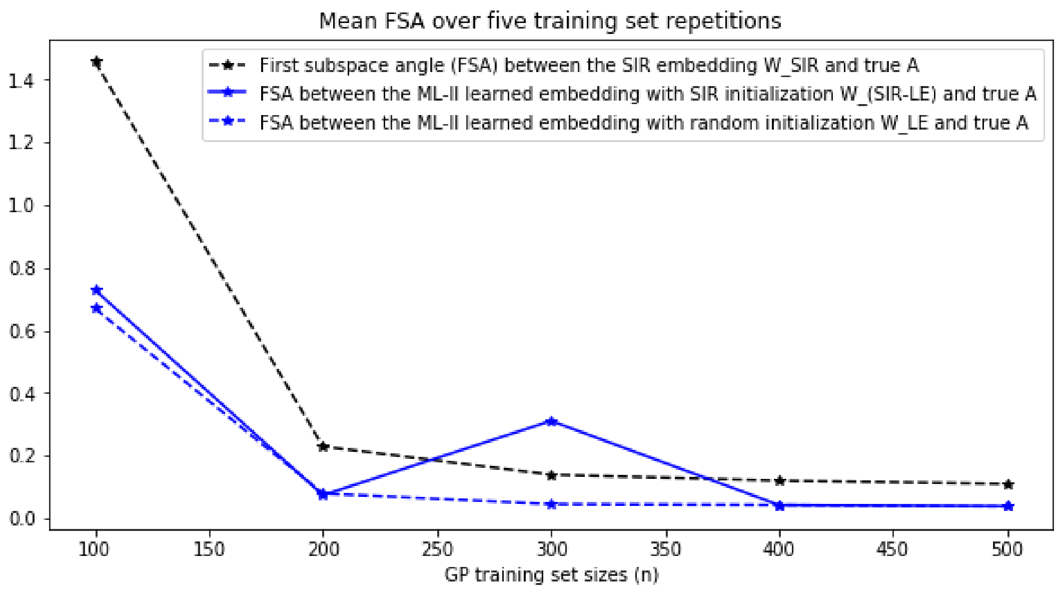


Figure 34: (Best viewed in colour) FSA for the Elliptic PDE model with $D = 100, d = 1$

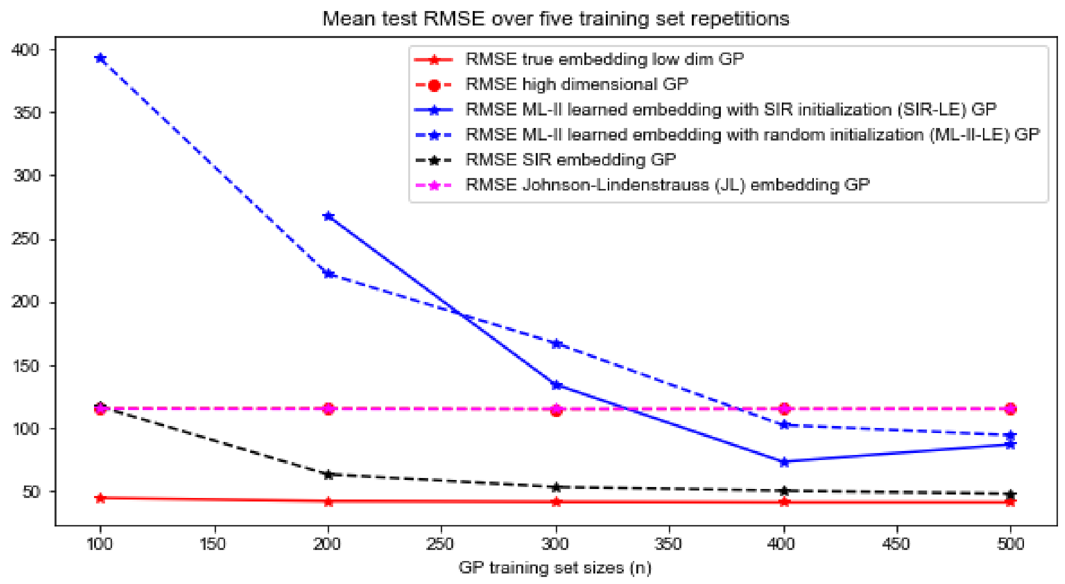


Figure 35: (Best viewed in colour) RMSE for the hydrological flow model with $D = 100, d = 1$

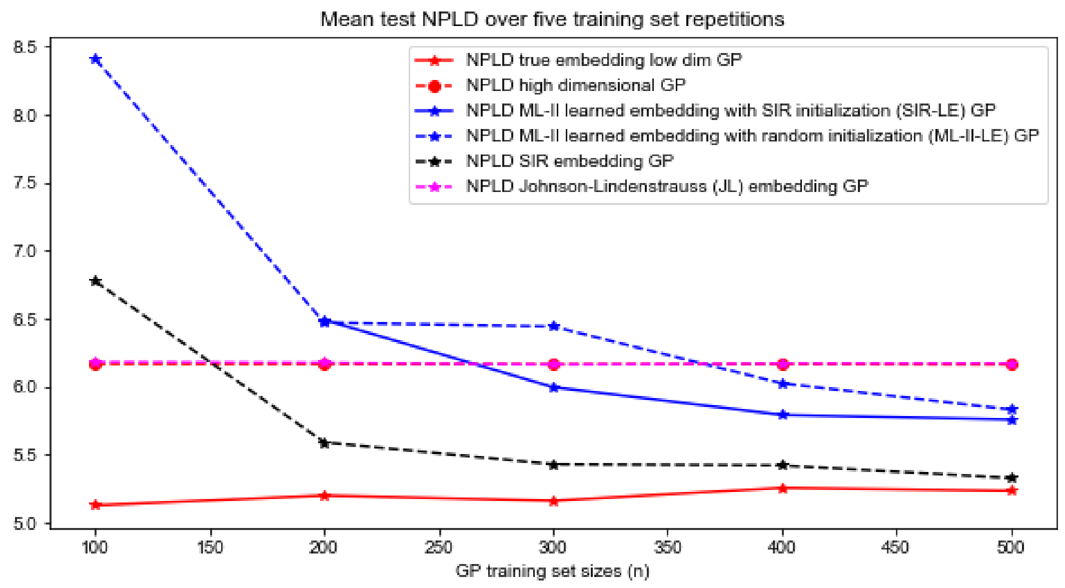


Figure 36: (Best viewed in colour) NPLD for the hydrological flow model with $D = 100, d = 1$

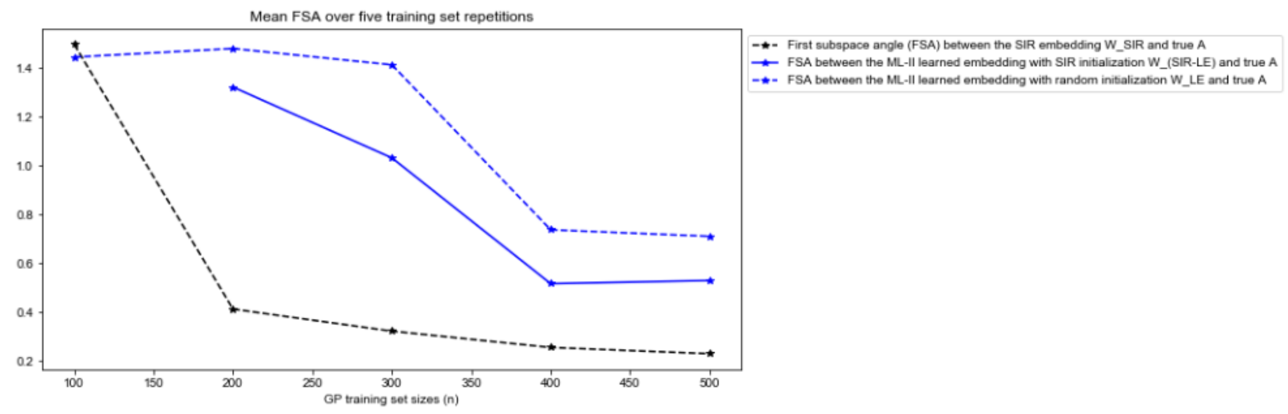


Figure 37: (Best viewed in colour) FSA for the hydrological flow model with $D = 100, d = 1$

these methods in the context of Gaussian Process (GP) emulation; among the methods considered are Sufficient Dimension Reduction (SDR), Principal Component Analysis (PCA), Active Subspaces (AS), random embeddings, and embedding learning. While we provide an introduction and some theoretical properties for each family of methods, our focus is to perform a method comparison between all these approaches. We complement the existing results with our new findings. For various synthetic and real-world simulators, we show that the most straightforward embedding learning methods (i.e., without orthogonality constraints) tend to outperform the SDR methods for GP emulation.

4 GP EMULATION FOR LOG-LIKELIHOODS WITH A LOW-DIMENSIONAL ACTIVE SUBSPACE

We will introduce the setting of Bayesian Inverse Problems, in particular via the emulation based solution of replacing the underlying simulator (e.g. physics-based model) or the log-likelihood with a GP emulator. We provide a new approach of exploiting the low-dimensionality of the log-likelihood in some of these problems, and present very promising results on a quadratic toy model, as well as on a more challenging high-dimensional Elliptic PDE.

4.1 INTRODUCTION

As dicussed in Section 1.2, we consider a Bayesian inverse problem, where the goal is to obtain the posterior distribution

$$p(x|\mathcal{D}) = \frac{p(\mathcal{D}|x)p(x)}{p(\mathcal{D})}$$

over the unknown parameters $x \in \mathbb{R}^D$ given the observed data $\mathcal{D} \in \mathbb{R}^m$.

Typically, the likelihood follows a Gaussian distribution

$$\mathcal{D}|x \sim \mathcal{N}_m(f(x), \Sigma),$$

where $f(x) : \mathbb{R}^D \to \mathbb{R}^m$ is the simulator function, usually modelling an underlying physical process through a partial differential equation (PDE). The covariance matrix Σ contains the modelling and observational errors and is typically diagonal, as the errors are assumed to be uncorrelated.

This assumption is clearly violated in applications where the data $\mathcal{D}$ is a collection of (potentiallly spatially-correlated) time series, e.g., subsurface measurments from a ground-penetrating radar [67]. Nonetheless, due to extremely tight budget under which some inverse problems communities operate (e.g., limited to at most $\approx 10^3 - 10^4$ simulator evaluations for a wind-turbine application [192]), the estimation of the covariance matrix Σ is deliberately ignored, as we do in this work (we follow the rule-of-thumb $\Sigma_{ii} := (\rho/100)^2|\mathcal{D}_i|^2$ for $i \in [m]$, i.e., the estimated standard deviation for the errors is $\rho-$percent of the observed data $|\mathcal{D}_i|$, where ρ is usually selected by trial-and-error, e.g., $\rho \in [0.1, 5]$). In our setting where a GP emulator is used for $f(x)$, a more principled approach would be to assign a GP prior to both the modelling errors and the observations errors (see equation (S3) and (S2), respectively, in [138]); note that the History Matching (HM) methods for inverse problems relax the independence assumption between modelling and observational errors to merely uncorrelation [138]. Furthrmore, Approximate Bayesian Computation (ABC) methods can also be potentially used to estimate Σ (e.g., Algorithm 2 from [167]).

As discussed so far in the thesis, the simulator function f is quite often expensive to evaluate, and we only have access to a limited budget of N simulations $\{x_i, f(x_i)\}_{i=1}^N$, or equivalently N log-likelihood evaluations $\{x_i, \log p(\mathcal{D}|x_i)\}_{i=1}^N$, where $p(\mathcal{D}|x_i) := \mathcal{N}_m(\mathcal{D}|f(x_i), \Sigma)$ is the Gaussian density with mean $f(x_i)$ and covariance Σ, evaluated at $\mathcal{D}$. Furthermore, there might be no gradient information available for the simulator.

One standard approach in this case is to build a GP emulator $g_{\text{GP}}(x)$ (3) for $f(x)$ or $\log p(\mathcal{D}|x)$, using an i.i.d. sample of size N from the prior distribution $\{x_i\}_{i=1}^N \sim p(x)$, along with the corresponding training sets $\{x_i, f(x_i)\}_{i=1}^N$ [15] or $\{x_i, \log p(\mathcal{D}|x_i)\}_{i=1}^N$ [127]. The former work [15] shows a petroleum engineering application, whereas the latter work [127] presents an epidemiological application. With a large enough N, both approaches lead to a global emulator over the whole prior support $p(x)$.

GP emulators are known to deal very well with small training budgets N, but may struggle with high input dimensionality D, as discussed in Section 1.5. Here, we adopt the key assumption from [31], where the log-likelihood

$$\log p(\mathcal{D}|x) \propto L(x) := -||\mathcal{D} - f(x)||^2_{\Sigma_{\text{obs}}} \tag{48}$$

has a low-dimensional active subspace. As in Section 2.3, we can write $L(x)$ as $L(x) \approx \hat{g}(\hat{W}_{AS}^T x)$ (17), where $\hat{g} : \mathbb{R}^d \to \mathbb{R}$ with $d \ll D$, and $\hat{W}_{AS} \in \mathbb{R}^{D \times d}$ is the matrix whose columns are the d dominant eigenvectors of $\hat{W}$ (16).

4.2 LEARNED EMBEDDING ACTIVE SUBSPACE MCMC (LE-AS-MCMC)

The seminal work [31] introduces a methodology for exploiting this active subspace structure (without emulators) in order to efficiently solve the Bayesian inverse problem. In other words, using a simulator-based low-dimensional approximation $\hat{g}(\hat{W}_{AS}^T x)$, a much smaller number of simulator evaluations were needed to perform Bayesian inversion, compared with using the high-dimensional log-likelihood $L(x)$ directly. Note that by 'simulator-based', we mean that every evaluation $\hat{g}(\hat{W}_{AS}^T x)$ requires (at least) one simulator query $f(x)$. This methodology was applied in a groundwater modelling application [126], and was extended by replacing the simulator-based approximation $\hat{g}(\hat{W}_{AS}^T x)$ with GP emulators $g_{\text{GP}}(\hat{W}_{AS}^T x)$ (19) [127]. Another extension is to replace the active subspace approximation $\hat{g}(\hat{W}_{AS}^T x)$ with an alternative (simulator-based) low-dimensional approximation $\hat{g}(\hat{W}_{LIS}^T x)$, from the family of likelihood informed subspaces $\hat{W}_{LIS} \in \mathbb{R}^{D \times d}$ [185].

In this work, our approach is similar to [127], but instead of using the gradient-based GP emulator $g_{\text{GP}}(\hat{W}_{AS}^T x)$, we will use the gradient-free approach $g_{\text{GP}}(\hat{W}_{\text{GP}}^T x)$, where $\hat{W}_{\text{GP}}$ is obtained by maximizing the marginal likelihood (5) for $g_{\text{GP}}(W_{\text{GP}}^T x)$ (20) with respect to W_{GP}. Our hope is that the resulting approximation $g_{\text{GP}}(\hat{W}_{\text{GP}}^T x) \approx L(x)$ is satisfactory for the posterior sampling procedure proposed in [31], which we will describe next.

Firstly, note that the low-dimensional approximation $\hat{g}(\hat{W}_{AS}^T x) \approx L(x)$ leads to an approximate posterior $\tilde{p}(x|\mathcal{D}) \propto \exp(-\hat{g}(\hat{W}_{AS}^T x))p(x)$. The Hellinger distance between $p(x|\mathcal{D})$ and $\tilde{p}(x|\mathcal{D})$ is defined as

$$H^2(p, \tilde{p}) := \frac{1}{2} \int_{\mathbb{R}^D} \left(\sqrt{p(x|\mathcal{D})} - \sqrt{\tilde{p}(x|\mathcal{D})} \right)^2 dx, \tag{49}$$

and can be bounded from above according to Equation (3.13) in [31], which is derived from the upper bound on the approximation $\hat{g}(\hat{W}_{AS}^T x) \approx L(x)$ (17). In this chapter, we assume that the prior is a standard multivariate Gaussian distribution, i.e. $x \sim \mathcal{N}_D(0, I_D)$. Note that any $x \in \mathbb{R}^D$ can be decomposed as $x = W_1 W_1^T x + W_2 W_2^T x$, where $W_1 := \hat{W}_{AS}$ and $W_2 \in \mathbb{R}^{D \times (D-d)}$ is the orthogonal complement of W_1 (i.e. the columns of W_1 are orthogonal to the columns of W_2, and $W_2^T W_2 = I_{D-d}$). Let us write $y := W_1^T x \in \mathbb{R}^d$ and $z := W_2^T x \in \mathbb{R}^{D-d}$. This is particularly useful, since we can factorize the prior distribution as $p(x) = p(y)p(z)$, since $y \sim \mathcal{N}_d(0, W_1^T W_1) = \mathcal{N}_d(0, I_d)$ and $z \sim \mathcal{N}_{D-d}(0, W_2^T W_2) = \mathcal{N}_{D-d}(0, I_{D-d})$. Using this factorization, we can write the approximate posterior $\tilde{p}(x|\mathcal{D})$ as

$$\tilde{p}(x|\mathcal{D}) \propto \exp(-\hat{g}(y))p(y)p(z). \tag{50}$$

The decomposition (50) is key for obtaining computational advantages compared with the high-dimensional posterior $p(x|\mathcal{D}) \propto \exp(-L(x))p(x)$. Namely, in order to obtain approximate posterior samples $x \sim \tilde{p}(x|\mathcal{D})$, it is enough to obtain samples from the low-dimensional $y \sim \tilde{p}(y|\mathcal{D}) \propto \exp(-\hat{g}(y))p(y)$ and from the prior distribution $z \sim p(z) = \mathcal{N}_d(0, I_{D-d})$, followed by $x = W_1 y + W_2 z$. Samples $z \sim p(z)$ can be obtained cheaply without simulator evaluations, and we can apply various efficient sampling methods for the low-dimensional problem $y \sim \tilde{p}(y|\mathcal{D})$. The work [31] uses the Metropolis-Hastings Markov Chain Monte Carlo (MH-MCMC) algorithm, which is arguably the most popular method and is presented in Step 1 to 5 of Algorithm 1. In order to sample from the approximate posterior distribution $\tilde{p}(y|\mathcal{D})$, MH-MCMC simulates a Markov Chain $\{y_k\}_{k=1}^{\infty}$ whose stationary distribution is $\tilde{p}(y|\mathcal{D})$. Theorem 2.1 in [113] shows that MH-MCMC is guaranteed to eventually produce approximate posterior samples $\{y_k\}_{k=n}^{\infty} \sim$

$\tilde{p}(y|\mathcal{D})$ for large enough n, as long as the ratio between the target density $\tilde{p}(y|\mathcal{D})$ and the proposal density $q(y)$ is bounded ($q(y) := \mathcal{N}_d(y, \sigma^2_{prop} I_d)$ from Step 1 of Algorithm 1 is a standard choice). Note that the variance σ^2_{prop} of the proposal density is a tuning parameter which controls the acceptance rate of the Markov Chain (Step 4 in Algorithm 1 is known as the acceptance step). As discussed in [148], a very small σ_{prop} often leads to very high acceptance rates, which is an indication that the chain moves slowly in $\mathbb{R}^d$ and the exploration of the target density is likely to be inefficient. On the other hand, a very large σ_{prop} often leads to very low acceptance rates, which suggests that exploration is again inefficient as the chain rarely moves. The computer experiments from [31] result in acceptance rates between 10% and 80%.

The whole procedure from Algorithm 1 is known as active subspace MCMC (AS-MCMC). It was shown that AS-MCMC is much more efficient in exploring the posterior landscape than performing MCMC in the high-dimensional space $\mathbb{R}^D$ for $p(x|\mathcal{D})$, as it produces uncorrelated samples in a much smaller number of iterations (see Table 5.1 in [31]) and it does not sacrifice accuracy. Indeed, in two computer experiments that will be revisited in this chapter, various tests were performed to compare the accuracy of the AS-MCMC approximate posterior samples with respect to the 'true' posterior samples obtained from performing high-dimensional MCMC for $p(x|\mathcal{D})$. One example of such test performed by [31] is a comparison between the resulting mean and variance of the samples produced by the two procedures. As observed in [31], a small Hellinger distance (49) between two distributions indeed leads to a small difference between the corresponding means and variances, according to Lemma 6.37 in [159].

Note that the original AS-MCMC procedure requires access to the gradient-based embedding $\hat{W}_{AS}$, as well as a relatively large number of log-likelihood evaluations for $\{\hat{g}(y_i)\}_{i=1}^{N_{large}}$. Instead, we will use our learned (gradient-free) low-dimensional subspace $\hat{W}_{\mathrm{GP}}$, together with the computationally fast approximation $\{g_{\mathrm{GP}}(y_i)\}_{i=1}^{N_{large}}$ obtained from the GP prior (20) and training set

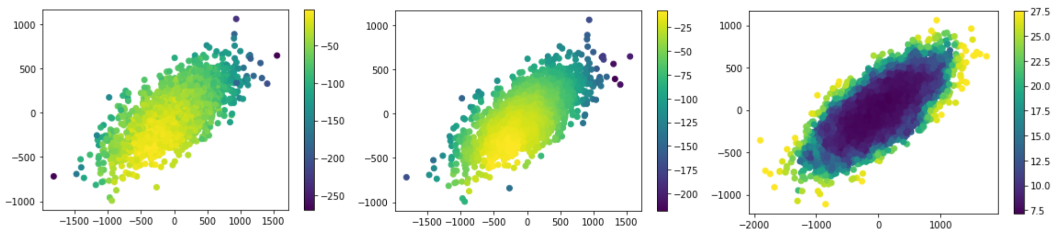
Algorithm 1 Active Subspace MCMC (AS-MCMC), Algorithm 1 from [31]

```
$W_1 \in \mathbb{R}^{D\times d}$ : the gradient-based active subspace $\hat{W}_{AS}$
$W_2 \in \mathbb{R}^{D\times(D-d)}$ : orthogonal complement of $W_1$
$\hat{g}(y) : \mathbb{R}^d \to \mathbb{R}$ : low-dimensional (simulator-based) approximation for the log-likelihood
Pick an initial value $y_1 \in \mathbb{R}^d$ and compute $\hat{g}(y_1)$. Set $k = 1$.
    1. Draw $y' \sim \mathcal{N}_d(y_k, \sigma^2_{prop} I_d)$ from the proposal density centred at $y_k$
    2. Compute the acceptance ratio
           $\gamma(y_k, y') = \min\left(1, \frac{\exp \hat{g}(y')p(y')}{\exp \hat{g}(y_k)p(y_k)}\right),$
       where $p(y) \sim \mathcal{N}(0, I_d)$
    3. Draw $t$ uniformly from $[0, 1]$.
    4. If $\gamma(y_k, y') \geq t$, set $y_{k+1} = y'$. Otherwise, set $y_{k+1} = y_k$.
    5. Increment k and repeat.
For each sample $y_k \in \mathbb{R}^d$, construct $M$ (correlated) samples $x_{k,m} \in \mathbb{R}^D$:
           $x_{k,m} = W_1 y_k + W_2 z_{k,m},$
where $z_{k,m} \sim p(z) = \mathcal{N}(0, I_{D-d})$ are i.i.d. samples. For large enough $k$, $x_{k,m} \sim \tilde{p}(x|\mathcal{D})$.
```

$\{x_i, L(x_i)\}_{i=1}^N$ of medium size N. We summarize our procedure in Algorithm 2, which is a simple modification of Algorithm 1. Note that we only use the predictive mean $\bar{m}(y)$ to approximate the log-likelihood. Incorporating the predictive uncertainty $\bar{k}(y, y)$ is an interesting avenue for future work. Note, however, that the predictive uncertainty can be well-calibrated, as we demonstrate through $\bar{k}(y_i, y_i)$ for i.i.d. samples $y_i = \hat{W}_{GP}^T x_i$ in the right plot of Figure 38 for the Elliptic-PDE simulator described in the next section.

Algorithm 2 Learned Embedding Active Subspace MCMC (LE-AS-MCMC)

```
$W_1 \in \mathbb{R}^{D \times d}$ : orthogonal basis for the learned subspace $\hat{W}_{\text{GP}}$
$W_2 \in \mathbb{R}^{D \times (D-d)}$ : orthogonal complement of $W_1$
$\bar{m}(y) : \mathbb{R}^d \to \mathbb{R}$ : the predictive mean of the GP posterior $\mathcal{N}(\bar{m}(y), \bar{k}(y, y))$
for the GP prior $g_{\text{GP}}(y = \hat{W}_{\text{GP}}^T x)$ (20) and training set $\{x_i, L(x_i)\}_{i=1}^N$
Pick an initial value $y_1 \in \mathbb{R}^d$ and compute $\bar{m}(y_1)$. Set $k = 1$.

    1. Draw $y' \sim \mathcal{N}(y_k, \sigma_{prop}^2)$ from the proposal density centred at $y_k$.

    2. Compute the acceptance ratio

       $$\gamma(y_k, y') = \min\left(1, \frac{\exp \bar{m}(y')p(y')}{\exp \bar{m}(y_k)p(y_k)}\right),$$

       where $p(y) \sim \mathcal{N}(0, I_d)$

    3. Draw $t$ uniformly from $[0, 1]$.

    4. If $\gamma(y_k, y') \geq t$, set $y_{k+1} = y'$. Otherwise, set $y_{k+1} = y_k$.

    5. Increment k and repeat.

For each sample $y_k \in \mathbb{R}^d$, construct $M$ (correlated) samples $x_{k,m} \in \mathbb{R}^D$:

       $$x_{k,m} = W_1 y_k + W_2 z_{k,m},$$

where $z_{k,m} \sim p(z) = \mathcal{N}(0, I_{D-d})$ are i.i.d. samples. For large enough $k$,
$x_{k,m} \sim \tilde{p}_{\text{GP}}(x|\mathcal{D})$.
```



(a) $\{y_i = \hat{W}_{GP}^T x_i, L(x_i)\}$ (b) $\bar{m}(y_i)$ (c) $\sqrt{\bar{k}(y_i, y_i)}$

Figure 38: For the Elliptic-PDE simulator $f(x)$ described in Section 4.3, we project 1000 i.i.d. prior samples $x_j \sim \mathcal{N}_{100}(0, I)$ onto the 2D learned subspace $\hat{W}_{GP}$ (from Algorithm 2). We plot the corresponding log-likelihood $L(x_j)$ versus the projected inputs $y_j = \hat{W}_{GP}^T x_j$ (left), the GP predictive mean $\bar{m}(y_j)$ (center), and the GP predictive standard deviation $\sqrt{\bar{k}(y_j, y_j)}$ (right). We see that $\bar{m}(y_j)$ approximates well $f(x_j)$, and the predictive uncertainties $\bar{k}(y_j, y_j)$ are well-calibrated (note that the uncertainty grows as we move towards the tails of the prior $p(x)$ and thus away from the training set, as the training set consists of samples from $p(x)$).

4.3 COMPUTER EXPERIMENTS

As in the original source [31] for addressing log-likelihoods with a low-dimensional active subspace, we consider the following two experiments:

- Synthetic quadratic simulator $f : \mathbb{R}^2 \to \mathbb{R}$, with $f(x) = \frac{1}{2}x^T A x$ and standard Gaussian prior $p(x) \sim \mathcal{N}_2(0, I)$. The observed data is $\mathcal{D} = 0.9$, and the noise level follows $\sigma^2 = 0.1$. For $A = Q \begin{bmatrix} 1 & 0 \\ 0 & 0.01 \end{bmatrix} Q^T$ and $Q = \frac{1}{2}\begin{bmatrix} \sqrt{2} & \sqrt{2} \\ -\sqrt{2} & \sqrt{2} \end{bmatrix}$ which we use here, the log-likelihood (48) has a one-dimensional active subspace, $L(x) \approx g(\hat{W}_{AS}^T x)$ with $g : \mathbb{R} \to \mathbb{R}$.
- Elliptic-PDE simulator $f : \mathbb{R}^{100} \to \mathbb{R}^7$, where $y = f(x)$ is the solution of the PDE described in Section 2.4, evaluated at seven points on the

right boundary of the domain $[0,1]^2$. The inputs follow a standard multivariate Gaussian prior $x \sim \mathcal{N}_{100}(0, I)$, which is used via (21) to generate a Gaussian random field (GRF) prior of short lengthscale (23) for the PDE coefficients. We sample x_{true} from $p(x)$, which is used to generate the observations $\mathcal{D} = f(x_{true}) + \epsilon$. The noise level $\Sigma_{obs} := \sigma_n^2 I_7$ is considered to be $\sigma_n^2 = 10^{-4}||f(x_{true})||_2^2$, i.e. approximately 1% of the noise-free data $f(x_{true})$. Finally, the resulting log-likelihood (48) has been shown to have a two-dimensional active subspace, $L(x) \approx g(\hat{W}_{AS}^T x)$ with $g : \mathbb{R}^2 \to \mathbb{R}$ [31].

