Adaptive exploration of computer experiment parameter spaces

Robert B. Gramacy, Herbert K. H. Lee and William G. Macready
{rbgramacy,herbie}@ams.ucsc.edu, wgm@email.arc.nasa.gov

Many complex phenomena are difficult to investigate directly through controlled experiments. Instead, computer simulation is becoming a commonplace alternative to providing insight into such phenomena. However, the drive towards higher fidelity simulation continues to tax the fastest of 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 article we discuss the problem of fitting a response surface for a computer model when we also have the ability to design the experiment adaptively, updating the experiment as we learn about the model— a task to which we feel the Bayesian approach is particularly well-suited.

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. This is called a *computer experiment.*

High fidelity computer experiments are usually run on clusters of independent computing agents, or processors (e.g. a `Beowulf` cluster). At any given time each agent is working on a single input configuration. Multiple agents allow several input configurations to be run in parallel, starting and finishing at different (even 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 naive approach of running the experiment over a dense grid of possible inputs. However, computationally inexpensive surrogate models can often provide accurate approximations to the simulation, especially in regions of the input space where the response is easily predicted.

For example, NASA is developing a new re-usable rocket booster called the Langley Glide-Back Booster (LGBB). Much of the development is done through computer models. In particular, they are interested in learning about the response in flight characteristics (lift, drag, pitch, side-force, yaw, roll) of the LGBB as a function of three inputs (side slip angle, Mach number, angle of attack). For each input configuration triplet, CFD simulations yield six response outputs. The left panel of Figure 2 shows one of these outputs, lift, as a function of speed and angle of attack. Of note is the large ridge at Mach 1, where the flight abruptly transitions from subsonic to supersonic. While most of the output space is rather smooth, the ridge is clearly not. Thus there is interest in being able to automatically explore this surface, learning about the ridge and spending relatively more effort there than in the smooth regions. 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.

The above experiment is an example of a situation where surrogate models combined with active learning techniques could direct future sampling, dramatically reducing the size of the final experimental design, saving thousands of hours of computing time. Sampling can be focused on input configurations where the surrogate model is least sure of its predicted response, either because the output response is changing significantly or because there are relatively few nearby data points already examined.

The traditional surrogate model used to approximate outputs to computer experiments is the Gaussian process (GP). GPs are conceptually straightforward, easily accommodate prior knowledge in the form of covariance functions, and return a confidence around predictions. In spite of its simplicity, there are three important disadvantages to standard GPs in our 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 applications we have in mind, 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 directly depend on any of the previously observed output responses.

All of these shortcomings may be addressed by partitioning the input space into regions, and fitting separate GPs within each region. Partitioning allows for modeling of non-stationary behavior, and can ameliorate some of the computational demands by fitting models to less data. Finally, a fully Bayesian approach yields uncertainty measures for predictive inference which can help direct future sampling. However, the MCMC required to estimate the parameters of a Bayesian model can be computationally intensive. Careful but thrifty implementation is required to ensure the development of a cost-effective aid in the sequential design of computer experiments.

Bayesian Treed GP Models

A tree model partitions the input space and infers a separate model within each region. Partitioning is accomplished by making (recursive) binary splits on the value of a single variable (e.g., speed > 0.8) so that partition boundaries are parallel to coordinate axes. These sorts of models are often referred to as Classification and Regression Trees (CART). CART has become popular because of its ease of use, clear interpretation, and ability to provide a good fit in many cases. The Bayesian approach is straightforward to apply to tree models, provided that one can specify a meaningful prior for the size of the tree. We follow Chipman et al. (1998) who specify the prior through a tree-generating process. Starting with a null tree (all data in a single partition), the tree $\mathcal{T}$ is probabilistically split recursively with each partition, η, being split with probability $p_{\mathrm{SPLIT}}(\eta, \mathcal{T}) = a(1+q_\eta)^{-b}$ where q_η is the depth of η in $\mathcal{T}$ and a and b are parameters chosen to give an appropriate size and spread to the distribution of trees. We expect a relatively small number of partitions, and choose a and b accordingly.

