Adaptive design and analysis of supercomputer experiments

Robert B. Gramacy
bobby@statslab.cam.ac.uk
Statistical Laboratory
University of Cambridge

Herbert K. H. Lee
herbie@ams.ucsc.edu
Dept of Applied Math & Statistics
University of California, Santa Cruz

Abstract

Computer experiments are often performed to allow modeling of a response surface of a physical experiment that can be too costly or difficult to run except using a simulator. Running the experiment over a dense grid can be prohibitively expensive, yet running over a sparse design chosen in advance can result in insufficient information in parts of the space, particularly when the surface calls for a nonstationary model. We propose an approach that automatically explores the space while simultaneously fitting the response surface, using predictive uncertainty to guide subsequent experimental runs. The newly developed Bayesian treed Gaussian process is used as the surrogate model, and a fully Bayesian approach allows explicit measures of uncertainty. We develop an adaptive sequential design framework to cope with an asynchronous, random, agent–based supercomputing environment, by using a hybrid approach that melds optimal strategies from the statistics literature with flexible strategies from the active learning literature. The merits of this approach are borne out in several examples, including the motivating computational fluid dynamics simulation of a rocket booster.

Key words: nonstationary spatial model, treed partitioning, sequential design, active learning

1 Introduction

Many complex phenomena are difficult to investigate directly through controlled experiments. Instead, computer simulation is becoming a commonplace alternative to provide insight into such phenomena (Sacks et al., 1989; Santner et al., 2003). However, the drive towards higher fidelity simulation continues to tax the fastest computers, even in highly distributed computing environments. Computational fluid dynamics (CFD) simulations in which fluid flow phenomena are modeled are an excellent example—fluid flows over complex surfaces may be modeled accurately but only at the cost of supercomputer resources. In this paper we explore the problem of fitting a response surface for a computer model when the experiment can be designed adaptively, i.e., online—a task to which the Bayesian approach is particularly well–suited. To do so, we will combine elements from treed modeling (Chipman et al., 2002) with modern Bayesian surrogate modeling (Kennedy and O'Hagan, 2001), and elements of the sequential design of computer experiments (Santner et al., 2003) with active learning (MacKay, 1992; Cohn, 1996). The result is a fast and flexible design interface for the sequential design of supercomputer experiments.

Consider a simulation model which defines a mapping, perhaps non-deterministic, from parameters describing the inputs to one or more output responses. Without an analytic representation of this mapping, simulations must be run for many different input configurations in order to build up an understanding of its possible outcomes. Even in extremely parallel computing environments, computational expense of the simulation and/or high dimensional input often prohibits the naïve approach of running the experiment over a dense grid of input configurations. More sophisticated design strategies, such as a Latin Hypercube sample (LHS), maximin designs, orthogonal arrays, and maximum entropy designs can offer an improvement over gridding. Sequential versions of these are better still. However, such traditional approaches are "stationary" (or global, or uniform) in the sense they are based on a metric (e.g., distance) which is measured identically throughout the input space. The resulting designs are sometimes called “sparse”, or “space-filling”. Maximum entropy designs are literally stationary when based on stationary models (e.g., linear or Gaussian Process models). Such sparse, or stationary, design strategies are a mismatch when the responses necessitate a nonstationary model—common in experiments modeling physical processes, e.g., fluid dynamics—as they cannot learn about, and thus concentrate exploration in, more interesting or complicated regions of the input space.

For example, NASA is developing a new re-usable rocket booster called the Langley Glide–Back Booster (LGBB). Much of its development is being done with computer models. In particular, NASA is interested in learning about the response in flight characteristics (lift, drag, pitch, side–force, yaw, and roll) of the LGBB as a function of three inputs (speed in Mach number, angle of attack, and side slip angle) when the vehicle is re-entering the atmosphere. For each input configuration triplet, CFD simulations yield six response outputs. Figure 1 shows the lift response as a function of speed (Mach) and angle of attack (alpha) with the side-slip angle (beta) fixed at zero. The figure shows how the characteristics

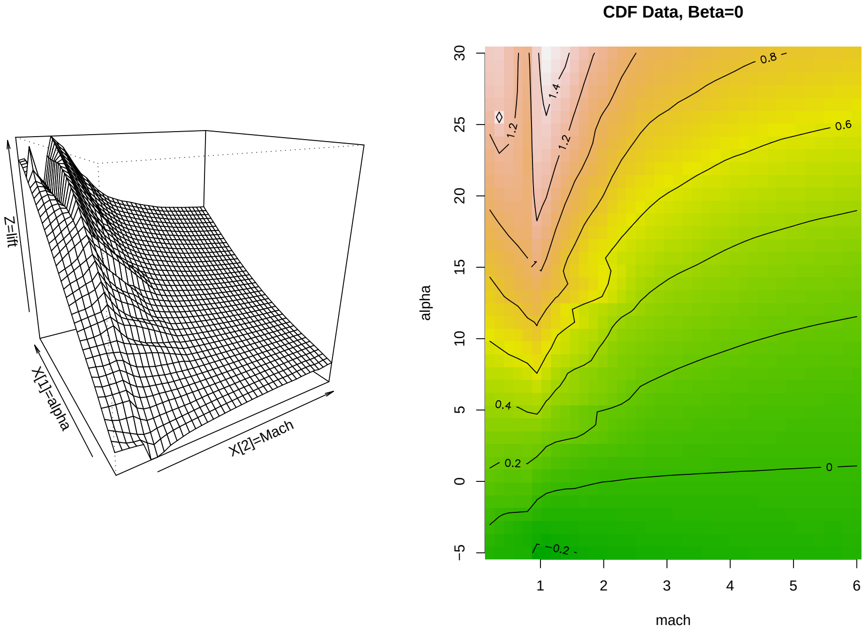

Figure 1: Lift plotted as a function of Mach (speed) and alpha (angle of attack) with beta (side-slip angle) fixed to zero. The ridge at Mach 1 separates subsonic from supersonic cases.

of subsonic flows can be quite different from supersonic flows, as indicated by the ridge in the response surface at Mach 1. Moreover, the transition between subsonic and supersonic is distinctly non-linear and may even be non-differentiable or non-continuous. The CFD simulations involve the iterative integration of inviscid Euler equations and are thus computationally demanding. Each run of the Euler solver for a given set of parameters takes on the order of 5–20 hours on a high end workstation. Since simulation is expensive, there is interest in being able to automatically and adaptively design the experiment to learn about where response is most interesting, e.g., where uncertainty is largest or the structure is richest, and spend relatively more effort sampling in these areas. However, before a clever sequential design strategy can be devised, a nonstationary model is needed that can capture the differences in behavior between subsonic and supersonic physical regimes.

The surrogate model commonly used to approximate outputs to computer experiments is the Gaussian process (GP) (Santner et al., 2003). The GP is conceptually straightforward, easily accommodates prior knowledge in the form of covariance functions, and returns estimates of predictive confidence. In spite of its simplicity, there are two important disadvantages to the standard GP in this setting. Firstly, the computation time for inference on the GP scales poorly with the number of data points, typically growing with the cube of the sample size. But most importantly, GP models are usually stationary in that the same covariance structure is used throughout the entire input space. In the application of high–velocity computational fluid dynamics, where subsonic flow is quite different from supersonic flow, this limitation is unacceptable. Therefore, the error (standard deviation) associated with a predicted response under a GP model does not locally depend on any of the previously observed output responses. The Bayesian treed GP model (Gramacy and Lee, 2008a) was designed to overcome these limitations.

Ideally, we would like to be able to combine the treed GP model with classic model–based optimal design algorithms. However, classic design algorithms are ill–suited to partition models and Bayesian Monte Carlo–based inference. They are inherently serial and thus tacitly assume a controlled and isolated computing environment. The modern supercomputer has thousands of computing nodes, or agents, designed to serve a multitude of diverse users. If the design strategy is not prepared to engage an agent as soon as it becomes available, then that resource is either wasted or devoted to another process. If design is to be sequential (which it must, in order to learn about the responses online, and adapt the model), then the interface must be asynchronous, and any computation must execute in parallel. There may not be time to re-fit the model and compute the next optimal design. So the final ingredients in our flexible design framework are active learning strategies from the Machine Learning literature. Such strategies have been used as fast, Monte Carlo–friendly, approximate alternatives to optimal sequential design (Seo et al., 2000).

Thus this paper makes two primary contributions: an integrated sequential design strategy for a modern nonstationary model, and methods for designing experiments in an asynchronous parallel computing environment. Our hybrid design strategy puts together the treed GP model, classic sequential design, and active learning, resulting in a highly efficient nonstationary model and sequential design combination that balances optimality and flexibility. The remainder of this paper is organized as follows. Section 2 reviews the main ingredients: from conventional optimal designs and active learning strategies with the canonical stationary GP model, to the nonstationary treed GP model with Bayesian model averaging and hybrid sequential design. Section 3 details our approach to the sequential design of supercomputer experiments with the treed GP surrogate model. Illustrative examples are given on synthetic data in Section 4. The motivating example of a supercomputer experiment involving computational fluid dynamics code for designing a re-usable launch vehicle (LGBB) is described in Section 5. Section 6 offers some discussion and avenues for further research.

2 Review

Our approach to the adaptive design of supercomputer experiments combines classic design strategies and active learning with a modern approach to nonstationary spatial modeling. These topics are reviewed here.

2.1 Surrogate Modeling

In a computer experiment, the simulation output $z(\mathbf{x})$, for a particular (multivariate) input configuration value $\mathbf{x}$, is typically modeled as a zero mean random process with covariance $C(\mathbf{x}, \mathbf{x}') = \sigma^2 K(\mathbf{x}, \mathbf{x}')$. The stationary Gaussian process (GP) is a popular example of such a model, and consequently is the canonical surrogate model used in designing computer experiments (Sacks et al., 1989; Santner et al., 2003).

For data $D = \{\mathbf{x}_i^\top, z_i\}_{i=1}^N = \{\mathbf{X}, \mathbf{Z}\}$ of m_X–dimensional inputs $\mathbf{X}$ and scalar observations $\mathbf{Z}$ under a GP model, the density over outputs at a new point $\mathbf{x}$ has a Normal distribution with

$$\begin{aligned} &\text{mean} && \hat{z}(\mathbf{x}) = \mathbf{k}^\top(\mathbf{x})\mathbf{K}^{-1}\mathbf{Z}, \quad \text{and} \\ &\text{variance} && \hat{\sigma}^2(\mathbf{x}) = \sigma^2[K(\mathbf{x}, \mathbf{x}) - \mathbf{k}^\top(\mathbf{x})\mathbf{K}^{-1}\mathbf{k}(\mathbf{x})] \end{aligned} \tag{1}$$

where $\mathbf{k}^\top(\mathbf{x})$ is the N–vector whose i^{th} component is $K(\mathbf{x}, \mathbf{x}_i)$, and $\mathbf{K}$ is the $N \times N$ matrix with i, j element $K(\mathbf{x}_i, \mathbf{x}_j)$. It is important to note that the uncertainty, $\hat{\sigma}^2(\mathbf{x})$, associated with the prediction has no direct dependence on the nearby observed simulation outputs $\mathbf{Z}$; because of the assumption of stationarity, all response points contribute to the estimation of the local error through their influence on the correlation function $K(\cdot, \cdot)$, and the induced correlation matrix $\mathbf{K}_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$. We follow Gramacy and Lee (2008) in specifying that $K(\cdot, \cdot)$ have the form

$$K(\mathbf{x}_j, \mathbf{x}_k|g) = K^*(\mathbf{x}_j, \mathbf{x}_k) + g\delta_{j,k}, \tag{2}$$

where $\delta_{\cdot,\cdot}$ is the Kronecker delta function, and K^* is a *true* correlation function. The g term, referred to as the *nugget*, is positive ($g > 0$) and provides a mechanism for introducing measurement error into the stochastic process. It arises when considering a model of the form $z(\mathbf{x}) = w(\mathbf{x}) + \eta(\mathbf{x})$ where $w(\mathbf{x})$ is the random process with covariance C, and $\eta(\cdot)$ is independent Gaussian noise. Valid correlation functions $K^*(\cdot, \cdot)$ are usually generated as a member of a parametric family, such as the separable power or Matérn families. A general reference for families of correlation functions K^* is provided by Abrahamsen (1997). Hereafter we use the separable power family,

$$K^*(\mathbf{x}_j, \mathbf{x}_k|\mathbf{d}) = \exp\left\{ -\sum_{i=1}^{m_X} \frac{|x_{ij} - x_{ik}|^{p_0}}{d_i} \right\}, \tag{3}$$

which is a standard choice in modeling computer experiments (Santner et al., 2003). We fix $p_0 = 2$ and infer the range parameters $\{d_i\}_{i=1}^{m_X}$ as part of our estimation procedure.

While many authors (e.g., Santner et al., 2003; Sacks et al., 1989) deliberately omit the nugget parameter on the grounds that computer experiments are deterministic, we have found it more helpful to include a nugget. Most importantly, we found that the LGBB simulator was only theoretically deterministic, but not necessarily so in practice. Researchers at NASA explained to us that their numerical CFD solvers are typically started with random initial values, and involve forced random restarts when diagnostics indicate that convergence is poor. Furthermore, due to the sometimes chaotic behavior of the systems, input configurations arbitrarily close to one another can fail to achieve the same estimated convergence, even after satisfying the same stopping criterion. Thus a conventional GP model without a small–distance noise process (nugget) can be a mismatch to such potentially non-smooth data. As a secondary concern, numerical stability in decomposing covariance matrices can be improved by using a small nugget term (Neal, 1997).