As usual, we consider a constant GP prior mean $m(W_{\text{GP}}^T x) = C$ for (20); the performance of the GP emulators was not improved by a linear or a quadratic GP prior mean. We have used the LBFGS routine for maximizing the marginal likelihood (5) with respect to the GP hyperparameters $\theta = (C, \sigma, l)$ and the embedding $W_{\text{GP}} \in \mathbb{R}^{D \times d}$, with learning rate 0.1 for 15 iterations. If computational time permits, we suggest restarting the training process several times, and select the configuration $\{\hat{\theta}, \hat{W}_{\text{GP}}\}$ which leads to the best approximation of the log-likelihood $L(x)$ via cross-validation (see Section 2.1), or by trial-and-error on a separate validation (test) set of small size $\{x_i^\star, L(x_i^\star)\}_{i=1}^{N_{small}}$.

In Table 7, we look at the performance of our procedure in terms of approximating the gradient-based active subspace $\hat{W}_{AS}$ and solving the Bayesian inverse problem. Namely, we use the first subspace angle (FSA) criterion introduced in Section 3.3 as a measure of discrepancy between the subspace spanned by the columns of our learned embedding $\hat{W}_{\text{GP}}$ and the columns of $\hat{W}_{AS}$. Furthermore, we look at the KL divergence (for simplicity)

$$D_{KL}(p||\tilde{p}_{\text{GP}}) := \int_{\mathbb{R}^D} \log\left(\frac{p(x|\mathcal{D})}{\tilde{p}_{\text{GP}}(x|\mathcal{D})}\right) p(x|\mathcal{D}) dx \tag{51}$$

between the true posterior $p(x|\mathcal{D}) \propto \exp L(x) p(x)$ and our GP approximate posterior $\tilde{p}_{\text{GP}}(x|\mathcal{D}) \propto \exp \bar{m}(\hat{W}_{\text{GP}}^T x) p(x)$, obtained using the GP predictive mean.

In particular, we note the success of the gradient-free learned embedding $\hat{W}_{\text{GP}}$ in approximating the gradient-based active subspace $\hat{W}_{AS}$ in both cases. Note that the KL divergence is computed using numerical integration over the high-dimensional space $\mathbb{R}^D$, and thus we cannot use it for the $D = 100$ Elliptic PDE. Instead, we perform a qualitative comparison between the samples of $\tilde{p}_{\text{GP}}(x|\mathcal{D})$ produced by our procedure LE-AS-MCMC and the posterior samples $p(x|\mathcal{D})$ obtained via a high-dimensional MH-MCMC procedure in [31].

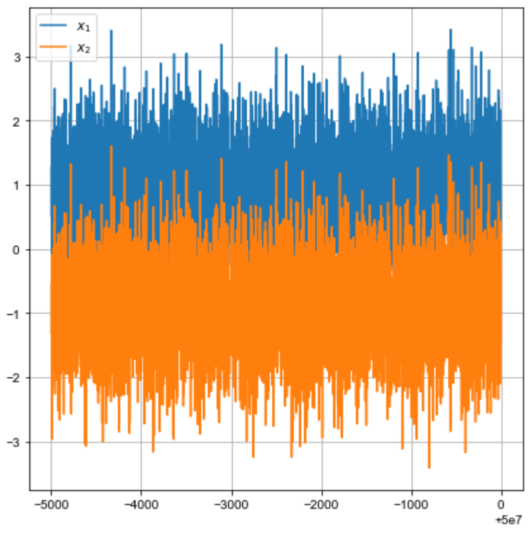
Table 7: Performance of our GP emulator $g_{\text{GP}}(\hat{W}_{\text{GP}}^T x)$ for the synthetic quadratic simulator with training size $N = 100$, and the elliptic PDE simulator with training size $N = 5000$.

	quadratic	elliptic PDE
FSA	0.000	0.133
KL	0.005	-

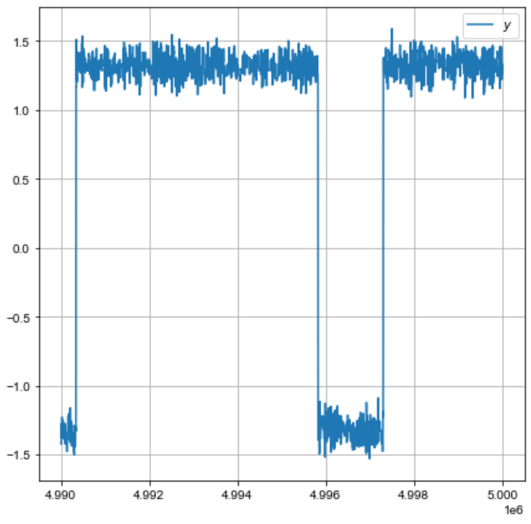
4.3.1 *Synthetic quadratic function*

For the synthetic quadratic function, we have used a proposal density with $\sigma_{prop} = \sqrt{0.5}$ as in [31], and a Markov chain of length $K = 5 \cdot 10^6$, which leads to $M \cdot K = 10 \cdot 5 \cdot 10^6$ samples $x_{m,k}$ according to Algorithm 2. We discarded the first 20% samples of the chain as burn-in, and used the rest of the samples $x_{m,k} \sim \tilde{p}_{\text{GP}}(x|\mathcal{D})$ for qualitative analysis.

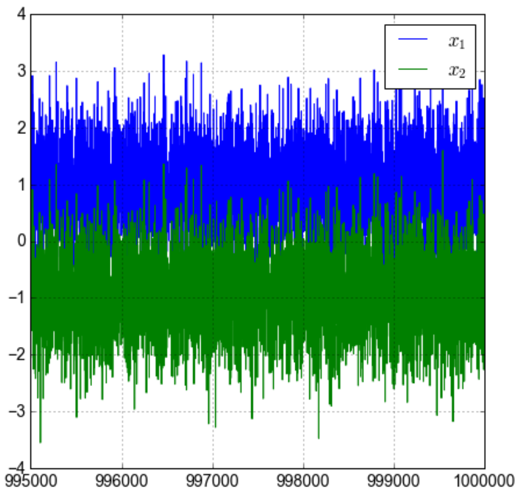
We present in Figure 39a the last $5 \cdot 10^3$ samples $x_{k,m} \sim \tilde{p}_{\text{GP}}(x|\mathcal{D})$, and in Figure 39b the last 10^4 samples y_k, both figures obtained using LE-AS-MCMC (Algorithm 2). As a comparison, we present in Figure 39c and 39d, respectively, the analogue figures (trace plots) from [27] obtained using AS-MCMC (Algorithm 1) for a Markov chain of length $K = 10^5$. The source [27] is the Github page associated with the paper [31]. We note that the results are very similar (i.e. we observe fast exploration (mixing) within the the two posterior modes of $y_k \sim \tilde{p}_{\mathrm{GP}}(y|\mathcal{D})$ and $y_k \sim \tilde{p}(y|\mathcal{D})$, respectively), which demonstrate that LE-AS-MCMC also benefits from the very good mixing performance of AS-MCMC. In Figure 39e and 39f we compare the kernel density estimates obtained from the LE-AS-MCMC samples $y_k \sim \tilde{p}_{\mathrm{GP}}(y|\mathcal{D})$ and the AS-MCMC samples $y_k \sim \tilde{p}(y|\mathcal{D})$ from [27], respectively.



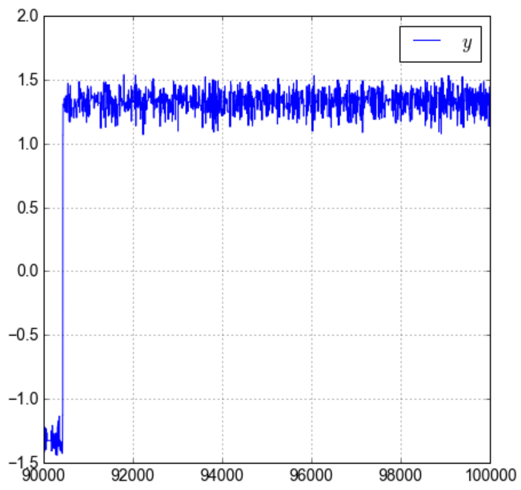
(a) LE-AS-MCMC samples $x_{k,m} \in \mathbb{R}^2$



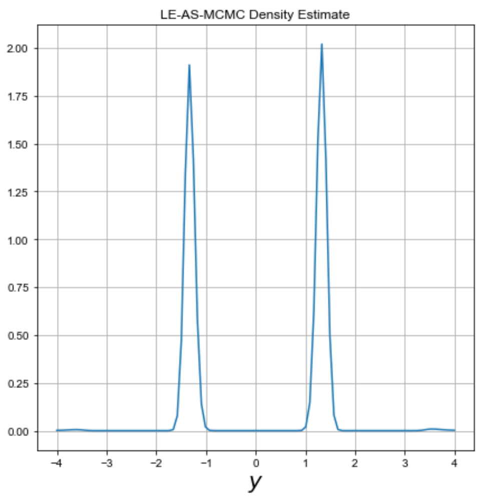
(b) LE-AS-MCMC samples $y_k \in \mathbb{R}$



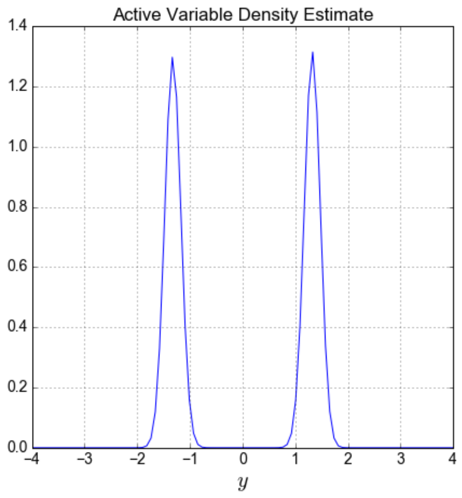
(c) AS-MCMC samples $x_{k,m} \in \mathbb{R}^2$ [27]



(d) AS-MCMC samples $y_k \in \mathbb{R}$ [27]



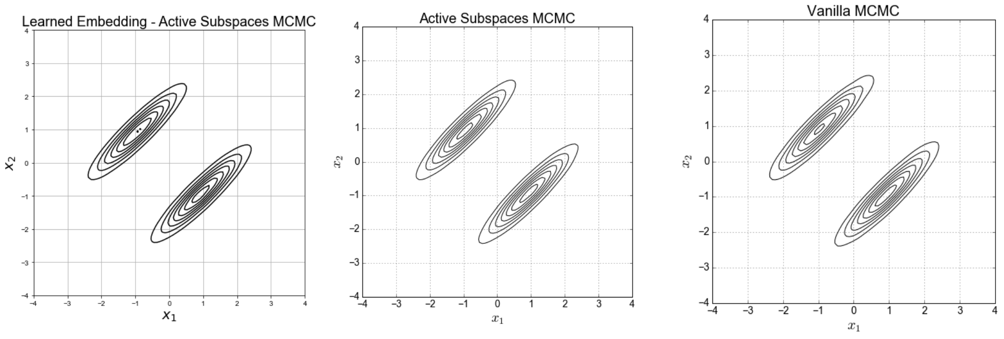
(e) $\tilde{p}_{\mathrm{GP}}(y|\mathcal{D})$ (LE-AS-MCMC)



(f) $\tilde{p}(y|\mathcal{D})$ (AS-MCMC) [27]

Figure 39: MCMC performance for LE-AS-MCMC versus AS-MCMC

Finally, in Figure 40a, 40b, and 40c, we compare the (approximate) posterior distributions obtained from the LE-AS-MCMC samples $x_{k,m} \sim \tilde{p}_{\mathrm{GP}}(x|\mathcal{D})$, the AS-MCMC samples $x_{k,m} \sim \tilde{p}(x|\mathcal{D})$ from [27], and the vanilla MH-MCMC over the high-dimensional space $x_{k,m} \sim p(x|\mathcal{D})$ from [27], respectively. Note that on all the benchmarks, we successfully match the impressive performance of the original AS-MCMC procedure.



(a) $\tilde{p}_{\mathrm{GP}}(x|\mathcal{D})$ (b) $\tilde{p}(x|\mathcal{D})$ [27] (c) $p(x|\mathcal{D})$ [27]

Figure 40: (Approximate) posterior density from LE-AS-MCMC samples versus AS-MCMC and (vanilla) MH-MCMC; recall the small KL divergence between $p(x|\mathcal{D})$ and $\tilde{p}_{\mathrm{GP}}(x|\mathcal{D})$ (Table 7)

4.3.2 *Elliptic PDE*

For the elliptic PDE, we have used a proposal density with $\sigma_{prop} = 0.1$ as in [31], and a Markov chain of length $K = 10^5$, which lead to $M \cdot K = 10 \cdot 10^5$ samples $x_{m,k}$ according to Algorithm 2. We discarded the first 20% samples of the chain as burn-in, and used the rest of the samples $x_{m,k} \sim \tilde{p}_{\mathrm{GP}}(x|\mathcal{D})$ for qualitative analysis.

As a first check (see e.g. [15]), we look at whether the approximate posterior samples $x_{m,k}$ are in line with the observations $\mathcal{D}$ by comparing the corresponding simulator responses $f(x_{m,k})$ with both the observed measurements $\mathcal{D}$ and the noise-free simulator response $f(x_{true})$ in Figure 41. As we

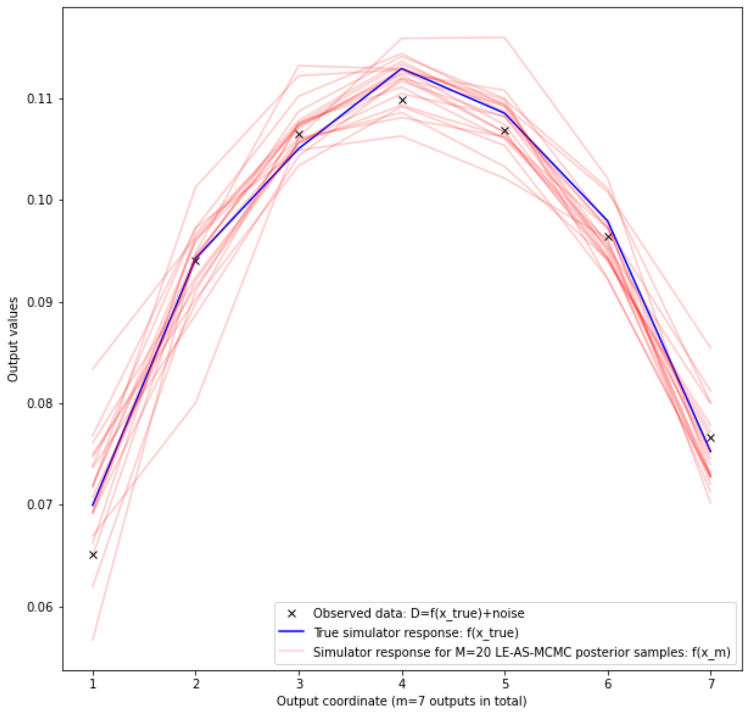
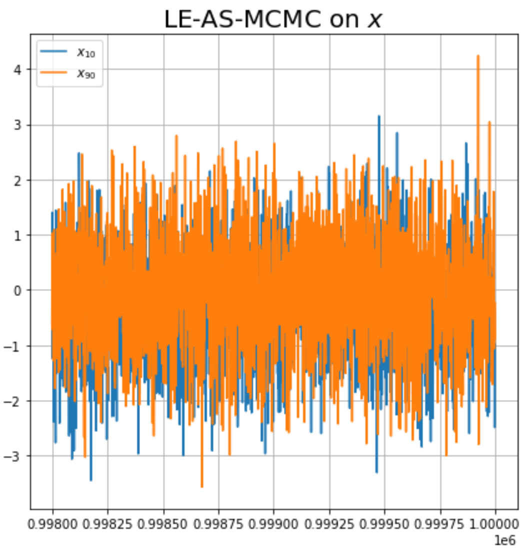


Figure 41: Simulator responses for the LE-AS-MCMC samples $x_{k,m}$ match the observed data $\mathcal{D}$

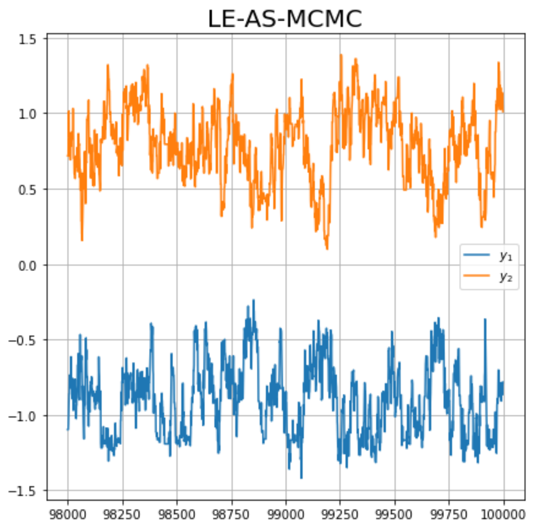
can see, the LE-AS-MCMC samples produce responses in line with the noise level in the observed data.

Note that since we use a different observed data $\mathcal{D}$ for our Bayesian inverse problem compared with [31], we also have a different active subspace $\hat{W}_{AS}$. Nonetheless, some characteristics of the posterior distribution $p(x|\mathcal{D})$ should remain the same, as shown in the new set of experiments from [27]. Therefore, we proceed in comparing our LE-AS-MCMC performance with the analogue results for AS-MCMC and high-dimensional (vanilla) MH-MCMC from [31].

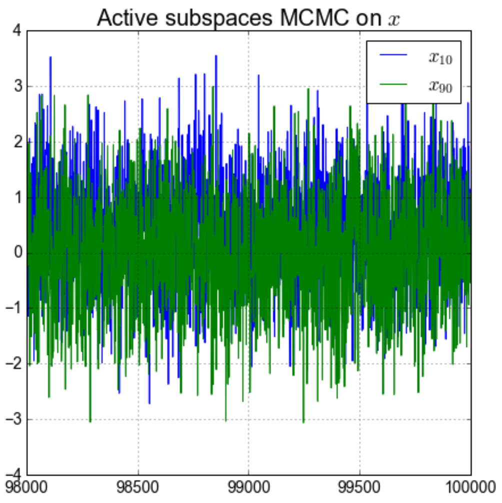
We present in Figure 42a and 42b two dimensions (10 and 90) out of $D = 100$ of the last 2000 samples $x_{k,m}$, and the last 2000 samples y_k, respectively, both figures obtained using LE-AS-MCMC (Algorithm 2). As a comparison, we present in Figure 42c and 42d, respectively, the analogue figures from [27] obtained using AS-MCMC (Algorithm 1), for a Markov chain of length $K = 10^4$. We note that the results are very similar, as in the case of the synthetic quadratic simulator, which again demonstrates that the efficiency (fast mixing) of AS-MCMC translates to LE-AS-MCMC.



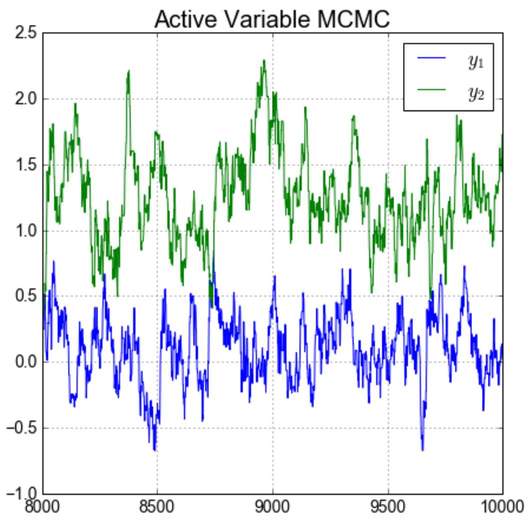
(a) LE-AS-MCMC samples $x_{k,m} \in \mathbb{R}^{100}$



(b) LE-AS-MCMC samples $y_k \in \mathbb{R}^2$



(c) AS-MCMC samples $x_{k,m} \in \mathbb{R}^{100}$ [27]



(d) AS-MCMC samples $y_k \in \mathbb{R}^2$ [27]

Figure 42: MCMC performance for LE-AS-MCMC versus AS-MCMC

See [31] for a thorough comparison between AS-MCMC and the high-dimensional vanilla MH-MCMC for this experiment, including autocorrelation plots (Fig 5.8, [31]), comparison between the resulting means and variances (Fig 5.9, [31]), and effective sample sizes (Table 5.1, [31]). Note that the aforementioned comparison between means and variances is challenging due to the high autocorrelation and low effective sample size for the high-dimensional vanilla MH-MCMC.

Similar to [31], we project the LE-AS-MCMC samples $x_{k,m}$ onto the active subspace $\hat{W}_{AS}^T x_{k,m}$ and the inactive subspace $(\hat{W}_{AS}^{\perp})^T x_{k,m} \in \mathbb{R}^{D-d}$, respectively. In [31], 'true' posterior samples $x_{m,k} \sim p(x|\mathcal{D})$ from MH-MCMC were used in order to show that the true posterior only gets updated in the (log-likelihood) active directions, whereas in the inactive $D-d$ directions it stays close to the $\mathcal{N}(0,1)$ uncorrelated prior. Here, we want to demonstrate that the LE-AS-MCMC approximate posterior $\tilde{p}_{\mathrm{GP}}(x|\mathcal{D})$ also follows the same property.

We present the contours of the marginals posterior distributions using high-dimensional MH-MCMC for $p(x|\mathcal{D})$ in Figure 43 (MCMC chain length $K = 10^5$); this figure is borrowed from [31]. The figure considers the active directions $\hat{W}_{AS}^T x$, denoted by (y_1, y_2), and the first two inactive directions (i.e. the first two elements of $(\hat{W}_{AS}^{\perp})^T x$), denoted by (y_3, y_4). We can see the joint marginal posterior in the two active directions (1 and 2), the joint marginal posterior in one active direction (1 or 2) and one inactive direction (3 or 4), and finally, the joint marginal posterior in two inactive directions (3 and 4). We can see that the posterior differs significantly from the $\mathcal{N}(0,1)$ uncorrelated prior only in the active directions, as mentioned in [31].

We show the analogue contours arising from the projected LE-AS-MCMC samples $(\hat{W}_{AS}^T x_{m,k}, (\hat{W}_{AS}^{\perp})^T x_{m,k})$ in Figure 44, together with our corresponding data generating coefficients $(\hat{W}_{AS}^T x_{true}, (\hat{W}_{AS}^{\perp})^T x_{true})$ as a red dot. It seems that the LE-AS-MCMC contours are concentrated in accordance to the characteristics of the vanilla MH-MCMC contours from [31], and they manage to capture the true coefficients $(\hat{W}_{AS}^T x_{true}, (\hat{W}_{AS}^{\perp})^T x_{true})$. As discussed above, we conclude that the approximate posterior $\tilde{p}_{\mathrm{GP}}(x|\mathcal{D})$ also satisfies the same property as the true posterior $p(x|\mathcal{D})$, i.e. it only gets updated in the active directions, whereas in the inactive directions it stays close to the $\mathcal{N}(0,1)$ uncorrelated prior.

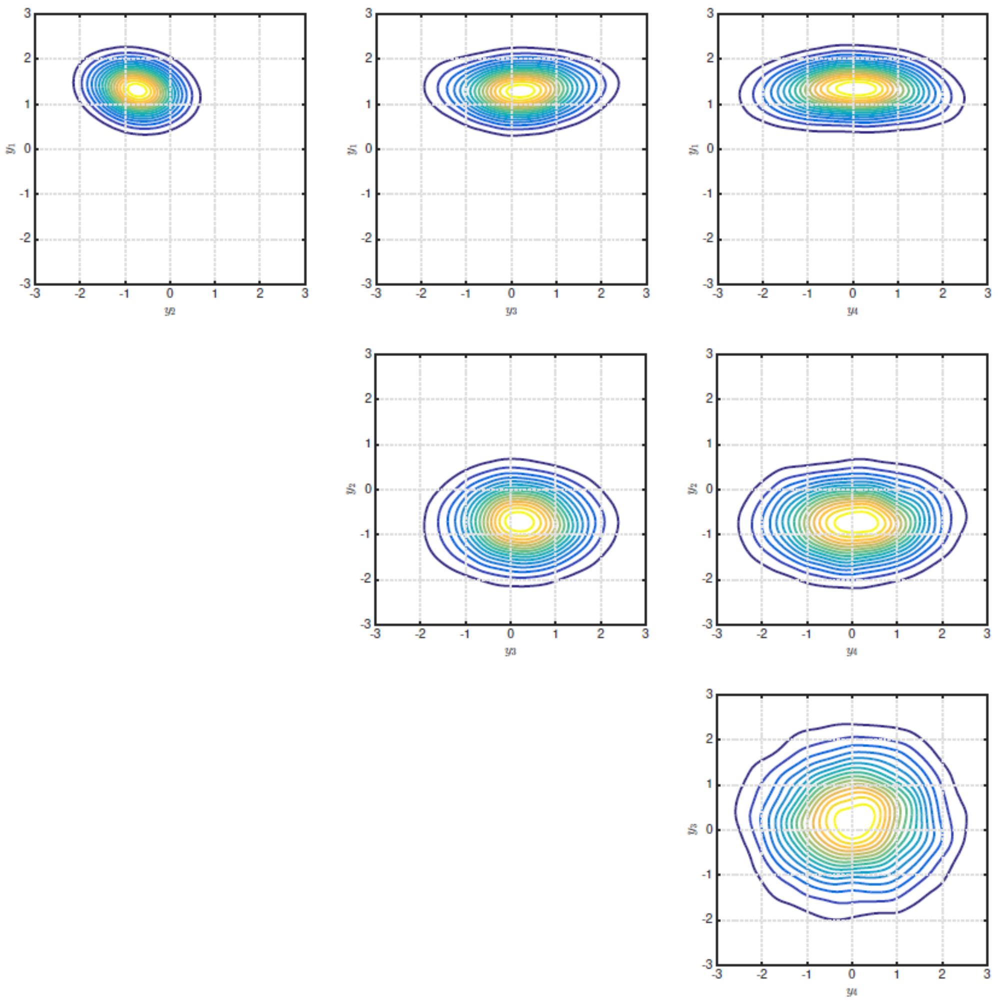


Figure 43: Figure from [31]; marginal posterior distributions for the active directions (y_1, y_2) and the first two inactive directions (y_3, y_4), using vanilla high-dimensional MH-MCMC for $x_{m,k} \sim p(x|\mathcal{D})$

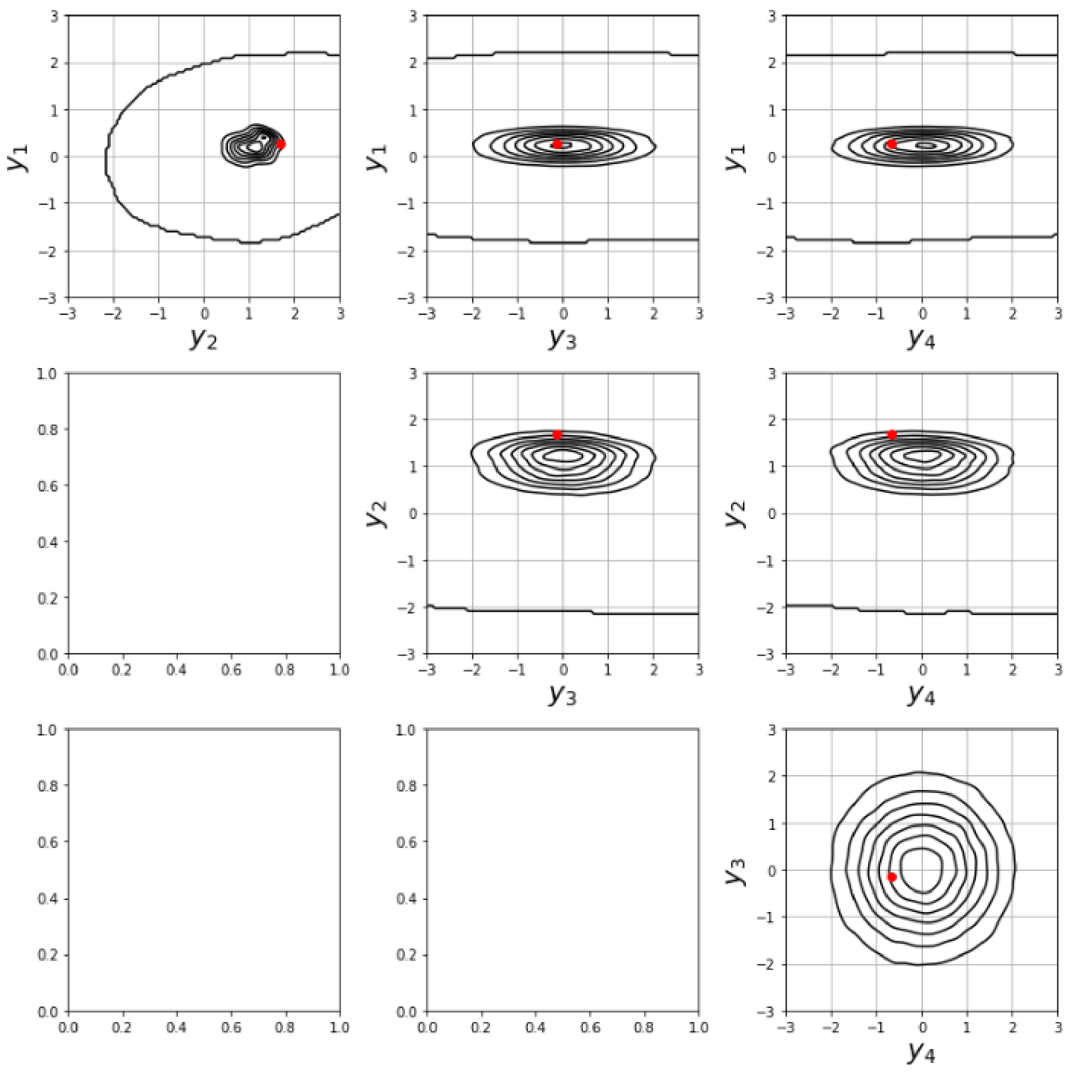


Figure 44: Marginal posterior distributions for the active directions (y_1, y_2) and the first two inactive directions (y_3, y_4) using LE-AS-MCMC samples $x_{m,k}$, together with our data generating coefficients $(\hat{W}_{AS}^T x_{true}, (\hat{W}_{AS}^{\perp})^T x_{true})$ as a red dot

4.3.3 *Conclusion*

In conclusion, following the observation that many log-likelihoods used in Baysian inverse problems possess an active subspace [31], we have shown that our gradient-free procedure LE-AS-MCMC can efficiently exploit the (latent) low-dimensional active subspace structure for posterior sampling. For the synthetic quadratic simulator (Subsection 4.3.1), we have demonstrated fast mixing (in line with the AS-MCMC procedure, Figure 39), as well as accurate posterior approximation (Table 7 and Figure 40). For the Elliptic PDE simulator (Subsection 4.3.2), we have also demonstrated fast mixing (in line with the AS-MCMC procedure, Figure 42), as well as a good match $f(x_{m,k})$ for the observed data $\mathcal{D}$ ($x_{m,k} \approx \tilde{p}_{\text{GP}}(x|\mathcal{D})$, Figure 41), and a good recovery of the unknown x_{true} (according to the active subspace representation, Figure 44). Using the active subspace representation of the approximate posterior samples $x_{m,k} \sim \tilde{p}_{\text{GP}}(x|\mathcal{D})$ (Figure 44), we have seen that $\tilde{p}_{\text{GP}}(x|\mathcal{D})$ only gets updated from $p(x)$ in the active subspace directions (property satisfied by the true posterior $p(x|\mathcal{D})$), and that the posterior contours of $\tilde{p}_{\text{GP}}(x|\mathcal{D})$ resemble the characteristics of the true posterior contours $p(x|\mathcal{D})$ from [31]. Recall that posterior contours $p(x|\mathcal{D})$ should have similar characteristics for different pairs $(\mathcal{D}, x_{true})$, as shown in [31] and [27]; we did not produce posterior contours $p(x|\mathcal{D})$ for our $\mathcal{D}$ and x_{true}, due to the large computational time.