Extending the work of Chipman et. al (2002), we fit a stationary GP with linear trend independently within each of R regions, $\{r_\nu\}_{\nu=1}^R$, depicted by the tree $\mathcal{T}$. Conditioning on $\mathcal{T}$, the hierarchical generative model we use is:

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

with $\mathbf{F}_\nu = (\mathbf{1}, \mathbf{X}_\nu)$, and $\mathbf{W}$ a $(m_X+1) \times (m_X+1)$ matrix. Constants $\boldsymbol{\mu}, \mathbf{B}, \mathbf{V}, \rho, \alpha_\sigma, q_\sigma, \alpha_\tau, q_\tau$ are known. The GP correlation structure $\mathbf{K}_\nu$ for each partition r_ν is chosen either from the isotropic power family, or separable power family of unknown (random) parameterization. In both cases, the correlation function takes the form $K_\nu(\mathbf{x}_j, \mathbf{x}_k) = K_\nu^*(\mathbf{x}_j, \mathbf{x}_k) + g_\nu \delta_{j,k}$ where $\delta_{\cdot,\cdot}$ is the Kronecker delta function, and K_ν^* is a *true* correlation representative from a parametric family. Priors which encode our belief that the global covariance structure is non-stationary are chosen for parameters to K_ν^* and g_ν.

Most literature on the *Design and Analysis of Computer Experiments* (Santner et al., 2003; Sacks et al., 1989) deliberately omits the nugget parameter (g) on grounds that computer experiments are deterministic. However, there are many reasons why one may wish to study a computer experiment with a model that includes an explicit noise component. In particular, the experiment may, in fact, be non-deterministic. Our collaborators tell us that their CFD solvers are often started with random initial conditions, involve forced random restarts when diagnostics indicate that convergence is poor, and that input configurations arbitrarily close to one another often fail to achieve the same estimated convergence. Thus a conventional GP model without a small-distance noise process (e.g. a nugget) can be a mismatch to such inherently non-smooth data.

The data $\{\mathbf{X}, \mathbf{t}\}_\nu$ 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 available in Gramacy et al. (2004). The predicted value of $y(\mathbf{x} \in r_\nu)$ is normally distributed with mean and variance

$$\hat{y}(\mathbf{x}) = \mathbf{f}^\top(\mathbf{x})\tilde{\boldsymbol{\beta}}_\nu + \mathbf{k}_\nu(\mathbf{x})^\top \mathbf{K}_\nu^{-1}(\mathbf{t}_\nu - \mathbf{F}_\nu \tilde{\boldsymbol{\beta}}_\nu),$$
$$\hat{\sigma}(\mathbf{x})^2 = \sigma_\nu^2[\kappa(\mathbf{x},\mathbf{x}) - \mathbf{q}_\nu^\top(\mathbf{x})\mathbf{C}_\nu^{-1}\mathbf{q}_\nu(\mathbf{x})],$$

where $\tilde{\boldsymbol{\beta}}_\nu$ is the posterior mean estimate of $\boldsymbol{\beta}_\nu$, and

$$\mathbf{C}_\nu^{-1} = (\mathbf{K}_\nu + \mathbf{F}_\nu \mathbf{W}\mathbf{F}_\nu^\top/\tau^2)^{-1}$$
$$\mathbf{q}_\nu(\mathbf{x}) = \mathbf{k}_\nu(\mathbf{x}) + \tau^2 \mathbf{F}_\nu \mathbf{W}_\nu \mathbf{f}(\mathbf{x})$$
$$\kappa(\mathbf{x},\mathbf{y}) = K_\nu(\mathbf{x},\mathbf{y}) + \tau^2 \mathbf{f}^\top(\mathbf{x})\mathbf{W}\mathbf{f}(\mathbf{x})$$

with $\mathbf{f}^\top(\mathbf{x}) = (1, \mathbf{x}^\top)$, and $\mathbf{k}_\nu(\mathbf{x})$ a $n_\nu-$vector with $\mathbf{k}_{\nu,j}(\mathbf{x}) = K_\nu(\mathbf{x},\mathbf{x}_j)$, for all $\mathbf{x}_j \in \mathbf{X}_\nu$. Notice that $\hat{\sigma}(\mathbf{x})^2$ does not directly depend on the observed responses $\mathbf{t}_\nu$.

