Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

GRADIENT BASED CRITERIA FOR SEQUENTIAL DESIGN

Collin B. Erickson
Bruce E. Ankenman
Matthew Plumlee

Northwestern University
2145 Sheridan Road
Evanston, IL 60208, USA

Susan M. Sanchez

Naval Postgraduate School
1 University Circle
Monterey, CA 93943, USA

ABSTRACT

Computer simulation experiments are commonly used as an inexpensive alternative to real-world experiments to form a metamodel that approximates the input-output relationship of the real-world experiment. While a user may want to understand the entire response surface, they may also want to focus on interesting regions of the design space, such as where the gradient is large. In this paper we present an algorithm that adaptively runs a simulation experiment that focuses on finding areas of the response surface with a large gradient while also gathering an understanding of the entire surface. We consider the scenario where small batches of points can be run simultaneously, such as with multi-core processors.

1 INTRODUCTION

There is a vast literature on design and analysis of computer experiments, e.g., Sacks et al. (1989), Santner et al. (2003), Forrester et al. (2008). In the general setup, there is a function or simulation that can be evaluated at any point, but each evaluation is expensive. After the experiment, a metamodel (also called an emulator or surrogate model) is created that approximates the true response function and is much faster to evaluate. Traditional experiment designs, or one-shot designs, select an entire list of inputs where the computer model will be evaluated before any data is collected.

Sequential experiment designs allow for design points to be added to the existing design indefinitely, and are designed to have favorable properties at any stopping point. For example, Loeppky et al. (2010) propose a batch sequential design that adds design points using orthogonal array based Latin hypercubes. Adaptive sequential experiment designs are sequential designs that add design points to the existing design based on the data obtained from the existing design in order to achieve some objective.

Existing methods for adaptive experiments can generally be classified under two separate categories: objective seeking and global fitting. Objective seeking experiments aim to find a single point that optimizes a function, and the most celebrated method is the expected improvement algorithm (Jones et al. 1998; Vazquez and Bect 2010). Global fitting methods attempt to fit a model that estimates a surface over its entire domain. A common goal for global fitting is to minimize the average squared error over the surface. Thus, new points are likely to be selected where the model uncertainty is highest. For example, Gramacy and Lee (2009) use treed Gaussian processes to fit a model and select points with either the highest standard deviation in predicted output or to maximize the expected reduction in squared error averaged over the surface. There are some hybrid algorithms that mix global fitting and optimization, such as the sequential minimum energy design of Joseph et al. (2015), which spaces out points to minimize the energy using a charge function. Kim et al. (2017) extends this work when the number of batches of points to be run is known before beginning the experiment for response functions with output in $[0,1]$.

In our setting, we posit that practitioners often try to find a general understanding of the entire response surface with some focus on interesting areas. Instead of defining interesting to be an optimum, we consider

situations where steep areas are the primary focus. That is, a user would like to understand the function in areas where the derivative of the computer model is large (in absolute value). Identifying where the derivative is large allows the user to identify regions where small changes in the inputs can result in relatively large changes in the output. Finding where the gradient is large may also let the user know where the output is not robust, allowing them to avoid unstable input regions. While our problem is different than those above, the appropriate balance of exploration and exploitation remains important. Our goal is that at the end of our adaptive approach, we can build a metamodel that behaves well throughout the surface, but does a bit better where the derivative is large.

This paper is organized as follows. Section 2 describes the general setting for sequential computer experiments. Our contribution for focusing on areas where the derivative is large is detailed in Section 3. Section 4 specifies the Gaussian process modeling framework for carrying out our proposed algorithm. Section 5 outlines the complete algorithm we propose. A graphical example and results from numerical experiments are illustrated in Section 6, demonstrating the efficacy of the proposed algorithm. We conclude the paper with a discussion in Section 7.

2 SETTING

Label the computer model $y(\cdot)$ that takes d-dimensional input point in some input space $\mathscr{X}$ and maps it to a scalar output. This function is deterministic, thus repeated evaluations using the same inputs will yield the same outputs. We assume that after evaluating the computer model at some locations, we can create a metamodel $\hat{y}(\cdot)$ for the entire surface via Gaussian process prediction (Sacks et al. 1989). Moreover, metamodels built with Gaussian process technology also tend to provide a pointwise estimate of the squared error between the $y(\cdot)$ and $\hat{y}(\cdot)$, labeled $\widehat{\sigma}^2(\cdot)$.