4.4 NEURAL NETWORK COMPARISON

For the high-dimensional log-likelihood $L(x) : \mathbb{R}^{100} \rightarrow \mathbb{R}$ with a 2-dimensional active subspace corresponding to the Elliptic PDE simulator considered in the previous section, we perform a quick comparison between the performance of GP emulators and (artificial) neural network emulators.

Let us consider a one hidden layer neural network

$$g_{NN}(x) = W_2^T \phi(W_1^T x + b_1) + b_2$$

with standard (element-wise) ReLU activation function $\phi(u) := \max(u, 0)$. The matrices $W_1 \in \mathbb{R}^{D \times h_1}$ and $W_2 \in \mathbb{R}^{h_1 \times 1}$ are known as weights, where h_1 is size of the hidden layer; $b_1 \in \mathbb{R}^{h_1}$ and $b_2 \in \mathbb{R}$ are known as biases. Our approach is motivated by the theoretical result from [8], which shows that when the target function has a low-dimensional linear structure, i.e. $L(x) = g(A^T x)$ for some $A \in \mathbb{R}^{D \times d}$ with $d \ll D$, shallow (one-hidden layer) neural networks with ReLU activation function are able to break the curse of dimensionality by automatically adapting to the low intrinsic dimension d. This result was further discussed and extended in [66], where a theoretical comparison with kernel methods was included, as well as practical experiments from image classification with deep (i.e. multiple hidden layers) and convolutional neural networks; see [187] for an introduction to modern neural network models.

For our Elliptic PDE log-likelihood $L(x) : \mathbb{R}^{100} \to \mathbb{R}$, we have used a neural network $g_{NN}(x) = W_2^T \phi(W_1^T x + b_1) + b_2$ with one hidden layer of size $h_1 = 20$ and ReLU activation function ϕ. We have tried changing the hidden size h_1, as well as adding multiple layers, but the predictive performance of the neural network was relatively insensitive to these changes. The neural network parameters $(\hat{W}_1, \hat{b}_1, \hat{W}_2, \hat{b}_2)$ are selected by minimizing the standard training loss

$$\frac{1}{N} \sum_{i=1}^{N} (L(x_i) - g_{NN}(x_i))^2$$

with respect to (W_1, b_1, W_2, b_2) via gradient descent optimization. The training inputs $\{x_i\}_{i=1}^N$ are sampled independently from the prior distribution $x \sim \mathcal{N}(0, I_{100})$, whereas a separate test set $\{x_i^\star\}_{i=1}^{N_{test}}$ of independent prior samples was used to test the performance of the resulting neural network

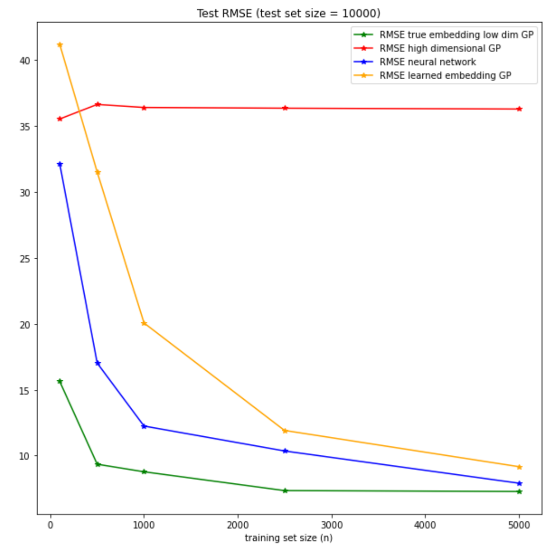
$\hat{g}_{NN}(x) = \hat{W}_2^T \phi(\hat{W}_1^T x + \hat{b}_1) + \hat{b}_2$ via the root mean squared error (RMSE) criterion:

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N_{test}} (L(x_i^\star) - \hat{g}_{NN}(x_i^\star))^2}.$$

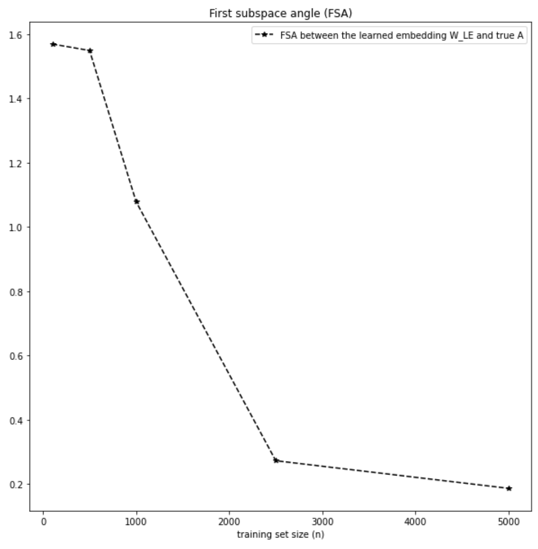
We have compared $\hat{g}_{NN}$ with the following GP emulators:

- high-dimensional GP baseline (NO-DR) $g_{\mathrm{GP}}(x) : \mathbb{R}^{100} \to \mathbb{R}$ (3)
- gradient-based active subspace $g_{\mathrm{GP}}(\hat{W}_{AS}^T x) : \mathbb{R}^2 \to \mathbb{R}$ (19)
- (gradient-free) learned embedding $g_{\mathrm{GP}}(\hat{W}_{\mathrm{GP}}^T x) : \mathbb{R}^2 \to \mathbb{R}$, which is used in the previous section for our LE-AS-MCMC procedure.

In Figure 45a, we present the RMSE comparison between the neural network predictions $\hat{g}_{NN}(x_i^\star)$ and the GP predictions obtained via the posterior means $\bar{m}(x_i^\star)$ (high-dimensional GP), $\bar{m}(\hat{W}_{AS}^T x_i^\star)$ (active subspace GP), and $\bar{m}(\hat{W}_{\mathrm{GP}}^T x_i^\star)$ (learned embedding GP). In Figure 45b, we plot the first subspace angle (FSA) between the subspace spanned by the columns of the learned embedding $\hat{W}_{\mathrm{GP}}$ and the columns of $\hat{W}_{AS}$. Note that the FSA plot uses 'true A' to denote the gradient-based active subspace $\hat{W}_{AS}$, whereas the RMSE plot uses 'true embedding low dim GP' to denote $g_{\mathrm{GP}}(\hat{W}_{AS}^T x)$.



(a) NN vs GP emulators (first $N = 100$)



(b) FSA between $\hat{W}_{\mathrm{GP}}$ and $\hat{W}_{AS}$

Figure 45: (Best viewed in colour) Neural network (NN) vs GP emulators

Our conclusions can be summarized as follows. First, note that the high-dimensional GP suffers from the curse of dimensionality. This result is perhaps surprising, as we know from [103] that the high-dimensional GP can work very well for the related simulator $f_{\lambda^S, u^S}(x)$ from Section 2.4 based on the same elliptic PDE, and with the same short lengthscale GRF prior for the PDE coefficients. The neural network adapts faster to the intrinsic low-dimensionality compared with the learned embedding $g_{\text{GP}}(\hat{W}_{\text{GP}}^T x)$, as we can see a large performance gap for training size $N \in \{500, 1000\}$. As the FSA between $\hat{W}_{\text{GP}}$ and $\hat{W}_{AS}$ decreases significantly ($N \in \{2500, 5000\}$), the gap between the neural network and $g_{\text{GP}}(\hat{W}_{\text{GP}}^T x)$ closes down, although the neural network maintains a slight advantage. At the largest training budget $N = 5000$, the neural network performs close to the gradient-based $g_{\text{GP}}(\hat{W}_{AS}^T x)$. Nonetheless, it is important to mention that due to its high-dimensional input, the neural network emulator $\hat{g}_{NN}(x) \approx L(x)$ cannot be easily adapted for efficient low-dimensional posterior sampling, as opposed to the GP emulators $g_{\text{GP}}(y = \hat{W}_{AS}^T x)$ (AS-MCMC) or $g_{\text{GP}}(y = \hat{W}_{\text{GP}}^T x)$ (LE-AS-MCMC).

5 RANDOMIZED MAXIMUM LIKELIHOOD VIA HIGH-DIMENSIONAL BAYESIAN OPTIMIZATION

As discussed in Section 1.5 and Chapter 4, posterior sampling for high-dimensional Bayesian inverse problems where the log-likelihood $L(x)$ has a low-dimensional active subspace structure is a problem commonly faced in real-world applications. Existing approaches assume that the computational budget is sufficient to estimate the active subspace, either via gradient evaluations $\{x_i, \nabla L(x_i)\}_{i=1}^{M}$, or gradient-free methods $\{x_i, L(x_i)\}_{i=1}^{N}$ for large enough N. Here, we tackle the more challenging (and practically relevant) case where we do not have sufficient computational budget to satisfactorily estimate the active subspace. In this regard, we develop a high-dimensional Bayesian optimization (HD-BO) approach to solve the Randomized Maximum Likelihood (RML) problem. Both HD-BO and RML will be introduced in detail throughout this chapter; here we present a quick summary of our contribution.

RML is an approximate posterior sampling methodology based on multi-objective optimization, first developed for petroleum engineering applications in [124]. By sharing data between the different objective functions, we are able to implement RML at a greatly reduced computational cost compared to existing methods, allowing us to efficiently sample from the posterior distribution of the inverse problem. We demonstrate the benefits of this approach in comparison to alternative optimization methods on a variety of synthetic and real-world problems, including medical and fluid dynamics applications. Furthermore, we show that the samples produced by our method cover well the high-posterior density regions in all of the experiments.

5.1 INTRODUCTION

As in Chapter 4, we consider Bayesian inverse problems, where the goal is to sample from the posterior distribution

$$p(x|\mathcal{D}) = \frac{p(\mathcal{D}|x)p(x)}{p(\mathcal{D})}$$

of the unknown parameters $x \in \mathbb{R}^D$ given the observed data $\mathcal{D} \in \mathbb{R}^m$.

Randomized Maximum Likelihood (RML) was introduced by [124], as an approximate posterior sampling methodology. RML is formulated for the situation where the observations are subject to Gaussian distributed errors (i.e., a Gaussian likelihood) and where the prior distribution is also Gaussian: $x \sim \mathcal{N}_D(\mu, \Sigma)$. The algorithm proceeds by first perturbing the data and the prior mean, and then optimizing the unnormalised log-posterior using these perturbed values. See Algorithm 3.

Algorithm 3 Randomized Maximum Likelihood (RML)

```
$n_{RML}$ : number of samples required
for $n \in [n_{RML}]$ do
    1. Sample $\mathcal{D}_n \sim \mathcal{N}_m(\mathcal{D}, \Sigma_{\text{obs}})$ from the Gaussian likelihood
    2. Sample $\mu_n \sim \mathcal{N}_D(\mu, \Sigma)$ from the Gaussian prior
    3. Construct $\log p(\mathcal{D}|x)p(x)$ w.r.t. the randomizations $(\mathcal{D}_n, \mu_n)$

        $O_n(x) := \log \mathcal{N}_m(f(x)|\mathcal{D}_n, \Sigma_{\text{obs}}) + \log \mathcal{N}_D(x|\mu_n, \Sigma)$    (52)

    4. Obtain $x_n^\star$ as the maximizer $x_n^\star = \arg\max_x O_n(x)$.
end for
```

Here, we use the notation $[n_{RML}] = \{1, 2, \ldots, n_{RML}\}$. Thus, RML solves n_{RML} optimization problems, each with a different objective function, $O_n(x)$.

The resulting solutions $\{x_n^\star\}_{n=1}^{n_{RML}}$ are regarded as approximate samples from the posterior distribution $p(x|\mathcal{D})$; the samples are only exact draws from the posterior when the simulator $f(x)$ is linear, as we will demonstrate next. For the original proof, see [124].

Indeed, in case of a linear simulator $f(x) := Hx$, Gaussian prior distribution $x \sim \mathcal{N}(\mu, \Sigma)$, and Gaussian likelihood $\mathcal{D}|x \sim \mathcal{N}(Hx, \Sigma_{\text{obs}})$, the posterior distribution $p(x|\mathcal{D})$ is also Gaussian: $x|\mathcal{D} \sim \mathcal{N}(m, V)$, where $V := (\Sigma^{-1} + H^\top \Sigma_{\text{obs}}^{-1} H)^{-1}$ and $m := V(\Sigma^{-1}\mu + H^\top \Sigma_{\text{obs}}^{-1}\mathcal{D})$. To see why randomized maximum likelihood (RML) from Algorithm 3 produces exact samples from the posterior distribution in this case, we first recall the RML objective functions from (52):

$$O_n(x) := \log \mathcal{N}_m(Hx|\mathcal{D}_n, \Sigma_{\text{obs}}) + \log \mathcal{N}_D(x|\mu_n, \Sigma),$$

where $\mu_n := \mu + \epsilon_n$ and $\mathcal{D}_n := \mathcal{D} + \eta_n$, with $\epsilon_n \sim \mathcal{N}(0, \Sigma)$ and $\eta_n \sim \mathcal{N}(0, \Sigma_{\text{obs}})$. The maxima $\{x_n^\star\}_{n=1}^{n_{RML}}$ of these (log-concave) objective functions are the critical points $\nabla O_n(x_n^\star) = 0$. We thus differentiate with respect to x, which gives

$$\nabla O_n(x) = 2\Sigma^{-1}x - 2\Sigma^{-1}\mu_n + 2H^\top \Sigma_{\text{obs}}^{-1} Hx - 2H^\top \Sigma_{\text{obs}}^{-1}\mathcal{D}_n$$

and setting this equal to zero and rearranging gives

$$\begin{aligned} x &= (\Sigma^{-1} + H^\top \Sigma_{\text{obs}}^{-1} H)^{-1}(\Sigma^{-1}\mu_n + H^\top \Sigma_{\text{obs}}^{-1}\mathcal{D}_n) \\ &= m + V(\Sigma^{-1}\epsilon_n + H^\top \Sigma_{\text{obs}}^{-1}\eta_n). \end{aligned}$$

The distribution of x can then easily be seen to be

$$\begin{aligned} x &\sim \mathcal{N}(m, V(\Sigma^{-1}\Sigma\Sigma^{-1} + H^\top \Sigma_{\text{obs}}^{-1}\Sigma_{\text{obs}}\Sigma_{\text{obs}}^{-1} H)V^\top) \\ &= \mathcal{N}(m, V), \end{aligned}$$

i.e., x is a sample from the true posterior distribution.

Some of the practical success in a variety of petroleum engineering applications [55, 46] can be associated with the “weakly nonlinear" nature of the simulators used [152, 49]. Additional care needs to be taken in more challenging scenarios, such as multi-modal posteriors with highly nonlinear simulators, and there is a series of works which addresses these challenges both theoretically and empirically [10, 123, 7]. Nonetheless, good practical performance has been observed for nonlinear (deep) neural network parametrized simulators in a series of more recent petroleum engineering publications [104, 161].

Instead of focusing on the accuracy of the approximate samples $\{x_n^\star\}_{n=1}^{n_{RML}}$ with respect to the true posterior, we address solving the optimization problems (52) efficiently in the challenging case of a high-dimensional input space $\mathbb{R}^D$. We focus on the specific scenario from Chapter 4, where the log-likelihood (48)

$$\log p(\mathcal{D}|x) \propto L(x) := -||\mathcal{D} - f(x)||^2_{\Sigma_{\text{obs}}}$$

has a low-dimensional active subspace. Although a gradient-based active subspace $\hat{W}_{AS}$ (16) exists, we assume that we do not have access to $\hat{W}_{AS}$, and moreover, we do not have sufficient budget to estimate it from likelihood evaluations $\{x_i, L(x_i)\}_{i=1}^N$, as we did in Chapter 4. Furthermore, even though the log-likelihood has a low-dimensional active subspace, an additional difficulty comes from the fact that the prior might not have such a structure, for example when $x \sim \mathcal{N}(0, I_D)$. Indeed, we saw in the previous chapters that $x \sim \mathcal{N}(0, I_D)$ is a common prior in applied problems, as it can be used to generate a Gaussian random field (GRF) prior (21) for the coefficients $\log a_x$ of a PDE simulator $f(x)$. Note that sometimes the goal of the Bayesian inverse problem is to use the samples $x \sim p(x|\mathcal{D})$ to obtain posterior samples for the PDE coefficients $\log a_x$ via (21). One such example comes from geological applications, where a_x can represent a permeability field to be recovered from the observed measurements $\mathcal{D}$ [86].

If we ignore the prior and the multi-objective nature of our problem for now, the task of maximizing an objective $O(x) \approx \hat{g}(\hat{W}_{AS}^T x)$ (in this case, the log-likelihood) under the tight budget assumption mentioned above is common in the high-dimensional Bayesian Optimization (HD-BO) literature, which is key to our work.

Bayesian Optimization (BO, i.e. finding $\arg\max_x O(x)$ using a Gaussian process approximation $g_{\mathrm{GP}}(x) \approx O(x)$ (3)) is based on a standard exploration-exploitation principle. Namely, an acquisition function based on $g_{\mathrm{GP}}(x)$ is used such that in the exploration phase, the target function $O(x)$ is explored globally, whereas in the exploitation phase, points $\tilde{x}$ that are likely to satisfy $\tilde{x} = \arg\max_x O(x)$ are sampled until the maximum is eventually found. We will introduce the acquisition function used in our computer experiments later in the chapter. See [52] for a comprehensive introduction to BO, and [180] for a discussion on the various choices of acquisition functions available and their use in complex BO settings. The high-dimensional Bayesian Optimization (HD-BO) literature mostly deals with the prevalent case where the approximation $g_{\mathrm{GP}}(x) \approx O(x)$ is unsatisfactory, as a result of the curse of dimensionality (Section 1.5). In our setting where an active subspace exists but is unknown, the most common solution is the use of random embeddings, $R \in \mathbb{R}^{D \times d_e}$, instead of the true low-dimensional embedding $\hat{W}_{AS} \in \mathbb{R}^{D \times d}$; see [171] for the seminal work on this topic. Here we abuse notation, and use R to denote both the random subspace and the $D \times d_e$ projection matrix from $\mathbb{R}^D$ into that space.

Algorithm 4 presents a generic HD-BO algorithm with random embeddings. The random embedding, R, transforms the original high-dimensional BO problem $O(x) \approx g_{\mathrm{GP}}(x)$ for $x \in \mathbb{R}^D$ into a low-dimensional BO problem $O(Ry) \approx \tilde{g}_{\mathrm{GP}}(y)$ for $y \in \mathbb{R}^{d_e}$. In other words, instead of trying to maximize $O(x)$, we try to maximize $O(Ry)$, which is the objective function on the subspace R.

Algorithm 4 Generic HD-BO with random embeddings

```
$M$ : number of evaluations of $O(\cdot)$ possible given the computational budget;
$d_e$ : chosen dimensionality of the embedding $R$;
$R \in \mathbb{R}^{D \times d_e}$ : random embedding;
$m_0$ : initial training points $\{y_i, O(Ry_i)\}_{i=1}^{m_0}$, with $y_i \in \mathbb{R}^{d_e}$
for $m \in \{m_0 + 1, \dots, M\}$ do
    1. Construct a GP approximation $O(Ry) \approx \tilde{g}_{\mathrm{GP}}(y)$ using the available objective function evaluations $\{y_i, O(Ry_i)\}_{i=1}^{m-1}$
    2. Select $y_m = \arg\max_y a_m(y)$ as the maximizer of a BO acquisition function for the GP approximation $O(Ry) \approx \tilde{g}_{\mathrm{GP}}(y)$
    3. Update the training data to $\{y_i, O(Ry_i)\}_{i=1}^{m}$.
end for
Obtain $x_\star = Ry_{m_\star}$ as the maximizer
    $$m_\star = \arg\max_m O(Ry_m), \;\; m \leq M.$$
```

In practice, we can use multiple random projections giving maximizers $x_\star^1, \dots, x_\star^K$, and select the quantity $x_\star := \arg\max_{x_\star^k} O(x_\star^k)$ that gives the largest objective function value [171].

To summarize our contributions, in this work we

- propose a new methodology for posterior sampling via HD-BO (Algorithm 5);
- propose a natural way to exploit the shared simulator $f(x)$ that is present in all of the objective functions (52), as well as an adjustment needed to incorporate a prior distribution without a low-dimensional structure;
- show that in the limited budget setting, our methodology usually outperforms alternative gradient-free optimization methods, demonstrating this in a series of synthetic and real-world experiments;

- visualize the posterior distribution in the active subspace [31], together with the samples produced by our methodology; we show that the samples are indeed close to 'true' RML samples (collected via an unlimited computational budget), while also covering well the high posterior density regions.

Although there is an extensive literature for HD-BO with random embeddings, which offers theoretical guarantees and good practical performance (see, e.g., [171, 118, 99], the last including a recent survey on the topic), there is no methodology designed for posterior sampling. Yet RML is a natural way to use HD-BO for high-dimensional posterior sampling.

The closest related work to ours is [78], where BO is compared with alternative gradient-free optimization methods for maximizing the log-likelihood in a variety of low-dimensional experiments related to petroleum engineering. We extend their work by considering the multi-objective setting (n_{RML} randomized log-likelihoods) in order to do posterior sampling, as well as considering the challenging high-dimensional case and incorporating high-dimensional priors.

Another methodology related to RML for simulators with stochastic outputs is reverse sampling [50] and Optimization Monte Carlo (OMC) [112]. In particular, in Algorithm 3 in the Robust OMC algorithm of [88], a GP model is used for a collection of randomized objectives $R_j(x)$ maximized with Bayesian Optimization. The GP approximations are used as a sampling tool at every iteration. It would be interesting to see if these ideas can be applied in the deterministic simulator setting considered here.

5.2 METHODOLOGY

Firstly, we consider the simpler setting of a uniform prior, $x \sim U[a_i, b_i]_{i=1}^D$, where $[a_i, b_i]_{i=1}^D := [a_1, b_1] \times \cdots \times [a_D, b_D]$. In this case, the posterior distribution is proportional to the likelihood, and using our active subspace assumption the RML objective functions (52) become

$$O_n(x) = L_n(x) := \log \mathcal{N}_m(f(x)|\mathcal{D}_n, \Sigma_{\text{obs}}) \approx \hat{g}_n((\hat{W}_{AS}^{(n)})^T x), \tag{53}$$

where the objective functions now need to be maximized over the prior support $[a_i, b_i]_{i=1}^D$.

A naive strategy would be to perform HD-BO (see Algorithm 4) independently for each objective function (53), i.e., train GP approximations

$$g_{\text{GP}}^{(n)}(y) \approx O_n(Ry), \text{ for } n \in [n_{RML}]. \tag{54}$$

Assuming that each HD-BO procedure is run over T iterations, this strategy will result in a collection of Tn_{RML} simulations $\{(y_t^n, f(Ry_t^n))\}$ for $t \in [T]$ and $n \in [n_{RML}]$. Note that the limited budget, N, constrains us to $T \leq N/n_{RML}$ iterations for each objective. However, note that every data point $\{(y_t^n, f(Ry_t^n))\}$ can be shared by all the objectives (53). Since all the objective functions $O_n(x)$ have similar structure and are based on the same underlying simulator $f(x)$, we expect that the exploration stage can be performed at once for all objective functions, which can potentially lead to faster HD-BO convergence. For example, we could run HD-BO for $O_1(x)$ (T_1 iterations say), and then reuse the training data $\{(y_t^1, f(Ry_t^1))\}$ to warm-start/speed-up convergence for $O_2(x)$ (where hopefully a much smaller number of iterations $T_2 \ll T_1$ will be required). Whilst this is an attractive strategy, it poses the difficulty of having to choose a stopping time T_n for every objective.

To circumvent the challenges mentioned above, we propose our procedure in Algorithm 5, and a schematic representation of the first two iterations is shown in Figure 46. As suggested by [171], we use a collection of K interleaved random embeddings instead of a single random embedding for the entire procedure. Indeed, during our computer experiments, we notice that some embeddings get stuck in the exploitation phase and thus select samples from the same very small region of the posterior space for all the RML objectives.

