Cholesky-based experimental design for Gaussian process and kernel-based emulation and calibration

H. Harbrecht, J.D. Jakeman, P. Zaspel

Departement Mathematik und Informatik
Fachbereich Mathematik
Universität Basel
CH-4051 Basel

Preprint No. 2020-05
November 2020

dmi.unibas.ch

Cholesky-based experimental design for Gaussian process and kernel-based emulation and calibration

H. Harbrecht[1],J.D. Jakeman[2*], and P. Zaspel[3]

[1] *Department of Mathematics and Computer Science, University of Basel, Basel, Switzerland.*
[2] *Optimization and Uncertainty Quantification, Sandia National Laboratories, Albuquerque, NM, USA.*
[3] *Computer Science and Electrical Engineering, Jacobs University Bremen, Bremen, Germany.*

Abstract. Gaussian processes and other kernel-based methods are used extensively to construct approximations of multivariate data sets. The accuracy of these approximations is dependent on the data used. This paper presents a computationally efficient algorithm to greedily select training samples that minimize the weighted L^p error of kernel-based approximations for a given number of data. The method successively generates nested samples, with the goal of minimizing error in high probability regions of densities specified by users. The algorithm presented is extremely simple and can be implemented using existing pivoted Cholesky factorization methods. Training samples are generated in batches which allows training data to be evaluated (labeled) in parallel. For smooth kernels, the algorithm performs comparably with the greedy integrated variance design but has significantly lower complexity. Numerical experiments demonstrate the efficacy of the approach for bounded, unbounded, multi-modal and non-tensor product densities. We also show how to use the proposed algorithm to efficiently generate surrogates for inferring unknown model parameters from data using Bayesian inference.

AMS subject classifications: 62F15, 62K20, 65D05

Key words: Experimental design, active learning, Gaussian process, radial basis function, uncertainty quantification, Bayesian inference

1 Introduction

This article considers the approximation of a function $u:\Gamma\to\mathbb{R}$ using kernel based interpolation. We are particularly interested in constructing approximations $\Pi_V(u)\in V\subset\mathcal{V}_\omega$

*Corresponding author. *Email addresses:* `helmut.harbrecht@unibas.ch` (H. Harbrecht), `jdjakem@sandia.gov` (J.D. Jakeman), `p.zaspel@jacobs-university.de` (P. Zaspel)

of u, which have a small approximation error

$$\epsilon_\omega(u,V,p) := \left(\int_\Gamma |u(\boldsymbol{y}) - (\Pi_V u)(\boldsymbol{y})|^p \omega(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} \right)^{1/p}. \tag{1.1}$$

when measured with respect to a measure $\omega : \Gamma \to \mathbb{R}$, which introduces the weighting on $\mathcal{V}_\omega$. In other words, we aim to minimize the approximation error measured using the ω-weighted $L^p(\Gamma)$-norm

$$\|e\|_{L^p_\omega(\Gamma)} := \left(\int_\Gamma |e(\boldsymbol{y})|^p \omega(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} \right)^{1/p}.$$

While we are in principle not bound to a specific choice of p, we will consider $p=2,4,6$ in this work.

Such weighted function approximation is often essential for uncertainty quantification (UQ). In this setting, u usually corresponds to the parameter-to-solution map of a numerical model – e.g. a partial differential equation (PDE) – the argument $\boldsymbol{y}$ is a finite-dimensional random variable, and ω is its probability density function (PDF). Approximations are built to reduce the number of computationally expensive simulations (i.e. point evaluations of u) needed to estimate statistics.

The techniques discussed in this paper are not limited to UQ of PDEs. The method presented can be utilized to build approximations of any data generating information source. Furthermore, ω need not be a probability measure. For example, ω can be the unnormalized posterior of Bayesian inference, or a biasing measure used for importance sampling that explores important regions of Γ.

Numerous techniques have been developed to build approximations of expensive information sources. Within the computational science and engineering community, some of the most widely adopted methods for approximating models are those based on generalized polynomial chaos expansions [9, 39], sparse grid approximation [23, 38], Gaussian process models [27, 44], low-rank tensor decompositions [12, 24] and neural networks [45, 46]. These methods can be very efficient when building approximations of functions parameterized by independent random variables. However, there is a dearth of algorithmic options for when the variables are dependent.

The objective of this article is to propose a methodology to efficiently generate nested and greedy-pseudo-optimal sample designs that minimize the number of evaluations of u needed to build an approximation with a pre-specified accuracy measured in the L^p_ω norm. The algorithm is pseudo optimal because it minimizes an upper bound on approximation error. The algorithm is greedy because it both selects from a candidate set which discretizes the design space and chooses points one at a time.

Accurate approximations can be built without tailoring the sampling and approximation strategy to the measure ω. However such approaches, called *domination methods* [2, 14], are sub-optimal and require a larger number of evaluations of u than methods that consider ω [15]. Instead of building an approximation which minimizes the error measured by the L^p_ω norm, *domination methods* build an approximation that minimizes the error measured by another norm L^p_g, where g is a simpler measure. Typically g is simply a constant, i.e. the PDF of independent bounded uniform variables on a bounded domain. The use of the simpler incorrect measure g allows the use of existing unweighted approximation methods, but comes at a cost [2,14,39].

The following lemma characterizes the accuracy in a ω-weighted norm given an approximation that is accurate in the g-weighted norm.

Theorem 1.1 (Strong convergence [2]). *Let $g:\hat{\Gamma}\to\mathbb{R}$ and $\omega:\Gamma\to\mathbb{R}$ denote two densities which satisfy*

$$\delta = 1 - \int_{\Gamma\cap\hat{\Gamma}} \omega(\boldsymbol{y})\mathrm{d}\boldsymbol{y}$$

Now assume that the error of the approximation $\Pi_g(u)$ of u satisfies

$$\epsilon := \|u-\Pi_g u\|_{L^p_g(\Gamma)}, \quad p\geq 1, \tag{1.2}$$

and that u is bounded with $C_u = \|u\|_{L^\infty(\Gamma)}$ Then, there holds

$$\|u-\Pi_g u\|_{L^p_\omega(\Gamma)} \leq C_r^{1/p}\epsilon + C_u\delta^{1/p}, \quad \textit{provided} \quad C_r := \max_{\boldsymbol{y}\in\Gamma\cup\hat{\Gamma}} \frac{\omega(\boldsymbol{y})}{g(\boldsymbol{y})} < \infty. \tag{1.3}$$

This theorem suggests that tailoring an approximation to the true measure ω can improve approximation accuracy. The constant C_r is characterized as the maximum deviation of g from the original measure ω. If Γ is unbounded, we can still apply approximation methods for bounded domains $\hat{\Gamma}$. However, this induces an additional error which is proportional to the probability δ (measured with respect to ω) of $\boldsymbol{y}$ falling outside the bounded domain.

In mesh-free kernel-based approximation, it is common, see e.g. [7], to evaluate u at samples drawn from quasi-random number sequences. However, these sequences were primarily designed for unweighted function approximation, i.e. they are good samples for $\mathcal{V}_g$ with g being a PDF of a uniformly distributed random variable. Theorem 1.1 suggests that we can significantly reduce the approximation error if we instead use samples optimized for the measure of interest ω.

In this article we present a ω-aware approximation technique based on nested greedy-pseudo-optimal sampling using a weighted version of the so-called *power function*, i.e. the worst case error functional in kernel-based approximation. The power function has been used to construct efficient sampling schemes for unweighted approximation [29]. Our new contribution is to employ a weighting in the sample allocation process, which adapts the samples to the measure ω. We will show that this approach outperforms approximations constructed using power function based sampling [17] and quasi Monte Carlo sampling. We also compare our approach with methods based upon Quasi Monte Carlo sequences and integrated variance (IVAR) experimental design [11, 43], also known as integrated mean squared error design, used by the Gaussian process community.

When ω is a tensor-product measure, our approach produces approximations with accuracy comparable to approximations built using samples obtained from inverse transform sampling [6, 16] applied to quasi Monte Carlo sequences. However, the latter approach can only be applied easily to tensor-product measures, while our approach is also applicable to non-product measures, for example that occur in Bayesian inference problems. We also demonstrate that our approach produces approximations with similar accuracy to those constructed using IVAR. However, the latter is much more computationally intensive and difficult to implement. Our approach only requires a simple linear algebra task (i.e. the pivoted Cholesky factorization), while IVAR requires a more complex and expensive procedure. Furthermore, unlike IVAR, we show Cholesky sampling can be used to minimize L^p_ω norms with values other than $p=2$.

While beyond the scope of this paper, it is worth mentioning that nested-greedy procedures, similar to those discussed here for kernel based approximation, have also been developed for other approximation methods. Discrete Leja sequences [15,41], often used for polynomial approximation, are constructed by using partially pivoted LU factorization to greedily select points from an ordered set of basis functions evaluated at a set of candidate samples. Similarly, the *Empirical Interpolation Method* utilizes complete pivoting to greedily select both basis functions and samples (*Magic points*) [42].

The rest of this article is organized as follows. In Section 2, we provide an overview of kernel-based function approximation. Then, in Section 3, we introduce a new approach for greedily generating samples for kernel based approximation based upon a weighted modification of pivoted Cholesky factorization. A detailed discussion on the relationships to existing strategies used for kernel based approximation is also provided. Finally, in Section 4, we provide numerical results that highlight the strengths and performance of our new approach.

2 Kernel-based function approximation

This section provides a summary of radial basis function approximation using scattered data. Our exposition is similar to the discussion on function approximation in finite-dimensional *reproducing kernel Hilbert spaces* found in [34]. We also discuss the connections between radial basis function and Gaussian process approximation.

2.1 Reproducing kernel Hilbert spaces

Definition 2.1 (Reproducing kernel Hilbert space)**.** A *reproducing kernel* $\mathcal{K}$ for a general Hilbert space $\mathcal{V}$ with inner product $(\cdot,\cdot)_\mathcal{V}$ is a function $\mathcal{K}:\Gamma\times\Gamma\to\mathbb{R}$ such that

1. $\mathcal{K}(\cdot,\boldsymbol{y})\in\mathcal{V}$ for all $\boldsymbol{y}\in\Gamma$,

2. $u(\boldsymbol{y})=(u,\mathcal{K}(\cdot,\boldsymbol{y}))_\mathcal{V}$ for all $u\in\mathcal{V}$ and all $\boldsymbol{y}\in\Gamma$.

A Hilbert space $\mathcal{V}$ with reproducing kernel $\mathcal{K}:\Gamma\times\Gamma\to\mathbb{R}$ is called *reproducing kernel Hilbert space (RKHS).*

A continuous kernel $\mathcal{K}:\Gamma\times\Gamma\to\mathbb{R}$ is denoted *positive semi-definite* on $\Gamma\subseteq\mathbb{R}^d$, if we have

$$\sum_{j=1}^{N}\sum_{j'=1}^{N}\alpha_j\alpha_{j'}\mathcal{K}(\boldsymbol{y}_j,\boldsymbol{y}_{j'})\geq 0 \tag{2.1}$$

for all $N\in\mathbb{N}$, all pairwise distinct $X=\{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_N\}\subset\Gamma$, and all $\boldsymbol{\alpha}=(\alpha_1\cdots\alpha_N)^\top\in\mathbb{R}^N\setminus\{0\}$. It becomes *positive definite* if the inequality in (2.1) holds strictly.

In this article, we use radial basis functions, i.e. *radial kernels*

$$\mathcal{K}(\boldsymbol{y},\boldsymbol{y}'):=\varphi(\|\boldsymbol{y}-\boldsymbol{y}'\|) \tag{2.2}$$

with $\varphi:[0,\infty)\to\mathbb{R}$. A typical example of such a radial kernel is the *Gaussian / squared exponential kernel*, with

$$\varphi(r)=e^{-\epsilon^2r^2}. \tag{2.3}$$

which is a special instance of the class of *Matérn kernels* given by

$$\varphi(r)=\frac{K_{\nu-\frac{d}{2}}(\epsilon^2r)(\epsilon^2r)^{\nu-\frac{d}{2}}}{2^{\nu-1}\Gamma(\nu)},\quad \nu>\frac{d}{2}. \tag{2.4}$$

Here K_ν being the modified Bessel function of the second kind of order ν and Γ is the gamma function. Note that the parameter ν dictates for the smoothness of the kernel function. The analytic squared-exponential kernel can be obtained as $\nu\to\infty$.