Experiment designs for computer models seeks to build the best possible emulator by choosing points that minimize a design criterion (Santner et al. 2003). Let S denote our set of design points. Our metamodel then can be leveraged to provide an estimate of the squared prediction error anywhere in $\mathscr{X}$, given the outputs observed at each point in this design; we label this $\widehat{\sigma}^2(\cdot|S)$. One of the first design criteria used for sequential experiments was the integrated estimated squared prediction error given by

$$\int_{\mathscr{X}} \widehat{\sigma}^2(x|S)\mathrm{d}x.$$

In modern computing environments, simulations can often be run simultaneously on different processors or nodes in a cluster. Thus we consider the case where n input points can be evaluated simultaneously as a batch, and all take a similar amount of time to be evaluated. We then have the power to choose which points to include in each batch S_1, S_2, S_3, …, S_b, …. After stage b, we have collected the response from the computer model at all locations in S_1, S_2, S_3, …, S_b. The challenge in this sequential setting is how to select the next batch of input points, labeled S_{b+1}. To compact our notation in the sequential setting, let

$$\widehat{\sigma}_b^2(x|S) = \widehat{\sigma}^2(x| \underbrace{S_1 \cup S_2 \cup \cdots \cup S_b \cup S}_{\text{all points including new points}}).$$

In the sequential setting, the standard method of improvement would then be to select the next points as

$$S_{b+1} = \arg\min_S \int_{\mathscr{X}} \widehat{\sigma}_b^2(x|S)\mathrm{d}x.$$

We define $\Psi_b(S)$ as our general criteria such that

$$S_{b+1} = \arg\min_S \Psi_b(S).$$

Thus, the vanilla version of sequential design has

$$\Psi_b(S) = \int_{\mathscr{X}} \widehat{\sigma}_b^2(x|S)\mathrm{d}x. \tag{1}$$

The approach in (1) is adequate approach for building a metamodel (emulator) in a sequential setting if all points in the space are equally valued. We generalize this approach in Section 3.

3 VALUE ENHANCED SEQUENTIAL DESIGN

This article's primary contribution relates to improving sequential design approaches when the objective is more general than building a metamodel that is equally accurate over the input space. Instead, we would like to have improved prediction in areas where the function has interesting behavior. This section will motivate and describe our value enhanced sequential design.

3.1 The Univariate Input Case

If our function has a single dimension, consider the criterion that looks like

$$\Psi_b(S) = \int_{\mathscr{X}} \left(\frac{\mathrm{d}y}{\mathrm{d}x}(x)\right)^2 \widehat{\sigma}_b^2(x|S)\mathrm{d}x, \tag{2}$$

where the square of the derivative of the function serves as the value function for the criterion. Clearly, this criterion would build a metamodel with improved predictive performance where the function has a great deal of change.

Unfortunately, this approach in untenable in practice because it depends on the unknown function y. But, at a step b, we are privy to a metamodel labeled $\hat{y}_b(\cdot)$ built with the data observed using the first b evaluations. A naive approach would be simply to use the metamodel to provide a "plug-in" value function to (2), obtaining

$$\Psi_b(S) = \int_{\mathscr{X}} \left(\frac{\mathrm{d}\hat{y}_b}{\mathrm{d}x}(x)\right)^2 \widehat{\sigma}_b^2(x|S)\mathrm{d}x. \tag{3}$$

This approach is misguided since the derivative of the metamodel does not account for the estimation error in the metamodel. For example, a region with sparse data may have a flat predicted line, which has a derivative of zero, not because the metamodel has evidence that it is flat, but rather because it does not have much information in that region. In that case, using the criteria in (3) would cause an adaptive sequential algorithm to allocate no points in regions with sparse data. This could result in poor performance. However, the metamodel in this case would give large error estimates in regions of sparse data along with its predictions. These large error estimates tell the user that the gradient may be large, even though the predicted gradient is small.

Thus we consider a new approach, which considers not just the estimated value function, but also the potential incorrectness in the value function estimation. Say that our metamodel also provides $\hat{v}_b(x)$, an estimate of the variance of the value function estimator at the point x, for all x. In other words,

$$\hat{v}_b(x) \text{ estimates } \left(\frac{\mathrm{d}\hat{y}_b}{\mathrm{d}x}(x) - \frac{\mathrm{d}y}{\mathrm{d}x}(x)\right)^2.$$

We now adopt the criterion

$$\Psi_b(S) = \int_{\mathscr{X}} \left(\left(\frac{\mathrm{d}\hat{y}_b}{\mathrm{d}x}(x)\right)^2 + \hat{v}_b(x)\right) \widehat{\sigma}_b^2(x|S)\mathrm{d}x. \tag{4}$$

The concept behind this approach is that at points where we can predict the value function well, then $\hat{v}_b(x)$ will be small and we will place points according to the estimated value function. Otherwise, the criterion will encourage the design to place points in the regions where the value function is quite unknown. Consider (4) to be the integration of the weighted estimated squared error at stage b, where the weight is given by

$$\hat{\omega}_b(x) = \left(\frac{\mathrm{d}\hat{y}_b}{\mathrm{d}x}(x)\right)^2 + \hat{v}_b(x).$$

3.2 A Simple Illustration in One Dimension

Figure 1 demonstrates the proposed criterion and how it differs from other criteria. Suppose we have a simulation that takes one dimensional input in $[0,1]$ and gives output as shown by the solid line in Figure 1a. Our goal is to estimate this function, but with higher accuracy where the gradient is large, i.e., for $x > 0.6$. However, we do not know where the gradient is large until we have gathered data and used the data to model the unknown function. Furthermore, suppose each simulation evaluation is expensive, meaning we must be efficient in choosing which points to evaluate.