Algorithm 5 HD-BO-RML (additional steps for Gaussian priors are shown in brackets)

```
$N$ : max possible number of evaluations of $f(\cdot)$;
$d_e$ : choice of embedding dimensionality;
$R_1, \dots, R_K \in \mathbb{R}^{D \times d_e}$ : collection of random embeddings (see Section 5.3);
$n_0 \times K$ initial points: $\{y_i^k, f(R_k y_i^k)\}_{i=1}^{n_0}$, with $y_i^k \in \mathbb{R}^{d_e}$, $k \in [K]$;
**for** $k \in \{1, \dots, K\}$ **do**
    **for** $n \in \{n_0 + 1, \dots, \lfloor N/K \rfloor\}$ ($n \in \{n_0 + 1, \dots, \lfloor N/2K \rfloor\}$ in case of a Gaussian prior) **do**
        1. Let $n' := n \mod n_{RML}$
        2. Construct a GP approximation to $L_{n'}(R_k y)$ using simulations $\{y_i^k, f(R_k y_i^k)\}_{i=1}^{n-1}$
        3. Select $y_n^k = \arg\max_y a_n^k(y)$ as the maximizer of a BO acquisition function using the GP approximation
        4. Perform $f(R_k y_n^k)$ and update the shared simulation ensemble to $\{y_i^k, f(R_k y_i^k)\}_{i=1}^{n}$
        (Gaussian prior) 5G. Perform local optimization in $\mathbb{R}^D$ around $x_0 = R_k y_n^k$ with respect to the prior $p_{n'}(x)$ (see Step 1G. below)
    **end for**
**end for**
**for** $n \in \{1, \dots, n_{RML}\}$ **do**
    (Uniform prior) 1U. Obtain $x_n^\star = R_{k_\star} y_{m_\star}^{k_\star}$ as the maximizer
        $$k_\star, m_\star = \arg\max_{k,m} O_n(R_k y_m^k),\ k \in [K], m \leq \lfloor N/K \rfloor$$
    (Gaussian prior) 1G. Writing $z_m^k$ for the local maxima from step 5G. above, obtain $x_n^\star = z_{m_\star}^{k_\star}$ via
        $$k_\star, m_\star = \arg\max_{k,m} O_n(z_m^k),\ k \in [K], m \leq \lfloor N/2K \rfloor$$
**end for**
```

The algorithm is based on a cyclic pass through all the objective functions (53), where for every random embedding $k \in [K]$, the simulations

$$\begin{array}{cc} & \begin{matrix} 1 & \dots & n & n+1 \end{matrix} \\ \begin{matrix} O_1(y) \\ O_2(y) \\ \vdots \\ O_M(y) \end{matrix} & \begin{pmatrix} y_1, O_1(y_1) & \vdots & y_n, O_1(y_n) & y_{n+1} \\ y_1, O_2(y_1) & \vdots & y_n, O_2(y_n) & \\ \vdots & \vdots & \vdots & \\ y_1, O_M(y_1) & \vdots & y_n, O_M(y_n) & \end{pmatrix} \end{array} \rightarrow \begin{array}{c} \begin{matrix} 1 & \dots & n & n+1 \end{matrix} \\ \begin{pmatrix} y_1, O_1(y_1) & \vdots & y_n, O_1(y_n) & y_{n+1}, O_1(y_{n+1}) \\ y_1, O_2(y_1) & \vdots & y_n, O_2(y_n) & y_{n+1}, O_2(y_{n+1}) \\ \vdots & \vdots & \vdots & \vdots \\ y_1, O_M(y_1) & \vdots & y_n, O_M(y_n) & y_{n+1}, O_M(y_{n+1}) \end{pmatrix} \end{array}$$

$$\begin{array}{c} \begin{matrix} 1 & \dots & n+1 & n+2 \end{matrix} \\ \begin{pmatrix} y_1, O_1(y_1) & \vdots & y_{n+1}, O_1(y_{n+1}) & \\ y_1, O_2(y_1) & \vdots & y_{n+1}, O_2(y_{n+1}) & y_{n+2} \\ \vdots & \vdots & \vdots & \\ y_1, O_M(y_1) & \vdots & y_{n+1}, O_M(y_{n+1}) & \end{pmatrix} \end{array} \rightarrow \begin{array}{c} \begin{matrix} 1 & \dots & n+1 & n+2 \end{matrix} \\ \begin{pmatrix} y_1, O_1(y_1) & \vdots & y_{n+1}, O_1(y_{n+1}) & y_{n+2}, O_1(y_{n+2}) \\ y_1, O_2(y_1) & \vdots & y_{n+1}, O_2(y_{n+1}) & y_{n+2}, O_2(y_{n+2}) \\ \vdots & \vdots & \vdots & \vdots \\ y_1, O_M(y_1) & \vdots & y_{n+1}, O_M(y_{n+1}) & y_{n+2}, O_M(y_{n+2}) \end{pmatrix} \end{array}$$

Figure 46: To condense notation, we write $n := n_0$ for the initial points and $M := n_{RML}$ for the RML objectives. Also, we suppress the superscript k corresponding to the random embedding R_k, i.e., $y_p := y_p^k$ and $O_m(y) := O_m(R_k y)$ for $p \in [n]$ and $m \in [M]$. From top to bottom, according to the arrows: first HD-BO iteration to collect $y_{n+1} := y_{n+1}^k = \arg\max_y a_{n+1}^k(y)$ from the GP approximation of $O_1(y)$, then perform $f(R_k y_{n+1}^k)$ which generates data points $y_{n+1}, O_m(y_{n+1}) := y_{n+1}^k, O_m(R_k y_{n+1}^k)$ for all $m \in [M]$; second HD-BO iteration to collect $y_{n+2} := y_{n+2}^k = \arg\max_y a_{n+2}^k(y)$ from the GP approximation of $O_2(y)$, then perform $f(R_k y_{n+2}^k)$ which generates data points $y_{n+2}, O_m(y_{n+2}) := y_{n+2}^k, O_m(R_k y_{n+2}^k)$ for all $m \in [M]$.

$\{(y_m^k, f(R_k y_m^k))\}_{m=1}^n$ collected up to some iteration n will serve as the basis for the training set used in the next iteration, where the objective function $O_{(n+1)'}(x)$ is considered, with $(n+1)' := n+1 \mod n_{RML}$.

As discussed above, we expect that the exploration stage will be performed simultaneously for all objective functions, as opposed to n_{RML} times in the naive strategy discussed in the context of (54). When reaching the exploitation phase for $O_{n'}(x)$, it is likely that $O_{(n+1)'}(x)$ will also benefit from the shared basis for the training data $\{(y_m^k, f(R_k y_m^k))\}_{m=1}^{n-1}$ with $O_{n'}(x)$ and enter the exploitation phase. Also, due to the cyclic pass through all the objectives, we sidestep the difficulty of having to choose a stopping time T_n for every objective. Finally, we select the HD-BO-RML samples $x_n^\star$ for $n \in [n_{RML}]$ to be the maximizers of the n_{RML} objective functions (53) out of all the points selected by our procedure.

We now address the case of a Gaussian prior distribution, $x \sim \mathcal{N}_D(\mu, \Sigma)$. Using again the active subspace assumption, the objective functions (52) become:

$$O_n(x) = L_n(x) + \log p_n(x) \approx \hat{g}_n((\hat{W}_{AS}^{(n)})^T x) + \log \mathcal{N}_D(x|\mu_n, \Sigma).$$

Due to the potential lack of a low-dimensional linear structure in the prior, as for example if $x \sim \mathcal{N}(0, I_D)$, running HD-BO by directly modelling the objective function on a random linear subspace $O_n(Ry)$ as a GP might be unsatisfactory, and hence we cannot simply use the algorithm from the uniform prior case. Nonetheless, if we try to ignore the prior and use Algorithm 5 for the log-likelihood alone (i.e. without step 5G), we can still incorporate the prior in the final step: $k_\star, m_\star = \arg\max_{k,m} O_n(R_k y_m^k),\ k \in [K], m \leq \lfloor N/K \rfloor$. This can lead to two obvious problems: we select either points of high-likelihood but low-prior (from the exploitation stage), or points of low-likelihood but reasonably high-prior (from the exploration stage). In our practical experiments, we encountered the second problem.

Steps 5G and 1G in Algorithm 5 propose an ad-hoc fix to the issue mentioned above. While we keep performing HD-BO with respect to the log-likelihood as in the uniform prior case, we try to increase the prior value for the selected points of potentially high-likelihood $x_0 = R_k y_n^k$ by performing local optimization in the high-dimensional space $\mathbb{R}^D$, starting from x_0; during the exploitation stage, the resulting solutions z_n^k try to achieve a trade-off between high-likelihood and high-prior. Note that the local maximization of $p_{n'}(x) \sim \mathcal{N}_D(\mu_{n'}, I_D)$ ($n' := n \mod n_{RML}$) does not require simulator evaluations and can be performed efficiently by using fast gradient-based optimization methods. For the experiments presented here, we have restricted the number of steps performed by the global optimization routine CMA-ES [80], which is a gradient-free evolution strategy algorithm.

As mentioned above, the CMA-ES algorithm starts at the point selected by maximizing the HD-BO acquisition function $x_0 = R_k y_n^k$; we have chosen a standard deviation (step-size) of 0.1 for CMA-ES. Note that in our experiments $x \sim \mathcal{N}(0, I_D)$, so the CMA-ES standard deviation for the local maximization of $p_{n'}(x) \sim \mathcal{N}_D(\mu_{n'}, I_D)$ around x_0 is 10 times smaller than the prior standard deviation. We have also restricted the number of function evaluations $\{x_i, p_{n'}(x_i)\}_{i=1}^{N_{max}}$ performed by CMA-ES to a maximum of $N_{max} = 500$, in order to prevent the points being too far away from x_0, which would lead to a severe decrease in the log-likelihood during the exploitation phase.

5.3 EXPERIMENTAL SETUP

We now give empirical results showing that the proposed algorithms perform well in comparison with competing methods in a variety of synthetic and real-world Bayesian inverse problems. All the experiments use simulators that can be found on the Active Subspaces github page [27]. Unless otherwise stated, we use a uniform prior distribution $p(x)$ for all parameters, x_{true} is sampled from $p(x)$, and the measurements $\mathcal{D} = f(x_{true}) + \epsilon$ include additive noise with standard deviation that is 5% of the signal ($\sigma = 5\% \cdot ||f(x_{true})||_2$). For some of the experiments, parameters sampled from the prior $x_{true} \sim p(x)$ lead to uninformative data, i.e., the posterior $p(x|\mathcal{D})$ is very similar to the uniform prior $p(x)$, and in this case the optimization landscape for RML is uninteresting. To avoid this, we selected x_{true} samples that led to a significant difference between the prior and posterior. We present results for four simulators:

- Elliptic-PDE simulator $f : \mathbb{R}^{100} \to \mathbb{R}^7$ with measurements noise level of 1% and standard Gaussian prior $x \sim \mathcal{N}_{100}(0, I)$ [31]. This simulator was also considered in Chapter 4.
- Ebola: $f : \mathbb{R}^8 \to \mathbb{R}$, an 8-parameter dynamical system model for the geographic spread of Ebola in Liberia [44].
- MHD: $f : \mathbb{R}^5 \to \mathbb{R}$, a 5-parameter magnetohydrodynamics power generation model [71].
- HIV long-term model: $f : \mathbb{R}^{27} \to \mathbb{R}$ is the cell count at time $t = 24$ [105]; the log likelihood has a one-dimensional active subspace, $L(x) \approx \hat{g}(\hat{W}_{AS}^T x)$ with $g : \mathbb{R} \to \mathbb{R}$. This simulator was also considered in Chapter 3.

In each case (except for the HIV long-term model), the resulting log-likelihood has been shown to have a two-dimensional active subspace, $L(x) \approx \hat{g}(\hat{W}_{AS}^T x)$ with $g : \mathbb{R}^2 \to \mathbb{R}$. Note that we do not assume knowledge of $\hat{W}_{AS}$.

For each experiment, we use RML to sample from the posterior distribution, comparing our high dimensional Bayesian optimization (HD-BO) approach from Algorithm 5 against the following alternative gradient-free optimization methods: BOBYQA [132] (which is a trust-region optimization algorithm, used for method comparison versus HD-BO in [47]), CMA-ES [80] (which is used for method comparison versus HD-BO in [47, 99]), and random design [171] (which is often used as a standard baseline for method comparison in the BO literature). We measure performance of the different optimization methods by comparing the mean return, i.e., the method with the highest mean return can be considered to have performed best, where the mean return is defined as

$$\frac{1}{n_{RML}} \sum_{n=1}^{n_{RML}} O_n(x_n^\star), \tag{55}$$

where $x_n^\star$ is the approximate maximizer of $O_n(x)$ as selected by the different methods considered.

All the experiments use the Gaussian Process Upper Confidence Bound (GP-UCB) acquisition function [155], i.e., $a_n^k(y) = \bar{m}_n^k(y) + \beta\sqrt{\bar{k}_n^k(y,y)}$, where $\bar{m}_n^k(y)$ is the GP predictive mean which tries to approximate the log-likelihood $L_{n'}(R_k y)$ ($n' := n \mod n_{RML}$), and $\sqrt{\bar{k}_n^k(y,y)}$ is the GP posterior standard deviation (i.e. the GP predictive uncertainty). This acquisition function has been studied in the HD-BO literature with random embeddings [171], and it was the first acquisition function for which theoretical guarantees were proven in the BO literature [155]. We choose $\beta = 3$ for all experiments as a heuristic to facilitate a more exploratory behaviour, as the objective functions change iteratively from $O_{n'}(y)$ to $O_{(n+1)'}(y)$, but more principled approaches for choosing β from BO literature could also be used [156]. As it is the case throughout the thesis, we use a standard squared-exponential covariance function, i.e. $\mathbb{C}\mathrm{ov}(O(Ry_1), O(Ry_2)) := \sigma^2 \exp(-\frac{||y_1-y_2||_2^2}{2l^2})$ for each GP model, together with a constant GP prior mean $m(y) = C$. The GP hyperparameters are trained at every iteration via type II maximum likelihood optimization (5) with a standard LBFGS algorithm, performed for 5 steps with learning rate 0.1.

For the random embeddings R_k, each row is sampled independently from the uniform distribution on the unit hypersphere $\mathbb{S}^{d_e-1}$, as suggested in [16, 99] (if $r \sim \mathcal{N}(0, I_{d_e})$, then $r/||r||$ is uniformly distributed on $\mathbb{S}^{d_e-1}$). In the case of a uniform prior $x \sim U[a_i, b_i]_{i=1}^D$, we constrain the maximization of the acquisition function to make sure that $R_k y$ projects back to the correct prior box via the transformation $[a_i, b_i]_{i=1}^D \to [-1,1]^D$, followed by

$$\max_{y \in \mathbb{R}^{d_e}} a_n^k(y) \text{ subject to } -1 \leq (R_k y)_d \leq 1 \text{ for all } d \in [D], \tag{56}$$

as suggested in [99]; this constrained optimization is performed by SciPy's gradient-free COBYLA routine. In the experiments performed in [99], the unit hypershpere random embedding R_k lead to higher probability of containing the maximum $x_\star := \arg\max_x O(x)$ of a generic objective function $O(x)$ (i.e., $x_\star = R_k y_\star$ for $y_\star := \arg\max_y O(R_k y)$); this comparison was performed versus other random embeddings such as i.i.d. Gaussian (i.e., $(R_k)_{ij} \sim \mathcal{N}(0, 1)$ [171]) or the Count Sketch (CS) random embedding [118] discussed in Section 3.2.2. As presented in S4 of [99], the projection matrix as a hypersphere sample rounds out the vertices of the polytope (56) and expands the space to capture the optima of $O(x)$ (compared with the i.i.d. Gaussian $(R_k)_{ij} \sim \mathcal{N}(0, 1)$). More about the uniform distribution over hyperspheres can be found in e.g., [106].

We have used the unconstrained CMA-ES optimization algorithm to maximize the acquisition function in the case of a Gaussian prior. More concretely, we start at $\mathbf{0}_{d_e}$ and set up a standard deviation of $\sqrt{d_e}$ for the CMA-ES optimization search, motivated by the heuristic that $y \in [-\sqrt{d_e}, \sqrt{d_e}]^{d_e}$ is likely to project back to $R_k y \in [-1, 1]^D$, and our prior $x \sim N(0, I_D)$ has standard deviation 1. This heuristic was introduced by [171] for random embeddings R with independent $\mathcal{N}(0, 1)$ entries, but our unit hypersphere random embeddings R_k can be obtained by normalizing the rows of R, as described in the previous paragraph.

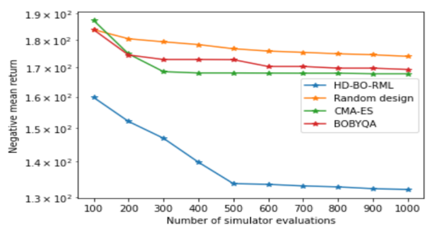
5.4 EXPERIMENTAL RESULTS

Table 8 gives the mean return (55) and its standard deviation over 5 optimization trials, and Figure 47 shows the (negative) mean returns for various budgets, averaged over the same 5 trials. Note that each experiment has a fixed set of measurements $\mathcal{D}$ and thus a fixed set of RML objectives (52); the variance over the 5 trials comes from different optimization runs (e.g. different starting points for BOBYQA, different random embeddings used for HD-BO-RML). Each trial has a budget of $N = 1000$ simulations in order

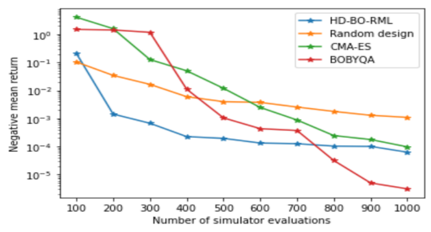
Table 8: Mean return (55), together with the standard deviation over 5 trials. In bold, we outline the best competing method, together with the methods that outperform the best method in at least one trial. The oracle results are obtained with an unlimited computational budget (i.e., unlimited number of simulations) and are not part of the competing methods.

	PDE	Ebola	MHD	HIV
Random	-173.9 ± 1.0	$-1.0 \cdot 10^{-3} \pm 5.5 \cdot 10^{-4}$	$-4.0 \cdot 10^{-3} \pm 3.8 \cdot 10^{-3}$	$\mathbf{-1.6 \cdot 10^{-5} \pm 3.9 \cdot 10^{-6}}$
BOBYQA	-169.3 ± 8.2	$\mathbf{-3.1 \cdot 10^{-6} \pm 2.8 \cdot 10^{-6}}$	$\mathbf{-2.6 \cdot 10^{-7} \pm 2.7 \cdot 10^{-7}}$	$-4.5 \cdot 10^{-2} \pm 7.3 \cdot 10^{-2}$
CMA-ES	-167.8 ± 5.0	$-9.7 \cdot 10^{-5} \pm 5.2 \cdot 10^{-5}$	$-3.3 \cdot 10^{-4} \pm 3.1 \cdot 10^{-4}$	$\mathbf{-6.7 \cdot 10^{-6} \pm 2.8 \cdot 10^{-6}}$
HD-BO-RML	**-132.2 ± 0.9**	$\mathbf{-6.3 \cdot 10^{-5} \pm 6.8 \cdot 10^{-5}}$	$\mathbf{-1.4 \cdot 10^{-5} \pm 1.9 \cdot 10^{-5}}$	$\mathbf{-1.4 \cdot 10^{-5} \pm 1.3 \cdot 10^{-5}}$
Maximum (Oracle)	-99.73	$-3.3 \cdot 10^{-15}$	0.0	$-4.6 \cdot 10^{-14}$

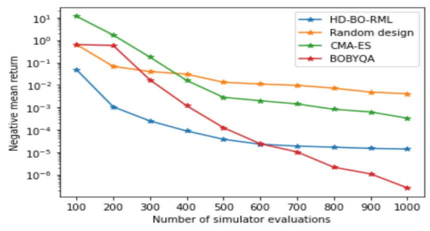
to find $n_{RML} = 20$ samples. As n_{RML} grows, HD-BO-RML has an increasing advantage over the other methods, and so we have chosen a deliberately small number of samples, towards the lower end of what might be considered a useful number of posterior samples, in order to demonstrate that even in simple problems there is an advantage in using our approach.



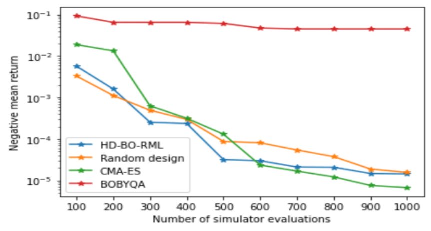
(a) Elliptic PDE



(b) Ebola



(c) MHD



(d) HIV

Figure 47: (Best viewed in colour) Averaged over the 5 trials presented in Table 8, we plot negative mean returns (55) (lower values are better). We do not plot the standard errors over the 5 trials, as they are not particularly insightful.

The random design samples N points from $p(x)$, and returns the points of largest mean return (55). We use the libraries Py-BOBYQA [20] and py-cma [80] for BOBYQA and CMA-ES, respectively, with default settings. Indeed, for the uniform prior experiments, we use the prior box constraints $[a_i, b_i]_{i=1}^D \rightarrow [-1, 1]^D$ for CMA-ES and BOBYQA optimization. We use the center 0_D of the (transformed) box $[-1, 1]^D$ as the starting point for CMA-ES, together with a standard deviation of $1/3$ such that 3 standard deviations cover the entire box. For BOBYQA, we use a starting point sampled at random from $[-1, 1]^D$. For the Gaussian prior experiment with $x \sim \mathcal{N}(0, I_D)$, we use the mean 0_D as the starting point for CMA-ES, together with a standard deviation of 1. For BOBYQA, we use a starting point sampled at random from a $\mathcal{N}(0, I_D)$ distribution.

Since BOBYQA and CMA-ES cannot share data between different objectives $O_i(x)$ and $O_j(x)$, unlike HD-BO-RML which shares data through the common simulator via the GP training sets, we employ BOBYQA and CMA-

ES independently for each objective $O_n(x)$ for $n \in [n_{RML}]$, using 50 simulator evaluations per objective to stay within budget. Similar to HD-BO-RML, we select the RML samples for BOBYQA and CMA-ES, respectively, to be the points of largest mean return (55) out of the $N = 1000$ simulator evaluations.

For HD-BO-RML (Algorithm 5) we use $K = 10$ random embeddings, sampled as described in the Section 5.3. We decided to choose a relatively large number of embeddings in order to encourage a good coverage of the high posterior density regions, as some random embeddings might get stuck in the exploitation phase and select samples from the same (small) region for all the RML objectives, as discussed in Section 5.2.

We choose $d_e = d = 2$ for the Gaussian prior (Elliptic-PDE) experiment, i.e., the dimension of the random embedding is the same as for the true active subspace, and $d_e = d + 1$ for all the other experiments, i.e., the dimension of the random embedding is slightly larger than the true active subspace. These choices are common in the HD-BO literature.

In the uniform prior experiments for HD-BO-RML, we select $n_0 = 5$ initial points $y \in \mathbb{R}^{d_e}$ from $[-1, 1]^{d_e}$, using rejection sampling to ensure that they project back to the (transformed) prior box $[a_i, b_i]_{i=1}^D \to [-1, 1]^D$ (i.e. $-1 \leq (R_k y)_d \leq 1$ for all $d \in [D]$), as suggested in [99]. For the Gaussian prior experiment with $x \sim \mathcal{N}(0, I_D)$, we select $n_0 = 5$ initial points $y \in \mathbb{R}^{d_e}$, sampled at random from a $\mathcal{N}(0, I_{d_e})$ distribution. As discussed in Section 5.3, using rejection sampling from $[-\sqrt{d_e}, \sqrt{d_e}]^{d_e}$ (uniform prior) or sampling from $\mathcal{N}(0, (\sqrt{d_e})^2 I_{d_e})$ (Gaussian prior) is generally more appropriate, as $y \in [-\sqrt{d_e}, \sqrt{d_e}]^{d_e}$ is likely to project back to $R_k y \in [-1, 1]^D$. Nonetheless, in the experiments considered here $d_e \in \{2, 3\}$ (thus $\sqrt{d_e}$ is close to 1), and so the difference should not be too significant (in the worst case scenario, our initial points for HD-BO-RML cover less of the prior space than the generally more appropriate design).

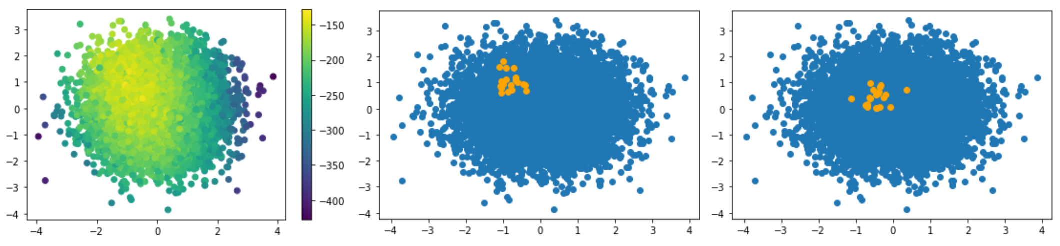
Our conclusions can be summarized as follows:

- **Elliptic-PDE simulator.** For this challenging high-dimensional experiment with a Gaussian prior, HD-BO-RML is the best performing method across the full range of computational budgets.
- **Ebola.** HD-BO-RML works best in the low budget regime $N \leq 500$, but BOBYQA has better performance for $N \geq 900$, although it is outperformed by HD-BO-RML in one of the trials.
- **MHD.** HD-BO-RML works best in the low budget regime $N \leq 400$, but BOBYQA is the best method at full budget, although it is outperformed by HD-BO-RML in one of the trials.
- **HIV.** All methods perform similarly, with a slight advantage for CMA-ES at full budget (although it was outperformed by HD-BO-RML in three trials and by the random design in one trial). BOBYQA often fails due to small budget with respect to the input dimensionality in this problem.

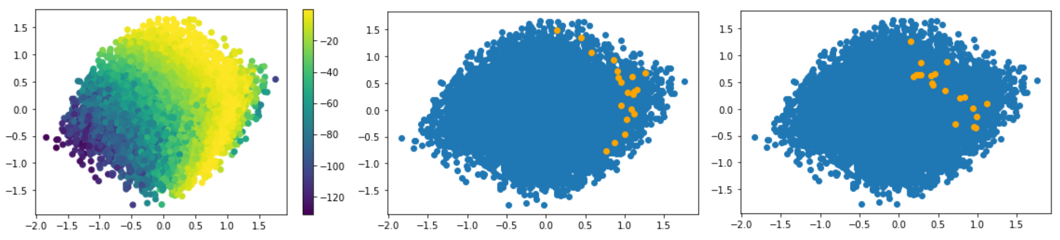
Note that there is a significant performance gap between the competing methods (which all have budget constraints) and the oracle results obtained by numerical optimization without constraints on the number of simulations available; we refer to the oracle results as the 'true' optimal mean returns. This gap can be potentially tightened by HD-BO-RML with a more involved GP design and training [99], or by a multi-output GP $\tilde{g}_{\text{GP}} : \mathbb{R}^{d_e} \to \mathbb{R}^{n_{RML}}$ which takes into account the correlation between all the objective functions $O(Ry) := (O_1(Ry), ..., O_{n_{RML}}(Ry)) \approx \tilde{g}_{\text{GP}}(y)$ [39]. With a fixed budget, the performance gap for HD-BO-RML should remain relatively constant as the number n_{RML} of RML samples required increases (since exploration/exploitation will still be performed jointly); the performance gap for CMA-ES and BOBYQA will significantly increase due to less iterations and no data sharing between objectives.

5.5 VISUALIZATION OF POSTERIOR SAMPLES

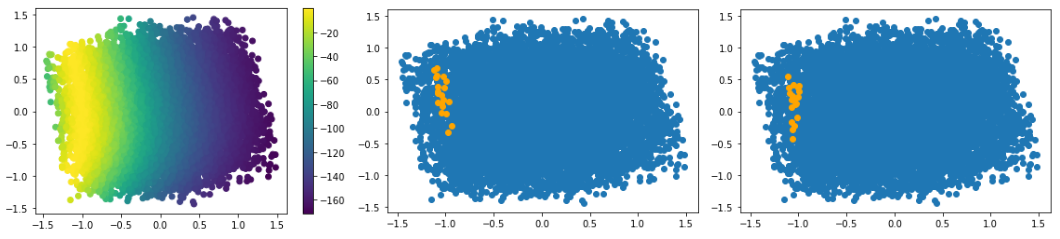
As in [31], we can visualize each posterior landscape $p(x|\mathcal{D})$ in the active subspace. Figures 48a, 48d, 48g, and 48j show 10^4 independent prior samples $x_i \sim p(x)$ projected into the true two-dimensional (for Figs. 48a, 48d, 48g) and one-dimensional (Figure 48j) active subspace, coloured by their unnormalized log-posterior density, i.e., $\{\hat{W}_{AS}^T x_i, \log p(\mathcal{D}|x_i) + \log p(x_i)\}_{i=1}^{10^4}$. Indeed, points that have similar posterior densities are close in the active subspace representation. We expect that the $n_{RML} = 20$ RML samples will be situated in the high posterior density regions. We use our oracle optimizer with access to an unlimited amount of simulator evaluations to obtain 'true' RML samples. The resulting samples (projected into the true active subspace) are shown in Figures 48b, 48e, 48h, and 48k. We see that samples cover well the high posterior density regions in all the experiments, according to the active subspace representation. While RML has been mostly used in synthetic experiments and petroleum engineering applications, this observation attests to its usefulness in a variety of real-world problems.



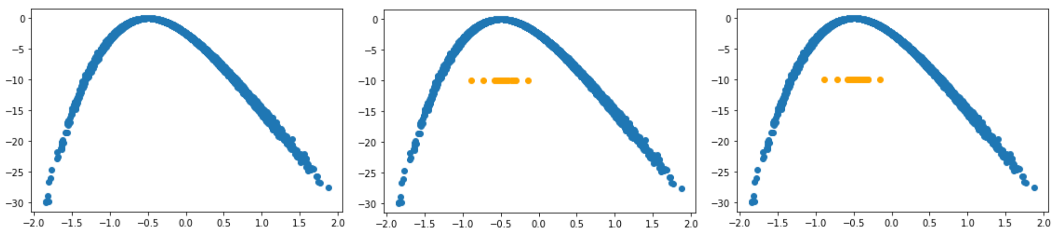
(a) PDE (b) RML oracle (c) HDBO-RML



(d) Ebola (e) RML oracle (f) HDBO-RML



(g) MHD (h) RML oracle (i) HDBO-RML



(j) HIV posterior (k) RML oracle (l) HDBO-RML

Figure 48: (Best viewed in colour) Posterior landscape in the active subspace (left), oracle RML samples (middle), and RML samples obtained by our procedure (right). The RML samples are shown in orange, with prior samples given in blue. For better visualization in the HIV experiment with a 1D active subspace, the RML samples from the x-axis are shifted towards the middle of the HIV plots.