2.1.1 Treed Gaussian process model

Because of concerns about the inadequacy of the stationarity assumption, we propose a surrogate model that is new in the realm of sequential design of experiments: the Bayesian treed Gaussian process (treed GP) model (Gramacy and Lee, 2008a). The treed GP model extends the Bayesian treed linear model by using a GP model with linear trend independently within each region, instead of constant (Chipman et al., 1998; Denison et al., 1998) or linear (Chipman et al., 2002) models in the partitions. A process prior (Chipman et al., 1998) is placed on the tree $\mathcal{T}$, and conditional on $\mathcal{T}$, parameters for R independent GPs in regions $\{r_\nu\}_{\nu=1}^R$ are specified via a hierarchical generative model:

$$\begin{array}{lll} \mathbf{Z}_\nu|\boldsymbol{\beta}_\nu, \sigma_\nu^2, \mathbf{K}_\nu \sim N_{n_\nu}(\mathbf{F}_\nu\boldsymbol{\beta}_\nu, \sigma_\nu^2\mathbf{K}_\nu) & \boldsymbol{\beta}_0 \sim N_m(\boldsymbol{\mu}, \mathbf{B}) & \sigma_\nu^2 \sim IG(\alpha_\sigma/2, q_\sigma/2) \\ \boldsymbol{\beta}_\nu|\sigma_\nu^2, \tau_\nu^2, \mathbf{W}, \boldsymbol{\beta}_0 \sim N_m(\boldsymbol{\beta}_0, \sigma_\nu^2\tau_\nu^2\mathbf{W}) & \mathbf{W}^{-1} \sim W((\rho\mathbf{V})^{-1}, \rho) & \tau_\nu^2 \sim IG(\alpha_\tau/2, q_\tau/2) \end{array} \quad (4)$$

where $\mathbf{F}_\nu = (\mathbf{1}, \mathbf{X}_\nu)$, $\mathbf{W}$ is a $m \times m$ matrix, and $m = m_X + 1$. N, IG, and W are the (Multivariate) Normal, Inverse–Gamma, and Wishart distributions, respectively. $\mathbf{K}_\nu$ is the separable power family covariance matrix with a nugget, as in (2–3). The data $\{\mathbf{X}, \mathbf{Z}\}_\nu$ in region r_ν are used to estimate the parameters $\boldsymbol{\theta}_\nu = \{\boldsymbol{\beta}, \sigma^2, \boldsymbol{K}, \tau^2\}_\nu$ of the model active in the region. Parameters to the hierarchical priors depend only on $\{\boldsymbol{\theta}_\nu\}_{\nu=1}^R$. Samples from the posterior distribution are gathered using Markov chain Monte Carlo (MCMC). All parameters can be sampled using Gibbs steps, except for the covariance structure, whose parameters can be sampled via Metropolis–Hastings.

The predicted value of $Z(\mathbf{x} \in r_\nu)$ is normally distributed with mean and variance

$$\hat{z}(\mathbf{x}) = E(Z(\mathbf{x})|\text{ data}, \mathbf{x} \in r_\nu) \ = \ \mathbf{f}^\top(\mathbf{x})\tilde{\boldsymbol{\beta}}_\nu + \mathbf{k}_\nu(\mathbf{x})^\top\mathbf{K}_\nu^{-1}(\mathbf{Z}_\nu - \mathbf{F}_\nu\tilde{\boldsymbol{\beta}}_\nu), \quad (5)$$

$$\hat{\sigma}^2(\mathbf{x}) = \text{Var}(Z(\mathbf{x})|\text{ data}, \mathbf{x} \in r_\nu) \ = \ \sigma_\nu^2[\kappa(\mathbf{x}, \mathbf{x}) - \mathbf{q}_\nu^\top(\mathbf{x})\mathbf{C}_\nu^{-1}\mathbf{q}_\nu(\mathbf{x})], \quad (6)$$

$$\text{where} \qquad \mathbf{C}_\nu^{-1} = (\mathbf{K}_\nu + \tau_\nu^2 \mathbf{F}_\nu \mathbf{W} \mathbf{F}_\nu^\top)^{-1} \qquad\qquad \mathbf{q}_\nu(\mathbf{x}) = \mathbf{k}_\nu(\mathbf{x}) + \tau_\nu^2 \mathbf{F}_\nu \mathbf{W}_\nu \mathbf{f}(\mathbf{x}) \tag{7}$$

$$\kappa(\mathbf{x},\mathbf{y}) = K_\nu(\mathbf{x},\mathbf{y}) + \tau_\nu^2 \mathbf{f}^\top(\mathbf{x}) \mathbf{W} \mathbf{f}(\mathbf{y})$$

with $\mathbf{f}^\top(\mathbf{x}) = (1, \mathbf{x}^\top)$, and $\mathbf{k}_\nu(\mathbf{x})$ a n_ν–vector with $\mathbf{k}_{\nu,j}(\mathbf{x}) = K_\nu(\mathbf{x}, \mathbf{x}_j)$, for all $\mathbf{x}_j \in \mathbf{X}_\nu$. The global process is nonstationary because of the tree ($\mathcal{T}$) and thus $\hat{\sigma}^2(\mathbf{x})$ in (6) is region–specific. The predictive surface can be discontinuous across the partition boundaries of a particular tree $\mathcal{T}$. However, in the aggregate of samples collected from the joint posterior distribution of $\{\mathcal{T}, \boldsymbol{\theta}\}$, the mean tends to smooth out near likely partition boundaries as the tree operations *grow, prune, change, swap*, and *rotate* integrate over trees and GPs with larger posterior probability (Gramacy and Lee, 2008a). Uncertainty in the posterior for $\mathcal{T}$ translates into higher posterior predictive uncertainty near region boundaries. When the data actually indicate a non-smooth process, e.g., as in the LGBB experiment in Section 5, the treed GP retains the capability to model discontinuities.

The Bayesian treed linear model of Chipman et al. (2002) is implemented as a special case of the treed GP model, called the treed GP LLM (short for: “with jumps to the Limiting Linear Model”). Detection of linearity in the response surface is facilitated on a per-dimension basis via the introduction of m_X indicator–parameters $\mathbf{b}_\nu$, in each region $r_\nu = 1, \ldots, R$, which are given a prior conditional on the range parameter(s) to $K_\nu(\cdot, \cdot)$. The boolean $b_{\nu i}$ determines whether the GP or its LLM governs the marginal process in the i^{th} dimension of region r_ν. The result, through Bayesian model averaging, is an adaptively semiparametric nonstationary regression model which can be faster, more parsimonious, and numerically stable (Gramacy and Lee, 2008b). Empirical evidence suggests that many computer experiments involve responses which are either linear in most of the input dimensions, or entirely linear in a subset of the input domain [see Section 5]. Thus the treed GP LLM is particularly well–suited to be a surrogate model for computer experiments.

Compared to other approaches to nonstationary modeling, including using spatial deformations (Sampson and Guttorp, 1992; Schmidt and O'Hagan, 2003) and process convolutions (Higdon et al., 1999; Paciorek, 2003), the treed GP LLM approach yields an extremely fast implementation of nonstationary GPs, providing a divide–and–conquer approach to spatial modeling. Although the method is especially well–suited to axis–aligned nonstationarity, which is common in computer experiments, it has been found to compare favorably in situations when the nature of the nonstationarity is more general (Gramacy and Lee, 2008a,b). For example, in Section 4.2 we consider a dataset where the treed GP is quite appropriate even though the correlation structure varies radially, and moreover, which is well–fit by a stationary model for small designs. In Section 4.3 we consider a high dimensional dataset where the nature of the nonstationarity is unknown. Software implementing the treed GP LLM model and all of its special cases (e.g., stationary GP, CART & the treed linear model, linear model, etc.) is available as an R package (R Development Core Team, 2004), and can be obtained from CRAN:

`http://www.cran.r-project.org/web/packages/tgp/index.html`.

The package implements a family of default prior specifications, i.e., settings for the constants in (4). In this paper we use these defaults unless otherwise noted. For more details see the `tgp` documentation (Gramacy and Taddy, 2008) and tutorial (Gramacy, 2007).

2.2 Sequential design of experiments

In the statistics community, the traditional approach to sequential data solicitation is called *(Sequential) Design of Experiments* (DOE) or *Sequential Design and Analysis of Computer Experiments* (SDACE) when applied to computer simulations (Sacks et al., 1989; Currin et al., 1991; Welch et al., 1992; Santner et al., 2003). Depending on whether the goal of the experiment is inference or prediction, as described by a choice of utility, different algorithms for obtaining optimal designs can be derived. For example, one can choose the Kullback–Leibler distance between the posterior and prior distributions as a utility.

For Gaussian process models with correlation matrix $\mathbf{K}$, this is equivalent to maximizing $\det(\mathbf{K})$. Subsequently chosen input configurations are called maximum entropy designs (e.g., Shewry and Wynn, 1987; Santner et al., 2003, Chapter 6). An excellent review of Bayesian approaches to DOE is provided by Chaloner and Verdinelli (1995).

Finding optimal designs can be computationally intensive, especially for stationary GP surrogate models, because the algorithms usually involve repeated decompositions of large covariance matrices. Determinant–space, for example, can have many local maxima which can be sought–after via stochastic search, i.e., simulated annealing, genetic algorithms (Hamada et al., 2001), etc. A parametric family is assumed, either with fixed parameter values, or a preliminary analysis is used to find maximum likelihood estimates for its parameters, which are then treated as "known". In a sequential design, parameters estimated from previous designs can be used, whereas a Bayesian decision theoretic approach may "choose" a parameterization and optimal design jointly (Müller et al., 2004). In all of these approaches, it is important to note that optimality is only with respect to the assumed parametric form. Should this form not be known a priori, as is often the case in practice, then the resulting designs could be far from optimal.

Other nonparametric approaches used in the statistics literature include space filling designs, e.g., maximin distance designs and LHS (McKay et al., 1979; Santner et al., 2003). Computing maximin distance designs can also be computationally intensive, whereas LHSs are easy to compute and result in well–spaced configurations relative to random sampling, though there are some degenerate cases, such as diagonal LHSs (Santner et al., 2003). LHSs can also be less advantageous in a sequential sampling environment since there is no mechanism to ensure that the configurations will be well–spaced relative to previously sampled (fixed) locations. Maximum entropy designs, and maximin designs, may be more computationally demanding, but they are easily converted into sequential design methods by simply fixing the locations of samples whose response has already been obtained, and then optimizing only over new sample locations.

2.2.1 An active learning approach sequential experimental design

In the world of Machine learning, design of experiments would (loosely) fall under the blanket of a research focus called *active learning*. In the literature (Angluin, 1987; Atlas et al., 1990), active learning, or equivalently *query learning* or *selective sampling*, refers to the situation where a learning algorithm has some, perhaps limited, control over the inputs it trains on. There are essentially two active learning approaches to DOE using the GP. The first approach tries to maximize the information gained about model parameters by selecting from a set of candidates $\tilde{\mathbf{X}}$, the location $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ which has the greatest standard deviation in predicted output. This approach, called ALM for Active Learning–MacKay, has been shown to approximate maximum expected information designs (MacKay, 1992). Kleijnen and van Beers (2004) take a similar approach.

An alternative algorithm, called ALC for Active Learning–Cohn, is to select $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ maximizing the expected reduction in squared error averaged over the input space (Cohn, 1996). Using the notation from (1) for stationary GPs, and supposing that the location $\tilde{\mathbf{x}}$ is added into the design, a global reduction in predictive variance can be obtained by averaging over other locations $\mathbf{y}$:

$$\Delta\hat{\sigma}^2(\tilde{\mathbf{x}}) = \int_{\mathbf{y}} \Delta\hat{\sigma}^2_{\tilde{\mathbf{x}}}(\mathbf{y}) = \int_{\mathbf{y}} \hat{\sigma}^2(\mathbf{y}) - \hat{\sigma}^2_{\tilde{\mathbf{x}}}(\mathbf{y}) = \int_{\mathbf{y}} \frac{\sigma^2 \left[\mathbf{k}^\top(\mathbf{y})\mathbf{K}_N^{-1}\mathbf{k}(\tilde{\mathbf{x}}) - K(\tilde{\mathbf{x}},\mathbf{y})\right]^2}{K(\tilde{\mathbf{x}},\tilde{\mathbf{x}}) - \mathbf{k}(\tilde{\mathbf{x}})^\top\mathbf{K}_N^{-1}\mathbf{k}(\tilde{\mathbf{x}})}. \tag{8}$$

In practice the integral is replaced by a sum over a grid of locations $\tilde{\mathbf{Y}}$, typically with $\tilde{\mathbf{Y}} = \tilde{\mathbf{X}}$, and the parameterization to the model, i.e., $K(\cdot,\cdot)$ and σ^2, is assumed known in advance. Seo et al. (2000) provide a comparison for stationary GPs between ALC and ALM.