We begin by selecting a few well-spaced points. Suppose we start with $x = 0$, 0.7, and 1, and evaluate the simulation at these points. We fit a metamodel to the resulting data that gives a predicted value along with a 95% confidence interval over the entire interval $[0,1]$, as shown in Figure 1a. The prediction interval is minuscule near the data points, is largest far away from data points, and overall does a good job of capturing the true function. Our task now is to decide which point to evaluate next to best reach our goal.

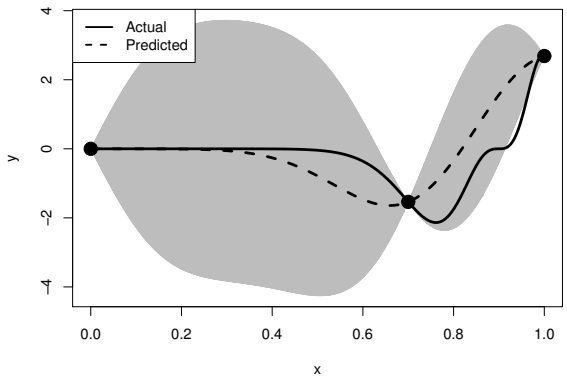
To make this decision we can use one of the three criteria functions discussed above, and select the point that minimizes the criterion. Figure 1b shows the criteria functions discussed above under the current scenario. For every x value in $[0,1]$, each trace shows the value of one of the criteria if a new design point were to be added at x. The y-axis is unlabeled since the criteria are on different scales and only the relative values matter. The IMSE criterion (solid) is smallest near $x = 0.35$ since adding a point in that region will reduce the error estimate over that large, unexplored section. The criterion using the square of the plug-in estimate of the gradient (dashed), shown in (3), is much lower on the right side of $x = 0.7$ since the gradient is estimated to be largest there. Thus adding a point in that area will improve the prediction accuracy in that important region. Our proposed criterion (dotted), shown in (4), is more similar to the IMSE criterion since there is high uncertainty in the metamodel. Note that all criteria have spikes near the three design points since those points would provide no new information in the deterministic setting.

Now suppose that we evaluate the simulation at four additional points, $x = 0.2$, 0.4, 0.6, and 0.83. The updated metamodel is shown in Figure 1c. Incorporting this additional data leads to an estimated function that is closer to the true function and to smaller error estimates, as we would expect. Again, our task is to use this updated model to decide which point to evaluate next. The corresponding criteria functions are shown in Figure 1d. The IMSE criterion would still select a point with x near 0.2 because the points are further spread out in that region. The plug-in criterion is nearly the same as before since the gradient is still estimated to be much larger for $x > 0.6$. Now the proposed criterion much more closely matches the plug-in criterion, and would select a point near 0.9 to be evaluated next.

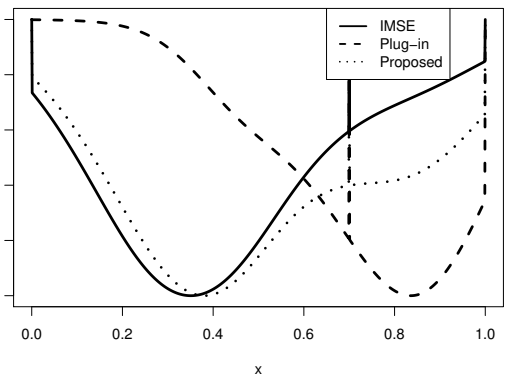
This simple example demonstrates that our proposed criterion behaves like the IMSE criterion when there is high metamodel uncertainty, since the gradient estimate will have high uncertainty, too. But as the metamodel and the gradient estimate become more accurate, it will be more likely to recommend points where the gradient is larger. We can think of this as an exploration-exploitation trade-off: when the uncertainty is high the proposed criterion will advocate for exploration, but as the estimation error shrinks it will advocate for focusing on areas where the gradient is large.

3.3 The Multivariate Input Case

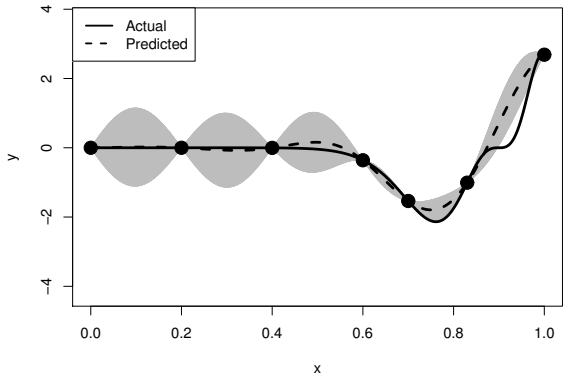
Let $g(x) = \nabla y(x)$ denote the gradient of y. In the one dimensional case, the gradient is a scalar and acts as a weighting function in the criteria functions. In multiple dimensions, the gradient is a vector and cannot



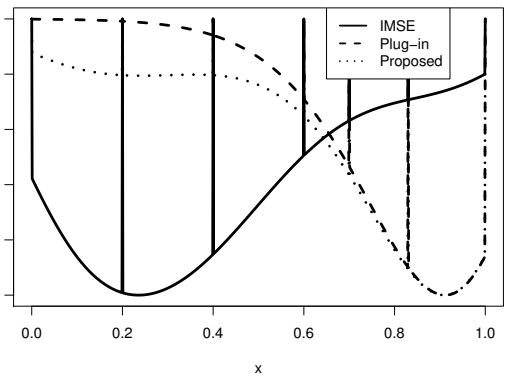
(a) The true function (solid) along with the metamodel (dashed) and its 95% prediction interval (dotted) for model with three data points.