In Figures 48c, 48f, 48i, and 48l, we show the analogue samples obtained from using HD-BO-RML instead of the oracle optimizer, averaged over the

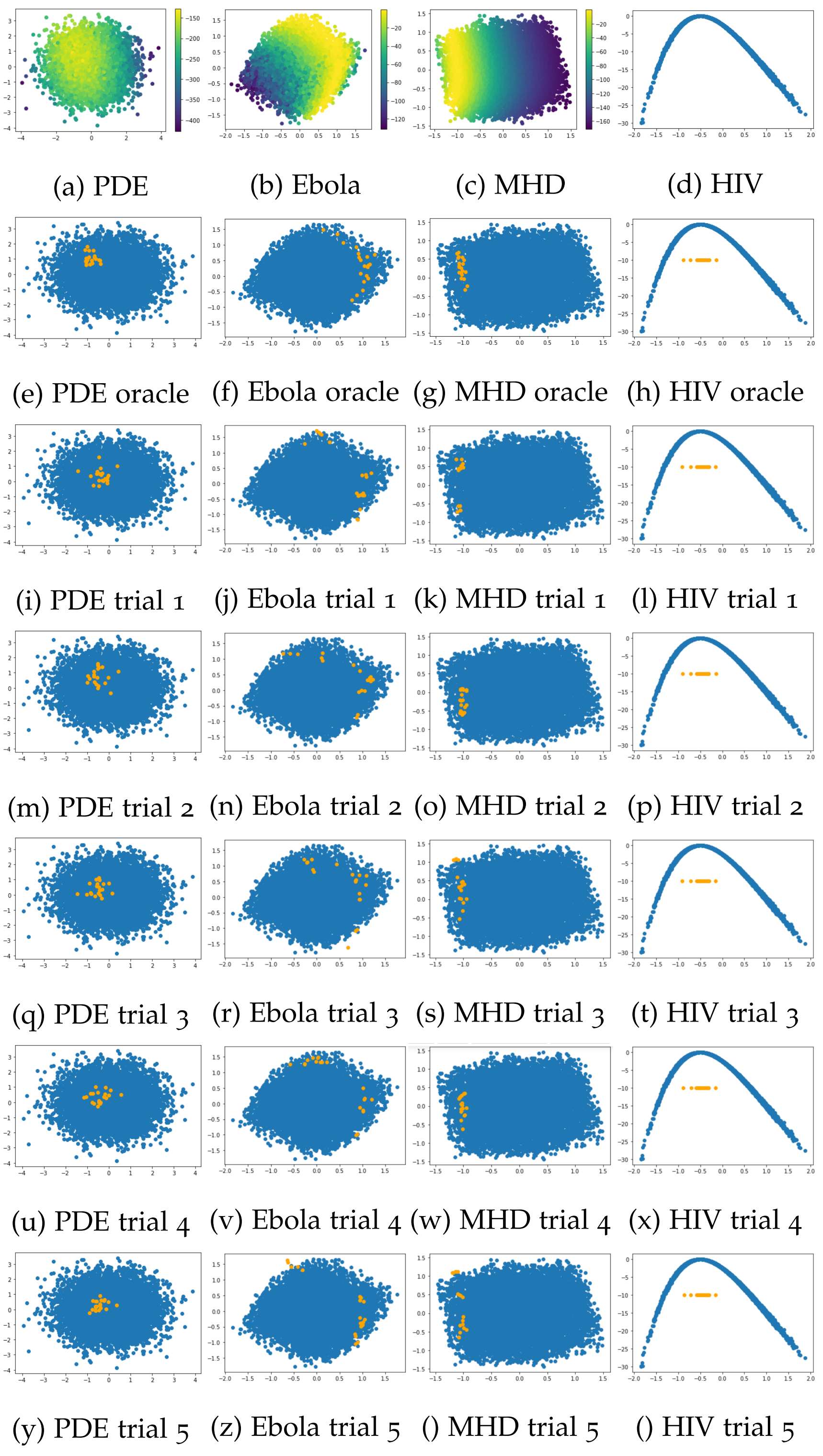
5 trials presented in Table 8. While we note a slight difference with the 'true' RML samples, as suggested by the mean return gap from Table 8, we see that the HD-BO-RML samples are nonetheless relatively close to the oracle samples in the active subspace representation, and still cover the high posterior density regions well. For the HIV experiment with a one-dimensional active subspsace, the HD-BO-RML samples seem to correspond exactly to the 'true' RML samples, according to the active subspace representation. Finally, whilst there is some variance between samples from different trials due to the different random embeddings used, the HD-BO-RML samples belong to the high posterior density regions in all the trials (Figure 49).

5.6 CONCLUSION AND FUTURE WORK

We propose HD-BO-RML, in which we use the HD-BO machinery to tackle the multi-objective RML methodology for posterior sampling. To the best of our knowledge, this is the first GP based approach for high-dimensional posterior sampling in the widespread setting of log-likelihoods with an active subspace, dealing with the challenging case where we do not have access to the active subspace nor have enough computational budget to estimate it. As demonstrated in the experiments arising from various domains, the methodology can outperform alternative gradient-free optimization methods and has potential to be used in many real-world applications.

To demonstrate the potential of HD-BO-RML, we presented a vanilla version using default choices of embeddings, GP approximations, and acquisition function etc. By specializing these aspects, further performance gains can be made. In addition to the suggestions in the main text, which mostly refer to developing a more accurate HD-BO procedure, we could also look at removing the active subspace assumption by replacing the random embeddings with an alternative HD-BO methodology that does not require this assumption, such as trust region BO (TuRBO) [47], which uses local GP models in high-dimensions. In this regard, a newly introduced multi-objective

HD-BO methodology can be found in [42], although the experiments only consider at most 4 objective functions. Moreover, the procedure from Algorithm 5 for combining BO with local optimization when using non-uniform priors could be replaced by the more principled approach from [111]. Finally, it would be interesting to extend the findings of [39] regarding multi-objective BO with multi-output GPs to our methodology and to the HD-BO setting in general.


(a) PDE (b) Ebola (c) MHD (d) HIV

(e) PDE oracle (f) Ebola oracle (g) MHD oracle (h) HIV oracle

(i) PDE trial 1 (j) Ebola trial 1 (k) MHD trial 1 (l) HIV trial 1

(m) PDE trial 2 (n) Ebola trial 2 (o) MHD trial 2 (p) HIV trial 2

(q) PDE trial 3 (r) Ebola trial 3 (s) MHD trial 3 (t) HIV trial 3

(u) PDE trial 4 (v) Ebola trial 4 (w) MHD trial 4 (x) HIV trial 4

(y) PDE trial 5 (z) Ebola trial 5 () MHD trial 5 () HIV trial 5

Figure 49: (Best viewed in colour) Each column gives results for one simulator. The rows are: the approximate posterior landscape in the true active subspace (first row); the oracle samples obtained via RML with unlimited computational budget (second row); samples from HD-BO-RML for the 5 trials presented in Table 8 (third to seventh row). The oracle RML and HD-BO-RML samples are shown in orange, with prior samples given in blue.

6 PROBABILISTIC MACHINE LEARNING WITHIN ENSEMBLE KALMAN INVERSION METHODS

So far, we have considered MCMC and RML as tools for solving Bayesian inverse problems. In this chapter, we will introduce the Ensemble Kalman Inversion framework (EKI). As discussed in Section 1.2, EKI are a family of methods which try to recover the unknown parameter in the classical inverse problem sense (i.e., x_{true} which generated the observations via $\mathcal{D} \sim \mathcal{N}_m(f(x_{true}), \Sigma_{\text{obs}}))$, although they are motivated from a Bayesian perspective as an iterative update from the prior distribution towards the posterior [87, 82].

In Section 6.1, we have developed a new EKI framework based on a (Bayesian) Neural Network (BNN) parametrization of Gaussian random field priors $p(x)$ for inverse problems $\mathcal{D} = f(x) + \epsilon$ [143]; this is an alternative for the classical Karhunen-Loève decomposition (9). For a recent work that analyzes various EKI parameterizations for Gaussian random field priors, see [22]. In Subsection 6.1.3, we present promising results on a two-phase (oil-water) Darcy flow simulator regarding permeability inversion. This is a standard inverse problem in petroleum engineering [46, 83]; the Darcy flow simulator is another example of an Elliptic PDE, in addition to the one considered thus far in the thesis (see e.g. Section 2.4).

In Section 6.2, we have investigated the performance of Gaussian Process emulators as computationally fast replacements for the expensive forward simulator within EKI. We complement the original work for GP emulators within EKI [91] with the state-of-the art EKI algorithm EKI-DMC [82], which we briefly introduce in Subsection 6.2.1. Furthermore, we propose one potential improvement for the selection of GP training points within EKI in Subsection 6.2.2, where we adapt a new experimental design [59] from the family of inverse problems methods known as History Matching [168, 175]. We demonstrate the benefits of our methodology on a steady-state groundwater modelling inverse problem in Subsection 6.2.3.

6.1 NEURAL NETWORK PRIOR FOR ENSEMBLE KALMAN INVERSION (EKI)

This section is organized as follows. In Subsection 6.1.1, we will introduce the (Bayesian) Neural Network parametrization of Gaussian random field priors from [143]. In subsection 6.1.2, we will introduce the Ensemble Kalman Inversion algorithm known as ensemble smoother with multiple data assimilation (ES-MDA [45]). Finally, in Subsection 6.1.3, we will combine these methods in a collection of computer experiments for a two-phase (oil-water) Darcy flow simulator regarding permeability inversion.

6.1.1 *Bayesian neural networks (BNN)*

In the Elliptic PDE experiments from the previous chapters, we have considered a Gaussian random field (GRF) prior for the 2D PDE coefficients $\log a_x : [0,1]^2 \to \mathbb{R}$ through the Karhunen-Loève decomposition (22), which leads to a Gaussian prior measure on the space of square-integrable functions $L^2([0,1]^2, \mathbb{R})$. In this subsection, we will consider an alternative Gaussian prior measure on $L^2([0,1]^2, \mathbb{R})$, which is based on Bayesian (deep) neural networks (BNN) and was recently introduced in [143]. The original work [143] shows that this new BNN construction leads to a valid prior for a Bayesian inverse problem with a 2D Elliptic PDE simulator regarding groundwater flow modelling [13].

Recall from Section 4.4 that we can write a deep neural network model $g_{NN}(y) : [0,1]^2 \to \mathbb{R}$ with L hidden layers as:

$$g_{NN}(y) = \phi(W_{L+1}^T \phi \ldots \phi(W_2^T \phi(W_1^T y + b_1) + b_2)) + b_{L+1}. \tag{57}$$

We write $W_l \in \mathbb{R}^{N_{l-1} \times N_l}$ and $b_l \in \mathbb{R}^{N_l}$ for $l \in \{1, \ldots, L+1\}$, i.e. N_l is the size of the l^{th} hidden layer ($N_0 = 2, N_{L+1} = 1$). Recall that ϕ is the (element-wise) activation function ($\phi := \tanh$ was used in [143]). According to [143], we transform g_{NN} into a Bayesian neural network (BNN) by assigning Gaussian priors to the weights and biases, as follows. For a fixed constant $\beta > 1$:

$$(W_1)_{ij} \sim \mathcal{N}\Big(0, \frac{\sigma_{w_1}^2}{i^\beta}\Big) \ \text{ for } \ i \in [N_2],\ j \in [N_1]; \tag{58}$$

$$(W_l)_{ij} \sim \mathcal{N}\Big(0, \frac{\sigma_{w_l}^2}{(ij)^\beta}\Big) \ \text{ for } \ i \in [N_l],\ j \in [N_{l-1}],\ l \in \{2, \ldots, L+1\}; \tag{59}$$

$$(b_l)_i \sim \mathcal{N}\Big(0, \frac{\sigma_{b_l}^2}{i^\beta}\Big) \ \text{ for } \ i \in [N_l],\ l \in [L+1]. \tag{60}$$

For a PDE problem with (log-)coefficients $\log a_x \in \mathbb{R}^{D \times D}$ for some $D \times D$ discretization of $[0,1]^2$, we can sample the log-coefficients from this BNN prior as follows: every coordinate $(y_{ij})_{i=1,j=1}^{D,D}$ of the discretization is propagated through the network, i.e. $\log a_x = (g_{NN}(y_{ij}))_{i=1,j=1}^{D,D}$, for a set of weights $(W_l)_{l=1}^{L+1}$ and biases $(b_l)_{l=1}^{L+1}$ sampled from their Gaussian priors (58), (59), and (60). Theorem 1 in [143] shows that as size of the hidden layers $N_l \to \infty$, samples $g_{NN}(y) : [0,1]^2 \to \mathbb{R}$ correspond to a Gaussian prior measure on $L^2([0,1]^2, \mathbb{R})$. While not exploited in our work, this result is important as it allows for the application of dimension-independent MCMC methods (e.g. preconditioned Crank-Nicolson MCMC [34]) for $p((W_l)_{ij}, (b_l)_i | \mathcal{D})$ given some observed data $\mathcal{D}$ (e.g. outputs of a PDE simulator). The key property of these methods is that the Metropolis-Hastings acceptance rate is independent of the size of the (finite) truncation N_l. This property was also empirically demonstrated for the BNN prior (58), (59), and (60) in the computer experiments from Section 5.2, [143].

Note that the prior variance of the weights $(W_l)_{ij}$ and biases $(b_l)_i$ decreases as $i \in [N_l]$ moves from $i = 1$ towards the tail node $i = N_l$; this is indeed required for Theorem 1 in [143] to hold. This approach is different from the existing BNN literature, where the Gaussian prior within each hidden layer usually has a constant variance $(\sigma^2_{w_l}, \sigma^2_{b_l})$ [173]. For this more popular scenario, the resulting prior distribution of $g_{NN}(y) : [0,1]^2 \to \mathbb{R}$ for hidden layers of finite size $(N_l)_{l=1}^{L+1} < \infty$ has been characterized in [121, 186], where it was shown that deeper models (i.e. large L) lead to heavier-tail priors for $g_{NN}(y)$.

6.1.2 *Ensemble smoother with multiple data assimilation (ES-MDA)*

Let $\mathcal{D} = f(\log a_x) + \epsilon$ be an inverse problem, where the goal is to recover the (log-)coefficients $\log a_{x^\star}$ corresponding to a PDE simulator f (e.g. the Elliptic PDE considered in Section 2.4) and a collection of observations $\mathcal{D}$. As usual, $\epsilon \sim \mathcal{N}(0, \Sigma_{obs})$ describes the modelling and observational errors. Let $p(\log a_x)$ be the prior distribution which encodes our prior beliefs about the unknown $\log a_{x^\star}$. As discussed in Section 1.2 and 1.3, one family of methods that can be used to efficiently tackle this problem are the Ensemble Kalman Inversion (EKI) algorithms. In this subsection, we present the EKI algorithm known as ensemble smoother with multiple data assimilation (ES-MDA [45]); a different EKI algorithm will be discussed in Section 6.2.

We describe the procedure in Algorithm 6. As demonstrated in Theorem 2.1 from [85], for any EKI algorithm, the particles $\log a_{x_m}^{(n)}$ belong to the span of the initial ensemble $\{\log a_{x_m}^{(0)}\}_{m=1}^{M}$ for every iteration n. In this regard, it is interesting to see how sampling the initial particles from the BNN prior (58), (59), and (60) affects the EKI performance.

The regularization parameters $\{\alpha_n\}_{n=1}^{N_{iter}}$ control the movement of particles from the prior distribution towards the posterior $p(\log a_x | \mathcal{D})$ [87, 82], since at every iteration, $\{\log a_{x_m}^{(n)}\}_{m=1}^M$ approximates a Gaussian distribution, which in turn approximates an intermediate (tempering) measure

$$p_n(\log a_x|\mathcal{D}) \propto \exp\Big[-\frac{1}{2}||(\alpha_n\Sigma_{obs})^{-1/2}(\mathcal{D}-f(\log a_x))||_2^2\Big]p_{n-1}(\log a_x|\mathcal{D}), \tag{61}$$

with $p_0(\log a_x|\mathcal{D}) = p(\log a_x)$ and $p_{N_{iter}}(\log a_x|\mathcal{D}) = p(\log a_x|\mathcal{D})$, given that $\sum_{n=1}^{N_{iter}} \alpha_n^{-1} = 1$ (see Appendix A from [82]).

Algorithm 6 Ensemble smoother with multiple data assimilation ([45])

```
$M$ : number of particles
$\{\log a_{x_m}^{(0)}\}_{m=1}^M$ : initial ensemble of particles, sampled from the prior
$N_{iter} = 10$ : number of iterations
$\{\alpha_n\}_{n=1}^{N_{iter}}$ : regularization parameters (62) that satisfy $\sum_{n=1}^{N_{iter}} \alpha_n^{-1} = 1$
for $n \in \{0,\dots,N_{iter}-1\}$ do
    1. Update each particle $m \in \{1,\dots,M\}$
    $\log a_{x_m}^{(n+1)} = \log a_{x_m}^{(n)} + \hat{C}_n^{xf}(\hat{C}_n^{ff} + \alpha_{n+1}\Sigma_{obs})^{-1}\big(\mathcal{D} + \sqrt{\alpha_{n+1}}z_n - f(\log a_{x_m}^{(n)})\big),$
    where $z_n \sim \mathcal{N}(0,\Sigma_{obs})$ is a perturbation of the data, and $\hat{C}_n^{xf}$, $\hat{C}_n^{ff}$ are
    the empirical covariance matrices
    $\hat{C}_n^{xf} := \frac{1}{M-1}\sum_{m=1}^M \big(\log a_{x_m}^{(n)} - \overline{\log a_x^{(n)}}\big)\big(f(\log a_{x_m}^{(n)}) - \bar{f}_n\big)^T,$
    $\hat{C}_n^{ff} := \frac{1}{M-1}\sum_{m=1}^M \big(f(\log a_{x_m}^{(n)}) - \bar{f}_n\big)\big(f(\log a_{x_m}^{(n)}) - \bar{f}_n\big)^T,$
    with $\overline{\log a_x^{(n)}} := \frac{1}{M}\sum_{m=1}^M \log a_{x_m}^{(n)}$ and $\bar{f}_n := \frac{1}{M}\sum_{m=1}^M f(\log a_{x_m}^{(n)})$.
end for
return $\log a_{\hat{x}} = \frac{1}{M}\sum_{m=1}^M \log a_{x_m}^{(N_{iter})}$ : the final ensemble mean, which is
our estimate for the (true) log-permeability $\log a_{x^\star}$
```

ES-MDA fixes $N_{iter} = 10$ and the regularization parameters $\{\alpha_n\}_{n=1}^{10}$ to:

$$\alpha_1 = 57.017, \alpha_2 = 35, \alpha_3 = 25, \alpha_4 = 20, \alpha_5 = 18, \alpha_6 = 15, \alpha_7 = 12, \alpha_8 = 8, \alpha_9 = 5, \alpha_{10} = 3. \tag{62}$$

An adaptive selection of N_{iter} and $\{\alpha_n\}_{n=1}^{N_{iter}}$ is preferred in more recent EKI algorithms [84, 87, 82], where various strategies for selecting $\{\alpha_n\}_{n=1}^{N_{iter}}$ and stopping criteria for N_{iter} were introduced; see [82] for a recent review of these methods. The work [84] demonstrates the benefits of choosing a large regularization parameter at the beginning of the procedure, followed by gradually decreasing α_n at further iterations in a more principled approach, based on the observation that EKI can be viewed as a derivative-free approximation of the regularizing Levenberg-Marquardt (LM) scheme [79] for solving inverse problems. If we use the final ensemble mean $\log a_{\bar{x}} := \frac{1}{M}\sum_{m=1}^{M} \log a_{x_m}^{(N_{iter})}$ as our estimate for the (true) log-coefficients $\log a_{x^\star}$, choosing a constant regularization value $\alpha_n = 1/N_{iter}$ can lead to a good data approximation $f(\log a_{\bar{x}}) \approx \mathcal{D}$, but a poor recovery $\log a_{\bar{x}} \not\approx \log a_{x^\star}$ [84].

6.1.3 *Computer experiments*

The 2D Elliptic PDE considered in this section consists of a two-phase (oil-water) Darcy flow simulator. Various versions of this simulator were used in petroleum engineering and groundwater modeling applications [25, 46, 83, 84, 4]. The work [46] is also the source of our numerical simulator; for a mathematical description, see Equations 79-80 in [83]. We use a 31×31 discretization of $[0,1]^2$ for a reservoir model, with one injection well I_1 at location $(3,3)$ and two producing wells $P1$ and $P2$ at location $(15,29)$ and $(27,7)$, respectively (see Figure 50). The inverse problem $\mathcal{D} = f(\log a_x) + \epsilon$ consists of recovering the PDE log-coefficients $\log a_{x^\star} \in \mathbb{R}^{31\times 31}$ according to this discretization, which corresponds to recovering the log-permeability field of the reservoir. In this regard, we use a collection of observations $\mathcal{D} := (\mathcal{D}_1, \ldots, \mathcal{D}_{36}) \in \mathbb{R}^{36}$, where at $T_0 := 12$ time points we observe the bottomhole presure at the injector well $I1$ $(\mathcal{D}_1, \ldots, \mathcal{D}_{12})$, and oil extraction rates at $P1$ $(\mathcal{D}_{13}, \ldots, \mathcal{D}_{24})$ and $P2$ $(\mathcal{D}_{25}, \ldots, \mathcal{D}_{36})$, respectively. The noise level $\epsilon \sim \mathcal{N}_{36}(0, \sigma_n^2 I_{36})$ is considered to be $\sigma_n^2 = 25 \cdot 10^{-4} ||f(\log a_{x^\star})||_2^2$, i.e. approximately 5% of the noise-free data $f(\log a_{x^\star})$.

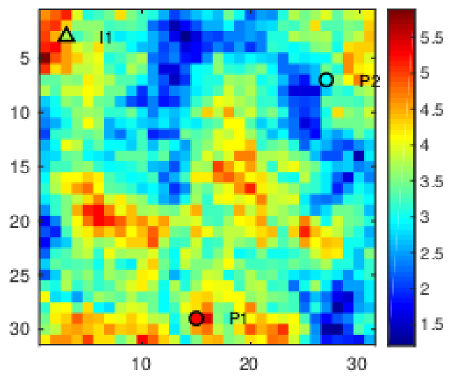


Figure 50: The reservoir model considered in this section, with one injection well I_1 at location $(3,3)$ and two producing wells $P1$ and $P2$ at location $(15,29)$ and $(27,7)$, respectively. We plot one configuration for PDE coefficients $\log a_x \in \mathbb{R}^{31\times 31}$ (i.e. one possible log-permeability field for the reservoir), sampled from a generic GRF prior which is not used in this work.

We employ the BNN prior from Subsection 6.1.1. As discussed in [121], the BNN parameter β from (58), (59), and (60) controls the complexity of the prior samples $\log a_x = (g_{NN}(y_{ij}))_{i=1,j=1}^{31,31}$, whereas $(\sigma^2_{w_l}, \sigma^2_{b_l})_{l=1}^{L+1}$ controls the variance of these samples. Smaller values of β (i.e. $\beta \approx 1$) lead to more complex samples (see Figure 3 in [121]); indeed, we use $\beta = 1.0001$ in this work. We consider two cases, i.e. a shallow (one hidden layer) neural network ($L = 1$), and a deep neural network ($L = 3$), respectively. In these cases, we have chosen the variance parameters $(\sigma^2_{w_l}, \sigma^2_{b_l})$ by trial-and-error, such that the resulting prior samples $\log a_x$ have an interesting structure. In Figure 51, we show three prior samples from the BNN prior with one hidden layer, whereas in Figure 52 we show three prior samples from the deep BNN prior with three hidden layers.

The true log-permeability $\log a_{x^\star} \in \mathbb{R}^{31\times 31}$ (Figure 53) is sampled using a Karhunen-Loève decomposition (22) of a squared-exponential GRF of long lengthscale:

$$\mathrm{Cov}[\log a_x(s), \log a_x(t)] = \exp(-l^{-2}||s-t||_2^2), \tag{63}$$

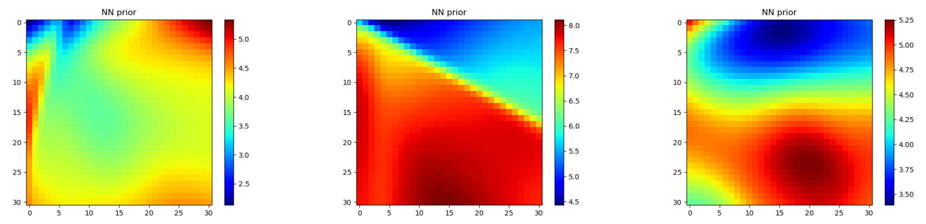


Figure 51: (Best viewed in colour) Samples from the BNN prior with one hidden layer.

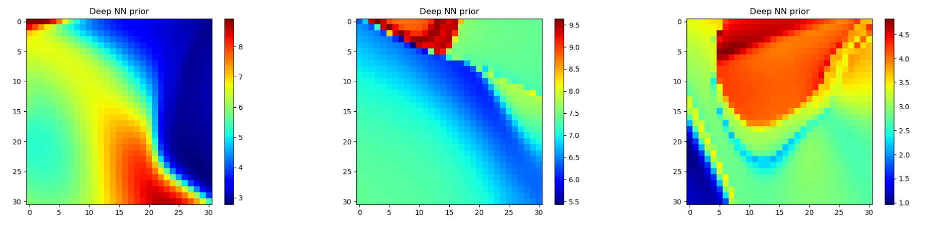


Figure 52: (Best viewed in colour) Samples from the deep BNN prior with three hidden layers.

where $l = 10$. We use a GRF with a long lengthscale, since we have noticed that the BNN prior samples do not look similar to samples produced via a Karhunen-Loève decomposition of GRFs of short lengthscale.

Since we use the Darcy flow simulator from [46], we also use the same EKI algorithm ES-MDA (Algorithm 6), with a fairly low number of particles $M = 100$. In Figure 54, we plot the true log-permeability field $\log a_{x^\star}$, together with the final EKI ensemble mean $\log a_{\bar{x}} = \frac{1}{M}\sum_{m=1}^{M} \log a_{x_m}^{(N_{iter})}$ from

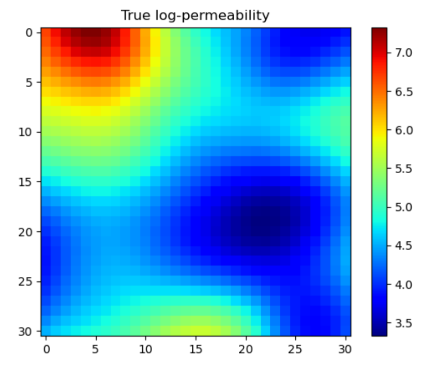


Figure 53: (Best viewed in colour) True log-permeability $\log a_{x^\star} \in \mathbb{R}^{31\times 31}$

the three different prior distributions, and thus three different ensembles of initial particles $\{\log a_{x_m}^{(0)}\}_{m=1}^{M}$. The prior distributions are as follows: the one-hidden layer BNN with samples from Figure 51 ('NN prior' in Figure 54), the three hidden layer BNN with samples from Figure 52 ('Deep NN prior'), and finally, the same prior which generated $\log a_{x^\star}$ ('GRF prior').

While the well-specified GRF prior leads to the best approximation of $\log a_{x^\star}$, we see that the two misspecified BNN priors also lead to a good recovery of the main features of $\log a_{x^\star}$. The single hidden layer BNN ('NN prior' in Figure 54) seems to better recover the smoothness of $\log a_{x^\star}$ compared to its three layer counterpart ('Deep NN prior'). This is better demonstrated in Figure 55 and 56, where for the single hidden layer BNN and the three layer BNN, respectively, we plot various samples $\log a_{x_m}^{(N_{iter})}$ from the final ensemble of particles, which we regard as approximate samples from the posterior distribution $p(\log a_x|\mathcal{D})$. For completeness, we present the corresponding EKI samples obtained from the GRF prior (63) in Figure 57, which seem to best capture the smoothness of $\log a_{x^\star}$ (Figure 53), as expected.

We conclude with a standard computer experiment in petroleum engineering [46], where the simulator outputs corresponding to samples from the final EKI iteration $\{f(\log a_{x_m}^{(N_{iter})})\}_{m=1}^{M}$ are compared with the true simulator response $f(\log a_{x^\star})$. As discussed, the observations $\mathcal{D}$ are collected at $T_0 = 12$ time points. We will extend the simulator to $T := T_0 + t = 17$ time points, i.e. we include 5 additional (future) time points. We would like to check how well $\{f(\log a_{x_m}^{(N_{iter})})\}_{m=1}^{M}$ compare with $f(\log a_{x^\star})$ at those future time points. Furthermore, we have only assimilated bottomhole pressure values at the injector well ($I1$ in Figure 50), and oil extraction rates at the two producers $P1$ and $P2$ (Figure 50); we will additionally extend the simulator to include the water production rates (at producer $P1$) for the same $T = T_0 + t = 17$ time points. In this way, we assess the performance of the final EKI ensemble in terms of prediction for new physical measurements.