The first property of Definition 2.1 implies that $\mathcal{V}$ contains all functions of the form

$$u=\sum_{j=1}^{N}\alpha_j\mathcal{K}(\cdot,\boldsymbol{y}_j) \tag{2.5}$$

provided that the points $\{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_N\}$ satisfy $\boldsymbol{y}_j\in\Gamma$. Vice versa, we can construct the *native space* $\mathcal{N}_\mathcal{K}(\Gamma)$ for a given symmetric, positive definite kernel $\mathcal{K}$ by completion of the pre-Hilbert space

$$F_\mathcal{K}(\Gamma):=\operatorname{span}\{\mathcal{K}(\cdot,\boldsymbol{y}):\boldsymbol{y}\in\Gamma\}.$$

It is shown in [34] that $\mathcal{N}_\mathcal{K}(\Gamma)$ is a RHKS for $\mathcal{K}$. For functions of the form (2.5), the native space carries the norm induced from the inner product

$$\left(\sum_{j=1}^{N}\alpha_j\mathcal{K}(\cdot,\boldsymbol{y}_j),\sum_{j'=1}^{N}\beta_{j'}\mathcal{K}(\cdot,\boldsymbol{y}_{j'})\right)_{\mathcal{N}_\mathcal{K}(\Gamma)}:=\sum_{j=1}^{N}\sum_{j'=1}^{N}\alpha_j\beta_{j'}\mathcal{K}(\boldsymbol{y}_j,\boldsymbol{y}_{j'}).$$

For example, the native space $\mathcal{N}_\mathcal{K}(\Gamma)$ of the Matérn kernel with $\nu>d/2$ and $\Gamma=\mathbb{R}^d$ is the well-known Sobolev space $H^\nu(\mathbb{R}^d)$.

2.2 Function approximation

The focus on this paper is the approximation of functions u that are elements of the native space of a kernel $\mathcal{K}$, i.e. $u \in \mathcal{N}_{\mathcal{K}}(\Gamma)$. With this goal we introduce a finite-dimensional approximation subspace of $\mathcal{N}_{\mathcal{K}}(\Gamma)$ by choosing a finite set of sampling points

$$X = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\} \subset \Gamma,$$

resulting in the finite-dimensional subspace

$$V(X) := \operatorname{span}\{\mathcal{K}(\cdot, \boldsymbol{y}) : \boldsymbol{y} \in X\} \subset \mathcal{N}_{\mathcal{K}}(\Gamma).$$

We are then interested in constructing a 'good' orthogonal projection $\Pi_{V(X)}(u)$ of $u \in \mathcal{N}_{\mathcal{K}}(\Gamma)$ to $V(X)$.

Interpolation. In reproducing kernel Hilbert spaces, the best approximation, given by the orthogonal projection $\Pi_{V(X)}$, and the *interpolation* of a function $u \in \mathcal{N}_{\mathcal{K}}(\Gamma)$ with given data $\{(\boldsymbol{y}_i, u(\boldsymbol{y}_i))\}_{i=1}^N$ turns out to be identical [34]. That is, we observe for the interpolant $\mathcal{I}_{V(X)} u$ the equality

$$(\mathcal{I}_{V(X)} u)(\boldsymbol{y}) = \Pi_{V(X)}(f)(\boldsymbol{y}) := \sum_{j=1}^{N} \alpha_j \mathcal{K}(\boldsymbol{y}, \boldsymbol{y}_j) \quad \text{for all} \quad \boldsymbol{y} \in \Gamma,$$

where the coefficients $\{\alpha_j\}_{j=1}^N, \alpha_j \in \mathbb{R}$, are computed by solving a linear system of equations with

$$\boldsymbol{A}_X \boldsymbol{\alpha} = \boldsymbol{u}, \quad \boldsymbol{\alpha} := (\alpha_1 \cdots \alpha_N)^\top, \quad \boldsymbol{u} := \big(u(\boldsymbol{y}_1) \cdots u(\boldsymbol{y}_N)\big)^\top \tag{2.6}$$

and the kernel matrix

$$\boldsymbol{A}_X := \begin{pmatrix} \mathcal{K}(\boldsymbol{y}_1, \boldsymbol{y}_1) & \ldots & \mathcal{K}(\boldsymbol{y}_1, \boldsymbol{y}_N) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(\boldsymbol{y}_N, \boldsymbol{y}_1) & \ldots & \mathcal{K}(\boldsymbol{y}_N, \boldsymbol{y}_N) \end{pmatrix}. \tag{2.7}$$

For positive definite kernels, $\boldsymbol{A}_X$ is symmetric, positive definite and regular.

Worst-case error. The *power function* [28, 36] or *kriging variance* [8] for a subspace $V(X)$ is the interpolation / approximation worst case error

$$P_{V(X)}(\boldsymbol{y}) = \sup_{f \in \mathcal{N}_{\mathcal{K}}(\Gamma), f \neq 0} \frac{|f(\boldsymbol{y}) - (\mathcal{I}_{V(X)} f)(\boldsymbol{y})|}{\|f\|_{\mathcal{N}_{\mathcal{K}}(\Gamma)}}. \tag{2.8}$$

For more detail refer to [34]. The power function gives us a simple estimate for the interpolation error.

Theorem 2.1 (Power function interpolation error [7]). *Let Γ, $\mathcal{K}$, X be as before and we have $f\in\mathcal{N}_{\mathcal{K}}(\Gamma)$ with its interpolant $\mathcal{I}_{V(X)}(f)$ on X. Then, there holds the error bound*

$$|f(\boldsymbol{y})-\mathcal{I}_{V(X)}(\boldsymbol{y})|\leq P_{V(X)}(\boldsymbol{y})\|f\|_{\mathcal{N}_{\mathcal{K}}(\Gamma)}$$

for all $\boldsymbol{y}\in\Gamma$.

The proof is trivial based on the above definition of the power function. Following Theorem 2.1, the upper bound for the interpolation error decouples into a product of the norm of the function and a term which only depends on the particular kernel and the samples in X. The first term requires knowledge of the function, so we focus on constructing designs that minimize the second term, that is the power function.

The power function can be evaluated easily using the following theorem.

Theorem 2.2 ([7]). *For two sets X,X' with elements $\boldsymbol{y}_j$ and $\boldsymbol{y}'_{j'}$ and sizes $|X|=N$, $|X'|=N'$, we use the extended notation*

$$\boldsymbol{A}_{X,X'}:=\begin{pmatrix}\mathcal{K}(\boldsymbol{y}_1,\boldsymbol{y}'_1) & \dots & \mathcal{K}(\boldsymbol{y}_1,\boldsymbol{y}'_{N'})\\ \vdots & \ddots & \vdots\\ \mathcal{K}(\boldsymbol{y}_N,\boldsymbol{y}'_1) & \dots & \mathcal{K}(\boldsymbol{y}_n,\boldsymbol{y}'_{N'})\end{pmatrix}. \tag{2.9}$$

Then, the power function (2.8) *can numerically be evaluated by*

$$P_{V(X)}(\boldsymbol{y})=\sqrt{\mathcal{K}(\boldsymbol{y},\boldsymbol{y})-\boldsymbol{A}_{X,\{y\}}^{\top}\boldsymbol{A}_X^{-1}\boldsymbol{A}_{X,\{y\}}}. \tag{2.10}$$

Note that the problem of finding a point set $X\subset\Gamma$ which minimizes the power function corresponds to approximating a given matrix by a low-rank approximation. Finding the best possible point set is therefore strongly related to the search for submatrices of maximal volume, see [10], and to adaptive cross approximation [1,13].

Remark 2.1. For large sets X, the evaluation of the power function by (2.10) can become unstable. A stable evaluation becomes possible by using the Newton basis of $V(X)$. We will discuss this procedure in Section 3.

Regularized function approximation. For large N, $\boldsymbol{A}_X$ often becomes ill-conditioned. This is why we usually consider a regularized function approximation. In regularized kernel-based approximation, which is sometimes also called *kernel ridge regression*, Tikhonov regularization [32] is used. It replaces the original linear system of equations by a modified, but more stable one

$$(\boldsymbol{A}_X+\epsilon_{reg}\boldsymbol{I})\boldsymbol{\alpha}=\boldsymbol{u},$$

where $\boldsymbol{I}$ the identity matrix. Depending on the size of the regularization parameter ϵ_{reg}, the matrix $\boldsymbol{A}_X+\epsilon_{reg}\boldsymbol{I}$ can have a much smaller condition number. Nonetheless, regularization introduces a small error of the order of the regularization parameter ϵ_{reg}.

2.3 Relation to Gaussian process regression

There is a strong connection between radial basis function (RBF) approximation and Gaussian processes. RBF approximations are treated as deterministic, where as Gaussian process (GP) regression uses Bayesian inference to provide probabilistic approximations [25,27,35]. Let us assume that for a given function $u:\Gamma\to\mathbb{R}$, we have a set of sample locations $X=\{\boldsymbol{y}_1,\dots,\boldsymbol{y}_N\}$ and *noisy* evaluations $\hat{u}_i:=u(\boldsymbol{y}_i)+\xi_i$, $\xi_i\sim\mathcal{N}(0,\sigma^2)$, collected in a vector $\hat{\boldsymbol{u}}$. Given a kernel $\mathcal{K}$ and mean function $m(\boldsymbol{y})$, Gaussian process approximation assumes that the joint prior distribution of u, conditional on kernel hyper-parameters θ, e.g. r in (2.3), is multivariate normal such that

$$u(\cdot)\,|\,\theta\sim\mathcal{N}\left(m(\cdot),\mathcal{K}(\cdot,\cdot;\theta)+\sigma^2 I\right)$$

The posterior distribution of this Gaussian process at a sample location $\boldsymbol{y}$, conditioned on the finite set of noisy observations $\hat{\boldsymbol{u}}$, is given by

$$u(\cdot)\,|\,\theta,\hat{\boldsymbol{u}}\sim\mathcal{N}(m^\star(\cdot),\mathcal{K}^\star(\cdot,\cdot;\theta)),$$

where

$$m^\star(\boldsymbol{y})=m(\boldsymbol{y})+\boldsymbol{\alpha}^\top\boldsymbol{A}_{X,\{y\}},\quad \boldsymbol{\alpha}:=\left(\boldsymbol{A}_X+\sigma^2\boldsymbol{I}\right)^{-1}\left(\hat{\boldsymbol{u}}-m(\boldsymbol{y})\right)$$

and

$$C(\boldsymbol{y})=\mathcal{K}(\boldsymbol{y},\boldsymbol{y})-\boldsymbol{A}_{X,\{y\}}^\top\left(\boldsymbol{A}_X+\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{A}_{X,\{y\}}. \tag{2.11}$$

When $m\equiv 0$ and the same kernel function $\mathcal{K}$ is used, the posterior mean of a Gaussian process coincides with RBF function approximation obtained using Tikhonov regularization with $\epsilon_{reg}=\sigma^2$.

In general, the kernel variance, and the noise variance and the hyper-parameters θ of the kernel, such as the length-scale which is the inverse of r in (2.3), are unknown and need to be inferred from the data. The Gaussian process community typically estimates these hyper-parameter by maximizing log-marginal-likelihood

$$\log p(\hat{\boldsymbol{u}}|X,\boldsymbol{\theta})=-\frac{1}{2}\hat{\boldsymbol{u}}^\top(\boldsymbol{A}_X+\sigma^2\boldsymbol{I})^{-1}\hat{\boldsymbol{u}}-\frac{1}{2}\log|\boldsymbol{A}_X+\sigma^2\boldsymbol{I}|-\frac{N}{2}\log 2\pi. \tag{2.12}$$

We will show in Section 4 that weighted Cholesky sampling can be used to construct designs for RBF and GP approximation, while employing this optimization strategy to learn the kernel hyper-parameters.

3 Sampling strategies

In this section, we present various sampling strategies for kernel-based approximation and Gaussian process regression. We first present an existing approach for nested unweighted-greedy pseudo-optimal sampling using the power function, cf. (2.8). We then extend this approach to generate samples which minimize the weighted error functional (1.1) of the resulting approximations. These two methods are then compared with popular sampling strategies from the Gaussian process and radial basis function communities, based upon integrated variance experimental design and low-discrepancy sequences, respectively.

3.1 Greedy nested sampling using the power function

Global optimization problem. From Theorem 2.1, we observe that we can reduce the error of an approximation by finding a point set X such that the L_∞ norm becomes small. Given a large set of candidate points X_{cand}, the best possible subset of points $X^\star$ with $|X^\star| = m$ can be found by solving the following optimization problem:

$$X^\star = \underset{X \subset X_{cand}}{\operatorname{argmin}} \left\| P_{V(X)} \right\|_{L^\infty(\Gamma)} \quad \text{subject to} \quad |X| \leq m. \tag{3.1}$$

Unfortunately, solving the global optimization problem in (3.1) is often intractable. Consequently a number of greedy strategies have been introduced [7, 17, 34]. Note that the specific choice of X_{cand} has a strong impact on the error reduction in interpolation. In Section 4 we describe a procedure for for constructing a candidate set that represents the domain Γ well.

Greedy-pseudo-optimal nested sample selection. The greedy version of the optimization problem in (3.1) we consider uses two properties of the power function $P_{V(X)}$. The first property is given by the following lemma.

Lemma 3.1 (Monotonicity [17]). *With Γ satisfying an interior cone condition, $\mathcal{K}$ being positive definite and $X', X'' \subseteq \Gamma$ being finite sets of samples with $X' \subseteq X''$, it holds*

$$P_{V(X')}(\boldsymbol{y}) \geq P_{V(X'')}(\boldsymbol{y}) \quad \textit{for all} \quad \boldsymbol{y} \in \Gamma.$$

The proof of this lemma can be found in [30]. The interior cone condition is given by the following definition and should be fulfilled in all discussed examples.

