Adaptive design of supercomputer experiments

Robert B. Gramacy
bobby@statslab.cam.ac.uk
The 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 obtaining insufficient information in parts of the space, particularly when the surface is nonstationary. We propose an approach which 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 nonstationary measures of uncertainty. Our adaptive sequential design framework has been developed to cope with an asynchronous, random, agent-based supercomputing environment. We take a hybrid approach which 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 example of a computational fluid dynamics simulation of 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 methods (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 new class of surrogate models and 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 the mapping between inputs and outputs, simulations must be run for many different input configurations in order to build up an understanding of its possible outcomes. High fidelity supercomputer experiments are usually run on clusters of independent computing agents, or processors. Agents can process one input configuration at a time. Multiple agents allow several input configurations to be run in parallel, starting and finishing at different, possibly random, times. The cluster is usually managed by master controller *(emcee)* program that gathers responses from finished simulations, and keeps free agents busy with new inputs.

Even in extremely parallel computing environments, computational expense of the simulation and/or high dimensional inputs often prohibit the naïve approach of running the experiment over a dense grid of input configurations. More sophisticated design strategies, such as Latin Hypercube (LH) and maximin designs, orthogonal arrays, and D/A–optimal 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". D/A–optimal 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 are nonstationary—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 (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 plotted 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 of subsonic flows can be

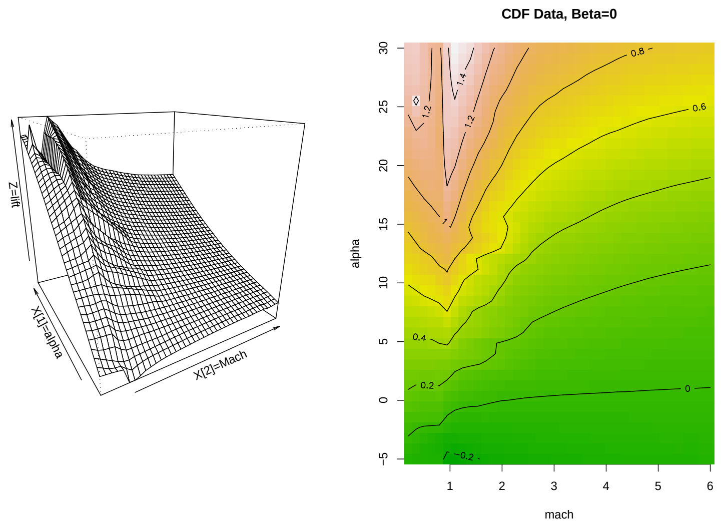

Figure 1: Lift plotted as a function of Mach (speed) and alpha (angle of attack) with beta (side-slip angle) fixed to zero. The ridge at Mach 1 separates subsonic from supersonic cases.

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 in this experiment involve the integration of the inviscid Euler equations over a mesh of 1.4 million cells. 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 (Rogers et al., 2003). 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 model is needed which can capture the nonstationary behavior between subsonic and supersonic physical regimes. If such a model could efficiently produce region-specific estimates of distance, and/or variance, it could be used as the basis of an adaptive nonstationary sequential design.

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 three important disadvantages to the standard GP in this setting. Firstly, inference on the GP scales poorly with the number of data points, typically requiring computing time that grows with the cube of the sample size. Secondly, 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 than supersonic flow, this limitation is unacceptable. Thirdly, 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, which is related to its stationarity. The Bayesian treed GP model (Gramacy and Lee, 2006) 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, are inherently serial, and thus tacitly assume a controlled and isolated computing environment. The modern supercomputer has thousands of computing nodes, or agents, designed 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 is no 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). The treed GP model, classic sequential design, and active learning, taken all together, result in a highly efficient nonstationary modeling and hybrid design strategy that balances optimality and flexibility in order to adaptively design a supercomputer experiment.

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 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. Finally, Section 6 offers some discussion, and avenues for further research.

2 Review

Our approach to 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 $z(\mathbf{x})$, is typically modeled as $z(\mathbf{x}) = \boldsymbol{\beta}^\top \mathbf{x} + w(\mathbf{x})$ for a particular (multivariate) input configuration value $\mathbf{x}$, where $\boldsymbol{\beta}$ are linear trend coefficients, and $w(\mathbf{x})$ is 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 a model that fits this description, and consequently is the canonical surrogate model used in designing computer experiments (Sacks et al., 1989; Santner et al., 2003).

Let D be the set of m_X dimensional input configurations $(\mathbf{x})$, and outputs (z) from a computer code: $D = \{\mathbf{x}_i, z_i\}_{i=1}^N$. The collection of inputs is indicated as the $N \times m_X$ matrix $\mathbf{X}$ whose i^{th} row is $\mathbf{x}_i^\top$, and the outputs are collected in an N–vector called $\mathbf{Z}$. Given a set of observations D, under a GP model the density over outputs at a new point $\mathbf{x}$ has a Normal distribution with (assuming, for now, that $\boldsymbol{\beta} = \mathbf{0}$)

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

where $\mathbf{k}^\top(\mathbf{x})$ is the N-vector whose i^{th} component is $K(\mathbf{x}, \mathbf{x}_i)$, $\mathbf{K}$ is the $N \times N$ matrix with i, j element $K(\mathbf{x}_i, \mathbf{x}_j)$, and $\mathbf{Z}$ is the N-vector of observations with i^{th} component z_i. 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 to inference 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 (2006) 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 $Z(\mathbf{x}) = \boldsymbol{\beta}^\top \mathbf{x} + \mathbf{w}(\mathbf{x}) + \eta(\mathbf{x})$ where $\eta(\cdot)$ is independent Gaussian noise. Valid correlation functions $K^*(\cdot, \cdot)$ are usually generated as a member of a parametric family. Examples include the isotropic or separable power or Matérn families. A nice general reference for families of correlation functions K^* is provided by Abrahamsen (1997). Hereafter we use the separable power family, a standard choice in computer experiments (Santner et al., 2003).

