Version of Record: https://www.sciencedirect.com/science/article/pii/S0045782521004783
Manuscript_0874350b103bee7ef11fc2d4f30772c4

Bayesian learning of orthogonal embeddings for multi-fidelity Gaussian Processes

Panagiotis Tsilifis[a,*], Piyush Pandita[a], Sayan Ghosh[a], Valeria Andreoli[b], Thomas Vandeputte[b], Liping Wang[a]

[a] *Probabilistic Design & Optimization Group, General Electric Research, Niskayuna, NY 12309, USA*
[b] *Aerodynamics & Computational Fluid Dynamics Group, General Electric Research, Niskayuna, NY 12309, USA*

Abstract

Uncertainty propagation in complex engineering systems often poses significant computational challenges related to modeling and quantifying probability distributions of model outputs, as those emerge as the result of various sources of uncertainty that are inherent in the system under investigation. Gaussian Processes regression (GPs) is a robust meta-modeling technique that allows for fast model prediction and exploration of response surfaces. Multi-fidelity variations of GPs further leverage information from cheap and low fidelity model simulations in order to improve their predictive performance on the high fidelity model. In order to cope with the high volume of data required to train GPs in high dimensional design spaces, a common practice is to introduce latent design variables that are typically projections of the original input space to a lower dimensional subspace, and therefore substitute the problem of learning the initial high dimensional mapping, with that of training a GP on a low dimensional space. In this paper, we present a Bayesian approach to identify optimal transformations that map the input points to low dimensional latent variables. The "projection" mapping consists of an orthonormal matrix that is considered a priori unknown and needs to be inferred jointly with the GP parameters, conditioned on the available training data. The proposed Bayesian inference scheme relies on a two-step iterative algorithm that samples from the marginal posteriors of the GP parameters and the projection matrix respectively, both using Markov Chain Monte Carlo (MCMC) sampling. In order to take into account the orthogonality constraints imposed on the orthonormal projection matrix, a Geodesic Monte Carlo sampling algorithm is employed, that is suitable for exploiting probability measures on manifolds. We extend the proposed framework to multi-fidelity models using GPs including the scenarios of training multiple outputs together. We validate our framework on three synthetic problems with a known lower-dimensional subspace. The benefits of our proposed framework, are illustrated on the computationally challenging aerodynamic optimization of a last-stage blade for an industrial gas turbine, where we study the effect of an 85-dimensional shape parameterization of a three-dimensional airfoil on two output quantities of interest, specifically on the aerodynamic efficiency and the degree of reaction.

Keywords: Gaussian Process Regression, Multi-fidelity simulations, dimension reduction, Geodesic Monte Carlo, Bayesian Inference, Uncertainty Propagation

1. Introduction

In addressing reliability and design optimization challenges in modern engineering and manufacturing processes, the need for an increasing accuracy in the predictive capabilities of the analysis codes has been

*Corresponding author
Email addresses: panagiotis.tsilifis@ge.com (Panagiotis Tsilifis), piyush.pandita@ge.com (Piyush Pandita), sayan.ghosh1@ge.com (Sayan Ghosh), valeria.andreoli@ge.com (Valeria Andreoli), vandeput@ge.com (Thomas Vandeputte), wangli@ge.com (Liping Wang)

© 2021 published by Elsevier. This manuscript is made available under the Elsevier user license
https://www.elsevier.com/open-access/userlicense/1.0/

of paramount importance. Taking into account the presence of aleatoric uncertainties resulting from the severe complexity of the physical system under investigation and the limited knowledge available to the experimenter, simulation-based uncertainty quantification (UQ) tasks can result in computational costs that are unacceptable in practice [1, 2, 3].

Metamodeling techniques [4] have gained increasing popularity in recent years, as they provide a way to alleviate the computational burden associated with quantifying uncertainties in computer model outputs and inverse modeling [5, 6]. The key idea is to replace the expensive computer solver with a surrogate that is cheap to evaluate and at the same time it maintains high predictive accuracy. These metamodels can consist of functional representations of the forward solver such as Polynomial Chaos [7, 8, 9], kernel-based methods [10, 11], relevance vector machines [12, 13] and neural networks [14, 15] or they can be nonparametric models such as Gaussian Processes [16, 17, 18].

Gaussian Process (GP) regression techniques allow us to quantify the epistemic uncertainty associated with the limited number or training data points by providing explicit Bayesian posterior updates on the error bars [19, 20]. Recent works have demonstrated their wide applicability for UQ analysis and probabilistic machine learning and have been exhaustively used for predicting solutions to differential equations, including local adaptations for detecting discontinuities [21], discovering governing equations [22] and constructing latent variable models for solving inverse problems [23]. Challenges associated with efficiently training an accurate GP model, include acquiring the training dataset necessary to guarantee the desired predictive performance and tuning the model's hyperparameters [24]. Collecting the data often requires a significant number of model simulations that increases exponentially as a function of the dimensionality of the input parameters and can easily become prohibitive due to computational budget limitations. On the other hand, training GPs using massive datasets, when available, poses numerical challenges that can result in poor performance. These are mainly associated with repeatedly performing covariance matrix inversions that can be extremely inaccurate and computer memory-demanding. Applying sparse techniques [25] and batch optimization [26] have only partially managed to address these issues.

An alternative approach involves leveraging a cheaper computer solver that provides with low-cost, yet less accurate observations to be used for training the GP, leading to formulations where the observable depends on more that one covariates. Such approaches, although already known to the geostatistics community as co-kriging [27], were formalized within the context of multi-fidelity simulations in the pioneering work of Kennedy & O'Hagan [28]. The auto-regressive scheme presented therein, decomposes the expensive simulation code as a sum of its cheap approximation and a discrepancy term, both modeled as independent Gaussian Processes. Techniques for training the model have been proposed [29] and involve learning the degree of correlation between the codes, while a model calibration variation of the scheme is also available [30]. Similarly to standard GP regression in single-fidelity settings, training the auto-regressive scheme includes inverting large, ill-conditioned covariance matrices, leading to numerical inaccuracies. Le Gratiet & Garnier [31] offered an elegant solution to the problem that consists of decoupling the information stemming from different levels of fidelity and thus simplifying the covariance matrix structure. These ideas found wide applicability in discovering high dimensional response surfaces arising from dynamical systems [32], model inversion [33], multi-fidelity Bayesian optimization [34] and uncertainty propagation in the big-data regime [35].

From a different perspective, an efficient strategy for constructing a GP metamodel would involve addressing input dimensionality by means of reducing the number of variables in the input design space. Standard ways of doing so include sensitivity analysis [36] and unsupervised learning methods that explore correlations of the input variables, such as PCA [37], kernel PCA [38] or even truncating Karhunen-Loève expansions [7]. More sophisticated approaches to dimensionality reduction in the context of GPs include applying embeddings on the input design space in order to obtain a low dimensional latent variable space. Several criteria for discovering linear embeddings have been proposed in the literature. Constantine in his seminal work [39, 40], used a gradient-based approach to exploit what is termed Active Subscape (AS). The key step in this approach is to map the input space using a matrix derived by an orthogonal decomposition of the covariance of the gradient vector of the observable quantity. Intuitively the approach is sound as it explores the directions along which the output quantity exhibits most of its variability. The major limitation however is that it is impractical in black-box simulations where gradient information is hardly available. An

extension of the AS approach to multi-fidelity contexts was also developed later on [41], where a cheaper simulator is used to enhance estimation of the covariance of the gradient vector, yet the challenge of accessing gradient information persists. Gradient-free approaches that learn orthogonal [42] or arbitrary [43] linear mappings on-the-fly, have also been developed. In these works, the matrices were inferred using MLE or the Laplace approximation of the posterior of the linear mapping, respectively. At last, attempts to reduce the dimensionality using nonlinear mappings have been successfully applied for Bayesian optimization [44], where the forward mapping to the latent variables is modeled using a feed-forward neural net while the original input reconstruction mapping is carried out using multi-output GPS.

The ultimate goal in this work is to combine dimensionality reduction principles with multi-fidelity simulations so as to manifest their benefits in a single context. We seek to simultaneously address numerical issues and computational limitations arising in the presence of high input dimensionality and expensive data acquisition procedures. We thus present a unified framework where the classic auto-regressive Gaussian Process (ARGP) scheme can be built on a linearly embedded subspace that can be learned while training the model. To the authors best knowledge, such an approach is presented for the first time. Specifically, we build on the Kennedy-O'Hagan autoregessive scheme for modeling expensive computer outputs using a multi-fidelity source of information. Although restricted in characterizing the successive levels of fidelity through linear dependence we opt out of recent nonlinear data fusion approaches [45], since training those models in a general setting currently poses enormous challenges due to the large number of hyperparameters that are present in the *deep* GP formulations, albeit elegant solutions are derived for data sets relying on nested design points. Regardless of the quality of their output as an approximation of the high fidelity solver, we assume that the different fidelity codes exhibit similar dependence on the input variables, thus intuitively it is natural to assume that a common linear embedding can be applied to all GPs at different levels of fidelity. A training procedure for tuning the hyperparameters of the model is presented where the ARGP parameters are estimated using either MLE or Bayesian methods and the orthogonal embedding is inferred using a fully Bayesian approach that relies on the Geodesic Monte Carlo algorithm [46] that has been particularly tailored for sampling orthogonal matrices [47]. Such an approach to learn linear embeddings is novel in the context of dimensionality reduction for GPs and clearly prevails previously presented techniques [42, 43] in that the full posterior of the orthogonal matrix can be exploited as opposed to a Laplace approximations while it relieves us of the computational burden caused by high-dimensional gradient-based optimization of the likelihood.

We structure this paper as follows: Section 2.1 presents the basic elements of the classic autoregressive Gaussian Process scheme, Sec. 2.2 presents the prior setting and predictive distributions and Sec. 2.3 discusses the model training approach. Section 3.1 then formulates the ARGP model defined on a linearly embedded subspace using an orthogonal projection, that is trained using the proposed algorithm presented in Sec. 3.2. Our numerical examples include two toy problems with known embeddings that are learned using observations from three levels of fidelity (Sec. 4.1 & 4.2) and a challenging three-dimensional airfoil optimization problem with a 85-dimensional input space where observations are available from a high- and a low-fidelity simulator.

2. Recursive Multi-fidelity Gaussian Processes

2.1. Autoregressive Gaussian Process model

We consider the following scenario where a hierarchy of s computer codes is available, say $\{z_i(\mathbf{x})\}_{i=1,\ldots,s}$, indexed by input vector $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^D$ where $\mathcal{X}$ is the design space consisting of all feasible inputs. The codes are in order of increasing fidelity from the cheapest one $z_1(\mathbf{x})$, to the most accurate one, $z_s(\mathbf{x})$. The key assumption as was first stated by Kennedy & O'Hagan [28] for any two consecutive levels of code $z_t(\cdot)$ and $z_{t-1}(\cdot)$, is that given observation of the low fidelity code $z_{t-1}(\cdot)$ at $\mathbf{x}$, nothing more can be learnt about $z_t(\mathbf{x})$ by observing $z_{t-1}(\mathbf{x}')$ at any $\mathbf{x}' \neq \mathbf{x}$. This translates to the Markov property

$$\operatorname{cov}\left[z_t(\mathbf{x}), z_{t-1}(\mathbf{x}')|z_{t-1}(\mathbf{x})\right] = 0, \tag{1}$$

for any $\mathbf{x}' \neq \mathbf{x}$. Based on this principle, we write the relation between any two consecutive codes using the autoregressive model

$$\begin{cases} z_t(\mathbf{x}) = \rho_{t-1}(\mathbf{x}) z_{t-1}(\mathbf{x}) + \delta_t(\mathbf{x}) + \epsilon_t(\mathbf{x}) \\ z_{t-1}(\mathbf{x}) \perp \delta_t(\mathbf{x}),\ z_{t-1}(\mathbf{x}) \perp \epsilon_t(\mathbf{x}),\ z_1(\mathbf{x}) \perp \epsilon_1(\mathbf{x}) \quad , \quad 1 < t \leq s \\ \delta_t(\mathbf{x}) \perp \epsilon_t(\mathbf{x}), \end{cases} \tag{2}$$

where "$\perp$" denotes statistical independence and therefore the discrepancy between successive levels is characterized by an independent correction term $\delta_t(\mathbf{x})$ and by the scaling coefficient ρ_{t-1} which captures the correlation between the models, as it satisfies

$$\rho_{t-1}(\mathbf{x}) = \frac{\text{cov}\left[z_t(\mathbf{x}), z_{t-1}(\mathbf{x})\right]}{\text{var}\left[z_{t-1}(\mathbf{x})\right]}. \tag{3}$$

As suggested in [28], in what follows we assume for simplicity that ρ_t is constant, although dependence on $\mathbf{x}$ has been modeled using regression functions and was shown to be worthwhile [48, 31]. At last $\epsilon_t(\mathbf{x})$ accounts for measurement noise in the obserbations at level t that is assumed to be independent of both z_{t-1} and δ_t.

2.2. Prior and predictive distributions

Prior to observing any outputs of the multi-fidelity codes, we assume that $z_1(\mathbf{x})$ and $\{\delta_t(\mathbf{x})\}_{t=2,\cdots,s}$ are Gaussian Processes

$$\begin{cases} z_1(\mathbf{x}) \sim \mathcal{GP}\left(0, \sigma_1^2 r_1(\mathbf{x}, \mathbf{x}')\right), \\ \delta_t(\mathbf{x}) \sim \mathcal{GP}\left(0, \sigma_t^2 r_t(\mathbf{x}, \mathbf{x}')\right), \quad 1 < t \leq s, \end{cases} \tag{4}$$

where σ_t^2, $t \leq 1, \ldots, s$ are scaling factors and $r_t(\mathbf{x}, \mathbf{x}')$ are covariance kernel functions. For the measurement noise we take

$$\epsilon_t(\mathbf{x}) \sim \mathcal{N}(0, \sigma_{\epsilon_t}^2), \quad \mathbf{x} \in \mathcal{X},\ 1 \leq t \leq s. \tag{5}$$

The kernel functions are modeled using the squared exponential kernel

$$r_t(\mathbf{x}, \mathbf{x}') = \exp\left[-\sum_{i=1}^{D} \frac{(x_i - x_i')^2}{\theta_{i,t}^2}\right], \tag{6}$$

where $\boldsymbol{\theta}_t = (\theta_{1,t}, \ldots, \theta_{D,t})^T$ are the lengthscales along the D dimensions. The choice of the covariance kernel depends primarily on the prior belief about the smoothness of the response surface (Ch.4, [24]).