Definition 3.1 (Interior cone condition [34]). A set $\Gamma \subset \mathbb{R}^d$ satisfies an *interior cone condition*, if there exists a radius $r > 0$ and an angle $\theta \in (0, \pi/2)$ such that for all $\boldsymbol{y} \in \Gamma$ a unit vector $\boldsymbol{\xi}(\boldsymbol{y})$ exists such that

$$C(\boldsymbol{y}, \boldsymbol{\xi}, \theta, r) := \{\boldsymbol{y} + \lambda \boldsymbol{y}' | \boldsymbol{y}' \in \mathbb{R}^d, \|\boldsymbol{y}'\|_2 = 1, \boldsymbol{y}^\top \boldsymbol{\xi}(\boldsymbol{y}) \geq \cos\theta, \lambda \in [0, r]\}$$

is contained in Γ.

Lemma 3.1 states that, as the number of samples in a nested sample set increases, the power function $P_{V(X)}(\boldsymbol{y})$ decreases monotonically for all $\boldsymbol{y} \in \Gamma$. This property guarantees, that when using a greedy optimization strategy to select nested samples, the power-function and approximation error will never increase.

The following property is also useful for constructing a greedy sampling scheme.

Lemma 3.2. *With Γ and $\mathcal{K}$ as before and $X = \{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n\} \subset \Gamma$ a finite set of samples, it holds*

$$P_{V(X)}(\boldsymbol{y}_j) = 0 \quad \textit{for all} \quad \boldsymbol{y}_j \in X.$$

This lemma states that the power function $P_{V(X)}$ is zero for all elements of X. This statement is obvious since the numerator in the definition of the power function becomes zero at all samples $\boldsymbol{y} \in X$ because $\mathcal{I}_{V(X)}$ is an *interpolant*.

The two aforementioned properties can be used to design the greedy sampling algorithm summarized in Algorithm 1. Let $X_{cand} \subset \Gamma$ be a large, but finite candidate set and assume a set $X_{j-1} \subset X_{cand}$, with $|X_{j-1}| = j-1$, has already been selected. Because the power function is zero at all samples X_{j-1}, we greedily choose the next sample $\boldsymbol{y}_j \in X_{cand} \setminus X_{j-1}$ at which $\|P_{V(X_{j-1})}\|$ is maximal. If X_{cand} adequately covers the full domain Γ then $\boldsymbol{y}_j$ will be a reasonable approximation to the solution of $\boldsymbol{y}_{max} = \operatorname{argmax}_{\boldsymbol{y}\in\Gamma} P_{V(X_{j-1})}(\boldsymbol{y})$ in the sense that $P_{V(X_{j-1}\cup\{\boldsymbol{y}_j\})}(\boldsymbol{y}_{max}) \approx 0$. This process is repeated until the desired number of samples is reached.

Algorithm 1 Greedy sample selection by the power function [17].

```
1: function GETGREEDYSAMPLES($\mathcal{K}$, $X_{cand}$, $m$)
2:     $\boldsymbol{y}_1 = \operatorname{argmax}_{\boldsymbol{y}\in X_{cand}} P_{V(\{\boldsymbol{y}\})}(\boldsymbol{y})$
3:     $X_1 = \{\boldsymbol{y}_1\}$
4:     for $j = 2,3,\ldots,m$ do
5:         $\boldsymbol{y}_j = \operatorname{argmax}_{\boldsymbol{y}\in X_{cand}\setminus X_{j-1}} |P_{V(X_{j-1})}(\boldsymbol{y})|$
6:         $X_j = X_{j-1} \cup \{\boldsymbol{y}_j\}$
7:     return $X_m$
```

Efficient and stable implementation. As mentioned before, the evaluation of the power function using equation (2.10) is not stable [29, 34]. To improve stability, [20] proposes to iteratively build a Newton basis $\{\mathfrak{N}_1,\ldots,\mathfrak{N}_j\}$ for $V(X_j)$ during the greedy algorithm (Algorithm 1). By using this Newton basis, the power function can be evaluated stably using the expression

$$P^2_{V(X_j)}(\boldsymbol{y}) = \mathcal{K}(\boldsymbol{y},\boldsymbol{y}) - \sum_{i=1}^{j} \mathfrak{N}_i^2(\boldsymbol{y}).$$

Moreover, it is discussed in [26] that greedy-pseudo-optimal sampling using the Newton basis of a positive definite kernel $\mathcal{K}$ can be performed by the *pivoted Cholesky factorization* [13].

Given the candidate set matrix $\boldsymbol{A}_{X_{cand}}$, the first j steps of the pivoted Cholesky factorization given in Algorithm 2 compute the column matrix $L_j \in \mathbb{R}^{N_{cand}\times j}$ and a permutation vector $\boldsymbol{p}$. Each column of L_j defines a Newton basis vector of $V(X_j)$ evaluated on the candidate set X_{cand}. The first j pivots in $\boldsymbol{p}$ denote the indices of the points from the candidate set that maximize the power-function, see [26].

The computational cost of the pivoted Cholesky factorization is dominated by the calculation of the new vector $\boldsymbol{\ell}_r$ in step 10 of Algorithm 2. Line 10 requires individually $O(r)$ operations. The inner loop of line 9 then requires $O\left(\sum_{i=r+1}^{n} r\right) = O(n \cdot r)$ operations. To compute the first j steps of the loop in line 5, we thus need $O\left(\sum_{r=1}^{j} n \cdot r\right) = O\left(n \cdot j^2\right)$ operations. Consequently selecting j from N_{cand} candidate points, requires $O\left(N_{cand} \cdot j^2\right)$ operations.

Note that, in practice, the entire matrix $\boldsymbol{A}_{X_{cand}}$ needs not to be computed. Instead, the entries $\mathcal{K}$ should only be evaluated when required. Also note, that this Cholesky procedure produces a nested set of samples and can be used to enrich an existing set of points. Given m samples we can add an additional sample from the candidate set X_{cand} by performing another factorization step on the matrix A and adjusting the current Cholesky factor using the steps associated with the inner loop of Algorithm 2.

Algorithm 2 Pivoted Cholesky factorization [13].

```
1:  function PIVOTEDCHOLESKY($\boldsymbol{A}$)                          ▷ $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ s.p.d.
2:      $r = 1$                                                      ▷ current row
3:      $\boldsymbol{d} = \mathrm{diag}(\boldsymbol{A})$                                        ▷ diagonal of $\boldsymbol{A}$
4:      $\boldsymbol{p} = (1, \dots, n)$                                   ▷ initialization of permutation
5:      while $r \leq n$ do
6:          $i_{max} = \mathrm{argmax}_{j \in \{r, r+1, \dots, n\}} d_{p_j}$                  ▷ find pivot
7:          swap $p_r$ and $p_{i_{max}}$                                ▷ exchange columns
8:          $\ell_{p_r, r} = \sqrt{d_{p_r}}$                                  ▷ compute diagonal entry
9:          for $i \in \{r+1, \dots, n\}$ do                           ▷ comp. other entries
10:             $\ell_{p_i, r} = \left(a_{p_r, p_i} - \sum_{\nu=1}^{r-1} \ell_{p_r, \nu} \ell_{p_i, \nu}\right) / \ell_{p_r, r}$
11:             $d_{p_i} = d_{p_i} - \ell_{p_i, r}^2$
12:         $r = r + 1$                                              ▷ go to next row
        return $\left(\boldsymbol{L} = (\ell_{j,j'})_{j,j'}, \boldsymbol{p}\right)$
```

3.2 Sampling using a weighted power function

In accordance with Theorem 2.1, the sampling strategy from the last subsection is tailored to approximating the unweighted interpolant, i.e., it minimizes $|f(\boldsymbol{y}) - \mathcal{I}_{V(X)}(\boldsymbol{y})|$. With the goal of generating approximations that are accurate with respect to the weighted L^p_ω norm (1.1), we aim to generate approximations that minimize

$$\left|(f(\boldsymbol{y}) - (\mathcal{I}_{V(X)} f)(\boldsymbol{y}))g(\boldsymbol{y})\right|$$

for the weight function $g(\boldsymbol{y}) = \omega(\boldsymbol{y})^{\frac{1}{p}}$. This choice is motivated by noting that

$$\epsilon_\omega(u, V, p) := \left(\int_\Gamma |u(\boldsymbol{y}) - (I_V u)(\boldsymbol{y})|^p \omega(\boldsymbol{y}) \mathrm{d}\boldsymbol{y}\right)^{1/p} = \left(\int_\Gamma \left|(u(\boldsymbol{y}) - (I_V u)(\boldsymbol{y}))g(\boldsymbol{y})\right|^p \mathrm{d}\boldsymbol{y}\right)^{1/p}.$$

With the goal of generating samples tailored to fg, instead of f, we define the *weighted kernel*

$$\tilde{\mathcal{K}}(\boldsymbol{y},\boldsymbol{y}'):=g(\boldsymbol{y})\mathcal{K}(\boldsymbol{y},\boldsymbol{y}')g(\boldsymbol{y}').$$

Then, given a function of the form

$$f(\boldsymbol{y})=\sum_{j=1}^{N}\alpha_j\mathcal{K}(\boldsymbol{y},\boldsymbol{y}_j),$$

we have

$$f(\boldsymbol{y})g(\boldsymbol{y})=\sum_{j=1}^{N}\tilde{\alpha}_j\tilde{\mathcal{K}}(\boldsymbol{y},\boldsymbol{y}_j),\quad \text{where}\quad \tilde{\alpha}_j=\frac{\alpha_j}{g(\boldsymbol{y}_j)}$$

and

$$\|fg\|_{\mathcal{N}_{\mathcal{K}}(\Gamma)}=\|f\|_{\mathcal{N}_{\tilde{\mathcal{K}}}(\Gamma)},$$

where $\mathcal{N}_{\tilde{\mathcal{K}}}$ is the native space of the weighted kernel. If we further define the *weighted power function*

$$P_{g,V(X)}(\boldsymbol{y})=\sup_{f\in\mathcal{N}_{\mathcal{K}}(\Gamma),f\neq 0}\frac{\left|\left(f(\boldsymbol{y})-(\mathcal{I}_{V(X)}f)(\boldsymbol{y})\right)g(\boldsymbol{y})\right|}{\|fg\|_{\mathcal{N}_{\mathcal{K}}(\Gamma)}},\tag{3.2}$$

we immediately arrive at the error estimate

$$\left|\left(f(\boldsymbol{y})-(\mathcal{I}_{V(X)}f)(\boldsymbol{y})\right)g(\boldsymbol{y})\right|\leq P_{g,V(X)}(\boldsymbol{y})\|fg\|_{\mathcal{N}_{\mathcal{K}}(\Gamma)}.$$

In the weighted analogy to Theorem 2.2, the weighted power function (2.8) is equal to

$$P_{g,V(X)}(\boldsymbol{y})=\sqrt{g(\boldsymbol{y})\mathcal{K}(\boldsymbol{y},\boldsymbol{y})g(\boldsymbol{y})-\tilde{\boldsymbol{A}}_{X,\{\boldsymbol{y}\}}^{\top}\tilde{\boldsymbol{A}}_{X,X}^{-1}\tilde{\boldsymbol{A}}_{X,\{\boldsymbol{y}\}}},$$

where

$$\tilde{\boldsymbol{A}}_{X,X'}:=\begin{pmatrix} g(\boldsymbol{y}_1)\mathcal{K}(\boldsymbol{y}_1,\boldsymbol{y}_1')g(\boldsymbol{y}_1') & \cdots & g(\boldsymbol{y}_1)\mathcal{K}(\boldsymbol{y}_1,\boldsymbol{y}_{N'}')g(\boldsymbol{y}_{N'}') \\ \vdots & \ddots & \vdots \\ g(\boldsymbol{y}_N)\mathcal{K}(\boldsymbol{y}_N,\boldsymbol{y}_1')g(\boldsymbol{y}_1') & \cdots & g(\boldsymbol{y}_N)\mathcal{K}(\boldsymbol{y}_N,\boldsymbol{y}_{N'}')g(\boldsymbol{y}_{N'}') \end{pmatrix}$$

for two sets X,X' with elements $\boldsymbol{y}_j$ and $\boldsymbol{y}_{j'}'$ and sizes $|X|=N$, $|X'|=N'$. Hence, the pivoted Cholesky factorization should be appropriately modified in order to find an optimal set of points X which minimizes the $L^\infty(\Gamma)$-norm of the weighted power function.

