Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

CALIBRATING SIMULATION MODELS USING THE KNOWLEDGE GRADIENT WITH CONTINUOUS PARAMETERS

Warren R. Scott

Operations Research and Financial Engineering
Princeton University
Princeton, NJ 08544, USA

Warren B. Powell

Operations Research and Financial Engineering
Princeton University
Princeton, NJ 08544, USA

Hugo P. Simão

Operations Research and Financial Engineering
Princeton University
Princeton, NJ 08544, USA

ABSTRACT

We describe an adaptation of the knowledge gradient, originally developed for discrete ranking and selection problems, to the problem of calibrating continuous parameters for the purpose of tuning a simulator. The knowledge gradient for continuous parameters uses a continuous approximation of the expected value of a single measurement to guide the choice of where to collect information next. We show how to find the parameter setting that maximizes the expected value of a measurement by optimizing a continuous but nonconcave surface. We compare the method to sequential kriging for a series of test surfaces, and then demonstrate its performance in the calibration of an expensive industrial simulator.

1 INTRODUCTION

We consider the problem of tuning the parameters of an expensive simulator to achieve specific performance metrics. This problem requires searching over a continuous, multidimensional parameter space to find the settings which produce the best results. Each measurement is time consuming and yet produces noisy measurements. In this paper, we propose an algorithm that is asymptotically optimal but which promises fast convergence.

We use as our algorithmic framework the correlated knowledge gradient algorithm presented in (Frazier, Powell, and Dayanik 2009) which combines a model for the function being maximized with a criterion for choosing the sampling decision based on the value of an observation. The knowledge gradient policy is a widely applicable policy that has had promising results maximizing a variety of standard test functions. The strength of the policy is based on the policy's implicit handling of exploration versus exploitation when choosing the sampling decision which leads to convergence properties.

A popular approach for optimizing expensive functions is Bayesian global optimization which combines a model and a sampling criterion to sequentially choose points to sample in order to maximize a function. Common sampling criteria include the probability of improvement presented in (Kushner 1964) or expected improvement criteria (see (Zhilinskas 1975); (Mockus 1993); (Locatelli 1997); (Jones, Schonlau, and Welch 1998); (Huang, Allen, Notz, and Zeng 2006)). These sampling criteria can easily be exactly calculated with a kriging or Gaussian process regression model (see (Matheron 1963); (Sacks, Welch, Mitchell, and Wynn 1989); (Cressie 1990); (Kleijnen 2009); (Rasmussen and Williams 2006)). Gaussian process regression treats the truth as a realization of a Gaussian process and is convenient for interpretation purposes because it combines a regression function with the distribution of uncertainty about the regression function. The correlated knowledge gradient presented in (Frazier, Powell, and Dayanik 2009) computes the expected improvement in the performance of a design as a result of a single measurement, for problems with a finite (and not too large) number of potential measurements. In this paper, we present an adaptation of the knowledge gradient for problems with multidimensional, continuous parameters. Our presentation is based on the work in (Scott, Frazier, and Powell 2010).

We begin by writing our model calibration problem as an optimization model. Next we describe the knowledge gradient framework for sequentially searching for the best set of parameters, where the goal is fast convergence in the face of noisy measurements. We sketch the logic used to handle continuous parameters by using an approximation of the knowledge gradient, which allows us to find the best measurement by solving a nonlinear programming problem over a nonconcave surface. Finally, we compare its performance against sequential kriging on structured test problems as well as an industrial simulator.

1.1 CALIBRATION PROBLEM

We want to calibrate our simulator over p continuous parameters, $x = [x_1, ..., x_p]^T$. After the simulator runs with a fixed x, it outputs q statistics of the company's simulated operations, $G_1(x), ..., G_q(x)$; our observations of these statistics contain noise because there is randomness in the simulator. We want to calibrate the simulator against the company's actual operations so we use the company's true statistics which are $g_1, ..., g_q$. We want to find parameters x such that $G_i(x)$ is close to g_i for $i = 1, ..., q$. Using a quadratic loss function, we can write our objective as maximizing the function $\mu(x)$ given by

$$\mu(x) = -\sum_{i=1}^{q} (G_i(x) - g_i)^2 . \tag{1}$$

Our optimization problem can now be written

$$\max_{x \in \mathscr{X}} \mu(x), \tag{2}$$

where $x \in \mathbb{R}^p$ and $\mathscr{X}$ is a compact set. Our observations of $\mu(x)$ contain noise since there is randomness in the simulator. We will sequentially choose parameter settings x^n for $n = 0, ..., N-1$ where N is the number of times we are allowed to run the simulator due to time constraints. For $n = 0, ..., N-1$, after deciding to try parameter setting x^n, we run the simulator which gives us a noisy observation of $\mu(x^n)$ which we call $\hat{y}^{n+1}$. We next describe a model for $\mu(x)$.