Let now $D_t = \{\mathbf{x}_1^{(t)}, \ldots, \mathbf{x}_{n_t}^{(t)}\}$ be the experimental design set at level t that consists of n_t input points in $\mathcal{X}$, $t = 1, \ldots, s$ and $\mathcal{Z}_t := \mathcal{Z}_t(D_t) = (z_t(\mathbf{x}_1^{(t)}), \ldots, z_t(\mathbf{x}_{n_t}^{(t)}))^T$ be the observations of the t-th code z_t. We denote with $\boldsymbol{\phi}_t$ the parameters introduced at level t, so that $\boldsymbol{\phi}_1 := (\phi_1^1, \phi_1^2, \phi_1^3) = (\boldsymbol{\theta}_1, \sigma_1^2, \sigma_{\epsilon_1}^2)$ and $\boldsymbol{\phi}_t := (\phi_t^1, \ldots, \phi_t^4) = (\boldsymbol{\theta}_t, \rho_{t-1}, \sigma_t^2, \sigma_{\epsilon_t}^2)$, $1 < t \leq s$ and $\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_s)$. For a fixed set of parameter values $\boldsymbol{\phi}$ by stacking all observations to form a vector $\mathcal{Z} = \left(\mathcal{Z}_t^T, \ldots, \mathcal{Z}_s^T\right)^T$, we write the predictive distribution of the highest level of code $z_t(D^*)$ at any new set of test points $D^* = \{\mathbf{x}_i^*\}_{i=1}^{n^*}$, for $\mathbf{x}_i^* \in \mathcal{X}$, $i = 1, \ldots, n^*$ as

$$z_s(D^*) \big| \mathcal{Z}, \boldsymbol{\phi} \sim \mathcal{GP}(\mathbf{m}(D^*), \sigma_{Z_s}(D^*)) \tag{7}$$

where the predictive mean is given by

$$\mathbf{m}_{Z_s}(D^*) = t_s(D^*) V_s^{-1} \mathcal{Z} \tag{8}$$

and the predictive variance is

$$\sigma_{Z_s}^2(D^*) = \sigma_s^2(D^*) - t_s(D^*) V_s^{-1} t_s(D^*), \tag{9}$$

In the above expressions we have

$$V_s = \begin{bmatrix} V^{(1,1)} & \dots & V^{(1,s)} \\ \vdots & \ddots & \vdots \\ V^{(s,1)} & \dots & V^{(s,s)} \end{bmatrix} \tag{10}$$

where the diagonal block matrices are given by

$$V^{(t,t)} = \sigma_t^2 \left(R_t(D_t) + \sigma_{\epsilon_t} I\right) + \sigma_{t-1}^2 \rho_{t-1}^2 \left(R_{t-1}(D_t) + \sigma_{\epsilon_{t-1}}^2 I\right) + \cdots + \sigma_1^2 \left(\prod_{i=1}^{t-1} \rho_i^2\right) \left(R_1(D_t) + \sigma_{\epsilon_1}^2 I\right), \tag{11}$$

where $R_i(D_t)$, $i = 1, \dots, t$ is the correlation matrix with entries $r_i(\mathbf{x}, \mathbf{x}')$, $\mathbf{x}, \mathbf{x}' \in D_t$. The off-diagonal blocks are written

$$V^{(t,t')} = \left(\prod_{i=t}^{t'-1} \rho_i\right) V^{(t,t)}(D_t, D_{t'}), \quad 1 \le t < t' \le s, \tag{12}$$

with $V^{(t,t)}(D_t, D_{t'}) = \sigma_t^2 R_t(D_t, D_{t'}) + \cdots + \sigma_1^2 \left(\prod_{i=1}^{t-1} \rho_i^2\right) R_1(D_t, D_{t'})$. Analogously, $R_i(D_t, D_{t'})$ is the correlation matrix with entries $r_i(\mathbf{x}, \mathbf{x}')$, $\mathbf{x} \in D_t, \mathbf{x}' \in D_{t'}$. Further, the vector $t_s(D^*)$ is defined as $t_s(D^*) = (\bar{t}_1(\mathbf{x}, D_1)^T, \dots, \bar{t}_s(\mathbf{x}, D_s)^T)^T$, where

$$\bar{t}_t(D^*, D_t)^T = \rho_{t-1} \bar{t}_{t-1}(D^*, D_t)^T + \left(\prod_{i=t}^{s-1} \rho_i\right) \sigma_t^2 R_t(D^*, D_t), \quad 1 < t \le s, \tag{13}$$

and $\prod_{i=s}^{s-1} \rho_i = 1$ and $\bar{t}_1(D^*, D_1)^T = \left(\prod_{i=1}^{s-1} \rho_i\right) \sigma_1^2 R_1(D^*, D_1)$. At last, the variance $\sigma_{Z_s}^2$ is defined as

$$\sigma_s^2(D^*) = \sum_{t=1}^{s} \sigma_t^2 \left(\prod_{j=t}^{s-1} \rho_j^2\right) r_t(\mathbf{x}^*, \mathbf{x}^{*'}) \tag{14}$$

for $\mathbf{x}^*, \mathbf{x}^{*'} \in D^*$.

2.3. Estimating the model parameters

In order for the above predictive distribution to be of practical use, it is crucial to train the model by means of finding the optimal set of parameters $\boldsymbol{\phi}$. Pursuing a fully Bayesian approach to model training, although robust, it can be computationally challenging as the number of levels, and consequently the dimension of $\boldsymbol{\phi}$, increases. It can be therefore preferable in such cases to resort to more efficient strategies, such as maximum likelihood estimation (MLE). For the sake of completeness, we present below both approaches, that we use interchangeably in our numerical examples.

2.3.1. Maximum-likelihood estimation

Here we explore the possibility of training the model using maximum likelihood estimation (MLE). More specifically we seek to minimize the negative log-likelihood, that is to identify $\boldsymbol{\phi}^*$ such that

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \boldsymbol{\ell}(\boldsymbol{\phi}), \tag{15}$$

where

$$\boldsymbol{\ell}(\boldsymbol{\phi}) := -\log p(\mathcal{Z}|\boldsymbol{\phi}) = \frac{1}{2} \mathcal{Z}^T V_s(\boldsymbol{\phi})^{-1} \mathcal{Z} + \frac{1}{2} \log |V_s(\boldsymbol{\phi})| + \frac{N}{2} \log(2\pi), \tag{16}$$

writing $V_s(\boldsymbol{\phi})$ to emphasize the dependence of the covariance matrix on the parameters. Minimization of $\boldsymbol{\ell}$ can be performed using standard gradient based algorithms. The gradient of $\boldsymbol{\ell}$ with respect to any of its arguments is given by

$$\frac{\partial \boldsymbol{\ell}(\boldsymbol{\phi})}{\partial \phi_t^i} = -\frac{1}{2}\mathrm{tr}\left[\left\{V_s^{-1}\mathcal{Z}\left(V_s^{-1}\mathcal{Z}\right)^T - V_s^{-1}\right\}\frac{\partial V_s}{\partial \phi_t^i}\right]. \tag{17}$$

Note that a convenient simplification applies in the special case considered in [28, 48, 31], where the design points corresponding to the observations are nested, that is $D_t \subset D_{t-1}$. By expanding the likelihood using conditional probabilities and making use of the Markov property in eq. (1) we write

$$p(\mathcal{Z}|\boldsymbol{\phi}) = p(\mathcal{Z}_s|\mathcal{Z}_{s-1}, \boldsymbol{\phi}_s)p(\mathcal{Z}_{s-1}|\mathcal{Z}_{s-2}, \boldsymbol{\phi}_{s-1})\cdots p(\mathcal{Z}_1|\boldsymbol{\phi}_1), \tag{18}$$

thus the autoregressive model can be trained by solving s distinct optimization problems with respect to the parameters $\boldsymbol{\phi}_t$, corresponding to the different levels of fidelity $t = 1, \ldots s$. At an arbitrary level t, the log-likelihood $\boldsymbol{\ell}_t(\boldsymbol{\phi}_t) := -\log p(\mathcal{Z}_t|\mathcal{Z}_{t-1}, \boldsymbol{\phi}_t)$ is written

$$\boldsymbol{\ell}_t(\boldsymbol{\phi}_t) = \frac{1}{2}\left(\mathcal{Z}_t - \rho_{t-1}\mathcal{Z}_{t-1}(D_t)\right)^T V_s^{-1}\left(\mathcal{Z}_t - \rho_{t-1}\mathcal{Z}_{t-1}(D_t)\right) + \frac{1}{2}\log|V_s| + \frac{n_t}{2}\log(2\pi). \tag{19}$$

After differentiating with respect to the components of $\boldsymbol{\phi}_t$ and setting equal to zero, one can derive the maximum-likelihood estimates

$$\hat{\rho}_{t-1} = \left[\mathbf{h}_t^T(R_t(D_t) + \sigma_{\epsilon_t}^2)^{-1}\mathbf{h}_t\right]^T \mathbf{h}_t^T\left(R_t(D_t) + \sigma_{\epsilon_t}^2\right)^{-1}\mathcal{Z}_t, \tag{20}$$

and

$$\hat{\sigma}_t^2 = \frac{1}{c}\left(\mathcal{Z}_t - \hat{\rho}_{t-1}\mathcal{Z}_{t-1}(D_t)\right)^T\left(R_t(D_t) + \sigma_{\epsilon_t}^2\right)^{-1}\left(\mathcal{Z}_t - \hat{\rho}_{t-1}\mathcal{Z}_{t-1}(D_t)\right), \tag{21}$$

where $\mathbf{h}_t = [\mathbf{1}_{n_t}\ \mathcal{Z}_{t-1}(D_t)]$, $c = (n_t - 1)\mathbb{I}_{t=1} + (n_t - 2)\mathbb{I}_{t>1}$, $\mathcal{Z}_{t-1}(D_t)$ is the set of observations from code z_{t-1} corresponding only to the design points in D_t. Further, $\mathbf{1}_{n_t}$ is a vector of length n_t, filled with ones and $\mathbb{I}_A$ is the indicator function that is one in A, and zero otherwise. At last, the two estimates given above are dependent on the lengthscales $\boldsymbol{\theta}_t$ and the noise variance $\sigma_{\epsilon_t}^2$. Those can be estimated by maximizing the concentrated restricted log-likelihood function

$$\boldsymbol{\ell}_t^{res} := \log|R_t(D_t) + \sigma_{\epsilon_t}^2\mathbf{I}| + c\log\hat{\sigma}_t^2, \quad t = 1, \ldots, s. \tag{22}$$

2.3.2. Markov Chain Monte Carlo sampling

A fully Bayesian updating strategy of the model parameters can be carried out using Markov Chain Monte Carlo (MCMC) sampling in order to generate samples of the posterior distribution of $\boldsymbol{\phi}$. The latter is written as

$$p(\boldsymbol{\phi}|\mathcal{Z}) \propto p(\mathcal{Z}|\boldsymbol{\phi})p(\boldsymbol{\phi}) \tag{23}$$

where the likelihood distribution is $p(\mathcal{Z}|\boldsymbol{\phi}) = \exp(-\boldsymbol{\ell}(\boldsymbol{\phi}))$ with $\ell(\boldsymbol{\phi})$ given in eq. (16). The prior distributions of parameters ϕ corresponding to lengthscales at various levels of fidelity are modeled using independent Beta distributions and those corresponding to the variances are modeled using independent inverse gamma distributions. In our implementations we use a Metropolis-Hastings algorithm [49] that allows jumps in order to fully explore possible multimodal behavior.

3. Multi-fidelity Gaussian Processes on low-dimensional embeddings

3.1. Dimensionality reduction using projection matrices

As discussed in the introduction, the main goal in this paper is to learn a response surface on a high dimensional input space within a multi-fidelity context, i.e. by leveraging observations from low accuracy simulators that are cheaper to evaluate. Several shortcomings can make the learning process problematic in the presence of high dimensions and large datasets. For instance, the large number of parameters in the

case of anisotropic kernels can result in poor performance of the MCMC algorithm or convergence of the MLE procedure to suboptimal solutions. Below, we develop an dimensionality reduction framework where the ARGP model is trained on a low dimensional input space that is the result of a linear embedding of the original space. We assume throughout this work that the target function $z_s : \mathbb{R}^D \in \mathbb{R}$ can be described or be well-approximated by a function $f_s : \mathbb{R}^d \to \mathbb{R}$, defined in a d-dimensional space $\mathcal{X}_d$, where $d \ll D$, such that

$$z_s(\mathbf{x}) \approx f_s(\mathbf{W}^T\mathbf{x}). \tag{24}$$

Here, $\mathbf{W}$ is assumed to be a $D \times d$ orthonormal matrix that maps the original design space $\mathcal{X}$ to $\mathcal{X}_d$. The choice of orthonormality is made so that the columns of $\mathbf{W}$ form a basis on $\mathcal{X}_d$ and therefore the matrix itself is a projection from $\mathcal{X}$ to $\mathcal{X}_d$. That further implies that once $\mathcal{X}_d$ is identified, any other set of basis vectors forms a projection that can describe the same approximation of z_s.