2.2.2 Other approaches to designing computer experiments

Bayesian and non-Bayesian approaches to surrogate modeling and design for computer experiments abound (Sacks et al., 1989; Currin et al., 1991; Welch et al., 1992; Bates et al., 1996; Sebastiani and Wynn, 2000; Kennedy and O'Hagan, 2000, 2001). A recent approach, which bears some similarity to ours, uses stationary GPs and a so–called *spatial aggregate language* to aid in an *active data mining* of the input space of the experiment (Ramakrishnan et al., 2005). Our use of nonstationary surrogate models within a highly distributed supercomputer architecture distinguishes our work from the methods described in those papers, and yields a more dynamic approach to sequential design.

3 Adaptive sequential design

Much of the current work in large scale computer models starts by evaluating the model over a hand crafted set of input configurations, such as a full grid or some reduced design. After the initial set has been run, a human may identify interesting regions and perform additional runs if desired. We are concerned with automating this process, based on local estimates of uncertainty that can provided by the nonstationary treed GP LLM surrogate model.[1]

3.1 Asynchronous distributed supercomputing

High fidelity supercomputer experiments are usually run on clusters of independent computing agents, or processors. A `Beowulf` cluster is a good example. At any given time, each agent is working on a single input configuration. Multiple agents allow several input configurations to be run in parallel. Simulations for new configurations begin when an agent finishes execution and becomes available. Therefore, simulations may start and finish at different, perhaps even random, times. The cluster is managed asynchronously by a master controller *(emcee)* program that gathers responses from finished simulations, and supplies free agents

[1] We shall drop the LLM tag in what follows, and consider it implied by the label treed GP.

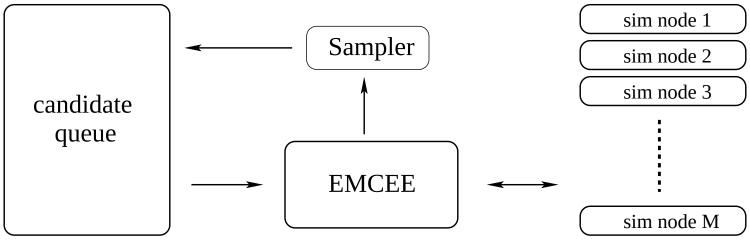

Figure 2: *Emcee* program gives finished simulations to the sampler (which populates the queue based on a surrogate model) and gets new ones from the queue.

with new input configurations. The goal is to have the *emcee* program interact with a non-stationary modeling and sequential design program that maintains a queue of well–chosen candidates, and to which it provides finished responses as they become available, so that the surrogate model can be updated [see Figure 2].

3.2 Adaptive sequential DOE via active learning

In the statistics community, there are a number of established methodologies for (sequentially) designing experiments [see Section 2.2]. However, some classic criticisms for traditional DOE approaches precluded such an approach here. The primary issue is that "optimally" chosen design points are usually along the boundary of the region, where measurement error can be severe, responses can be difficult to elicit, and model checking is often not feasible (Chaloner and Verdinelli, 1995). Furthermore, boundary points are only optimal when the model is known precisely. For example, in one–dimensional linear regression, putting half the points at one boundary and half at the other is only optimal if the true model is linear; if it turns out that the truth may be quadratic, then forcing all points to the boundaries is highly suboptimal. Similarly here, where we do not know the full form of the model in advance, it is important to favor internal points so that the model (including the partitions) can be learned correctly. Other drawbacks to the traditional DOE approach include speed, the difficulty inherent in using Monte Carlo to estimate the surrogate model, lack of support for partition models, and the desire to design for an asynchronous *emcee* interface where responses and computing nodes become available at random times.

Instead, we take a two-stage (hybrid) approach that combines standard DOE with methods from the active learning literature. The first stage is to use optimal sequential designs from the DOE literature, such as maximum entropy, maximin designs, or LHS, as *candidates* for future sampling. This ensures that candidates for future sampling are well–spaced out relative to themselves, and to the already sampled locations. In the second stage, the treed GP surrogate model can provide Monte Carlo estimates of region–specific model uncertainty, via the ALM or ALC algorithm, which can be used to populate, and sequence, the candidate queue used by the *emcee* [see Figure 2]. This ensures that the most informative of the optimally spaced candidates can be first in line for simulation when agents become available.

3.3 ALM and ALC algorithms

Given a set of candidate input configurations $\tilde{\mathbf{X}}$, Section 2.2.1 introduced two active learning criteria for choosing amongst—or ordering—them based on the posterior predictive distribution. ALM chooses the $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ with the greatest standard deviation in predicted output (MacKay, 1992). MCMC posterior predictive samples provide a convenient estimate of location–specific variance, namely the width of predictive quantiles.

Alternatively, ALC selects the $\tilde{\mathbf{x}}$ that maximizes the expected reduction in squared error averaged over the input space (Cohn, 1996). Conditioning on $\mathcal{T}$, the reduction in variance at a point $\mathbf{y} \in r_\nu$, given that the location $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}_\nu$ is added into the data, is defined as (region subscripts suppressed):

$$\begin{aligned} \Delta\hat{\sigma}^2_{\tilde{\mathbf{x}}}(\mathbf{y}) = \hat{\sigma}^2(\mathbf{y}) - \hat{\sigma}^2_{\tilde{\mathbf{x}}}(\mathbf{y}) \qquad \text{where} \quad & \hat{\sigma}^2(\mathbf{y}) = \sigma^2[\kappa(\mathbf{y},\mathbf{y}) - \mathbf{q}_N^\top(\mathbf{y})\mathbf{C}_N^{-1}\mathbf{q}_N^\top(\mathbf{y})], \\ \text{and} \quad & \hat{\sigma}^2_{\tilde{\mathbf{x}}}(\mathbf{y}) = \sigma^2[\kappa(\mathbf{y},\mathbf{y}) - \mathbf{q}_{N+1}^\top(\mathbf{y})\mathbf{C}_{N+1}^{-1}\mathbf{q}_{N+1}(\mathbf{y})] \end{aligned}$$

using notation for the GP predictive variance for region r_ν given in (6). Note that the $N+1^{\text{st}}$ component of $\mathbf{q}_{N+1}(\mathbf{y})$, and the corresponding column and row of $\mathbf{C}_{N+1}$, are a function of

$\tilde{\mathbf{x}}$. The partition inverse equations (Barnett, 1979), for a covariance matrix $\mathbf{C}_{N+1}$ in terms of $\mathbf{C}_N$, yield:

$$\Delta\hat{\sigma}^2_{\tilde{\mathbf{x}}}(\mathbf{y}) = \frac{\sigma^2\left[\mathbf{q}_N^\top(\mathbf{y})\mathbf{C}_N^{-1}\mathbf{q}_N(\tilde{\mathbf{x}}) - \kappa(\tilde{\mathbf{x}},\mathbf{y})\right]^2}{\kappa(\tilde{\mathbf{x}},\tilde{\mathbf{x}}) - \mathbf{q}_N^\top(\tilde{\mathbf{x}})\mathbf{C}_N^{-1}\mathbf{q}_N(\tilde{\mathbf{x}})}. \tag{9}$$

The details of this derivation are included in Appendix A.1. For $\mathbf{y}$ and $\tilde{\mathbf{x}}$ not in the same region r_ν, let $\Delta\sigma^2_{\tilde{\mathbf{x}}}(\mathbf{y}) = 0$. Rather than integrating, as in (8), the reduction in predictive variance that would be obtained by adding $\tilde{\mathbf{x}}$ into the dataset is calculated in practice by averaging over a grid or candidate set of $\mathbf{y} \in \mathbf{Y}$:

$$\Delta\sigma^2(\tilde{\mathbf{x}}) = |\mathbf{Y}|^{-1}\sum_{\mathbf{y}\in\mathbf{Y}} \Delta\hat{\sigma}^2_{\tilde{\mathbf{x}}}(\mathbf{y}) \tag{10}$$

which can be approximated using MCMC methods. Compared to ALM, adaptive samples under ALC are less heavily concentrated near the boundaries of the partitions. Both provide a ranking of a set of candidate locations $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$. Computational demands are in $O(|\tilde{\mathbf{X}}|)$ for ALM, and $O(|\tilde{\mathbf{X}}||\mathbf{Y}|)$ for ALC. McKay et al. (1979) provide a comparison between ALM, and LHS, on computer code data. Seo et al. (2000) provide comparisons between ALC and ALM using standard GPs, taking $\mathbf{Y} = \tilde{\mathbf{X}}$ to be the full set of un-sampled locations in a pre-specified dense uniform grid. In both papers, the model is assumed known in advance.

However, that last assumption, that the model is known *a priori* is at loggerheads with sequential design—if the model were already known then why design sequentially? In the treed GP application of ALC, the model is not assumed known *a priori*. Instead, Bayesian MCMC posterior inference on $\{\mathcal{T}, \boldsymbol{\theta}\}$ is performed, and then samples from $\Delta\sigma^2_{\tilde{\mathbf{x}}}(\mathbf{y})$ are taken conditional on samples from $\{\mathcal{T}, \boldsymbol{\theta}\}$. To mitigate the (possibly enormous) expense of sampling $\Delta\sigma^2_{\tilde{\mathbf{x}}}(\mathbf{y})$ via MCMC on a dense high–dimensional grid (with $\mathbf{Y} = \tilde{\mathbf{X}}$), a smaller and more cleverly–chosen set of candidates can come from the sequential treed maximum entropy design, described in the following subsection. The idea is to sequentially select candidates which are well–spaced relative both to themselves and to the already sampled configurations, in order to encourage exploration.

Applying the ALC algorithm under the limiting linear model (LLM) is computationally less intensive compared to ALC under a full GP. Starting with the predictive variance given in (7), the expected reduction in variance under the linear model is given in (11), below, and averaging over $\mathbf{y}$ proceeds as in (10), above.

$$\Delta\hat{\sigma}^2_{\tilde{\mathbf{x}}}(\mathbf{y}) = \frac{\sigma^2[\mathbf{f}^\top(\mathbf{y})\mathbf{V}_{\tilde{\beta}_N}\mathbf{f}(\tilde{\mathbf{x}})]^2}{1 + g + \mathbf{f}^\top(\tilde{\mathbf{x}})\mathbf{V}_{\tilde{\beta}_N}\mathbf{f}(\tilde{\mathbf{x}})} \tag{11}$$

Appendix A.2 contains details of the derivation. The $m \times m$ matrix $\mathbf{V}_{\tilde{\beta}_N}$ is the posterior variance of $\boldsymbol{\beta}$ based on the N data points in the current design. Since only an $O(m^3)$ inverse operation is required, Eq. (11) is preferred over replacing $\mathbf{K}$ with the $N \times N$ matrix $\mathbf{I}(1+g)$ in (9), which requires an $O(N^3)$ inverse.

3.4 Choosing candidates

We have already discussed how a large, i.e., densely gridded, candidate set $\tilde{\mathbf{X}}$ can make for computationally expensive ALM and (especially) ALC calculations. In an asynchronous parallel environment, there is another reason why candidate designs should not be too dense. Suppose we are using the ALM algorithm, and we estimate the uncertainty to be highest in a particular region of the space. If two candidates are close to each other in this region, then they will have the highest and second–highest priority, and the emcee could send both of them to agents. However, if we knew we were going to send off two runs, we generally would not want to pick those two right next to each other, but would want to pick two points from different parts of the space. If each design point could be picked sequentially, then the candidate spacing is not an issue, because the model can be re-fit and the points re-ordered between runs. In the reality of an asynchronous parallel environment, there may not be time to re-fit the model before the emcee needs an additional run configuration to send to another agent. Thus there is a real need for well–spaced candidates.

A sequential maximum entropy design (Shewry and Wynn, 1987; Sacks et al., 1989; Currin et al., 1991; Welch et al., 1992; Santner et al., 2003, Chapter 6) may seem like a reasonable approach because it encourages exploration. But traditional maximum entropy designs are based on a known parameterization of a single GP model, and are thus not well–suited to MCMC based treed partition models wherein "closeness" is not measured homogeneously throughout the input space. Furthermore, a maximum entropy design may not choose candidates in the "interesting" part of the input space because sampling is high there already. E.g., in the rocket booster application we want to continue to sample close to Mach one because we need many more points to understand the function where it is changing quickly. Another disadvantage to maximum entropy designs is computational, requiring repeated decompositions of large covariance matrices.

One possible solution to both computational and nonstationary modeling issues is to use what we call a (sequential) treed maximum entropy design. That is, a separate sequential maximum entropy design can be obtained in each of the partitions depicted by the maximum *a posteriori* (MAP) tree $\hat{\mathcal{T}}$. The number of candidates selected from each region, $\{\hat{r}_\nu\}_{\nu=1}^{\hat{R}}$ of $\hat{\mathcal{T}}$, can be proportional to the volume or the number of grid locations in the region. MAP parameters $\hat{\boldsymbol{\theta}}_\nu|\hat{\mathcal{T}}$ can be used in creating the candidate design, or "neutral" or "exploration encouraging" parameters can be used instead. Separating design from inference by using custom parameterizations in design steps, rather than inferred ones, is a common practice in the SDACE community (Santner et al., 2003). Small range parameters, for learning about the wiggliness of the response, and a modest nugget parameter for numerical stability, tend to work well together.