In order to adapt Algorithm 2 for the weighted power function, we shall make first the following considerations. Let $\boldsymbol{A}=(a_{j,j'})_{j,j'}\in\mathbb{R}^{n\times n}$ be a symmetric and positive (semi-) definite matrix and $\boldsymbol{D}=\operatorname{diag}(d_1,\ldots,d_n)\in\mathbb{R}^{n\times n}$ be a diagonal matrix with positive diagonal entries $d_j>0$. Then, the pivoted Cholesky factorization for the matrix

$$\tilde{\boldsymbol{A}}=\left(\tilde{a}_{j,j'}\right)_{j,j'}=\left(d_ja_{j,j'}d_{j'}\right)_{j,j'}=\boldsymbol{DAD}$$

yields a rank-m approximation $\tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^\top$ with $\tilde{\boldsymbol{L}} = (\tilde{\ell}_{j,j'})_{j,j'} \in \mathbb{R}^{n\times m}$:

$$\tilde{\boldsymbol{A}} = \boldsymbol{DAD} \approx \tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^\top.$$

The matrix $\boldsymbol{L} = (\ell_{j,j'})_{j,j'} := \boldsymbol{D}^{-1}\tilde{\boldsymbol{L}}$ obviously results in a rank-m factorization $\boldsymbol{A} \approx \boldsymbol{L}\boldsymbol{L}^\top$, where it is especially inferred that

$$\tilde{\ell}_{j,j'} = d_j \ell_{j,j'} \quad \text{for all} \quad j = 1,\dots,n,\ j' = 1,\dots,m.$$

The only difference between applying the pivoted Cholesky factorization for $\boldsymbol{A}$ and for $\tilde{\boldsymbol{A}}$ is thus the different choice of the pivots. Therefore, by setting

$$\boldsymbol{A} = \big(\mathcal{K}(\boldsymbol{y}_j, \boldsymbol{y}_{j'})\big)_{j,j'}, \quad \boldsymbol{D} = \operatorname{diag}\big(g(\boldsymbol{y}_1), \dots, g(\boldsymbol{y}_{N_{cand}})\big),$$

we readily verify that the pivoted Cholesky factorization with respect to

$$\tilde{\boldsymbol{A}} = \big(g(\boldsymbol{y}_j)\mathcal{K}(\boldsymbol{y}_j, \boldsymbol{y}_{j'})g(\boldsymbol{y}_{j'})\big)_{j,j'}$$

can be realized by Algorithm 3, where we directly compute the matrix $\boldsymbol{L}$ as output instead of the matrix $\tilde{\boldsymbol{L}}$.

Algorithm 3 Weighted pivoted Cholesky factorization for kernels.

```
1:  function WEIGHTEDPIVOTEDCHOLESKY($X_{cand}$, $\mathcal{K}$, $g$, $m$)
2:      $r = 1$                                                        ▷ current row
3:      $N_{cand} = |X_{cand}|$
4:      $\boldsymbol{d} = \operatorname{diag}\big(\mathcal{K}(\boldsymbol{y}_1,\boldsymbol{y}_1),\dots,\mathcal{K}(\boldsymbol{y}_{N_{cand}},\boldsymbol{y}_{N_{cand}})\big)$
5:      $\boldsymbol{p} = (1,\dots,N_{cand})$                          ▷ initialization of permutation
6:      while $r \le m$ do
7:          $i_{max} = \operatorname{argmax}_{j\in\{r,r+1,\dots,N_{cand}\}}\{d_{p_j} g^2(\boldsymbol{y}_{p_j})\}$   ▷ find weighted pivot
8:          swap $p_r$ and $p_{i_{max}}$                               ▷ exchange columns
9:          $\ell_{p_r,r} = \sqrt{d_{p_r}}$                             ▷ compute diagonal entry
10:         for $i \in \{r+1,\dots,N_{cand}\}$ do                       ▷ comp. other entries
11:             $\ell_{p_i,r} = \big(a_{p_r,p_i} - \sum_{v=1}^{r-1}\ell_{p_r,v}\ell_{p_i,v}\big)/\ell_{p_r,r}$
12:             $d_{p_i} = d_{p_i} - \ell^2_{p_i,r}$
13:         $r = r+1$                                                   ▷ go to next row
        return $\big(\boldsymbol{L} = (\ell_{j,j'})_{j,j'}, \boldsymbol{p}\big)$
```

Note that weighted pivoted Cholesky factorization can be implemented with existing pivoted Cholesky algorithms. To do this one simply needs to factor the matrix DAD, but this requires forming these matrices at initiation, which is not necessary with Algorithm 3.

3.3 Integrated variance experimental design

An experimental design strategy, motivated by goals similar to those discussed here, was developed for Gaussian process regression in [43]. In this approach, optimal experimental designs, i.e. optimal samplings, are created by minimizing the a-posterior integrated variance (IVAR).

The a-posteriori variance of a Gaussian process, cf. Subsection 2.3, conditioned on input samples X is

$$c(\boldsymbol{y}\,|\,X) = \mathcal{K}(\boldsymbol{y},\boldsymbol{y}) - A_{X,\{y\}}^{\top}\left(A_X + \sigma^2 \boldsymbol{I}\right)^{-1} A_{X,\{y\}}.$$

IVAR design finds a set of samples $X^\star \subset \mathcal{U} \subset \Gamma$ of size $|X^\star| = m$, from a set of feasible *experiments* $\mathcal{U}$, by solving the minimization problem

$$X^\star = \underset{X\subset\mathcal{U},|X|=m}{\operatorname{argmin}} \int_\Gamma c(\boldsymbol{y}|X)\omega(\boldsymbol{y})\mathrm{d}\boldsymbol{y}. \tag{3.3}$$

In accordance with (2.10), the posterior variance $c(\boldsymbol{y}\,|\,X)$ in Gaussian process regression for a noise variance of $\sigma^2 = 0$ is just the square of the power function, i.e. we have

$$c(\boldsymbol{y}|X) \equiv P_{V(X)}^2.$$

Hence, the minimization of (3.3) is in fact equivalent to the minimization

$$X^\star = \underset{X\subset\mathcal{U},|X|=m}{\operatorname{argmin}} \int_\Gamma P_{V(X)}^2(\boldsymbol{y})\omega(\boldsymbol{y})\mathrm{d}\boldsymbol{y}.$$

This means, instead of minimizing the $L^\infty(\Gamma)$-norm of the power function as proposed in the last section, IVAR minimizes the (squared) weighted $L^2_\omega(\Gamma)$-norm of the power function.

Computing IVAR designs is much more expensive than computing weighted power-function designs using pivoted Cholesky factorization. The integrated variance of a Gaussian process can be expressed as

$$\int_\Gamma P_{V(X)}^2(\boldsymbol{y})\omega(\boldsymbol{y})\mathrm{d}\boldsymbol{y} = 1 - \mathrm{Trace}\left[A_X^{-1}P\right], \qquad P = \int_\Gamma A_{X,\{y\}}A_{X,\{y\}}^{\top}\omega(\boldsymbol{y})\mathrm{d}\boldsymbol{y},$$

which can by minimized by maximizing Trace $\left[A_X^{-1}P\right]$.

When the kernel is separable, that is it can be represented as the product of univariate kernels (e.g. the squared exponential kernel), and the density $\omega(\boldsymbol{y})$ is also the tensor product of univariate marginals, then the matrix $P \in \mathbb{R}^{m\times m}$ can be computed using the product of 1D integrals. In this paper, we compute P using Monte Carlo quadrature, that is sampling from the probability density $\omega(\boldsymbol{y})$ and computing the sample average of P. Monte Carlo quadrature allows IVAR design to be used with many popular non-product kernels, such as the Matérn kernel (2.4) with finite ν, and applied to densities of dependent random variables. In the following we use 10000 samples to compute P. Increasing the number of Monte Carlo samples did not significantly change the properties of the resulting design.

The algorithm used to computed greedy IVAR designs is summarized in Algorithm 4. The UPDATEIVARTRACE and UPDATEINVERSECHOLESKYFACTOR algorithms are summarized in the Appendix. The loop at step 10 is the most computationally expensive part of the algorithm. Each iteration of the loop involves computing the trace component of the integrated variance at each of the remaining candidate points (step 11). This trace can be computed naively by forming and inverting $A_{X_{r-1}\cup \boldsymbol{y}}$ and computing $P(X_{r-1}\cup \boldsymbol{y})$, where X_{r-1} is the set previously selected training samples associated with the first r pivots of $\boldsymbol{p}$ and $y \in X_{cand} \setminus X_{r-1}$. This cost can be reduced by pre-computing these matrices and extracting only the relevant elements at the rth step, however inverting $A_{X_{r-1}\cup \boldsymbol{y}}$ requires $O(r^3)$ operations. Once $A^{-1}_{X_{r-1}\cup \boldsymbol{y}}$ has been computed, the Trace $\left[A_{X_{r-1}\cup \boldsymbol{y}} P(X_{r-1}\cup \boldsymbol{y})\right]$ can be computed in $O(r^2)$ operations using the identity Trace $\left[B^T C\right] = \sum_{i=1}^{r}\sum_{j=1}^{r} B_{ij} C_{ij}$.

Algorithm 4 Sample averaged integrated variance greedy design procedure.

```
1: function INTEGRATEDVARIANCEDESIGN($X_{cand}$, $\mathcal{K}$, $Y_{mc}$, $m$)
2:     $r = 1$                                                        ▷ current row
3:     $N_{cand} = |X_{cand}|$, $N_{mc} = |Y_{mc}|$
4:     $\boldsymbol{p} = (1,\dots,N_{cand})$                          ▷ initialization of permutation
5:     $A_{X_{cand}} = \mathcal{K}(X_{cand}, X_{cand})$               ▷ Precompute kernel matrix
6:     $P = \frac{1}{N_{mc}}\mathcal{K}(X_{cand}, Y_{mc})\mathcal{K}(X_{cand}, Y_{mc})^T$   ▷ Precompute integrals of kernel basis
7:     $L_0 = []$, $L_0^{-1} = []$                                    ▷ Initialize Cholesky factors
8:     $v_{best} = 0$                                                 ▷ Initialize IVAR trace
9:     while $r \le m$ do
10:        for $i \in \{r+1,\dots,N_{cand}\}$ do                      ▷ comp. IVAR for other points
11:            $v_i$ = UPDATEIVARTRACE($A_{X_{cand}}$, $\boldsymbol{p}$, $P$, $L_{r-1}^{-1}$, $v_{best}$)
12:        $i_{max} = \text{argmax}_{j \in \{r, r+1, \dots, N_{cand}\}} v_i$   ▷ find pivot
13:        $L_r, L_r^{-1}$ = UPDATEINVERSECHOLESKYFACTOR($L_{r-1}$, $L_{r-1}^{-1}$)
14:        $v_{best} = v_{p_{i_{max}}}$
15:        swap $p_r$ and $p_{i_{max}}$                               ▷ exchange points
16:        $r = r+1$
       return $[p_1,\dots,p_r]$                  ▷ Return the pivots of the selected candidate points
```

In this paper, we reduce the calculation of the trace to $O(r^2)$ operations by updating the current Cholesky decomposition instead of recomputing it from scratch for each candidate sample. Using the Cholesky update (see the Appendix), the inner loop at step 10 requires $O(\sum_{i=r}^{N_{cand}} r^2)$ operations. Thus, computing m samples using Algorithm 4 requires $O(N_{cand} \cdot m^3)$ operations. In contrast the naive procedure, which inverts A_{X_r}, requires

$O(N_{cand} \cdot m^4)$ operations. To our knowledge the discussion in this section represents the first discussion of using Cholesky updates to reduce the cost of computing IVAR designs.

3.4 Low discrepancy sequences

It is common in both the Gaussian process regression and radial basis function literature to use low-discrepancy sequences as point sets for approximation [5,21,37]. Low discrepancy sequences are typically used for approximation in unweighted spaces. However, if the PDF ω has product structure, i.e.

$$\omega(\boldsymbol{y}) = \omega_1(y_1) \cdot \ldots \cdot \omega_d(y_d),$$

then we can apply *inverse transform sampling* [6,16] to generate points sets that follow the underlying distribution of ω. To this end, let

$$F_{\omega_j}(y_j) = \int_{t \le y_j} \omega_j(t) \mathrm{d}t$$

denote the cumulative distribution function of the j-th variable y_j. Given a set of samples $x^\star$ drawn uniformly on [0,1], e.g. from an unweighted low-discrepancy sequence, we can obtain a new set of samples

$$X := \left\{ \left(F_{\omega_1}^{-1}(y_1), \ldots, F_{\omega_d}^{-1}(y_d) \right) : \boldsymbol{y} \in X^\star \right\},$$

which can be used for weighted approximation. In this article, we will use Halton sequences for building RBF approximations.

Note that inverse transform sampling can be applied to non-product densities. However, doing so requires the use of non-linear transformations, such as the Rosenblatt transformation which can be infeasible even in moderate dimensions and introduces strong non-linearities which decrease the accuracy of approximations for fixed sample sizes. A method for constructing low-discrepancy sequences for non-product measures, which does not requires non-linear transformations was developed in [47], however such attempts are rare. Consequently, unlike our weighted greedy strategy and weighted IVAR, the vast majority of low-discrepancy sequences cannot be used for approximation for PDFs of dependent variables. Nevertheless, we include the use of inverse transform sampling in a numerical test case with tensor-product density in order to provide a more complete method comparison.

4 Numerical examples

In this section, we demonstrate the efficacy of our proposed approach on a number of numerical examples and compare its performance with the alternative sampling approaches presented in Section 3. In all the following examples, we will measure the performance of an approximation using the relative weighed $L^p_\omega(\Gamma)$-norm, i.e.

$$\epsilon_{\omega,p}(u,V(X)) := \frac{\left(\int_\Gamma \left| u(\boldsymbol{y}) - (\Pi_{V(X)}u)(\boldsymbol{y}) \right|^p \omega(\boldsymbol{y}) \mathrm{d}\boldsymbol{y}\right)^{1/p}}{\left(\int_\Gamma |u(\boldsymbol{y})|^p \omega(\boldsymbol{y}) \mathrm{d}\boldsymbol{y}\right)^{1/p}}. \tag{4.1}$$

Unless otherwise stated, we set $p=2$. In Section 4.5, we investigate the performance of weighted Cholesky sampling for $p \neq 2$.