Integrating out dependence on the tree structure $\mathcal{T}$ is accomplished by reversible-jump MCMC (RJ-MCMC). We implement the tree operations *grow, prune, change*, and *swap* similar to those in Chipman et al. (1998). To keep things simple, proposals for new parameters—via an increase in the number of partitions R—are drawn from their priors, thus eliminating the Jacobian term usually present in RJ-MCMC. We augment the *swap* operation with a *rotate* option to improve mixing. Rotations are typically used in balancing the Binary Search Tree (BST) data structure, as in Red Black Trees, to keep the depth of the BST small. In the context of Bayesian CART, rotations can be useful when the proposed parent-child pair to be swapped happen to split on the same dimension, i.e., when the child split is a recursive split. A *swap* in this case always fails, but a *rotate* operation would almost always accept and possibly lead to the pruning of higher-level recursively-partitioned regions.

We coded the treed GP model in a mixture of `C` and `C++`: `C++` for the tree data structure ($\mathcal{T}$) and `C` for GP at each leaf of $\mathcal{T}$. The `C` code can interface with either standard (platform-specific) `Fortran` `BLAS/Lapack` libraries for the essential linear algebra necessary to estimate the parameters of the GP, or link to those automatically configured for fast execution on a variety of platforms via the `ATLAS` library (Whaley & Petitet, 2004). We found that in most cases the `ATLAS` implementation was significantly faster than standard `BLAS/Lapack`.

After conditioning on the tree and parameters ($\{\mathcal{T}, \boldsymbol{\theta}\}$) we noticed that prediction could be parallelized by using a producer/consumer model. This allowed us to use `PThreads` in order to take advantage of multiple processors, and get speed-ups of at least a factor of two. This is particularly relevant since dual processor workstations and multi-processor servers are becoming a commonplace in modern research labs. We have also had some preliminary success with parallelizing the sampling of the posterior of $\boldsymbol{\theta}|\mathcal{T}$ by drawing each of the $\{\theta_\nu\}_{\nu=1}^R$ in parallel. However, the speed-up in this second case was less impressive. To ice the cake we wrapped the whole thing up in an intuitive `R` interface (R Development Core Team, 2004). Compared to existing methods, the above approach lead to an extremely fast implementation of non-stationary GPs.

As an empirical proof of concept, we fit our treed GP model to the Motorcycle Accident Dataset (Silverman, 1985)— a classic non-stationary data set used in recent literature (Rasmussen & Ghahramani, 2002) to demonstrate the success of non-stationary models. The goal of the data set is to model the acceleration force on the head of a motorcycle rider as a function of time in the first moments after an impact. In addition to being non-stationary, the data has input-dependent noise.

Figure 1 shows the data, and the fit given by our treed GP model. The top pane shows the estimate of the surface with 90%-quantile error bars; the bottom pane shows the difference in quantiles. Vertical lines on both panes show a typical partition ($\mathcal{T}$). The error bars, and estimated error spread, can give insight into the uncertainty in the posterior distribution for $\mathcal{T}$. The sharp rise in estimated variance from the leftmost to the center region contrasted with a more gradual, stepwise, decent in variance from the center to the rightmost region shows that there was higher posterior certainty in the left split than the right one. The average number of partitions in the posterior for $\mathcal{T}$ was 3.11.

These results are quite different from those reported by Rasmussen & Ghahramani (2002). In particular, their error-bars in the leftmost region seem too large relative to the center and rightmost regions. They use a what they call an "infinite" mixture of GP "experts"; technically a Dirichet Process mixture of GPs. They report that the posterior distribution uses between 3 and 10 experts to fit this data (which they admit has "roughly" three regions), and even 10-15 experts have considerable mass. Our treed GP model almost always partitions into three regions.