(b) Three criteria for selecting the next point based on the metamodel in Figure 1a: IMSE (solid), plug-in method (dashed), and proposed method (dotted).



(c) The updated model after four new data points added to the initial three.



(d) The criteria based on the model in Figure 1c.

Figure 1: Simple 1-D example demonstrating the differences between criteria.

be used as a value function in this manner. However, the gradient can be converted to a scalar using a norm function. The value function is a function not only of x, but also of the true function y, through its gradient. Thus the value function, which we will write as $\omega(x)$, is

$$\omega(x) = \|g(x)\|^2 = g(x)^T g(x).$$

If the gradient were known, then

$$\Psi_b(S) = \int_{\mathscr{X}} \|g(x)\|^2 \, \widehat{\sigma}_b^2(S) \mathrm{d}x$$

could be used as a criterion. But, as in the one dimensional case, we must use an estimate of the gradient, $\hat{\omega}_b(x)$, obtained from the available data. The proposed criterion for multiple dimensions is

$$\Psi_b(S) = \int_{\mathscr{X}} \hat{\omega}_b(x) \widehat{\sigma}_b^2(S) \mathrm{d}x. \tag{5}$$

Note that the relative scaling of the input variables affects the gradient. Thus care must be taken in placing the input variables on appropriate scales. This is an engineering decision that must be made before conducting any of this analysis. For the examples below, we assume that scaling the input ranges from 0 to 1 is a good relative scaling. This completes the description of the approach.

4 GAUSSIAN PROCESS ESTIMATION

We now provide the background necessary to compute the functions $\hat{y}_b(x)$, $\widehat{\sigma}_b(x|S)$, and $\hat{\omega}_b(x)$ given a collection of data. For this section, say that our data at batch b consists of N input points labeled $x_1,\ldots,x_N$ and corresponding responses labeled $y_1,\ldots,y_N$. The vector of responses is collected into a vector

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}.$$

4.1 Gaussian Process Prediction

A Gaussian process is characterized such that the output from any set of input points has a multivariate normal distribution. Since these derivations closely follow Sacks et al. (1989), our discussions will be terse except when new material needs to be presented.

The surface is modeled as a Gaussian process with mean 0, a constant variance ϕ^2, and a correlation function labeled $r(\cdot,\cdot)$. The correlation function maps any two points in the input space to the correlation between those two outputs.

Under the Gaussian process model the distribution of the outputs, $\boldsymbol{y}$, is multivariate normal with mean $\boldsymbol{0}$, where $\boldsymbol{0}$ is the N-length zero vector. The covariance matrix of this multivariate normal distribution is

$$\phi^2 \boldsymbol{R} = \phi^2 \begin{pmatrix} r(x_1,x_1) & \cdots & r(x_1,x_N) \\ \vdots & \ddots & \vdots \\ r(x_N,x_1) & \cdots & r(x_N,x_N) \end{pmatrix}.$$

Say we are curious about a point in the input space x. Then the best linear unbiased predictor (BLUP) of $y(x)$, x being any point in the input space, is given by

$$\hat{y}(x) = \boldsymbol{r}(x)^T \boldsymbol{R}^{-1} \boldsymbol{y}, \text{ where } \boldsymbol{r}(x) = \begin{pmatrix} r(x,x_1) \\ \vdots \\ r(x,x_N) \end{pmatrix}.$$

The associated estimate of the squared error of $\hat{y}$ at $\boldsymbol{x}$ is

$$\widehat{\sigma}^2(\boldsymbol{x}) = \phi^2 \left[1 - \boldsymbol{r}(x)^T \boldsymbol{R}^{-1} \boldsymbol{r}(x)\right].$$

In terms of our stated criteria, ϕ^2 is clearly a scaling parameter that does not impact the choice of the next point. However, since the correlation function r is often unknown, some estimation must be used. The typical approach is to select a class of correlation functions that is endowed with parameters and then estimate those parameters based on the data. We refer simply to Santner et al. (2003) to cover the estimates of these correlation parameters, and to Erickson et al. (2018) for guidance on the use of Gaussian process software.

4.2 Predicting $g(x)^\top g(x)$

The gradient of the estimator $\hat{y}(x)$ can be calculated as

$$\hat{g}(x) = (\nabla \boldsymbol{r}(x))^\top \boldsymbol{R}^{-1} \boldsymbol{y},$$

where we are leveraging vector calculus here to compact notation such that

$$\nabla \boldsymbol{r}(x) = \begin{pmatrix} (\nabla r(x,x_1))^\mathsf{T} \\ \vdots \\ (\nabla r(x,x_N))^\mathsf{T} \end{pmatrix}.$$

This estimate is unbiased under our model selection. We can go a step further, as Theorem 1 indicates.