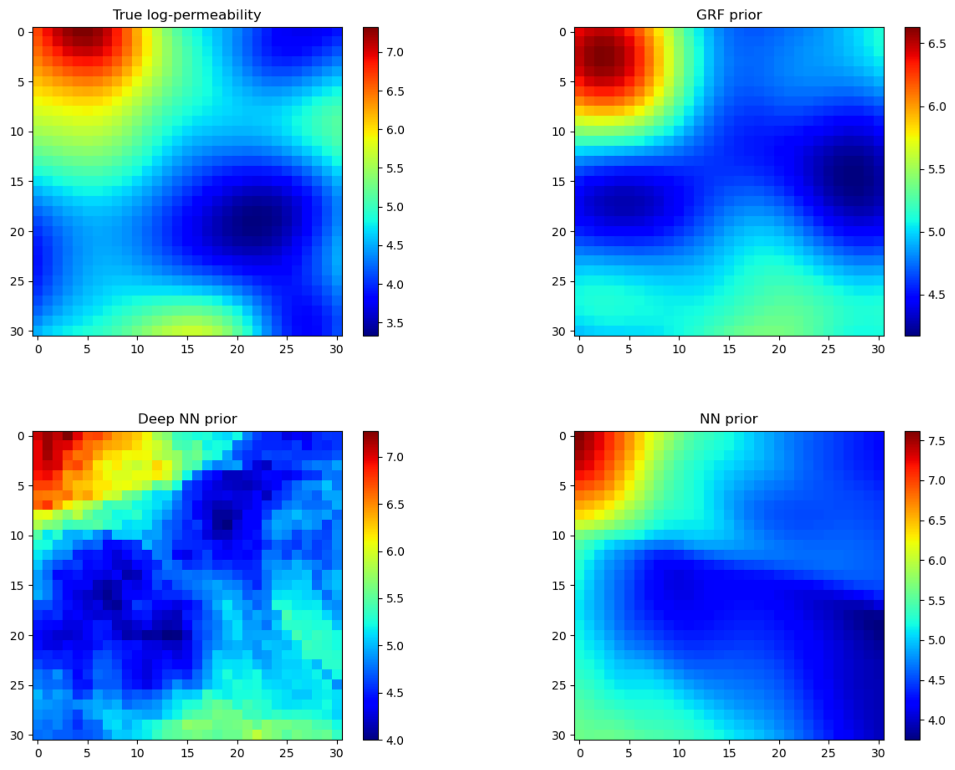


Figure 54: (Best viewed in colour) The true log-permeability field $\log a_{x^\star}$, together with estimates $\log a_{\tilde{x}}$ from the final EKI iteration for various prior distributions for the initial particles $\{\log a_{x_m}^{(0)}\}_{m=1}^{M}$. The axes of all the plots represent the 31×31 discretization of $[0,1]^2$

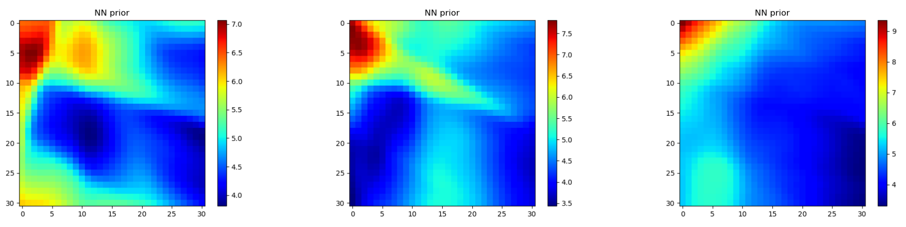


Figure 55: (Best viewed in colour) Samples from the last EKI iteration for the BNN prior with one hidden layer

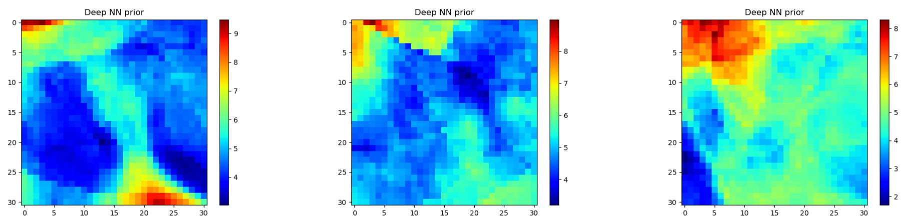


Figure 56: (Best viewed in colour) Samples from the last EKI iteration for the deep BNN prior with three hidden layers

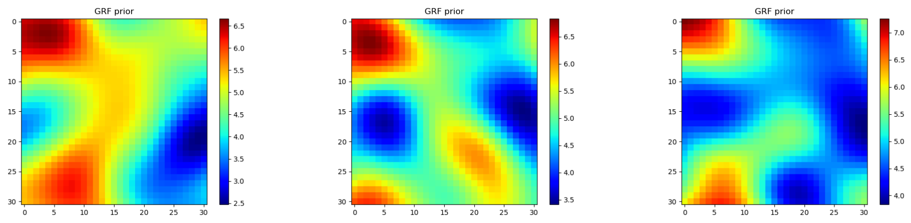


Figure 57: (Best viewed in colour) Samples from the last EKI iteration for the GRF prior (63)

We present the results for the EKI procedure using as initial particles the classical (Karhunen-Loève) GRF prior samples (Figure 58), the one hidden layer BNN prior samples (Figure 59), and the three hidden layer deep BNN prior samples (Figure 60), respectively. We see that $\{f(\log a_{x_m}^{(N_{iter})})\}_{m=1}^{M}$ is able to match the observed data $\mathcal{D}$ for all the priors, although some of the EKI samples with the deep BNN prior are significantly outside the noise level σ_n (e.g. Figure 60 for bottomhole pressure at the injector well). In terms of prediction at the additional 5 time points (i.e. $t \in \{13, \ldots, 17\}$), and for the new physical measurements (i.e. water production rates), we observe that the EKI samples obtained from the one hidden layer BNN prior (Figure 59) return similar predictions to the EKI samples with the classical GRF prior (Figure 58); recall that the true log-permeability field $\log a_{x^\star}$ was sampled from this classical GRF prior. Finally, the predictions provided by the EKI samples with the deep BNN prior (Figure 60) have a significantly larger variance, with many of the samples far away from $f(\log a_{x^\star})$.

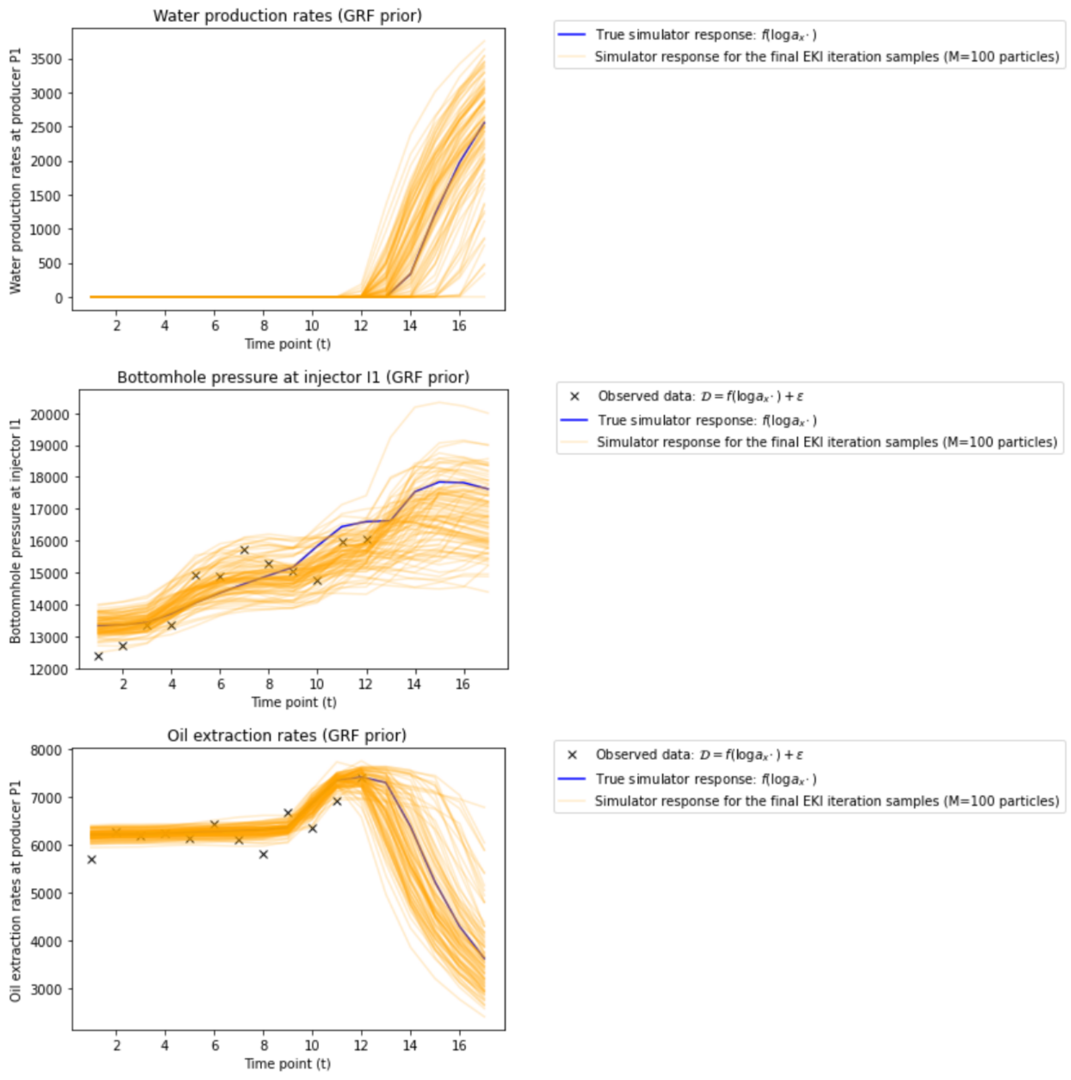


Figure 58: Simulator responses for the last EKI iteration for the GRF prior (63)

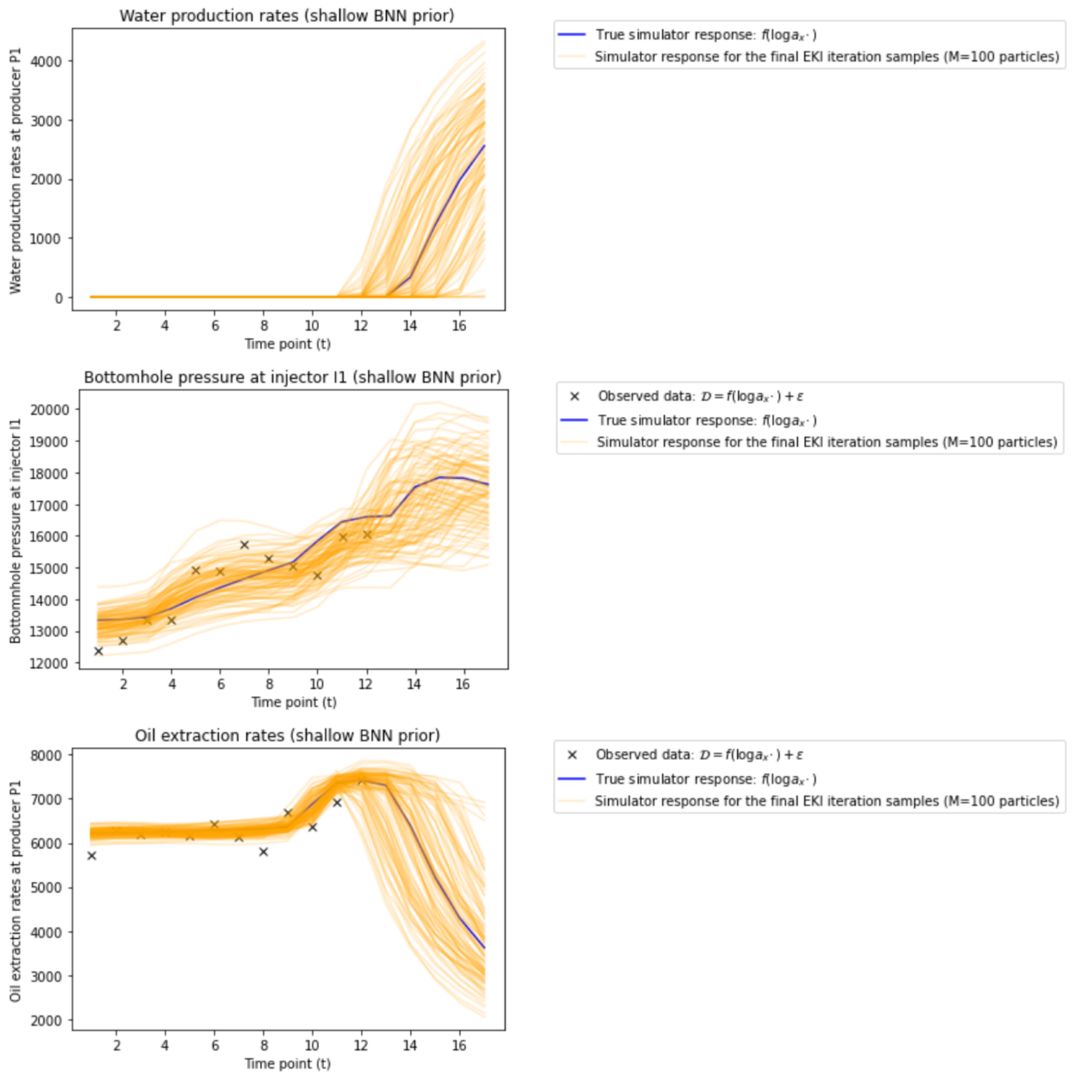


Figure 59: Simulator responses for the last EKI iteration for the one hidden layer BNN prior

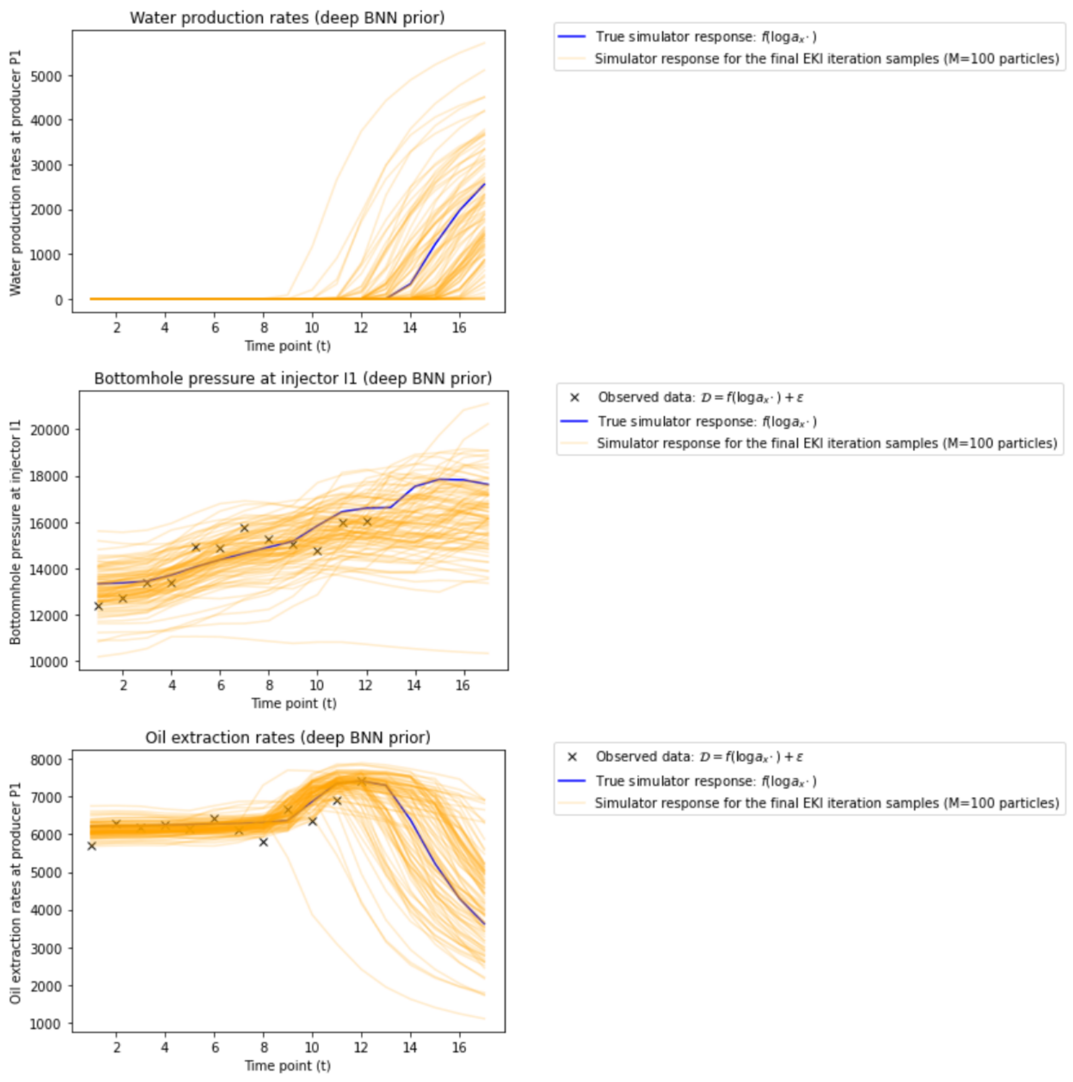


Figure 60: Simulator responses for the last EKI iteration for the three hidden layer BNN prior

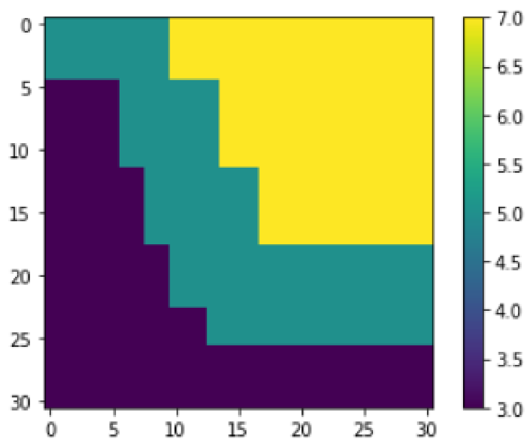


Figure 61: Discontinuous true log-permeability $\log a_{x^\star} \in \mathbb{R}^{31\times 31}$

Therefore, one conclusion is that the BNN parametrization (with one hidden layer) exhibits similar ES-MDA inversion performance with the GRF prior (63) that was used to generate the true log-permeability field (Figure 53). We will now demonstrate that for a discontinuous log-permeability field $\log a_{x^\star}$ (Figure 61), the BNN parametrization can significantly outperform the GRF prior (63). Channelized log-permeability fields similar to Figure 61 have been extensively considered in the inverse problems literature, due to their application in petroleum engineering and groundwater modelling [86, 21].

In Figure 62, we plot the true discontinuous log-permeability field $\log a_{x^\star}$, together with the final ES-MDA ensemble mean $\log a_{\bar{x}} = \frac{1}{M}\sum_{m=1}^{M} \log a_{x_m}^{(N_{iter})}$ from the two different prior distributions, and thus two different ensembles of initial particles $\{\log a_{x_m}^{(0)}\}_{m=1}^{M}$. The prior distributions are as follows: the one-hidden layer BNN discussed so far ('NN prior' in Figure 62), and the GRF prior (63) ('GRF prior'). We see that the one-hidden layer BNN parametrization captures the features of $\log a_{x^\star}$, since it correctly recovers the three discontinuous regions (channels) of different permeability values; the classical GRF prior is unable to capture these features. For completeness, we present Figure 63 and 64, where for the single hidden layer BNN and the classical GRF prior, respectively, we plot various samples $\log a_{x_m}^{(N_{iter})}$ from the final ensemble of particles, which we regard as approximate samples from

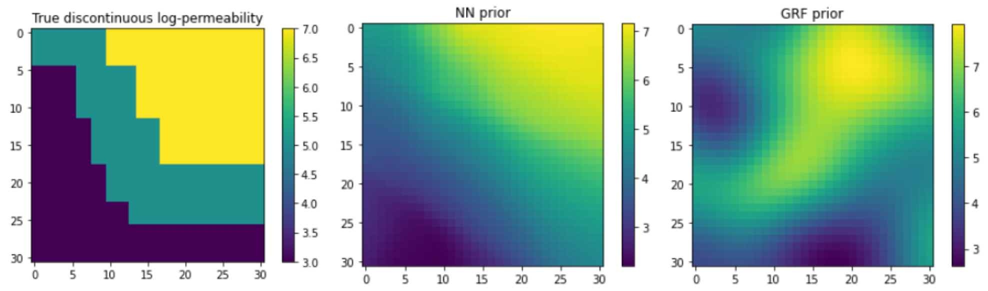


Figure 62: (Best viewed in colour) The true discontinuous log-permeability field $\log a_{x^\star}$, together with estimates $\log a_{\tilde{x}}$ from the final EKI iteration for various prior distributions, and hence various initial particles $\{\log a_{x_m}^{(0)}\}_{m=1}^{M}$

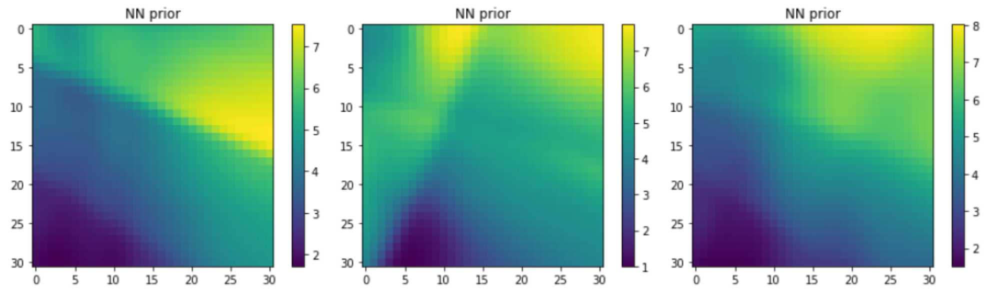


Figure 63: (Best viewed in colour) Samples from the last EKI iteration for the BNN prior with one hidden layer

the posterior distribution $p(\log a_x|\mathcal{D})$. The same conclusion drawn from Figure 62 applies to these samples, i.e. the BNN parametrization recovers the features of $\log a_{x^\star}$, as opposed to the classical GRF prior.

Finally, regarding the simulator responses of the final EKI iteration samples for different physical measurements, we present analogue results for Figure 59 and Figure 58, in Figure 65 and Figure 66, respectively, for the the one hidden layer BNN parametrization and the classical GRF prior. Note that for this discontinuous true log-permeability case, the water production rates are zero for all $t \in \{1, \ldots, 17\}$. The EKI samples corresponding to the BNN parametrization return far fewer non-zero water production rates (Figure 65) compared with the classical GRF prior (Figure 66); recall that water production rates were not included (assimilated) in our observed measure-

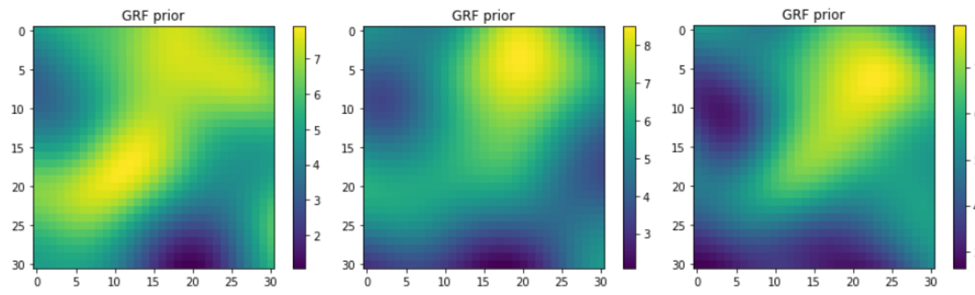


Figure 64: (Best viewed in colour) Samples from the last EKI iteration for the GRF prior (63)

ments $\mathcal{D}$. Regarding bottomhole pressure and oil extraction rates, the two priors lead to similar performance.

In conclusion, we have shown that the BNN parametrization for GRF priors that was recently introduced in [143] can potentially be useful for solving inverse problems with Darcy flow simulators via EKI. One interesting question is what happens if instead of using a fixed set of BNN parameters $(\alpha, \sigma^2_{w_l}, \sigma^2_{b_l})_{l=1}^{L+1}$, we use M samples $\{(\alpha^{(m)}, (\sigma^2_{w_l})^{(m)}, (\sigma^2_{b_l})^{(m)})_{l=1}^{L+1}\}_{m=1}^{M}$ as our initial particles for EKI (e.g. sampled from a Uniform of Gaussian hierarchical prior distribution). The existing work [22] showed that hierarchical EKI parametrizations can significantly outperform the standard EKI parametrization for initial particles $\{\log a^{(0)}_{x_m}\}_{m=1}^{M}$. Unfortunately, while trying these experiments, we did not see any benefits from using this hierarchical BNN parametrization for EKI, nor from using multiple layers or convolutional neural networks (CNN) (e.g. very deep and relatively wide networks with the alternative BNN prior variances suggested by [121], or the convolutional architecture used for the recently introduced deep Gaussian Markov random fields [150]).

6.2 GAUSSIAN PROCESS EMULATION WITHIN EKI

The work [91] proposed using a GP emulator within an EKI algorithm known as the Iterative Ensemble Smoother (IES) [25, 24], so that the com-

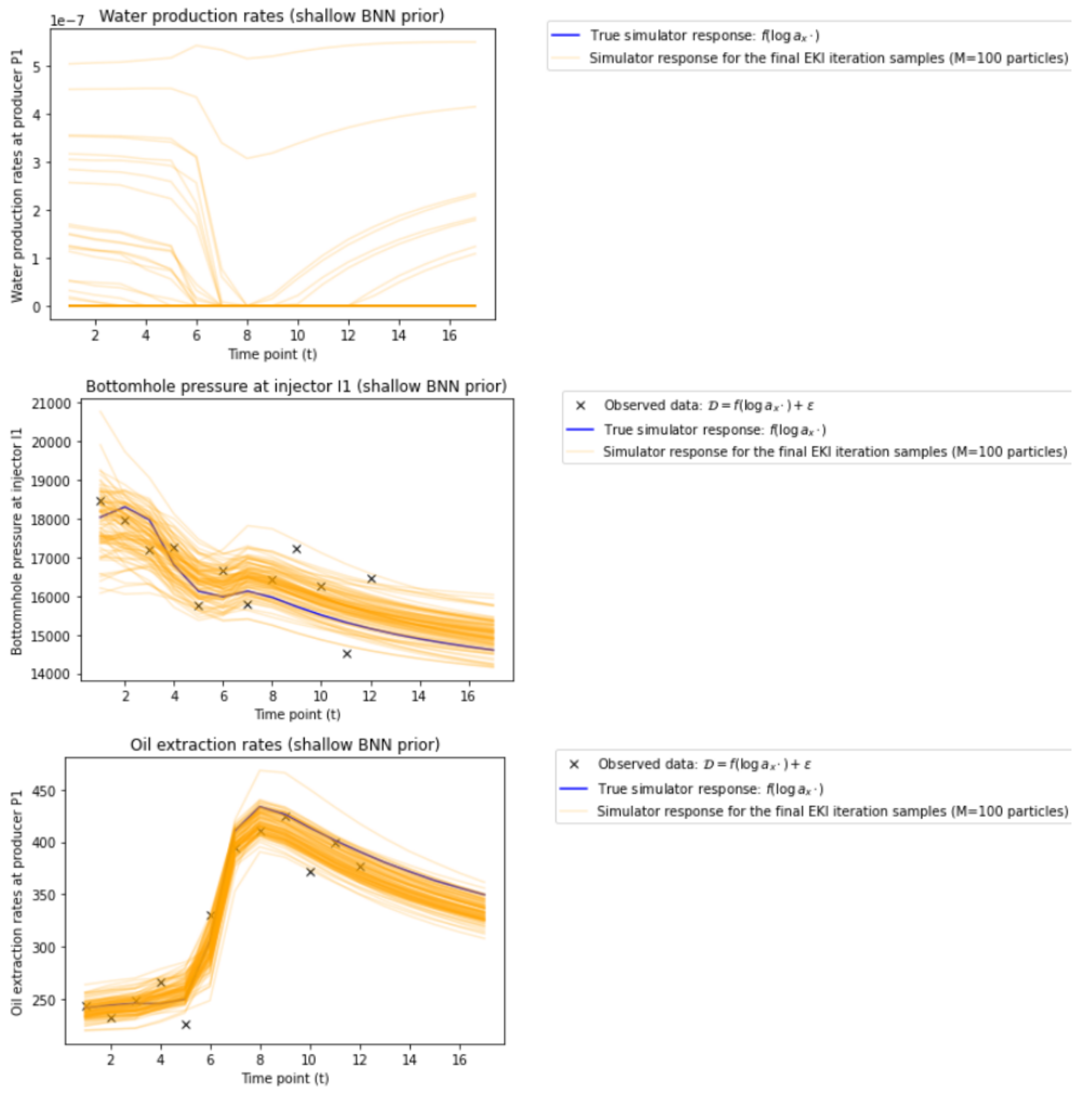


Figure 65: Simulator responses for the last EKI iteration for the one hidden layer BNN prior (discontinuous true log-permeability field from Figure 61); for the water rates, the blue line overlaps with 0

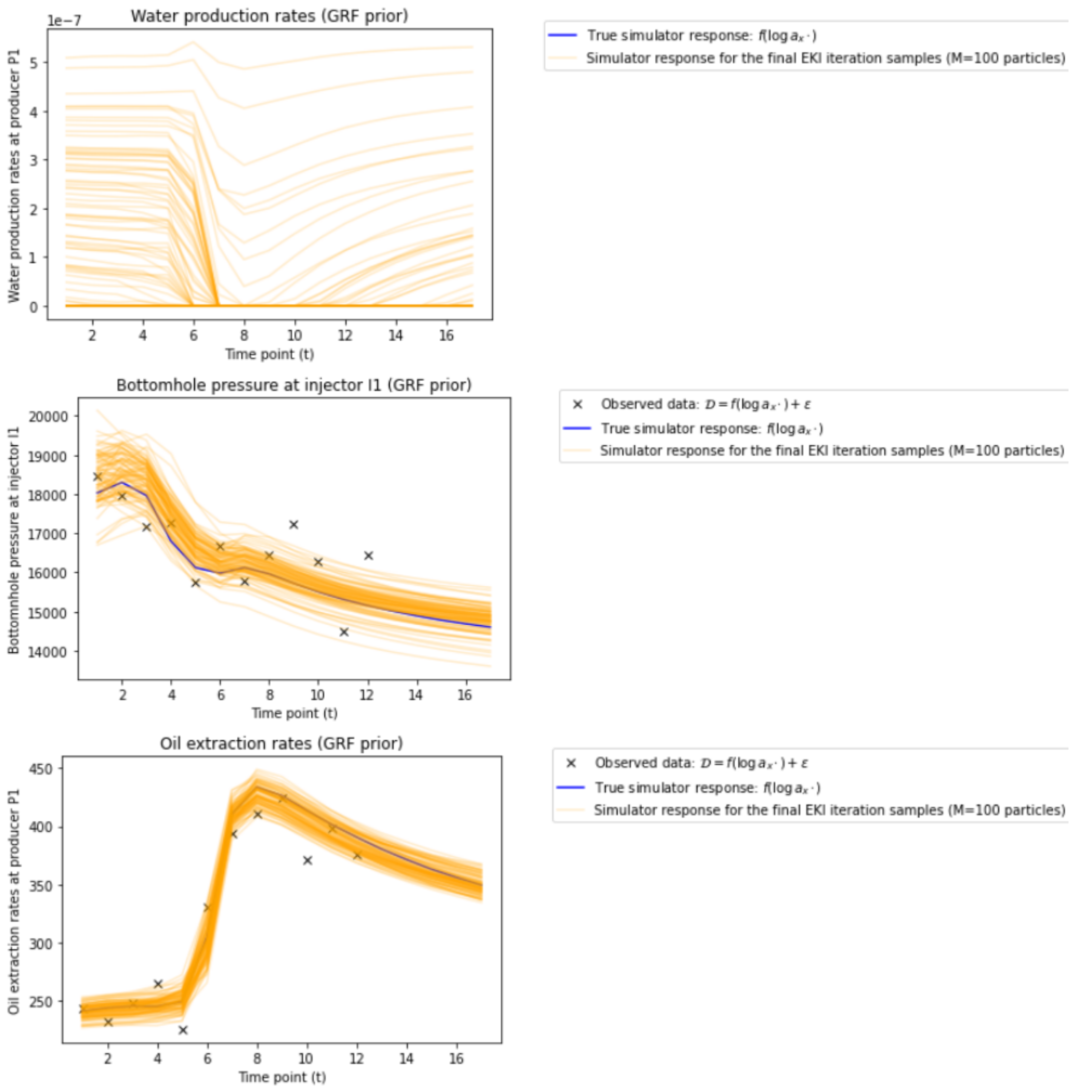


Figure 66: Simulator responses for the last EKI iteration for the GRF prior (63) (discontinuous true log-permeability field from Figure 61); for the water rates, the blue line overlaps with 0

putational burden of applying the forward simulator for each of the M particles at every EKI iteration (see e.g. Algorithm 6) will be alleviated by replacing the simulator with a fast GP emulator for some of the particles. Our goal is to look at this procedure within the novel EKI method EKI-DMC [82], and to propose one potential improvement regarding the GP training set design.

