International Journal for Uncertainty Quantification, 1 (4): 321–349 (2011)

BIAS MINIMIZATION IN GAUSSIAN PROCESS SURROGATE MODELING FOR UNCERTAINTY QUANTIFICATION

Vadiraj Hombal & Sankaran Mahadevan*

Vanderbilt University, Nashville, TN 37235

Original Manuscript Submitted: 04/07/2011; Final Draft Received: 09/08/2011

Uncertainty quantification analyses often employ surrogate models as computationally efficient approximations of computer codes simulating the physical phenomena. The accuracy and economy in the construction of surrogate models depends on the quality and quantity of data collected from the computationally expensive system models. Computationally efficient methods for accurate surrogate model training are thus required. This paper develops a novel approach to surrogate model construction based on the hierarchical decomposition of the approximation error. The proposed algorithm employs sparse Gaussian processes on a hierarchical grid to achieve a sparse nonlinear approximation of the underlying function. In contrast to existing methods, which are based on minimizing prediction variance, the proposed approach focuses on model bias and aims to improve the quality of reconstruction represented by the model. The performance of the algorithm is compared to existing methods using several numerical examples. In the examples considered, the proposed method demonstrates significant improvement in the quality of reconstruction for the same sample size.

KEY WORDS: *surrogate models, simulation, Gaussian processes, regression, interpolation, model error*

1. INTRODUCTION

High-fidelity computational models of physical systems play an important role in engineering analyses. Real-world systems are subject to various sources of uncertainty, such as physical variability, data uncertainty, and model errors. Robust analyses of such systems typically require multiple simulations for quantification and integration of these uncertainties [1, 2]. However, the high computational costs involved in executing these simulations prohibit their recurrent use in analysis. In such cases, engineering analyses employ a surrogate model to approximate these simulations due to the low computational costs in evaluating the responses of such approximations [3, 4].

These approximations by surrogate models are not low-fidelity versions of the computer models derived by simplifying the physics of the underlying phenomena. Instead, surrogate model-based approximations aim to reproduce the input-output relationship implemented by an actual simulation code. Toward this end, surrogate models use the response of the simulation code over a small subset of the design space and generalize the information contained in that data over the entire design space, Ω.

The quality of the engineering analysis depends on the accuracy of approximation implemented by the surrogate model. A typical measure of the accuracy of approximation may be written as follows:

$$I(X_s) = \int_{\Omega} \| f(x) - \hat{f}\,(x|Z_s) \| \mathbf{dx} \tag{1}$$

where $f(x)$ is the input-output mapping implemented by the simulation, $Z_s = \{y_s, X_s\} = \{f(\mathbf{x}_i), \mathbf{x}_i\}_{i=1}^{k}$ is the training data and $\hat{f}(x)$ is the approximation to $f(x)$ implemented by the surrogate model. A low value of $I(X_s)$

*Correspond to Sankaran Mahadevan, E-mail: sankaran.mahadevan@vanderbilt.edu

2152–5080/11/$35.00

indicates high-fidelity between the underlying function, f, and its reconstruction, $\hat{f}$, and the quality of reconstruction depends on the choice of Z_s.

The quality of approximation depends on two key factors: (i) selection of the surrogate model form that is able to match the complexity of the underlying function f, and (ii) selection of the most informative or useful training data. The two factors are interlinked: the utility of the data depends on the existence of a model which can extract the information contained in it, and the ability of the model to accurately approximate the underlying code depends on the amount and the quality of data. Thus, in the construction of accurate surrogate models, the two issues—model construction and training data selection—must be considered together.

In previous work, researchers have employed various types of surrogate models such as Gaussian process regression [5, 6], radial basis functions [7], polynomial response surfaces [3, 8], Splines [9] and polynomial chaos expansions [10, 11]. These models are based either on statistical or functional approximation theory and are known to be capable of approximating a wide class of function types [12]. For some applications, studies such as [3, 13] have evaluated the relative performance of various surrogate model types. Although such studies describe the relative performance models for a given application; in general, the choice of the particular surrogate modeling technique employed to approximate a simulation code is, in practice, arbitrary because, functionally, many of the above-mentioned surrogate modeling methods are closely related, even if they have different internal parametrizations [5, 14, 15] and have similar approximation capabilities [9].

Once a surrogate modeling technique is selected, the design of an approximate model then involves (i) selection of training points and (ii) estimation of the internal parameters of the model. Because the mapping $f(x)$ implied by the simulation is not known and is expensive to evaluate, the selection of the optimal training point distribution that minimizes Eq. (1) can neither be determined a priori nor be based on selection criteria based directly on $f(x)$, which requires extensive evaluation. Consequently, sample selection is necessarily sequential and based on selection criteria defined on the intermediate estimates of the underlying function provided by the surrogate models.

The use of Gaussian process (GP) regression in the modeling of deterministic computer codes was introduced by [6]. In that formulation, the underlying function is assumed to represent a particular realization of a GP and the object of modeling is to identify the particular realization. Typical approaches to training data selection belong to the variance reduction class of techniques that borrow from the optimal experiment designs discussed previously [16–18]. Using the Bayesian approach, MacKay and Tong [19, 20] have described entropy based-sample selection criteria for reduction of parameter uncertainty and model discrimination. These criteria are analogous to their counterparts in optimal experiment design. More recently, Guestrin et al. [21, 22] have proposed a mutual information-based criterion for reducing a posteriori prediction uncertainty in GP models and presented an approximate polynomial-time algorithm for the problem. Gramacy and Lee [23] employ Gaussian trees to represent nonstationary processes and employ prediction variance for the design of sample using the space-filling Latin hypercube designs [24, 25]. In contrast to the above approaches, Bichon et al [26] employ an error measure defined on a GP models to approximate the reliability limit state, which allows concentration of the sample distribution around the limit state.

Prediction variance in GP is independent of the actual realization of the underlying function and is instead a function of the parameters of the assumed process covariance. Thus the variance minimizing sample (VMS) designs mentioned above are optimal for a class of functions represented by a given set of GP parameters. The resulting distribution of the training points is uniform throughout the domain and designed to capture the behavior of an entire class of functions (represented by the surrogate model parameters) rather than the particular realization implied by the underlying function.

The density and distribution of training points in GP is determined by the covaraiance parameters, which in turn represent the variation in the underlying function. When the variation in the underlying function is even and spread throughout the domain, then the distribution of training points that is spread uniformly over the domain is neccessary and efficient. However, if the variation in the underlying function is localized to only a small part of the domain, such distributions result in sampling the entire domain based on localized variation. In such cases, VMS represents an inefficient use of a scarce sample budget. In such cases, a sample distribution correlated with the variation in the underlying function is desirable. This paper focuses on the generation of such training point distributions and the development of surrogate models to extract of information contained in them.

In contrast to the selection of training points based on prediction variance (which is solely a property of the approximating model), this paper explores the possibility of training point selection based on prediction bias (which directly describes the relation between the model and the underlying function). Toward this end, first a novel modeling technique based on hierarchical decomposition of a given function in terms of approximation error is proposed. In this method, sparse Gaussian processes are employed in a hierarchical decomposition of the model error—each layer of the model approximates the error in the approximation based on previous layers.

Next, the bias minimizing sampling (BMS) algorithm is developed as an adaptive algorithm for sequential realization of the hierarchical error decomposition model. Using the information contained in the residuals, the BMS algorithm traverses this tree sequentially and adjusts the complexity of the model (in terms of the number of nodes) according to the local variation in the underlying function. Furthermore, by using covariance functions with localized/quasi-localized kernels, and sampling only in the support of the kernels, the algorithm also achieves a nonuniform training point distribution in which the sample density depends on the variation in the underlying function.

Preliminary results have established the efficiency of this method relative to the VMS distributions proposed earlier [16–20]. Specifically, in tests conducted thus far, BMS samples are seen to provide a significantly better quality of approximation than VMS for the same number of samples and, thus, BMS appears to represent a much more efficient procedure for the construction of surrogate models.

The rest of this paper is organized as follows: In Section 2, Gaussian process regression (GPR) is introduced and its predictive distribution is analyzed. The distribution of the training points selected to minimize the prediction variance is also analyzed in Section 2. In Section 3, sparse Gaussian processes are motivated as approximations to the GPR. Sections 2 and 3 form the background for the discussion in Section 4, where the proposed hierarchical error decomposition using sparse Gaussian processes is introduced. In Section 5, the proposed sequential algorithm to achieve the hierarchical decomposition suggested in Section 4 is presented. The BMS algorithm is first compared to VMS in terms of the quality of approximation in Section 5.7. The computational complexities of the two algorithm are compared in Section 5.8, and finally in Section 5.9, the use of both the algorithms for uncertainty quantification is illustrated and the estimates of the surrogate models are compared.

2. BACKGROUND

The following notation will be used $f : \Omega \to \Re$, $\Omega \in \Re^D$ is the underlying function. $\mathbf{x}_T = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t \,|\, \mathbf{x}_i \in \Omega\}$ represents the t training points. The set $\mathcal{D}_T = \{\mathbf{x}_T, \mathbf{y}_T\}$ represents the training data, where $\mathbf{y}_T = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_t \,|\, \mathbf{y}_i = f(\mathbf{x}_i) + \epsilon_i\}$ is the set of measured function values or samples, $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$ is the measurement noise, and $\mathbf{f}_T = \{f_1, f_2, \ldots, f_t \,|\, f_i = f(\mathbf{x}_i)\}$ is the set of underlying function values.

Our objective is to generalize the information contained in data, $\mathcal{D}_T$, so as to infer the value of the underlying function, $\mathbf{f}_P$, at any arbitrary set of p prediction points $\mathbf{x}_P \in \Omega$. In this section, GP- and SGP-based regression techniques are reviewed. Both are probabilistic methods in which inference about $\mathbf{f}_P$ is made by computing the distribution $p(\mathbf{f}_P|\mathcal{D}_T)$. The primary motivation for SGP is developed as a computationally efficient approximation to GP.

2.1 Gaussian Process Regression

In GPR, it is assumed that the underlying function values represent a particular realization of a GP, and the objective then is to identify the particular realization based on the training data.

The GP is a generalization of the multivariate Gaussian distribution and thus may be thought of as a collection (indexed by points in the domain) of multivariate Gaussian random variables. A GP is fully defined with the specification of the mean function, $m(\mathbf{x})$, and the covariance function $k(\mathbf{x}, \mathbf{x}')$. In general, these functions must be selected so as to reflect our assumptions about the underlying function, such as about its stationarity, periodicity, etc. [5, 14, 27]. As is common in literature [5, 6, 28], in this paper it is assumed that the mean function, $m(\cdot) = 0$ and the covariance function is the squared exponential function

$$k(\mathbf{x}_i, \mathbf{x}_j; \theta) = \theta_1 \exp\left[-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_i - x_j)^2}{l_i}\right] \tag{2}$$

Together, $\Theta = \{l, \theta_1, \sigma_n\}$ form the parameters of the covariance function and thus of the GP model. In order to be able to make predictions using the model, these parameters must be inferred from the given data. A common method employed is the maximization of the log marginal likelihood [5]:

$$\log p(\mathbf{y}_T|\mathbf{x}_T;\Theta) = -\frac{1}{2}\mathbf{y}^{\mathrm{T}}(\mathbf{K}_{TT} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_{TT} + \sigma_n^2\mathbf{I}| + \frac{D}{2}\log 2\pi \tag{3}$$

Of special interest are the parameters, $l = \{l_1, l_2, \ldots, l_D\}$ known as length-scale parameters, which correspond to the length of the variation in function values implied by $k(\cdot)$ in each dimension of Ω. Authough a small value of l_i indicates significant variation in the function values in the ith dimension, a large value of l_i indicates that the variablity in the function value is not impacted by changes in the ith dimension.

The GP predictive distribution is given by [5]: $p(\mathbf{f}_P|\mathbf{y}_T, \mathbf{x}_T, \mathbf{x}_P, \Theta) \sim \mathcal{N}(\mathbf{m}, \mathbf{S})$ with

$$\mathbf{m} = \mathbf{K}_{PT}\left(\mathbf{K}_{TT} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}_T \tag{4}$$

$$\mathbf{S} = \mathbf{K}_{PP} - \mathbf{K}_{PT}\left(\mathbf{K}_{TT} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{K}_{TP} \tag{5}$$

where, $\mathbf{K}_{TT} = [k(x_i, x_j)]_{i,j}$ is the $t \times t$ matrix of the covariances between the training points $\mathbf{x}_T$, $\mathbf{K}_{PP}$ is the $p \times p$ matrix of the covariances between the prediction points $\mathbf{x}_P$, and $\mathbf{K}_{PT}$ is the $p \times t$ matrix of covariances between $\mathbf{x}_P$ and $\mathbf{x}_T$ with $\mathbf{K}_{TP}$ as its transpose. The mean of the posterior distribution, $\mathbf{m}$, is taken to be the predicted values of the underlying function at the $\mathbf{x}_P$. This estimate is a function of the (i) measurement uncertainty, (ii) the separation between the training points, (iii) the measured value of the underlying function at the training points, and (iv) separation between the training points and the prediction points. By collecting all the terms that do not depend on the prediction points, $\mathbf{x}_P$, Eq. (4) may be rewritten as follows:

$$\mathbf{m} = \mathbf{K}_{PT}\mathbf{w}_T \tag{6}$$

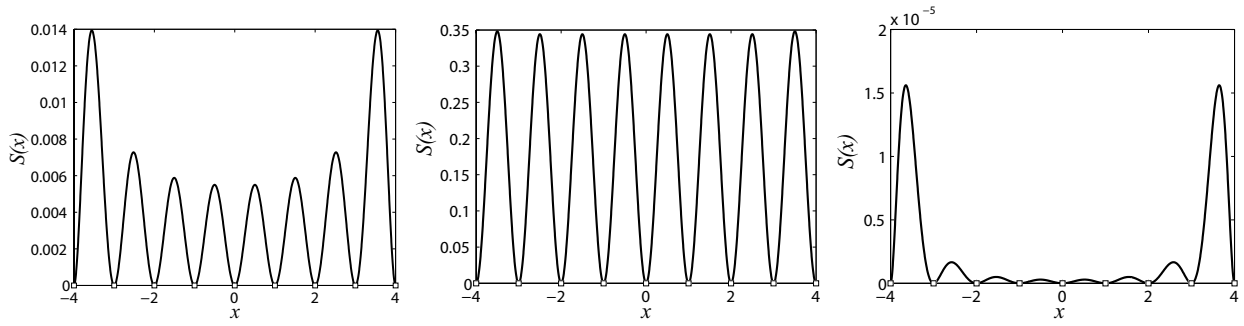
where $\mathbf{w}_T = \left(\mathbf{K}_{TT} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}_T$. The function value predicted by GPR at a test location is thus a weighted sum of the correlation between the test location and each of the training points.