For the examples considered here, we cannot compute the error (4.1) exactly. Consequently, we approximate it by the Monte Carlo estimator

$$\tilde{\epsilon}_{\omega,p}(u,V(X)) := \frac{\left(\sum_{i=1}^N \left| u(\boldsymbol{y}_i) - (\Pi_{V(X)}u)(\boldsymbol{y}_i) \right|^p\right)^{1/p}}{\left(\sum_{i=1}^N |u(\boldsymbol{y}_i)|^p\right)^{1/p}},$$

where the samples $\boldsymbol{y}_i$ in the Monte Carlo estimator are drawn i.i.d. from the underlying distribution of the PDF ω. In the following examples we use $N=1000$ samples to compute the error in approximations built using the Gaussian/squared exponential and Matérn kernels from equations (2.3) and (2.4). We explore the the generation of samples in two situations; when the hyper-parameters of the kernel are fixed and when they are optimized with the procedure described in Subsection 2.3.

Unless otherwise stated we will build approximations of functions drawn from the native space of the kernel $\mathcal{K}$. Specifically we randomly construct functions

$$u(\boldsymbol{y}) = \mathcal{K}(\boldsymbol{y},Y)\eta \tag{4.2}$$

where Y are is a set of $N_Y=1000$, d-dimensional, samples drawn uniformly on the compact domain $\hat{\Gamma}$ from Theorem 1.1 and the elements of the vector $\eta \in \mathbb{R}^{N_Y}$ are drawn independently from a standard normal distribution. For bounded variables we $\hat{\Gamma}=\Gamma$. We also consider unbounded Gaussian variables, for which we set $\hat{\Gamma}=[a,b]^d$, where we choose the interval $[a,b]$ to contain 99.9% of the probability of the univariate marginals centered around the marginal mean. Note, this truncation is only used to construct random functions. We do not truncate the unbounded density when constructing weighted Cholesky designs.

Finally, for all greedy design methods we employ candidate sets X_{cand} that consist of 5000 samples drawn from $\omega(\boldsymbol{y})$ and the first 5000 samples of the Halton sequence defined over the domain $\hat{\Gamma}$. All the methods considered in the following examples have been made available in the software package PyApprox [49].

4.1 Impact of the dominating density

In Section 1 we argued, based upon Lemma 1.1, that incorporating knowledge of the measure ω into the sampling process improves the convergence of the L^p_ω error of an approximation by a (possibly large) constant factor. In this section, we provide numerical evidence to support this claim. With this goal we apply weighted Cholesky design, using the squared exponential (2.3), when $\omega(\boldsymbol{y})$ is the product

$$\omega^{\mathrm{Beta}(\alpha,\beta)}(\boldsymbol{y}) := \prod_{j=1}^{d} \omega_j^{\mathrm{Beta}(\alpha,\beta)}(y_j)$$

of the PDFs of univariate Beta random variables

$$\omega_j^{\mathrm{Beta}(\alpha,\beta)}(y_j) := \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} y_j^{\alpha-1}(1-y_j)^{\beta-1}$$

with parameters α and β, where Γ is the gamma function.

In Figure 1, we compare the convergence in error for different dominating measures g when $d=2,4,10$ and the parameters of the product-Beta density ω are $\alpha=\beta=20$. Here we set the length scale $\ell=\frac{1}{r}$ of the kernel to be 0.1, 0.25 and 0.5 for $d=2,4,10$, respectively. We define the dominating measures to also be product-Beta densities with parameters α_g and β_g and compare errors for different values of these parameters $\alpha_g=\beta_g=\tau$. For each choice of dominating measure we approximate the constant C_r from Lemma 1.1, using 10000 random samples from ω, as a measure of the distance between the densities g and ω. For this example $\delta=0$ (see Lemma 1.1). As the dominating measure approaches ω, i.e. $\alpha_g \to \alpha$ and $\beta_g \to \beta$ which corresponds to $C_r \to 1$, the dominating measure approximation becomes more efficient. The plot clearly shows that constructing an approximation from sample designs targeting a density g, which is not ω, results in a degradation of accuracy, and the penalty grows as the quantity C_r grows. Although using a dominating measure only affects the constant of convergence and not the rate of convergence (see Lemma 1.1), the change in the constant is significant.

Note the increase in error, for $\tau=20$ in the left plot, is due to the condition number of the kernel evaluated at the training samples becoming very large. The impact on condition number on the design will be discussed in more detail in Section 4.2.

4.2 Comparison to existing sampling strategies

With the next numerical study, we compare the performance of the weighted Cholesky sampling strategy to other existing approaches in the field, namely the unweighted Cholesky sampling strategy, the IVAR approach, Halton sequences, and the inverse transform sampling approach applied to Halton sequences, which we refer to as transformed Halton designs. Note that the last approach is only feasible for product densities. Therefore, we again consider random functions (4.2) with $\omega^{Beta(20,20)}$.

The left of Figure 2 plots the error in the squared-exponential kernel approximation for various sampling strategies as the number of samples is increased. We consider functions of $d=3$ and $d=6$ variables for which we set the correlation length of the kernel $l=\frac{1}{r}$

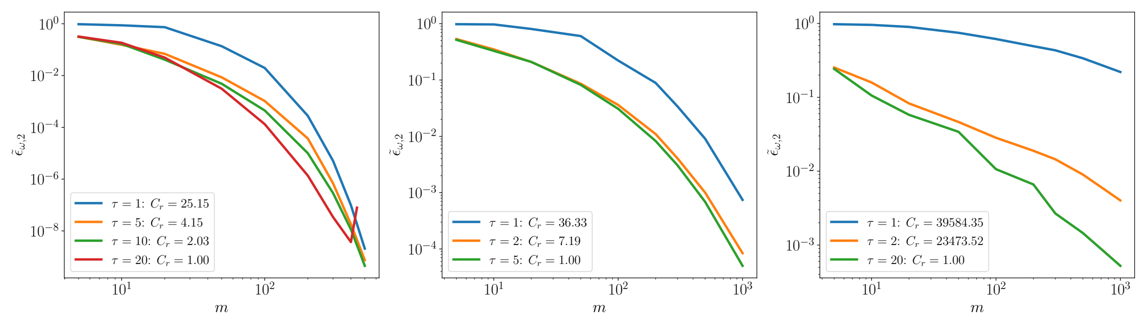

Figure 1: A comparison of the median error in approximations of 100 random functions (4.2), drawn from the native space of the squared-exponential kernel, obtained by weighted Cholesky sampling using different domination measures in (*left*) $d=2$, (*middle*) $d=4$ and (*right*) $d=10$ dimensions. Here, ω is a tensor product of identical univariate Beta PDFs with parameters $\alpha=\beta=20$ and we vary the distribution parameters of the domination measure $\alpha_g=\beta_g=\tau$. As predicted by Theorem 1.1 the error in the approximation increases as the distance between the dominating and true measure increases.

to be 0.4 and 0.5 respectively. Transformed Halton, IVAR and Weighted Cholesky perform comparably in both dimensions and out perform the domination methods, which do not leverage probability information.

The condition number of the kernel matrix evaluated at the training points can significantly affect the ability to compute large sample designs. When the condition number becomes larger than $O(10^{16})$ designs can no longer be enriched. The number of training points, at which this phenomenon occurs, depends on the sampling method, length scale and dimensionality. When the dimension is small and correlation length is long ill-conditioning occurs much sooner than when the dimension is high and/or the correlation length of the kernel is small. The condition number $\kappa=\text{Cond}[A_X]$ of each sample design is plotted in the right column of Figure 2. Weighted Cholesky sampling appears to be the least effected by ill-conditioning of any method that uses information on the density ω. This is indicated by the fact that it can select more samples before ill-conditioning terminates the algorithm. The dominating methods are better conditioned but have a significantly larger error.

Note that computing the trace component of the objective used to compute IVAR designs requires computing the inverse of the matrix A_X at all candidate samples $\boldsymbol{y}$ not selected in the current training set. Some of these points will produce very poorly conditioned matrices $A_{X\cup\boldsymbol{y}}$. We found that this can significantly degrade the performance of IVAR designs but identified that adding a small 'nugget' of 10^{-8} to the diagonal of $A_{X\cup\boldsymbol{y}}$ helped improve the designs significantly. Addition of a nugget is not necessary for Cholesky based designs.

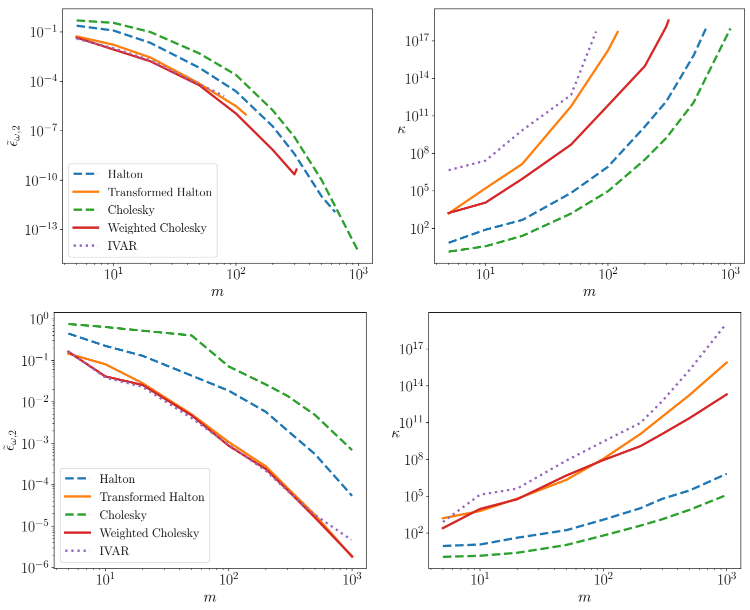

Figure 2: A comparison of the median error in approximations of 100 random functions (4.2), drawn from the native space of the squared-exponential kernel, obtained using different sampling strategies in (*top*) $d=3$ and (*bottom*) $d=6$ dimensions. The column on the right-and side shows the conditioning of the resulting designs

.

4.3 Sample designs for non-product PDFs

In this section we compare the performance of sampling strategies when used with non-product densities. Specifically, we consider a correlated Gaussian density, a multi-modal density and a non-product density with Beta marginals.

The first density we consider is a Gaussian distribution with mean zero and covariance

$$\left(\boldsymbol{R}^{V}\right)_{j,j'} = \begin{cases} 1, & \text{if } j=j', \\ (-1)^{j+j'}0.9, & \text{otherwise.} \end{cases} \tag{4.3}$$

whose PDF we denote ω^{Gauss}. The second density we consider is a mixture of two tensor-product Beta PDFs, that is

$$\omega^{mix}(\boldsymbol{y}) = \frac{1}{2}\left(\omega^{\text{Beta}(10,2)}(\boldsymbol{y}) + \omega^{\text{Beta}(5,10)}(\boldsymbol{y})\right)$$

We construct the third, and final, density using the *Nataf transform* [22] which assumes that the dependencies between variables can be expressed using a Gaussian copula. Specifically, we set

$$\omega^{Nat}(\boldsymbol{y}) = \frac{\eta_{R^V}(\boldsymbol{z})}{\prod_{j=1}^{d}\eta(z_j)}\prod_{j=1}^{d}\bar{\omega}_j(y_j), \tag{4.4}$$

with

$$\eta_{R^V}(z) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{R}^V)}} \exp\left(-\frac{1}{2} z^\top (\boldsymbol{R}^V)^{-1} z\right).$$

The $\bar{\omega}_j$ are the marginal probability distributions of ω^{Nat} and $z_j := \Phi^{-1}\left(F_j^{\omega^{Nat}}(y_j)\right)$, where η and Φ are the univariate standard normal PDF and CDF and $F_j^{\omega^{Nat}}(y_j)$ are the marginal cumulative distributions of ω^{Nat}. Here we set the marginals to be the PDFs of univariate Beta variables with parameters $(\alpha,\beta) = (2,5)$.

In the following we generate designs using the squared exponential kernel with correlation length $l = \frac{1}{r} = 0.1$ for the bounded PDFs and 0.6 for the unbounded Gaussian PDF. These values were chosen to ensure that the functions for all test cases had approximately the same variability, relative to the size of the compact domain $\hat{\Gamma}$ used to approximate Γ. Figure 3 depicts each PDF for $d=2$. The first 100 samples generated by the weighted Cholesky design for the squared exponential kernel are also shown. The samples clearly concentrate in regions of high-probability.