Since optimal design is only used to select candidates, and is not the final step in adaptively choosing samples, employing a high-powered search algorithm (e.g., a genetic algorithm) is unnecessary. Finding a local maximum is generally sufficient to get well–spaced candidates. We use a simple stochastic ascent algorithm[2] in each of the $\hat{R}$ regions $\{r_\nu\}_{\nu=1}^{\hat{R}}$ of $\hat{\mathcal{T}}$ to find local maxima without calculating too many determinants. The $\hat{R}$ search algorithms can be run in parallel, and typically invert matrices much smaller than $N \times N$.

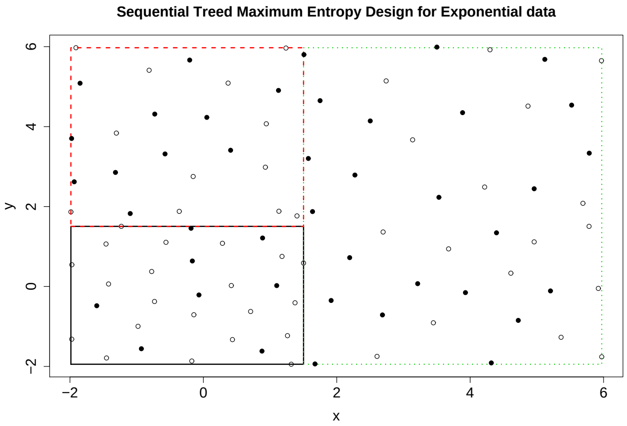

Figure 3: Example of a treed maximum entropy design in 2-d. *Open Circles* represent previously sampled locations. *Solid dots* are the candidate design based on $\hat{\mathcal{T}}$, also shown.

Figure 3 shows an example sequential treed maximum entropy design for the 2-d Exponential data [Section 4.2], found by simple stochastic search. Input configurations are sub-sampled from a LHS of size 400, and the chosen candidate design is of size ∼40 (i.e., $\lceil 10\% \rceil$). Dots in the figure represent the chosen locations of the new candidate design $\tilde{\mathbf{X}}$ relative to the existing sampled locations $\mathbf{X}$ (circles). Candidates are reasonably spaced–out relative to one another, and to existing inputs, except possibly near partition boundaries. There are roughly the same number of candidates in each quadrant, despite the fact that the density of samples (circles) in the first quadrant is almost two-times that of the others. A classical (non-treed) maximum entropy design would have chosen fewer points in the first quadrant, where all the action is, in order to equalize the density relative to the other three quadrants.

[2] For example, we find that the following works well for constructing a set of candidates $\tilde{\mathbf{X}}$ of size $|\tilde{\mathbf{X}}| = N'$. Construct a LHS $\mathbf{L}$ of size $10N'$, and initialize $\tilde{\mathbf{X}}$ to be a random subsample of $\mathbf{L}$ of size N', without replacement. Then randomly propose to swap a single element of $\tilde{\mathbf{X}}$ with one from $\mathbf{L} \setminus \tilde{\mathbf{X}}$ and accept only upon an observed increase in $\det(\mathbf{K}([\mathbf{X}, \tilde{\mathbf{X}}]))$. Repeat until the acceptance rate is low, possibly with reference to N'.

3.5 Implementation methodology

Bayesian adaptive sampling (BAS) proceeds in trials. Suppose that N samples and their responses have been gathered in previous trials, or from a small initial design before the first trial. In the current trial, a treed GP model is estimated for data $\{\mathbf{x}_i^\top, z_i\}_{i=1}^N$. Samples are gathered in accordance with ALM or ALC conditional on $\{\boldsymbol{\theta}, \mathcal{T}\}$, at candidate locations $\tilde{\mathbf{X}}$ chosen to follow a sequential treed maximum entropy design, using the MAP tree obtained in the previous trial. The candidate queue is then populated with a sorted list of candidates. BAS gathers finished and running input configurations from the *emcee* and adds them into the design. Predictive mean estimates are used as surrogate responses for unfinished (running) configurations until the true response is available. New trials start with fresh candidates.

An artificial clustered simulation environment, with a fixed number of agents, was developed in order to simulate the parallel and asynchronous evaluation of input configurations, whose responses finish at random times. It was implemented `Perl` and was designed to mimic, and interface with, the `Perl` modules at NASA which drive their experimental design software. Experiments on synthetic data, in the next section, will use this artificial environment. The LGBB experiment in Section 5 uses the real `Perl` modules to submit jobs to the real NASA supercomputer. Multi-dimensional responses, as in the LGBB experiment, are treated as independent. That is, each response has its own treed GP surrogate model, m_Z surrogates total for an m_Z–dimensional response. Uncertainty estimates (via ALM or ALC) are normalized and pooled across the models for each response in order to develop a single (sequential) design for the entire process. Treating highly correlated physical measurements as independent is a crude approach. However, it still affords remarkable results, and allows the use of the `PThreads` parallel computing library to get a highly parallel implementation and take advantage of multi-core processors that are becoming commonplace. Coupled with the producer/consumer model for parallelizing prediction and estimation (Gramacy and Lee, 2008a), a factor of $2m_Z$ speedup for $2m_Z$ processors can be obtained.[3] Cokriging (Ver Hoef and Barry, 1998), co-regionalization (Schmidt and Gelfand, 2003), and other approaches to modeling multivariate responses are obvious extensions, but lie beyond the scope of the present work, and are not easily parallelizable. The MAP tree $\hat{\mathcal{T}}$, used for creating sequential treed maximum entropy candidates, is taken from the treed GP surrogates of each of the m_Z responses in turn.

Chipman et al. (1998) recommend running several parallel chains, and sub-sampling from all chains in order better explore the posterior distribution of the tree ($\mathcal{T}$). Rather than run multiple chains explicitly, the trial nature of adaptive sampling can be exploited: at the beginning of each trial the tree can be randomly pruned back. Although the tree chain associated with an individual trial may find itself stuck in a local mode of the posterior, in the aggregate of all trials the chain(s) explore the posterior of tree–space nicely. Random pruning represents a compromise between restarting and initializing the tree at a well–chosen starting place. This *tree inertia* usually affords shorter burn–in of the MCMC at the beginning of each trial. The tree can also be initialized with a run of the Bayesian treed LM, for a faster burn–in of the treed GP chain.

Each trial executes at least B burn–in and T total MCMC sampling rounds. Samples are saved every E rounds in order to reduce the correlation between draws by thinning. Samples of ALM and ALC statistics need only be gathered every E rounds, so thinning cuts down on the computational burden as well. If the *emcee* has no responses waiting to be incorporated by BAS at the end of T MCMC rounds, then BAS can run more MCMC rounds, either continuing where it left off, or after re-starting the tree (but saving all samples from each chain). New trials, with new candidates, start only when the *emcee* is ready with a new finished response. Such is the design so that the computing time of each BAS trial does not affect the rate of sampling. Rather, a slow BAS runs fewer MCMC rounds per

[3] For more information on the parallel implementation, please see Appendix C.2 of Gramacy (2007) or the `tgp` package vignette.

finished response, and re-sorts candidates less often compared to a faster BAS. A slower adaptive sampler yields less optimal sequential samples, but still offers an improvement over naïve gridding. In the experiments that follow in the next two sections, the MCMC for the surrogate model was run with $B = 2000$, $T = 7000$ and $E = 2$.

4 Illustrative examples

In this section, sequential designs are built for three synthetic datasets with the treed GP LLM as a surrogate model. The supercomputer was simulated as having five independent nodes which could provide responses for inputs in a time of 20 seconds plus a random number of seconds having a Poisson distribution with mean 20. The examples in this section use the default prior specification provided by the `tgp` package, which involves using an improper prior for $\boldsymbol{\beta}$ obtained by fixing $\boldsymbol{\beta}_0 = \mathbf{0}$ and $\tau^2 = \infty$ in Eq. (4) of Section 2.1.1.

4.1 1-d Synthetic Sinusoidal data

Consider some synthetic sinusoidal data first used by Higdon (2002), and then augmented by Gramacy and Lee (2008a) to contain a linear region:

$$z(x) = \begin{cases} \sin\left(\frac{\pi x}{5}\right) + \frac{1}{5}\cos\left(\frac{4\pi x}{5}\right) & x < 10 \\ x/10 - 1 & \text{otherwise,} \end{cases} \tag{12}$$

observed with $N(0, \sigma = 0.1)$ noise. Figure 4 shows three snap shots, illustrating the evolution of BAS on this data using the ALC algorithm with sequential treed maximum entropy candidates. The first column shows the estimated surface in terms of posterior predictive means (solid-black) and 90% intervals (dashed–red). The MAP tree $\hat{\mathcal{T}}$ is shown as well. The second column summarizes the ALM and ALC statistics (scaled to show alongside ALM) for comparison. Ten samples from a sequential maximum entropy design were used to start things off, and twenty candidates from a treed maximum entropy design were proposed during each adaptive sampling round.

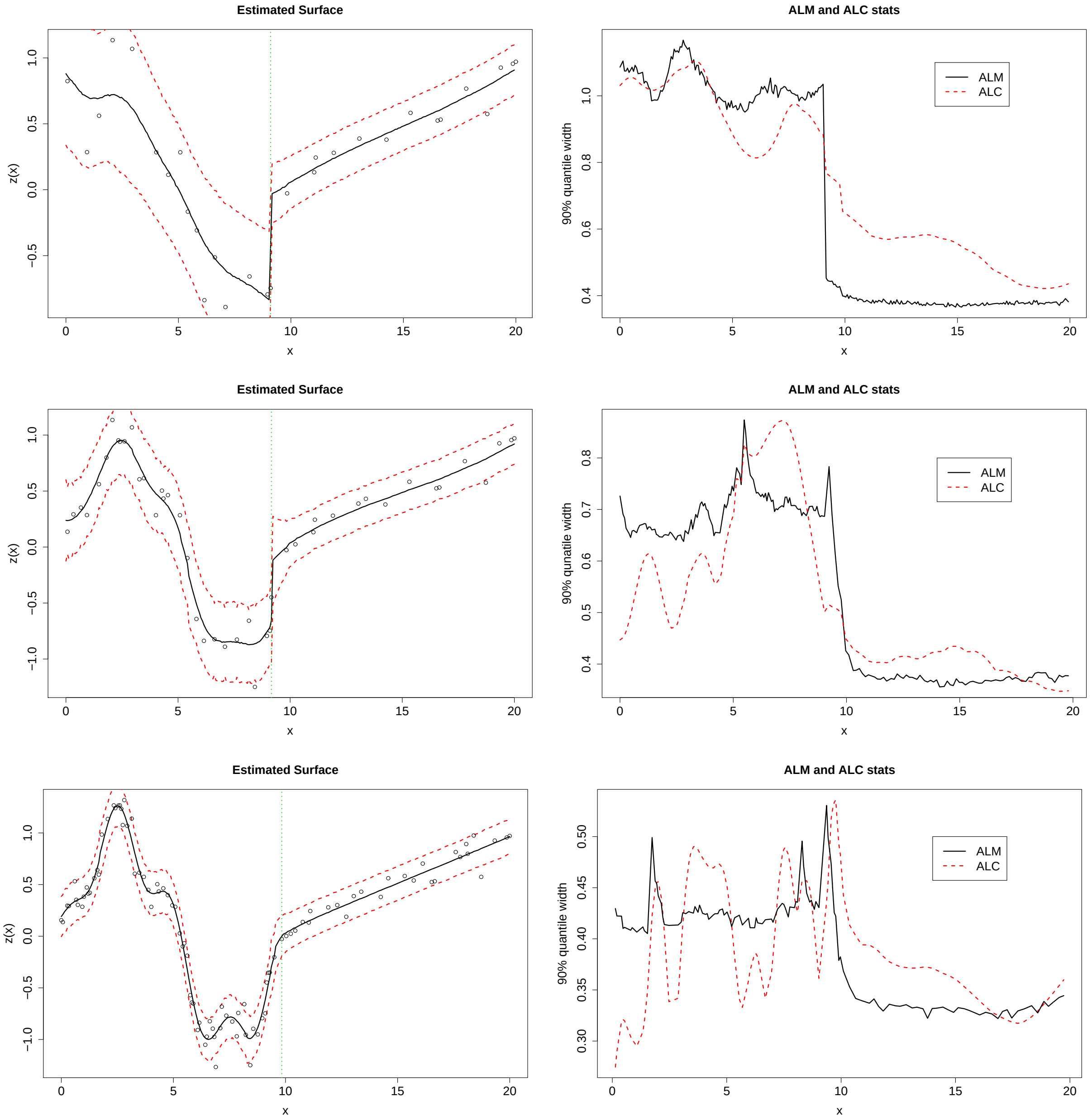

Figure 4: Sine data after 30 *(top)*, 45 *(middle)*, and 97 *(bottom)* adaptively chosen samples. *Left:* posterior predictive mean and 90% quantiles, and MAP partition $\hat{\mathcal{T}}$. *Right:* ALM (black-solid) and ALC (red-dashed).