If a computer code is used to evaluate $\mathbf{f}_T$ or $\mathbf{y}_T$, then typically, there is no measurement noise and σ_n may be set to zero. Under such circumstances, the GPR model is interpolative, i.e., $m(\mathbf{x}_T) = \mathbf{f}_T$ and $S(\mathbf{x}_T) = 0$. However, in this paper, σ_n value is set to a very small value, so as to stabilize the inversion of the covariance matrix without affecting the value of the prediction numerically.

2.1.1 GP Prediction Variance and Variance-Based Training Point Selection

The covariance of the predicted values, $\mathbf{S}$, represents the uncertainty in prediction [Eq. (5)]. As with prediction mean, for a given model (Θ fixed), the uncertainty in prediction is a function of the measurement noise, and the separation between the training points and the separation between the prediction points and the training points. This posterior uncertainty is always less than the prior uncertainty $\mathbf{K}_{PP}$; however, unlike the prediction mean, it is independent of the measured values of the function at any of the training points, $\mathbf{y}_T$. When the measurements are taken from an unknown GP, the influence of $\mathbf{y}_T$ is implied indirectly in the estimation of Θ of the unknown GP.

The predicted variance of a 1D GP with $\Theta = \{1, 1, 10^{-6}\}$ and sampled at nine uniformly distributed points in $[-4, 4]$ is shown in Fig. 1(a). The predicted variance is a function of the distance between a prediction point and the training points. The prediction variance has local peaks at the midpoint between two neighboring data points. This shape remains the same throughout the domain; however for points near the boundaries of the domain, the value of the variance increases. This is due to the asymmetry in the number of data points and the effect of length scale. At length scale $l = 1$, prediction variance at points around the center of the domain is influenced by a greater number of data points than the points near the boundary of the domain. As a consequence, in Eq. (5) the correction to prior variance is larger for test points near the center than at those near the boundaries. This effect is exaggerated for a model with a larger length scale, $l = 2$ [see Fig. 1(c)]. For a model with smaller length scale, $l = 0.45$, the effect of neighboring data points is more localized and therefore, the shape of variance curve is uniform throughout the domain



(a) (b) (c)

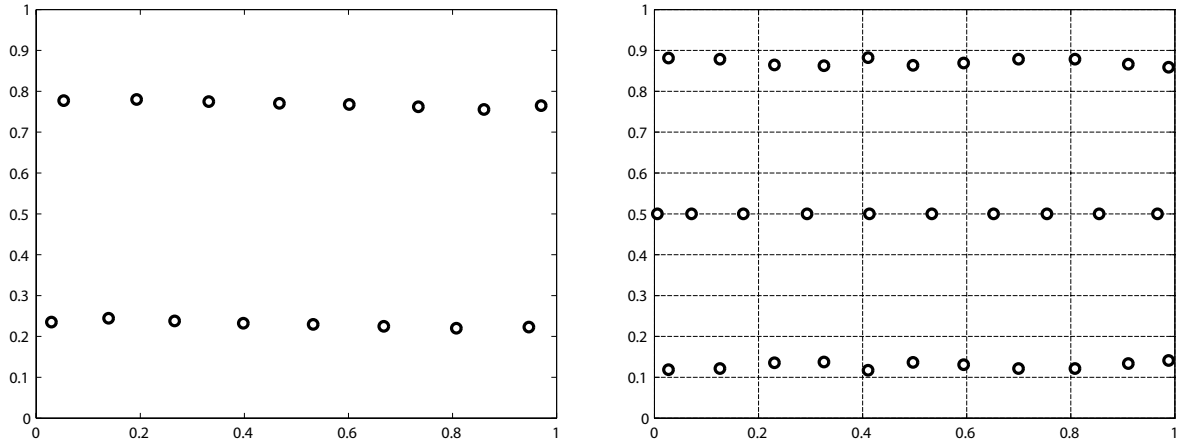
FIG. 1: Predicted variance of Gaussian process regression. For the same data locations (squares), prediction variances of three models, with different values of the length-scale parameters are shown. (a) $\Theta = \{1, 1, 10^{-6}\}$, (b) $\Theta = \{0.45, 1, 10^{-6}\}$, and (c) $\Theta = \{2, 1, 10^{-6}\}$.

[see Fig. 1(b)]. The behavior in higher dimensions is similar to the 1D case, except that the symmetry is with respect to the value of the length scale in each dimension.

An important consequence of this behavior is that for a given Θ, prediction variance may be minimized only by equidistribution of sample points in the domain. Thus, algorithms for VMS designs, tend to generate sample distributions, which are relatively uniform and have a relatively high concentration of sample points along the boundaries of the domain. The behavior in higher dimensions is similar to the 1D case, except that the equidistribution is with respect to the value of the length scale in each dimension.

A simple numerical experiment illustrates this point. Suppose, $\Omega = [0, 1]^2$. We consider a uniform grid of $51 \times 51 = 2601$ prediction points, $\mathbf{X}_P \in \Omega$. We assume that the underlying function is a GP with known parameters, Θ, and seek the locations of the n training points, $\mathbf{X}_T \in \Omega$, that minimize the total prediction variance, $T = \text{trace}(\mathbf{S})$, where $\mathbf{S}$ is a 51×51 covariance matrix defined according to Eq. (5).

We first consider the anisotropic case $\Theta = \{0.1, 0.5, 1, 1e-6\}$. The sample distribution for $n = 16$ and $n = 32$ are shown in Figs. 2(a) and 2(b), respectively. For $n = 16$, the sample distribution is aligned in two rows. In each row, the average separation between neighboring points is 0.1311 and the average separation between the two rows is 0.5383. The total prediction variance is 156.312. Doubling the number of training points to $n = 32$, reduces the total



(a) (b)

FIG. 2: Prediction variance minimizing sample distribution: 2D anisotropic GP with known $\Theta = \{0.1, 0.5, 1, 10^{-6}\}$. (a) $n = 16$ and (b) $n = 32$.

prediction variance to 13.8549 and distributes the samples in three rows. In each row, the average separation between neighboring points is 0.0997 and the average separation between the rows is 0.3714.

Figure 3(a) shows the sample distribution for the isotropic case. The Voronoi tesselation of the domain with the sample locations as Voronoi centers is also shown in Fig. 3(a) to reinforce the equipartition of the domain by the sample locations. Cells in the tesselation are approximately congruent to each other.

The above examples illustrate that the optimal sample distribution that minimizes the total prediction variance over the domain is such that the average distance of the training points to the prediction points is minimized. Furthermore, when the prediction points are uniformly distributed over the domain, this results in equidistribution (with respect to the distance metric defined in the process covariance) of the training points in the domain.

In the modeling of simulation codes, model parameters are not known a priori and must be estimated based on the training data. For such cases, starting with an initial sample distribution, a sequentialized version of the above algorithm may be considered [28, 29]. In each iteration, the model parameters are estimated using existing training data. In turn, a new training point is added at the location of the highest prediction variance indicated by the model. In subsequent sections of this paper, this sequentialized algorithm is referred to as VMS and serves as the benchmark for comparisons.

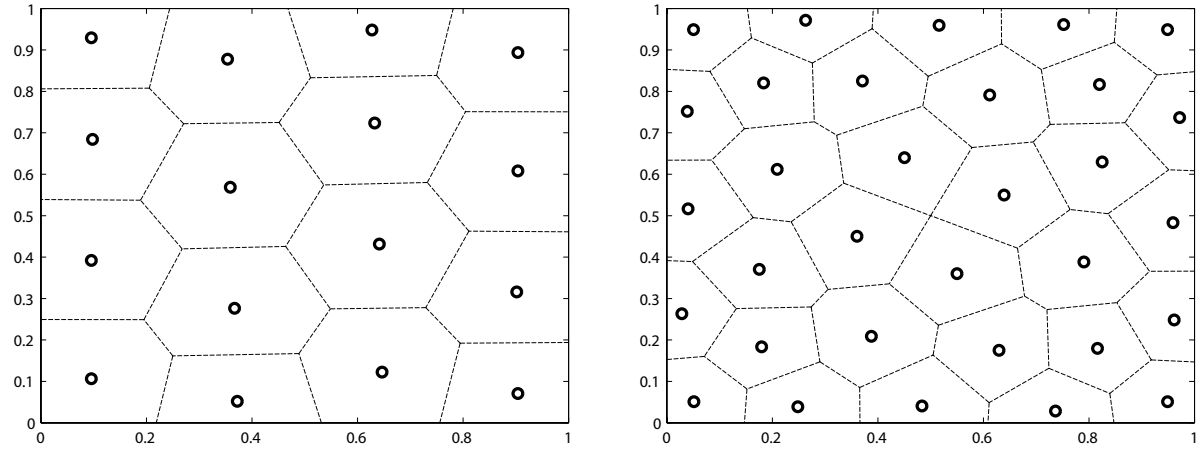
3. SPARSE GAUSSIAN PROCESS REGRESSION

Driven by computational concerns, several approximations to GPR models known as sparse Gaussian process regression (SGPR) models, have been developed [30–34]. Quiñonero-Candela and Rasmussen [35] has shown that the common idea behind these approximations is to induce a relation between the prediction variables $\mathbf{f}_P$ and the training variables, $\mathbf{f}_T$, through another subset of inducing variables, $\mathbf{f}_U$, defined at locations, $\mathbf{x}_U \in \Omega$.

The inducing variables, $\mathbf{f}_U$, follow the same distribution as $\mathbf{f}_T$ and $\mathbf{f}_P$, i.e., $\mathbf{f}_U \sim \mathcal{N}(0, \mathbf{K}_{UU})$, and $\mathrm{P}(\mathbf{f}_T, \mathbf{f}_P)$ may be computed by marginalizing the joint prior of all three variable sets:

$$\mathrm{P}(\mathbf{f}_T, \mathbf{f}_P) = \int \mathrm{P}(\mathbf{f}_T, \mathbf{f}_P, \mathbf{f}_U) d\mathbf{f}_U \tag{7}$$

As a consequence of marginalization, the actual values of the inducing variables, $\mathbf{f}_U$, do not influence the inference. However, as discussed subsequently, the locations of the inducing variables, $\mathbf{x}_U$, are critical to inducing a relationship between the prediction and training variables. Following the terminology of Quiñonero-Candela and Rasmussen [35], these locations are called *inducing inputs* in this paper.



(a) (b)

FIG. 3: Prediction variance minimizing sample distribution: 2D Isotropic GP with known $\Theta = \{0.25, 0.25, 1, 10^{-6}\}$. (a) $n = 16$ and (b) $n = 32$.

Sparse approximations to GP are based on two key assumptions. The first assumption, which leads to approximation of the exact joint prior above, is the conditional independence between f_T and f_P, given f_U. Equation (7) may then be written as

$$\mathrm{P}(\mathbf{f}_T, \mathbf{f}_P) \simeq \int \mathrm{P}(\mathbf{f}_T|\mathbf{f}_U)\mathrm{P}(\mathbf{f}_P|\mathbf{f}_U)\mathrm{P}(\mathbf{f}_U)d\mathbf{f}_U \tag{8}$$

Under this assumption, interaction between $\mathbf{f}_T$ and $\mathbf{f}_P$ is induced through $\mathbf{f}_U$. This is in contrast to GPR, where the relation between $\mathbf{f}_P$ and $\mathbf{f}_T$ is direct because the joint posterior $p(\mathbf{f}_T, \mathbf{f}_P|\mathbf{y}_T)$ is described in terms of the exact joint prior $p(\mathbf{f}_T, \mathbf{f}_P)$.

In addition, formulation of SGP involves assumptions regarding the approximations of conditionals, $\mathbf{f}_P|\mathbf{f}_U$ and $\mathbf{f}_T|\mathbf{f}_U$. Several different approximations have been suggested, leading to a different joint prior between $\mathbf{f}_T$ and $\mathbf{f}_P$ and, consequently ,to different approximate posterior distributions [30, 32, 35]. In this paper, we use the fully independent training conditional (FITC) approximation described in [34].

As a reference, with $\mathbf{Q}_{A,B} \triangleq \mathbf{K}_{AU}\mathbf{K}_{UU}^{-1}\mathbf{K}_{UB}$, prediction based on this approximation is provided as follows:

$$\mathbf{m} = \mathbf{Q}_{PT}\left(\mathbf{Q}_{TT} + \mathrm{diag}[\mathbf{K}_{TT} - \mathbf{Q}_{TT} + \sigma_n^2\mathbf{I}]\right)^{-1}\mathbf{y}_T \tag{9}$$

$$\mathbf{S} = \mathbf{K}_{PP} - \mathbf{Q}_{PT}\left(\mathbf{Q}_{TT} + \mathrm{diag}[\mathbf{K}_{TT} - \mathbf{Q}_{TT} + \sigma_n^2\mathbf{I}]\right)^{-1}\mathbf{Q}_{TP} \tag{10}$$

where $\mathrm{diag}(\cdot)$ is a diagonal matrix containing only the diagonal elements of the argument matrix.

By collecting all the terms that do not contain the prediction points, $\mathbf{x}_P$, the mean predicted value may be written in the form

$$\mathbf{m} = \mathbf{K}_{PU}\mathbf{w}_U \tag{11}$$

where

$$\mathbf{w}_U = \mathbf{K}_{UU}^{-1}\mathbf{K}_{UT}\left(\mathbf{Q}_{TT} + \mathrm{diag}[\mathbf{K}_{TT} - \mathbf{Q}_{TT} + \sigma_n^2\mathbf{I}]\right)^{-1}\mathbf{y}_T \tag{12}$$

Thus, unlike in GPR [Eq (6)], in SGP, predicted values are a weighted sum of the correlation between the prediction location(s) and the inducing inputs. The weighting itself is a function of the correlation between the inducing inputs, the training locations and the measurement noise. As a consequence of inducing the relation between $\mathbf{f}_P$ and $\mathbf{f}_T$ through variables $\mathbf{f}_U$, the SGP model is, in general, not interpolative and is rather an approximation.

As in GP, the prediction variance in SGP is also a function of the distance; however, the variance is minimum at the location of the inducing inputs, rather than the training points.

Training in SGP involves estimation of $\mathbf{x}_U$, apart from Θ and may, in general, involve selecting these parameters via maximization of the marginal likelihood

$$\log[p(\mathbf{y}_T|\mathbf{x}_U)] = -\frac{1}{2}\log|\mathbf{Q}_{TT} + \Lambda| - \frac{1}{2}\mathbf{y}^{\mathrm{T}}\left(\mathbf{Q}_{TT} + \Lambda\right)^{-1}\mathbf{y} - \frac{n}{2}\log 2\pi \tag{13}$$

where $\Lambda = \mathrm{diag}[\mathbf{K}_{TT} - \mathbf{Q}_{TT}] + \sigma_n^2\mathbf{I}$. In this paper, for reasons that become apparent in subsequent discussion, the $\mathbf{x}_U$ are specified independently of the parameters Θ.