Next, it is important to assume that all the lower fidelity codes can be approximated by similar "link" functions defined on the same low dimensional space $\mathcal{X}_d$, that is using the same projection function $\mathbf{W}$. We therefore assume that for each $t = 1, \ldots, s-1$

$$z_t(\mathbf{x}) \approx f_t(\mathbf{W}^T\mathbf{x}), \tag{25}$$

for functions $f_t : \mathbb{R}^d \to \mathbb{R}$, $t = 1, \ldots, s-1$. Thus, we can define the autoregressive model (2) on $\mathcal{X}_d$ as

$$\begin{cases} f_t(\tilde{\mathbf{x}}) = \rho_{t-1}(\tilde{\mathbf{x}}) f_{t-1}(\tilde{\mathbf{x}}) + \delta_t(\tilde{\mathbf{x}}) + \epsilon_t(\tilde{\mathbf{x}}) \\ f_{t-1}(\tilde{\mathbf{x}}) \perp \delta_t(\tilde{\mathbf{x}}),\ f_{t-1}(\tilde{\mathbf{x}}) \perp \epsilon_t(\tilde{\mathbf{x}}),\ f_1(\tilde{\mathbf{x}}) \perp \epsilon_1(\tilde{\mathbf{x}}) \quad , \quad 1 < t \leq s, \\ \delta_t(\tilde{\mathbf{x}}) \perp \epsilon_t(\tilde{\mathbf{x}}), \end{cases} \tag{26}$$

where $\tilde{\mathbf{x}} = \mathbf{W}^T\mathbf{x}$ and $\mathbf{x} \in \mathcal{X}$.

The idea behind this formulation is that, by identifying $\mathbf{W}$ such that the above approximations are accurate, the multi-fidelity output quantify of interest is described as a function of a low dimensional input, thus, it becomes simpler to characterize its predictive distribution. Applying the same projection $\mathbf{W}$ at all levels of fidelity, practically means that all codes exhibit most of their variability within the same "active" subspace, as it was termed by Constantine [39, 40]. Although this might seem as a strong assumption, in fact, considering that different fidelity codes are typically highly correlated as they simulate the same physical process at different levels of accuracy, the assumption is fairly plausible. Furthermore, it is worth pointing out that the ARGP model (26) where the low dimensional spaces are defined using different $\mathbf{W}_t$, $t = 1, \ldots, s$ at different levels, does no longer honor the Markov property (1). We do not pursue further such a scenario in this work.

Assigning the same prior distributions as in eq. (4) for the reduced dimensionality ARGP introduced above, results in the same posterior expressions given in eqs. (8)-(9), where the covariance matrices describe the correlations of the training points in $\mathcal{X}_d$. By denoting the projections of all design points where the model outputs are observed, as $D_t^{\mathbf{W}} = \{\tilde{\mathbf{x}} = \mathbf{W}^T\mathbf{x} : \mathbf{x} \in D_t\}$, $t = 1, \ldots s$, we can rewrite the covariance kernels $R_i(D_t^{\mathbf{W}})$ and $R_i(D_t^{\mathbf{W}}, D_{t'}^{\mathbf{W}})$ as functions of the high dimensional inputs in the original space $\mathcal{X}$, with entries

$$r_i(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = r_i(\mathbf{W}^T\mathbf{x}, \mathbf{W}^T\mathbf{x}'), \quad \mathbf{x}, \mathbf{x}' \in D_t, \quad \text{and} \quad r_i(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) = r_i(\mathbf{W}^T\mathbf{x}, \mathbf{W}^T\mathbf{x}'), \quad \mathbf{x} \in D_t, \mathbf{x}' \in D_{t'}, \tag{27}$$

respectively. By incorporating these expressions in the likelihood and posterior distributions, it becomes clear that $\mathbf{W}$ can be considered as an additional set of model parameters that needs to be inferred from observations, in a similar manner as in the single-fidelity setting presented in [42].

3.2. Simultaneous autoregressive GP training and embedding learning

As highlighted above, training the autoregressive GP model and identifying the low-dimensional design space requires learning the model parameters $\boldsymbol{\phi}$ and the rotation matrix $\mathbf{W}$ simultaneously. To do so, we propose a two-step iterative procedure that iterates between tuning the $\boldsymbol{\phi}$ parameters for a fixed $\mathbf{W}$ and updating $\mathbf{W}$ while keeping the parameters $\boldsymbol{\phi}$ fixed. Such algorithms have been used in the past for dimensionality reduction purposes within the context of GP regression [42, 43] and Polynomial Chaos adaptations

[47, 50] and have demonstrated great potential. To further justify the choice of updating scheme, we can write the Bayesian posterior of the joint parameters $(\boldsymbol{\phi}, \mathbf{W})$ given observations $\mathcal{Z}$, as

$$p(\boldsymbol{\phi}, \mathbf{W}|\mathcal{Z}) \propto p(\mathcal{Z}|\boldsymbol{\phi}, \mathbf{W})p(\boldsymbol{\phi})p(\mathbf{W}). \tag{28}$$

Assuming that sampling from the marginal posteriors of $\boldsymbol{\phi}$ and $\mathbf{W}$ conditional on each other, a Gibbs sampler would consist of generating a Markov chain that eventually converges to the joint posterior above [51], therefore generating a chain of $\boldsymbol{\phi}_n$ and $\mathbf{W}_n$ samples in such a fashion will ultimately explore $p(\boldsymbol{\phi}, \mathbf{W}|\mathcal{Z})$.

In order to carry out such a sampling scheme, we employ the following strategy: When $\mathbf{W}$ is given, the $\boldsymbol{\phi}$ updating step is performed using the methods presented in Section 2.3. For updating $\mathbf{W}$ while $\boldsymbol{\phi}$ is kept fixed, we use the Geodesic Monte Carlo method that samples from target distributions defined on embedded manifolds, as it is described in Byrne & Girolami [46]. In our case, $\mathbf{W}$ is a matrix that takes values on the Stiefel manifold of orthonormal d-frames on $\mathbb{R}^D$ [52], and the target density is the marginal posterior $p(\mathbf{W}|\boldsymbol{\phi}, \mathcal{Z})$. The iterative procedure is summarized in Algorithm 1. The details of Geodesic Monte Carlo sampling are presented in the next section.

Algorithm 1: Two-step iterative update of $\boldsymbol{\phi}$ and $\mathbf{W}$

Require: Design input sets $\{D_t\}_{t=1,\dots,s}$, observations $\{\mathcal{Z}_t\}_{t=1,\dots,s}$, initial guess $\mathbf{W}_0 \sim p(\mathbf{W})$, assign priors on $\boldsymbol{\phi}$ or initialize to $\boldsymbol{\phi}_0$.

repeat

- $\boldsymbol{\phi}^{(n)} \leftarrow$ Run MCMC or MLE optimization (Sec. 2.3.2-2.3.1)
- $\mathbf{W}^{(n)} \leftarrow$ Run Geodesic MC Algorithm 2 with target density $p(\mathbf{W}|\mathcal{Z}, \boldsymbol{\phi}^{(n)}) \propto p(\mathcal{Z}|\mathbf{W}, \boldsymbol{\phi}^{(n)})p(\mathbf{W})$

until *relative change in Hamiltonian function (30) is less than tolerance* ϵ_H.

3.2.1. Geodesic Monte Carlo

The Geodesic Monte Carlo algorithm developed in [46] is at its core a Hamiltonian Monte Carlo (HMC) sampling technique [53] defined on a Riemannian manifold embedded in $\mathbb{R}^D$. A Hamiltonian function is defined that describes the dynamics of a spatial variable and is characterized by its target density. Next, the Hamiltonian flows are simulated through numerical integration in order to propose new samples that are to be accepted or rejected, based on a Metropolis-Hastings step [54]. In our case, we are working on the Stiefel manifold that is defined as

$$\mathcal{V}_{d,D} = \{\mathbf{w} \in \mathbb{R}^{D\times d} : \mathbf{w}^T\mathbf{w} = \mathbf{I}_d\}, \tag{29}$$

for $d \leq D$, where the special cases $d = D$ and $d = 1$ correspond to the set of all orthonormal square matrices and the $(D-1)$-dimensional hypersphere on $\mathbb{R}^D$, respectively. The Hamiltonian corresponding to our target posterior density is defined as

$$H(\mathbf{w}, \mathbf{u}) := H^{[1]}(\mathbf{w}, \mathbf{u}) + H^{[2]}(\mathbf{w}, \mathbf{u}) = -\log p(\mathbf{w}|\boldsymbol{\phi}, \mathcal{Z}) + \frac{1}{2}\mathbf{u}^T\mathbf{u}, \tag{30}$$

where $\mathbf{u} \in \mathbb{R}^{D\times d}$ is an auxiliary velocity variable and the dynamics of $H(\mathbf{w}, \mathbf{u})$ are described by

$$\begin{cases} \dot{\mathbf{w}} = & \dfrac{\partial H}{\partial \mathbf{u}} = & \mathbf{u} \\ \dot{\mathbf{u}} = & -\dfrac{\partial H}{\partial \mathbf{w}} = & \nabla_{\mathbf{w}} \log p(\mathbf{w}|\boldsymbol{\phi}, \mathcal{Z}) \end{cases}. \tag{31}$$

The key difference of the sampling strategy proposed in [46] from traditional HMC algorithms is that instead of relying on symplectic integrators [55] to simulate the Hamiltonian flow, we take advantage of the fact that the dynamics of the kinetic term $H^{[2]}(\mathbf{w}, \mathbf{u})$ describe a flow over a geodesic curve that are explicitly known for certain manifolds, therefore numerical integration applies only on the potential term, thus improving accuracy and performance.

In summary, at the n-th step of the algorithm, a random velocity vector is proposed that is tangent on the manifold at the previously accepted step $\mathbf{W}^{(n-1)}$, and it specifies the direction along which the Hamiltonian is going to move. Then the two terms $H^{[1]}(\mathbf{w},\mathbf{u})$ and $H^{[2]}(\mathbf{w},\mathbf{u})$ are integrated over time $t=\epsilon$ by first updating $H^{[1]}(\mathbf{w},\mathbf{u})$ for a time step $t=\epsilon/2$, followed by updating $H^{[2]}(\mathbf{w},\mathbf{u})$ for $t=\epsilon$ using the known geodesic curve formula, and then $H^{[1]}(\mathbf{w},\mathbf{u})$ is integrated again for $t=\epsilon/2$. The procedure is repeated until a user-defined final time $t=T$ is reached. At last, the resulting "spatial" coordinate $\mathbf{w}^*$ will be accepted in the chain with probability

$$\alpha_{acc}=\min\left\{1,\exp\left[-H(\mathbf{w}^*,\mathbf{u}^*)+H(\mathbf{w}_0,\mathbf{u}_0)\right]\right\}. \tag{32}$$

The full mathematical details of the integrator scheme for the Hamiltonian flow and the geodesic formulas on $\mathcal{V}_{d,D}$ are given in Appendix A. Algorithm 2 describes one complete accept-reject step of the above procedure. As a prior on $\mathbf{W}$, we use a Matrix-Langevin (mL) distribution [56] whose density function is given by

$$p(\mathbf{w})=\frac{1}{c(\mathbf{F})}\exp\left\{\mathrm{Tr}\left[\mathbf{F}^T\mathbf{w}\right]\right\}, \tag{33}$$

where $c(\mathbf{F})$ is the normalizing constant that is parametrized by the matrix $\mathbf{F}\in\mathbb{R}^{D\times d}$. Details on the geometric interpretation of the mL density and how to tune the prior parameters $\mathbf{F}$ are given in Appendix B. Note that updating $\mathbf{u}$ by integrating the potential term $H^{[1]}(\mathbf{w},\mathbf{u})$ requires computing the gradient of the marginal log-posterior distribution $p(\mathbf{W}|\boldsymbol{\phi},\mathcal{Z})\propto p(\mathcal{Z}|\boldsymbol{\phi},\mathbf{W})p(\mathbf{W})$. The gradient of the marginal log-posterior becomes

$$\nabla_{\mathbf{w}}\log p(\mathbf{W}|\boldsymbol{\phi},\mathcal{Z})=\nabla_{\mathbf{w}}\log p(\mathcal{Z}|\boldsymbol{\phi},\mathbf{W})+\nabla_{\mathbf{w}}\log p(\mathbf{W}) \tag{34}$$

where the likelihood gradient is given by (17) and the gradient of V_s consists of gradients of block matrices $\frac{\partial V^{t,t'}}{\partial w_{ij}}$, $1\le t\le t'\le s$. Those involve differentiating $R_i(D_t^{\mathbf{W}},D_t^{\mathbf{W}})$ that have (i,j)-th entries

$$\frac{\partial r_i(\tilde{\mathbf{x}},\tilde{\mathbf{x}}')}{\partial w_{ij}}=\frac{\partial}{\partial\tilde{\mathbf{x}}_j}\left[r_i\left(\mathbf{W}^T\mathbf{x},\mathbf{W}^T\mathbf{x}'\right)\right]\mathbf{x}_i+\frac{\partial}{\partial\tilde{\mathbf{x}}_j'}\left[r_i\left(\mathbf{W}^T\mathbf{x},\mathbf{W}^T\mathbf{x}'\right)\right]\mathbf{x}_i'. \tag{35}$$

At last, the log-prior gradient is $\nabla_{\mathbf{w}}\log p(\mathbf{W})=\mathbf{F}^T$.

4. Numerical examples

For all numerical examples presented in this section we have used the following settings:

1. The number of time steps and the step-size used in the integration of the Hamiltonian in Algorithm 2 are set to $T=10$ and $\epsilon=0.05$ respectively. These values have been carefully selected after numerical experimentation with the algorithm and they provide a moderate acceptance rate while keeping running times to reasonably low levels. Intuitively, a highly accurate integration resulting in high acceptance rate would require a large value for T and very small ϵ but it would slow down the algorithm significantly. On the other hand, coarse integration using large ϵ and small T would speed up the algorithm but would reduce the acceptance rate. Another characteristic of the algorithm that manifests in high dimensions for both d and D is that for an arbitrary initial point $\mathbf{W}_0$ the acceptance rate during the first few iterations is significantly low, as a result of the standard Gaussian proposal from which we sample $\mathbf{u}$, leading to completely random directions along with the Hamiltonian flow evolves. To improve performance of the algorithm, we change the mL prior at each iteration n by setting its parameter $\mathbf{U}=\mathbf{W}_{n-1}$. Each iteration stops when one single sample $\mathbf{W}$ is accepted and we reduce the step size ϵ to $\epsilon/1.2$ after 20 successive rejections in order to refine the Hamiltonian integration and reset on the next iteration. We stop sampling as soon as the first sample is accepted.