While many authors (e.g., Santner et al., 2003; Sacks et al., 1989) deliberately omit the nugget parameter on 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 conditions, 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).

2.2 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 simulation applications (Sacks et al., 1989; Santner et al., 2003; Currin et al., 1991; Welch et al., 1992). 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 $D-$optimal designs. Choosing quadratic loss leads to $A-$optimal designs. 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, simulated annealing, tabu-search (Glover and Laguna, 1997), genetic algorithms (Hamada et al., 2001), etc. (Welch et al., 1992; Currin et al., 1991; Mitchell, 1974). 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 design 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 approaches used by the statistics community do not require a model of covariance. These include space-filling designs: e.g., maximin distance and LH designs (Box et al., 1978; Santner et al., 2003; McKay et al., 1979). Computing maximin distance designs can also be computationally intensive, whereas LH designs are easy to compute and result in well-spaced configurations relative to random sampling, though there are some degenerate cases, like diagonal LH designs (Santner et al., 2003). LH designs 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. Whereas most optimal design methods like D-optimal, A-optimal, and maximin, are more computationally intense, 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 the DOE using the GP. The first approach tries to maximize the information gained about model parameters by selecting from a pool 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).

An alternative algorithm, called ALC for Active Learning–Cohn, is to select $\tilde{\mathbf{x}}$ minimizing the expected squared error averaged over the input space (Cohn, 1996). The global reduction in predictive variance, given that the location $\tilde{\mathbf{x}}$ is added into the data, is obtained by averaging over other locations $\mathbf{y}$:

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

using the notation from (1). In practice the integral is usually 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 between ALC and ALM using standard GPs.

2.2.2 Other approaches to designing computer experiments

Bayesian & non-Bayesian approaches to surrogate modeling and design for computer experiments abound (Sebastiani and Wynn, 2000; Welch et al., 1992; Currin et al., 1991; Mitchell and Morris, 1992; Sacks et al., 1989; Bates et al., 1996). References for the Bayesian approach usually include the landmark papers by Kennedy & O'Hagan et al. (2000, 2001), which provide a framework using stationary GPs. 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). Specifically, the use of nonstationary surrogate models distinguishes our work from the methods described in those papers, encouraging a more adaptive approach to sequential design. We presume to be pioneers in this respect. Generally, the methods we describe create a much broader framework. Motivations for a fresh approach to SDACE range from the inadequate nature of standard surrogate models (both in terms of speed and modeling capacity, particularly stationarity assumptions) to the challenges inherent in adapting standard optimal design techniques to modern supercomputer experiments which tend to run on highly parallel and distributed supercomputers.

2.3 Treed Gaussian process model

We propose a new surrogate model for the sequential DOE: the Bayesian treed Gaussian process (treed GP) model (Gramacy and Lee, 2006). The treed GP model extends Bayesian linear CART 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. 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:

$$\begin{aligned} \mathbf{Z}_\nu|\boldsymbol{\beta}_\nu, \sigma_\nu^2, \mathbf{K}_\nu &\sim N_{n_\nu}(\mathbf{F}_\nu\boldsymbol{\beta}_\nu, \sigma_\nu^2\mathbf{K}_\nu) & \boldsymbol{\beta}_0 &\sim N_{m_X}(\boldsymbol{\mu}, \mathbf{B}) & \sigma_\nu^2 &\sim IG(\alpha_\sigma/2, q_\sigma/2) \\ \boldsymbol{\beta}_\nu|\sigma_\nu^2, \tau_\nu^2, \mathbf{W}, \boldsymbol{\beta}_0 &\sim N_{m_X}(\boldsymbol{\beta}_0, \sigma_\nu^2\tau_\nu^2\mathbf{W}) & \mathbf{W}^{-1} &\sim W((\rho\mathbf{V})^{-1}, \rho) & \tau_\nu^2 &\sim IG(\alpha_\tau/2, q_\tau/2) \end{aligned} \tag{3}$$

with $\mathbf{F}_\nu = (\mathbf{1}, \mathbf{X}_\nu)$, and $\mathbf{W}$ is a $(m_X+1)\times(m_X+1)$ matrix. N, IG, and W are the (Multivariate) Normal, Inverse-Gamma, and Wishart distributions, respectively. $\mathbf{K}_\nu$ is the separable power family covariance matrix, as in (2). The data $\{\mathbf{X}, \mathbf{Z}\}_\nu$ in region r_ν are used to estimate the parameters $\boldsymbol{\theta}_\nu$ 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 and nugget parameters, and their hyperparameters, which can be sampled via Metropolis-Hastings. More details are provided by Gramacy and Lee (2006).

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

$$\hat{z}(\mathbf{x}) = E(\mathbf{Z}(\mathbf{x})|\text{ data}, \mathbf{x} \in D_\nu) = \mathbf{f}^\top(\mathbf{x})\tilde{\boldsymbol{\beta}}_\nu + \mathbf{k}_\nu(\mathbf{x})^\top\mathbf{K}_\nu^{-1}(\mathbf{Z}_\nu - \mathbf{F}_\nu\tilde{\boldsymbol{\beta}}_\nu), \tag{4}$$

$$\hat{\sigma}(\mathbf{x})^2 = \text{Var}(\mathbf{z}(\mathbf{x})|\text{ data}, \mathbf{x} \in D_\nu) = \sigma_\nu^2[\kappa(\mathbf{x},\mathbf{x}) - \mathbf{q}_\nu^\top(\mathbf{x})\mathbf{C}_\nu^{-1}\mathbf{q}_\nu(\mathbf{x})], \tag{5}$$

where

$$\mathbf{C}_\nu^{-1} = (\mathbf{K}_\nu + \tau_\nu^2\mathbf{F}_\nu\mathbf{W}\mathbf{F}_\nu^\top)^{-1} \qquad \mathbf{q}_\nu(\mathbf{x}) = \mathbf{k}_\nu(\mathbf{x}) + \tau_\nu^2\mathbf{F}_\nu\mathbf{W}_\nu\mathbf{f}(\mathbf{x}) \tag{6}$$