Theorem 1 If $y(\cdot)$ follows a Gaussian process with correlation parameters θ and variance ϕ^2, an unbiased estimate of the expected value of $\|g(x)\|^2$ given the data is

$$\hat{g}(x)^\mathsf{T}\hat{g}(x) + \phi^2 \mathrm{tr}\left(\nabla^2 r(x,x) - (\nabla \boldsymbol{r}(x))^\mathsf{T} R^{-1} (\nabla \boldsymbol{r}(x))\right).$$

Proof. From Morris et al. (1993), the gradient of the Gaussian process given that the responses for design matrix X are y has a normal distribution with mean

$$E[g(x)] = \nabla \hat{y}(x) = \hat{g}(x) = (\nabla \boldsymbol{r}(x))^\mathsf{T} \boldsymbol{R}^{-1} \boldsymbol{y}$$

and covariance matrix

$$\mathrm{Cov}(g(x)) = \phi^2 \left(\nabla^2 r(x,x) - (\nabla \boldsymbol{r}(x))^\mathsf{T} R^{-1} (\nabla \boldsymbol{r}(x))\right).$$

The result follows from decomposing the expectation into a mean and variance term, both of which are known.

$$\begin{aligned} &E[g(x)^T g(x)] \\ &\quad = E\left[(g(x) - \hat{g}(x))^T (g(x) - \hat{g}(x)) + 2\hat{g}(x) g(x) - \hat{g}(x)\hat{g}(x)\right] \\ &\quad = \hat{g}(x)^T \hat{g}(x) + E\left[(g(x) - \hat{g}(x))^T (g(x) - \hat{g}(x))\right] \\ &\quad = \hat{g}(x)^T \hat{g}(x) + \sum_{i=1}^{d} \mathrm{Var}\left[g_i(x)\right] \\ &\quad = \hat{g}(x)^T \hat{g}(x) + \phi^2 \mathrm{tr}\left(\nabla^2 r(x,x) - (\nabla \boldsymbol{r}(x))^\mathsf{T} R^{-1} (\nabla \boldsymbol{r}(x))\right). \end{aligned}$$

□

In practice, since the parameters must be estimated and the function we are estimating is never an actual Gaussian process, unbiasedness of the estimator is no longer guaranteed. However, the estimator should be useful given the limited information provided the model is adequate. Thus the estimator of $\omega(x)$ given the data up through batch b is

$$\hat{\omega}_b(x) = \hat{g}(x)^T \hat{g}(x) + \hat{\phi}^2 \mathrm{tr}\left(\nabla^2 r(x,x) - (\nabla \boldsymbol{r}(x))^\mathsf{T} R^{-1} (\nabla \boldsymbol{r}(x))\right). \tag{6}$$

As shown in the proof of Theorem 1, (6) represents an expectation-variance decomposition of the gradient estimate. The first term on the right hand side of (6) is based solely on the mean of the gradient, while the second term relates to the variance of the gradient. Thus, the expected value of the squared gradient norm will be large where the gradient is expected to be large, or there is large estimation error in what its value is. When using this as the value function, it will lead to points being placed in areas where the gradient is expected to be large or is very uncertain. As more points are evaluated the variance (second term) will decrease, while the first term will approach the true value. This will lead to points being placed proportionally to the true squared gradient norm. In the long run, then, the points will be distributed as if the weight function were the true weight function that we can only approximate.

5 ALGORITHM

Algorithm 1 gives the general outline of the adaptive sampling algorithm, which selects a batch of points to be evaluated in each iteration. The experiment begins by evaluating a space filling initial batch of n points. A Gaussian process model is fit to this data. Each iteration of the adaptive loop selects points to evaluate, evaluates the points, and updates the model by adding the new data. While still in Stage 1, space filling points are selected for evaluation. In this paper we use an sFFLHD (Duan et al. 2017) as the sequential design to generate candidate points since it has good projective and space-filling properties. Once the initial space filling design is completed, the experiment is in Stage 2 and Algorithm 2 is used to select a batch of n points. Once these design points are evaluated, the model is updated with the new data, and the loop is restarted until ended by the experimenter.

Algorithm 1: Adaptive algorithm

```
Data: Batch size
Result: Completed simulation runs
Initialization;
Select n initial space-filling points and evaluate them;
Create a GP model fit to the data;
while Stopping criterion not met do
    if in Stage 1 then
        Select next n points from sFFLHD
    end
    if in Stage 2 then
        add 5n points from sFFLHD as candidates;
        use Algorithm 2 to select the next points to run;
    end
    Evaluate the selected points;
    Update model;
end
```

We break the simulation process into two stages, similar to Forrester et al. (2008), since our criterion cannot be used until the model built from data is reasonably accurate. The first stage selects points that are space-filling until the model is sufficiently accurate to guide point selection. In the second stage the model is used to select points, sometimes called the infill sample, according to our proposed criterion.

In the first stage, a space-filling design must be chosen to build a metamodel that can be used to guide adaptive point selection in the second stage. The number of points needed to achieve a useful metamodel depends on the input density and the roughness of the true function. Forrester et al. (2008) recommends using 35% of the available computation budget. In practice it is best to monitor the model and diagnostics as data comes in, and manually decide when the model is good enough. Automatic diagnostics can be used, but are not discussed here.