2 KNOWLEDGE GRADIENT FRAMEWORK

We summarize the correlated knowledge gradient framework as presented in (Frazier, Powell, and Dayanik 2009) and extend it to the case of continuous decision variables. The framework combines a model for the function being maximized as well as a method for choosing the sampling decision based on the value of an observation.

2.1 GAUSSIAN PROCESS REGRESSION

Gaussian process regression (see (Rasmussen and Williams 2006)) or equivalently kriging in geosciences (see (Matheron 1963)) uses a Bayesian approach to model the function $\mu(x)$. We treat $\mu(x)$ as a realization of a Gaussian process; for any set of points $[x^0, ..., x^n]$, $[\mu(x^0), ..., \mu(x^n)]$ is a realization of a multivariate normal distribution. That is, $[\mu(x^0), ..., \mu(x^n)]^T \sim \mathscr{N}(\mu^0([x^0, ..., x^n]), \Sigma^0([x^0, ..., x^n]))$, where $\mu^0([x^0, ..., x^n])$ is $\mathbb{E}([\mu(x^0), ..., \mu(x^n)]^T)$ and $\Sigma^0([x^0, ..., x^n])$ is $\text{Cov}([\mu(x^0), ..., \mu(x^n)]^T)$. The mean of the prior, $\mathbb{E}(\mu(x))$, is often set to a constant, and the initial covariance matrix can be fully specified by using a covariance function. For arbitrary points x^0 and x^1, define $\Sigma^0(x^0, x^1) = \text{Cov}\left(\mu(x^0), \mu(x^1)\right)$. We use the power exponential covariance function as in (Jones, Schonlau, and Welch 1998) to assign the initial covariance,

$$\Sigma^0(x^0, x^1) = \text{Cov}\left(\mu(x^0), \mu(x^1)\right) = \beta e^{-\sum_{i=1}^{p} \alpha_i (x_i^0 - x_i^1)^2}, \tag{3}$$

where $\beta > 0$ is a scalar which specifies the uncertainty of μ (larger variance is larger uncertainty), and $\alpha_i > 0$ specifies the smoothness in dimension i (larger correlations create a smoother regression function). Intuitively, if x^0 and x^1 are close, $\mu(x^0)$ and $\mu(x^1)$ will be highly correlated, roughly meaning $\mu(x^0)$ and $\mu(x^1)$ are expected to be similar.

To formally denote the information available at iteration n, we define a filtration $\mathscr{F}^n$ as the sigma-algebra generated by $x^0, \hat{y}^1, ..., x^{n-1}, \hat{y}^n$. We denote our posterior distribution as $\mu^n([x^0, ..., x^n]) = \mathbb{E}([\mu(x^0), ..., \mu(x^n)]^T | \mathscr{F}^n)$, $\Sigma^n([x^0, ..., x^n]) = \text{Cov}([\mu(x^0), ..., \mu(x^n)]^T | \mathscr{F}^n)$, and $\Sigma^n(x^0, x^1) = \text{Cov}(\mu(x^0), \mu(x^1) | \mathscr{F}^n)$. We assume our observations are normally distributed around the truth with variance $\lambda(x)$, that is $\hat{y}^{n+1} | \mu, x^n \sim \mathscr{N}(\mu(x^n), \lambda(x^n))$. The multivariate normal distribution is a conjugate family when the observations come from a normal distribution with known variance, and hence $[\mu(x^0), ..., \mu(x^n)]$ conditioned on $\mathscr{F}^n$ also has a multivariate normal distribution. Let I_n be an $n \times n$ identity matrix and suppose we are at iteration n, meaning we have observed $\hat{y}^1, ..., \hat{y}^n$ at the decisions $x^0, ..., x^{n-1}$. We now give the equations for the regression (see (Meinhold and Singpurwalla 1983);

(Rasmussen and Williams 2006)). The residuals are given by

$$\tilde{y}^n = \begin{bmatrix} \hat{y}^1 \\ \vdots \\ \hat{y}^n \end{bmatrix} - \begin{bmatrix} \mu^0(x^0) \\ \vdots \\ \mu^0(x^{n-1}) \end{bmatrix}, \tag{4}$$

and the covariance of the residuals are given by,

$$S^n = \Sigma^0 + \begin{bmatrix} \lambda(x^0) & & \\ & \ddots & \\ & & \lambda(x^{n-1}) \end{bmatrix}. \tag{5}$$