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

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, in particular, $\hat{\sigma}(\mathbf{x})^2$ in (5) is thus 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 (i.e., Bayesian model averaging). 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 linear CART 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 $\mathbf{b}$ which are given a prior conditional on the range parameter to $K(\cdot, \cdot)$. The boolean b_i determines whether the GP or its LLM governs the marginal process in the i^{th} dimension. 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, 2006). 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; Damian et al., 2001; Schmidt and O'Hagan, 2003) and process convolutions (Higdon et al., 1999; Fuentes and Smith, 2001; Paciorek, 2003), the treed GP LLM approach yields an extremely fast implementation of nonstationary GPs, providing a divide-and-conquer approach to spatial modeling. Software implementing the treed GP LLM model and all of its special cases (e.g., stationary GP, linear CART, 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/src/contrib/Descriptions/tgp.html`.

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

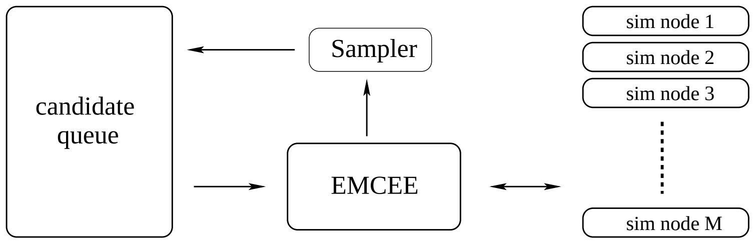

Figure 2: *Emcee* program gives finished simulations to the sampler and gets new ones from the queue.

developing improvements to this approach. The first task is to introduce the asynchronous distributed supercomputer model commonly used to run complex computer codes. Methodologies can then be explored for choosing new input configurations based on region-specific estimates of uncertainty, provided by the nonstationary treed GP LLM surrogate model. (We shall drop the LLM tag in what follows, and consider it implied by the label treed GP.)

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 usually managed asynchronously by a master controller *(emcee)* program that gathers responses from finished simulations, and supplies free agents with new input configurations. The goal is to have the *emcee* program interact with a 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 a traditional 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 no longer optimal. 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. Aside from the boundary-favoring, other drawbacks to the standard statistical 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 $D-$optimal, maximin, or LH, 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 minimizes the expected reduction in squared error averaged over the input space (Cohn, 1996). Rather than focusing solely on design points which have large predictive variance, ALC selects configurations that would lead to a global reduction in predictive variance. Conditioning on $\mathcal{T}$, the reduction in variance at a point $\mathbf{y} \in \mathbf{Y}_\nu$, given that the location $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}_\nu$ is added into the data, is defined as (region subscripts suppressed):

$$\Delta\hat{\sigma}^2_{\mathbf{y}}(\mathbf{x}) = \hat{\sigma}^2_{\mathbf{y}} - \hat{\sigma}^2_{\mathbf{y}}(\mathbf{x}) \qquad \text{where} \qquad \hat{\sigma}^2_{\mathbf{y}} = \sigma^2[\kappa(\mathbf{y},\mathbf{y}) - \mathbf{q}_N^\top(\mathbf{y})\mathbf{C}_N^{-1}\mathbf{q}_N^\top(\mathbf{y})],$$

$$\text{and} \qquad \hat{\sigma}^2_{\mathbf{y}}(\mathbf{x}) = \sigma^2[\kappa(\mathbf{y},\mathbf{y}) - \mathbf{q}_{N+1}(\mathbf{y})^\top\mathbf{C}_{N+1}^{-1}\mathbf{q}_{N+1}(\mathbf{y})].$$

The above equations use notation for the GP predictive variance for region r_ν given in (5). The partition inverse equations (Barnett, 1979), for a covariance matrix $\mathbf{C}_{N+1}$ in terms of $\mathbf{C}_N$, gives a means to arrive at a nice expression for $\Delta\sigma^2_{\mathbf{y}}(\mathbf{x})$:

$$\Delta\hat{\sigma}^2_{\mathbf{y}}(\mathbf{x}) = \frac{\sigma^2\left[\mathbf{q}_N^\top(\mathbf{y})\mathbf{C}_N^{-1}\mathbf{q}_N(\mathbf{x}) - \kappa(\mathbf{x},\mathbf{y})\right]^2}{\kappa(\mathbf{x},\mathbf{x}) - \mathbf{q}_N^\top(\mathbf{x})\mathbf{C}_N^{-1}\mathbf{q}_N(\mathbf{x})}. \tag{7}$$

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 $\Delta\sigma^2_{\mathbf{y}}(\tilde{\mathbf{x}}) = 0$. The reduction in predictive variance that would be obtained by adding $\mathbf{x}$ into the data set is calculated by averaging over $\mathbf{y} \in \mathbf{Y}$:

$$\Delta\sigma^2(\mathbf{x}) = |\mathbf{Y}|^{-1} \sum_{\mathbf{y}\in\mathbf{Y}} \Delta\hat{\sigma}^2_{\mathbf{y}}(\mathbf{x}) \tag{8}$$