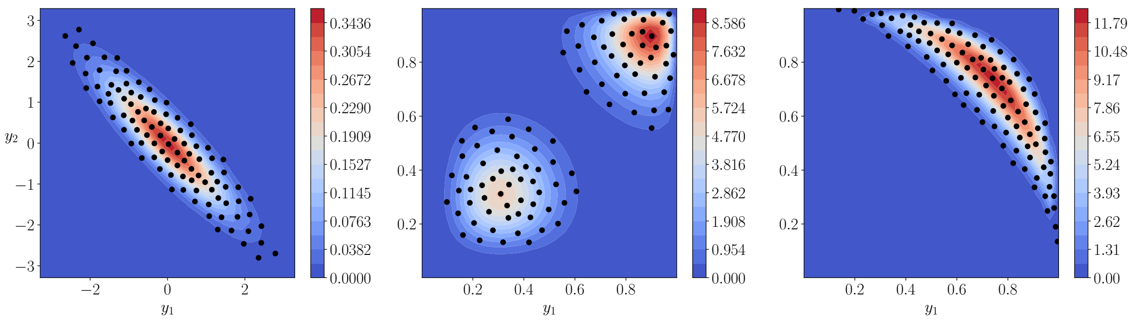

Figure 3: The first 100 samples generated by the weighted Cholesky design for the squared exponential kernel with (*left*) a correlated Gaussian density, (*middle*) a multi-modal density and (*right*) a Gaussian copula with Beta marginals. The coloring depicts the contours of each density.

For non-product PDFs, the inverse transform sampling approach is not applicable. Consequently, in Figure 4, we compare the convergence in error of $d=2$ dimensional approximations built using designs from unweighted pivoted Cholesky factorization, weighted pivoted Cholesky factorization and IVAR. Again IVAR and weighted Cholesky sampling are comparable for small sample sizes. But unlike the previous examples, the latter is more accurate for moderate to large sample sizes. The degradation in the performance of IVAR designs is due to the error introduced inverting the kernel matrix $A_{\mathcal{X}\cup \boldsymbol{y}}$ when adding the candidate sample $\boldsymbol{y}$ results in a nearly singular matrix. See the discussion in the previous section.

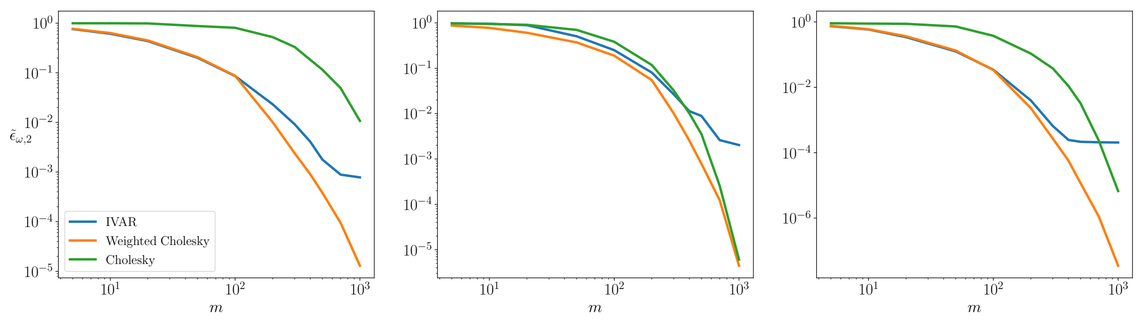

Figure 4: A comparison of the median error in approximations of 100 random functions (4.2), drawn from the native space of the squared-exponential kernel using different sampling strategies for (*left*) a correlated Gaussian density, (*middle*) a multi-modal density and (*right*) a Gaussian copula with Beta marginals.

4.4 Adaptive designs for unknown kernel hyper-parameters

In many situations, one does not know the hyper-parameters of a kernel that best represent the function being approximated. In this section we show that the kernel hyper-parameters can be learned while constructing weighted Cholesky designs.

Adaptive learning of the kernel hyper-parameters can be implemented with a minor modification of the pivoted Cholesky algorithm. Before any data is collected we assume that the correlation length of the kernel is $l = \frac{1}{r} = 1$. We then generate a sample design with m_0 points, evaluate u at these points and use the data obtained to estimate the kernel hyper-parameters by maximizing the marginal log-likelihood (2.12). If the kernel hyper-parameters are fixed we can enrich a sample design by simply storing and reusing the internal state of the pivoted Cholesky algorithm. When we change the kernel hyper-parameters we can no longer do this. Instead we must recompute the pivoted Cholesky factorization with the new kernel and enforce the new design is nested by ensuring that any pivots previously chosen are selected again before new pivots are identified.

In the following examples the kernel length scale is optimized when the number of samples in X equals $1, 2, \ldots, 10, 15, 20, 25, 50, 75, 100, 300$ and 500. We use the bound-constrained limited memory BFGS, available in Scipy (`https://www.scipy.org`), to maximize the marginal log-likelihood. Because the marginal log-likelihood is non-convex, we solve the optimization problem using 10 random initial guesses sampled uniformly in the hypercube $[10^{-3}, 1]^d$ and select the solution with the smallest objective.

Figure 5 plots the median error in approximations, of 10 random functions from the native space of the squared exponential kernel, constructed using weighted Cholesky samples when the correlation length of the kernel is optimized each time a batch of samples is requested (Weighted-Adaptive). We consider ω^{gauss} with $d = 3$ and $d = 6$ and compare performance with unweighted Cholesky sampling (Unweighted-Oracle) and weighted Cholesky sampling (Weighted-Oracle), which both use the true length scale of the random functions which is set to 0.1 and 0.5 in each dimension respectively. For both these examples, solving an optimization problem for the correlation length results in designs with accuracy comparable to the oracle designs. The difference in performance between the adaptive and oracle weighted designs is significantly smaller than the difference between using unweighted and weighted Cholesky sampling.

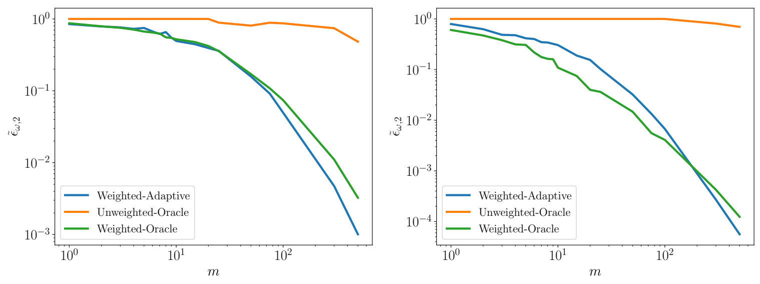

Figure 5: A comparison of the median error in approximations of 10 random functions (4.2), drawn from the (*left*) d=3 and (*right*) d=6 dimensional native spaces of the squared-exponential kernel using weighted-Cholesky factorization when the correlation length is not-known a priori but is rather optimized as data is collected.

Note, that when adaptively constructing designs, the final design is sensitive to the estimates of the kernel hyper-parameters during the early stages of point selection. We found that re-optimizing the hyper-parameters each time a sample is collected, for the first 10 samples, resulted in good designs. After this point samples can be collected in larger batches allowing for increasing parallelism when labeling the training data, that is evaluating the function at the new design samples.

4.5 Different weighted p-norms

In this section, we generate weighted Cholesky designs that target different values of q in $\epsilon_{\omega,q}(u, V(X))$. Figure 6 plots the median error in approximations of 100 three-dimensional random functions (4.2), drawn from the native space of the squared-exponential kernel using different sampling strategies; the length scale of the kernel is set to 0.1. Specifically, using $\omega^{Beta(20,20)}$, we compare IVAR designs with weighted Cholesky designs targeting values of $q=4$ and $q=6$. In both examples weighted Cholesky designs produce approximations which are more accurate than those produced using IVAR designs.

Figure 7 depicts the samples sets ($m=100$) generated, for ω^{Nat}, by the weighted and the unweighted Cholesky sampling strategies when d=2. The weighted approach clearly allocates more samples to regions of high-probability. However, as q is increased, more samples are allocated to regions of lower probability. IVAR cannot be tailored to a particular choice of q.

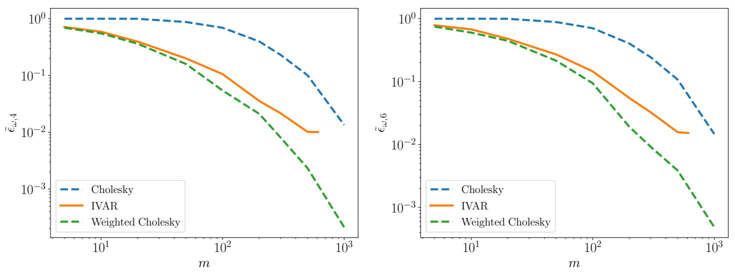

Figure 6: A comparison of the median error in approximations of 100 3D random functions (4.2), drawn from the native space of the squared-exponential kernel using weighted-Cholesky factorization minimizing (*left*) L^4_ω and (*right*) L^6_ω errors.

Figure 7: Comparison of samples sets, for ω^{Nat}, generated by *(left)* the unweighted Cholesky sampling strategy and *(middle)* the weighted strategy with $q{=}2$ and *(right)* $q{=}6$. The samples are plotted on top of the contours of the non-product density (4.4) with correlation (4.3), $d{=}2$, and univariate Beta marginals each with parameters $(\alpha,\beta)=(2,5)$.

4.6 Kernels of varying smoothness

This paper focuses on generating designs for the popular squared exponential kernel. However, weighted Cholesky sampling can also be applied to other kernels. In this section we investigate the performance of weighted Cholesky sampling for Matérn kernels of varying smoothness. For $\omega^{Beta(20,20)}$, Figure 8 compares the median accuracy of 3D approximations constructed using IVAR and weighted Cholesky designs for functions generated from the native space of the Matérn kernel with parameters $\nu=5/2$ and $\nu=9/2$. IVAR designs produce approximations which are more accurate than weighted Cholesky designs as the smoothness of the kernel decreases. Specifically IVAR produces more accurate approximations of functions drawn from the Matérn kernel with $\nu{=}5/2$ which have continuous second derivatives (see left of Figure 8). However, the difference between IVAR and weighted Cholesky is negligible for smoother functions with higher-order differentiability, e.g. $\nu=9/2$ which is 3 times differentiable (see right of Figure 8) and the squared exponential kernel which is analytic and obtained when $\nu\to\infty$ (see top-left of Figure 2). As ν, and thus smoothness decreases, unweighted Cholesky becomes more competitive for large sample sizes.

5 Bayesian inference

In this section, we will use Cholesky sampling to construct posterior surrogates for efficient Bayesian inference of model parameters from observational data, cf. [31]. To make

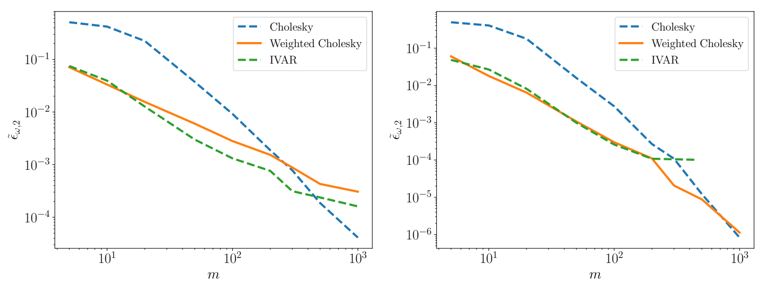

Figure 8: A comparison of the median error in approximations of 100 3D random functions (4.2), drawn from the native space of the Matérn kernel with (*left*) $\nu = 5/2$ and (*right*) $\nu = 9/2$.

this precise, let $f(\boldsymbol{y}): \mathbb{R}^d \to \mathbb{R}^n$ be a vector of n observable quantities, parameterized by d random variables $\boldsymbol{y}$. Bayes' rule can be used to define the posterior density for the model parameters $\boldsymbol{y}$ given observational data $\boldsymbol{d}$:

$$\pi(\boldsymbol{y}\,|\,\boldsymbol{d}) = \frac{\pi(\boldsymbol{d}\,|\,\boldsymbol{y})\pi(\boldsymbol{y})}{\int_{\mathcal{D}} \pi(\boldsymbol{d}\,|\,\boldsymbol{y})\pi(\boldsymbol{y})\mathrm{d}\boldsymbol{y}},$$

where any prior knowledge on the model parameters is incorporated into the prior density $\pi(\boldsymbol{y})$ and $\pi(\boldsymbol{d}\,|\,\boldsymbol{y})$ is the likelihood function which specifies the probability of observing data $\boldsymbol{d}$ given a realization of the model parameters.

5.1 Determining the likelihood

The form of the likelihood is determined by the statistical model that relates the data to the simulation model. In the following, we will assume the following relationship $\boldsymbol{d} = f(\boldsymbol{y}) + \boldsymbol{\epsilon}$, where the noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ is normally distributed with mean zero and covariance $\boldsymbol{\Sigma}$. Under this assumption, the likelihood takes the form

$$\pi(\boldsymbol{d}\,|\,\boldsymbol{y}) = \exp\left(-\frac{1}{2}(\boldsymbol{d} - f(\boldsymbol{y}))^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{d} - f(\boldsymbol{y}))\right) = \exp\left(-u(\boldsymbol{y})\right),$$

where u is often referred to as the negative log likelihood.