The optimal Kalman gain can be calculated as,

$$K^n = \Sigma^0 [S^n]^{-1}, \tag{6}$$

and the updated means and covariance matrix of the first n decisions can be written in matrix notation as,

$$\begin{bmatrix} \mu^n(x^0) \\ \vdots \\ \mu^n(x^{n-1}) \end{bmatrix} = \begin{bmatrix} \mu^0(x^0) \\ \vdots \\ \mu^0(x^{n-1}) \end{bmatrix} + K^n \tilde{y}^n, \tag{7}$$

$$\Sigma^n = (I_n - K^n)\Sigma^0. \tag{8}$$

The updated mean and variance of an arbitrary deicision, x, is called Gaussian process regression and is given by

$$\mu^n(x) = \mu^0(x) + \left[\Sigma^0(x^0,x) \quad ,\cdots, \quad \Sigma^0(x^{n-1},x)\right] [S^n]^{-1} \tilde{y}^n, \tag{9}$$

$$\Sigma^n(x,x) = \Sigma^0(x,x) - \left[\Sigma^0(x^0,x) \quad ,\cdots, \quad \Sigma^0(x^{n-1},x)\right] [S^n]^{-1} \begin{bmatrix} \Sigma^0(x^0,x) \\ \vdots \\ \Sigma^0(x^{n-1},x) \end{bmatrix}. \tag{10}$$

For a complete derivation of recursive updating equations and their relation to a standard normal random variable see (Frazier, Powell, and Dayanik 2009). The regression can be updated recursively with the equation

$$\begin{bmatrix} \mu^{n+1}(x^0) \\ \vdots \\ \mu^{n+1}(x^n) \end{bmatrix} = \begin{bmatrix} \mu^n(x^0) \\ \vdots \\ \mu^n(x^n) \end{bmatrix} + \frac{y^{n+1} - \mu^n(x^n)}{\lambda(x^n) + \Sigma^n(x^n,x^n)} \begin{bmatrix} \Sigma^n(x^0,x^n) \\ \vdots \\ \Sigma^n(x^n,x^n) \end{bmatrix}. \tag{11}$$

Noting that $\mathrm{Var}(y^{n+1} - \mu^n(x^n)|\mathscr{F}^n) = \lambda(x^n) + \Sigma^n(x^n,x^n)$, equation (11) can be rewritten

$$\begin{bmatrix} \mu^{n+1}(x^0) \\ \vdots \\ \mu^{n+1}(x^n) \end{bmatrix} = \begin{bmatrix} \mu^n(x^0) \\ \vdots \\ \mu^n(x^n) \end{bmatrix} + \frac{Z^{n+1}}{\sqrt{\lambda(x^n) + \Sigma^n(x^n,x^n)}} \begin{bmatrix} \Sigma^n(x^0,x^n) \\ \vdots \\ \Sigma^n(x^n,x^n) \end{bmatrix}, \tag{12}$$

where Z^{n+1} is a standard normal random variable which equals $\left(y^{n+1} - \mu^n(x^n)\right) / \sqrt{\lambda(x^n) + \Sigma^n(x^n,x^n)}$; all the other terms in equation (12) are $\mathscr{F}^n$ measurable and hence constants.

2.2 KNOWLEDGE GRADIENT

The knowledge gradient is a nonnegative number which gives the value of a measurement defined to be the expected amount the maximum of the regression will increase due to one additional observation at a decision

x. Mathematically this is written in (Frazier, Powell, and Dayanik 2009) as

$$\nu^{KG,n}(x) = \mathbb{E}\left[\max_{u\in\mathscr{X}} \mu^{n+1}(u) \middle| \mathscr{F}^n, x^n = x\right] - \max_{u\in\mathscr{X}} \mu^n(u), \tag{13}$$

where the first term is the expected maximum of the regression after one additional observation at decision x and the second term is the current maximum of the regression function. In equation (13), $\mu^{n+1}(u)$ is a random variable because it depends on the observation y^{n+1}. The knowledge gradient is larger at decisions where the regression function is higher, but it is also larger at points where there is more uncertainty. The knowledge-gradient policy chooses the sampling decision at time n by maximizing the knowledge gradient,

$$x^n \in \operatorname{argmax}_{x\in\mathscr{X}} \nu^{KG,n}(x). \tag{14}$$

When the decision space is discrete and finite, the knowledge gradient can be exactly calculated using the algorithms described in (Frazier, Powell, and Dayanik 2009). To search over a continuous parameter space, approximations are necessary to calculate the knowledge gradient. Below we propose an adaptation of the knowledge gradient where we approximate (13) in a way that makes it possible to search over $\nu^{KG}(x)$ as a continuous surface.