Algorithm 2: Geodesic Monte Carlo algorithm [46]

Initialize: Choose integration period T, time step ϵ and sample $\mathbf{W}_0 \sim p(\mathbf{W})$.
At the n-th step assume $\mathbf{W}_n = \mathbf{W}$:
$\mathbf{u} \sim \mathcal{N}(0, \mathbf{I}_{D,d})$
$\mathbf{u} \leftarrow \Pi_{\mathbf{W}}(\mathbf{u})$
$H \leftarrow \log p(\mathbf{W}|\boldsymbol{\phi}, \mathcal{Z}) - \frac{1}{2}\mathbf{u}^T\mathbf{u}$
$\mathbf{W}^* \leftarrow \mathbf{W}$
for $h = 1$ *to* T **do**
  $\mathbf{u} \leftarrow \mathbf{u} + \frac{\epsilon}{2}\nabla_{\mathbf{W}} \log p(\mathbf{W}^*|\boldsymbol{\phi}, \mathcal{Z})$
  $\mathbf{u} \leftarrow \Pi_{\mathbf{W}}(\mathbf{u})$
  Update $(\mathbf{W}^*, \mathbf{u})$ by following the geodesic flows (A.7)-(A.8) for a time interval ϵ
  $\mathbf{u} \leftarrow \mathbf{u} + \frac{\epsilon}{2}\nabla_{\mathbf{W}} \log p(\mathbf{W}^*|\boldsymbol{\phi}, \mathcal{Z})$
  $\mathbf{u} \leftarrow \Pi_{\mathbf{W}}(\mathbf{u})$
end
$H^* \leftarrow \log p(\mathbf{W}^*|\boldsymbol{\phi}, \mathcal{Z}) - \frac{1}{2}\mathbf{u}^T\mathbf{u}$
$u \sim \mathcal{U}(0, 1)$
if $u < \exp(H^* - H)$ **then**
  $\mathbf{W} \leftarrow \mathbf{W}^*$
end

2. To train the ARGP using MCMC sampling, we assign beta priors with parameters $(1, 0.1)$ on the lengthscales $\boldsymbol{\theta}_t$ and inverse Gamma priors with parameters $(5, 5)$ and $(1, 10^{-4})$ on the variance parameters σ_t^2 and $\sigma_{\epsilon_t}^2$ respectively, for $t = 1, \ldots, s$. To proceed with sampling $\mathbf{W}$, the hyperparameters are then fixed to the median values of the full MCMC chain after 200 samples are accepted. The MCMC algorithm is run using General Electric Global Research Center's in-house Bayesian modeling toolbox GEBHM [57]. To train the ARGP using MLE, the log-likelihood or the restricted log-likelihoods are maximized using the BFGS algorithm [58].

4.1. Academic example 1: Three-fidelity model with known 1-dimensional embedding

We consider the following three levels of code

$$\begin{cases} z_1(\mathbf{x}) = & f_1(\mathbf{w}^T\mathbf{x}) \\ z_2(\mathbf{x}) = & f_2(\mathbf{w}^T\mathbf{x}) \\ z_3(\mathbf{x}) = & f_3(\mathbf{w}^T\mathbf{x}) \end{cases}, \tag{36}$$

where $\mathbf{w} \in \mathcal{V}_{1,D}$ so that $\mathbf{w}^T\mathbf{x}$ is a scalar variable and the *link* functions f_1, f_2, f_3 are given by

$$\begin{cases} f_1(\mathbf{w}^T\mathbf{x}) = \frac{1}{2}(8\mathbf{w}^T\mathbf{x} - 2)^2 \sin(5\mathbf{w}^T\mathbf{x} - 4) + 10(\mathbf{w}^T\mathbf{x} - 1/2) \\ f_2(\mathbf{w}^T\mathbf{x}) = 2f_1(\mathbf{w}^T\mathbf{x}) - 20\mathbf{w}^T\mathbf{x} + 20 \\ f_3(\mathbf{w}^T\mathbf{x}) = \frac{3}{2}f_2(\mathbf{w}^T\mathbf{x}) + 30(\mathbf{w}^T\mathbf{x})^2 \end{cases}. \tag{37}$$

For this example we take $D = 10$ and we generate $\mathbf{w}$ randomly, by fixing the random seed in order to

ensure reproducibility. For our numerical experiments, the projection matrix is fixed to

$$\mathbf{w} = \begin{bmatrix} 0.14042 \\ -0.35474 \\ 0.42674 \\ -0.09312 \\ -0.21463 \\ 0.26425 \\ 0.25603 \\ -0.18959 \\ 0.00467 \\ -0.66800 \end{bmatrix}. \tag{38}$$

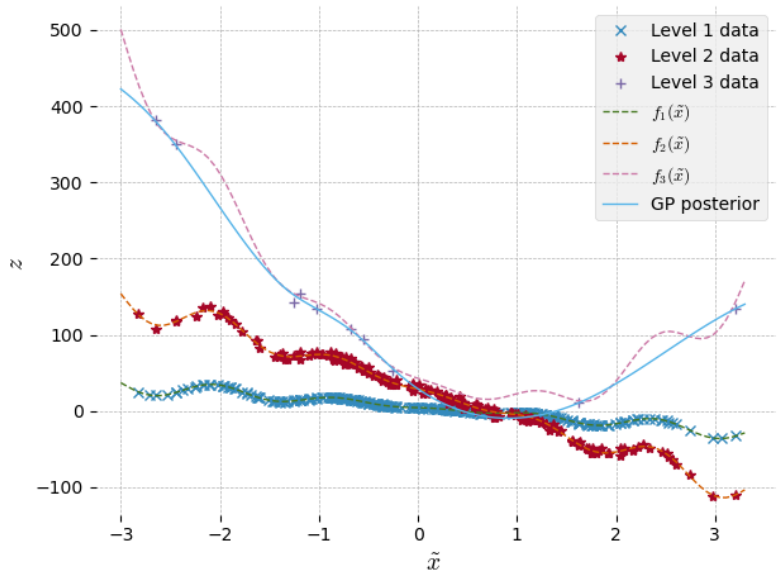


Figure 1: Academic example 1. Training data along with the 1-dimensional representation of z_1, z_2 and z_3 and the GP posterior mean mean using only D_3 data.

We generate synthetic data that consists of observations obtained on nested design points D_1, D_2 and D_3 from the low, intermediate and high fidelity codes respectively, where $|D_1| = 300$, $|D_2| = 200$ and $|D_3| = 10$. The observations are corrupted by Gaussian noise at all levels of fidelity with their standard deviations being equal to 0.5, 3 and 5, corresponding to z_1, z_2 and z_3 respectively. Fig. 1 shows the full training data set along with the true 1-dimensional link functions f_1, f_2 and f_3. To motivate our study, a single-level Gaussian process regression is performed on the 10 high-fidelity data points that are available. The predictive mean, depicted with solid blue line, matches the training points but clearly is unable to capture the true model's fluctuations due to the absence of a sufficient amount of data, let alone the fact that the true $\mathbf{w}$ and the resulting 1-dimensional representation is hypothetically not available to the experimenter. That essentially means that a dimensionality reduction approach performed using only the high fidelity dataset, as proposed in [42] would provide such a predictive mean only in the best case scenario, where $\mathbf{w}$ would be recovered exactly. In addition. even learning this rotation matrix would be challenging due to the limited data availability and convergence of the two step algorithm would be expected to be extremely slow. Lastly, it is needless to say that gradient-based methods such as [39] would simply fail dramatically due to the poor gradient Monte Carlo estimate using only 10 data points. On the other hand, in a multi-fidelity GP regression setting in the original 10-dimensional design space, the available data might fail to provide

meaningful inference results, while training the model becomes again challenging due to the large number of parameters to be inferred, including different lengthscales that are present in the anisotropic kernels, as well as the repeated use of Cholesky decomposition for inverting large covariance matrices.

We run Algorithm 1 for this particular setting and after only eleven iterations we obtain the converged rotation. Fig. 2 (left) shows the drawn samples during iterations 5-11. It can be seen that the values are in full agreement with -$\mathbf{w}$ which is a valid rotation since the representation of z_3 through its link function f_3 is invariant under reflections about the origin. Fig. 2 (right) shows the plots of the predictive mean $f_3(\tilde{x})|\mathcal{Z}$ and the true link function $f_3(\tilde{x})$ from eq. (36). The excellent agreement between the two is apparent. At last, Table 1 shows the estimated model parameter values $\boldsymbol{\phi}$. For comparison we display the value of the single-level GP regression shown in Fig. 1. As one can conclude from the plot, the single-level GP overestimates the lengthscale $\hat{\theta}_3 = 5.223$ and interprets the data discrepancies as observation noise ($\log \sigma^2_{\epsilon_3} = 4.017$). On the contrary, the obtained ARGP gives high fidelity lengthscale $\hat{\theta}_3 = 1.7$ while the noise variances for all fidelities are almost negligible. By capturing accurately the correlations between the different levels $\hat{\rho}_1 = 1.955$ and $\hat{\rho}_2 = 1.242$ (recall the true values $\rho_1 = 2$ and $\rho_2 = 1.5$), the autoregressive model leverages the low fidelity data effectively and captures the model's fluctuations even in areas such as the interval $\tilde{x} \in [2, 3]$ where high fidelity training points are fully absent.

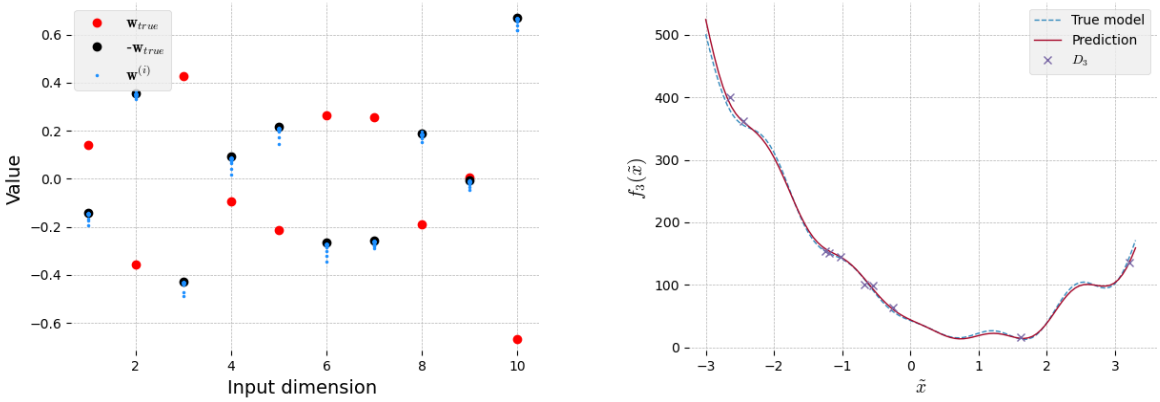


Figure 2: Academic example 1. Left: Entry values of the posterior samples $\mathbf{w}^{(i)}$ and the true values of $\mathbf{w}$ and $-\mathbf{w}$. Right: Comparison of the multi-fidelity predictive mean and the true link function $f_3(\tilde{x})$.

	θ_1	$\log \sigma_1^2$	$\log \sigma_{\epsilon_1}^2$	ρ_1	θ_2	$\log \sigma_2^2$	$\log \sigma_{\epsilon_2}^2$	ρ_2	θ_3	$\log \sigma_3^2$	$\log \sigma_{\epsilon_3}^2$
GP									5.223	11.601	4.017
ARGP	-0.508	7.022	-8.297	1.955	3.296	11.518	-17.868	1.242	1.700	13.107	-20.123

Table 1: Academic example 1. Final maximum likelihood estimates of the model parameters.

4.2. Academic example 2: Three-fidelity model with known 2-dimensional embedding

A three-level autoregressive Gaussian Process is presented in this example where the low dimensional embedding $\mathcal{X}_d$ is now 2D and the additional challenge of identifying the dimensionality of $\mathcal{X}_d$ is also explored. Let $\mathbf{W} = [\mathbf{w}_1\ \mathbf{w}_2] \in \mathcal{V}_{2,D}$ be a fixed projection matrix with columns $\mathbf{w}_1$ and $\mathbf{w}_2$. The three levels of code are given by the functions

$$\begin{cases} f_1(\mathbf{W}^T\mathbf{x}) = \sin(\mathbf{w}_1^T\mathbf{x}) \\ f_2(\mathbf{W}^T\mathbf{x}) = f_1(\mathbf{W}^T\mathbf{x}) - 7\sin^2(\mathbf{w}_2^T\mathbf{x}) \\ f_3(\mathbf{W}^T\mathbf{x}) = \frac{3}{2}f_2(\mathbf{W}^T\mathbf{x}) + 5\left(\mathbf{w}_2^T\mathbf{x}\right)^2\sin(\mathbf{w}_1^T\mathbf{x}) \end{cases}. \tag{39}$$

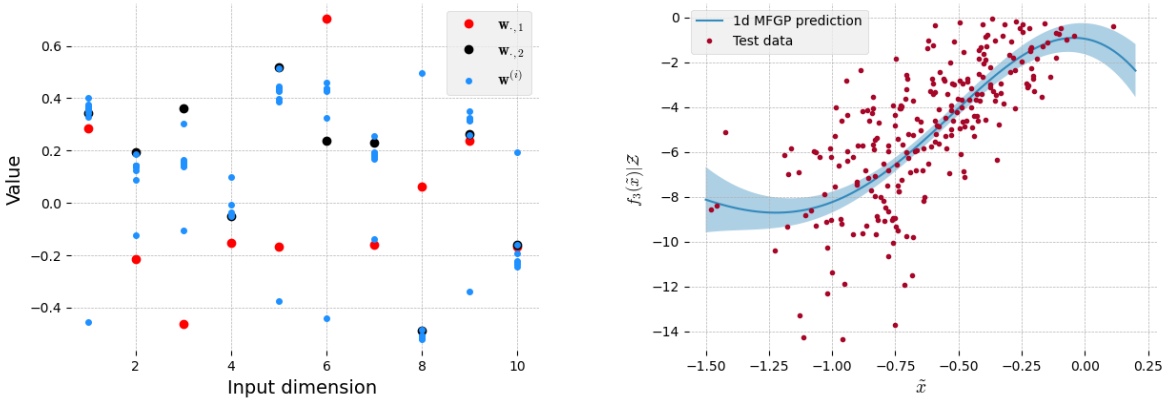


Figure 3: Academic example 2. Left: Comparison of samples $\mathbf{w}^{(i)}$ versus values of columns of true $\mathbf{w}$. Right: Predictive mean and 3-standard deviation confidence bands of the 1-dimensional adapted MFGP along with 250 test data projected in the embedded space.