This section is organized as follows. In Subsection 6.2.1, we will briefly introduce EKI-DMC. In Subsection 6.2.2, we propose a potential improvement for the original procedure from [91]; our contribution involves a new approach for selecting the GP training points, based on the strategy proposed by [59] for a different family of inverse problem methods known as History Matching (HM) [175, 168]. We demonstrate the benefits of our proposed procedure for a groundwater modelling inverse problem in Subsection 6.2.3.

6.2.1 *EKI with data misfit controller (EKI-DMC)*

Let $\mathcal{D} = f(\log a_x) + \epsilon$ be the inverse problem setting from Subsection 6.1.2. Compared with ES-MDA (Algorithm 6) from the previous section, EKI-DMC (Algorithm 3 from [82]) is an adaptive EKI algorithm, i.e. the number of iterations N_{iter} and the regularization parameters $\{\alpha_n\}_{n=1}^{N_{iter}}$ are not fixed a-priori. At every iteration, α_n is chosen (see Step 2. in Algorithm 7) such that the Jeffrey's divergence between the intermediate (tempering) measures (61)

$$D_{KL,2}(p_{n+1}, p_n) := D_{KL}(p_{n+1}||p_n) + D_{KL}(p_n||p_{n+1}),$$

is bounded by θ, where θ is selected according to the classical Morozov's discrepancy principle [116], applied for the inverse (sub-)problem $\mathcal{D} = f(\log a_x) + \sqrt{\alpha_n}\epsilon$, with prior distribution $p_n(\log a_x|\mathcal{D})$ and posterior distribution $p_{n+1}(\log a_x|\mathcal{D})$ (see Remark 1 in [82]); recall that the KL divergence D_{KL} was defined in (51). EKI-DMC is the first EKI algorithm which does not require any tuning hyperparameters (in the original version with no emulators [82]; the emulator setting presented here does require few hyperparameters, as discussed in Section 6.2.2), and its advantages compared with the existing EKI methods in terms of speed (i.e. low N_{iter}) and accuracy (e.g. $\log a_{\tilde{x}} \approx \log a_{x^\star}$, where $\log a_{\tilde{x}} := \frac{1}{M}\sum_{m=1}^{M} \log a_{x_m}^{(N_{iter})}$ is the final ensemble mean) have been shown in the computer experiments of [82].

6.2.2 *Gaussian Process emulation within EKI-DMC*

Let $\mathcal{D} = f(x) + \epsilon$ be a classical inverse problem with forward simulator $f : \mathbb{R}^D \to \mathbb{R}^q$. As usual, $\epsilon \sim \mathcal{N}_q(0, \sigma_n^2 I_q)$ describes the modelling and observational errors. The original idea from [91] is very simple and straightforward to implement within any EKI algorithm. Namely, for each output $f_j(x)$ ($j \in [q]$), the forward simulator is replaced by a GP emulator $[f_{\text{GP}}(x)]_j$ (3), while taking into account the GP predictive uncertainty; we will shortly discuss the selection of training points and thus the construction of the GP emulator. For any input x, the GP predictive uncertainty $\bar{k}_j(x, x)$ is added to the original noise of the inverse problem, which results in the following modified likelihood: $p_{\text{GP}}(\mathcal{D}_j|x) = \mathcal{N}(\mathcal{D}_j|\bar{m}_j(x), \sigma_n^2 + \bar{k}_j(x, x))$, where $\bar{m}_j(x)$ is the GP predictive mean which tries to approximate $f_j(x)$.

We present a generic algorithmic version of GP emulation within EKI-DMC in Algorithm 7. The GP training set, and thus the GP emulator is updated at every EKI iteration. The initial particles $\{x_m^{(0)}\}_{m=1}^{M}$ provide $N_{base} < M$ training points $\{x_{m_{k_0}}^{(0)}, f(x_{m_{k_0}}^{(0)})\}_{k_0=1}^{N_{base}}$; the resulting GP emulator $f_{\text{GP}}(x) := ([f_{\text{GP}}(x)]_1, \ldots, [f_{\text{GP}}(x)]_q)^T$ is used as a fast computational replacement of the expensive simulator for the remaining $M - N_{base}$ particles. At every further iteration n, we augment the training set with additional $N_{add} < N_{base}$ points $\{x_{m_{k_n}}^{(n)}, f(x_{m_{k_n}}^{(n)})\}_{k_n=1}^{N_{add}}$.

Algorithm 7 GP emulation within EKI-DMC

```
$M$ : number of particles
$n = 0$ : first iteration
$\{x_m^{(0)}\}_{m=1}^M$ : initial ensemble of particles, sampled from the prior $p(x)$
$t_n = 0$ : initial regularization
$\{x_{m_{k_0}}^{(0)}, f(x_{m_{k_0}}^{(0)})\}_{k_0=1}^{N_{base}}$: initial GP training points, selected from the initial ensemble of particles ($N_{base} < M$)
$N_{add} < N_{base}$ : no. of additional GP training points at every EKI iteration
while $t_n < 1$ do
    1. Train the GP emulator using the current training set; the resulting GP posterior $\mathcal{N}([f_{\text{GP}}(x)]_j | \bar{m}_j(x), \bar{k}_j(x,x))$ will replace $f_j(x)$ for $j \in [q]$
    2. Compute the regularization parameter $\alpha_{n+1}$ via
       $$\alpha_{n+1}^{-1} := \min\Big\{ \max\Big\{ \frac{q}{2\mathbb{E}[\delta^2]}, \sqrt{\frac{q}{2\mathbb{V}\text{ar}[\delta^2]}} \Big\}, 1 - t_n \Big\},$$
       where $\mathbb{E}[\delta^2]$ and $\mathbb{V}\text{ar}[\delta^2]$ are the empirical mean and variance, respectively, of the data misfits $\delta^2(x_m^{(n)}) := 1/2||(\Sigma_{obs} + \bar{k}(x_m^{(n)}, x_m^{(n)}))^{-1/2}(\mathcal{D} - \bar{m}(x_m^{(n)}))||_2^2$ for $m \in [M]$ (we write $\bar{m}(x) := (\bar{m}_1(x), \ldots, \bar{m}_q(x))^T$ and $\bar{k}(x,x) := \text{diag}(\bar{k}_1(x,x), \ldots, \bar{k}_q(x,x)) \in \mathbb{R}^{q \times q}$ using our GP emulator)
    3. Update each particle $m \in \{1, \ldots, M\}$ according to Step 1. in Algorithm 6 (we point to Algorithm 6 in order to keep the notation here compressed), where the likelihood $p(\mathcal{D}_j|x) = \mathcal{N}(\mathcal{D}_j | f_j(x), \sigma_n^2)$ is replaced by $p_{\text{GP}}(\mathcal{D}_j|x) = \mathcal{N}(\mathcal{D}_j | \bar{m}_j(x), \sigma_n^2 + \bar{k}_j(x,x))$ for every output $j \in [q]$
    4. $n := n + 1$
    5. $t_n := t_n + \alpha_n^{-1}$
    6. From the updated set of particles $\{x_m^{(n)}\}_{m=1}^M$, add $\{x_{m_{k_n}}^{(n)}, f(x_{m_{k_n}}^{(n)})\}_{k=1}^{N_{add}}$ to the GP training set
end while
return $N_{iter} := n$ (the total number of iterations) and $\tilde{x} = \frac{1}{M}\sum_{m=1}^M x_m^{(N_{iter})}$
```

This iterative strategy for constructing the training set proved to be more effective than using a training set of equivalent size $N_{base} + N_{iter} \times N_{add}$, but with inputs sampled from the prior distribution $p(x)$ before performing EKI (see Figure 4 in [91]). Intuitively, this makes sense, as we want our GP emulator to be more accurate in the high posterior density regions of $p(x|\mathcal{D})$, where the EKI-DMC particles are likely to eventually end up.

In the original work [91], the additional N_{add} training points $\{x_{m_{k_n}}^{(n)}, f(x_{m_{k_n}}^{(n)})\}_{k=1}^{N_{add}}$ (Step 6. in Algorithm 7) are sampled at random from the particles $\{x_m^{(n)}\}_{m=1}^{M}$ present at every iteration n. Our contribution is to adapt a maxi-min experimental design recently introduced in [59], in order to select N_{add} training points that will hopefully improve the quality of emulation and uncertainty quantification. Namely, all particles at the current iteration are ranked according to their GP predictive uncertainty (assuming it is well-calibrated, e.g., via cross-validation) $\{\sum_{j=1}^{q} \bar{k}_j(x_m^{(n)}, x_m^{(n)})\}_{m=1}^{M}$. Out of the most uncertain N_{unc} particles $\{\tilde{x}_m^{(n)}\}_{m=1}^{N_{unc}}$, we add N_{add} points $\{\tilde{x}_{m_k}^{(n)}, f(\tilde{x}_{m_k}^{(n)})\}_{k=1}^{N_{add}}$ to the GP training set according to Algorithm 8.

Algorithm 8 The addition of N_{add} GP training points at EKI iteration n

```
$\mathcal{D}_n$: GP training set of size $N_{base} + (n-1) \times N_{add}$ before EKI iteration $n$
$N_{unc} \leq M$: threshold hyperparameter
$\{\tilde{x}_m^{(n)}\}_{m=1}^{N_{unc}}$: particles of highest GP predictive uncertainty from $\{x_m^{(n)}\}_{m=1}^{M}$
$\tilde{x}_{m_1}^{(n)}$ : most uncertain point $\big(\tilde{x}_{m_1}^{(n)} := \arg\max_{m \leq N_{unc}} \sum_{j=1}^{q} \bar{k}_j(\tilde{x}_m^{(n)}, \tilde{x}_m^{(n)})\big)$
$\tilde{x}_{m_2}^{(n)}$ : next furthest point $(\tilde{x}_{m_2}^{(n)} := \arg\max_{x' \in \{\tilde{x}_m^{(n)}\}_{m=1}^{N_{unc}}} ||\tilde{x}_{m_1}^{(n)} - x'||_2)$
: include both $(\tilde{x}_{m_1}^{(n)}, f(\tilde{x}_{m_1}^{(n)}))$ and $(\tilde{x}_{m_2}^{(n)}, f(\tilde{x}_{m_2}^{(n)}))$ in the GP training set $\mathcal{D}_n$
for $p \in \{3, \ldots, N_{add}\}$ do
    1. Let $\tilde{x}_{m_p}^{(n)}$ be the point from $\{\tilde{x}_m^{(n)}\}_{m=1}^{N_{unc}}$ which is furthest away to the
    GP training set $\mathcal{D}_n$, using the distance $\max_{x' \in \mathcal{D}_n} ||\tilde{x}_{m_p}^{(n)} - x'||_2$
    2. Include $(\tilde{x}_{m_p}^{(n)}, f(\tilde{x}_{m_p}^{(n)}))$ in the GP training set $\mathcal{D}_n$
end for
```

This design was proposed for the inverse problems procedure known as History Matching (HM); see [168] for an introduction to HM and one application in cosmology, and [175] for an application in climate modelling. In HM, every iteration returns (smaller and smaller) plausible regions for the unknown parameter x_{true}, starting from the prior distribution $p(x)$ at the first iteration. The maxi-min design from Algorithm 8 was introduced by [59] as a way to use these plausible regions in order to improve the quality of a GP emulator for the simulator within the HM procedure. We share a similar goal in this work, although we use the particles returned by EKI instead of the regions returned by HM.

6.2.3 *Computer experiments*

In this subsection, we consider an inverse problem based on an Elliptic-PDE simulator $f : \mathbb{R}^D \to \mathbb{R}^{400}$, evaluated at 400 points of the domain $[0,1]^2$. The Elliptic-PDE simulator is still a Darcy flow model, but this time we consider a groundwater single-phase steady-state simulator, whose mathematical description can be found in Equations 11-14 of [84]. The inputs follow a standard multivariate Gaussian prior $x \sim \mathcal{N}_D(0, I)$, which generate the squared-exponential Gaussian random field (GRF) prior of long lengthscale (63) for the PDE coefficients, using the Karhunen-Loève decomposition (21). We sample x_{true} from $\mathcal{N}(0, I_{100})$, which is used to generate the observations $\mathcal{D} = f(x_{true}) + \epsilon$. The noise level $\epsilon \sim \mathcal{N}_{400}(0, \sigma_n^2 I_{400})$ is set to $\sigma_n^2 = 10^{-4}||f(x_{true})||_2^2$, i.e. approximately 1% of the noise-free data $f(x_{true})$. To simplify notation, we write $\Sigma_{obs} := \sigma_n^2 I_{400}$. We use $D = 12$ for our inverse problem, since we have noticed that the ratio between the (cumulative) Karhunen-Loève eigenvalues is $\sum_{i=1}^{12} \lambda_i / \sum_{i=1}^{100} \lambda_i \approx 0.95$, i.e. the first 12 Karhunen-Loève coefficients capture 95% of the variance explained by the first 100 Karhunen-Loève coefficients, which were used to sample x_{true}.

We use independent GP emulators for each output $\{[f_{\mathrm{GP}}(x)]_1, \ldots, [f_{\mathrm{GP}}(x)]_{400}\}$, as a standard multi-output GP implementation where the correlation between outputs is taken into account [17] lead to under-confident predictions (i.e. too large predictive uncertainties). We compare the two GP training designs for Step 6. of Algorithm 7, i.e. adding N_{add} training points

$\{x_{m_{k_n}}^{(n)}, f(x_{m_{k_n}}^{(n)})\}_{k_n=1}^{N_{add}}$ at random from the particles $\{x_m^{(n)}\}_{m=1}^{M}$ at every iteration n [91], versus the design proposed in Algorithm 8.

We will assess the performance of Algorithm 7 in terms of the total number of iterations N_{iter} and the final data misfit $\delta^2 := ||(\Sigma_{obs} + \bar{k}(\tilde{x}, \tilde{x}))^{-1/2}(\mathcal{D} - \bar{m}(\tilde{x}))||_2^2$, where $\tilde{x} := \frac{1}{M}\sum_{m=1}^{M} x_m^{(N_{iter})}$ is the ensemble mean at the final EKI iteration, $\bar{m}(\tilde{x}) := (\bar{m}_1(\tilde{x}), \ldots, \bar{m}_{400}(\tilde{x}))^T$ is the GP predictive mean, and $\bar{k}(\tilde{x}, \tilde{x}) := \text{diag}(\bar{k}_1(\tilde{x}, \tilde{x}), \ldots, \bar{k}_{400}(\tilde{x}, \tilde{x})) \in \mathbb{R}^{400\times 400}$ is the GP predictive uncertainty; the GP posterior $(\bar{m}, \bar{k})$ is the GP emulator from the final EKI iteration. Under the assumption $\mathcal{D} \sim \mathcal{N}_{400}(\bar{m}(\tilde{x}), \Sigma_{obs} + \bar{k}(\tilde{x}, \tilde{x}))$, we see that δ^2 follows a Chi-squared distribution with 400 degrees of freedom, and thus $\mathbb{E}[\delta^2/400] = 1$, $\mathbb{V}[\delta^2/400] = 2/400 = 0.005$. As a result, we consider $\delta^2/400 \approx 1$ to be indicative of a successful GP emulation procedure within EKI-DMC. We have used a standard number of particles $M = 300$, with $N_{base} = 50$ and $N_{add} = 10$ for training the GP emulator. We have used the uncertainty threshold $N_{unc} = 50$ in Algorithm 8, as a larger threshold $N_{unc} = 100$ or $N_{unc} = 200$ did not improve the EKI performance.

Table 9: Performance of GP emulation within EKI-DMC (mean and standard deviation over 10 EKI-DMC trials). Best performing method is shown in bold (i.e. better data misfit $\delta^2/400 \approx 1$, smaller number of iterations N_{iter} and simulator calls (evaluations)).

	random design [91]	proposed (Algorithm 8)
N_{iter}	8.1 ± 1.51	**5.6 ± 1.28**
Simulator calls	131 ± 15	**106 ± 13**
$\delta^2/400$	1.64 ± 0.95	**1.11 ± 0.16**

We present the results in Table 9. They are very encouraging, as the results show a clear advantage both in terms of speed of EKI-DMC convergence (smaller N_{iter}) and data misfit ($\delta^2/400 \approx 1$) for the proposed GP training design. We have additionally performed several EKI-DMC trials using the original simulator instead of the GP emulator; this resulted in a slightly smaller number of EKI-DMC iterations compared with the GP emulator approaches from Table 9, as also observed in the similar experiment from [91]. Therefore, the low number of iterations achieved by our proposed design is not unreasonable. Note that each EKI-DMC iteration with the original simulator requires $M = 300$ forward simulator evaluations, whereas our GP emulation based procedure runs for at most 10 EKI-DMC iterations, and hence it uses at most $N_{base} + 10 \times N_{add} = 50 + 10 \times 10 = 150$ simulator evaluations (calls) in total.

As shown for one of the 10 trials in Figure 67 (random design [91]) and Figure 68 (proposed design (Algorithm 8)), both GP training designs result in a relatively good visual approximation for the final EKI-DMC ensemble mean $\log a_{\bar{x}} := \frac{1}{M}\sum_{m=1}^{M} \log a_{x_m}^{(N_{iter})}$ with regard to the true log-conductivity field $\log a_{x_{true}}$ (as usual, we write $\log a_x$ for the Elliptic PDE log-coefficients corresponding to the Karhunen-Loève coefficients x (21)). Note that the same conclusion applies to all the trials, and that the visual approximation is in line with using EKI-DMC with the original simulator (i.e. with no GP emulation).

We will further investigate these results in future work; one observation is that the proposed design shortens the gap between the GP predictive uncertainties. In other words, using the random design from [91], there is a larger GP uncertainty gap between the high uncertainty particles and the low uncertainty particles, during most EKI-DMC iterations of Algorithm 7. One interesting question is whether we can find better GP training designs for EKI; one popular sequential design known as Mutual Information for Computer Experiments (MICE) [11] did not improve the EKI performance, and it was much slower than the proposed design (Algorithm 8), as for every particle $x_m^{(n)}$ for $m \in [M]$, MICE needs to employ $M \times 400$ separate GP models. The work [176] presents an alternative design for History Matching, which we will consider in future work.

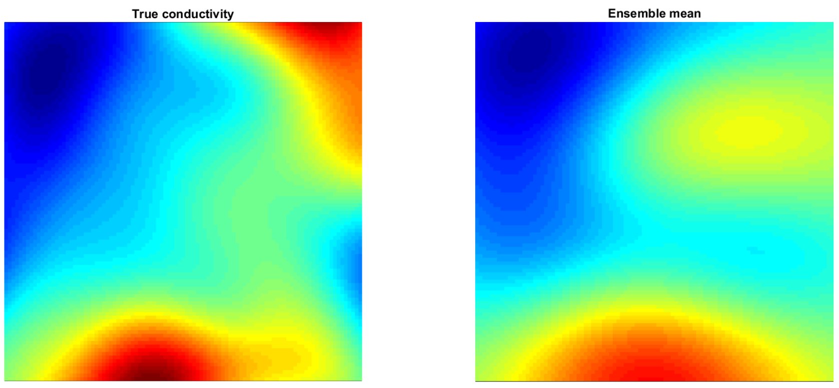


Figure 67: (Best viewed in colour) True log-conductivity $\log a_{x_{true}}$ (left) and the final EKI ensemble mean $\log a_{\bar{x}}$ for the GP emulation procedure with the random design [91] (right)

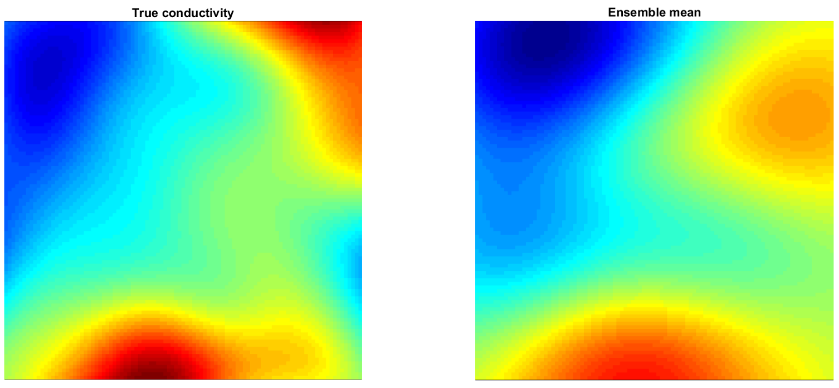


Figure 68: (Best viewed in colour) True log-conductivity $\log a_{x_{true}}$ (left) and the final EKI ensemble mean $\log a_{\bar{x}}$ for the GP emulation procedure with the proposed design (Algorithm 8) (right)

7 CONCLUSIONS AND FUTURE WORK

To recap, the methods studied in this thesis are motivated by problems in the field of Computational Statistics. We have focused on two types of methods, i.e. Gaussian Process (GP) emulators and Bayesian Inverse Problems. From the GP emulators perspective, we have thoroughly studied various dimension reduction methods such as Active Subspaces (AS), Principal Component Analysis (PCA), Sufficient Dimension Reduction (SDR), embedding learning methods, and random embedding methods. From the Bayesian Inverse Problems perspective, we have shown the applicability of our GP contributions through various inverse problems methods such as Markov Chain Monte Carlo (MCMC), Randomized Maximum Likelihood (RML), and Ensemble Kalman Inversion (EKI). We have demonstrated the versatility of our methods in a variety of real-world applications, such as epidemiology [44, 105], hydrology [68], aerodynamics [107], groundwater modelling [84], and petroleum engineering [46]. We now proceed by describing our contribution from each chapter.

In Chapter 2, we have shown that the magnitude of the Karhunen-Loéve eigenvalues (21) is crucial for method comparison, as for a fixed rate of decay for the eigenvalues, the high-dimensional GP baseline (with no dimension reduction) can outperform Principal Component Analysis (PCA) at smaller magnitudes, while the opposite is true at larger magnitudes. While some works only point out the percentage of the prior variance explained by the first d coefficients $\{\lambda_i\}_{i=1}^{d}$ when using PCA for GP emulation [15], our work is the first one to outline the additional importance of the magnitude of KL coefficients when considering PCA as an alternative option to the high-dimensional GP baseline. For future work, we would like to attempt a new theoretical result based on this observation, as well as to consider other simulators apart from the Elliptic PDE from Section 2.4.

In Chapter 3, we have presented a variety of computer experiments in order to showcase the performance of various supervised and unsupervised dimension reduction methods. One conclusion is that all the unsupervised dimension reduction methods achieved a similar performance, and that supervised dimension reduction methods are beneficial when we have enough training data. The gradient-based active subspace is the best performing method, although gradients of the simulator are often unavailable in practice, as discussed in Section 1.3. Regarding gradient-free methods, maximum likelihood embedding learning seems to perform best in the medium-large training data regime, with the caveat that it can actually be the worst performing method in the very small data regime. For future work, we would like to investigate different prior distributions for the input space, so that the unsupervised dimension reduction methods may exhibit a more diverse and (potentially) better performance.

In Chapter 4, we have presented a (gradient-free) version of the (gradient-based) AS-MCMC procedure [31], which we call learned embedding AS-MCMC (LE-AS-MCMC). We have demonstrated that LE-AS-MCMC can exploit the low-dimensional active subspace structure of log-likelihoods in various Bayesian inverse problems, and can achieve successful and efficient posterior sampling. In addition, we have shown that a neural network emulator can adapt faster than the learned embedding GP (20) to the intrinsic low-dimensionality corresponding to the active subspace (i.e. the NN requires a smaller number of training points), although the neural network cannot be easily adapted for efficient low-dimensional posterior sampling due to its high-dimensional input. For future work, we would like to investigate the theoretical properties of our procedure LE-AS-MCMC, starting from the theoretical guarantees offered by AS-MCMC in [31].

In Chapter 5, we propose HD-BO-RML, in which the high-dimensional Bayesian Optimization (HD-BO) machinery is used to tackle the Randomized Maximum Likelihood (RML) methodology for posterior sampling, in case of log-likelihoods with an active subspace structure. From a posterior sampling perspective, we are not aware of any existing methodology that tackles this challenging case of high-dimensional priors without a low-dimensional structure, coupled with high-dimensional log-likelihoods with an active subspace, where the tight computational budget does not allow for a good estimation of the active subspace. From a Bayesian optimization perspective, we are not aware of any previous work on multi-objective HD-BO with random embeddings, for a relatively large number of objective functions ($n_{RML} = 20$). As demonstrated in the experiments arising from various domains, the methodology can outperform alternative gradient-free optimization methods and has potential to be used in many real-world applications. For future work, as discussed in Section 5.6, we would like to consider a larger class of Bayesian inverse problems by removing the active subspace assumption, via an alternative multi-objective HD-BO methodology that does not require this assumption. One such example is MORBO [42], which uses local GP models in high-dimensions. It would be interesting to see if MORBO can be extended to a larger number of objective functions (e.g. 20), in order to obtain a significant number of approximate posterior samples via RML.

Finally, in Chapter 6, we have shown that the Bayesian neural network (BNN) parametrization for Gaussian random field (GRF) priors that was recently introduced in [143] can potentially be useful for solving inverse problems with Darcy flow simulators via Ensemble Kalman Inversion (EKI). For future work, we would like to find a useful hierarchical EKI parametrization based on the BNN parameters $\{((\sigma^2_{w_l}), (\sigma^2_{b_l}))_{l=1}^{L+1}\}$ (57), although our attempts so far were not particularly encouraging. Furthermore, we have demonstrated that for the GP emulation approach within EKI, the random design strategy for selecting the training points of the original work [91] can be significantly improved by the experimental design from Algorithm 8, which is adapted from the History Matching (HM) work [59]. For future work, we would like to investigate whether this observation translates to other Bayesian inverse problems, apart from the groundwater application considered here. We will also consider other GP training designs within EKI, such as the alternative strategy [176] from HM.

BIBLIOGRAPHY

[1] mogp_emulator (python package). https://mogp-emulator.readthedocs.io/en/master/implementation/DimensionReduction.html/. Accessed: 2021-01-22.

[2] sliced (python package). https://joshloyal.github.io/sliced/. Accessed: 2021-01-18.

[3] Leen Alawieh, Jonathan Goodman, and John B. Bell. Iterative construction of gaussian process surrogate models for bayesian inference. *Journal of Statistical Planning and Inference*, 207:55–72, jul 2020.