The following notation will be used henceforth: $\mathcal{GP}\left(\mathbf{x}; \mathbf{x}_T, \Theta\right)$ refers to a GP model in which parameters Θ are learned based on information in training data at $\mathbf{x}_T$. The corresponding SGP model based on inducing points, $\mathbf{x}_U$ is represented as $\mathcal{SGP}\left(\mathbf{x}; \mathbf{x}_T, \mathbf{x}_U, \Theta\right)$.

4. HIERARCHICAL ERROR DECOMPOSITION USING SGP

The principal objective of this paper is to develop an error-based sequential modeling and training point selection algorithm for an a priori unknown function. The proposed solution, BMS (presented in Section 4), relies on hierarchical decomposition of the approximation error space. Toward this end, in this section we begin by introducing the hierarchical Gaussian processes (HGPs), which generate a representation of a given function through a hierarchical decomposition of the approximation error using a multilayered hierarchy of sparse Gaussian processes. Subsequently, in Section 4.1, such hierarchical decompositions are sequentialized and employed toward the reconstruction of an unknown function. In this section, the proposed hierarchical decomposition of the approximation error space, synthesis analysis of HGP models, and their variation localizing properties are discussed.

4.1 Hierarchical Decomposition

The first step in the development of HGP models is to consider an M-level hierarchical decomposition of an a priori known function $f : \Omega \rightarrow \Re$, $\Omega \in \Re^D$, which is represented by its dense sampling: $\{x_T, y_T = f(x_T)\}$. With $k = \{1, 2, \ldots, M\}$ as an index into the M-level hierarchy, such a decomposition may be represented as

$$\begin{array}{c|lcl} & e_1 & = & f \\ k=1 & e_1 & = & \hat{e}_1 + e_2 \\ k=2 & e_2 & = & \hat{e}_2 + e_3 \\ \vdots & & \vdots & \\ k=M & e_M & = & \hat{e}_M + e_{M+1} \end{array} \tag{14}$$

The first layer of the model approximates $e_1 = f$ with $\hat{e}_1$ and $e_2 = e_1 - \hat{e}_1$ is the corresponding approximation error. Similarly, in subsequent layers, $\hat{e}_k$ is the kth layer approximation to the error, e_k, in the previous layer, and $e_{k+1} = e_k - \hat{e}_k$ is the corresponding approximation error of the current layer.

At the kth level of hierarchy, the error at point $\mathbf{x} \in \Omega$, $e_k(\mathbf{x})$, is approximated by a SGP network of N_k inducing inputs, $\mathbf{x}_U^k$, which are centered at locations $C_k = \{\mathbf{c}_{k,j} \in \Omega, j = 1, 2, \ldots N_k\}$ with weights $\mathbf{w}_U^k$, with parameters Θ_k, estimated using data $\{x_T, y_T = f(x_T)\}$:

$$\hat{e}_k(\mathbf{x}) = \mathcal{SGP}(\mathbf{x}; \mathbf{x}_T, \mathbf{x}_U^k, \Theta_k) = \mathbf{K}_{PU}^k \mathbf{w}_U^k \tag{15}$$

where K_{PU}^k is the correlation matrix with N_k columns, which span the subspace represented by the kth layer approximation. Note that the parameters of each layer are based on the estimates of approximation error computed at the location of the training points, x_T, and is therefore same for all the layers.

From Eq. (14), it follows that

$$f = \sum_{k=1}^{M} \hat{e}_k + e_{M+1} \triangleq \hat{f}_M + e_{M+1} \tag{16}$$

where $\hat{f}_M$ is defined as the M-level approximation to function f, and the corresponding error due to M-level approximation, $e_{M+1} = f - \hat{f}_M$. In general, e_k is the approximation error due to the $(k-1)$-level hierarchical approximation of f:

$$e_k = f - \hat{f}_{k-1} = f - \sum_{j=1}^{k-1} \hat{e}_j \tag{17}$$

Thus, each layer k approximates the error space due to the $(k-1)$-th level approximation of the function f. Furthermore, using the kth layer approximation model in Eq. (15), the M-level approximation may be written as

$$\hat{f}_M(\mathbf{x}) = \sum_{k=1}^{M} \mathcal{SGP}(\mathbf{x}; \mathbf{e}_k, \mathbf{x}_T, \mathbf{x}_U^k, \Theta_k) = \sum_{k=1}^{M} \mathbf{K}_{PU}^k \mathbf{w}_U^k \tag{18}$$

4.2 HGP Parameters

Equation (18) describes the reconstruction of the given function in terms of the set of inducing inputs, $C = \{C_k\}_{k=1}^M$, and the set of covariance function parameters, $\mathbf{\Theta} = [\Theta_1 \mid \Theta_2 \mid \Theta_2 \cdots \Theta_M]^T$. In what follows, the elements of C are referred to as the structural parameters and the elements of $\mathbf{\Theta}$ are referred to as the approximation parameters. The quality of approximation represented by the reconstruction in Eq. (18) depends on the choice of the approximation and structural parameters.

In general, given the number of inducing inputs, N_k, the optimal structural and approximation parameters may be simultaneously estimated as in [34]. However, as the number of inducing inputs increases, the complexity of the parameter identification problem increases because this optimization is with respect to $(N_k + 1)D + 1$ variables.

Instead, in this work, the structural parameters are set according to an ordering imposed by a hierarchical analysis grid defined on the problem domain, Ω. Then, given a set of structural parameters, the optimal $D+1$ approximation parameters are subsequently identified. Specifically, the kth layer approximation parameters, Θ_k are chosen so as to minimize

$$J[\boldsymbol{\Theta}_\mathbf{k}] = \sum_{\mathbf{x}\in\mathbf{X}_s} \Big[e_k(\mathbf{x}) - \hat{e}_k(\mathbf{x})\Big]^2 = \Big(\mathbf{e}_k - \mathbf{K}_{PU}^k \mathbf{w}_U^k\Big)^{\mathrm{T}} \Big(\mathbf{e}_k - \mathbf{K}_{PU}^k \mathbf{w}_U^k\Big) \tag{19}$$

where $\mathbf{e}_k = [e_k(\mathbf{x}_1), e_k(\mathbf{x}_2), \ldots, e_k(\mathbf{x}_N)]^{\mathrm{T}} = e_k(X_s)$ represents a sampling of e_k and $\mathbf{w}_U^k$ is defined as in Eq. (12).

This reduction in computational complexity comes at the price of having to maintain the analysis grid and the reduced approximation power due to the discretization of the allowable structural parameter values. As is discussed subsequently, this setting of structural parameters according to the geometry of the problem domain and independent of the parameters yields a localization of the variation in the underlying function.

4.3 Analysis Grid

The analysis grid employed in this paper is a ternary tree in which each layer is a dyadic partition of the previous layer. The intersections of such partitions form the nodes of the tree. The association to HGP is established by an ordered assignment of each layer of the tree to a corresponding layer in the HGP. The number and position of nodes at a given layer in the tree correspondingly determine the number of inducing inputs and their locations in each HGP layer. This tree achieves binary partition of the domain and thus allows a given region in the domain to be resolved in terms of multiple inducing inputs of possibly decreasing length scale and thus represents a richer set of inducing inputs than is provided by a simpler structure such as the binary tree.

An example of an analysis grid defined over a domain $\Omega = [a, b] \in \Re$ is shown in Fig. 4. The construction of the grid begins with the setting of N_1, the number of nodes in layer 1, including two on the domain boundary. These N_1 nodes $C_1 = \{c_{1,j}\}_{j=1}^{N_1}$ are separated from each other at a resolution $\rho_1 = (b-a)/(N_1-1)$. Thus, the jth node in C_1, $c_{1,j}$, is situated at location $(j-1) * \rho_1 + a$.

Nodes in subsequent layers are generated according to a refinement procedure that allows for a kth-level node $c_{k,j}$ to be expanded in terms of its child nodes, $C_{k+1}^{(k,j)}$ in level $k+1$. Specifically,

$$C_{k+1}^{(k,j)} = \begin{cases} \{c_{k+1,2j-1}, c_{k+1,2j}\} & \text{if } j = 1 \\ \{c_{k+1,2j-2}, c_{k+1,2j-1}, c_{k+1,2j}\} & \text{if } 1 < j < 2^k + 1 \\ \{c_{k+1,2j-2}, c_{k+1,2j-1}\} & \text{if } j = 2^k + 1 \end{cases} \tag{20}$$

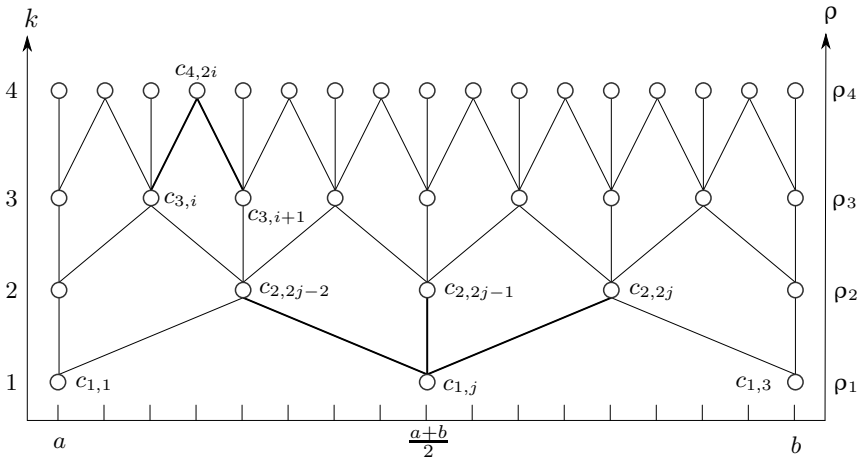


FIG. 4: Analysis grid.

where

$$\begin{aligned} c_{k+1,2j-2} &= \frac{1}{2}(c_{k,j} + c_{k,j-1}) \quad \text{(left child)} \\ c_{k+1,2j-1} &= c_{k,j} \quad \text{(middle child)} \\ c_{k+1,2j} &= \frac{1}{2}(c_{k,j+1} + c_{k,j}) \quad \text{(right child)} \end{aligned} \tag{21}$$

and, $\rho_{k+1} = (1/2)\rho_k$.

Thus, the left and right child nodes are of the type $c_{k,2i}$ ($i = 1, 2, \ldots, 2^{k-1}, k > 1$), and have two parents: $c_{k-1,i}$ and $c_{k-1,i+1}$, while the middle child nodes $c_{k,2j-1}$, $k > 1$, have only one parent, $c_{k-1,j}$.

Finally, the $(k+1)$th layer nodes are given by $C_{k+1} = \bigcup_{j=1}^{N_k} C_{k+1}^{(k,j)}$. In general, each level k consists of $N_k = 2^k + 1$ centers, $C_k = \{c_{k,\cdot}\}$, including two on the boundary of the domain. Furthermore, each node in the kth layer is separated from its neighbor by $\rho_k = \rho_1/2^{k-1}$ and located at $c_{k,j} = a + \rho_k(j-1)$, $j = 1, 2, \ldots, 2^k + 1$. Also, because each higher layer is obtained through partition of the previous layer, $C_1 \subseteq C_2 \subseteq \cdots C_{M-1} \subseteq C_M$.

As shown in Fig. 4, level 1 corresponds to the lowest layer with each node separated by $\rho_1 = (b-a)/2$ and at the highest level, there are $2^M + 1$ uniformly spaced nodes separated by $\rho_k = (b-a)/2^{M-1}$. Thus, ρ_k progressively decreases as k increases. This ordering on ρ_k as a function of level of hierarchy, facilitates a coarse-to-fine decomposition of the function such that coarse scale trends in f are modeled in the initial layers and fine scale features are modeled in successive layers of the hierarchical model.

Extensions to higher dimensions follow directly. If $f : \Re^D \to \Re$ is a function defined in a domain of interest $[a, b]^D$, then starting with $N_1 = 3$ in each dimension, each layer of the analysis grid contains $N_k = (2^k + 1)^D$ centers, $C_k = \{c_{k,\cdot}\}$. Furthermore, each node in the kth layer is separated from its neighbor by $\rho_k = (b-a)/2^k$. Note that it is not necessary to start with $N_1 = 3$. One could start with any arbitrary value for N_1 and proceed with dyadic partition of scale and domain as stated above.

4.4 HGP Illustration

We consider a $M = 6$ level HGP decomposition of the following test example:

$$f(x) = -1 + (1 - x + 19x^2)e^{-x^2} + 2\,\sin(6\pi x)e^{-(x-0.73)^2} \tag{22}$$

Figure 5 illustrates the $M = 6$ HGP decomposition. In Fig. 5, the position of the N_k centers C_k are indicated by squares, while the locations of the measurements are shown as dots.

At each layer, given measurements, $S_k = \{\{f(X_k), X_k\}, X_k = C_k\}$, the HGP analysis consists of estimation of the approximation parameters, Θ_k, of the kth layer according to Eq. (19). Thus, the weights corresponding to the N_k inducing inputs, $w_{k,j}$, are estimated using the N_k residues, $\mathbf{e}_k$ available at X_k locations.

As the model proceeds to higher scales, the approximation errors constitute only variations at finer scale. Thus, the algorithm captures the trends and coarse scale features in $f(x)$ in the initial layers, while the finer scale variations are captured in the higher layers. In addition, progression to higher layers leades to systematic reduction in approximation error. Table 1 lists the mean-square approximation error after each layer of approximation is added to the model.

4.5 Weight Matrix

Analysis of the weights [see Eq. (18)] corresponding to each inducing input shows the variation sensitive modeling implemented by HGP. Toward this end, we begin by considering a visualization of the weights according to the geometry of the approximation.