$\Delta\sigma^2(\mathbf{x})$ is easily approximated using MCMC methods. Compared to ALM, adaptive samples under ALC are less heavily concentrated near the boundaries of partitions. Both provide a ranking of the 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. ALC requires an order of magnitude more computing time than ALM and is more sensitive to the location and region-specific count of candidates, especially $\mathbf{Y}$. While this is generally desirable, for non-uniformly distributed $\mathbf{Y}$, $\Delta\sigma^2(\tilde{\mathbf{x}})$ may be (artificially) magnified closer to high-density $\mathbf{Y}$ regions. McKay et al. (1979) provide a comparison between ALM, ALC, and LH, 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 $\Delta\sigma^2_{\mathbf{y}}(\tilde{\mathbf{x}})$ are taken conditional on samples from $\{\mathcal{T}, \boldsymbol{\theta}\}$. To mitigate the (possibly enormous) expense of sampling $\Delta\sigma^2_{\mathbf{y}}(\tilde{\mathbf{x}})$ 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 D-optimal 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 intense compared to ALC under a full GP. Starting with the predictive variance given in (6), the expected reduction in variance under the linear model is given in (9), below, and averaging over $\mathbf{y}$ proceeds as in (8), above.

$$\Delta\hat{\sigma}^2_{\mathbf{y}}(\mathbf{x}) = \frac{\sigma^2[\mathbf{f}^\top(\mathbf{y})\mathbf{V}_{\tilde{\beta}_N}\mathbf{f}(\mathbf{x})]^2}{1 + g + \mathbf{f}^\top(\mathbf{x})\mathbf{V}_{\tilde{\beta}_N}\mathbf{f}(\mathbf{x})} \tag{9}$$

Appendix A.2 contains details of the derivation. Since only an $m_X \times m_X$ inverse is required, Eq. (9) is preferred over simply replacing $\mathbf{K}$ with $\mathbf{I}(1+g)$ in (7), which requires an $N \times N$ inverse.

3.4 Choosing candidates

We have already discussed how large, i.e., densly gridded, candidates $\tilde{\mathbf{X}}$ can make for computationally intense 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 we have a need for a candidate set of points which are spaced out relative to themselves and also relative to the configurations which have already been sampled.

A sequential D-optimal design may seem like a reasonable approach because it encourages exploration. But traditional D-optimal 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 across the input space. A D-optimal design may not choose candidates in the "interesting" part of the input space, because sampling is high there already. Another disadvantage to D-optimal designs is computational, requiring repeated decompositions of large covariance matrices.

One possible solution to both computational and nonstationary modeling issues is to use treed sequential D-optimal design. That is, a separate sequential D-optimal 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 (such as a genetic algorithm) seems excessive. Finding a local maximum is generally sufficient to get well-spaced candidates. We use a simple stochastic ascent algorithm in each of the $\hat{R}$ regions $\{r_\nu\}_{\nu=1}^{\hat{R}}$ depicted by $\hat{\mathcal{T}}$ which can 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 $N \times N$.

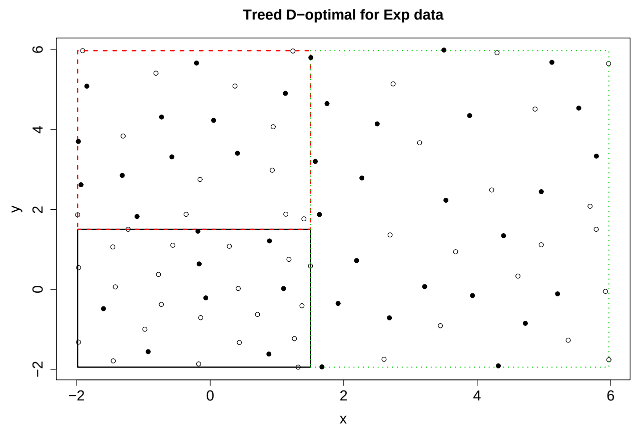

Figure 3: Example of a treed D-optimal design in 2-d. *Open Circles* represent previously sampled locations. *Solid dots* are the candidate design based on $\hat{\mathcal{T}}$, also shown.

Figure 3 shows an example sequential treed D-optimal design for the 2-d Exponential data [Section 4.2], found by simple stochastic search. Input configurations are sub-sampled from a LH design of size 400, and the chosen candidate design is of size $\sim$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). The number of candidates selected in each region is equal to the ceiling of 10% of the number of LH samples therein. Candidates are resonably 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. Since the action of the function is entirely in the first quadrant, we want to continue to sample most heavily in that quadrant. A classical (non-treed) D-optimal design would have chosen fewer points in the first quadrant in order to equalize the density relative to the other three quadrants.

3.5 Implementation methodology

Bayesian adaptive sampling (BAS) proceeds in trials. Suppose 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 $\{\mathbf{x}_i, z_i\}_{i=1}^{N}$. Samples are gathered in accordance with ALM or ALC conditional on $\{\boldsymbol{\theta}, \mathcal{T}\}$, at candidate locations $\tilde{\mathbf{X}}$ chosen from a sequential treed D-optimal design. The candidate queue is 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.

Two implementations of an artificial clustered simulation environment, with a fixed number of agents, were developed in order to simulate the parallel and asynchronous evaluation of input configurations, whose responses finish at random times. One implementation is in `C++` and uses the message passing features of `PVM` (Parallel Virtual Machine) to communicate with the adaptive sampler. The second implementation is in `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 interface.

BAS, as used for the LGBB experiment in Section 5, interfaces with the `Perl` module developed at NASA to submit jobs to their supercomputer. Multi-dimensional responses, as in the LGBB experiment, are treated as independent, i.e., 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. Treating highly correlated physical measurements as independent is a crude approach. However, it still affords remarkable results, and allows the use of `PThreads` to get a highly parallel implementation. Coupled with the producer/consumer model for parallelizing prediction and estimation (Gramacy and Lee, 2006), a factor of $2m_Z$ speedup for $2m_Z$ processors can be obtained. 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 likely not parallelizable. The MAP tree $\hat{\mathcal{T}}$, used for creating sequential treed D-optimal 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 is restarted, or 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 Linear CART model, i.e., the treed LLM, 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 only need 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. 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 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 naive gridding.

4 Illustrative examples

In this section, sequential experimental designs are built for synthetic 1-d and 2-d data with the treed GP LLM as a surrogate model. Section 5 returns to the motivating rocket booster experiment.