Once the model is considered to be reliable, we can use the value function to guide point selection, called Stage 2. Ideally, the batch of points selected will minimize the proposed criterion, shown in (5). Although points can be selected from anywhere in the design region, we assume that we have a pool of candidate points to choose from so that there is no need for continuous optimization. The candidate points are generated using a sequential design that can add points indefinitely to augment a design. Since each iteration selects n points, at least that many points must be added to the candidate set each iteration. We suggest adding roughly $5n$ points to the candidate set each iteration since it balances having enough points to focus on certain areas yet remaining small enough to be computationally feasible. Choosing the n points to minimize the proposed criterion out of a candidate set of p points is a combinatorially hard problem,

necessitating approximations to speed up the process. We use a variation on an exchange algorithm that starts with a batch of points then iterates to find if any other points would be a better replacement, similar to Kim et al. (2017). See Algorithm 2 for specifics.

Algorithm 2: Batch point selection

Input: The data collected up through iteration b; $U = \{u_1, \ldots, u_p\}$, set of p candidate points
Output: $S = \{s_1, \ldots, s_n\}$, the next n candidate points to evaluate

Initialize $S = \{s_1, \ldots, s_n\}$, the n points of U with the highest predicted standard error in increasing order
for j *in* $\{1, \ldots, n\}$ **do**
  $s^* = s_j$, keep track of point with the lowest Ψ^* thus far
  $\Psi^* = \Psi_b(S)$
  for u_i *in* $\{u_1, \ldots, u_m\} \setminus S$ **do**
    $S^i = (S \setminus s_j) \cup u_i$
    $\Psi_i = \Psi_b(S^i)$
    if $\Psi_i < \Psi^*$ **then**
      $s^* = u_i$
      $\Psi^* = \Psi_i$
    end
  end
  $s_j = s^*$, replace s_j with best alternate
end

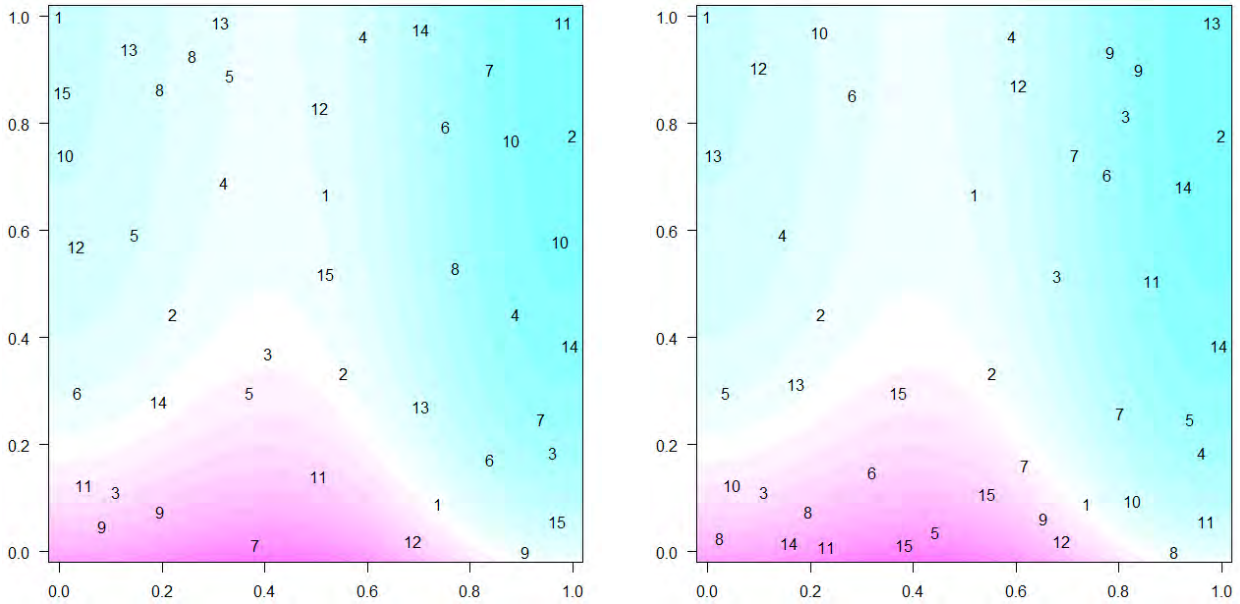
6 EMPIRICAL RESULTS

To demonstrate our algorithm, we plot the points our algorithm chooses when used for a single experiment. Instead of running a simulation to evaluate each input point, we use the two-dimensional nonpolynomial Lim function (Lim et al. 2002) to mimic the output of a simulation. We assume points can be evaluated in batches of $n = 3$, which may be the case if we have a computer with three cores. The first two batches, or six points, will be selected using an sFFLHD. The same sFFLHD will be used to generate candidates points that augment the design. In each iteration, $5n = 15$ candidates points are added to the candidate set. We continue adding points until we reach 45 points. Figure 2 shows the results of a single run of this experiment using the IMSE criterion (2a) and our proposed criterion (2b). In each figure, the contour plot shows the true function. The points are shown as numbers indicating in which batch they were selected. We would expect the IMSE criterion to place points fairly evenly throughout. We would expect the proposed criterion to place points, especially those in the later batches, in higher concentrations near the steep regions in the bottom and right of the plot. The plots closely match our expectations.