TABLE 1: HGP reconstruction of test function

Layer	1	2	3	4	5	6
MSE	10.035	8.4275	1.7372	0.3242	0.3559	2.6383e-6

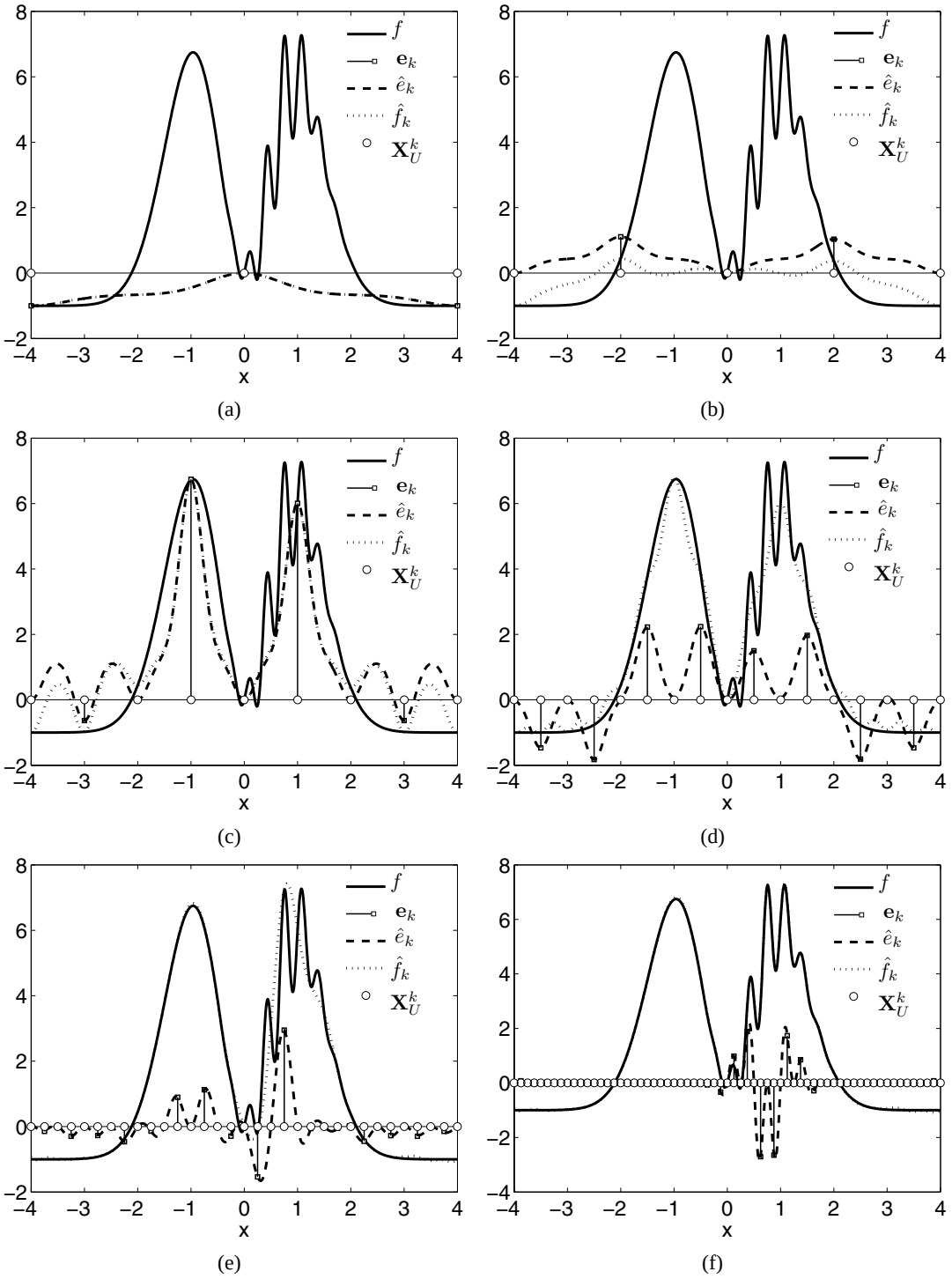


FIG. 5: HGP Illustration: f is the underlying function and $\hat{f}_k$ the kth level approximation for f. The error $e_k = f - \hat{f}_k$ is available through residuals $\mathbf{e}_k$, indicated by vertical lines. $\hat{e}_k$ is a SGP approximation to e_k with inducing inputs $\mathbf{X}_U^k$. (a) $k = 1$, $N_1 = 3$, (b) $k = 2$, $N_2 = 5$, (c) $k = 3$, $N_3 = 9$, (d) $k = 4$, $N_4 = 17$, (e) $k = 5$, $N_5 = 33$, and (f) $k = 6$, $N_6 = 65$.

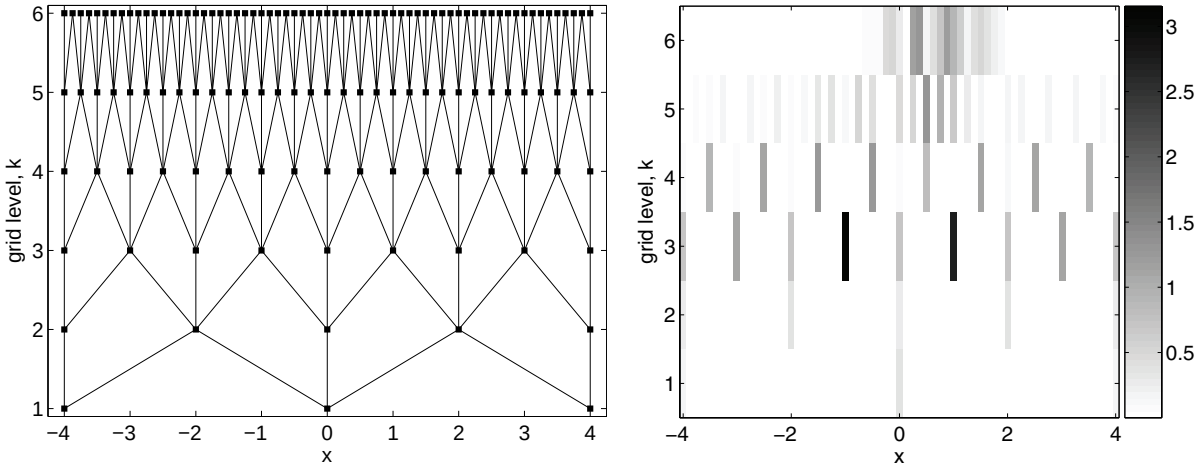
The HGP coarse-to-fine decomposition of a function employs a full grid of $(2^{M+1}+M-2)$ inducing inputs toward an M-level decomposition of the function. The weights corresponding to each inducing input in the hierarchical model can be collected in a weight matrix (WM) of size $M \times 2^{M+1}$ with nonzero elements only at positions corresponding to the position of the centers of the kth layer. An example is shown in the Fig. 6.

Such a weight matrix provides a visualization of the relative significance of the inducing inputs within the structure of the approximation. Figure 7(a) shows the inducing input tree for the six-level HGP decomposition of the test function (22). The corresponding weight matrix is shown in Fig. 7(b), where each rectangle is shaded according to the magnitude of the weights, $|w_{k,j}|$. As a consequence of using the squared exponential covariance functions, which have quasi-localized support, a high value of the weight $w_{k,j}$ indicates high correlation between the $(k-1)$th-layer approximation error, e_k, and the inducing input centered at $c_{k,j}$.

The correlation between the weights and the approximation error e_k is further extended to the underlying function itself. In Fig. 7, while lower layers account for coarse scale variations in the function, concentration of level six bases centered in the interval $[0, 2]$ correspond to fine scale variations in the function in the interval $[0, 2]$. This illustrates a localization of the function f. However, because HGP hierarchy is based on the decomposition of the error space, the layer at which fine scale variations of a function are characterized depends on the quality of the approximation in the previous layers rather than the function itself. Thus, the HGP does not directly characterize the variation scale in inherent in the function. Instead, the HGP scale characterization is apparent and depends on the approximation in the lower layers.

k																	m_k
4	$w_{4,1}$	$w_{4,2}$	$w_{4,3}$	$w_{4,4}$	$w_{4,5}$	$w_{4,6}$	$w_{4,7}$	$w_{4,8}$	$w_{4,9}$	$w_{4,10}$	$w_{4,11}$	$w_{4,12}$	$w_{4,13}$	$w_{4,14}$	$w_{4,15}$	$w_{4,16}$	$w_{4,17}$	17
3	$w_{3,1}$		$w_{3,2}$		$w_{3,3}$		$w_{3,4}$		$w_{3,5}$		$w_{3,6}$		$w_{3,7}$		$w_{3,8}$		$w_{3,9}$	9
2	$w_{2,1}$				$w_{2,2}$				$w_{2,3}$				$w_{2,4}$				$w_{2,5}$	5
1	$w_{1,1}$								$w_{1,2}$								$w_{1,3}$	3
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	

FIG. 6: Weight matrix.



(a) (b)

FIG. 7: HGP decomposition of the test example. (a) HGP model tree and (b) weight matrix, $\mathbf{W}$.

4.6 Variation Sensitive Modeling

A thresholding scheme similar to wavelets may be employed in HGP to yield an efficient representation of the function with a drastically reduced inducing input set. Specifically, given an M-level HGP decomposition of a function f, a compressed representation of f may be generated by a thresholding operation [36]:

$$\tilde{f}_M = \sum_{k=1}^{M} \sum_{j=1}^{N_k} \mathbf{K}_{PU} T(w_{k,j}) \tag{23}$$

where $\theta_T(\cdot)$ is the thresholding function

$$T(x) = \begin{cases} x & \text{if } |x| \geq T \\ 0 & \text{if } |x| < T \end{cases} \tag{24}$$

Figure 8 shows the result of a thresholded reconstruction of the test function. The thresholded weight matrix, $\mathbf{W}_t$, is shown in Fig. 8(b). Figure 8(d) identifies the nodes that were excluded from the reduced representation. Although a HGP construction has $2^{M+1}+(M-2) = 132$ inducing inputs, the thresholded representation uses only $R = 76$ nodes

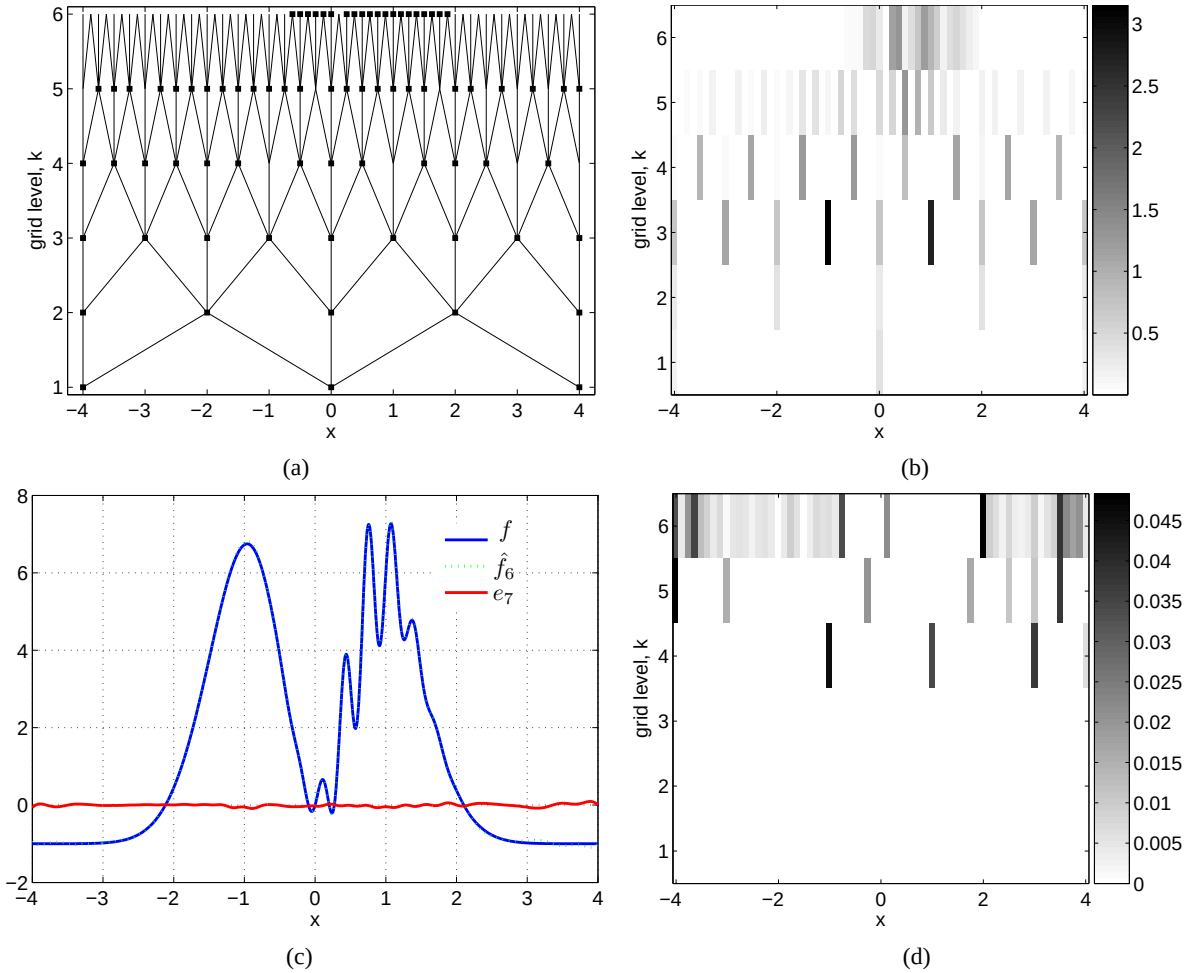


FIG. 8: HGP compression. (a) Reduced model tree, (b) thresholded weight matrix, $\mathbf{W}_t$, (c) reconstruction, and (d) $\mathbf{W} - \mathbf{W}_t$.

to generate a representation of similar quality. The reduced model tree is shown in Fig. 8(a), and the corresponding reconstruction based on the thresholded tree is shown in Fig. 8(d).

Although the MSE in HGP reconstruction is 2.6383×10^{-6}, the MSE due to reconstruction from reduced node set is 0.0011. As a measure of the significance of the retained nodes, $\sum|\mathbf{W}_t|/\sum|\mathbf{W}| = 0.98$. In the addition, as seen in Fig. 8(d), a significant number of the nodes belonging to the $k = 6$, that were pruned out are located outside the approximate interval $[-1, 2]$, which is the interval in which the function $f(x)$ displays its highest variation. On the other hand, Fig. 8(b) shows the concentration of higher layer nodes, with significant weights, within this interval. This example suggests a variation sensitive model, in which the density and the scale of inducing inputs must correspond to the localization of the features contained in the underlying function.

5. BIAS MINIMIZING SAMPLING

The HGP algorithm presented in Section 4 allows for the construction of a M-level hierarchical representation of an a priori known function. This representation, which consists of $\left(2^{M+1} + M - 2\right)$ inducing inputs, may then be compressed to produce a sparse representation by retaining only those inducing inputs with significant contributions to the overall representation. The resulting compressed model, apart from providing an efficient representation in terms of the approximation error, provides a variation-sensitive model in which the location of the inducing inputs correspond to the location of variation in the underlying function.