2.3 APPROXIMATE KNOWLEDGE GRADIENT

To handle problems where the measurement x is multidimensional and continuous, we need to find an analytical approximation which can be optimized using classical search techniques. We do this by replacing the maximum over $\mathscr{X}$ with a maximum over previously sampled points, which gives us the approximate knowledge gradient,

$$\bar{\nu}^{KG,n}(x) = \mathbb{E}\left[\max_{i=0,..,n} \mu^{n+1}(x^i) \middle| \mathscr{F}^n, x^n = x\right] - \max_{i=0,..,n} \mu^n(x^i)|_{x^n=x}. \tag{15}$$

Analogous to equation (14), the sampling decision is the decision which has the largest value,

$$x^n \in \operatorname{argmax}_{x\in\mathscr{X}} \bar{\nu}^{KG,n}(x). \tag{16}$$

The approximate knowledge gradient is now taken over a discrete set of points and hence can be calculated by the algorithm in (Frazier, Powell, and Dayanik 2009). We define $\tilde{\sigma}(\Sigma,x) = (\Sigma e_x)/(\sqrt{\lambda(x) + e_x^T \Sigma e_x})$, where e_x is a column vector of zeros with a 1 corresponding to decision x. Next, we define the pairs (a_i,b_i) for $i = 0,\dots,n$ as the sorted pairs $(\mu^n(x^i), \tilde{\sigma}_i(\bar{\Sigma}^n, x^n))$ conditioned on $\mathscr{F}^n$ and $x^n = x$ for $i = 0,\dots,n$. The pairs (a_i,b_i) are sorted such that $b_i \le b_{i+1}$ for $i = 0,\dots,n-1$. If there exists some $i \ne j$ such that $b_i = b_j$ and $a_i \le a_j$, then the pair (a_j,b_j) dominates (a_i,b_i) and you add (a_i,b_i) to a list of *initially* dominated lines. The a_i's can be viewed as the intercepts and the b_i's can be viewed as the slopes of the lines. We define A^0 as the index map such that $(a_i,b_i) = (\mu^n(x^{A_i^0}), \tilde{\sigma}_{A_i^0}(\bar{\Sigma}^n, x^n))$. For a fixed $x^n = x$, a_i and b_i are constants. Now, the first term in the approximate knowledge gradient can be calculated as

$$\begin{aligned}
\mathbb{E}\left[\max_{i=0,\dots,n} \mu^{n+1}(x^i) \middle| \mathscr{F}^n, x^n = x\right] &= \mathbb{E}\left[\max_{i=0,\dots,n} \mu^n(x^i) + \tilde{\sigma}_i(\bar{\Sigma}^n, x^n) Z^{n+1} \middle| \mathscr{F}^n, x^n = x\right] && (17)\\
&= \mathbb{E}\left[\max_{i=0,\dots,n} a_i + b_i Z\right]. && (18)\\
&= \mathbb{E}\left[\sum_{i=0}^{\tilde{n}} \left(a_{A_i^1} + b_{A_i^1} Z\right) \mathbf{1}_{[\tilde{c}_i, \tilde{c}_{i+1})}(Z)\right] && (19)\\
&= \sum_{i=0}^{\tilde{n}} \left[a_{A_i^1} \left(\Phi(\tilde{c}_{i+1}) - \Phi(\tilde{c}_i)\right) + b_{A_i^1} \left(\phi(\tilde{c}_i) - \phi(\tilde{c}_{i+1})\right)\right]. && (20)
\end{aligned}$$

In equation (17) we use the recursive update equation for $\mu^{n+1}(x^i)$ given by equation (12). Algorithm 1 is a scan-line algorithm that replaces the maximization in equation (18) with a piecewise linear function using indicator functions. In Algorithm 1, A^1 is called the accept set and is a vector of indices which keeps track of all the i's such that line $a_i + b_i z$ is part of the maximum in equation (18). We keep track of the values of z where

Table 1: Summary of Algorithm 1 which expresses $f(z) = \max_{i=0,\dots,n} a_i + b_i z$ as $f(z) = \sum_{i=0}^{\tilde{n}} (\tilde{a}_i + \tilde{b}_i z)\mathbf{1}_{[\tilde{c}_i, \tilde{c}_{i+1})}(z)$. .

```
$c_0 = -\infty$, $c_{n+1} = +\infty$, $A^1 = [0]$
for $i = 1 : n$
    if $(a_i, b_i)$ not initially dominated
        loopdone = false
        while loopdone == false
            $j = A^1(end)$
            $c_{j+1} = (a_j - a_i)/(b_i - b_j)$
            if $length(A^1) \neq 1$ & $c_{j+1} \leq c_{k+1}$ where $k = A^1(end-1)$
                Delete last element in $A^1$.
            else add $i$ to the end of $A^1$.
                loopdone = true
            end
        end
    end
end
```