The snapshot in the top row of Figure 4 was taken after BAS had gathered a total of 30 samples, having learned that there is probably one partition near $x = 10$, with roughly the same number of samples on each side. Predictive uncertainty (under both ALM and

ALC) is higher on the left side than on the right. ALM and ALC are in relative agreement, however the transition of ALC over the partition boundary is more smooth. The ALM statistics are "noisier" than ALC because the former is based on quantiles, and the latter on averages (10). Although both ALM and ALC are shown, only ALC was used to select adaptive samples. The middle row of Figure 4 shows a snapshot taken after 45 samples were gathered, where we can see that BAS has sampled more heavily in the sinusoidal region (by a factor of two), and learned a great deal. ALM and ALC are in less agreement here than in the row above. Also, ALC is far less concerned with uncertainty near the partition boundary, than it is, say, near $x = 7$. Finally, the snapshot in the bottom row of Figure 4 was taken after 97 samples had been gathered. By now, BAS has learned about the secondary cosine structure in the *left–hand* region. It has focused almost three times more of its sampling effort there. ALM and ALC agree that there is high uncertainty near the partition boundary ($\hat{\mathcal{T}}$), but otherwise disagree about where to sample next. Any further sampling would yield only marginal improvements since this final surface, in the *bottom–left* panel, is a very good approximation to the truth.

In summary, the left panels of Figure 4 track the treed GP model's improvements in its ability to predict the mean, via the increase in resolution from one figure to the next. From the scale of y-axes in the right column one can see that as more samples are gathered, the variance in the posterior predictive distribution of the treed GP decreases as well. Despite the disagreements between ALM and ALC on individual iterations during the evolution of BAS, it is interesting to note that difference between using ALC and ALM on this dataset is negligible. This is likely due to the high quality of the candidates $\tilde{\mathbf{X}}$, from a sequential treed maximum entropy design, which prevent the clumping behavior that tends to hurt ALM, but to which ALC is somewhat less prone.

Perhaps the best illustration of how BAS learns and adapts over time is to compare it to something that is, ostensibly, less adaptive. Gramacy et al. (2004) show how the mean-squared error (MSE) of BAS evolves over time on a similar dataset, but in a serial setting where only one sample is taken at a time, and the surrogate model is allowed to re-fit before the next (single) adaptive sample is chosen. They show how the MSE of BAS decreases steadily as samples are added, despite that fewer points are added in the linear region, yielding a sequential design strategy which is two times more efficient than LHS. They also show how BAS measures up against ALM and ALC, as implemented by Seo et al. (2000)—with a stationary GP surrogate model. Seo et al. make the very powerful assumption that the correct covariance structure is known at the start of sampling. Thus, the model need not be updated in light of new responses. Alternatively, BAS quickly gathers enough samples to *learn* the partitioned covariance structure, after which it outperforms ALM and ALC based on a stationary model.

4.2 2-d Synthetic Exponential data

The nonstationary treed GP surrogate model has an even greater impact on adaptive sampling in a higher dimensional input space. For an illustration, consider the domain $[-2, 6] \times [-2, 6]$ wherein the true response is given by $z(\mathbf{x}) = x_1 \exp(-x_1^2 - x_2^2)$, observed with $N(0, \sigma = 0.001)$ noise. We take an initial set of 16 configurations from a maximum entropy design, and twenty new candidates (from a sequential treed maximum entropy design) are used in each adaptive sampling round. The top row of Figure 5 shows a snapshot after 30 adaptive samples have been gathered with BAS under the ALC algorithm. Room for improvement is evident in the mean predictive surface (*left* column). The second column shows the ALC surface, and the single partition of $\hat{\mathcal{T}}$, with samples evenly split between the two regions. Observe in the ALC plot that the model also considers a tree with a single split along the other axis, indicating good mixing of the reversible jump Markov chain in tree space.

After 50 adaptive samples have been selected (*second–row* of Figure 5), the situation is

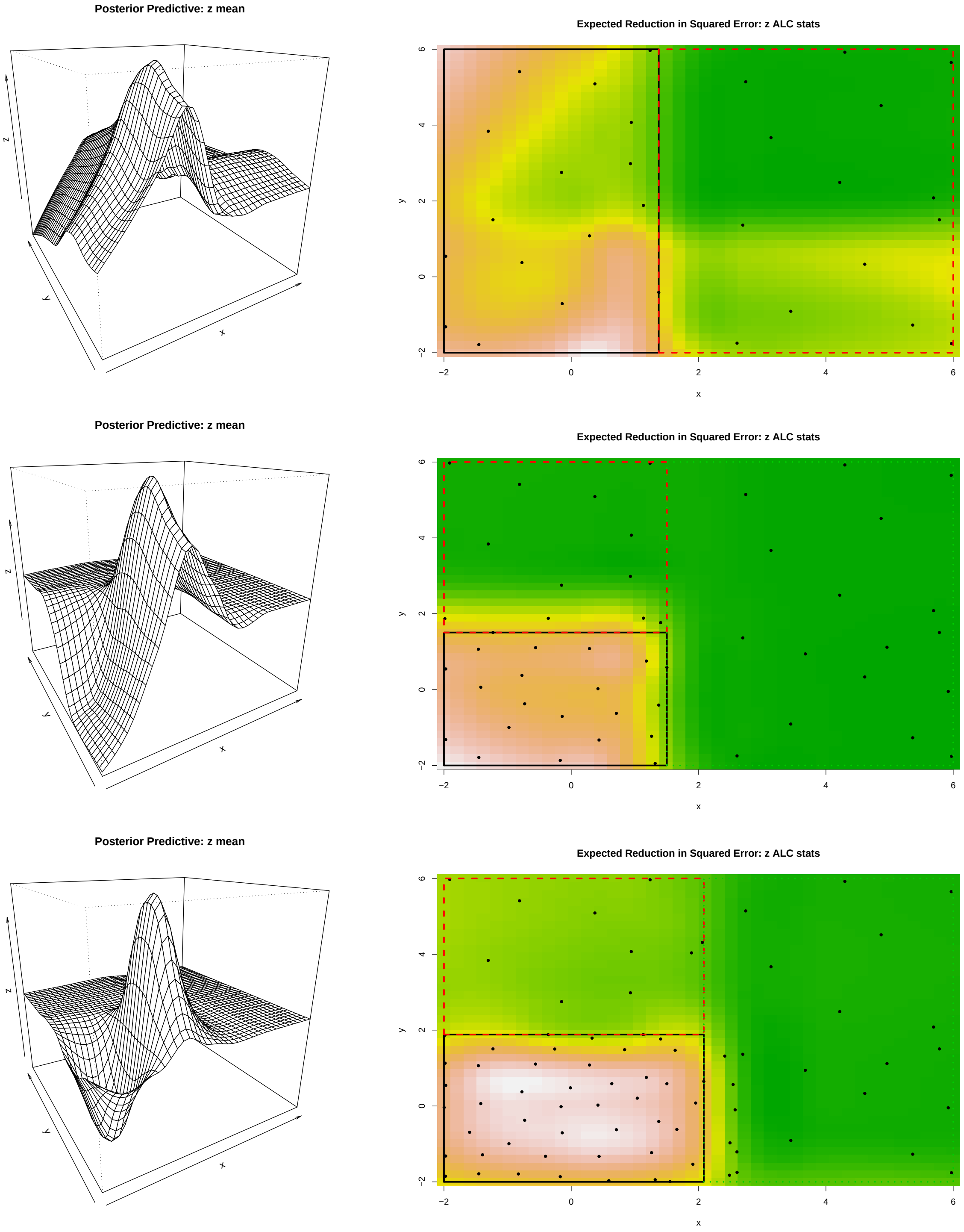

Figure 5: Exponential data after 30 *(top)*, 50 *(middle)* and 80 *(bottom)* adaptively chosen samples. *Left:* posterior predictive mean surface; *Right:* ALC criterion surface, with MAP tree $\hat{\mathcal{T}}$ and samples X overlayed.

greatly improved. ALC asserts that the first quadrant is most interesting, and as a result (adaptive) sampling is higher in this region. The dots in Figure 3 illustrate the candidates from a sequential treed maximum entropy design used during this round. [Notice that the **X** locations (circles) in Figure 3 match the dots in the center row of Figure 5.] Finally, the bottom row of Figure 5 shows the snapshot taken after 80 adaptive samples. More than 54% of the samples are located in the first quadrant which occupies only 25% of the total input space. As before, the final surface shown in the *bottom–left* of the figure is a very good approximation to the truth.

When comparing to LHS, ALM, and ALC with stationary GPs, much the same can be said here as with the sinusoidal data. Gramacy et al. (2004) show that the MSE of BAS decreases steadily as samples are added in a serial fashion, despite that most of the sampling occurs in the first quadrant, and that it is at least two–times more efficient than LHS. Crucially, the exponential data are not defined by step functions, in contrast with the sinusoidal data. Transitions between partitions are smooth. Thus it takes BAS longer to learn about $\mathcal{T}$—which in this case can be thought of as a design tool rather than a model assumption (since the data are well fit by a stationary GP)—and the corresponding three GP models in each region of $\hat{\mathcal{T}}$. Once it does however (after about 50 samples) BAS outperforms the (in hindsight) well–parameterized stationary model with ALM.

To highlight the benefits of using a treed model in sequential design, we consider a deeper comparison on this data with results summarized in Table 1. The comparison involves combinations of five models: Bayesian CART, treed linear models, (stationary) GP, treed GP, and treed GP LLM; three ways of generating AS candidates: LHS, maximum entropy and treed maximum entropy; and two adaptive sampling heuristics: ALC and ALM. The table shows RMSE to the truth as evaluated on a dense grid, for 30 repeated BAS runs each starting with a random initial maximum entropy design of 20 configurations, and then 55 samples chosen adaptively (for 75 total). For fairness, the final RMSE calculation (in each

model	cands	as	rmse
btgp	lh	alc	0.00346
btgp	lh	alm	0.00365
btgpllm	lh	alm	0.00366
bcart	tme	alc	0.00430
btgpllm	lh	alc	0.00459
bcart	tme	alm	0.00531
btgp	tme	alm	0.00561
btgpllm	tme	alm	0.00678
btgpllm	tme	alc	0.00722
btgp	tme	alc	0.00765
btlm	tme	alm	0.00874
bcart	lh	alm	0.00903
bcart	lh	alc	0.00934
btlm	lh	alm	0.00989
btlm	lh	alc	0.01148
btlm	tme	alc	0.01572
bgp	lh	alm	0.03519
btgp	me	alc	0.03747
bcart	me	alm	0.03885
bcart	me	alc	0.04002
btgpllm	me	alc	0.04028
btgp	me	alm	0.04122
btgpllm	me	alm	0.04245
btlm	me	alc	0.04269
btlm	me	alm	0.04344
bgp	lh	alc	0.04929
bgp	me	alm	0.05090
bgp	me	alc	0.05553

Table 1: Comparing a combination of five models: Bayesian CART, treed linear models, (stationary) GP, treed GP, and treed GP LLM (labeled bcart, btlm, bgp, btgp, btgpllm); three ways of generating AS candidates: LHS, sequential maximum entropy and sequential treed maximum entropy (lh, me, tme); and two AS heuristics ALC and ALM, in terms of RMSE to the truth. Observe that the bgp/tme combination is not run since the stationary GP model does not provide a MAP tree as is required for sequential treed maximum entropy design. Therefore there are 28 combinations instead of 30.

case) is based on the predictive means sampled from a full treed GP LLM model on a dense grid of predictive locations, regardless of the method used for sequential design. The table is sorted on the fourth column (RMSE). Somewhat surprisingly, Bayesian CART does really well if its candidates come from a sequential treed maximum entropy design.[4] Also, ALM and ALC perform about equally as well as one another on this data, though we suspect that ALC would do better than ALM if the data were heteroskedastic, in which case ALM would concentrate samples in the high noise region even if the mean in that region is tame. Finally, LHS candidates do better than ones from a sequential maximum entropy design (with the notable exception of Bayesian CART). The non-treed (stationary) GP model and non-treed

[4] Note that the using the full treed GP LLM for the RMSE calculation is crucial here. Had Bayesian CART been used instead it would have been ranked much lower.

maximum entropy designs are the worst in the study. That the treed GP does better than the treed GP LLM is perhaps to be expected as there is nothing at all linear about this dataset.

4.3 Six-dimensional example

As an example of a higher-dimensional problem, we present a 6-d example, with true response

$$z(x_1, x_2, x_3, x_4, x_5, x_6) = \exp\left\{\sin\left([0.9 * (x_1 + 0.48)]^{10}\right)\right\} + x_2 x_3 + x_4 \,. \tag{13}$$

This function has four active variables. It is of continuously varying wiggliness in the first dimension where the smoothness varies over the space without any natural threshold. The treed GP will usually partition on this dimension, typically somewhere between 0.6 and 0.85, which will allow more of the adaptive sampling effort to be put on the more quickly oscillating part near $x_1 = 1$. The response is smooth but non-linear in the second and third dimensions, and linear in the fourth dimension. The final two variables are pure noise, which the treed GP will need to learn about adaptively.

method	Avg(rmse)	SE(rmse)
btgpllm-linburn/tme/alc	0.02871943	0.0006446596
btgpllm/tme/alc	0.03217273	0.0037997994
no adaptive sampling	0.03598135	0.0034438090

Table 2: RMSE to the truth as evaluated on random LHSs of size 1000 in $[0,1]^6$, summarized for 10 repeated AS runs on the 6-d example.