However, in the construction of surrogate models for computer simulation, the underlying function is a priori unknown. The BMS method proposed in this section implements a sequential algorithm, which traverses the analysis grid sequentially and arrives at a sparse representation of an a priori unknown function through discovery of significant inputs in the grid. Furthermore, by linking sampling resolution and localization to nodes on the grid, BMS achieves sequential sensitive modeling and, as a consequence, sequential variation sensitive sample distribution.

As in HGP, BMS implements an hierarchical error decomposition. Furthermore, in BMS the nodes of the analysis grid correspond to both the location of the inducing inputs and the location of the training points. The algorithm starts with an initial coarse sample distribution and an initial model consisting of inputs at a coarse scale. It then implements an iterative feedback procedure that systematically selects nodes on the grid for sample refinement and subsequent model update.

The BMS algorithm implements an adaptive procedure for the refinement of an existing model and the corresponding sample distribution. In each iteration, the selection of a node for refinement and modeling is based on criteria that use information contained in the residuals, which represent a sampling of the true approximation error in the current function estimate.

5.1 Steps of the BMS Algorithm

A procedural outline of the algorithm is presented as follows:

1. select node from the set of allowable candidate nodes.
2. sample function at child nodes of the selected node.
3. add the input of the selected node to the model.
4. add children of the modeled node to the set of allowable candidate nodes.
5. repeat till the maximum number of samples allowed are taken.

Procedurally, steps 1–3 of the BMS algorithm outline implement a select-refine-model cycle. The trajectory of the sequential BMS algorithm through the analysis grid is governed by two key ideas:

1. In each iteration, the set of allowable candidate nodes, C_A, consists of all the unmodeled children of previously modeled nodes. This set represents the inducing inputs, that are allowed to enter the model in the next iteration. From this set of allowable candidate nodes, a subset of nodes is selected, according to the selection criteria (Section 5.4) for subsequent modeling.

2. A selected node is refined by sampling at its child node locations before it is modeled.

These ideas are illustrated in Fig. 9. In Fig. 9(a), nodes C_1 correspond to previously modeled that which correspond to $\hat{f}_1$. Nodes C_2, which form the set of all unmodeled children of the previously modeled nodes, form the set of allowable candidate nodes C_A, for the next iteration.

Suppose node $c_{2,2}$ is selected for modeling in the next iteration (i.e., an inducing input centered at the location of the node is to be added to the model). The selected node $C_i^* = c_{2,2}$ is first refined by taking measurements at the locations of the child nodes $C_R = \{c_{3,2}, c_{3,3}, c_{3,4}\}$, and is then introduced into the model. Modeling of the selected node $c_{2,2}$ introduces the child nodes into the set of allowable nodes for the next iteration. Thus, as seen in Fig. 9(b), the set of allowable candidate nodes is $C_A = (C_A \setminus C_i^*) \cup C_R = \{C_1 \setminus c_{2,2}\} \cup \{c_{3,2}, c_{3,3}, c_{3,4}\}$.

From the modeling perspective, the above procedure has the following two important consequences:

1. Allowing only unmodeled children of previously modeled nodes to enter the model ensures that each inducing input enters the model only once. This ensures that the iterative algorithm does not get stuck in a recursive loop in searching through the grid.

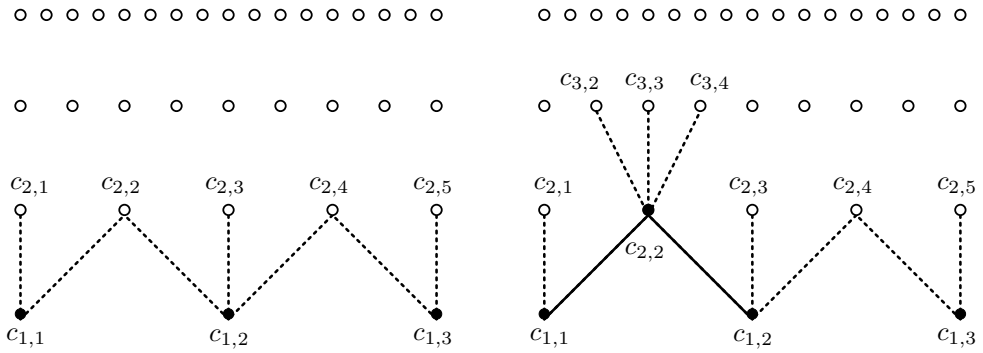
2. Because new inputs can only enter the pool of allowable candidate nodes through refinement of parent nodes, no child node is added to the model before at least one of its parent nodes is modeled. This imposes intergenerational continuity on the inputs and thus ensures that the model descends analysis grid in order.

The location of the unmodeled children of previously modeled nodes corresponds to all the previously sampled locations. Thus, in each iteration, all previously sampled locations are eligible for refinement. However, the resolution of sample refinement is limited by the corresponding resolution of allowable nodes. As a consequence of intergenerational continuity of inputs, a region in the domain is sampled at a given resolution ρ_k, only after its neighborhood has been sampled at resolution ρ_{k-1}.

As a consequence of the grid-based regime, if a child node of a selected node is already modeled or sampled, it is not sampled again. Also, if a node is selected after both its neighboring nodes $c_{\cdot,j-1}$ and $c_{\cdot,j+1}$ have been selected, then no new samples need be taken. In such an iteration, only an inducing input need be introduced into the model and only the input corresponding to the middle child is newly added to the set of allowable nodes because the other child nodes would have already been introduced by the neighboring nodes.

5.2 Maximum Sampling Resolution

The procedure described above allows for traversal of the analysis grid down to an arbitrary grid depth with arbitrarily high sampling resolution via refinement. However, in practice, the maximum number of samples allowed and the maximum sampling resolution are typically limited.



(a) (b)

FIG. 9: Grid traversal in BMS algorithm. (a) Iteration 1 and (b) iteration 2.

In this paper, the maximum allowable sampling rate is assumed to correspond to the highest resolution of the grid and the maximum sample size, $N_{\max}$, is assumed to be less than 2^M, where M is the highest grid level. Also, the Mth-level nodes are modeled without further refinement and therefore do not spawn new child nodes.

5.3 Modeling

In BMS, because the refinement of a selected node C_i^* precedes its modeling, the number of samples (N) is always more than the number of modeled nodes (N_k) and the resulting linear system represents an approximation rather than interpolation. In order to start the algorithm, the BMS is initialized by selecting all layer 1 nodes, C_1. Thus, $C_1^* = C_1$. Refinement of these nodes results in samples $f(X_1)$, at the location of all the layer 2 nodes, $C_R = C_2$. Thus, $X_1 = C_2$, which corresponds to the location of level $k = 2$ nodes on the grid. The BMS algorithm is presented as follows:

BMS Algorithm

1. $i \leftarrow 1$
2. `set` $N_{\max}$
3. `initialize:` $e_i \leftarrow f$, $\hat{f}_{i-1} \leftarrow 0$, $X_{i-1} \leftarrow \emptyset$
4. `bootstrap:` $C_A \leftarrow C_1$, $C_i^* \leftarrow C_1$
5. $C_R \leftarrow$ `REFINE(`C_i^*`)`
6. $X_i \leftarrow X_{i-1} \cup C_R$

7a. `estimate:` $\hat{\Theta}_i = \min J[\Theta_i]$

7b. `synthesis:` $\hat{e}_i \leftarrow \mathcal{SGP}\left(\mathbf{x}; C_R, C_i^*, \hat{\Theta}_i\right)$

8. `if` $|X_i| \geq N_{\max}$ `terminate`.
9. $C_A \leftarrow (C_A \setminus C_i^*) \cup C_R$
10. $C_{i+1}^* \leftarrow C_i^* \cup$ `SELECT`$(C_A, e_{i+1}(C_A))$
11. $i \leftarrow i + 1$
12. `goto` 5

where C_i^* is a node in the set of allowable candidate nodes C_A, which is selected for refinement and subsequent modeling in the ith iteration. This selection is based on a criterion, `SELECT`, which uses the residues at location of nodes in C_A to identify the selected node (see Section 5.4). Refinement of the selected node is implemented by the `REFINE`(C_i^*) procedure, which identifies the location of the child nodes of the selected nodes (Section 4.3). After the child nodes have been identified, the underlying function is sampled at the location of child nodes and, subsequently, the child nodes are added to the set of allowable candidate nodes for further possible refinement.

5.4 Selection Criteria

In BMS, before a selected node is modeled, it is refined by sampling at the location of its child nodes. As a consequence, information about the error in the ith-level approximation $\hat{f}_i$ is available as residues $e_{i+1}(X_i) = f(X_i) - \hat{f}_i(X_i)$ at all the previously sampled locations, X_i. These correspond to the location of all nodes in the set of allowable candidate nodes C_A. In BMS algorithm, the information contained in these residues is directly employed in each iteration to select a node, c_{i+1}^* from the set C_A such that the resulting error in approximation is systematically reduced. Specifically, the node corresponding to the greatest residue is selected from the set of allowable candidate nodes for modeling in the next iteration

$$c_{i+1}^* = \underset{c_\lambda \in C_A}{\arg\max} \left(|e_{i+1}(c_\lambda)|\right) \tag{25}$$

5.5 BMS Illustration

We revert to the test function in Eq. (22) to illustrate the BMS algorithm. A maximum sampling resolution of $(b-a)/2^6$ is assumed. Thus, $M = 6$ and the maximum number of samples is $2^6 + 1 = 65$.

The first 20 inducing inputs selected by BMS are shown in Fig. 10(a). The correspondence between the inducing input distribution and variation in the underlying function is noted.

Because the locations of unmodeled children (leaf nodes in the tree) of previously modeled nodes are the training sample locations, the correspondence of inducing input distribution to variation extends to training sample distribution [Fig. 10(b)]. In contrast, the training sample distribution due to VMS algorithm, shown in Fig. 10(b), is relatively uniformly distributed. As a consequence, as seen in Fig. 10(b), the approximation due to BMS algorithm captures the variability of the test function in $[-1, 2]$ whereas the approximation due to VMS does not. In addition, due to the behavior discussed previously, VMS expends more samples in the domain boundaries, where there is no variation in the underlying function.

In order to characterize the variation-sensitive training sample distribution generated by BMS, we consider the effect of translation of the underlying function within the domain. Toward this end, we consider the sample distributions generated for test function of the type $f(x - t)$, where $t \in \Omega$ is the translation parameter. Figure 11 shows the training-point distributions for various translations of the test function (22). Each row shows the distribution of the first 30 training points generated when $f(x)$ is translated by t. As the function, $f(x)$ moves from –3 to 3, the training-point distribution shifts correspondingly; hence, the density of the training points along the diagonal across the image is higher. On the basis of this experiment, for each t, the size of the interval $[t - 1, t + 2]$ (in Fig. 10(b), this interval corresponds to [–1,2]) represents 36.18% of the domain Ω and contains 59.92% of the samples, on average. This experiment illustrates that in addition to localization of the features in the underlying function, BMS training-point distributions achieve variation sensitivity because the density of the training points in feature regions is higher when compared to other regions of the domain.

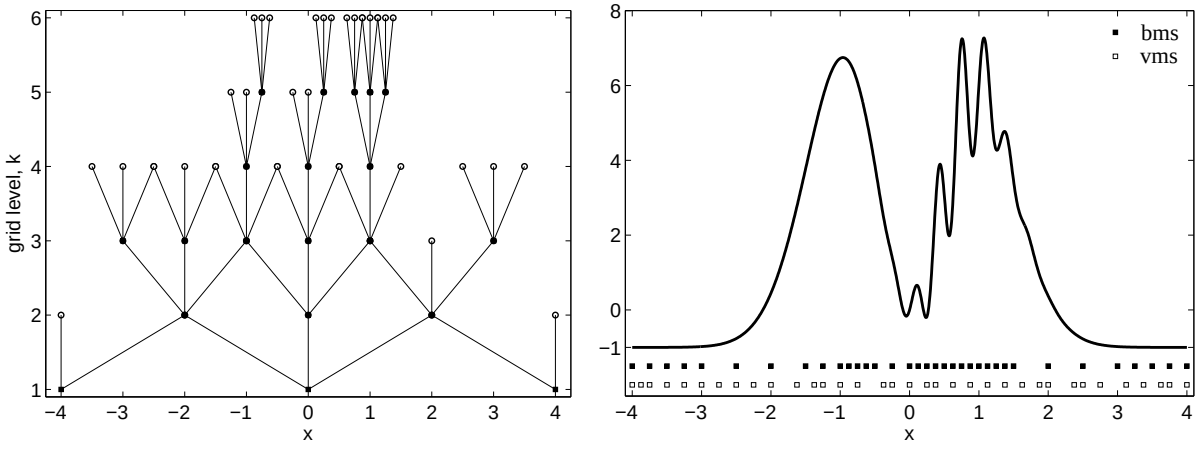
In addition to the 1D test function above, the performance of the BMS algorithm is evaluated against the following 2D functions:

$$f_1(x, y) = 2e^{-5\left(0.5(x-1.2)^2+2y^2\right)} \tag{26}$$

$$f_2(x, y) = \texttt{peaks}(x, y) \tag{27}$$

$$f_3(x, y) = f_1(x, y) + \text{erf}(x - .3) + 2\text{erf}(y - 2.21) \tag{28}$$

$$f_4(x, y) = f_2(x, y) + \text{erf}(x - .3) + \text{erf}(y - 2.21) \tag{29}$$



(a) (b)

FIG. 10: Comparison of BMS and VMS for the test example (22): (a) Shows the model tree and the first 20 modeled nodes, which correspond to the location of inducing inputs, C_i^* in BMS and (b) the corresponding 35 BMS sample locations, X_i are superimposed with the test function and the corresponding training points selected by VMS. Note the spatial localization of the variation in the underlying function by both the inducing inputs and sample locations in BMS.

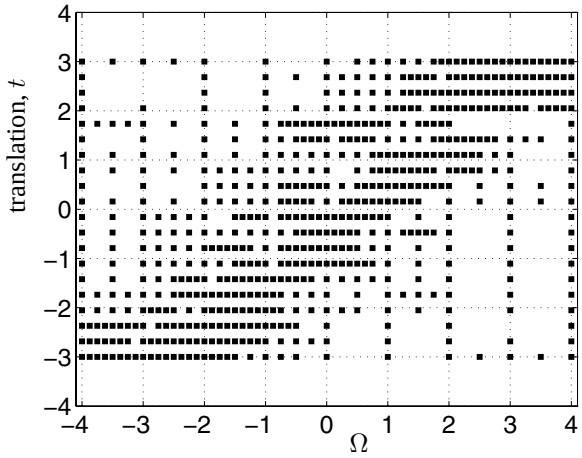


FIG. 11: Distribution of training points generated by BMS for various translations of the test function in Eq.(22).

where `peaks` is a MATLAB function:

$$f_2(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right) e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2} \tag{30}$$

and

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{31}$$

These functions are modeled at a maximum grid level, $M = 6$, and, thus, the maximum number of possible training-point locations is 4225. In subsequent comparisons, both BMS and VMS select locations from this set of possible training point locations and the difference between the predicted and the true function value at unselected locations is used a measure of model error.