the lines intersect in a vector c. c_{i+1} is the largest value of z such that line $a_i + b_i z$ is the maximum in equation (18). In terms of the lines in the accept set A^1, $c_{1+A^1_i}$ is the intersection of $a_{A^1_i} + b_{A^1_i} z$ and $a_{A^1_{i+1}} + b_{A^1_{i+1}} z$. Solving for the z such that these lines intersect we get $c_{1+A^1_i} = (a_{A^1_i} - a_{A^1_{i+1}})/(b_{A^1_{i+1}} - b_{A^1_i})$ for $i = 1, \dots, \tilde{n}$, where $\tilde{n}$ is the length of A^1 minus one. Also we set $c_0 = -\infty$ and $c_{n+1} = +\infty$. For convenience, we define $\tilde{a}_i = a_{(A^1_i)}$, $\tilde{b}_i = b_{(A^1_i)}$, $\tilde{c}_{i+1} = c_{(1+A^1_i)}$, and $\tilde{c}_0 = -\infty$ for $i = 0, \dots, \tilde{n}$. Algorithm 1, given in Table 1, efficiently calculates constants $\tilde{c}_0, \dots, \tilde{c}_{\tilde{n}+1}$ and the vector of indices, A^1, so that a function of the form $f(z) = \max_{i=0,\dots,n} a_i + b_i z$ can be rewritten as $f(z) = \sum_{i=0}^{\tilde{n}} (a_{A^1_i} + b_{A^1_i} z)\mathbf{1}_{[\tilde{c}_i, \tilde{c}_{i+1})}(z)$.

In addition, we show how to calculate the gradient of the approximate knowledge gradient,$\nabla_x \bar{v}^{KG,n}(x)$, at a fixed $x \in \mathscr{X}$. This will allow us to use gradient ascent to maximize the approximate knowledge gradient. Let $A = A^0[A^1]$; A contains the indices i such that $(\mu^n(x^{A^0_i}) + \tilde{\sigma}_{A^0_i}(\bar{\Sigma}^n, x^n))z$ is part of the maximum in equation (17) for some value of z. The gradient is given by using the chain rule because a, b, and c depend on x^n and by using the fact that $\frac{\partial}{\partial x}\Phi(f(x)) = \phi(f(x))\frac{\partial}{\partial x}f(x)$ and $\frac{\partial}{\partial x}\phi(f(x)) = -\phi(f(x))f(x)\frac{\partial}{\partial x}f(x)$,

$$\begin{aligned} & \nabla_x \mathbb{E}\left[\max_{i=0,..,n} \mu^{n+1}(x^i) \middle| \mathscr{F}^n, x^n = x\right] \\ = & \sum_{i=0}^{\tilde{n}} \left[\left(\nabla_{x^n}\mu^n(x^{A_i})\right)\left(\Phi(\tilde{c}_{i+1}) - \Phi(\tilde{c}_i)\right) + \left(\nabla_{x^n}\tilde{\sigma}_{A_i}(\bar{\Sigma}^n, x^n)\right)\left(\phi(\tilde{c}_i) - \phi(\tilde{c}_{i+1})\right)\right] \\ + & \sum_{i=0}^{\tilde{n}} \left[\left(\mu^n(x^{A_i}) + \tilde{\sigma}_{A_i}(\bar{\Sigma}^n, x^n)\tilde{c}_{i+1}\right)\phi(\tilde{c}_{i+1})\nabla_{x^n}\tilde{c}_{i+1} - \left(\mu^n(x^{A_i}) + \tilde{\sigma}_{A_i}(\bar{\Sigma}^n, x^n)\tilde{c}_i\right)\phi(\tilde{c}_i)\nabla_{x^n}\tilde{c}_i\right]. \end{aligned} \tag{21}$$

The calculation of $\nabla_{x^n}\tilde{c}_i$ for $i = 0, \dots, \tilde{n}+1$ is relatively straightforward. An equivalent equation for the $\tilde{c}_i$'s which are output from Algorithm 1 is $\tilde{c}_i = \frac{\tilde{a}_{i-1} - \tilde{a}_i}{\tilde{b}_i - \tilde{b}_{i-1}}$ for $i = 1, \dots, \tilde{n}$ with $\tilde{c}_0 = -\infty$ and $\tilde{c}_{\tilde{n}+1} = +\infty$. Then using the quotient rule we can calculate the following:

$$\nabla_{x^n}\tilde{c}_i = \begin{cases} \frac{(\tilde{b}_i - \tilde{b}_{i-1})(\nabla\tilde{a}_{i-1} - \nabla\tilde{a}_i) - (\tilde{a}_{i-1} - \tilde{a}_i)(\nabla\tilde{b}_i - \nabla\tilde{b}_{i-1})}{(\tilde{b}_i - \tilde{b}_{i-1})^2}, & \text{for } i = 1, \dots, \tilde{n} \\ \vec{0}, & \text{for } i = 0, \tilde{n}+1. \end{cases} \tag{22}$$