Our experiment allows the inputs to vary in $[0,1]^6$ and the response in (13) is observed with $N(0, \sigma = 0.05)$ noise. We used a similar artificial clustered simulation environment to the one described in Section 3.5. At any time there are five nodes available to evaluate responses, which finish in no sooner than 3 minutes, plus a random number of seconds distributed as Pois(180). Table 2 shows average RMSE to the truth as evaluated on 10 random LHSs of size 1000 in $[0,1]^6$ (with standard errors), for 10 repeated BAS runs. Each run starts with a random initial set of 400 configurations from a maximum entropy design, and then 400 samples are chosen adaptively. We compare the Bayesian treed GP LLM model with ALC, both with and without LM burn–in, to a non-adaptive maximum entropy design of size 800. A stationary (non-treed) GP model was excluded from the comparison due to time constraints. As before, we use the full treed GP LLM model to calculate RMSEs after the adaptive sampling run(s), for fairness. Though there is some variation across the 10 runs, the RMSE values obtained in each run were always row–ordered as they are, in the table, with the non-adaptive method coming last, and the treed GP LLM version with linear burn–in and ALC coming first. That is, although the differences between the average RMSEs in the table appear to be modest, the improvement obtained by BAS is statistically significant.

5 LGBB CFD experiment

The final experiment is our motivating example, a computational fluid dynamics simulator of a proposed reusable NASA launch vehicle, called the Langley Glide–Back Booster (LGBB). Three input parameters are varied (side slip angle, speed, and angle of attack), and for each setting of the input parameters, six outputs (lift, drag, pitch, side-force, yaw, and roll) are monitored. All six responses are computed simultaneously. In a previous experiment, a supercomputer interface was used to launch runs at over 3,250 input configurations in several hand–crafted batches. Figure 1 plots the resulting lift response as a function of Mach (speed) and alpha (angle of attack), with beta (side-slip angle) fixed to zero. A more detailed description of this system and its results are provided by Rogers et al. (2003). Some preliminary adaptive sampling of this data appeared in Gramacy et al. (2004), although that paper dealt with only a single output, considered only one-at-a-time updates, involved only a simulation of a computer experiment, and only resampled points from the hand–crafted initial run of 3,250. Here we describe the development of a live sequential design in the fully asynchronous NASA environment. In a separate, non-adaptive, analysis of this data Gramacy and Lee (2008a) noticed that the noise structure was heteroskedastic, so we have chosen to use the ALC statistic to guide the adaptive sampling.

BAS for the LGBB is illustrated pictorially by the remaining figures in this section. The experiment was implemented on the NASA supercomputer `Columbia`—a fast and highly parallelized architecture, but with an extremely variable workload. The *emcee* algorithm of Section 3.1 was designed to interface with `AeroDB`, a database queuing system used by NASA to submit jobs to `Columbia`, and a set of CFD simulation codes called `cart3d`. To minimize impact on the queue, the *emcee* was restricted to ten submitted simulation jobs at a time. Candidate locations were sub-sampled from a 3-d grid consisting of 37,909 configurations. The design was initialized with 30 candidates from a maximum entropy design, and 100 new candidates (from a treed maximum entropy design) were proposed during each AS round. We use full hierarchical prior for $\boldsymbol{\beta}$, described by Eq. (4) in Section 2.1.1, by augmenting the default with the augment `bprior = "b0"`. The pooling of means implied by the hierarchical prior is appropriate for this data since it is believed that the vast expanse of the response surface—for speeds greater than Mach 1—is largely homogeneous.

Figure 6 shows the 780 configurations sampled by BAS for the LGBB experiment. The left panel shows locations as a function of Mach (speed) and alpha (angle of attack), projecting over beta (side slip angle); the right panel shows Mach versus beta, projecting over alpha; the middle panel shows the beta = 0 slice. NASA recommended treating beta as discrete, so we used a set of values which they provided. We can see that most of the configurations chosen by BAS were located near Mach one, with highest density for large alpha. Samples are scarce for Mach greater than two and are relatively uniform across all beta settings. A small amount of random noise has been added to the beta values in the plots (*bottom–left*) for visibility purposes.

After samples are gathered, the treed GP model can be used for Monte Carlo estimation

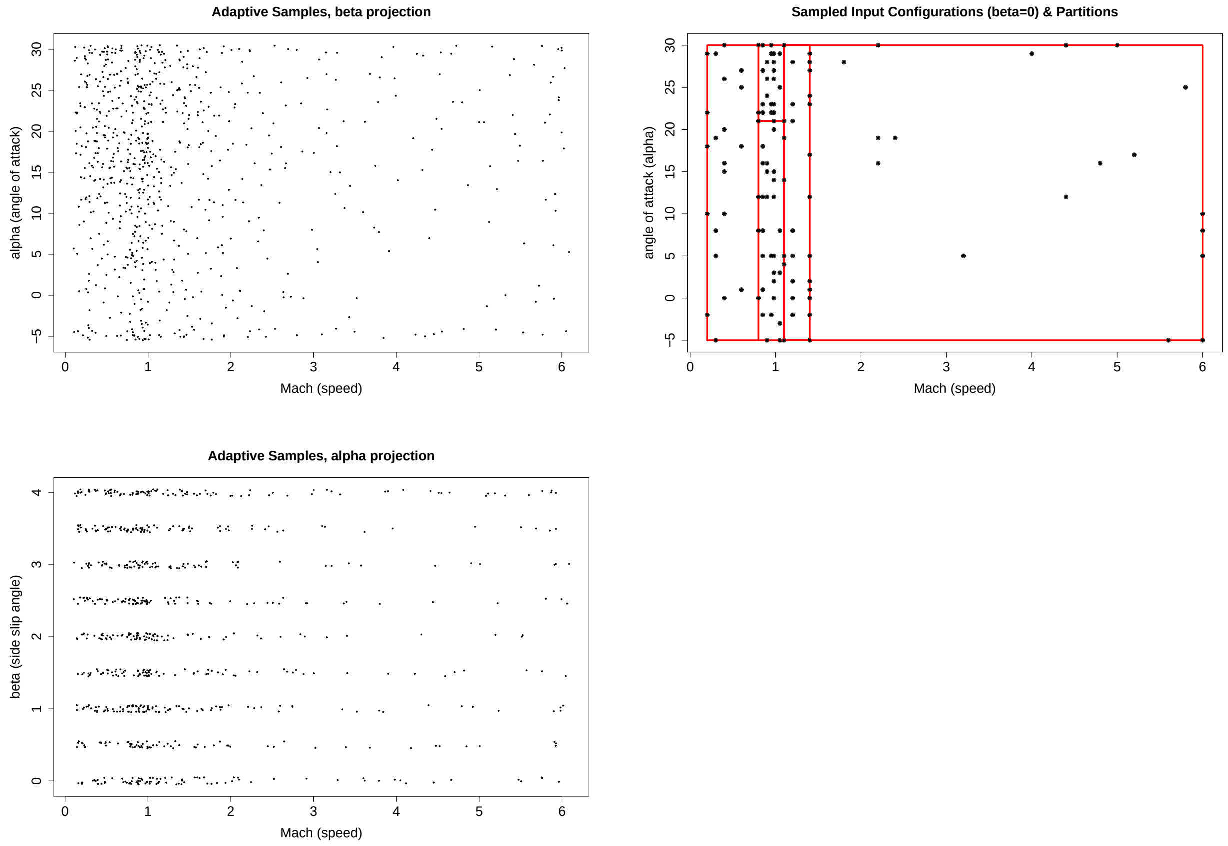

Figure 6: Adaptively sampled configurations projected over beta (side-slip angle; *top–left*), for fixed beta = 0 (*top–right*) with MAP partition $\hat{\mathcal{T}}$, and then over alpha (angle of attack; *bottom–left*).

of the posterior predictive distribution. The *upper–left* plot in Figure 7 shows a slice of the lift response, for fixed beta plotted as a function of Mach and alpha. [The *upper–right* panel of Figure 6 shows the corresponding adaptively sampled configurations and MAP tree $\hat{\mathcal{T}}$ (for beta = 0).] The MAP partition separates out the near-Mach-one region. Samples are densely concentrated in this region—most heavily for large alpha.

Figure 7 shows posterior predictive surfaces for the remaining five responses as well. Drag and Pitch are shown for the beta = 0 slice. Other slices look strikingly similar. Side, yaw, and roll are shown for the beta = 2 slice, as beta = 0 slices for these responses are essentially zero. All six responses exhibit similar characteristics, in that the supersonic cases are tame

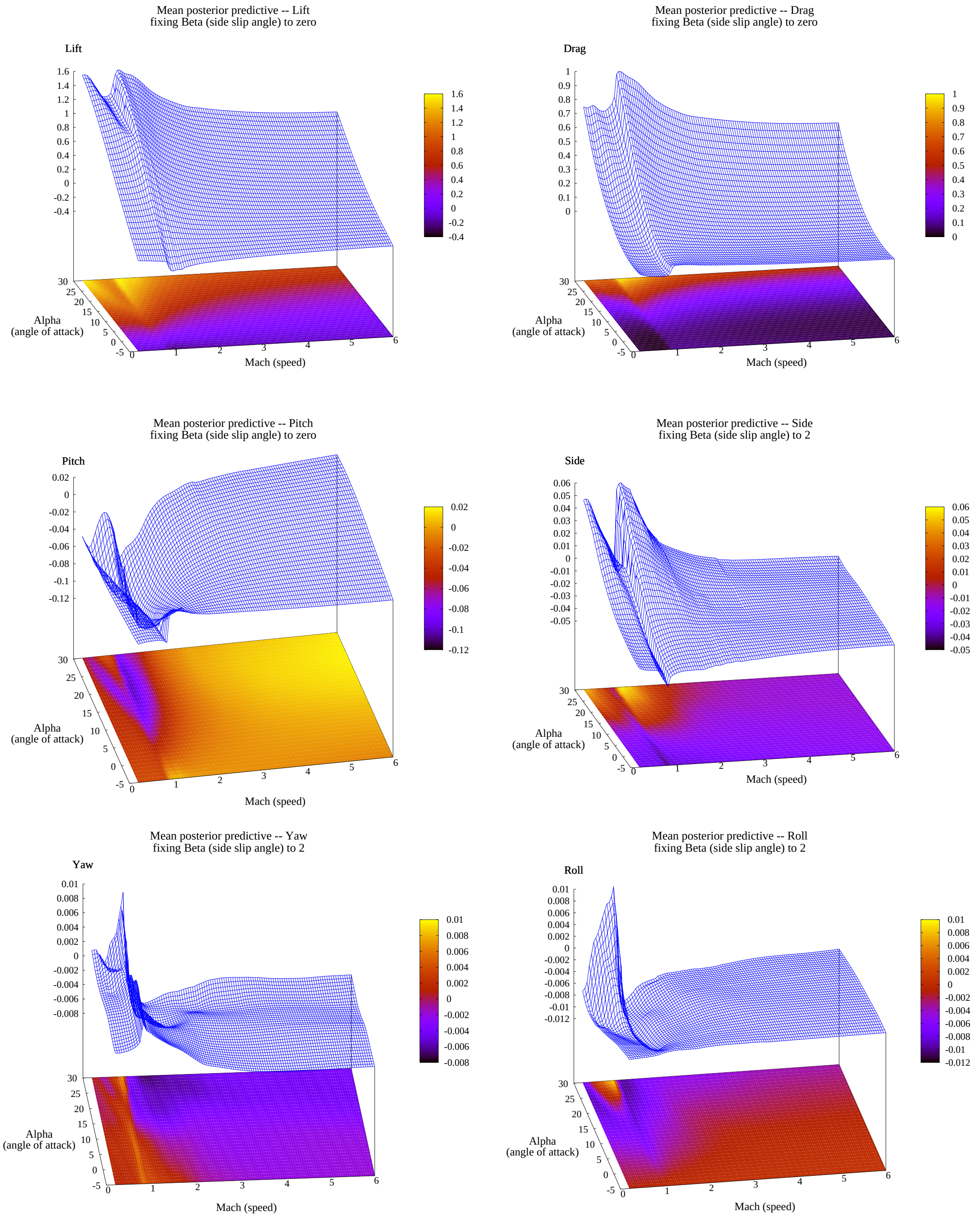

Figure 7: LGBB slice of mean posterior predictive surface for six responses (*lift, drag, pitch, side, yaw, roll*) plotted as a function of Mach (speed) and Alpha (angle of attack) with Beta (side slip angle) fixed at zero for the first three responses, and two for the last three.

relative to their subsonic counterparts, with the most interesting region occurring near Mach 1, and for large angle of attack (alpha). The treed GP model has enabled BAS to key in on this phenomenon and concentrate sampling there (Figure 6). Compared to the initial experiment, BAS reduced the simulation burden on the supercomputer by more than 75%.

6 Conclusion

We showed how the treed GP LLM can be used as a surrogate model in the sequential design of computer experiments. A hybrid approach, combining active learning and classical design methodologies, was taken in order to develop a flexible system for use in the highly variable environment of asynchronous agent–based supercomputing. Two sampling algorithms were proposed as adaptations to similar techniques developed for a simpler class of models. One chooses to sample configurations with high posterior predictive variance (ALM); the other uses a criterion based on an average global reduction in uncertainty (ALC). These model uncertainty statistics were used to determine which of a set optimally spaced candidate locations should go out for simulation next. Optimal candidate designs were determined by adapting a classic optimal design methodology to Bayesian partition models. The result is a highly efficient Bayesian adaptive sampling (BAS) strategy, representing a significant improvement on the state-of-the-art of computer experiment methodology at NASA. ALM, ALC, and treed maximum entropy design are implemented in the `tgp` package for `R` available on CRAN. Code for adaptive sampling via an asynchronous supercomputer (*emcee*) interface is available upon request.