On speed grounds, the treed GP is also a winner. Rasmussen & Ghahramani (2002) report that their

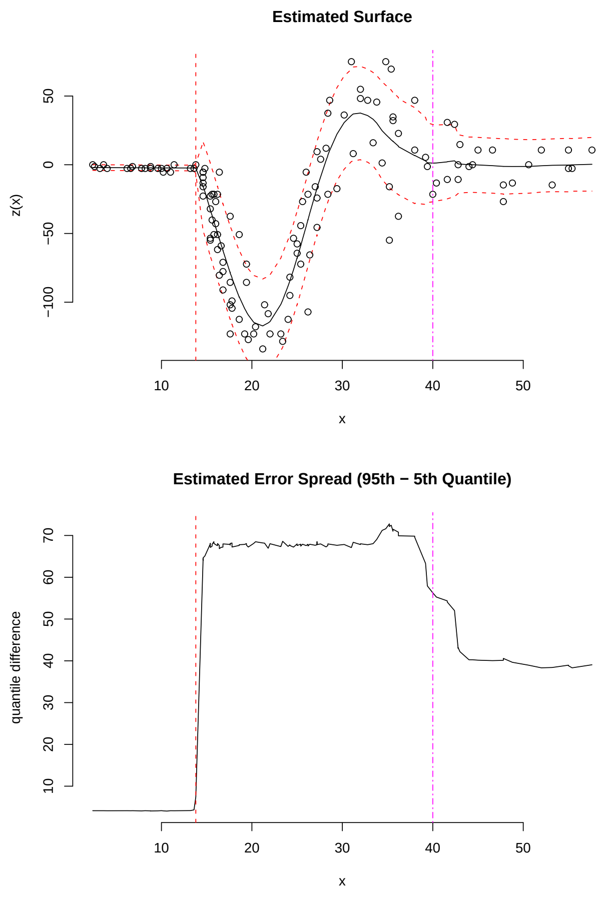

Figure 1: Motorcycle Data, fit by our treed GP.

code implementing the mixture of GP experts model on the motorcycle data took roughly one hour on a (single) 1 GHz Pentium. Our treed GP code took less than three minutes on a (single) 1.8 GHz Athalon.

Adaptive Sampling

In the world of Machine learning, adaptive sampling would fall under the blanket of a research focus called *active learning*. Active learning techniques have been proposed in areas such as computational drug design/discovery by aiding in the search for compounds that are active against a biological target (Warmuth et al., 2003). However, we are not aware of any other active learning algorithms that use non-stationary modeling to help select small designs.

In the statistics community, the traditional approach to sequential data solicitation goes under the general heading of *(Sequential) Design of Experiments* (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 (with parameters $\boldsymbol{\theta}$) 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 what are called $A-$optimal designs. An excellent review of Bayesian approaches to the design of experiments is contained in Chaloner & Verdinelli (1995). Other approaches used by the statistics community include space-filling designs: e.g. max-min distance and Latin Hypercube (LH) designs (Santner et al., 2003).

A hybrid approach to designing experiments employs active learning techniques. The idea is to choose a set of candidate input configurations $\tilde{\mathbf{X}}$ (say, a $D-$optimal or LH design) and an active learning rule for determining the order in which they should be added into the design. For example, consider an approach which maximizes the information gained about model parameters by selecting the location $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ which has the greatest standard deviation in predicted output. This approach has been called ALM for Active Learning–Mackay, and has been shown to approximate maximum expected information designs (MacKay, 1992). Given its simplicity this is the method we explored first. MCMC posterior predictive samples provide a convenient estimate of location-specific variance; namely the width of predictive quantiles.

An alternative algorithm is to select $\tilde{\mathbf{x}}$ minimizing the resulting expected squared error averaged over the input space (Cohn, 1996), called ALC for Active Learning–Cohn. Conditioning on $\mathcal{T}$, the reduction in variance given that the location $\tilde{\mathbf{x}}$ is added into the data (averaged over a reference set $\tilde{\mathbf{Y}}$) is:

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

which is easily computed using MCMC methods. A comparison between ALC and ALM using standard GPs appears in (Seo et al., 2000).