Equation (21) also requires calculating $\nabla_{x^n}\mu^n(x^i)$ and $\nabla_{x^n}\tilde{\sigma}_i(\bar{\Sigma}^n,x^n)$ for $i=0,...,n$. Let $\vec{0}$ be a column vector of zeros, let $B \triangleq \sqrt{\lambda(x^n)+e_{x^n}^T\bar{\Sigma}^n e_{x^n}}$, and let J^n be the following matrix of first-order partial derivatives,

$$\begin{aligned} J^n &= \left[\nabla_{x^n}\Sigma^0(x^0,x^n) \quad ,\cdots, \quad \nabla_{x^n}\Sigma^0(x^{n-1},x^n)\right] && (23)\\ &= 2\begin{bmatrix} \alpha_1(x_1^0-x_1^n)\Sigma^0(x^0,x^n) & \cdots & \alpha_1(x_1^{n-1}-x_1^n)\Sigma^0(x^{n-1},x^n)\\ \vdots & \ddots & \vdots \\ \alpha_p(x_p^0-x_p^n)\Sigma^0(x^0,x^n) & \cdots & \alpha_p(x_p^{n-1}-x_p^n)\Sigma^0(x^{n-1},x^n)\end{bmatrix}. && (24)\end{aligned}$$

Then we can write

$$\nabla_{x^n}\mu^n(x^i) = \begin{cases}\vec{0}, & \text{if } i<n\\ \nabla_{x^n}\mu^0(x^n)+J^n[S^n]^{-1}\tilde{y}^n, & \text{if } i=n.\end{cases}$$

and

$$\nabla_{x^n}\tilde{\sigma}_i(\bar{\Sigma}^n,x^n) = \frac{B\nabla_{x^n}e_{x^i}^T\bar{\Sigma}^n e_{x^n} - e_{x^i}^T\bar{\Sigma}^n e_{x^n}\nabla_{x^n}B}{B^2},$$

where

$$\nabla_{x^n}e_{x^i}^T\bar{\Sigma}^n e_{x^n} = \begin{cases} 2\mathrm{DIAG}(\alpha)(x^i-x^n)\Sigma^0(x^i,x^n)-J^n[S^n]^{-1}\Sigma^0 e_{x^i}, & \text{if } i<n\\ -2J^n[S^n]^{-1}\begin{bmatrix}\Sigma^0(x^0,x^n)\\ \vdots \\ \Sigma^0(x^{n-1},x^n)\end{bmatrix}, & \text{if } i=n\end{cases}$$

and

$$\nabla_{x^n}B = \frac{1}{2}(\lambda(x^n)+\Sigma^n(x^n,x^n))^{-\frac{1}{2}}\left(\nabla_{x^n}\lambda(x^n)-2J^n[S^n]^{-1}\begin{bmatrix}\Sigma^0(x^0,x^n)\\ \vdots \\ \Sigma^0(x^{n-1},x^n)\end{bmatrix}\right).$$

We have now shown how to calculate the approximate knowledge gradient in equation (20) and its gradient in equation (21). Detailed derivations for the procedures and equations in the section can be found in (Scott, Frazier, and Powell 2010).

2.4 MAXIMIZING THE APPROXIMATE KNOWLEDGE GRADIENT

Choosing the sampling decision in equation (16) requires care because the knowledge gradient is not a concave function. From observation, the knowledge gradient often has local minima near previously sampled points because the uncertainty near these points is very low. On the other hand, when two points have been sampled, often there is a local maximum in the middle of the points where the uncertainty is still large. Based on these observations and because equation (21) gives us the gradient of the approximate knowledge gradient, we propose a multistart gradient ascent algorithm started at the midpoints of every possible pair of previously sampled points. Each gradient ascent has a fixed stepsize chosen such that the magnitude of the initial step is one fourth the Euclidian distance between the pair of previously sampled points to ensure the gradient ascent algorithm does not immediately leave the region of interest. Each gradient ascent algorithm will converge to a local maximum. Afterwards, we take the maximum of all the local maxima to get an approximation of the maximum of the approximate knowledge gradient. Finding the exact maximum is not critical since this optimization problem is just to determine the next sampling decision.