4.1 1-d Synthetic Sinusoidal data

Consider synthetic sinusoidal data first used by Higdon (2002), and then augmented by Gramacy and Lee (2006) with a linear region:

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

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 treed D-optimal 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 D-optimally spaced samples were used as an initial design.

The snapshot in the top row of Figure 4 was taken after BAS had gathered a total of 30 samples. BAS recently 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 (8). 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. 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

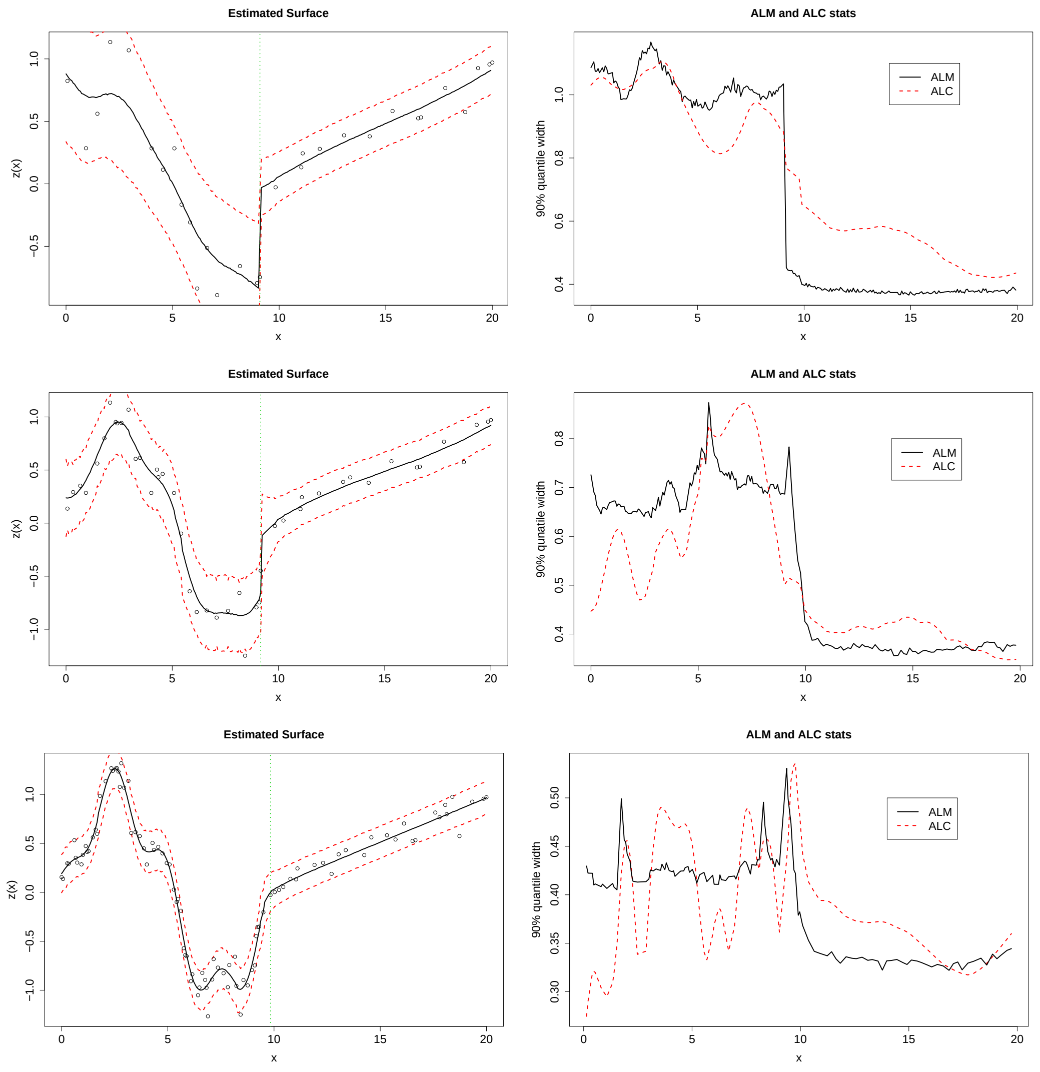

Figure 4: Sine data after 30 *(top)*, 45 *(middle)*, and 97 *(bottom)* adaptively chosen samples. *Left:* posterior predictive mean and 90% quantiles, and MAP partition $\hat{\mathcal{T}}$. *Right:* ALM (black-solid) and ALC (red-dashed).

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 both have high uncertainty near the partition boundary ($\hat{\mathcal{T}}$), but are otherwise do have some disagreement about where to sample next. ALM is larger everywhere on the left than it is on the right. ALC has peaks on the left which are higher than on the right, but its valleys are lower.

In summary, the left panels of Figure 4 track the treed GP surrogate 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 also 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 during the evolution of BAS, it is interesting to note that difference between using ALC and ALM on this data is negligible. This is likely due to the high quality of the treed D-optimal candidates $\tilde{\mathbf{X}}$ 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 data set, but in a more controlled 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 the fact that fewer points are added in the linear region, yielding a sequential design strategy which is two-times more efficient than LH sampling on this data. 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. 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 remaining two columns show surfaces for ALM and ALC, and the single partition of $\hat{\mathcal{T}}$, with samples evenly split between the two regions. Because of mixing over the tree space, one can see in both the ALM and ALC plots that the model also considers a tree with a single split along the other axis.