Given these two hybrid approaches to sequential design, constructing a list of input configurations to send to available computing agents is simply a matter of sorting candidate locations ranked via either ALM or ALC. That way, the most informative locations are first in line for simulation when agents become available.

We developed two implementations of an artificial clustered simulation environment with a fixed number of agents 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 used by NASA to drive their current experimental design software.

Experimental Results

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 the model is estimated for data $\{\mathbf{X}_i, t_i\}_{i=1}^N$. In accordance with the ALM algorithm, MCMC predictive quantiles are gathered based on sampled $\{\boldsymbol{\theta}, \mathcal{T}\}$, and sorted. After refreshing the sorted list of candidates, BAS gathers finished and running input configurations 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.

For the LGBB experiment we chose to model K^* as a member of the separable Power family—a practice common in the Design of Computer Experiments community (Santner et al., 2003)— so that correlation in speed can decay at a rate different than correlation in angle of attack. Since our current experiments are based on pre-calculated pairs of input configurations and responses delivered by NASA, candidates (for now) must be chosen via random-subsample from the available data. To deal with multiple responses we ran six independent copies of BAS, and pooled the ALM-based uncertainty estimates. Treating highly correlated physical measurements as independent is a crude approach, however it still affords remarkable results, and allows us to again employ `PThreads` to get a highly parallel implementation. Coupled with the producer/consumer model for parallelizing prediction and estimation we notice nearly a factor of $2M$ speedup for $2M$ processors, where M dimension of the response space. Co-kriging and other approaches to modeling multivariate (correlated) responses is part of our future work.

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, we take advantage of the trial nature of adaptive sampling: 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 we find that (empirically) the chain(s) explore the posterior of tree space nicely. Random pruning represents a compromise between restarting and initializing the tree-chain at a well-chosen starting place. We find that with this *tree inertia* we can usually afford shorter burn-in of the MCMC at the beginning of each trial.

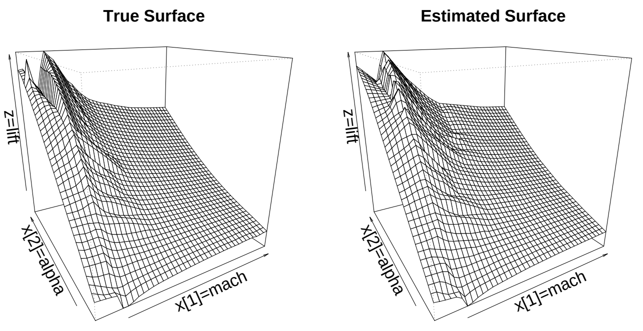

Figure 2: *Left:* true surface based on ∼3000 data points. *Right:* fitted surface based on 750 adaptive samples over 100 trials. (beta = 0)

We turn now to an analysis of results obtained by using BAS on the LGBB data. The left side of Figure 2 shows one of the six outputs (lift) plotted as a function of speed (Mach) and angle of attack (alpha) based on the full design of more than 3000 input configurations. The third input, side slip angle (beta), is fixed at zero. A fitted surface, based upon 750 total samples obtained over 100 BAS trials, is shown on the right side of Figure 2. Configurations gathered using BAS (for beta = 0) are shown in the Figure 3. Also shown in Figure 3 is a representative sample of the partitions obtained by integrating over the tree $\mathcal{T}$. BAS has the desired behavior in that it fits different models around and on either side of the Mach 1 regions, and focuses most of the adaptive sampling around Mach 1. Further partitioning and sampling occurs for large angle of attack (alpha) near Mach 1 as indeed the response there is changing most rapidly.