2.5 PARAMETER ESTIMATION DETAILS

Our observations, $y^1,...,y^n$, come from a multivariate normal distribution and hence we can easily write the likelihood. Assuming that the variance of the observation noise is a constant, λ, and the initial mean, $\mu^0(x)$, is a constant, μ^0, the likelihood can easily be written as,

$$L_{\hat{y}}(\alpha,\beta,\lambda,\mu^0) = (2\pi)^{-n/2}|\Sigma^0+\lambda I|^{-1/2}\exp\left(-\frac{1}{2}(\hat{y}-\mu^0\mathbf{1})^T(\Sigma^0+\lambda I)^{-1}(\hat{y}-\mu^0\mathbf{1})\right),$$

Table 2: AKG Policy.

```
for $n = 0,...,N-1$
    Choose sampling decision: $x^n \in \arg\max_{x\in\mathscr{X}} \bar{\nu}^{\mathrm{KG},n}(x)$.
    Get noisy observation $\hat{y}^{n+1}$ of function at $x^n$.
    Update the regression, $\mu^{n+1}$ and $\Sigma^{n+1}$.
end
Implement $x^\star \in \arg\max_{x\in\mathscr{X}} \mu^N(x)$.
```

where $\mathbf{1}$ is a $n \times 1$ column vector of ones and $\hat{y} = \begin{bmatrix}\hat{y}^1 & \cdots & \hat{y}^n\end{bmatrix}^T$. For the numerical work in this paper we use maximum likelihood estimation, although maximum a posteriori or other methods for estimating the parameters of a multivariate normal distribution are appropriate.

2.6 OUTLINE OF APPROXIMATE KNOWLEDGE GRADIENT POLICY

As outlined in (Scott, Frazier, and Powell 2010), the approximate knowledge gradient policy samples the maximum of the approximate knowledge gradient after every iteration in order to sequentially choose our N sampling decisions. Table 2 shows the approximate knowledge gradient policy. If necessary, an initial Latin hypercube sampling design (LHS) may be used to estimate the parameters. After our N observations, the natural choice for the implementation decision is to choose the decision we believe is best by maximizing $\mu^N(x)$, however this could be modified if we were risk adverse and were less inclined to implement decisions with high uncertainty.

2.7 CONVERGENCE OF APPROXIMATE KNOWLEDGE GRADIENT POLICY

The approximate knowledge gradient policy is a myopic policy by design, but it does not get stuck at a local maximum. The following assumptions are sufficient to show that as the number of iterations goes to infinity, our uncertainty of the function decrease to zero for every decision. In the proof we assume the observation noise is constant, the regression is bounded, and $\mu()$ does not become perfectly does not become perfectly correlated at two distinct decisions:

Assumption 1. $\lambda(x) = \lambda$

Assumption 2. $\limsup_{n\to\infty} |\mu^n(x) - \mu^n(u)|$ *is bounded for every* $x, u \in \mathscr{X}$ *almost surely.*

Assumption 3. *For any* $x \neq u$, $\exists c$ *s.t.* $\limsup_{n\to\infty} |Corr^n(\mu(x), \mu(u))| \le c < 1$ *almost surely.*

With these assumptions, (Scott, Frazier, and Powell 2010) shows that the approximate knowledge gradient policy will find the maximum of the function, formally written in the following theorem.

Theorem 4. *Under the Approximate Knowledge Gradient Policy, if Assumptions 1-3 are satisfied, then* $\lim_{n\to\infty} Var^n[\mu(x)] = 0$ *for all* x.

The proof begins by showing that for any accumulation point of the policy we become certain of the function at the accumulation point, $lim_{n\to\infty} Var^n(\mu(x^{acc})) = 0$. A formal argument is then used to show $\liminf_{n\to\infty} \sup_{x\in\mathscr{X}} \bar{\nu}^{\mathrm{KG},n}(x) = 0$ for every sample path, followed by a proof by contradiction based on the fact that having $\lim_{n\to\infty} Var^n[\mu(x)] > 0$ would result in sufficiently large approximate knowledge gradients. Although this theorem says nothing about the rate of convergence, it would be a weakness to lack this property.

3 NUMERICAL EXPERIMENTS

In this section we compare the approximate knowledge gradient policy (AKG) with sequential kriging optimization (SKO) from (Huang, Allen, Notz, and Zeng 2006). We first compare them on maximizing Gaussian processes in the presence of noise since both algorithms are based on Gaussian process regression. Finally, we compare them, along with LHS, on calibrating an airline simulator.

Table 3: Performance on Gaussian processes where the observations contain normally distributed noise with variance λ. Bold means that the value is statistically significantly different from the competition.