Often, when evaluating f, and thus u, is computationally expensive, a surrogate of either f or u is built to avoid the expensive evaluation of the model. For example, [18] proposes to replace the misfit u in the likelihood by an m-point approximation $\Pi_{m,g}u$ that is used to compute an approximation of the posterior distribution, i.e.

$$\pi_{m,g}(\boldsymbol{y}\,|\,\boldsymbol{d}) = \frac{\exp\left(-\Pi_{m,g}u(\boldsymbol{y})\right)\pi(\boldsymbol{y})}{\int_{\mathcal{D}} \exp\left(-\Pi_{m,g}u(\boldsymbol{y})\right)\pi(\boldsymbol{y})\mathrm{d}\boldsymbol{y}}. \tag{5.1}$$

The authors of [18] construct a polynomial chaos expansion (PCE) which is tailored to the prior distribution, i.e. $g(\boldsymbol{y}) = \pi(\boldsymbol{y})^{\frac{1}{2}}$. Using the prior as a dominating measure is inefficient, as it has often a much larger non-zero support than the posterior. Attempts have been made to increase the efficiency of Bayesian inference by using low-order localized surrogates to facilitate sampling in regions of high probability [4,19]. The use of localized surrogates results in small-rates of convergence. Recently, PCE tailored to the posterior distribution have been used to obtain higher rates of convergence [15, 33] and extended to use multiple models of varying cost and accuracy in [48].

5.2 Using weighted Cholesky sampling

In the following, we demonstrate that our weighted Cholesky sampling scheme can be used to efficiently construct RBF/Gaussian process surrogates of the misfit u. The adaptive algorithm we use is summarized in Algorithm 5 and described here.

Algorithm 5 Adaptive Cholesky sampling for Bayesian inference.

```
 1: function GETPOSTERIORAPPROX($\mathcal{K}$, $X_{cand}$, $\{\Delta m_j\}_{j=1}^{s}$)
 2:     $X_0 = \emptyset$, $u_0 = \emptyset$, $\beta_0 = 0$
 3:     for $j = 1,2,\ldots,s$ do
 4:         $g_j = \exp\left(-\Pi_{m_{j-1} g_{j-1}} u\right)^{\beta_{j-1}} \pi$
 5:         $\Delta X_j$ = UPDATEWEIGHTEDPIVOTEDCHOLESKY($X_{cand}$, $\mathcal{K}$, $X_{j-1}$,$g_j$, $\Delta m_j$)
 6:         $X_j = X_{j-1} \cup \Delta X_j$
 7:         $m_j = |X_j|$
 8:         $u_j = u_{j-1} \cup u(\Delta X_j)$                ▷ evaluate model at new samples
 9:         $\Pi_{m_j,g_j} u$ = APPROXIMATEFUNCTION($X_j$,$u_j$)
10:         $\beta_j$ = UPDATETEMPERINGPARAMETER($\beta_{j-1}$,$\Pi_{m_j,g_j} u$)
11:     return $X_s$, $\Pi_{m_s,g_s} u$
```

Because the true posterior density ω is unknown, we cannot directly use it to specify g. Instead, we will use a series of intermediate unnormalized PDFs which converge to the true distribution

$$g_j(\boldsymbol{y}) = \exp\left(-\Pi_{m_{j-1},g_{j-1}} u(\boldsymbol{y})\right)^{\beta_{j-1}} \pi(\boldsymbol{y}), \quad j = 1,\ldots,s, \tag{5.2}$$

where $0 = \beta_0 < \beta_1 < \ldots < \beta_s \leq 1$. A similar approach has been used to improve the performance of Markov chain Monte Carlo (MCMC) when sampling multi-modal posteriors [3]. Such methods are referred to as transitional MCMC.

Note that even when $\beta = 1$ the measure in (5.2) is not the posterior measure of the surrogate. The normalizing constant $\int_{\mathcal{D}} \exp\left(-\Pi_{m,g} u(\boldsymbol{y})\right) \pi(\boldsymbol{y}) \mathrm{d}\boldsymbol{y}$ is missing. However this constant does not effect the pivots chosen by the weighted Cholesky sampling procedure, and so can be ignored.

Starting with an initial set of points X_0 which is often the empty set, Algorithm 5 begins by setting the initial dominating measure to be the prior, i.e. $g_0(\boldsymbol{y}) = \pi(\boldsymbol{y})^{\frac{1}{q}}$. In the following we set $q = 2$. This measure is then adapted as information (evaluations) about u is obtained. Specifically, given an approximation $\Pi_{m_{j-1},g_{j-1}}u$ at iteration $j \geq 1$ built with m_{j-1} samples X_{j-1}, we set $g_j(\boldsymbol{y})$ using (5.2). This domination measure is then used to enrich the existing set of samples with another Δm_j samples ΔX_j, yielding $X_{j+1} = X_j \cup \Delta X_j$. This process continues until a sufficient accuracy in the posterior is reached or a computational budget is exhausted. Note that unlike the examples in Section 4 when the weighting density is known we cannot use samples from the unknown density in the candidate set X_{cand}. We can use samples from the posterior drawn from the surrogate using Markov Chain Monte Carlo sampling, however we found this unnecessary.

Our transitional approach has two advantages. Firstly, it can sample multi-model and/or concentrated PDFs. Moreover, it allows us to build in a level of conservativeness to prevent our weighted sampling approach from being misled during the initial stages of the algorithm when the approximate posterior is highly inaccurate. At the first iteration when we have no knowledge of the posterior, we simply revert to sampling from the prior π. However, as our approximation becomes more accurate, so does our approximation of the posterior which allows us to place increasing trust in the surrogate for determining the weighting function used with our pivoted Cholesky procedure. The level of trust is dictated by the value of β. The value chosen should be the largest $\beta_j \in (\beta_{j-1}, 1]$ such that the ratio of the previous posterior, using β_{j-1} and the posterior obtained using the new β are "close". Following [3], we choose β_j such that the coefficient of variation of the densities is equal to a pre-defined threshold τ, i.e.

$$\frac{\mathbb{V}\mathrm{ar}\left[\exp\left(-\Pi_{m_j,g_j}u(\boldsymbol{y})\right)^{\beta-\beta_{j-1}}\pi(\boldsymbol{y})\right]^{\frac{1}{2}}}{\mathbb{E}\left[\exp\left(-\Pi_{m_j,g_j}u(\boldsymbol{y})\right)^{\beta-\beta_{j-1}}\pi(\boldsymbol{y})\right]} = \tau.$$

Similar to [3], we found $\tau = 1$ to be a reasonable choice. To compute the expectation and variance we use 1000 samples drawn uniformly from Γ. The cost of this step is negligible as it only requires the evaluation of the surrogate.

5.3 Example: Rosenbrock function

This subsection demonstrates the benefit of using our transitional Cholesky sampling algorithm to construct surrogates for Bayesian inference. In the following, assume that the observational quantities are modeled by the function $f : \mathbb{R}^2 \to \mathbb{R}^2$, where

$$f_1(\boldsymbol{y}) = 4y_1 - 2, \qquad f_2(\boldsymbol{y}) = 4y_2 - 2 - (4y_1 - 2)^2.$$

In addition assume that the prior distribution $\pi(\boldsymbol{y})$ is a uniform distribution on $[0,1]^2$ and the observational data is $\boldsymbol{d} = (1,0)^T$. Using a Gaussian error model with covariance $\Sigma = \mathrm{diag}(1, 0.01)$ the negative log likelihood is an affine transformation of the two-dimensional Rosenbrock function

$$u(\boldsymbol{y}) = (1 - \hat{y}_1)^2 + 100(\hat{y}_2 - \hat{y}_1^2)^2, \qquad \hat{\boldsymbol{y}} = 4\boldsymbol{y} - 2.$$

Figure 9 compares the accuracy of surrogates constructed using three sampling schemes. The "Prior Weighted Cholesky" label in the legend refers to the weighted Cholesky sampling with the prior as the dominating measure, the "Adaptive Weighted Cholesky" refers to our transitional algorithm and "Halton" refers to simply using a untransformed Halton sequence. In all cases we simultaneously select samples and optimize the correlation lengths of the RBF/Gaussian process kernel.

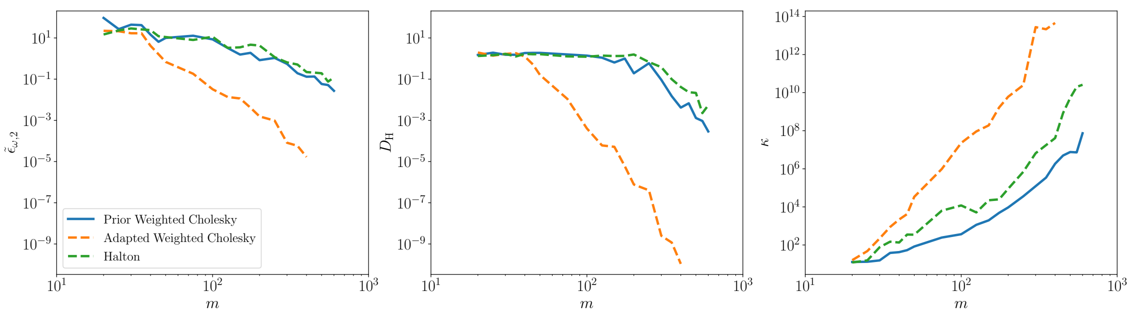

Figure 9: Comparison of sampling strategies used to build surrogates for Bayesian inference. L^2_ω error in the surrogates as a function of the number of samples *(left)*. Squared Hellinger divergence between the true posterior and approximate posterior obtained using a surrogate *(middle)*. Condition number of the kernel training matrix $\mathcal{K}(X_j,X_j)$ at each iteration j *(right)*.

The left plot of Figure 9 depicts the L^2_ω error of each method, computed using 10000 samples drawn from the true posterior distribution using rejection sampling. The middle plot depicts the squared Hellinger divergence

$$D^2_{\rm H}(\pi_{m,g}(\boldsymbol{y}\,|\,\boldsymbol{d}),\pi(\boldsymbol{y}\,|\,\boldsymbol{d}))=1-\int_\Gamma \pi_{m,g}(\boldsymbol{y}\,|\,\boldsymbol{d})\pi(\boldsymbol{y}\,|\,\boldsymbol{d})\mathrm{d}\boldsymbol{y}$$

between the approximate and true posteriors. The integral is evaluated using a high-degree tensor-product Gaussian quadrature rule. By both the L^2_ω and Hellinger divergence metrics, our adaptive weighted Cholesky algorithm is significantly more efficient than the alternatives. Notice that the convergence curves terminate at different sample sizes for each sampling method. The curves terminate when the condition number of the kernel matrix $\mathcal{K}(X_j,X_j)$ constructed using the training data X_j becomes highly ill-conditioned.

The right plot of Figure 9 compares the condition number of each sampling strategy. For a fixed sample size m, the condition number of our approach is largest. Using the prior as the dominating measure improves the condition number, however the error is much worse.

The performance of the adaptive algorithm is dependent on the transitional parameter β_j. In the left of Figure 10, we plot the evolution of β_j at each iteration of the sampling algorithm. For small sample sizes the approximate posterior changes significantly each

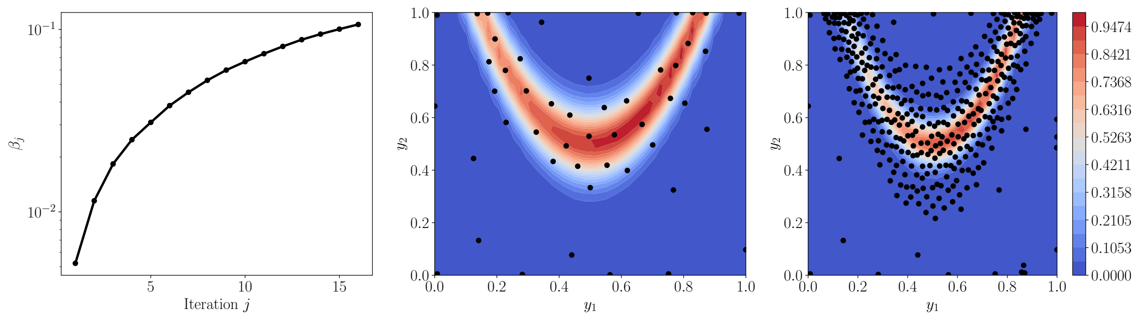

Figure 10: Transitional parameters β_j at each iteration of the adaptive weighted Cholesky algorithm *(left)*. The training set and approximate unnormalized posterior at iteration 6 *(middle)* and at the final iteration *(right)*.

time the training set is enriched. This causes β_j to remain small. However, as the accuracy of the surrogate of u improves, β also increases. The middle plot shows the approximate posterior and training samples after 50 model evaluations and the right plot after 450 evaluations. The approximate posterior is 'closer' to the prior for small sample sizes.

6 Conclusion

In this article, we presented a greedy algorithm for generating samples with the goal of minimizing the weighted L^p-error of kernel-based approximations. Most existing literature focuses on strategies for approximations that minimize the unweighted error. The major contribution of our work to the existing literature is the construction of a computationally simple and efficient algorithm based upon pivoted Cholesky factorization for selecting samples for weighted approximation. We demonstrate through extensive numerical examples that for smooth kernels the accuracy of approximations built using our designs is comparable to the accuracy of those built upon designs from slower existing approaches. For non-tensor product densities our algorithm produced more accurate approximations than the best alternative (IVAR), for moderate to large sampler sizes.