To further demonstrate that our algorithm is effective, we conducted repeated numerical experiments comparing it to other algorithms. As before, we assess the algorithms' performance using standard test functions in place of simulations. The statistic used to measure the quality of the algorithm is the integrated mean value-weighted squared error,

$$\Phi = \int_{\mathscr{X}} \|g(x)\|^2 \left(\hat{y}(x) - y(x)\right)^2 \mathrm{d}x. \tag{7}$$

This is what our proposed criterion attempts to minimize, except it has to estimate the gradient and squared error. Φ places higher value on having a smaller prediction error where the gradient is large. We expect this statistic to decrease each iteration as more data is added to the model, especially when the points are placed in areas where the gradient and squared error are large.



(a) Points selected using the IMSE criterion.

(b) Points selected using the proposed criterion.

Figure 2: Contour plots of the Lim function with the points selected indicated by the batch in which they were selected.

We compare our algorithm to four competitors. The first is a nonadaptive method that simply uses points from an sFFLHD in the order they are generated. The second is a nonadaptive method using points from a Sobol sequence (Sobol 1967). The third is the same as our algorithm except the criterion is unweighted, i.e., the IMSE criterion. The fourth is the same as our algorithm except it uses the plug-in criterion. All of the adaptive algorithms use points from an sFFLHD. In Stage 1, the points are selected in the order they are generated from the sFFLHD. In Stage 2, $5n$ candidate points are added from the same sFFLHD at the start of each iteration. We expect our algorithm to perform best since its selection criterion is most closely aligned with Φ.

We use the same setting as before: three points can be evaluated simultaneously, $n = 3$. We ran experiments on 3 different problems to compare our method to others. In addition to the Lim function, we also use the Branin (Dixon 1978) and Franke (Franke 1979) functions, all of which are two-dimensional functions. For each of the experiments, the initial six points are the first six points generated by the sFFLHD, then points are added in batches of size three until a total of 30 points have been evaluated. For each of the experiments, 10 batches, or 30 points, are evaluated.

Table 1 shows sample means and standard errors from the 100 replicates of Φ, where Φ is calculated from the Gaussian process model after all 30 points have been evaluated. The minimum mean in each column is bold, along with any that are not different from the minimum at an $\alpha = 0.05$ significance level. Selecting points using the plug-in criterion gives the lowest mean Φ for the Franke function by a wide margin. Our proposed algorithm does the best on the other two functions, with plug-in close behind. The plug-in criterion probably does well since the gradients of these functions can be estimated well with few points. The results may be different in higher dimensions or with more complex functions.

7 DISCUSSION

In this paper we have presented a sampling scheme for adaptive computer experiments that places points in order to focus on areas with large absolute gradients while still covering the whole space. The goal of the algorithm is to minimize the integrated mean value-weighted squared error over the entire surface,

Table 1: Comparison of methods on three functions.

	Mean Φ (std. error Φ) from 100 replicates		
Method	Branin ($\times 10^{-6}$)	Franke ($\times 10^{3}$)	Lim ($\times 10^{2}$)
IMSE	6.53 (0.85)	5.91 (0.91)	15.02 (1.07)
Proposed	**3.75 (0.50)**	2.37 (0.45)	**8.08 (0.68)**
Plug-in	**4.06 (0.59)**	**1.43 (0.06)**	**9.00 (0.84)**
sFFLHD	21.97 (4.87)	4.03 (0.59)	52.18 (4.44)
Sobol	13.10 (2.51)	3.30 (0.37)	53.27 (5.36)

from which the rest of the algorithm follows logically. A Gaussian process model is used to fit and update a metamodel to the data from completed simulations, although it can be replaced by any metamodel that gives predictions with standard errors. Candidate points are generated from an sFFLHD to ensure that well-spaced points are available. The metamodel is used to select which points to evaluate next, trying to choose points that will provide the best reduction in integrated mean value-weighted squared error over the entire surface. In numerical tests we found that using the simpler plug-in criterion did about as well as our proposed method, but both did significantly better than using the IMSE criterion or using points nonadaptively from a sFFLHD or Sobol sequence.

In further work we will conduct more rigorous numerical experiments as well as demonstrate the method on a real-world example based on our work at the Naval Postgraduate School. Each part of this algorithm is exchangeable, such as using a different design for candidate points, a different model type, or a different selection method. For future work we will investigate the use of different value functions to focus on different types of features, such as plateaus or regions where the second order derivatives are large. Furthermore, we will also improve the method so that it works well with higher dimensional input and larger sample sizes to enable it to be applied to a wider variety of real-world experiments.