There are some enhancements which can be made towards applying the methods herein to a broader array of problems. Three such closely related problems are of sampling to find extrema (Jones et al., 1998), to find contours (Ranjan et al., 2008) generally, or to find boundaries, i.e., contours with large gradients (Banerjee and Gelfand, 2006), a.k.a. Wombling. Other related problems include that of learning about, or finding extrema in, computer experiments with multi-fidelity codes of varying execution costs (Huang et al., 2006), and those which are paired with a *physical* experiment (Reese et al., 2004).

Acknowledgments

This work was partially supported by research subaward 08008-002-011-000 from the Universities Space Research Association and NASA, NASA/University Affiliated Research Center grant SC 2003028 NAS2-03144, Sandia National Laboratories grant 496420, and National Science Foundation grants DMS 0233710 and 0504851. The authors thank William Macready for originating the collaboration with NASA and for his help with the project, Thomas Pulliam and Edward Tejnil for their help with the NASA data, and the editor(s), associate editor, and two anonymous referees for their helpful comments and suggestions which have led to an improved paper.

A Active Learning – Cohn (ALC)

Section A.1 derives ALC for the hierarchical GP and Section A.2 does the same for the LLM.

A.1 For hierarchical Gaussian processes

The partition inverse equations (Barnett, 1979) can be used to write a covariance matrix $\mathbf{C}_{N+1}$ in terms of $\mathbf{C}_N$, so to obtain an equation for $\mathbf{C}_{N+1}^{-1}$ in terms of $\mathbf{C}_N^{-1}$:

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C}_N & \mathbf{m} \\ \mathbf{m}^\top & \kappa \end{bmatrix} \qquad \mathbf{C}_{N+1}^{-1} = \begin{bmatrix} [\mathbf{C}_N^{-1} + \mathbf{g}\mathbf{g}^\top \mu^{-1}] & \mathbf{g} \\ \mathbf{g}^\top & \mu \end{bmatrix} \tag{14}$$

where $\mathbf{m} = [C(\mathbf{x}_1, \mathbf{x}), \ldots, C(\mathbf{x}_N, \mathbf{x})]$, $\kappa = C(\mathbf{x}, \mathbf{x})$, for an $N+1^{\text{st}}$ point $\mathbf{x}$ where $C(\cdot, \cdot)$ is the covariance function, $\mathbf{g} = -\mu \mathbf{C}_N^{-1}\mathbf{m}$, and $\mu = (\kappa - \mathbf{m}^\top \mathbf{C}_N^{-1}\mathbf{m})^{-1}$. If $\mathbf{C}_N^{-1}$ is available, these partitioned inverse equations allow one to compute $\mathbf{C}_{N+1}^{-1}$, without explicitly constructing $\mathbf{C}_{N+1}$ (in $O(n^2)$ rather than the usual $O(n^3)$).

In the context of ALC sampling from the model in Eq. (4), the matrix which requires an inverse is $\mathbf{K}_{N+1} + \mathbf{F}_{N+1}\mathbf{W}\mathbf{F}_{N+1}^\top$, to calculate the predictive variance $\hat{\sigma}^2(\mathbf{x})$.

$$\mathbf{K}_{N+1} + \mathbf{F}_{N+1}^\top \mathbf{W}\mathbf{F}_{N+1} = \begin{bmatrix} \mathbf{K}_N & \mathbf{k}_N(\mathbf{x}) \\ \mathbf{k}_N^\top(\mathbf{x}) & K(\mathbf{x},\mathbf{x}) \end{bmatrix} + \begin{bmatrix} \mathbf{F}_N\mathbf{W}\mathbf{F}_N^\top & \mathbf{F}_N\mathbf{W}\mathbf{f}(\mathbf{x}) \\ \mathbf{f}(\mathbf{x})^\top\mathbf{W}\mathbf{F}_N^\top & \mathbf{f}(\mathbf{x})^\top\mathbf{W}\mathbf{f}(\mathbf{x}) \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{K}_N + \mathbf{F}_N\mathbf{W}\mathbf{F}_N^\top & \mathbf{k}_N(\mathbf{x}) + \mathbf{F}_N\mathbf{W}\mathbf{f}(\mathbf{x}) \\ \mathbf{k}_N^\top(\mathbf{x}) + \mathbf{f}(\mathbf{x})^\top\mathbf{W}\mathbf{F}_N^\top & K(\mathbf{x},\mathbf{x}) + \mathbf{f}(\mathbf{x})^\top\mathbf{W}\mathbf{f}(\mathbf{x}) \end{bmatrix}.$$

(*) Using the notation $\mathbf{C}_N = \mathbf{K}_N + \mathbf{F}_N\mathbf{W}\mathbf{F}_N^\top$, $\mathbf{q}_N(\mathbf{x}) = \mathbf{k}_N(\mathbf{x}) + \mathbf{F}_N\mathbf{W}\mathbf{f}(\mathbf{x})$, and $\kappa(\mathbf{x},\mathbf{y}) = K(\mathbf{x},\mathbf{y}) + \mathbf{f}(\mathbf{x})^\top\mathbf{W}\mathbf{f}(\mathbf{y})$ yields some simplification:

$$\mathbf{C}_{N+1} = \mathbf{K}_{N+1} + \mathbf{F}_{N+1}\mathbf{W}\mathbf{F}_{N+1}^\top = \begin{bmatrix} \mathbf{C}_N & \mathbf{q}_N(\mathbf{x}) \\ \mathbf{q}_N(\mathbf{x})^\top & \kappa(\mathbf{x},\mathbf{y}) \end{bmatrix}.$$

Applying the partitioned inverse equations (14) gives

$$\mathbf{C}_{N+1}^{-1} = (\mathbf{K}_{N+1} + \mathbf{F}_{N+1}^\top\mathbf{W}\mathbf{F}_{N+1})^{-1} = \begin{bmatrix} [\mathbf{C}_N^{-1} + \mathbf{g}\mathbf{g}^\top\mu^{-1}] & \mathbf{g} \\ \mathbf{g}^\top & \mu \end{bmatrix} \tag{15}$$

where $\mathbf{g} = -\mu\mathbf{C}_N^{-1}\mathbf{q}_N(\mathbf{x})$, and $\mu = (\kappa(\mathbf{x},\mathbf{x}) - \mathbf{q}_N(\mathbf{x})^\top\mathbf{C}_N^{-1}\mathbf{q}_N(\mathbf{x}))^{-1}$ from (*). We can now calculate the reduction in variance at $\mathbf{y}$ given that $\mathbf{x}$ is added into the data:

$$\begin{aligned} & \Delta\hat\sigma_{\mathbf{y}}^2(\mathbf{x}) = \hat\sigma^2(\mathbf{y}) - \hat\sigma_{\mathbf{x}}^2(\mathbf{y}), \\ \text{where} \quad & \hat\sigma^2(\mathbf{y}) = \sigma^2[\kappa(\mathbf{y},\mathbf{y}) - \mathbf{q}_N^\top(\mathbf{y})\mathbf{C}_N^{-1}\mathbf{q}_N^\top(\mathbf{y})], \\ \text{and} \quad & \hat\sigma_{\mathbf{x}}^2(\mathbf{y}) = \sigma^2[\kappa(\mathbf{y},\mathbf{y}) - \mathbf{q}_{N+1}(\mathbf{y})^\top\mathbf{C}_{N+1}^{-1}\mathbf{q}_{N+1}(\mathbf{y})]. \\ \text{Now} \quad & \Delta\hat\sigma_{\mathbf{y}}^2(\mathbf{x}) = \sigma^2[\kappa(\mathbf{y},\mathbf{y}) - \mathbf{q}_N^\top(\mathbf{y})\mathbf{C}_N^{-1}\mathbf{q}_N(\mathbf{y})] - \sigma^2[\kappa(\mathbf{y},\mathbf{y}) - \mathbf{q}_{N+1}^\top(\mathbf{y})\mathbf{C}_{N+1}^{-1}\mathbf{q}_{N+1}(\mathbf{y})] \\ & \qquad = \sigma^2[\mathbf{q}_{N+1}(\mathbf{y})^\top\mathbf{C}_{N+1}^{-1}\mathbf{q}_{N+1}(\mathbf{y}) - \mathbf{q}_N^\top(\mathbf{y})\mathbf{C}_N^{-1}\mathbf{q}_N(\mathbf{y})]. \end{aligned}$$

Focusing on $\mathbf{q}_{N+1}^\top(\mathbf{y})\mathbf{C}_{N+1}^{-1}\mathbf{q}_{N+1}(\mathbf{y})$, first decompose $\mathbf{q}_{N+1}$:

$$\begin{aligned} \mathbf{q}_{N+1} &= \mathbf{k}_{N+1}(\mathbf{y}) + \mathbf{F}_{N+1}\mathbf{W}\mathbf{f}(\mathbf{y}) \\ &= \begin{bmatrix} \mathbf{k}_N(\mathbf{y}) \\ K(\mathbf{y},\mathbf{x}) \end{bmatrix} + \begin{bmatrix} \mathbf{F}_N \\ \mathbf{f}^\top(\mathbf{x}) \end{bmatrix} \mathbf{W}\mathbf{f}(\mathbf{y}) = \begin{bmatrix} \mathbf{k}_N(\mathbf{y}) + \mathbf{F}_N\mathbf{W}\mathbf{f}(\mathbf{y}) \\ K(\mathbf{y},\mathbf{x}) + \mathbf{f}^\top(\mathbf{x})\mathbf{W}\mathbf{f}(\mathbf{y}) \end{bmatrix} = \begin{bmatrix} \mathbf{q}_N(\mathbf{y}) \\ \kappa(\mathbf{x},\mathbf{y}) \end{bmatrix}. \end{aligned}$$

Turning attention back to $\mathbf{C}_{N+1}^{-1}\mathbf{q}_{n+1}(\mathbf{y})$, with the help of (15):

$$\mathbf{C}_{N+1}^{-1}\mathbf{q}_{N+1}(\mathbf{y}) = \left[\begin{array}{cc} \mathbf{C}_N^{-1} + \mathbf{g}\mathbf{g}^\top\mu^{-1} & \mathbf{g} \\ \mathbf{g}^\top & \mu \end{array}\right]\left[\begin{array}{c} \mathbf{q}_N(\mathbf{y}) \\ \kappa(\mathbf{x},\mathbf{y}) \end{array}\right] = \left[\begin{array}{c} [\mathbf{C}_N^{-1} + \mathbf{g}\mathbf{g}^\top\mu^{-1}]\mathbf{q}_N(\mathbf{y}) + \mathbf{g}\kappa(\mathbf{x},\mathbf{y}) \\ \mathbf{g}^\top\mathbf{q}_N(\mathbf{y}) + \mu\kappa(\mathbf{x},\mathbf{y}) \end{array}\right]$$

$$\begin{aligned} \mathbf{q}_{N+1}^\top(\mathbf{y})\mathbf{C}_{N+1}^{-1}\mathbf{q}_{N+1}(\mathbf{y}) &= \left[\begin{array}{c} \mathbf{q}_N(\mathbf{y}) \\ \kappa(\mathbf{x},\mathbf{y}) \end{array}\right]^\top \left[\begin{array}{c} (\mathbf{C}_N^{-1} + \mathbf{g}\mathbf{g}^\top\mu^{-1})\mathbf{q}_N(\mathbf{y}) + \mathbf{g}\kappa(\mathbf{x},\mathbf{y})) \\ \mathbf{g}^\top\mathbf{q}_N(\mathbf{y}) + \mu\kappa(\mathbf{x},\mathbf{y}) \end{array}\right] \\ &= \mathbf{q}_N^\top(\mathbf{y})[(\mathbf{C}_N^{-1} + \mathbf{g}\mathbf{g}^\top\mu^{-1})\mathbf{q}_N(\mathbf{y}) + \mathbf{g}\kappa(\mathbf{x},\mathbf{y})] \\ &\quad + \kappa(\mathbf{x},\mathbf{y})[\mathbf{g}^\top\mathbf{q}_N(\mathbf{y}) + \mu\kappa(\mathbf{x},\mathbf{y})]. \end{aligned}$$

Finally

$$\begin{aligned} \Delta\hat{\sigma}_{\mathbf{y}}^2(\mathbf{x}) &= \sigma^2[\mathbf{q}_{N+1}(\mathbf{y})^\top\mathbf{C}_{N+1}^{-1}\mathbf{q}_{N+1}(\mathbf{y}) - \mathbf{q}_N^\top(\mathbf{y})\mathbf{C}_N^{-1}\mathbf{q}_N(\mathbf{y})]. \\ &= \sigma^2[\mathbf{q}_N^\top(\mathbf{y})\mathbf{g}\mathbf{g}^\top\mu^{-1}\mathbf{q}_N(\mathbf{y}) + 2\kappa(\mathbf{x},\mathbf{y})\mathbf{g}^\top\mathbf{q}_N(\mathbf{y}) + \mu\kappa(\mathbf{x},\mathbf{y})^2] \\ &= \sigma^2\mu\left[\mathbf{q}_N^\top(\mathbf{y})\mathbf{g}\mu^{-1} - \kappa(\mathbf{x},\mathbf{y})\right]^2 \\ \Delta\hat{\sigma}_{\mathbf{y}}^2(\mathbf{x}) &= \frac{\sigma^2\left[\mathbf{q}_N^\top(\mathbf{y})\mathbf{C}_N^{-1}\mathbf{q}_N(\mathbf{x}) - \kappa(\mathbf{x},\mathbf{y})\right]^2}{\kappa(\mathbf{x},\mathbf{x}) - \mathbf{q}_N^\top(\mathbf{x})\mathbf{C}_N^{-1}\mathbf{q}_N(\mathbf{x})}. \end{aligned}$$