In addition to generating accurate interpolants, the algorithm presented has four useful properties. Firstly, the sample sets are nested. Consequently, they can easily be enriched with additional samples if additional computational resources become available. Secondly, the samples can be generated in batches which allows data to be labeled (evaluated) in parallel. Thirdly, although we focused on the use of the squared exponential kernel, our algorithm can be used in conjunction with any other kernel. And finally the sample sets remain stable even when the length scales of the kernel are changed each time additional samples are added. This is extremely useful because the best length scales of a function are not typically known a priori.

The first part of the article focuses on the weighted approximation of functions when the weight-function is known. The article concludes with an example of how the proposed algorithm can be used for efficiently generating surrogates for the purpose of inferring unknown model parameters from data using Bayesian inference. In this setting the true weight function is the posterior distribution of the model parameters, which is unknown. The algorithm described iteratively builds up an approximation of the posterior, starting from the prior, which is used as a weighting function in the weighted Cholesky sampling procedure. For the example presented, the proposed approach produces approximations of the true posterior which are orders of magnitude more accurate for a pre-specified budget.

7 Acknowledgments

J.D. Jakeman was supported by the Laboratory Directed Research Development (LDRD) program at Sandia National Laboratories and the US Department of Energy, Office of Advanced Scientific Computing Research. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

During his time of working for University of Basel, Switzerland, P. Zaspel was supported by the Swiss National Science foundation through 407540_167186 NFP 75 Big Data.

References

[1] M. Bebendorf. Approximation of boundary element matrices. *Numer. Math.*, 86:565–589 (2000).

[2] X. Chen, E.-J. Park, and D. Xiu. A flexible numerical approach for quantification of epistemic uncertainty. *J. Comput. Phys.*, 240(1):211–224 (2013).

[3] J. Ching and Y.-C. Chen. Transitional Markov chain Monte Carlo method for Bayesian model updating, model class selection, and model averaging. *J. Eng. Mech.*, 133(7):816–832 (2007).

[4] P.R. Conrad, Y.M. Marzouk, N.S. Pillai, and A. Smith. Accelerating asymptotically exact MCMC for computationally intensive models via local approximations. *J. Amer. Statist. Assoc.*, 111(516):1591–1607 (2016).

[5] H. Dette and A. Pepelyshev. Generalized Latin hypercube design for computer experiments. *Technometrics*, 52(4):421–429 (2010).

[6] L. Devroye. *Non-Uniform Random Variate Generation*. Springer, New York, 1986.

[7] G.E. Fasshauer. *Meshfree Approximation Methods with MATLAB.* World Scientific Publishing, River Edge, NJ, 2007.

[8] G.E. Fasshauer. Positive definite kernels: Past, present and future. *Dolomite Res. Notes Approx.*, 4:21–63 (2011).

[9] R.G. Ghanem and P.D. Spanos. *Stochastic finite elements: a spectral approach*. Springer, New York, Inc., 1991.

[10] S.A. Goreinov, N.L. Zamarashkin, and E.E. Tyrtyshnikov. Pseudo-skeleton approximations by matrices of maximal volume. *Math. Notes*, 62(4):515–519 (1997).

[11] A. Gorodetsky and Y. Marzouk. Mercer kernels and integrated variance experimental design. Connections between Gaussian process regression and polynomial approximation. *SIAM/ASA J. Uncertain. Quantif.*, 4(1):796–828 (2016).

[12] A. Gorodetsky and J.D. Jakeman. Gradient-based optimization for regression in the functional tensor-train format. *J. Comput. Phys.* 374: 219–1238 (2018).

[13] H. Harbrecht, M. Peters, and R. Schneider. On the low-rank approximation by the pivoted Cholesky decomposition. *Appl. Numer. Math.*, 62(4):428–440 (2012).

[14] J.D. Jakeman, M. Eldred, and D. Xiu. Numerical approach for quantification of epistemic uncertainty. *J. Comput. Phys.*, 229(12):4648–4663 (2010).

[15] J.D. Jakeman, F. Franzelin, A. Narayan, M. Eldred, and D. Pflüger. Polynomial chaos expansions for dependent random variables. *Comput. Methods Appl. Mech. Engrg.*, 351:643–666 (2019).

[16] M. Kolonko. *Stochastische Simulation. Grundlagen, Algorithmen und Anwendungen*. Vieweg+Teubner, Wiesbaden, 2008.

[17] S.D. Marchi, R. Schaback, and H. Wendland. Near-optimal data-independent point locations for radial basis function interpolation. *Adv. Comp. Math.*, 23:317–330 (2005).

[18] Y. M. Marzouk and D. Xiu. A stochastic collocation approach to Bayesian inference in inverse problems. *Commun. Comput. Phys.* 6(1):826–847 (2009).

[19] S.A. Mattis and B. Wohlmuth. Goal-oriented adaptive surrogate construction for stochastic inversion. *Comput. Methods Appl. Mech. Engrg.*, 339:36–60 (2018).

[20] S. Müller and R. Schaback. A Newton basis for kernel spaces. *J. Approx. Theory*, 161:645–655 (2009).

[21] A.A. Mullur, and A. Messac. Metamodeling using extended radial basis functions: a comparative approach. *Eng. Comput.*, 21:203 (2006).

[22] A. Nataf. Determination des distributions dont les marges sont donnees. *C. R. Acad. Sci. Paris*, 225:42–43 (1962).

[23] F. Nobile, R. Tempone, and C.G. Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.*, 46(5):2309–2345 (2008).

[24] I.V. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317 (2011).

[25] A. O'Hagan and J.F.C. Kingman. Curve fitting and optimal design for prediction. *J. Roy. Statist. Soc. Ser. B*, 40:1–42 (1978).

[26] M. Pazouki and R. Schaback. Bases for kernel-based spaces. *J. Comput. Appl. Math.*, 236:575–588 (2011).

[27] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, 2006.

[28] R. Schaback. Error estimates and condition numbers for radial basis function interpolation. *Adv. Comp. Math.*, 3:251–264 (1995).

[29] R. Schaback and H. Wendland. Kernel techniques: From machine learning to meshless methods. *Acta Numer.*, 15:543–639 (2006).

[30] R. Schaback and J. Werner. Linearly constrained reconstruction of functions by kernels with applications to machine learning *Adv. Comp. Math.*, 25:237 (2006).

[31] A.M. Stuart. Inverse problems: A Bayesian perspective. *Acta Numer.*, 19:451–559 (2010).

[32] A.N. Tikhonov and V.Y. Arsenin. *Solution of Ill-posed Problems.* Winston & Sons, Washington, D.C., 1977.

[33] L.M.M. van den Bos, B. Sanderse, W.A.A.M. Bierbooms, and G.J.W. van Bussel. Bayesian model calibration with interpolating polynomials based on adaptively weighted Leja nodes. *Commun. Comput. Phys.*, 27(1):33–69 (2020).

[34] H. Wendland. *Scattered Data Approximation*. Cambridge University Press, Cambridge, 2004.

[35] C.K.I. Williams. Prediction with Gaussian processes. From linear regression to linear prediction and beyond. In: M.I. Jordan (eds) *Learning in Graphical Models*. NATO ASI Series (Series D: Behavioural and Social Sciences), vol 89. Springer, Dordrecht, 1998.

[36] Z. Wu and R. Schaback. Local error estimates for radial basis function interpolation of scattered data. *IMA J. Numer. Anal.*, 13(1):13–27 (1993).

[37] Z. Wu, D. Wang, P. Okolo, F. Hu, and W. Zhang. Global sensitivity analysis using a Gaussian Radial Basis Function metamodel. *Reliab. Eng. Syst. Safe.*, 154:171–179 (2016).

[38] D. Xiu and J. S. Hesthaven. High-order collocation methods for differential equations with random inputs. *SIAM J. Sci. Comput.*, 27(3):1118–1139 (2005).

[39] D. Xiu and G.E. Karniadakis. The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM J. Sci. Comput.*, 24(2):619–644 (2002).

[40] M. Bebendorf, Y. Maday and B. Stamm, Comparison of Some Reduced Representation Approximations. In: A. Qarteroni, G. Rozza (eds) *Reduced Order Methods for Modeling and Computational Reduction*, 67–100, Springer, Cham, 2014.

[41] L. Bos, S. De Marchi, A. Sommariva, and M. Vianello. Computing Multivariate Fekete and Leja Points by Numerical Linear Algebra. SIAM J. Numer. Anal., 48(5): 1984–1999 (2010).

[42] Y Maday, N. Nguyen, A. Patera, S. Pau. A general multipurpose interpolation procedure: the magic points. *Communications on Pure & Applied Analysis* 8(1):383–404 (2008).

[43] J. Sacks, W.J. Welch, T.J.Mitchell, H.P. Wynn Designs and analysis of computer experiments (with discussion). *Statistical Science*, 4:409-435 (1989).

[44] J.E Oakley, A O'Hagan. Probabilistic sensitivity analysis of complex models: a Bayesian approach. *Journal of the Royal Statistical Society: Series B*, 66(3):751-769 (2004).

[45] T. Qin, Z. Chen, J.D. Jakeman, D. Xiu. A neural network approach for uncertainty quantification for time-dependent problems with random parameters, *International Journal of Uncertainty Quantification*. To appear (2020).

[46] Y. Zhu, N. Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, *Journal of Computational Physics*, 366: 415-447 (2018).

[47] V.R. Joseph, T. Dasgupta, R. Tuo, C.F.J. Wu. Sequential Exploration of Complex Surfaces Using Minimum Energy Designs, *Technometrics*, 57:1, 64-74, (2015).

[48] L. Yan, T. Zhou. Adaptive multi-fidelity polynomial chaos approach to Bayesian inference in inverse problems, *Journal of Computational Physics*, 381:110-128 (2019).

[49] J.D. Jakeman. PyApprox: Probabilistic analysis and approximation of data and simulation. `https://github.com/sandialabs/pyapprox`.

Appendix

In this section we summarize how to update a Cholesky factor of a covariance matrix and use this update to compute the trace involved in the computation of the integrated variance of a Gaussian process. Neither of these linear algebra steps are novel but we believe their use for computing IVAR designs is and so present these steps here for completeness.

Cholesky update

In this section we summarize the UPDATEINVERSECHOLESKYFACTOR algorithm used in Algorithm 4. Let L_{11}^{-1} be the inverse of the current Cholesky factorization of a matrix A_{11}. To compute the inverse of the new Cholesky factorization of the matrix

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix},$$

where $A_{12} \in \mathbb{R}^{r\times k}$, we must first update the Cholesky factor L_{11}. The inverse of the Cholesky factor of A is then given by

$$\begin{bmatrix} L_{11} & 0 \\ L_{12}^T & L_{22} \end{bmatrix}^{-1} = \begin{bmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1}L_{12}^T L_{11}^{-1} & L_{22}^{-1} \end{bmatrix} = \begin{bmatrix} L_{11}^{-1} & 0 \\ C & L_{22}^{-1} \end{bmatrix}$$

where $L_{22} = \text{Chol}\left[A_{22} - L_{12}^T L_{12}\right]$ and $L_{12} = L_{11}^{-1}A_{12}$ If $k = 1$ the Cholesky inverse can be updated in $O(r^2)$ operations needed to compute L_{12}.

Trace update

In this section we summarize the UPDATEIVARTRACE algorithm used in Algorithm 4. Using the partitions of A and L introduced above

$$A^{-1} = \begin{bmatrix} L_{11}^{-T}L_{11}^{-1} + C^TC & C^TL_{22}^{-1} \\ L_{22}^{-T}C & L_{22}^{-T}L_{22}^{-1} \end{bmatrix}.$$

Thus

$$\begin{aligned}
\text{Trace}\left[A^{-1}P\right] &= \text{Trace}\left[\begin{bmatrix} L_{11}^{-T}L_{11}^{-1} + C^TC & C^TL_{22}^{-1} \\ L_{22}^{-T}C & L_{22}^{-T}L_{22}^{-1} \end{bmatrix}\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}\right] \\
&= \sum_i\sum_j \left(\begin{bmatrix} L_{11}^{-T}L_{11}^{-1} + C^TC & C^TL_{22}^{-1} \\ L_{22}^{-T}C & L_{22}^{-T}L_{22}^{-1} \end{bmatrix} \circ \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}\right)_{ij} \\
&= \sum_i\sum_j \left(L_{11}^{-T}L_{11}^{-1} \circ P_{11}\right)_{ij} + \sum_i\sum_j \left(C^TC \circ P_{11}\right)_{ij} + 2\sum_k\sum_l \left(C^TL_{22}^{-1} \circ P_{12}\right)_{kl} \\
&\quad + \sum_m\sum_n \left(L_{22}^{-T}L_{22}^{-1} \circ P_{22}\right)_{mn}
\end{aligned}$$

The first term is expensive to compute but this value can be stored each time the trace is updated and re-used. Again if $k = 1$, then dominant cost of evaluating the trace comes from evaluating the second term which can be computed in $O(r^2)$ operations.