[4] Smith Arauco C, Jose Castro, Júlia Potratz, Alexandre Emerick, and Marco Pacheco. Recent developments combining ensemble smoother and deep generative networks for facies history matching. *Computational Geosciences*, On-line First, 02 2021.

[5] Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *International Conference on Learning Representations*, 2019.

[6] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[7] Yuming Ba, Jana de Wiljes, Dean S. Oliver, and Sebastian Reich. Randomized maximum likelihood based posterior sampling. *Computational Geosciences*, 26:217–239, 2021.

[8] Francis Bach. Breaking the Curse of Dimensionality with Convex Neural Networks. *Journal of Machine Learning Research*, 18(19):1–53, December 2014.

[9] François Bachoc. Cross Validation and Maximum Likelihood estimation of hyper-parameters of Gaussian processes with model misspecification. *Computational Statistics and Data Analysis*, 66:55–69, October 2013.

[10] Johnathan M. Bardsley, Antti Solonen, Heikki Haario, and Marko Laine. Randomize-then-optimize: A method for sampling from posterior distributions in nonlinear inverse problems. *SIAM Journal on Scientific Computing*, 36(4):A1895–A1910, 2014.

[11] Joakim Beck and Serge Guillas. Sequential design with mutual information for computer experiments (mice): Emulation of a tsunami model. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):739–766, 2016.

[12] A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer US, 2011.

[13] Alexandros Beskos, Mark Girolami, Shiwei Lan, Patrick E. Farrell, and Andrew M. Stuart. Geometric MCMC for infinite-dimensional inverse problems. *Journal of Computational Physics*, 335:327–351, apr 2017.

[14] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo, 2017.

[15] Ilias Bilionis and Nicholas Zabaras. Solution of inverse problems with limited forward solver evaluations: A bayesian perspective. *Inverse Problems*, 30, 01 2014.

[16] Mickaël Binois. *Uncertainty quantification on pareto fronts and high-dimensional strategies in bayesian optimization, with applications in multi-objective automotive design*. Theses, Ecole Nationale Supérieure des Mines de Saint-Etienne, December 2015.

[17] Edwin V Bonilla, Kian Chai, and Christopher Williams. Multi-task gaussian process prediction. In J. Platt, D. Koller, Y. Singer, and

S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[18] Chris J.C. Burges. Dimension reduction: A guided tour. *Foundations and Trends in Machine Learning*, January 2010.

[19] Difeng Cai and Panayot S. Vassilevski. Eigenvalue problems for exponential-type kernels. *Computational Methods in Applied Mathematics*, 20(1):61–78, 2020.

[20] Coralia Cartis, Jan Fiala, Benjamin Marteau, and Lindon Roberts. Improving the flexibility and robustness of model-based derivative-free optimization solvers. *ACM Trans. Math. Softw.*, 45(3), aug 2019.

[21] Neil K. Chada, Marco A. Iglesias, Shuai Lu, and Frank Werner. On a dynamic variant of the iteratively regularized gauss-newton method with sequential data, 2022.

[22] Neil K Chada, Marco A Iglesias, Lassi Roininen, and Andrew M Stuart. Parameterizations for ensemble kalman inversion. *Inverse Problems*, 34(5):055009, apr 2018.

[23] Louis H.Y Chen. An inequality for the multivariate normal distribution. *Journal of Multivariate Analysis*, 12(2):306–315, 1982.

[24] Yan Chen and Dean Oliver. Levenberg-marquardt forms of the iterative ensemble smoother for efficient history matching and uncertainty quantification. *Computational Geosciences*, 17, 08 2013.

[25] Yan Chen and Dean S. Oliver. Ensemble randomized maximum likelihood method as an iterative ensemble smoother. *Mathematical Geosciences*, 44:1–26, 2011.

[26] Krzysztof Choromanski, Mark Rowland, and Adrian Weller. The unreasonable effectiveness of structured random orthogonal embeddings. In *Proceedings of the 31st International Conference on Neural In-*

formation Processing Systems, NIPS'17, page 218–227, Red Hook, NY, USA, 2017. Curran Associates Inc.

[27] P Constantine and R Howard. Active subspaces data sets. `https://github.com/paulcon/as-data-sets`.

[28] Paul Constantine and David Gleich. Computing active subspaces with monte carlo, 2014.

[29] Paul G. Constantine, Zachary del Rosario, and Gianluca Iaccarino. Many physical laws are ridge functions, 2016.

[30] 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, jan 2014.

[31] 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, jan 2016.

[32] R. Dennis Cook. Graphics for regressions with a binary response. *Journal of the American Statistical Association*, 91(435):983–992, 1996.

[33] R.D. Cook. *Regression Graphics: Ideas for Studying Regressions Through Graphics*. Wiley Series in Probability and Statistics. Wiley, 2009.

[34] S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White. MCMC methods for functions: Modifying old algorithms to make them faster. *Statistical Science*, 28(3), aug 2013.

[35] Sam Coveney, Cesare Corrado, Jeremy E. Oakley, Richard D. Wilkinson, Steven A. Niederer, and Richard H. Clayton. Bayesian calibration of electrophysiology models using restitution curve emulators. *Frontiers in Physiology*, 12, 2021.

[36] D. Crevillén-García, R.D. Wilkinson, A.A. Shah, and H. Power. Gaussian process modelling for uncertainty quantification in convectively-enhanced dissolution processes in porous media. *Advances in Water Resources*, 99:1–14, 2017.

[37] T Cui, J Martin, Y M Marzouk, A Solonen, and A Spantini. Likelihood-informed dimension reduction for nonlinear inverse problems. *Inverse Problems*, 30(11):114015, oct 2014.

[38] Tiangang Cui and Xin T. Tong. A unified performance analysis of likelihood-informed subspace methods, 2021.

[39] Sihui Dai, Jialin Song, and Yisong Yue. Multi-task bayesian optimization via gaussian process upper confidence bound. 2020.

[40] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, jan 2003.

[41] Masoumeh Dashti and Andrew M. Stuart. *The Bayesian Approach to Inverse Problems*, pages 311–428. Springer International Publishing, Cham, 2017.

[42] Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Multi-objective bayesian optimization over high-dimensional search spaces. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

[43] Ian Delbridge, David Bindel, and Andrew Gordon Wilson. Randomly projected additive gaussian processes for regression. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[44] Paul Diaz, Paul Constantine, Kelsey Kalmbach, Eric Jones, and Stephen Pankavich. A modified seir model for the spread of ebola

in western africa and metrics for resource allocation. *Applied Mathematics and Computation*, 324:141–155, 2018.

[45] Alexandre Emerick and Albert Reynolds. Ensemble smoother with multiple data assimilation. *Computers Geosciences*, 55:3–15, 06 2013.

[46] Alexandre Emerick and Albert Reynolds. Investigation of the sampling performance of ensemble-based methods with a simple reservoir model. *Computational Geosciences*, 17, 04 2013.

[47] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[48] L.C. Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, 2010.

[49] Geir Evensen. Analysis of iterative ensemble smoothers for solving inverse problems. *Computational Geosciences*, 22, 06 2018.

[50] Jean-Jacques Forneron and Serena Ng. A Likelihood-Free Reverse Sampler of the Posterior Distribution. In Gloria GonzÁlez-Rivera, R. Carter Hill, and Tae-Hwy Lee, editors, *Essays in Honor of Aman Ullah*, volume 36 of *Advances in Econometrics*, pages 389–415. Emerald Publishing Ltd, 2016.

[51] Joel N Franklin. Well-posed stochastic extensions of ill-posed linear problems. *Journal of Mathematical Analysis and Applications*, 31(3):682–716, 1970.

[52] Peter I. Frazier. A Tutorial on Bayesian Optimization. *arXiv e-prints*, page arXiv:1807.02811, July 2018.

[53] Kenji Fukumizu, Francis Bach, and Michael Jordan. Kernel dimensionality reduction for supervised learning. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003.

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

[55] G. Gao and Albert Reynolds. Quantifying uncertainty for the punq-s3 problem in a bayesian setting with rml and enkf. *SPE Journal*, 11:506–515, 12 2006.

[56] A. Garbuno-Inigo, F.A. DiazDelaO, and K.M. Zuev. Gaussian process hyper-parameter estimation using parallel asymptotically independent markov sampling. *Computational Statistics Data Analysis*, 103:367–383, 2016.

[57] A. Garbuno-Inigo, Franca Hoffmann, Wuchen Li, and Andrew M. Stuart. Interacting langevin diffusions: Gradient structure and ensemble kalman sampler. *SIAM J. Appl. Dyn. Syst.*, 19:412–441, 2020.

[58] Alfredo Garbuno-Inigo, F. A. DiazDelaO, and Konstantin M. Zuev. Transitional annealed adaptive slice sampling for gaussian process hyper-parameter estimation. *International Journal for Uncertainty Quantification*, 6(4):341–359, 2016.

[59] Alfredo Garbuno-Inigo, F. Alejandro DiazDelaO, and Konstantin M. Zuev. History matching with probabilistic emulators and active learning, 2020.

[60] Alfredo Garbuno-Inigo, Nikolas Nüsken, and Sebastian Reich. Affine invariant interacting langevin dynamics for bayesian inference. *SIAM Journal on Applied Dynamical Systems*, 19(3):1633–1658, 2020.

[61] Jacob R. Gardner, Geoff Pleiss, David Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 7587–7597, Red Hook, NY, USA, 2018. Curran Associates Inc.

[62] Roman Garnett, Michael A. Osborne, and Philipp Hennig. Active learning of linear embeddings for gaussian processes. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, page 230–239, Arlington, Virginia, USA, 2014. AUAI Press.

[63] Raphaël Gautier, Piyush Pandita, Sayan Ghosh, and Dimitri Mavris. A fully bayesian gradient-free supervised dimension reduction method using gaussian processes. *International Journal for Uncertainty Quantification*, 12(2), 2022.

[64] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Reproducing kernel hilbert space, mercer's theorem, eigenfunctions, nyström method, and use of kernels in machine learning: Tutorial and survey, 2021.

[65] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Sufficient dimension reduction for high-dimensional regression and low-dimensional embedding: Tutorial and survey, 2021.

[66] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When do neural networks outperform kernel methods? *Advances in Neural Information Processing Systems*, 33:14820–14830, 2020.

[67] Iraklis Giannakis, Antonios Giannopoulos, and Craig Warren. A machine learning-based fast-forward solver for ground penetrating radar with application to full-waveform inversion. *IEEE Transactions on Geoscience and Remote Sensing*, 57(7):4417–4426, 2019.

[68] James M. Gilbert, Jennifer L. Jefferson, Paul G. Constantine, and Reed M. Maxwell. Global spatial sensitivity of runoff to subsurface permeability using the active subspace method. *Advances in Water Resources*, 92:30–42, 2016.

[69] David Ginsbourger and Cedric Schärer. Fast calculation of gaussian process multiple-fold cross-validation residuals and their covariances, 2021.

[70] Andrew Glaws, Paul G. Constantine, and R. Dennis Cook. Inverse regression for ridge recovery II: Numerics. *arXiv e-prints*, page arXiv:1802.01541, February 2018.

[71] 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.

[72] Andrew T. Glaws, Paul G. Constantine, and R. Dennis Cook. Inverse regression for ridge recovery: A data-driven approach for parameter reduction in computer experiments. *arXiv e-prints*, page arXiv:1702.02227, February 2017.

[73] Hwan Goh, Sheroze Sheriffdeen, Jonathan Wittmer, and Tan Bui-Thanh. Solving bayesian inverse problems via variational autoencoders. In Joan Bruna, Jan Hesthaven, and Lenka Zdeborova, editors, *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*, volume 145 of *Proceedings of Machine Learning Research*, pages 386–425. PMLR, 16–19 Aug 2022.

[74] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19, 2006.

[75] Arthur Gretton, Alexander Smola, Olivier Bousquet, Ralf Herbrich, Andrei Belitski, Mark Augath, Yusuke Murayama, Jon Pauls, Bernhard Schölkopf, and Nikos Logothetis. Kernel constrained covariance for dependence measurement. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, pages 112–119. PMLR, 06–08 Jan 2005. Reissued by PMLR on 30 March 2021.

[76] Yaqing Gu and Dean S Oliver. History matching of the punq-s3 reservoir model using the ensemble kalman filter. In *SPE Annual Technical Conference and Exhibition*. OnePetro, 2004.

[77] Rajarshi Guhaniyogi and D. Dunson. Compressed gaussian process for manifold regression. *J. Mach. Learn. Res.*, 17:69:1–69:26, 2016.

[78] Hamidreza Hamdi, Ivo Couckuyt, Mario Costa Sousa, and Tom Dhaene. Gaussian processes for history-matching: application to an unconventional gas reservoir. *Computational Geosciences*, 21, 04 2017.

[79] Martin Hanke. A regularizing levenberg - marquardt scheme, with applications to inverse groundwater filtration problems. *Inverse Problems*, 13(1):79–95, feb 1997.

[80] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019.

[81] Philip B. Holden, Neil Robert Edwards, J. S. Hensman, and Richard D. Wilkinson. Abc for climate: Dealing with expensive simulators. *Handbook of Approximate Bayesian Computation*, 2018.

[82] Marco Iglesias and Yuchen Yang. Adaptive regularisation for ensemble kalman inversion. *Inverse Problems*, 37, 12 2020.

[83] Marco A Iglesias. Iterative regularization for ensemble data assimilation in reservoir models. *Computational Geosciences*, 19(1):177–212, 2015.

[84] Marco A Iglesias. A regularizing iterative ensemble kalman method for PDE-constrained inverse problems. *Inverse Problems*, 32(2):025002, jan 2016.

[85] Marco A Iglesias, Kody J H Law, and Andrew M Stuart. Ensemble kalman methods for inverse problems. *Inverse Problems*, 29(4):045001, mar 2013.

[86] Marco A Iglesias, Yulong Lu, and Andrew M Stuart. A bayesian level set method for geometric inverse problems. *Interfaces and free boundaries*, 18(2):181–217, 2016.

[87] Marco A. Iglesias, Minho Park, and Michael V. Tretyakov. Bayesian inversion in resin transfer molding. *Inverse Problems*, 2018.

[88] Borislav Ikonomov and Michael U Gutmann. Robust optimisation monte carlo. In *International Conference on Artificial Intelligence and Statistics*, pages 2819–2829. PMLR, 2020.

[89] Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. What are bayesian neural network posteriors really like? In *International conference on machine learning*, pages 4629–4640. PMLR, 2021.

[90] William Johnson and Joram Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 26:189–206, 01 1984.

[91] Lei Ju, Jiangjiang Zhang, Long Meng, Laosheng Wu, and Lingzao Zeng. An adaptive gaussian process-based iterative ensemble smoother for data assimilation. *Advances in Water Resources*, 115:125–135, 03 2018.

[92] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences, 2018.

[93] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. Query efficient posterior estimation in scientific experiments via bayesian active learning. *Artificial Intelligence*, 243:45–56, 2017.

[94] Marc C. Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63, 2001.

[95] Ilja Klebanov, Ingmar Schuster, and T. J. Sullivan. A rigorous theory of conditional mean embeddings. *SIAM Journal on Mathematics of Data Science*, 2(3):583–606, jan 2020.

[96] Vidhi Lalchand and Carl Edward Rasmussen. Approximate inference for fully bayesian gaussian process regression. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–12. PMLR, 2020.

[97] Hans Petter Langtangen and Svein Linge. *Finite difference computing with PDEs: a modern software approach*. Springer Nature, 2017.

[98] Minyong R Lee. *Prediction and Dimension Reduction Methods in Computer Experiments*. PhD thesis, 2017.

[99] Ben Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional bayesian optimization. *Advances in neural information processing systems*, 33:1546–1558, 2020.

[100] B. Li. *Sufficient Dimension Reduction: Methods and Applications with R*. Chapman & Hall/CRC Monographs on Statistics and Applied Probability. CRC Press, 2018.

[101] Ker-Chau Li. Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, 86(414):316–327, 1991.

[102] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *MATHEMATICAL PROGRAMMING*, 45:503–528, 1989.

[103] Xiaoyu Liu and Serge Guillas. Dimension reduction for emulation: application to the influence of bathymetry on tsunami heights. *SIAM/ASA Journal on Uncertainty Quantification*, 5, 03 2016.

[104] Yimin Liu, Wenyue Sun, and Louis Durlofsky. A deep-learning-based geological parameterization for history matching complex models. *Mathematical Geosciences*, 51, 03 2019.

[105] Tyson Loudon and Stephen Pankavich. Mathematical analysis and dynamic active subspaces for a long term model of hiv. *Mathematical Biosciences and Engineering*, 14(3):709–733, 2017.

[106] Lisandro Lovisolo and Eduardo da Silva. Uniform distribution of points on a hyper-sphere with applicationsto vector bit-plane encoding. *IEE Proceedings - Vision Image and Signal Processing*, pages 187 – 193, 07 2001.

[107] 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.

[108] Pulong Ma, Anirban Mondal, Bledar A Konomi, Jonathan Hobbs, Joon Jin Song, and Emily L Kang. Computer model emulation with high-dimensional functional output in large-scale observing system uncertainty experiments. *Technometrics*, 64(1):65–79, 2022.

[109] Yanyuan Ma and Liping Zhu. A review on dimension reduction. *International Statistical Review / Revue Internationale de Statistique*, 81(1):134–150, 2013.

[110] Yingbo Ma, Vaibhav Dixit, Michael J Innes, Xingjian Guo, and Chris Rackauckas. A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9. IEEE, 2021.

[111] Mark McLeod, Stephen Roberts, and Michael A Osborne. Optimization, fast and slow: optimally switching between local and bayesian optimization. In *International Conference on Machine Learning*, pages 3443–3452. PMLR, 2018.

[112] Ted Meeds and Max Welling. Optimization monte carlo: Efficient and embarrassingly parallel likelihood-free inference. *Advances in Neural Information Processing Systems*, 28, 2015.

[113] K. L. Mengersen and R. L. Tweedie. Rates of convergence of the hastings and metropolis algorithms. *The Annals of Statistics*, 24(1):101–121, 1996.

[114] Sean Meyn and Richard L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, USA, 2nd edition, 2009.

[115] Sebastian Mika, Bernhard Schölkopf, Alex Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel pca and de-noising in feature spaces. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11. MIT Press, 1998.

[116] Vladimir Alekseevich Morozov. *Methods for solving incorrectly posed problems*. Springer Science & Business Media, 2012.

[117] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.

[118] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for Bayesian optimization in embedded subspaces. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4752–4761. PMLR, 09–15 Jun 2019.

[119] Jelani Nelson and Huy L NguyÅn. Sparsity lower bounds for dimensionality reducing maps. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 101–110, 2013.

[120] Jelani Nelson and Huy L Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 ieee 54th annual symposium on foundations of computer science*, pages 117–126. IEEE, 2013.

[121] Lorenzo Noci, Gregor Bachmann, Kevin Roth, Sebastian Nowozin, and Thomas Hofmann. Precise characterization of the prior predictive distribution of deep relu networks. *Advances in Neural Information Processing Systems*, 34:20851–20862, 2021.

[122] 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.

[123] Dean S. Oliver. Metropolized randomized maximum likelihood for improved sampling from multimodal distributions. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):259–277, 2017.

[124] Dean S Oliver, Nanqun He, and Albert C Reynolds. Conditioning permeability fields to pressure data. In *ECMOR V-5th European conference on the mathematics of oil recovery*, pages cp–101. European Association of Geoscientists & Engineers, 1996.

[125] Dean S. Oliver, Albert C. Reynolds, and Ning Liu. *Inverse Theory for Petroleum Reservoir Characterization and History Matching*. Cambridge University Press, 2008.

[126] Mario Parente, Daniel Bittner, Steven Mattis, Gabriele Chiogna, and Barbara Wohlmuth. Bayesian calibration and sensitivity analysis for a karst aquifer model using active subspaces. *Water Resources Research*, 55, 08 2019.

[127] Mario Teixeira Parente. *Active Subspaces in Bayesian Inverse Problems*. PhD thesis, Technische Universität München, 2020.

[128] 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.

[129] Thomas Pinder, Christopher Nemeth, and David Leslie. Stein variational gaussian processes, 2020.

[130] Allan Pinkus. *Ridge Functions*. Cambridge Tracts in Mathematics. Cambridge University Press, 2015.

[131] Geoff Pleiss, Jacob Gardner, Kilian Weinberger, and Andrew Gordon Wilson. Constant-time predictive distributions for gaussian processes. In *International Conference on Machine Learning*, pages 4114–4123. PMLR, 2018.

[132] Michael JD Powell. A view of algorithms for optimization without derivatives. *Mathematics Today-Bulletin of the Institute of Mathematics and its Applications*, 43(5):170–174, 2007.

[133] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[134] Dushhyanth Rajaram. *Methods for Construction of Surrogates For Computationally Expensive High-Dimensional Problems*. PhD thesis, Georgia Institute of Technology, 2020.

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

[136] Sebastian Reich and Simon Weissmann. Fokker–planck particle systems for bayesian inference: Computational approaches. *SIAM/ASA Journal on Uncertainty Quantification*, 9(2):446–482, 2021.

[137] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005.

[138] James M. Salter, Daniel B. Williamson, John Scinocca, and Viatcheslav Kharin. Uncertainty quantification for computer models with spatial output using calibration-optimal bases. *Journal of the American Statistical Association*, 114(528):1800–1814, mar 2019.

[139] Thomas J Santner, Brian J Williams, William I Notz, and Brain J Williams. *The design and analysis of computer experiments*, volume 1. Springer, 2003.

[140] Daniel Sanz-Alonso, Andrew M. Stuart, and Armeen Taeb. Inverse problems and data assimilation, 2018.

[141] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[142] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. *Kernel Principal Component Analysis*, page 327–352. MIT Press, Cambridge, MA, USA, 1999.

[143] Torben Sell and Sumeetpal S. Singh. Trace-class gaussian priors for bayesian learning of neural networks with mcmc, 2020.

[144] Pranay Seshadri, Shaowu Yuchi, and Geoffrey T Parks. Dimension reduction via gaussian ridge functions. *SIAM/ASA Journal on Uncertainty Quantification*, 7(4):1301–1322, 2019.

[145] Vaishaal Shankar, Alex Fang, Wenshuo Guo, Sara Fridovich-Keil, Jonathan Ragan-Kelley, Ludwig Schmidt, and Benjamin Recht. Neural kernels without tangents. In *International Conference on Machine Learning*, pages 8614–8623. PMLR, 2020.

[146] J. Shawe-Taylor, C.K.I. Williams, N. Cristianini, and J. Kandola. On the eigenspectrum of the gram matrix and the generalization error of kernel-pca. *IEEE Transactions on Information Theory*, 51(7):2510–2522, 2005.

[147] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[148] Chris Sherlock and Gareth Roberts. Optimal scaling of the random walk Metropolis on elliptically symmetric unimodal targets. *Bernoulli*, 15(3):774 – 798, 2009.

[149] Ali Siahkoohi, Gabrio Rizzuti, Mathias Louboutin, Philipp Witte, and Felix Herrmann. Preconditioned training of normalizing flows for variational inference in inverse problems. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2021.

[150] Per Sidén and Fredrik Lindsten. Deep Gaussian Markov random fields. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8916–8926. PMLR, 13–18 Jul 2020.

[151] Fergus Simpson, Vidhi Lalchand, and Carl Edward Rasmussen. Marginalised gaussian processes with nested sampling. *Advances in Neural Information Processing Systems*, 34:13613–13625, 2021.

[152] Jan-arild Skjervheim and Geir Evensen. An ensemble smoother for assisted history matching. *SPE 141929, presented at the SPE Reservoir Simulation Symposium*, 2, 02 2011.

[153] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In Marcus Hutter, Rocco A. Servedio, and Eiji Takimoto, editors, *Algorithmic Learning Theory*, pages 13–31, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[154] Edward Snelson and Zoubin Ghahramani. Variable noise and dimensionality reduction for sparse gaussian processes. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'06, page 461–468, Arlington, Virginia, USA, 2006. AUAI Press.

[155] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 1015–1022, Madison, WI, USA, 2010. Omnipress.

[156] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.

[157] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[158] Charles J. Stone. Optimal Rates of Convergence for Nonparametric Estimators. *The Annals of Statistics*, 8(6):1348 – 1360, 1980.

[159] A. M. Stuart. Inverse problems: A bayesian perspective. *Acta Numerica*, 19:451–559, 2010.

[160] Timur Takhtaganov and Juliane Müller. Adaptive Gaussian process surrogates for Bayesian inference. *arXiv e-prints*, page arXiv:1809.10784, September 2018.

[161] Meng Tang, Yimin Liu, and Louis J. Durlofsky. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *Journal of Computational Physics*, 413:109456, July 2020.

[162] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, USA, 2004.

[163] Liang Tian, Richard Wilkinson, Zhibing Yang, Henry Power, Fritjof Fagerlund, and Auli Niemi. Gaussian process emulators for quantifying uncertainty in co2 spreading predictions in heterogeneous media. *Computers & Geosciences*, 105:113–119, 2017.

[164] Michalis Titsias and Miguel Lazaro-Gredilla. Variational inference for mahalanobis distance metrics in gaussian process regression. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[165] Cédric Travelletti, David Ginsbourger, and Niklas Linde. Uncertainty quantification and experimental design for large-scale linear inverse problems under gaussian process priors, 2021.

[166] 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, September 2016.

[167] Elske van der Vaart, Dennis Prangle, and Richard M Sibly. Taking error into account when fitting models using approximate bayesian computation. *Ecological applications*, 28(2):267–274, 2018.

[168] Ian Vernon, Michael Goldstein, and Richard Bower. Galaxy formation: Bayesian history matching for the observable universe. *Statistical Science*, 29(1), feb 2014.

[169] Francesco Vivarelli and Christopher Williams. Discovering hidden features with gaussian processes regression. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11. MIT Press, 1999.

[170] Hongqiao Wang and Jinglai Li. Adaptive gaussian process approximation for bayesian inference with expensive likelihood functions. *Neural computation*, 30(11):3072–3094, 2018.

[171] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando De Feitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

[172] Larry Wasserman. *All of nonparametric statistics*. Springer Science & Business Media, 2006.

[173] Florian Wenzel, Kevin Roth, Bastiaan Veeling, Jakub Swiatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the Bayes posterior in deep neural networks really? In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10248–10259. PMLR, 13–18 Jul 2020.

[174] Richard Wilkinson. Accelerating abc methods using gaussian processes. In *Artificial Intelligence and Statistics*, pages 1015–1023. PMLR, 2014.

[175] Daniel Williamson, Michael Goldstein, Lesley Allison, Adam Blaker, Peter Challenor, Laura Jackson, and Kuniko Yamazaki. History matching for exploring and reducing climate model parameter space using observations and a large perturbed physics ensemble. *Climate Dynamics*, 41:1703–1729, 10 2013.

[176] Daniel Williamson and Ian Vernon. Efficient uniform designs for multi-wave computer experiments. *arXiv preprint arXiv:1309.3520*, 2013.

[177] Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International conference on machine learning*, pages 1775–1784. PMLR, 2015.

[178] Andrew Gordon Wilson, Christoph Dann, and Hannes Nickisch. Thoughts on massively scalable gaussian processes, 2015.

[179] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

[180] James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for bayesian optimization. *Advances in neural information processing systems*, 31, 2018.

[181] Nathan Wycoff, Mickael Binois, and Stefan M Wild. Sequential learning of active subspaces. *Journal of Computational and Graphical Statistics*, 30(4):1224–1237, 2021.

[182] Yun Yang and David B Dunson. Bayesian manifold regression. *The Annals of Statistics*, 44(2):876–905, 2016.

[183] Ivan Yashchuk. Bringing {pde}s to {jax} with forward and reverse modes automatic differentiation. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[184] Olivier Zahm, Paul G Constantine, Clémentine Prieur, and Youssef M Marzouk. Gradient-based dimension reduction of multivariate vector-valued functions. *SIAM Journal on Scientific Computing*, 42(1):A534–A558, 2020.

[185] Olivier Zahm, Tiangang Cui, Kody Law, Alessio Spantini, and Youssef Marzouk. Certified dimension reduction in nonlinear bayesian inverse problems. *Mathematics of Computation*, 91(336):1789–1835, 2022.

[186] Jacob Zavatone-Veth and Cengiz Pehlevan. Exact marginal prior distributions of finite bayesian neural networks. *Advances in Neural Information Processing Systems*, 34:3364–3375, 2021.

[187] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2020. `https://d2l.ai`.

[188] Hao Zhang. Inconsistent estimation and asymptotically equal interpolations in model-based geostatistics. *Journal of the American Statistical Association*, 99(465):250–261, 2004.

[189] Miao Zhang, Huiqi Li, and Steven Su. High dimensional bayesian optimization via supervised dimension reduction. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 4292–4298. AAAI Press, 2019.

[190] Shandian Zhe, Wei Xing, and Robert M. Kirby. Scalable high-order gaussian process regression. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2611–2620. PMLR, 16–18 Apr 2019.

[191] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, aug 2018.

[192] M. Munoz Zuniga, A. Murangira, and T. Perdrizet. Structural reliability assessment through surrogate based importance sampling with dimension reduction. *Reliability Engineering System Safety*, 207:107289, 2021.