The four 2D test functions and the sample distributions due to BMS and VMS are shown in Figs. 12 and 13. In all the cases, when compared to VMS, BMS achieves higher sample rates in feature regions within the boundary. In addition to the features within the boundary, functions f_3 and f_4 have variations in the interval [–2,2] along the four boundaries. In the case of f_3, BMS accounts for this variation on three dimensions; in addition, the sample density in the interior feature is now sparser. In the case of f_4, at 150 samples, BMS has not accounted for variations along the boundary at all. This behavior is due to the sequential nature of the algorithm and the fact that the node selection is based only on the residues and not the error estimate, which is presented in Section 5.6. Thus, through sequential search and modeling, BMS achieves a training sample distribution that is representative of the features of the underlying function.

5.6 Error Estimate

The internal hierarchical model, $\hat{f}_k$, employed by the algorithm represents a mechanism with which information content in the sample distribution may be extracted. In addition, a sampling of the corresponding approximation error $\mathbf{e}_{k+1} = f - \hat{f}_k$ is available at the location of the unmodeled nodes, C_A. In each iteration of BMS, the information contained in these residues can be used to obtain an estimate of the true error, $\mathbf{e}_{k+1}$. In addition to $\hat{f}_k$, such an estimate may then be employed to characterize the quality of approximation to the underlying function, f_k, implemented by BMS.

One such estimate of the error, e_{k+1}, is obtained by employing the bases in the set of allowable candidate nodes, C_A, to model the residue

$$\hat{\mathbf{e}}_{k+1} = \mathcal{GP}(x, C_A, \Theta) \tag{32}$$

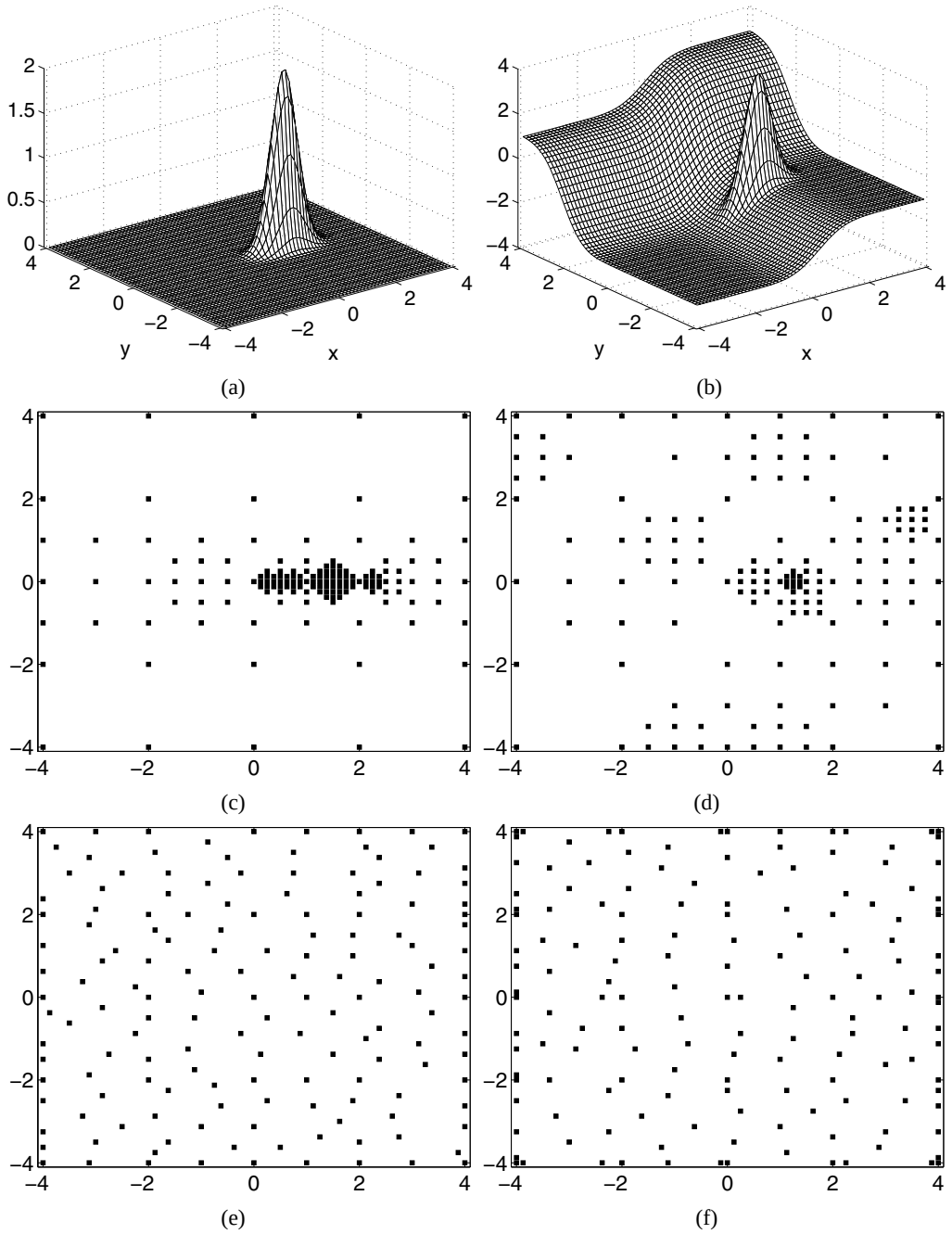


FIG. 12: Comparison of BMS and VMS for 2D test examples. (a) f_1, (b) f_3, (c) f_1 BMS samples, (d) f_3 BMS samples, (e) f_1 VMS samples, and (f) f_3 VMS samples.

When the sample density is low, such an estimate is expected to be biased. However, as the sample density increases, a richer set of possible inducing inputs become available in C_A, and the fidelity of the estimate to the true approximation error improves. The estimate described here is intended to provide a practical feedback about the quality of the function estimate $\hat{f}_k$, and may be used as a qualitative description of the sequential search implemented by BMS.

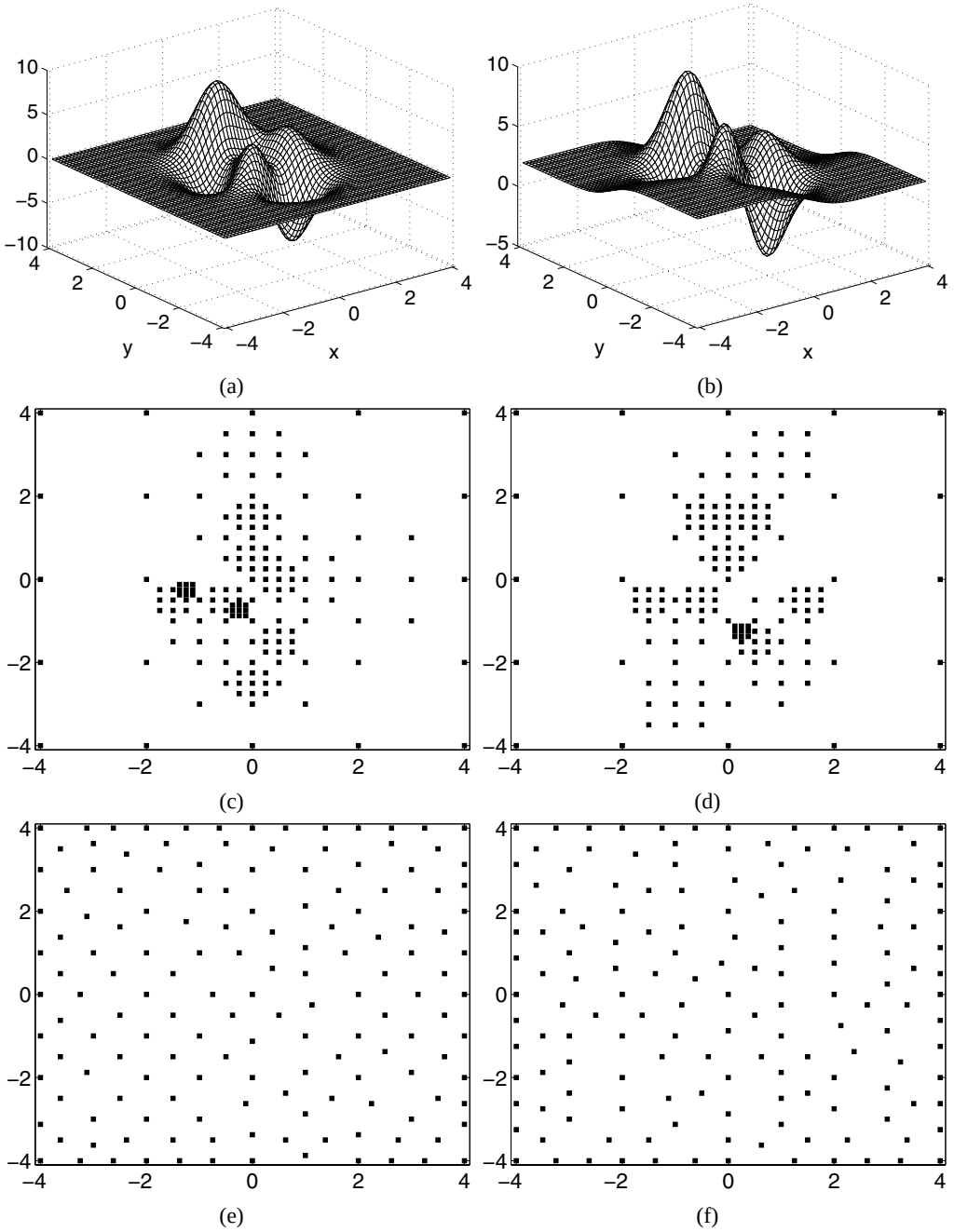


FIG. 13: Comparison of BMS and VMS for 2D test examples. (a) f_2, (b) f_4, (c) f_2 BMS samples, (d) f_4 BMS samples, (e) f_2 VMS samples, and (f) f_4 VMS samples.

5.7 Error Rate Comparison

In Fig. 14, the BMS reconstruction error as a function of the number of samples is shown for the 1D test function. Also shown is the reconstruction error due to the VMS algorithm. Figure 15 shows the corresponding curves for the 2D test functions. As is seen in those figures, BMS achieves a systematic reduction in the error through sequential addition

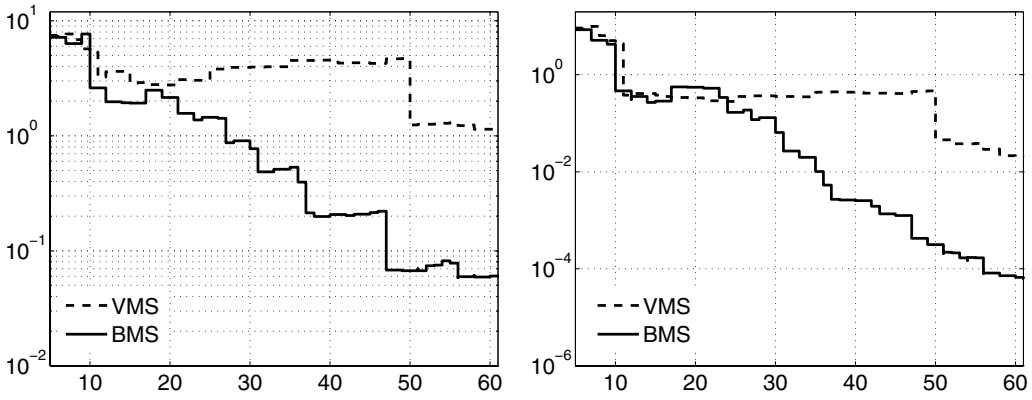


FIG. 14: Log error versus number of samples for the 1D test example. (a) Max error and (b) mean error.

of localized bases with localized sampling. Furthermore, the performance of BMS is superior to the performance of VMS.

5.8 Computational Complexity

The computational complexity in model construction for GPR is $O(n^3)$ and therefore directly related to the number of training points, n. However, SGP reduces the computational complexity in model construction to $O(nm^2)$, where m is the number of inducing inputs, which is always less than the number of training points. Thus, the computational complexity of the HGP and BMS algorithms is primarily in terms of the number of inducing inputs, rather than the number of training points and for the same number of training points the computational complexity in the construction of BMS is lower than VMS. In terms of memory requirements, the VMS algorithm requires storage of the n training points and, in addition to the parameters of the GP, which depend on the choice of the covariance function. For the squared exponential covariance, the number of parameters is $D + 1$; thus, the memory requirements of the VMS algorithm are of the order of $O(n + D)$. In BMS, in addition to the location of training points, the location and parameters corresponding to the inducing inputs must also be stored. Thus, the memory requirements are of the order of $O[n + (m + 1)D]$.

BMS represents a sequential algorithm in which $(2^1 + 1)^D$ inducing inputs are introduced in the model in the first iteration of the algorithm. In subsequent iterations, only one inducing input is introduced into the model and the computational complexity for these iterations is therefore linear in the number of training points $O(n)$.

However, in the initial iteration, even with three inducing inputs in each dimension, the complexity grows exponentially as the number of dimensions increases. This exponential growth is due to the use of the ternary tree as the hierarchical grid. Although such a geometric grid allows for localization of the features of the underlying function, it makes the first iteration of the BMS algorithm expensive for higher dimensional problems.

The principal advantage of VMS over BMS is in that VMS allows for the locations and number of the training points in the first iteration to be arbitrarily selected. In future work, a similar extension of BMS may be considered where an arbitrary subset of the first layer of the analysis grid is selected to start the algorithm.

It is however noted that in both VMS and BMS, the initial set of training points guides the selection of the subsequent training-point locations and; therefore, affects the quality of the surrogate model and the number of samples required to achieve a model of given quality. For BMS, the initial-sample distribution should provide good coverage of the domain, which can lead to subsequent discovery of the features of the underlying function.