After 50 adaptive samples have been selected, the situation is greatly improved. The second row of Figure 5 shows a posterior predictive surface with the right main features. ALM and ALC, in the middle and right columns of the figure, agree that the first quadrant is most interesting, and as a result (adaptive) sampling is higher in this region. The dots in Figure 3 illustrate treed D-optimal candidates used during this round. [Notice that the $\mathbf{X}$ 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. The predictive surface looks

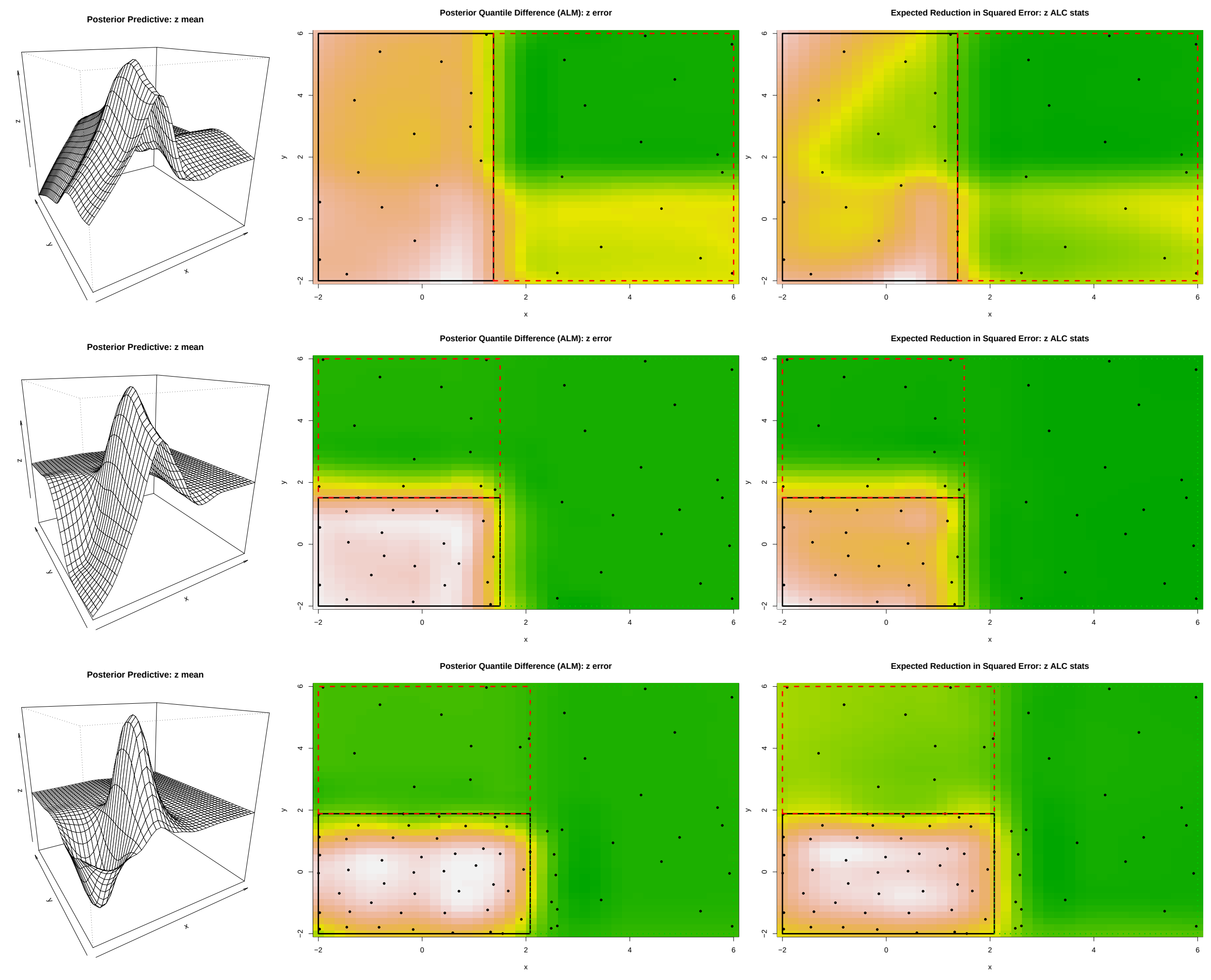

Figure 5: Exponential data after 30 *(top)*, 50 *(middle)* and 80 *(bottom)* adaptively chosen samples. *Left:* posterior predictive mean surface; *Center:* ALM criterion surface; *Right:* ALC criterion surface, with MAP tree $\hat{\mathcal{T}}$ and samples X overlayed.

flawless. More than 54% of the samples are located in the first quadrant which occupies only 25% of the total input space. ALM would choose to sample there next, nearest to the global minima/maxima. ALC, while mostly in agreement, is also concerned about the locations of the split points.

In terms of comparing to other methods such as LH sampling, 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, despite that most of the sampling occurs in the first quadrant and that it is at least two-times more efficient than LH sampling on this data. However, the unlike the sinusoidal data, the exponential data is not defined by step functions. Transitions between partitions are smooth. Thus it takes BAS longer to learn about $\mathcal{T}$, 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 using the ALM algorithm.

5 LGBB CFD experiment

The final experiment is our motivating example. It is the output from computational fluid dynamics simulations of a proposed reusable NASA launch vehicle, called the Langley Glide-Back Booster (LGBB). Simulations involved the integration of the inviscid Euler equations over a mesh of 1.4 million cells (0.8 million cells were used for the supersonic cases). Three input parameters are varied over (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. Each run of the Euler solver on an input triplet takes on the order of 5-20 hours on a high end workstation. 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).

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.