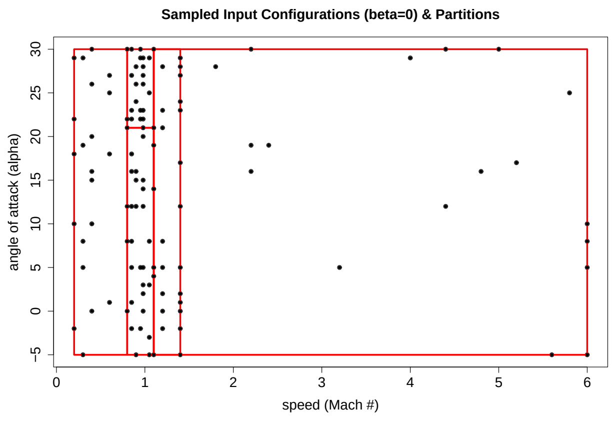

Figure 3: Adaptively sampled configurations (beta = 0).

Visually, there is little difference between the true surface (left) and the estimated surface (right) shown in Figure 2. However, BAS requires fewer than 1/4 as many samples compared to a simple gridding. Relative to the computing time needed to evaluate the each response (5-20 hours), the execution of BAS was negligible, saving thousands of hours of computing time.

Notice that the experiment was set up such that the computing time of each BAS trial does not affect the rate of sampling. Rather, a slow BAS would simply incorporate sampled responses and re-sort candidates less often compared to a faster BAS, leading to a less optimal sequential sample, but always one better than obtained by naive gridding. Future work will focus on obtaining speedups by employing time-saving approximations: e.g. iterative and sparse (via tapering) methods for quickly inverting large covariance matrices, empirical Bayes alternatives to full MCMC, and priors on $\mathcal{T}$ which prefer smaller partitions.

For further analysis on the LGBB experiment, experiments on other data, and comparisons with other approaches, the interested reader is referred to a paper we presented at ICML 2004 (Gramacy et al., 2004). Our future work includes running a live experiment on the NASA supercomputers.

In conclusion, creating a surrogate model for computer experiments is a problem that will continue to be of interest, as additional computing resources are put toward more accurate simulations rather than faster results. The Bayesian approach allows a natural mechanism for creating a sequential design based on the current estimated uncertainty.

References

Chaloner, K., & Verdinelli, I. (1995). Bayesian experimental design, a review. *Statistical Science*, *10 No. 3*, 273–1304.

Chipman, H., George, E., & McCulloch, R. (1998). Bayesian CART model search (with discussion). *Journal of the American Statistical Association*, *93*, 935–960.

Chipman, H. A., George, E. I., & McCulloch, R. E. (2002). Bayesian treed models. *Machine Learning*, *48*, 303–324.

Cohn, D. A. (1996). Neural network exploration using optimal experimental design. *Advances in Neural Information Processing Systems* (pp. 679–686). Morgan Kaufmann Publishers.

Gramacy, R. B., Lee, H. K. H., & Macready, W. (2004). Parameter space exploration with Gaussian process trees. *Proceedings of the International Conference on Machine L earning* (pp. 353–360). Omnipress & ACM Digital Library.

MacKay, D. J. C. (1992). Information-based objective functions for active data selection. *Neural Computation*, *4*, 589–603.

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.

Rasmussen, C. E., & Ghahramani, Z. (2002). Infinite mixtures of Gaussian process experts. *Advances in Neural Information Processing Systems*. MIT Press.

Sacks, J., Welch, W. J., Mitchell, T. J., & Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science*, *4*, 409–435.

Santner, T. J., Williams, B. J., & Notz, W. I. (2003). *The design and analysis of computer experiments*. New York, NY: Springer-Verlag.

Seo, S., Wallat, M., Graepel, T., & Obermayer, K. (2000). Gaussian process regression: Active data selection and test point rejection. *Proceedings of the International Joint Conference on Neural Networks IJCNN 2000* (pp. 241–246). IEEE.

Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric curve fitting. *Journal of the Royal Statistical Society Series B*, *47*, 1–52.

Warmuth, M. K., Liao, J., Ratsch, G., Mathieson, M., Putta, S., & Lemmen, C. (2003). Support vector machines for active learning in the drug discovery process. *Journal of Chemical Information Sciences*, *43(2)*, 667–672.

Whaley, R. C., & Petitet, A. (2004). ATLAS (Automatically Tuned Linear Algebra Software). http://math-atlas.sourceforge.net/.