We consider again $D = 10$ to be the dimensionality of $\mathcal{X}$ and the projection matrix is fixed to

$$\mathbf{W} = \begin{bmatrix} 0.28490 & 0.34201 \\ -0.21608 & 0.19310 \\ -0.46249 & 0.36223 \\ -0.15187 & -0.05088 \\ -0.16601 & 0.51910 \\ 0.70297 & 0.23900 \\ -0.16004 & 0.23084 \\ 0.06096 & -0.48747 \\ 0.23763 & 0.26276 \\ -0.16620 & -0.15930 \end{bmatrix}. \tag{40}$$

For this example we generate data again from nested design points D_1, D_2 and D_3 where this time $|D_1| = 200$, $|D_2| = 100$ and $|D_3| = 25$. Observations are contaminated with Gaussian noise whose standard deviation is equal to $\sigma_{\epsilon_1} = 0.1$, $\sigma_{\epsilon_2} = 0.1$ and $\sigma_{\epsilon_3} = 0.05$ at the corresponding levels.

	$\boldsymbol{\theta}_1$	$\log \sigma_1^2$	$\log \sigma_{\epsilon_1}^2$	ρ_1	$\boldsymbol{\theta}_2$	$\log \sigma_2^2$	$\log \sigma_{\epsilon_2}^2$	ρ_2	$\boldsymbol{\theta}_3$	$\log \sigma_3^2$	$\log \sigma_{\epsilon_3}^2$
d=1	2.005	0.651	-3.082	2.419	0.327	4.901	-6.903	2.505	2.510	8.807	-21.089
d=2	(2.67, 4.40)	4.932	-15.004	0.123	(2.96, 0.26)	4.756	-8.888	-0.529	(0.49, -0.31)	4.083	-18.466

Table 2: Academic example 2. Final maximum likelihood estimates of the model parameters.

We run again Algorithm 1, first for $d = 1$ and then for $d = 2$ and we report our results below. For the $d = 1$ case, the algorithm converged after roughly 12 iterations and, as expected, the trained model fails to capture a suitable embedded space that could honor all training data points and further represent the high fidelity code as a 1-dimensional function. Fig. 3 (left), compares the posterior values of $\mathbf{W}$ accepted while running the Geodesic MC algorithm, with the values of the two columns of the true $\mathbf{W}$ used to generate the training data, Fig. 3 (right) shows the 1-dimensional predictive mean of the posterior ARGP with 3-standard-deviation-wide confidence bands along with 250 test data points. One can observe that the entries of the inferred $\mathbf{w}$ tend mostly towards the values of the 2nd column of the true $\mathbf{W}$. This can be intuitively explained from the fact that the terms including $\mathbf{w}_2$ appear to be more dominant in the overall expression of $f_3(\mathbf{W}^T\mathbf{x})$ and particularly the low fidelity model $f_1(\mathbf{W}^T\mathbf{x})$ has small impact on the high fidelity code. At

the same time, the predictive capabilities of the posterior model clearly fail to span the regions where the additional test data might be observed, indicating that a higher dimensional embedding should be learned. In the $d = 2$ case, the situation improves significantly. As can be seen in Fig. 4, the sampled posterior values of $\mathbf{w}_1$ and $\mathbf{w}_2$ are in agreement with the true values of $\mathbf{W}$ in both columns. The samples have been obtained during iterations 10-20. Furthermore, Fig. 5 shows a 45-degree line plot comparing observations versus predictions using 250 test data points, where the predictions on the left are made using the 2D trained ARGP model while the predictions on the right are made using an ARGP model built on the original 10D input space. Their root mean squared error (RMSE) values are 0.407 and 0.906 respectively. At last, the bottom plot shows the 2-dimensional predictive mean of the train ARGP model on the same test points. The overall predictive performance of the model is in agreement with the true model output, while the 45-degree line plot suggests that, building the 2D model, provides with better predictive capabilities than using a multifidelity GP on the original input space.

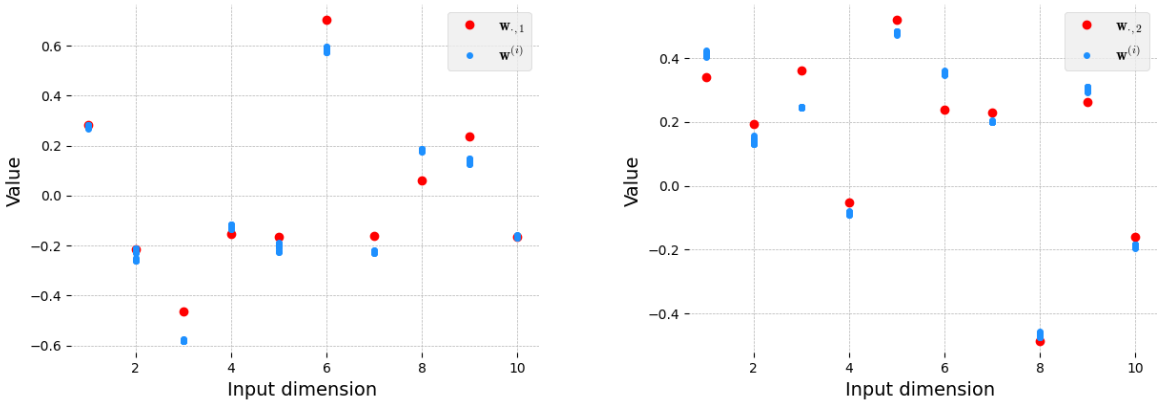


Figure 4: Academic example 2. Comparison of posterior samples of $\mathbf{w}_1$ and $\mathbf{w}_2$ columns of $\mathbf{W}$ versus the nominal values used to generate the training data.

5. Airfoil shape optimization problem

The aerodynamic optimization problem analyzed in this section is the full 3D design of a typical large last stage blade (LSB) of an industrial gas turbine (IGT). As the largest rotating component in the turbine, last stage blades are one of the most mechanically challenged components in an IGT and often determine the total power output of the machine. Increasing push for larger and hotter turbines to drive down cost of electricity has resulted in increasingly challenging LSB designs. With deference typically skewed towards durability, this has generally resulted in greater aerodynamic compromises that negatively impact blade efficiency. This also drives longer design cycles as designers incrementally search for acceptable aero-mechanical solutions. In this type of design, aerodynamic assessments of power, efficiency, and flow capacity are based on expensive high-fidelity computational fluid dynamics (CFD) simulations. Each design iteration requires one or more CFD simulations, depending on the number of inner aerodynamic iterations required to satisfy the cycle's flow capacity requirements. A full 3D optimization would enable the enhancement of turbine performance. However, the process normally implies a large computational cost because of the high dimensionality of the design space.

In the current work, turbine performance is evaluated using steady-state RANS CFD at two different fidelity levels; a fast running coarse mesh for broader design space exploration, and a slower running fine mesh for accuracy refinement. The 3D airfoil is parametrized using the approach illustrated in Figure 7, which shows the airfoil parametrization at one section on the left, and a view of the full 3D rotor on the right. The airfoil surface is constructed with 7 2D profiles at different spanwise locations from hub to tip. Each

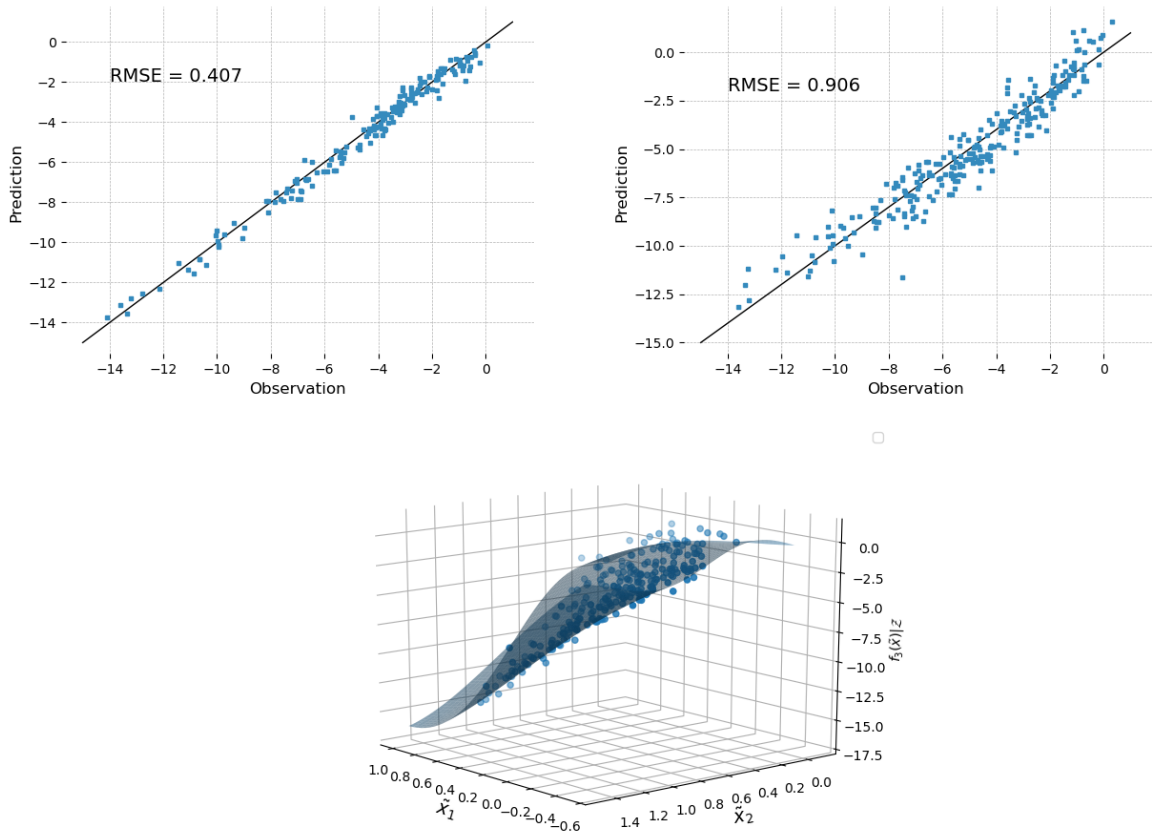


Figure 5: Academic example 2. Top: Prediction of the trained MFGP model on 250 test points versus observations using the inferred projection (right) and prediction of trained MFGP model on the 10-dimensional input space (right). Perfect predictions would fall on the 45° line depicted in black solid color. Bottom: Predictive posterior mean as a function of its two arguments $\tilde{x}_1$, $\tilde{x}_2 \in \mathcal{X}_2$ along with 250 test observations generated from the true model $z_3(\mathbf{x})$.

airfoil section is characterized by 12 independent parameters. The ranges for these parameters are selected to provide a wide design space while respecting geometrical constraints. The 12 parameters, as sketched in Figure 7, include the stagger angle, leading and trailing edge metal angles, leading edge diameter, suction and pressure side wedge angles, leading edge and trailing edge metal angles, and additional parameters to control the airfoil curvature between the leading and trailing edges.

The 2D sections are aligned relative to each other in circumferential and axial space by aligning the section centers of gravity (CG) along a radial line through the hub section CG (referred to as the stacking line). After the section CGs are aligned, one additional parameter, referred to as the airfoil lean angle, is applied to reorient the stacking line relative to the radial direction. Surfaces are fit through the seven stacked sections to create the full, continuous, 3D airfoil definition. Eighty five total parameters are therefore required to define the complete 3D airfoil shape.

The parameters are expressed as offsets from a baseline, requiring that each section starts from an appropriate reference design. An in-house software package tailored specifically for turbomachinery design uses the parameters described above to create the airfoil coordinates, and these coordinates are then transformed into a full 3D CAD model of the rotor blade for the CFD grid generation. A 3D structured mesh is built using a commercially available software package. Grid templates are built from the baseline geometry and used consistently for all the cases throughout the optimization. With this approach, all the grids have sim-

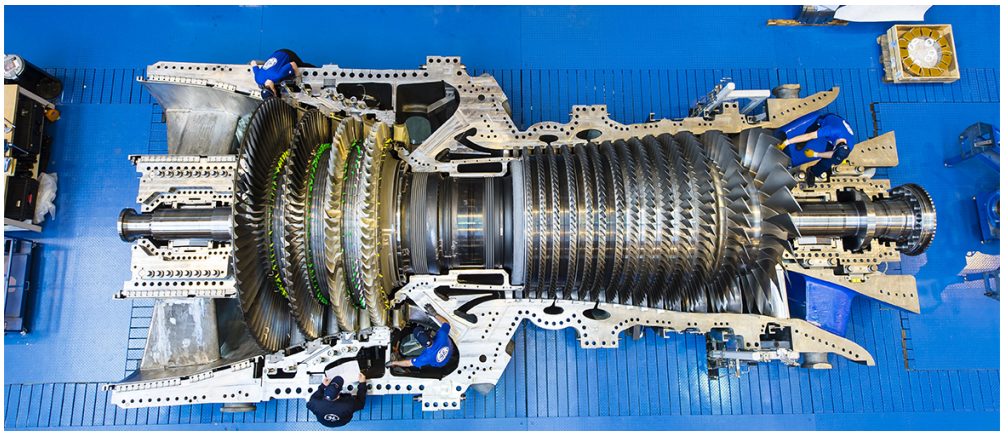


Figure 6: 9HA Industrial Gas Turbine (`https://www.ge.com/power/gas/gas-turbines/h-class`). Last stage blades are visible on the right end of the engine.

ilar refinement and quality metrics. To simulate the full stage, the upstream stator is included in the CFD calculation for each case.The design of the vane and the vane mesh are not altered through the optimization.