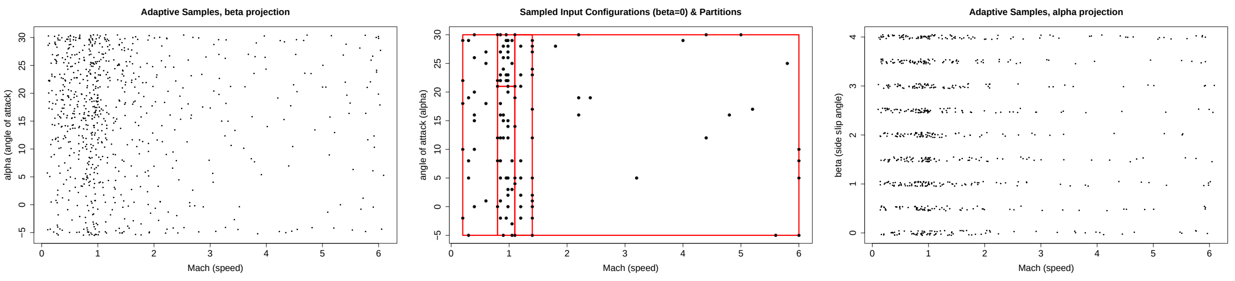

Figure 6: Full set of adaptively sampled configurations projected over beta (side-slip angle; *left*), for fixed beta = 0 (*center*) with MAP partition $\hat{\mathcal{T}}$, and then over alpha (angle of attack; *right*).

Figure 6 shows the 780 configurations sampled by BAS for the LGBB experiment, in two projections, and a subset of those which appear in the beta = 0 slice (middle). The left panel shows locations as a function

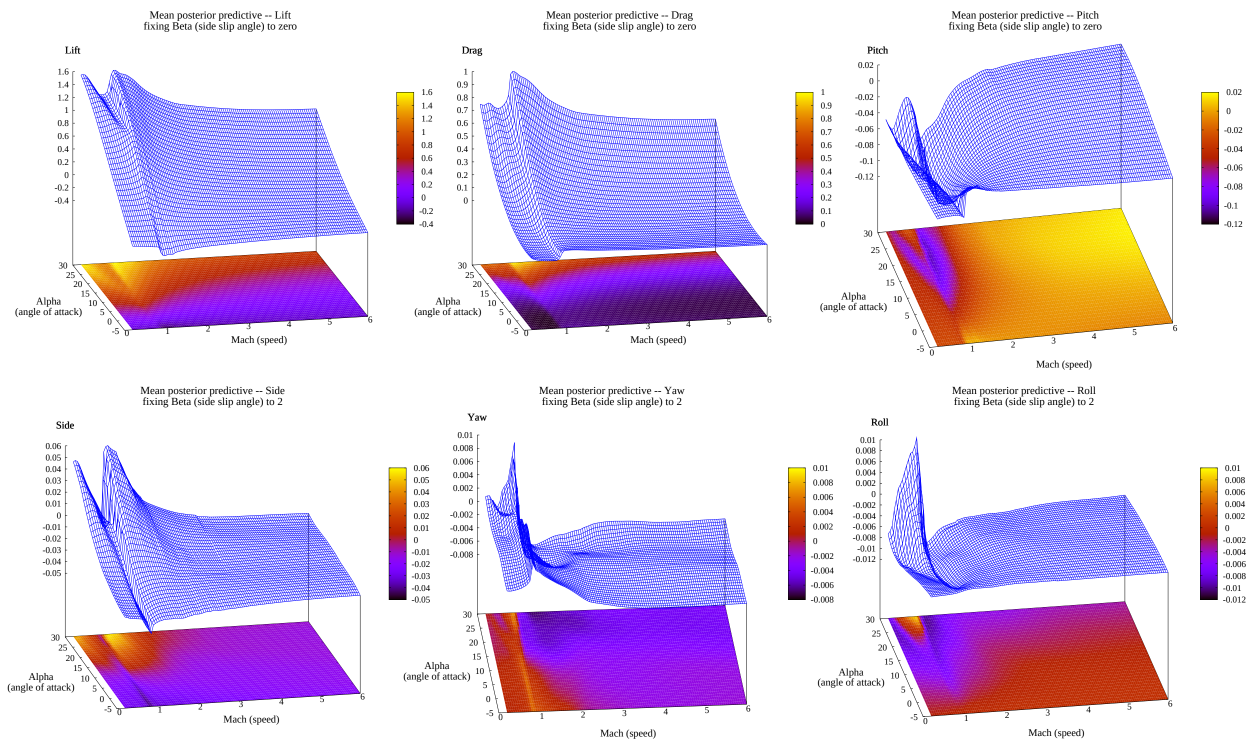

Figure 7: LGBB slice of mean posterior predictive surface for six responses (*lift, drag, pitch, side, yaw, roll*) plotted as a function of Mach (speed) and Alpha (angle of attack) with Beta (side slip angle) fixed at zero.

of Mach (speed) and alpha (angle of attack), projecting over beta (side slip angle); the right panel shows Mach versus beta, projecting over alpha. 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 relatively uniform across all beta settings. A small amount of random noise has been added to the beta values in the plot for visibility purposes.

After samples are gathered, the treed GP model can be used for Monte Carlo estimation of the posterior predictive distribution. The first plot in Figure 7 shows a slice of the lift response, for beta = 0, plotted as a function of Mach and alpha. The center panel of Figure 6 shows the 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 exibit similar characteristics, in that the supersonic cases are tame relative to their subsonic counterparts, with the most interesting region occuring near Mach 1, especially for large angle of attack (alpha). The treed GP model has enabled BAS to key in on this phenomenon and concentrate sampling in these areas (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. In other words, a flexible and adaptive approach was taken, rather than strictly "optimal" one. 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 criteria 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 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 strategy, representing a large improvement on the state-of-the-art of computer experiment methodology at NASA.

Bayesian adaptive sampling (BAS) was illustrated on two nonstationary synthetic data sets. Finally, BAS was implemented on a supercomputer at NASA in order to sequentially design the computer experiment for a proposed reusable launch vehicle. With fewer than one-quarter of the samples used in a previous experiment, BAS was able to produce superior response surfaces by sampling more heavily in the interesting, or challenging, parts of the input space, relying on treed GP model to fill in the gaps in less-sampled regions.

ALM, ALC, and treed D-optimal 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.