For efficient training of VMS, the initial sample distribution should yield a surrogate model whose estimated length-scale parameters are appropriate to the variation in the underlying function. This requires an initial sample distribution that has a sufficiently large number of samples spread throughout the domain. In other words, the VMS

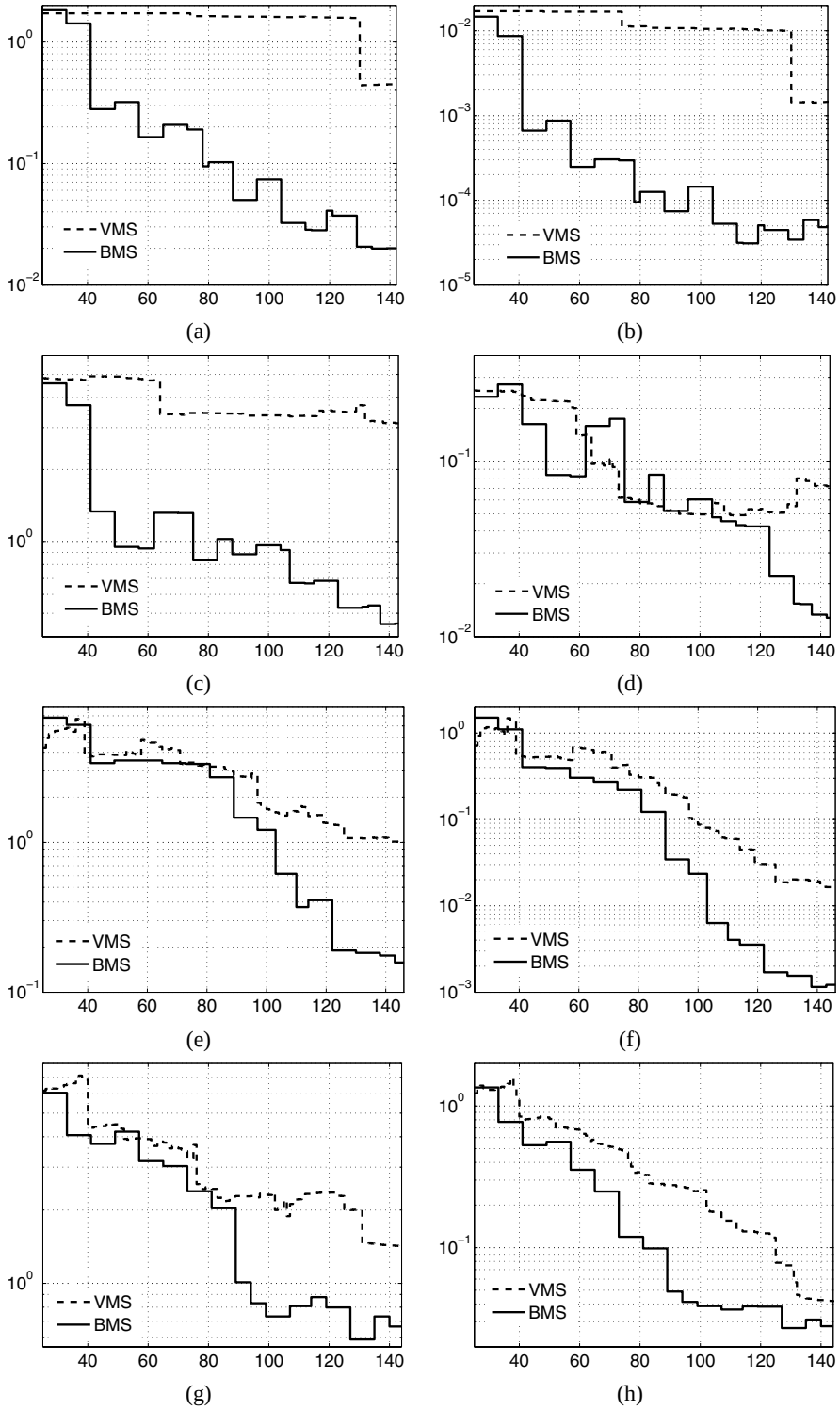


FIG. 15: Log error versus number of samples for 2D test examples. (a) f_1 log max error, (b) f_1 log mean error, (c) f_3 log max error, (d) f_3 log mean error, (e) f_2 log max error, (f) f_2 log mean error, (g) f_4 log max error, and (h) f_4 log mean error.

algorithm aims to achieve space filling of the domain with respect to the prediction variance. Because the prediction variance of GP depends on the distance between the training points and the prediction point, in general, the VMS algorithm selects the farthest unsampled regions. However, as seen in Fig. 1, due to the asymmetry of training points induced by the domain boundary, the prediction variance tends to be higher in the neighborhood of the domain boundary. Consequently, if the initial training-point density is not sufficiently high, the VMS algorithm tends to choose training points along the boundary at a higher rate than is required. As seen in Figs. 12(e), 12(f), 13(e) and 13(f), the sample density along the boundary points is higher than it is in the interior of the domain.

Thus, both VMS and BMS require a sufficiently large number of training points in the initial set to provide domain coverage. In high-dimensional problems such initial training points are prohibitive to obtain due to the "curse of dimensionality". In addition to the initial training points, the curse of dimensionality also affects the subsequent sample distribution of VMS. Few functions exhibit consistent variation all across the domains. In higher dimensions, due to the increase in volume of the input space, this concentration of features in the domain is more pronounced. Thus, VMS-based surrogate models require a significantly higher number of samples spread throughout the domain to capture the true variation in the underlying function. In contrast, localization provided by the BMS algorithm allows for more efficient (in terms of number of samples) construction of the surrogate model even in higher dimensions. This is illustrated by the relative performance of the VMS and BMS algorithms for the 2-D test function f_1. In general, construction of surrogate models for high-dimensional problems involves a pronounced trade-off between the quality of approximation and the number of training points.

5.9 Uncertainty Quantification

Computational efficiency of surrogate models allows for efficient computation of output uncertainty due to uncertainties in the input variables. In this section, the two surrogate models are employed for uncertainty quantification and the effect of the quality of the approximation implemented by the surrogate models on the quality of the uncertainty estimates is discussed.

Toward this end, we consider the two-dimensional test function f_1 [Eq. (26)]. The uncertainties in the two input variables are assumed to be independently normally distributed: $x \sim \mathcal{N}(1, 0.05)$ and $y \sim \mathcal{N}(0, 0.05)$. Baseline estimates of the statistics of the response value were computed using Monte Carlo sampling of the underlying function with 10^5 samples of the two input variables. The corresponding estimates of the mean and variance of the response variable were 1.7575 and 0.0112, respectively, and the histogram of the relative frequencies is shown in Fig. 16(a).

The VMS and BMS surrogate models, trained with various numbers of training points, were evaluated with the same samples of the input variables used above. The histogram of the relative frequencies of the responses of the VMS and BMS surrogate models constructed using 200 training points is shown in Figs. 16(b) and 16(b), respectively. It is seen that the distribution of the predicted values of the surrogate model constructed with BMS is similar to the histogram of the original function. In contrast, at 200 samples, the corresponding distribution of the surrogate model constructed with VMS is considerably different from that of the original function. The estimated means and variances of the two surrogate models constructed using a different number of training points is shown in Fig. 17. As is seen in Fig. 17(a), for BMS the estimated mean and variance of the output converges faster than VMS to the mean and variance computed using the original function.

The dominant feature in the function f_1 is an exponential peak that is highly localized in the domain [Fig. 12(a)]. The surrogate model constructed using VMS is dominated by samples along the boundary and in the interior of the domain, where the function displays no variation and is therefore unable to localize the feature. The VMS surrogate model is consistently biased over several sample rates [Fig. 15(a)]. This results in poor estimates of the uncertainties in the response variable. On the other hand, the surrogate model constructed using the BMS algorithm localizes the feature and thus provides consistently better estimates of the means and variances over several sample rates. This is further illustrated by considering the one-dimensional test function

$$f_5(x) = -0.4 \tanh(50x), \quad x \in [-4, 4] \tag{33}$$

which displays a discontinuity in the functional behavior due to the sharp variation around $x = 0$ (Fig. 18). In order to consider uncertainty propagation estimates in the region of the discontinuity, the input distribution was chosen to be

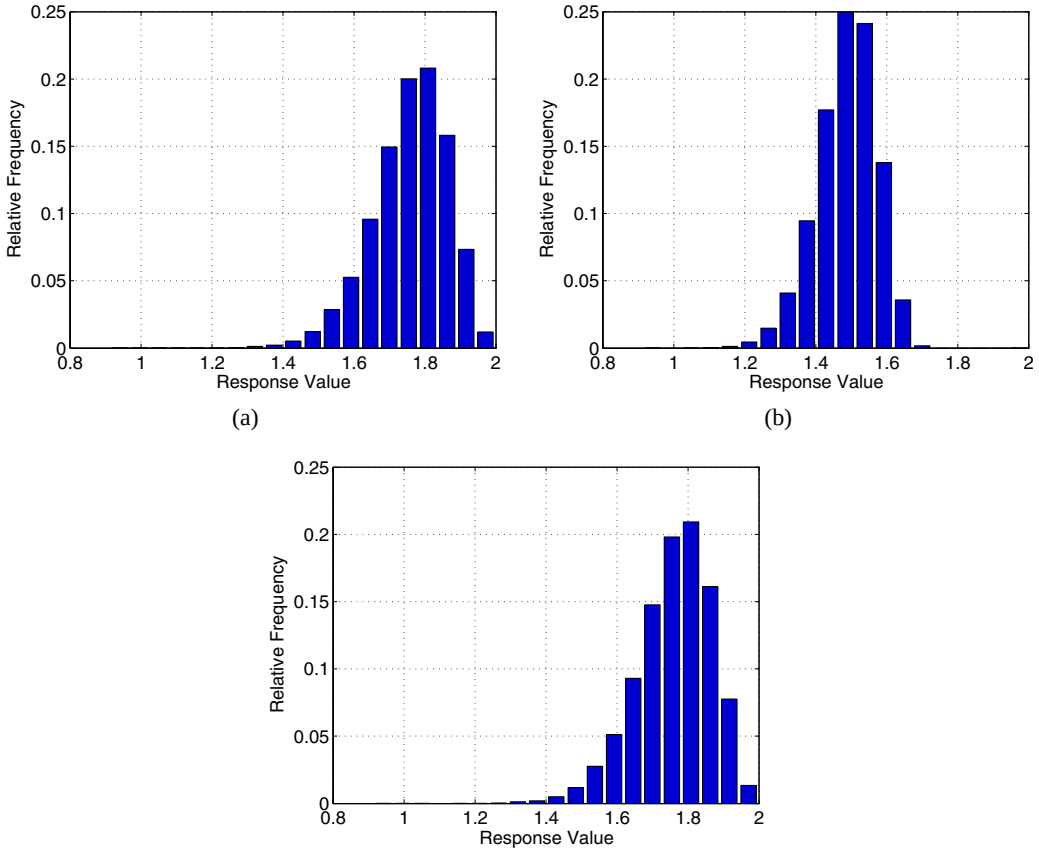


FIG. 16: Uncertainty quantification for function f_1 using Monte Carlo simulation: Histograms of relative frequencies of the response value of function f_1. (a) f_1, (b) VMS (200 training points), and (c) BMS (200 Training points).

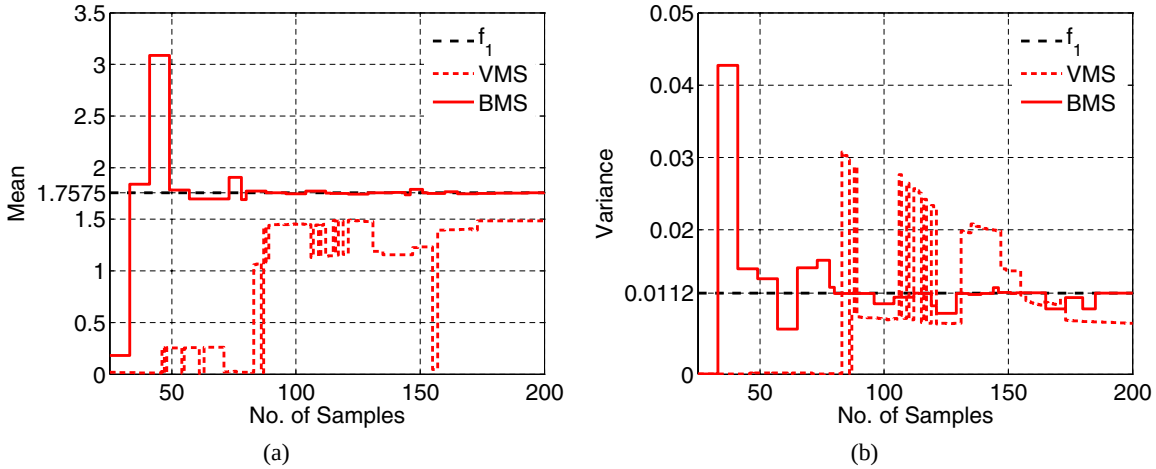


FIG. 17: Uncertainty Quantification for function f_1: estimated mean and variances at test point $(1, 0)$ with variance of 0.05. Thick horizontal line indicates the mean and the variance estimated using the original function. (a) Mean and (b) variance.

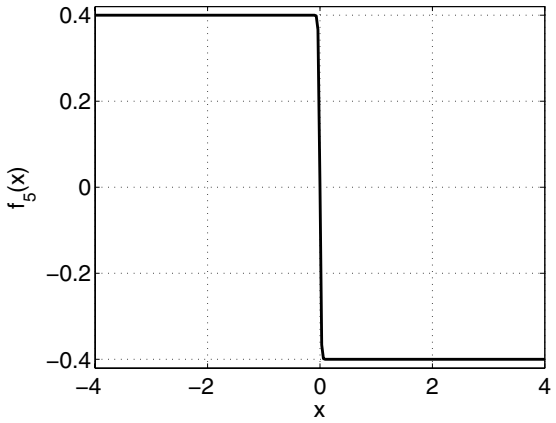


FIG. 18: Function f_5 for uncertainty quantification.

$x \sim \mathcal{N}(0.07, 0.01)$, which corresponds to the location of the lower bend in Fig. 18. On the basis of 10^5 samples from this input distribution, the corresponding estimates of mean and variance of the response variable were estimated to be -0.3988 and 2.7318×10^{-6}. A performance similar to that of the original test function (f_1) was observed: BMS provides better localization and approximation quality (Fig. 19). In addition, the uncertainty estimates of BMS converged faster than those due to VMS (Fig. 20).

6. CONCLUSION

In the approximation of high-fidelity computer simulations using surrogate models, the underlying function is invariably undersampled because factors such as the size of the domain and the number of input variables limit the sample budget. Thus, perfect reconstruction of the underlying function is not possible. Instead, the underlying function must be reconstructed as an approximation using the limited data available.

The principal motivation behind this work was to seek alternatives to VMS designs for the construction of the widely used GP surrogate models. Our preliminary studies revealed that VMS designs are geared toward improving the statistical properties of the approximation. Although such designs are desirable, they are not directly related to reducing the bias in approximation with respect to the underlying function. Our primarily goal was to investigate the possibility of generating bias-minimizing designs and to understand the differences between bias-minimizing designs and variance-minimizing designs.