A.2 For hierarchical (limiting) linear models

Under the (limiting) linear model, computing the ALC statistic is more straightforward.

$$\begin{aligned} \Delta\hat{\sigma}_{\mathbf{y}}^2(\mathbf{x}) = \hat{\sigma}^2(\mathbf{y}) - \hat{\sigma}_{\mathbf{x}}^2(\mathbf{y}) &= \sigma^2[1 - \mathbf{f}^\top(\mathbf{y})\mathbf{V}_{\tilde{\beta}_N}\mathbf{f}(\mathbf{y}) - 1 - \mathbf{f}^\top(\mathbf{y})\mathbf{V}_{\tilde{\beta}_{N+1}}\mathbf{f}(\mathbf{y})] \\ &= \sigma^2\mathbf{f}^\top(\mathbf{y})[\mathbf{V}_{\tilde{\beta}_N} - \mathbf{V}_{\tilde{\beta}_{N+1}}]\mathbf{f}(\mathbf{y}), \end{aligned}$$

where $\mathbf{V}_{\tilde{\beta}_{N+1}}$ from (Gramacy and Lee, 2008a) includes $\mathbf{x}$, and $\mathbf{V}_{\tilde{\beta}_N}$ does not. Expanding out $\mathbf{V}_{\tilde{\beta}_{N+1}}$:

$$\begin{aligned}
\Delta\hat{\sigma}^2_{\mathbf{y}}(\mathbf{x}) &= \sigma^2\mathbf{f}^\top(\mathbf{y})\left[\mathbf{V}_{\tilde{\beta}_N} - \left(\frac{\mathbf{W}^{-1}}{\tau^2} + \frac{\mathbf{F}^\top_{N+1}\mathbf{F}_{N+1}}{1+g}\right)^{-1}\right]\mathbf{f}^\top(\mathbf{y}) \\
&= \sigma^2\mathbf{f}^\top(\mathbf{y})\left[\mathbf{V}_{\tilde{\beta}_N} - \left(\frac{\mathbf{W}^{-1}}{\tau^2} + \frac{1}{1+g}\begin{bmatrix}\mathbf{F}_N \\ \mathbf{f}^\top(\mathbf{x})\end{bmatrix}^\top \begin{bmatrix}\mathbf{F}_N \\ \mathbf{f}^\top(\mathbf{x})\end{bmatrix}\right)^{-1}\right]\mathbf{f}(\mathbf{y}) \\
&= \sigma^2\mathbf{f}^\top(\mathbf{y})\left[\mathbf{V}_{\tilde{\beta}_N} - \left(\frac{\mathbf{W}^{-1}}{\tau^2} + \frac{\mathbf{F}^\top_N\mathbf{F}_N}{1+g} + \frac{\mathbf{f}(\mathbf{x})\mathbf{f}^\top(\mathbf{x})}{1+g}\right)^{-1}\right]\mathbf{f}(\mathbf{y}) \\
&= \sigma^2\mathbf{f}^\top(\mathbf{y})\left[\mathbf{V}_{\tilde{\beta}_N} - \left(\mathbf{V}^{-1}_{\tilde{\beta}_N} + \frac{\mathbf{f}(\mathbf{x})\mathbf{f}^\top(\mathbf{x})}{1+g}\right)^{-1}\right]\mathbf{f}(\mathbf{y}).
\end{aligned}$$

The Sherman-Morrison-Woodbury formula (Bernstein, 2005), where $\mathbf{V} \equiv \mathbf{f}^\top(\mathbf{x})(1+g)^{-\frac{1}{2}}$ and $\mathbf{A} \equiv \mathbf{V}^{-1}_{\tilde{\beta}_N}$ gives

$$\begin{aligned}
\Delta\hat{\sigma}^2_{\mathbf{y}}(\mathbf{x}) &= \sigma^2\mathbf{f}^\top(\mathbf{y})\left[\left(1 + \frac{\mathbf{f}^\top(\mathbf{x})\mathbf{V}_{\tilde{\beta}_N}\mathbf{f}(\mathbf{x})}{1+g}\right)^{-1}\mathbf{V}_{\tilde{\beta}_N}\frac{\mathbf{f}(\mathbf{x})\mathbf{f}^\top(\mathbf{x})}{1+g}\mathbf{V}_{\tilde{\beta}_N}\right]\mathbf{f}(\mathbf{y}) \\
\Delta\hat{\sigma}^2_{\mathbf{y}}(\mathbf{x}) &= \frac{\sigma^2[\mathbf{f}^\top(\mathbf{y})\mathbf{V}_{\tilde{\beta}_N}\mathbf{f}(\mathbf{x})]^2}{1+g+\mathbf{f}^\top(\mathbf{x})\mathbf{V}_{\tilde{\beta}_N}\mathbf{f}(\mathbf{x})}.
\end{aligned}$$

References

Abrahamsen, P. (1997). "A Review of Gaussian Random Fields and Correlation Functions." Tech. Rep. 917, Norwegian Computing Center, Box 114 Blindern, N-0314 Oslo, Norway.

Angluin, D. (1987). "Queries and concept learning." *Machine Learning*, 2, 319–342.

Atlas, L., Cohn, D., Ladner, R., El-Sharkawi, M., Marks, R., Aggoune, M., and Park, D. (1990). "Training Connectionist Networks with Queries and Selective Sampling." *Advances in Neural Information Processing Systems*, 566–753.

Banerjee, S. and Gelfand, A. E. (2006). "Bayesian Wombling: Curvilinear Gradient Assessment Under Spatial Process Models." *Journal of the American Statistical Association*, 101, 15, 1487–1501.

Barnett, S. (1979). *Matrix Methods for Engineers and Scientists*. McGraw-Hill.

Bates, R. A., Buck, R. J., Riccomagno, E., and Wynn, H. P. (1996). “Experimental Design and Observation for Large Systems.” *Journal of the Royal Statistical Society, Series B.*, 58, 77–94.

Bernstein, D. (2005). *Matrix Mathematics*. Princeton, NJ: Princeton University Press.

Chaloner, K. and Verdinelli, I. (1995). “Bayesian Experimental Design, A Review.” *Statistical Science*, 10 No. 3, 273–304.

Chipman, H., George, E., and McCulloch, R. (1998). “Bayesian CART Model Search (with discussion).” *Journal of the American Statistical Association*, 93, 935–960.

— (2002). “Bayesian Treed Models.” *Machine Learning*, 48, 303–324.

Cohn, D. A. (1996). “Neural Network Exploration using Optimal Experimental Design.” In *Advances in Neural Information Processing Systems*, vol. 6(9), 679–686. Morgan Kaufmann Publishers.

Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D. (1991). “Bayesian Prediction of Deterministic Functions, with Applications to the Design and Analysis of Computer Experiments.” *Journal of the American Statistical Association*, 86, 953–963.

Denison, D., Mallick, B., and Smith, A. (1998). “A Bayesian CART Algorithm.” *Biometrika*, 85, 363–377.

Gramacy, R. B. (2007). “tgp: An R Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models.” *Journal of Statistical Software*, 19, 9.

Gramacy, R. B. and Lee, H. K. H. (2008a). “Bayesian treed Gaussian process models with an application to computer modeling.” *Journal of the American Statistical Association*, 103, 1119–1130.

— (2008b). “Gaussian Processes and Limiting Linear Models.” *Computational Statistics and Data Analysis*, 53, 123–136.

Gramacy, R. B., Lee, H. K. H., and Macready, W. (2004). "Parameter Space Exploration With Gaussian Process Trees." In *ICML*, 353–360. Omnipress & ACM Digital Library.

Gramacy, R. B. and Taddy, M. A. (2008). `tgp`*: Bayesian treed Gaussian process models*. R package version 2.1-2.

Hamada, M., Martz, H., Reese, C., and Wilson, A. (2001). "Finding Near-optimal Bayesian Experimental Designs by Genetic Algorithms." *Journal of the American Statistical Association*, 55–3, 175–181.

Higdon, D. (2002). "Space and Space–time Modeling Using Process Convolutions." In *Quantitative Methods for Current Environmental Issues*, eds. C. Anderson, V. Barnett, P. C. Chatwin, and A. H. El-Shaarawi, 37–56. London: Springer-Verlag.

Higdon, D., Swall, J., and Kern, J. (1999). "Non-Stationary Spatial Modeling." In *Bayesian Statistics 6*, eds. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, 761–768. Oxford University Press.

Huang, D., Allen, T., Notz, W., and Miller, R. (2006). "Sequential Kriging Optimization Using Multiple Fidelity Evaluations." *Structural and Multidisciplinary Optimization*, 32, 5, 369–382.

Jones, D., Schonlau, M., and Welch, W. J. (1998). "Efficient Global Optimization of Expensive Black Box Functions." *Journal of Global Optimization*, 13, 455–492.

Kennedy, M. and O'Hagan, A. (2000). "Predicting the Output from a Complex Computer Code when Fast Approximations are Available." *Biometrika*, 87, 1–13.

— (2001). "Bayesian Calibration of Computer Models (with discussion)." *Journal of the Royal Statistical Society, Series B*, 63, 425–464.

Kleijnen, J. P. C. and van Beers, W. C. M. (2004). "Application-driven sequential designs for simulation experiments: Kriging metamodeling." *Journal of the Operational Research Society*, 55, 9, 876–883.

MacKay, D. J. C. (1992). "Information–based Objective Functions for Active Data Selection." *Neural Computation*, 4, 4, 589–603.

McKay, M. D., Conover, W. J., and Beckman, R. J. (1979). "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code." *Technometrics*, 21, 239–245.

Müller, P., Sansó, B., and de Iorio, M. (2004). "Optimal Bayesian Design by Inhomogeneous Markov Chain Simulation." *Journal of the American Statistical Association*, 99(467), Theory and Methods, 788–798.

Neal, R. (1997). "Monte Carlo implementation of Gaussian process models for Bayesian regression and classification"." Tech. Rep. CRG–TR–97–2, Dept. of Computer Science, University of Toronto.

Paciorek, C. (2003). "Nonstationary Gaussian Processes for Regression and Spatial Modelling." Ph.D. thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Ramakrishnan, N., Bailey-Kellogg, C., Tadepalli, S., and Pandey, V. (2005). "Gaussian Processes for Active Data Mining of Spatial Aggregates." In *Proceedings of the SIAM Data Mining Conference*.

Ranjan, P., Bingham, D., and Michailidis, G. (2008). "Sequential Experiment Design for Contour Estimation from Complex Computer Codes." *Technometrics*. *To appear.*

Reese, C., Wilson, A., Hamada, M., Martz, H., and Ryan, K. (2004). "Integrated Analysis of Computer and Physical Experiments." *Technometrics*, 46(2), 153–164.

Rogers, S. E., Aftosmis, M. J., Pandya, S. A., N. M. Chaderjian, E. T. T., and Ahmad, J. U. (2003). "Automated CFD Parameter Studies on Distributed Parallel Computers." In *16th AIAA Computational Fluid Dynamics Conference*. AIAA Paper 2003-4229.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). "Design and Analysis of Computer Experiments." *Statistical Science*, 4, 409–435.

Sampson, P. D. and Guttorp, P. (1992). “Nonparametric Estimation of Nonstationary Spatial Covariance Structure.” *Journal of the American Statistical Association*, 87(417), 108–119.

Santner, T. J., Williams, B. J., and Notz, W. I. (2003). *The Design and Analysis of Computer Experiments*. New York, NY: Springer-Verlag.

Schmidt, A. and Gelfand, A. (2003). “A Bayesian Coregionalization Approach for Multivariate Pollutant Data.” *Journal of Geophysical Research–Atmospheres*, D24, 8783, 108.

Schmidt, A. M. and O’Hagan, A. (2003). “Bayesian Inference for Nonstationary Spatial Covariance Structure via Spatial Deformations.” *Journal of the Royal Statistical Society, Series B*, 65, 745–758.

Sebastiani, P. and Wynn, H. P. (2000). “Maximum Entropy Sampling and Optimal Bayesian Experimental Design.” *Journal of the Royal Statistical Society, Series B*, 62, 145–157.

Seo, S., Wallat, M., Graepel, T., and Obermayer, K. (2000). “Gaussian Process Regression: Active Data Selection and Test Point Rejection.” In *Proceedings of the International Joint Conference on Neural Networks*, vol. III, 241–246. IEEE.

R Development Core Team (2004). R*: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Aus. ISBN 3-900051-00-3.

Shewry, M. and Wynn, H. (1987). “Maximum entropy sampling.” *Journal of Applied Statistics*, 14, 165–170.

Ver Hoef, J. and Barry, R. P. (1998). “Constructing and Fitting Models for Cokriging and Multivariate Spatial Prediction.” *Journal of Statistical Planning and Inference*, 69, 275–294.

Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T., and Morris, M. D. (1992). “Screening, Predicting, and Computer Experiments.” *Technometrics*, 34, 15–25.