The 3D CFD analysis is performed using GEs in-house CFD solver TACOMA, a 2nd-order accurate (in time and space), finite-volume, block-structured, compressible flow solver [59, 60]. The steady Reynolds-Averaged Navier-Stokes (RANS) calculations are solved with a mixing plane between rotating and stationary components. Source terms are included at various locations along the endwalls to simulate the injected cooling, leakage, and purge flows. The two objectives are to maximize the aerodynamic efficiency while matching the baseline degree of reaction. For this problem, with the full stage modeled, the degree of reaction can be calculated using standard turbine definitions.

1. Aerodynamic efficiency: The efficiency is calculated as the ratio of mechanical power and isentropic power. All inputs required to produce the efficiency value are available from the output of the CFD simulation. Design preference is to maximize efficiency.
2. Pseudo-reaction: In a stage calculation, the degree of reaction indicates the split of flow acceleration between stator and rotor. The degree of reaction can be calculated based on the full stage CFD, so the flow quantities between stator and rotor are extracted from the numerical results for each case. This objective is monitored to ensure that the changes in airfoil shape do not significantly affect the turbine operating condition. The design preference of this objective is to be as close as possible to the baseline value.

Multiple quantities are extracted and processed from the CFD, like ideal Mach number at various spanwise locations. Radial profiles of flow quantities, like pressure and flow angle, are obtained to characterize the quality of flow field propagating from the turbine to the downstream exhaust diffuser, which has not been modeled here. Additional objectives are formulated based on the desirability of the turbine exit flow profile. Diffusion rate, diffusion ratio, and shock intensity are assessed for each case to provide additional selection criteria. The resulting simulated data consists of 160 shape configuration points are generated to be run on the coarse grid and another 120 for the fine grid that have successfully converged to CFD solutions. The two main objectives, efficiency and degree of reaction, are plotted in Figure 8, as variations from each respective baseline. The cases assessed with a low fidelity grid are represented with gray circles, while the cases assessed with high fidelity are represented with blue diamonds. Given the small number of points compared to the number of parameters, a conclusive Pareto front has not yet been identified.

For this problem, we fix the correlation coefficient between the two models to $\rho_1 = 1$ since the different level of accuracy of the two solvers varies as a result of different mesh discretization, unlike the previous examples where higher fidelity codes resulted from scaling the low fidelity ones. We split the available high fidelity observations to 75 training points and 45 test points while all low fidelity observations are used

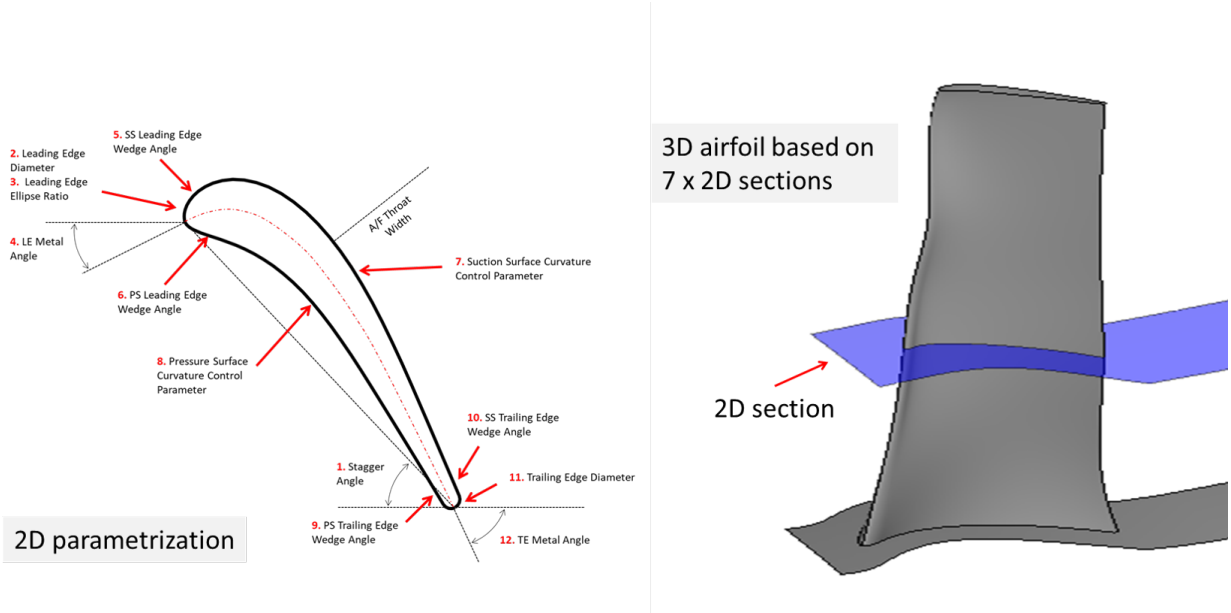


Figure 7: Last stage blade parametrization of 2D sections.

in training the ARGP using MCMC. We perform the training procedure for the two QoIs independently by fitting two different ARGPs and therefore, the reduced dimension and the low dimensional subspace will, in general, be different as well. As expected, Algorithm 1 converges slower than in the previous synthetic examples due to the higher dimensionality. Figs. 9 & 10 show the comparison of the ARGP mean predictions ($\pm$ 2-standard deviations) for the two quantities of interest vs the 45 test observations for reduced dimensionalities $d = 1, 2, 3$. Clearly the fit improves significantly as d increases and reported residual mean square error (RMSE) value simply confirms this fact by reducing to as low as 0.016 & 0.015 respectively. The quality of fit on the test data for $d = 3$ can serve as a stopping criterion when testing for different reduced dimensionalities d and no need for further exploration on $d \geq 4$ is required. To further support this claim we report values of the loglikelihodd function and the Bayesian Information criterion (BIC) [61]. The latter is a standard criterion used for model selection and is given by the loglikelihood expression penalized by a term that depends on the number of training data points and model parameters. Specifically,

$$\text{BIC} = -\ell(\boldsymbol{\phi}^*) - \frac{1}{2}|\boldsymbol{\phi}| \log \sum_{t=1}^{s} n_t, \tag{41}$$

where $|\boldsymbol{\phi}|$ is the number of model parameters to be inferred and $\boldsymbol{\phi}^*$ is the maximum likelihood estimate. The largest values of BIC typically indicate the most favorable model. We plot the maximum log-likelihood and the BIC values as a function of d in Fig. 11. We observe that the log-likelihood becomes almost constant, indicating that no further improvement in terms of fitting the training set can be achieved. The BIC on the other hand drops at $d = 3$, indicating that the number of parameters (penalty term) becomes significantly large, thus the trade-off between data-fit improvement and number of parameters to be inferred has a negative trend.

At last, we display the sample values of the 1st column of $\mathbf{W}$, denoted as $\mathbf{W}_{\cdot,1}$ for both QoIs in Fig. 12. The values corresponding to the same input dimension in general do not seem to be similar, indicating that the importance of each input parameter on the different QoIs varies. Most importantly, very few of the $\mathbf{W}\cdot, 1$ entries are near zero, thus only a small number of the parameters in the original input space are negligible and can be assumed to have no effect in the output QoIs. This allows us to conclude that the reduction from $D = 85$ to $d = 3$ is not the result of simply discarding unimportant parameters but that our algorithm was able to reveal the low-dimensional linear embedding that captures the model's observed

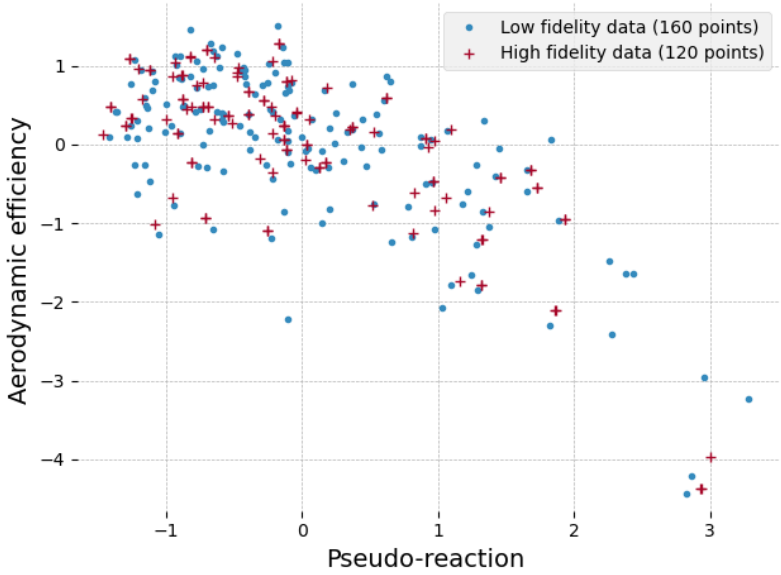


Figure 8: Airfoil shape optimization. Observed outputs from the two-fidelity solvers to be used to construct the ARGP metamodel.

behavior accurately.

6. Conclusions

We have thus far presented a methodology to exploit low dimensional subspaces for input dimensionality reduction and have demonstrated it's applicability to uncertainty propagation problems using multi-fidelity simulations. This was achieved by employing an autoregressive Gaussian Process scheme with a linear embedding that is modeled using an orthogonal matrix that maps the original input variable to its latent counterpart. We treat the matrix as an additional set of hyperparameters and we learn the mapping jointly with training the ARGP by employing a two-step algorithm that train the ARGP using standard MLE and MCMC techniques while it updates the orthogonal matrix using the proposed Geodesic Monte Carlo sampling. We validated our method on synthetic examples with known linear embedding and we further utilised it to study the problem of 3D airfoil optimization by building a low dimensional responce surface to understand the impact of airfoil shape parameters on the particular quantities of interest.

Our work continues on the findings of [39, 42] towards the development of a surrogate that enables a fully Bayesian exploration of the active subspace. Quoting Tripathy & Bilionis [42] "the big challenge is the construction of proposals that force $\mathbf{W}$ to remain on the Stiefel manifold ... such approaches would open the way for more robust AS dimensionality selection". We have achieved a Bayesian treatment that relieves us from such headaches by tailoring our HMC sampling method [46] particularly on the Stiefel manifold. In addition, the capability to leverage information from computer codes of a varying degree of accuracy and cost is another novel step that opens new directions in multi-fidelity simulations for uncertainty quantification, including multi-output Gaussian Processes and multi-objective Bayesian optimization. At last, more sophisticated multifidelity formulations relying on nonlinear autoregressive schemes, such as the work presented by Perdikaris *et al* [45] opens the way to more accurate explorations of the model correlations across different levels of fidelity, albeit with no free lunch. The cost of training *deep* GPs in realistic problems where data is not acquired along nested design points currently remains insurmountable. Such topics are to be explored in future research.

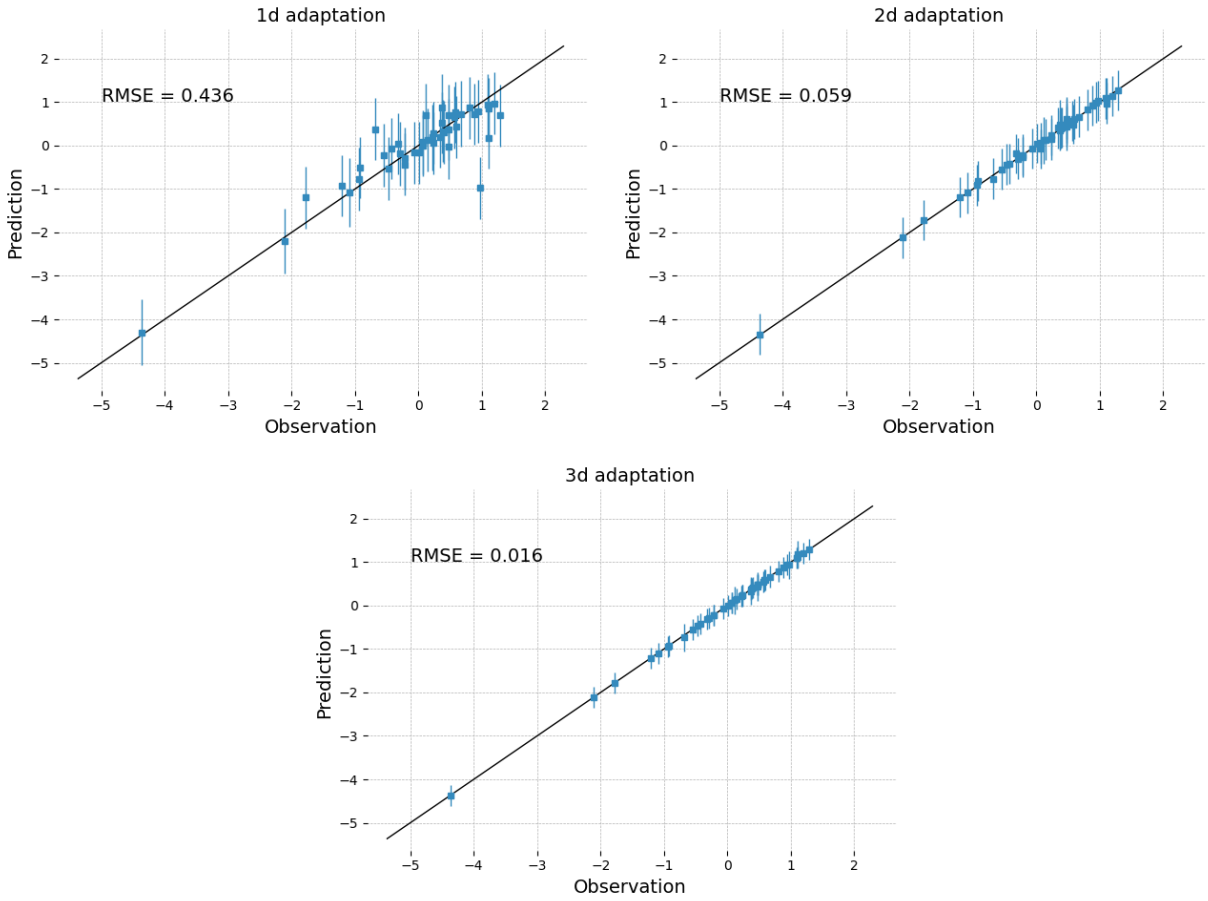


Figure 9: Airfoil shape optimization. Test observations of aerodynamic efficiency vs ARGP predictions for $d = 1, 2, 3$.