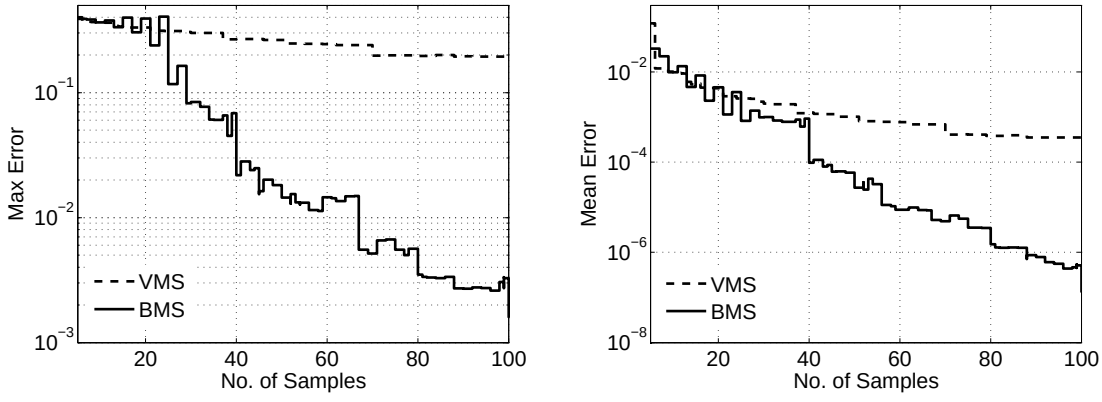
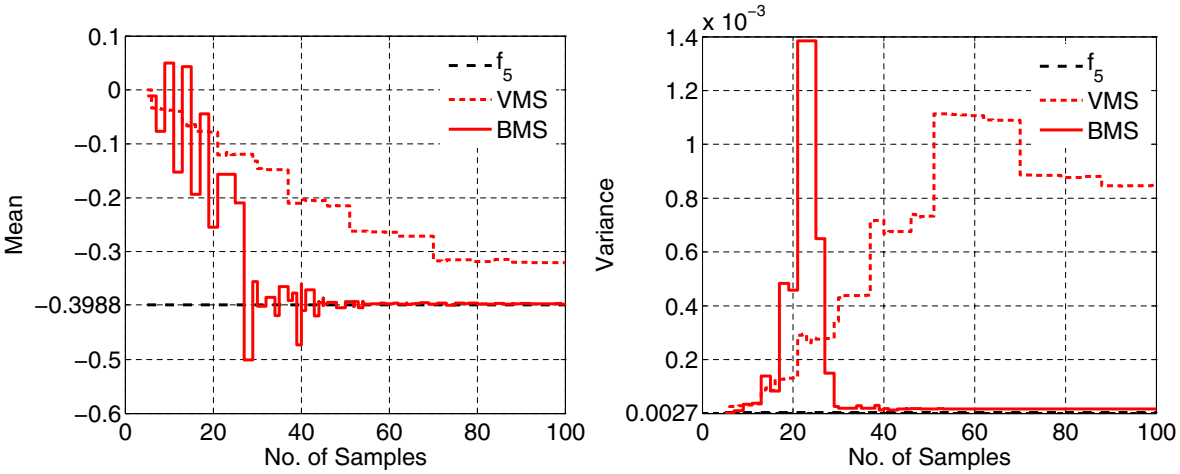


FIG. 19: Log error versus number of samples for function f_5. (a) f_5 log max error and (b) f_5 log mean error.



(a) (b)

FIG. 20: Uncertainty quantification for function f_5: Estimated mean and variances at test point 0.07 with variance of 0.01. Thick horizontal line indicates the mean and the variance estimated using the original function. (a) Mean and (b) variance.

Because the underlying function is not known, the optimal n-sample distribution cannot be determined a priori; instead, a suboptimal sequential estimate must be sought. Toward this end, we first developed the HGP algorithm, which achieves hierarchical decomposition of a known function on a tree and suggests the possibility of a variation-sensitive modeling and compressed representation. The BMS algorithm was then developed as a serialization of the HGP algorithm that achieves variation-sensitive modeling through sequential discovery of significant nodes in the tree and in turn achieves variation-sensitive sample distribution through coarse-to-fine sample refinement.

Using numerical examples, it is shown that using such sample distributions, critical features of the underlying function may be adequately resolved with greater sampling economy than previous methods. Thus, for the same number of samples, such distributions result in higher quality reconstructions when compared to previous methods.

In BMS, the location of the inducing inputs is restricted to the grid. In each iteration, the node with the largest error is selected as the inducing input to be introduced into the model. Restricting the location of inducing inputs provides a localization of the features in the underlying function. With the location of the inducing input fixed, the fidelity of the approximation is controlled by selecting the length-scale and variance parameters that result in the largest reduction of the approximation error.

Typically, the parameters of both GP and SGP are identified using MLE. However, in HGP and BMS algorithms the parameters of the approximating function are identified using a least-squares criterion [Eq. (19)] so as to directly reduce the approximation error.

The proposed framework is not unique to GPs, and any regression model that uses bases with local or quazi-local support may be employed in this framework (e.g., RBF [37]). However, the location of the training points depends on the class of approximations considered and, therefore, even within the overall framework of GPs, the best training points for the squared exponential covariance function will be different from those for the Matern covariance functions.

The proposed method is based on the hierarchical approximation of the fitting error of a surrogate model in which only the mean prediction is considered. A more robust inference would employ the entire predictive distribution and is part of the further work. Considering uncertainties at each level of the approximation will also lead to probabilistic node selection criteria, which considers both the mean and variance in the residues instead of the current deterministic criteria, which only considers the mean. Such a probabilistic node selection is expected to provide a balance between bias and variance and is likely to yield a “breadth-first” traversal of the grid rather than the “depth-first” traversal implemented by the current deterministic criteria.

One possible way to handle multivariate responses is to construct individual SGPs for each response. When combined with an appropriate metric that combines the errors in each response, the BMS algorithm may be extended for multivariate responses and we obtain one sample distribution for all the dimensions. However, it is noted that for highly multivariate responses, construction of individual surrogate models is computationally expensive. In addition, a fundamental issue with modeling each output independently is the correlation between the output variables. Although functional or statistical correlations across the individual responses may be exploited to "compress" the dimension of the output space, such compressions may not always be possible.

In addition to the issues concerning the modeling of the multiple outputs and the estimation of the surrogate model parameters, a metric that combines the errors in each output dimension must be identified. For multivariate responses, the selection of this metric is not trivial and expected to have consequences on the sample distribution and quality of the reconstruction. For this reason, the selection procedure could involve a multiobjective optimization to account for the errors and variations in each dimension or could consider aggregate measures, such as norms of the error residuals: $|\mathbf{e}|_1$, or $|\mathbf{e}|_2$, where $\mathbf{e}$ is a vector consisting of error residue in each dimension.

In general, training-point selection and the construction of surrogate models for multivariate outputs is not analogous to the univariate case, and the behavior of the algorithm for the multivariate case does not immediately follow from its behavior for the univariate case. The quality of the uncertainty estimates depends on the quality of the approximation. By considering the quality of the approximation over the entire domain, the constructed surrogate model exhibits high fidelity to the underlying function and thus provides accurate uncertainty quantification throughout the defined domain. In general, the domain represents the range of possible input variables. However, in certain uncertainty quantification applications, only certain regions of the domain may be of interest. In such cases, it may be more important to seek quality of approximation within the region of interest. Toward this end, the domain must be redefined by using the distributions of the input variables to identify the regions of interest.

In addition to the results presented in this work, further evaluation of the performance of the algorithm for different test cases is needed. In addition, further study of the properties and characteristics of the algorithm is warranted, including extension of the algorithm to higher dimensions and the possibility of a more efficient selection criteria apart from the greatest residue heuristic employed in this work.

ACKNOWLEDGMENTS

This study was sponsored in part by the Sandia National Laboratories (Contract No. BG-7732, Technical Monitor: Dr. Angel Urbina) and, in part, by NASA Langley Research Center under Cooperative Agreement No. NNX08AF56A1 (Technical Monitor: Lawrence Green). The support is gratefully acknowledged.

REFERENCES

1. Liang, B. and Mahadevan, S., Error and uncertainty quantification and sensitivity analysis in mechanics computational models, *Int. J. Uncertainty Quantification*, 1:147–161, 2011.
2. Sankararaman, S., Ling, Y., and Mahadevan, S., Uncertainty quantification and model validation of fatigue crack growth prediction, *Eng. Fracture Mech.*, 78:1487–1504, 2011.
3. Simpson, T., Poplinski, J., Koch, P., and Allen, J., Metamodels for computer-based engineering design: Survey and recommendations, *Eng. Comput.*, 17(2):129–150, 2001.
4. Xiong, Y., Chen, W., Apley, D., and Ding, X., A non-stationary covariance-based Kriging method for metamodelling in engineering design, *Int. J. Numer. Methods Eng.*, 71(6):733–756, 2007.
5. Rasmussen, C. and Williams, C., *Gaussian Processes for Machine Learning*, Springer, New York, 2006.
6. Sacks, J., Welch, W., Mitchell, T., and Wynn, H., Design and analysis of computer experiments, *Stat. Sci.*, 4(4):409–435, 1989.
7. Powell, M., Radial basis functions for multivariable interpolation: A review, In Algorithms for Approximation, Mason, J. C. and Cox, M. G. (eds.), Clarendon Press, NY, pp. 143–167, 1987.
8. Ma, X. and Zabaras, N., An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations, *J. Comput. Phys.*, 228(8):3084–3113, 2009.

9. Wahba, G., Spline models for observational data, *CBMS-NSF Regional Conference Series in Applied Mathematics*, Ohio State University, Columbus, SIAM, Philadelphia, March 23–27, 1987.
10. Eldred, M., Webster, C., and Constantine, P., Evaluation of non-intrusive approaches for wieneraskey generalized polynomial chaos, *Proceedings of 10th AIAA Non-Deterministic Approaches Conference*, Paper No. AIAA-2008-1892, Schaumburg, IL, 2008.
11. Ghanem, R. and Spanos, P., *Stochastic Finite Elements: A Spectral Approach*, Dover, New York, 2003.
12. Kennedy, M. and O'Hagan, A., Bayesian calibration of computer models, *J. R. Stat. Soc., Series B*, 63(3):425–464, 2001.
13. Simpson, T., Lin, D., and Chen, W., Sampling strategies for computer experiments: design and analysis, *Int. J. Reliab. Appl.*, 2(3):209–240, 2001.
14. MacKay, D., Introduction to Gaussian processes, *NATO ASI Series F Comput. Syst. Sci.*, 168:133–166, 1998.
15. Poggio, T. and Girosi, F., Networks for approximation and learning, *Proc. IEEE*, 78(9):1481–1497, Sep. 1990.
16. Cohn, D., Ghahramani, Z., and Jordan, M., Active learning with statistical models, *CoRR*, cs.AI/9603104, 1996.
17. Seo, S., Wallat, M., Graepel, T., and Obermayer, K., Gaussian process regression: Active data selection and test pointrejection, *IJCNN 2000, Proceedings of IEEE-INNS-ENNS International Joint Conference on Neural Networks, 2000*, 3:241–246, 2000.
18. Sung, K. and Niyogi, P., Active learning for function approximation, *Proceedings of Advances in Neural Information Processing Systems*, MIT Press, pp. 593–600, 1995.
19. MacKay, D., Information-based objective functions for active data selection, *Neural Comput.*, 4(4):590–604, 1992.
20. Tong, S., Active learning: Theory and applications, PhD thesis, Stanford University, 2001.
21. Guestrin, C., Krause, A., and Singh, A., Near-optimal sensor placements in Gaussian processes, *Proceedings of the 22nd International Conference on Machine learning*, ACM Press, New York, 22:265–272, 2005.
22. Krause, A. and Guestrin, C., Nonmyopic active learning of Gaussian processes: an exploration-exploitation approach, *Proceedings of the 24th International Conference on Machine Learning*, ACM Press, New York, pp. 449–456, 2007.
23. Gramacy, R., Lee, H., and Macready, W., Parameter space exploration with Gaussian process trees, *Proceedings of the 21st International Conference on Machine Learning*, ACM Press New York, 2004.
24. Box, G., Hunter, W., and Hunter, J., *Statistics for Experimenters: An Introductory to Design Data Analysis and Model Building*, Wiley Series in Probability and Mathematical Statistics, Wiley, Hoboken, NJ, 1978.
25. Sukhatme, P. and Sukhatme, B., *Sampling Theory of Surveys with Applications*, University Press, Ames, Iowa, 1970.
26. Bichon, B., Eldred, M., Swiler, L., Mahadevan, S., and McFarland, J., Efficient global reliability analysis for nonlinear implicit performance functions, *AIAA J.*, 46(10):2459–2468, 2008.
27. Paciorek, C., Nonstationary Gaussian processes for regression and spatial modelling, PhD thesis, Citeseer, 2003.
28. McFarland, J., Uncertainty analysis for computer simulations through validation and calibration, PhD thesis, Vanderbilt University, 2008.
29. Santner, T., Williams, B., and Notz, W., *The design and Analysis of Computer Experiments*, Springer-Verlag, Berlin, 2003.
30. Csató, L. and Opper, M., Sparse on-line Gaussian processes, *Neural Comput.*, 14(3):641–668, 2002.
31. Seeger, M., Williams, C., and Lawrence, N., Fast forward selection to speed up sparse Gaussian process regression, *Workshop on AI and Statistics*, Citeseer, 9:2003, 2003.
32. Silverman, B., Some aspects of the spline smoothing approach to non-parametric regression curve fitting, *J. R. Stat. Soc., Series B*, 47(1):1–52, 1985.
33. Smola, A. and Bartlett, P., Sparse greedy Gaussian process regression, *Proceedings of Advances in Neural Information Processing Systems*, MIT Press, pp. 619–625, 2001.
34. Snelson, E. and Ghahramani, Z., Sparse Gaussian processes using pseudo-inputs, *Proceedings of Advances in Neural Information Processing Systems*, MIT Press, pp. 1257–1264, 2006.
35. Quiñonero-Candela, J. and Rasmussen, C., A unifying view of sparse approximate Gaussian process regression, *J. Machine Learning Res.*, 6:1939–1959, 2005.

36. Mallat, S., *A Wavelet Tour of Signal Processing*, Academic Press, New York, 1999.

37. Hombal, V., Sanderson, A., and Blidberg, D., Adaptive multiscale sampling in robotic sensor networks, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 122–128, Oct. 2009.