ACKNOWLEDGMENTS

This work was supported in part by the Office of the Secretary of Defense, Director of Operational Test and Evaluation (OSD DOT&E) and the Test Resource Management Center (TRMC) within the Science of Test research consortium; the NPS Consortium for Robotics and Unmanned Systems Education and Research (CRUSER), and the NPS Naval Research Program (NRP). Disclaimer: The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Navy, Department of Defense, or the U.S. Government.

REFERENCES

Dixon, L. 1978. "The Global Optimization Problem. An Introduction". *Toward global optimization* 2:1–15.

Duan, W., B. E. Ankenman, S. M. Sanchez, and P. J. Sanchez. 2017. "Sliced Full Factorial-Based Latin Hypercube Designs as a Framework for a Batch Sequential Design Algorithm". *Technometrics* 59(1):11–22.

Erickson, C. B., B. E. Ankenman, and S. M. Sanchez. 2018. "Comparison of Gaussian Process Modeling Software". *European Journal of Operational Research* 266(1):179–192.

Forrester, A., A. Sobester, and A. Keane. 2008. *Engineering Design via Surrogate Modelling: A Practical Guide*. John Wiley & Sons.

Franke, R. 1979. "A Critical Comparison of Some Methods for Interpolation of Scattered Data". Technical report, Monterey, California: Naval Postgraduate School.

Gramacy, R. B., and H. K. Lee. 2009. "Adaptive Design and Analysis of Supercomputer Experiments". *Technometrics* 51(2):130–145.

Jones, D. R., M. Schonlau, and W. J. Welch. 1998. "Efficient Global Optimization of Expensive Black-Box Functions". *Journal of Global optimization* 13(4):455–492.

Joseph, V. R., T. Dasgupta, R. Tuo, and C. J. Wu. 2015. "Sequential Exploration of Complex Surfaces Using Minimum Energy Designs". *Technometrics* 57(1):64–74.

Kim, H., J. T. Vastola, S. Kim, J.-C. Lu, and M. A. Grover. 2017. "Batch Sequential Minimum Energy Design with Design-Region Adaptation". *Journal of Quality Technology* 49(1):11.

Lim, Y. B., J. Sacks, W. Studden, and W. J. Welch. 2002. "Design and Analysis of Computer Experiments When the Output is Highly Correlated Over the Input Space". *Canadian Journal of Statistics* 30(1):109–126.

Loeppky, J. L., L. M. Moore, and B. J. Williams. 2010. "Batch Sequential Designs for Computer Experiments". *Journal of Statistical Planning and Inference* 140(6):1452–1464.

Morris, M. D., T. J. Mitchell, and D. Ylvisaker. 1993. "Bayesian Design and Analysis of Computer Experiments: Use of Derivatives in Surface Prediction". *Technometrics* 35(3):243–255.

Sacks, J., W. J. Welch, T. J. Mitchell, and H. P. Wynn. 1989. "Design and Analysis of Computer Experiments". *Statistical Science* 4(4):409–423.

Santner, T. J., B. J. Williams, and W. Notz. 2003. *The Design and Analysis of Computer Experiments*. Springer Science & Business Media.

Sobol, I. M. 1967. "On the Distribution of Points in a Cube and the Approximate Evaluation of Integrals". *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 7(4):784–802.

Vazquez, E., and J. Bect. 2010. "Convergence Properties of the Expected Improvement Algorithm with Fixed Mean and Covariance Functions". *Journal of Statistical Planning and Inference* 140(11):3088 – 3095.

AUTHOR BIOGRAPHIES

COLLIN B. ERICKSON is PhD candidate in Industrial Engineering and Management Sciences at Northwestern University. His research interests include sequential design of experiments and Gaussian process models. His e-mail address is collinerickson@u.northwestern.edu.

BRUCE E. ANKENMAN is a Professor of Industrial Engineering and Management Sciences at Northwestern University. He holds a PhD in Industrial Engineering from the University of Wisconsin-Madison. His research interests include developing tools for the design and analysis of both physical and simulation-based experiments. As the Co-Director of the Segal Design Institute at Northwestern, he directs educational programs in human-centered design. His email address is ankenman@northwestern.edu.

MATTHEW PLUMLEE is an Assistant Professor of Industrial Engineering and Management Sciences at Northwestern University. His interests are in the interface of data and modeling; specifically in methods for experimentation and uncertainty quantification for complex systems. He holds a BA in Mechanical Engineering from Purdue University, and an MS and a PhD from the Georgia Institute of Technology in Statistics and Industrial Engineering, respectively. His email address is mplumlee@northwestern.edu.

SUSAN M. SANCHEZ is a Professor of Operations Research at the Naval Postgraduate School with a joint appointmen in the Graduate School of Business & Public Policy. She holds a PhD in Operations Research from Cornell University. Her interests include the design and analysis of large-scale simulation experiments, robust design, and applied statistics, with application to military operations, manufacturing, and health care. She co-founded and serves as Co-director of NPS's Simulation Experiments & Efficient Designs (SEED) Center for Data Farming. She has served as an officer in her INFORMS Simulation Society, on the Board of Directors for the Winter Simulation Conference, and on the editorial boards of several flagship journals. She is a Titan of Simulation and an INFORMS Fellow. Her email address is ssanchez@nps.edu.