Acknowledgments

The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0001204. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Appendix A. Mathematical background on Geodesic Monte Carlo

Appendix A.1. Hamiltonian dynamics on an embedded manifold $\mathcal{M}$

Consider the problem of sampling from a probability density $p(x)$ defined on a manifold $\mathcal{M}$ that is *isometrically* embedded in R^D and equipped with a metric tensor $G(x)$, serving as the isometry between the two spaces, thus, preserving the distance. In order to employ a Hamiltonian Monte Carlo algorithm [55], the Hamiltonian function is formed

$$H(x, v) = -\log p(x) + \frac{1}{2} v^T G^{-1}(x) v \tag{A.1}$$

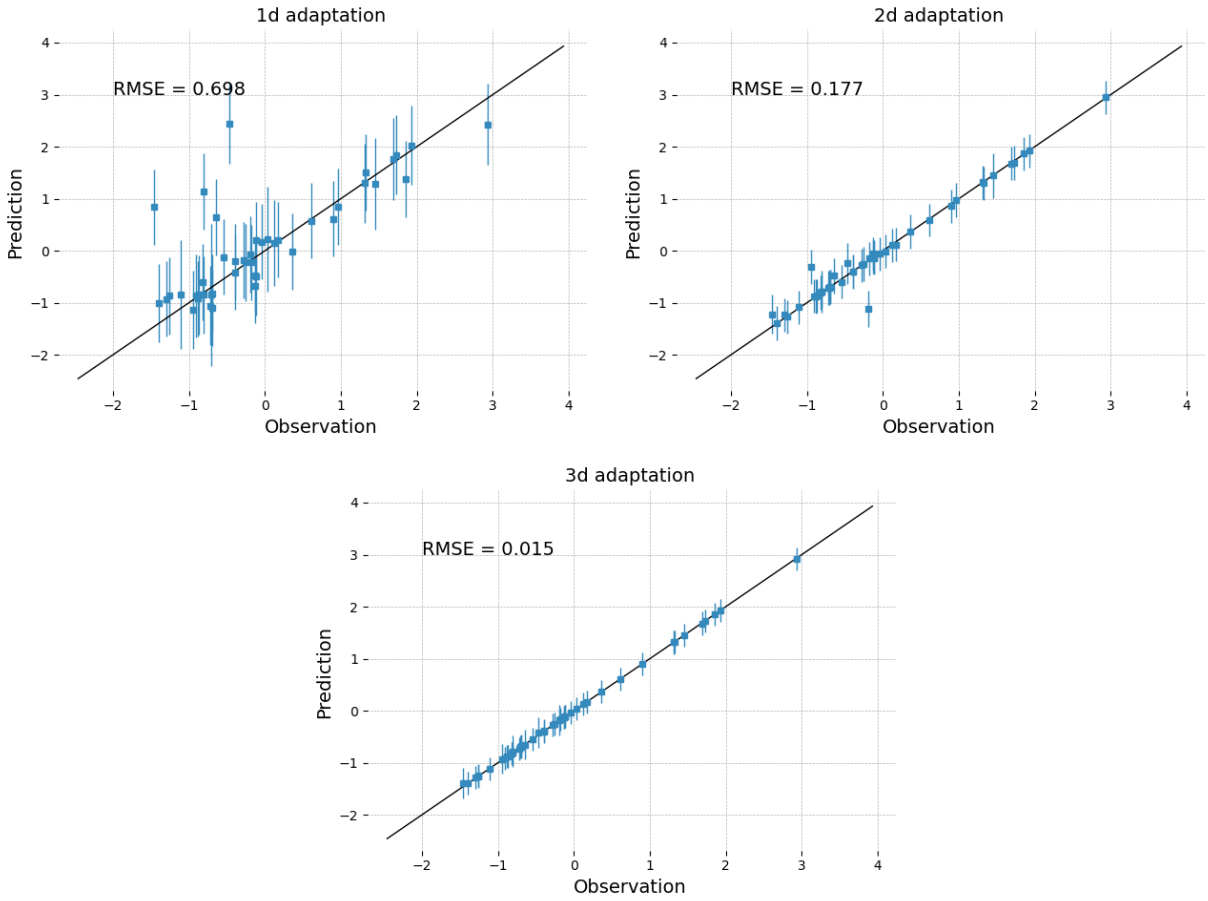


Figure 10: Airfoil shape optimization. Test observations of pseudo-reaction vs ARGP predictions for $d = 1, 2, 3$.

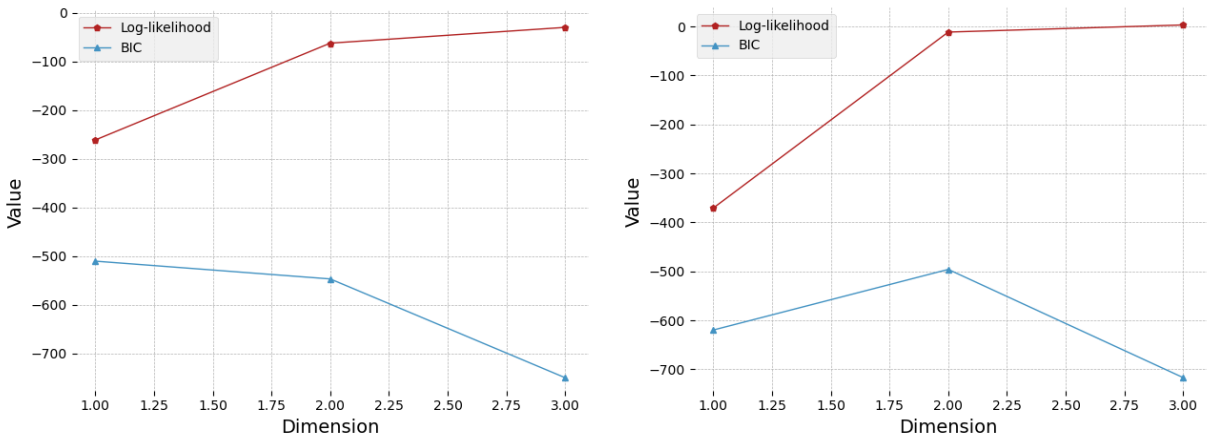
and its dynamics are described by

$$\begin{cases} \dot{x} = \dfrac{\partial H}{\partial v} = G(x)^{-1}u \\ \dot{v} = \dfrac{\partial H}{\partial x} = \nabla_x \left[\log p(x) - \frac{1}{2} v^T G^{-1}(x) v\right] \end{cases} \tag{A.2}$$

Integrating the Hamiltonian on a Euclidean space typically involves a *Stormer-Verlet leapfrog scheme* [61, 62] which in our case requires inverting the metric tensor $G(x)$. To avoid numerical drawbacks associated with such operations, Byrne & Girolami [46] suggest splitting $H(x, v)$ to two distinct Hamiltonians, namely $H^{[1]}(x, v) = -\log p(x)$ with dynamics

$$\begin{aligned} \dot{x} = \quad \frac{\partial H^{[1]}}{\partial v} &= 0 \\ \dot{v} = -\frac{\partial H^{[1]}}{\partial x} &= \nabla_x \log p(x) \end{aligned} \tag{A.3}$$

and $H^{[2]}(x, v) = \frac{1}{2} v^T G^{-1}(x) v$ whose dynamics are described as a geodesic flow over a geodesic curve $\gamma(t)$ that maintains a constant velocity $||\dot{\gamma}(t)||_G$ and is known as *Levi-Civita connection* of G [63].



(a) Aerodynamic efficiency (b) Pseudo-reaction

Figure 11: Airfoil shape optimization. Maximum log-likelihood and Bayesian Information criterion values for reduced dimension $d = 1, 2, 3$.

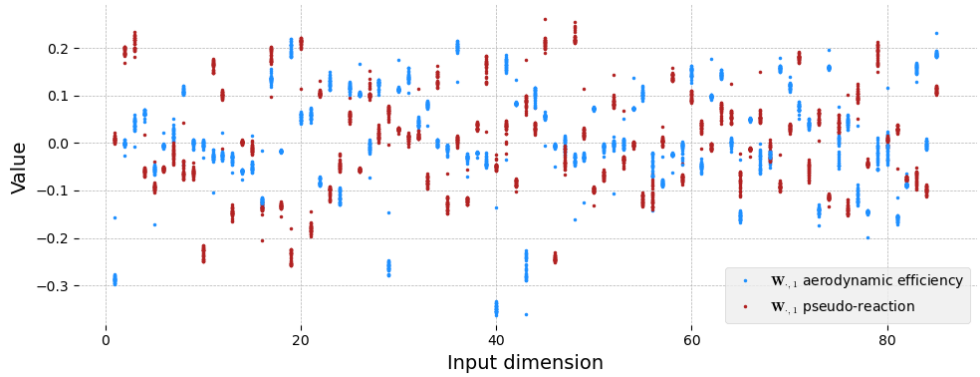


Figure 12: Airfoil shape optimization. Entries of the 1st column $\mathbf{W}_{\cdot,1}$ of the projection matrix $\mathbf{W}$ corresponding to the two QoIs.

Appendix A.2. Hamiltonian dynamics on the Stiefel manifold $\mathcal{V}_{d,D}$

For the special case where $\mathcal{M} = \mathcal{V}_{d,D}$, with target density function being $p(\mathbf{w}|\boldsymbol{\phi}, \mathcal{Z})$, the Hamiltonian is given by eq. (30). The tangent space at a point $\mathbf{w} \in \mathcal{V}_{d,D}$ is given by

$$T_{\mathbf{w}}\mathcal{V}_{d,D} = \left\{\mathbf{Z} \in \mathbb{R}^{D\times d} : \mathbf{Z}^T\mathbf{w} + \mathbf{w}^T\mathbf{Z} = 0\right\} \tag{A.4}$$

and the projection of a vector $\mathbf{u}$ on $T_{\mathbf{w}}\mathcal{V}_{d,D}$ is given by

$$\Pi_{\mathbf{w}}\mathbf{u} = \mathbf{u} - \frac{1}{2}\mathbf{w}\left(\mathbf{w}^T\mathbf{u} + \mathbf{u}^T\mathbf{w}\right). \tag{A.5}$$

Then one can see that for the Hamiltonian defined in eq. (30), the dynamics of $H^{[1]}$ imply that $\mathbf{w}(t)$ is constant everywhere, while $\mathbf{u}(t)$ satisfies

$$\mathbf{u}(t) = \mathbf{u}(0) + t\Pi_{\mathbf{w}_0}\nabla_{\mathbf{w}} \log p(\mathbf{w}|\boldsymbol{\phi}, \mathcal{Z})\big|_{\mathbf{w}=\mathbf{w}(0)}. \tag{A.6}$$

On the other hand, the dynamics of $H^{[2]}(\mathbf{w}, \mathbf{u})$ follow the geodesic flow given as [46]

$$[\mathbf{w}(t)\ \mathbf{u}(t)] = [\mathbf{w}(0)\ \mathbf{u}(0)] \exp\left\{t\begin{bmatrix} A & -S(0) \\ I & A \end{bmatrix}\right\}\begin{bmatrix} \exp\{-tA\} & 0 \\ 0 & \exp\{-tA\} \end{bmatrix}. \tag{A.7}$$

In the above, $A := \mathbf{w}^T\mathbf{u}(t)$ is constant over the geodesic and $S(t) = \mathbf{u}^T(t)\mathbf{u}(t)$. For the case where $d = 1$, that is $\mathcal{V}_{1,D}$ is the hypersphere $\mathcal{S}^{D-1}$, the geodesic flow simplifies to

$$[\mathbf{w}(t)\ \mathbf{u}(t)] = [\mathbf{w}(0)\ \mathbf{u}(0)] \begin{bmatrix} 1 & 0 \\ 0 & \beta^{-1} \end{bmatrix} \begin{bmatrix} \cos(\beta t) & -\sin(\beta t) \\ \sin(\beta t) & \cos(\beta t) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \beta \end{bmatrix}. \tag{A.8}$$

Here, $\beta := ||\mathbf{u}(t)||$ is the constant angular velocity. To initialize the Hamiltonian Monte Carlo sampling algorithm, the initial momentum v_0 in the arbitrary manifold case is sampled from a Gaussian $\mathcal{N}(0, G(x_0))$ which for $\mathcal{V}_{d,D}$ becomes $\mathbf{u}_0 \sim \mathcal{N}(0, \Pi_{\mathbf{w}_0})$. In practice we sample $u \sim \mathcal{N}(0, I_{D,d})$ and set $\mathbf{u}_0 = \Pi_{\mathbf{w}_0}(u)$.