Further improvements in computational time may be available. For example, a simple statistical analysis of the experimental apparatus, i.e., the supercomputer, may improve adaptive sampling. Accurate forecasts of how many agents will complete their runs before the next adaptive sampling trial could be used to tune the size of the optimal candidate designs. Configurations which are likely start running before the next round can be incorporated into the design ahead of time, with mean-predictive responses as surrogates, so that future candidates can be focused on other parts of the input space.

For the longer term 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., 2005) generally, or to find boundaries, i.e., contours with large gradients (Banerjee and Gelfand, 2005), 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., 2005), 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.

A Active Learning – Cohn (ALC)

Section A.1 derives the ALC algorithm for a hierarchical Gaussian process and Section A.2 does the same for a linear model.

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 $\mathbf{C}_N^{-1}$:

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C}_N & \mathbf{m} \\ \mathbf{m}^\top & \kappa \end{bmatrix} \qquad \mathbf{C}_{N+1}^{-1} = \begin{bmatrix} [\mathbf{C}_N^{-1} + \mathbf{g}\mathbf{g}^\top \mu^{-1}] & \mathbf{g} \\ \mathbf{g}^\top & \mu \end{bmatrix} \tag{11}$$

where $\mathbf{m} = [C(\mathbf{x}_1, \mathbf{x}), \dots, C(\mathbf{x}_N, \mathbf{x})]$, $\kappa = C(\mathbf{x}, \mathbf{x})$, 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 $\mathbf{C}_{N+1}$. Moreover, the partitioned inverse can be used to compute $\mathbf{C}_{N+1}^{-1}$ with time in $O(n^2)$ rather than the usual $O(n^3)$.

Using notation for a hierarchically specified Gaussian process, in the context of ALC sampling, the matrix which requires an inverse is $\mathbf{K}_{N+1} + \mathbf{F}_{N+1}\mathbf{W}\mathbf{F}_{N+1}^\top$. This matrix is key to the computation of the predictive variance $\hat{\sigma}(\mathbf{x})^2$.

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

(*) 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 (11) gives

$$\mathbf{C}_{N+1}^{-1} = (\mathbf{K}_{N+1} + \mathbf{F}_{N+1}^\top\mathbf{W}\mathbf{F}_{N+1})^{-1} = \begin{bmatrix} [\mathbf{C}_N^{-1} + \mathbf{g}\mathbf{g}^\top\mu^{-1}] & \mathbf{g} \\ \mathbf{g}^\top & \mu \end{bmatrix} \tag{12}$$

where $\mathbf{g} = -\mu\mathbf{C}_N^{-1}\mathbf{q}_N(\mathbf{x})$, and $\mu = (\kappa(\mathbf{x},\mathbf{x}) - \mathbf{q}_N(\mathbf{x})^\top\mathbf{C}_N^{-1}\mathbf{q}_N(\mathbf{x}))^{-1}$ using the most recent definitions of $\mathbf{C}_N$ and $\kappa(\cdot,\cdot)$, see (*).

From here an expression for the key quantity of the ALC algorithm from Seo et al. (2000) can be obtained. It calculates the reduction in variance at a point $\mathbf{y}$ given that the location $\mathbf{x}$ is added into the data:

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

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 (12):

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

Then, another multiplication:

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

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

A.2 For hierarchical (limiting) linear models

Under the (limiting) linear model, computing the ALC statistic is somewhat more straightforward. Starting back at the beginning; now with the predictive variance under the limiting linear model:

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

where $\mathbf{V}_{\tilde{\beta}_{N+1}}$ from (Gramacy and Lee, 2006) includes $\mathbf{x}$, and $\mathbf{V}_{\tilde{\beta}_N}$ does not. Expanding out $\mathbf{V}_{\tilde{\beta}_{N+1}}$:

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

The Sherman-Morrison-Woodbury formula (Bernstein, 2005), where $\mathbf{V} \equiv \mathbf{f}^\top(\mathbf{x})(1+g)^{-\frac{1}{2}}$ and $\mathbf{A} \equiv \mathbf{V}_{\tilde{\beta}_N}^{-1}$ 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. (2005). "Boundary Analysis: Significance and Construction of Curvilinear Boundaries." *Journal of the American Statistical Association*. To appear.

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.

Box, G. E. P., Hunter, W. G., and Hunter, J. S. (1978). *Statistics for Experimenters*. New York: Wiley.

Chaloner, K. and Verdinelli, I. (1995). "Bayesian Experimental Design, A Review." *Statistical Science*, 10 No. 3, 273–1304.

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.

Damian, D., Sampson, P. D., and Guttorp, P. (2001). "Bayesian Estimation of Semiparametric Nonstationary Spatial Covariance Structure." *Environmetrics*, 12, 161–178.

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

Fuentes, M. and Smith, R. L. (2001). "A New Class of Nonstationary Spatial Models." Tech. rep., North Carolina State University, Raleigh, NC.

Glover, F. W. and Laguna, M. (1997). *Tabu Search*. 1st ed. Springer. ISBN: 079239965X.

Gramacy, R. B. and Lee, H. K. H. (2006). "Bayesian treed Gaussian process models." Tech. rep., Dept. of Applied Math & Statistics, University of California, Santa Cruz.

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.

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. (2005). "Sequential Kriging Optimization Using Multiple Fidelity Evaluations." *submitted to:* Structural and Multidisciplinary Optimization.

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.

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.

Mitchell, T. (1974). "An Algorithm for the Construction of D-optimal Experimental Designs." *Technometrics*, 16, 203–210.

Mitchell, T. J. and Morris, M. D. (1992). "Bayesian Design and Analysis of Computer Experiments: Two Examples." *Statistica Sinica*, 2, 359–379.

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.

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

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*. Http://www.cs.dartmouth.edu/∼cbk/papers/sdm05.pdf.

Ranjan, P., Bingham, D., and Michailidis, G. (2005). "Sequential Experiment Design for Contour Estimation from Complex Computer Codes." Submitted.

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.

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.