Appendix B. The Matrix-Langevin distribution

The Matrix-Langevin, or von Mises-Fisher distribution defined on the Stiefel Manifold $\mathcal{V}_{d,D}$ has probability density given by

$$p(\mathbf{w}) = \frac{1}{c(\mathbf{F})} \exp\left\{\mathrm{Tr}\left[\mathbf{F}^T\mathbf{w}\right]\right\}, \tag{B.1}$$

where the normalizing constant is $c(\mathbf{F}) = {}_0F_1(\frac{1}{2}D, \frac{1}{4}\mathbf{F}^T\mathbf{F})$ and ${}_0F_1(\cdot,\cdot)$ is the hypergeometric constant with matrix arguments. It is often a convenient practice to parametrize the density function using a singular value decomposition (SVD) of the matrix $\mathbf{F} \in \mathbb{R}^{D\times d}$ such that $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are in $\mathcal{V}_{d,d}$ and $\mathcal{V}_{D,D}$ respectively and $\mathbf{\Sigma}$ is the $d \times D$ diagonal matrix that contains the singular values of $\mathbf{F}$. In this case, the normalizing constant simplifies to $c(\mathbf{F}) = {}_0F_1(\frac{1}{2}D, \frac{1}{4}\mathbf{\Sigma}^2)$ and the mode of the distribution is given by $\mathbf{U}\mathbf{V}^T$ [64]. This intuitively means that $\mathbf{U}$ and $\mathbf{V}$ are orientation matrices that determine the directions where $\mathbf{W}$ is concentrated while the diagonal entries of $\mathbf{\Sigma}$ control the level of concentration in those directions. Specifically, when $\mathrm{diag}(\mathbf{\Sigma}) \to 0$, the concentration becomes very broad and eventually converges to the uniform probability measure on the Stiefel manifold, while $\mathrm{diag}(\mathbf{\Sigma}) \to \infty$ results in Dirac measures on the directions specified by $\mathbf{U}$ and $\mathbf{V}$. To simplify things further, in our implementations we consider $\mathbf{V} = \mathbf{I}_D$, therefore $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}$ and the mode becomes $\mathbf{U}$ with the columns of $\mathbf{W}$ being thus, centered around the columns of $\mathbf{U}$. An illustrative configuration of $\mathbf{F}$ on $\mathcal{V}_{2,3}$ is shown in [47]. In our implementations, we take advantage of these features in order to adjust $\mathbf{\Sigma}$ accordingly so as to specify very broad or tight priors for the columns of $\mathbf{W}$ to be centered around orientations specified in $\mathbf{U}$ that become available from lower dimensional adaptations.

References

[1] R. Smith, Uncertainty quantification: theory, implementation, and applications (Vol. 12), SIAM, 2013.

[2] O. Le Maître, O. Knio, Spectral methods for uncertainty quantification: with applications to computational fluid dynamics, Springer Science & Business Media, 2010.

[3] P. Koch, T. Simpson, J. Allen, F. Mistree, Statistical approximations for multidisciplinary design optimization: the problem of size, Journal of Aircraft 36 (1999) 275–286.

[4] G. Wang, S. Shan, Review of metamodeling techniques in support of engineering design optimization, ASME. J. Mech. Des 129 (2007) 370–380.
URL https://doi.org/10.1115/1.2429697

[5] W. Morokoff, R. Caflisch, Quasi-monte carlo integration, Journal of computational physics 122 (1995) 218–230.
URL https://doi.org/10.1006/jcph.1995.1209

[6] A. Tarantola, Inverse problem theory and methods for model parameter estimation, Society for Industrial and Applied Mathematics, 2005.

[7] R. Ghanem, P. Spanos, Stochastic finite elements: a spectral approach, Courier Corporation, 2003.

[8] O. Knio, O. Le Maitre, Uncertainty propagation in cfd using polynomial chaos decomposition, Fluid dynamics research 38 (2006) 616.

[9] D. Xiu, G. Karniadakis, The wiener-askey polynomial chaos for stochastic differential equations, SIAM journal on scientific computing 24 (2002) 619–644.

[10] N. Cristianini, J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods, Cambridge university press, 2000.

[11] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (1995) 273–297.

[12] I. Bilionis, N. Zabaras, Multidimensional adaptive relevance vector machines for uncertainty quantification, SIAM Journal on Scientific Computing 34 (2012) B881–B908.
URL https://doi.org/10.1137/120861345
[13] P. Tsilifis, I. Papaioannou, D. Straub, F. Nobile, Sparse polynomial chaos expansions using variational relevance vector machines, Journal of Computational Physics (2020) 109498.
URL https://doi.org/10.1016/j.jcp.2020.109498
[14] R. Tripathy, I. Bilionis, Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification, Journal of computational physics 375 (2018) 565–588.
URL https://doi.org/10.1016/j.jcp.2018.08.036
[15] Y. Zhu, N. Zabaras, P. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, Journal of Computational Physics 394 (2019) 56–81.
URL https://doi.org/10.1016/j.jcp.2019.05.024
[16] I. Bilionis, N. Zabaras, B. Konomi, G. Lin, Multi-output separable gaussian process: Towards an efficient, fully bayesian paradigm for uncertainty quantification, Journal of Computational Physics 241 (2013) 212–239.
URL https://doi.org/10.1016/j.jcp.2013.01.011
[17] P. Chen, N. Zabaras, I. Bilionis, Uncertainty propagation using infinite mixture of gaussian processes and variational bayesian inference, Journal of computational physics 284 (2015) 291–333.
[18] M. Raissi, P. Perdikaris, G. Karniadakis, Numerical gaussian processes for time-dependent and nonlinear partial differential equations, SIAM Journal on Scientific Computing 40 (2018) A172–A198.
URL https://doi.org/10.1137/17M1120762
[19] M. Kennedy, A. O'Hagan, N. Higgins, Bayesian analysis of computer code outputs, in: Quantitative methods for current environmental issues, pp. 227-243, Springer, London, 2002.
[20] A. O'Hagan, Bayes-hermite quadrature, Journal of statistical planning and inference 29 (1991) 245–260.
[21] I. Bilionis, N. Zabaras, Multi-output local gaussian process regression: Applications to uncertainty quantification, Journal of Computational Physics 231 (2012) 5718–5746.
URL https://doi.org/10.1016/j.jcp.2012.04.047
[22] M. Raissi, P. Perdikaris, G. Karniadakis, Machine learning of linear differential equations using gaussian processes, Journal of Computational Physics 348 (2017) 683–693.
URL https://doi.org/10.1016/j.jcp.2017.07.050
[23] S. Atkinson, N. Zabaras, Structured bayesian gaussian process latent variable model: Applications to data-driven dimensionality reduction and high-dimensional inversion, Journal of Computational Physics 383 (2019) 166–195.
URL https://doi.org/10.1016/j.jcp.2018.12.037
[24] C. E. Rasmussen, C. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006, iSBN 0-262-18253-X.
[25] R. Herbrich, N. Lawrence, M. Seeger, Fast sparse gaussian process methods: The informative vector machine, in: In Advances in neural information processing systems (pp. 625-632), 2003.
[26] J. Hensman, N. Lawrence, Gaussian processes for big data through stochastic variational inference, in: Adv Neural Inf Process Syst, 25, 2012.
[27] N. Cressie, Statistics for spatial data, John Wiley & Sons, 2015.
[28] M. Kennedy, A. O'Hagan, Predicting the output from a complex computer code when fast approximations are available, Biometrika 87 (2000) 1–13.
[29] A. Forrester, A. Sóbester, A. Keane, Multi-fidelity optimization via surrogate modelling, Proceedings of the royal society a: mathematical, physical and engineering sciences 463 (2007) 3251–3269.
[30] M. Kennedy, A. O'Hagan, Bayesian calibration of computer models, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63 (2001) 425–464.
[31] L. Le Gratiet, J. Garnier, Recursive co-kriging model for design of computer experiments with multiple levels of fidelity, International Journal for Uncertainty Quantification 4.
[32] P. Perdikaris, D. Venturi, J. Royset, G. Karniadakis, Multi-fidelity modelling via recursive co-kriging and gaussian-markov random fields, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 471 (2015) 20150018.
URL https://doi.org/10.1098/rspa.2015.0018
[33] P. Perdikaris, G. Karniadakis, Model inversion via multi-fidelity bayesian optimization: a new paradigm for parameter estimation in haemodynamics, and beyond, Journal of The Royal Society Interface 13 (2016) 20151107.
URL https://doi.org/10.1098/rsif.2015.1107
[34] G. Pang, P. Perdikaris, W. Cai, G. Karniadakis, Discovering variable fractional orders of advection-dispersion equations from field data using multi-fidelity bayesian optimization, Journal of Computational Physics 348 (2017) 694–714.
URL https://doi.org/10.1016/j.jcp.2017.07.052
[35] P. Perdikaris, D. Venturi, G. Karniadakis, Multifidelity information fusion algorithms for high-dimensional systems and massive data sets, SIAM Journal on Scientific Computing 38 (2016) B521–B538.
URL https://doi.org/10.1137/15M1055164
[36] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, S. Tarantola, Global sensitivity analysis: the primer, John Wiley & Sons, 2008.
[37] K. 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 (1901) 559–572.
[38] X. Ma, N. Zabaras, Kernel principal component analysis for stochastic input model generation, Journal of Computational Physics 230 (2011) 7311–7331.
[39] P. Constantine, E. Dow, Q. Wang, Active subspace methods in theory and practice: applications to kriging surfaces, SIAM

Journal on Scientific Computing 36 (2014) A1500–A1524.
[40] P. Constantine, Active subspaces: Emerging ideas for dimension reduction in parameter studies, Society for Industrial and Applied Mathematics, 2015.
[41] R. Lam, O. Zahm, Y. Marzouk, K. Willcox, Multifidelity dimension reduction via active subspaces, SIAM Journal on Scientific Computing 42 (2020) A929–A956.
[42] R. Tripathy, I. Bilionis, M. Gonzalez, Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation, Journal of Computational Physics 321 (2016) 191–223.
URL https://doi.org/10.1016/j.jcp.2016.05.039
[43] R. Garnett, M. Osborne, P. Hennig, Active learning of linear embeddings for gaussian processes, in: Conference on Uncertainty in Artificial Intelligence (UAI 2014), 2014.
[44] R. Moriconi, M. Deisenroth, K. Kumar, High-dimensional bayesian optimization using low-dimensional feature spaces, arXiv preprint arXiv:1902.10675 (2019).
[45] P. Perdikaris, M. Raissi, A. Damianou, N. Lawrence, G. Karniadakis, Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 473 (2017) 20160751.
[46] S. Byrne, M. Girolami, Geodesic monte carlo on embedded manifolds, Scandinavian Journal of Statistics 40 (2013) 825–845.
URL https://doi.org/10.1111/sjos.12036
[47] P. Tsilifis, R. Ghanem, Bayesian adaptation of chaos representations using variational inference and sampling on geodesics, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 474 (2018) 20180285.
URL https://doi.org/10.1098/rspa.2018.0285
[48] L. Le Gratiet, Bayesian analysis of hierarchical multifidelity codes, SIAM/ASA Journal on Uncertainty Quantification 1 (2013) 244–269.
[49] A. Gelman, G. Roberts, W. Gilks, Efficient metropolis jumping rules, Bayesian statistics 5 (1996) 599–608.
[50] P. Tsilifis, X. Huan, C. Safta, K. Sargsyan, G. Lacaze, J. Oefelein, H. Najm, R. Ghanem, Compressive sensing adaptation for polynomial chaos expansions, Journal of Computational Physics 380 (2019) 29–47.
URL https://doi.org/10.1016/j.jcp.2018.12.010
[51] T. Chaspari, A. Tsiartas, P. Tsilifis, S. Narayanan, Markov chain monte carlo inference of parametric dictionaries for sparse bayesian approximations, IEEE Transactions on Signal Processing 64 (2016) 3077–3092.
[52] R. Muirhead, Aspects of multivariate statistical theory (Vol. 197), John Wiley & Sons, 2009.
[53] M. Girolami, B. Calderhead, Riemann manifold langevin and hamiltonian monte carlo methods, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 73 (2011) 123–214.
URL https://doi.org/10.1111/j.1467-9868.2010.00765.x
[54] D. Hitchcock, A history of the metropolis-hastings algorithm, The American Statistician 57 (2003) 254–257.
[55] R. Neal, Mcmc using hamiltonian dynamics, Handbook of markov chain monte carlo 2 (2011) 2.
[56] Y. Chikuse, Statistics on special manifolds (Vol. 174), Springer Science & Business Media, 2012.
[57] S. Ghosh, P. Pandita, S. Atkinson, W. Subber, Y. Zhang, N. C. Kumar, S. Chakrabarti, L. Wang, Advances in bayesian probabilistic modeling for industrial applications, ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg 6 (3).
[58] R. Byrd, P. Lu, J. Nocedal, C. Zhu, A limited memory algorithm for bound constrained optimization, SIAM Journal on scientific computing 16 (1995) 1190–1208.
[59] C. E. Seeley, C. Wakelam, X. Zhang, D. Hofer, W. M. Ren, Investigations of flutter and aero damping of a turbine blade: Part 1-experimental characterization, in: ASME Turbo Expo 2016: Turbomachinery Technical Conference and Exposition, American Society of Mechanical Engineers, 2016.
[60] W. M. Ren, C. E. Seeley, X. Zhang, B. E. Mitchell, H. Ju, Investigations of flutter and aero damping of a turbine blade: Part 2-numerical simulations, ASME Turbo Expo 2016: Turbomachinery Technical Conference and Exposition, American Society of Mechanical Engineers.
[61] C. Bishop, Pattern recognition and machine learning, Springer-Verlag, New York, 2006.
[62] E. Hairer, C. Lubich, G. Wanner, Geometric numerical integration: structure-preserving algorithms for ordinary differential equations, Vol. 31, Springer Science & Business Media, 2006.
[63] R. Abraham, J. Marsden, J. Marsden, Foundations of mechanics (Vol. 36), Reading, Massachusetts: Benjamin/Cummings Publishing Company, 1978.
[64] C. Khatri, K. Mardia, The von misesfisher matrix distribution in orientation statistics, Journal of the Royal Statistical Society: Series B (Methodological) 39 (1977) 95–106.