		AKG			SKO		
Test Function	$\sqrt{\lambda}$	$\mathbb{E}(OC)$	$Std(\mathbb{E}(OC))$	$Med(OC)$	$\mathbb{E}(OC)$	$Std(\mathbb{E}(OC))$	$Med(OC)$
GP $(\alpha = .1, \beta = 100)$	$\sqrt{.1}$	.0076	.0057	.0000	.0195	.0041	.0043
$\sigma = 8.417$	$\sqrt{1.0}$	.0454	.0243	.0018	.0888	.0226	.0182
	$\sqrt{10.0}$	.3518	.0587	.0337	.2426	.0216	.0535
GP $(\alpha = 1, \beta = 100)$	$\sqrt{.1}$	**.0077**	.0022	.0000	.0765	.0311	.0000
$\sigma = 9.909$	$\sqrt{1.0}$	**.0270**	.0045	.0000	.1993	.0486	.0255
	$\sqrt{10.0}$	**.4605**	.1028	.0489	.6225	.0669	.1558
GP $(\alpha = 10, \beta = 100)$	$\sqrt{.1}$	**.1074**	.0259	.0000	.5302	.0799	.0000
$\sigma = 10.269$	$\sqrt{1.0}$	**.1846**	.0286	.0000	.6638	.0839	.0839
	$\sqrt{10.0}$	**1.0239**	.1021	.1415	1.8273	.1450	.6290

3.1 PERFORMANCE ON GAUSSIAN PROCESS TEST FUNCTIONS

We generate one-dimensional Gaussian processes over a 300 point grid on the interval [0,15] with the covariance function given in equation (3). Each algorithm is run for 50 observations and each observation contains normally distributed noise with variance λ. We define the opportunity cost as the difference between the true maximum and the truth of the decision we believe is the best; $OC = \max_x \mu(x) - \mu(x^\star)$, where $x^\star$ is the implementation decision. Thus the closer the opportunity cost is to zero, the better. Table 3 summarizes the results of each algorithm with different covariance function parameters and levels of observation noise. Each row corresponds to 500 sample paths, and the bold numbers are significantly better using Welch's t-test at the .05 level. The first 4 sampling decisions for each algorithm are chosen using an LHS design in order to estimate the parameters. At that point the algorithms begin and MLE parameter estimation is done after every observation. Table 3 shows some significant benefits for the approximate knowledge gradient policy on Gaussian processes.

3.2 AIRLINE CALIBRATION

We now demonstrate the approximate knowledge gradient policy on a particular calibration problem of the form given by equations (1) and (2). An industrial computer model simulates one month of a business jet's operations which involves customers' demands for airplanes, crew scheduling, ferrying, and unexpected airplane repairs; the model takes approximately fifteen minutes to simulate a month of the company's operations once. After the program simulates the month of the operations one time, it outputs a set of statistics summarizing the behavior of the company over the month. The statistics of interest include the percentage of airplanes upgraded, $G_1(x)$, and the percentage of airplanes downgraded, $G_2(x)$; if the company cannot provide the exact airplane model requested by the customer, the company can upgrade or downgrade the customer's demand by providing a better or worse plane (each airplane is given a quality number based on its size and age). The other important output statistics are three utilities $(G_3(x), G_4(x), G_5(x))$ which are observable percentages based on the airlines operations. In order for the simulator to be useful, these output statistics should match the previous month's operations of the company fairly closely. The actual statistics of the company are given, and we call them our target goals, $g_1, ..., g_5$.

Before each run of the simulator, we may choose the parameter settings, x, of the simulator. For our particular problem we are allowed to set the maximum the plane can be downgraded, x_1, or upgraded, x_2. Parameters x_3 and x_4 are the simulator's scaling factors for the penalties for downgrading and upgrading a customer's demand; the company provides the customer with an airplane as similar to their request as possible. The last parameter, x_5, controls the simulator's amount of ferrying (flying airplanes without passengers). The company can fly planes without passengers in order to strategically place airplanes and crew in more useful locations.

Our calibration problem requires solving the optimization problem in equation (2). Since each run of the simulator requires fifteen minutes, we limit ourselves to 50 iterations to determine the best parameter settings.

3.3 PERFORMANCE ON CALIBRATION PROBLEM

In this section we compare the algorithms for calibrating the airline simulator. Based on knowledge of plausible parameter values, we specify our domain by requiring $x_1 \in [0,35.0]$, $x_2 \in [0,.4]$, $x_3 \in [20,60]$, $x_4 \in [0,.4]$, $x_5 \in [0,60]$, and the target goals for the output statistics in the objective are set to be $g_1 = 40$, $g_2 = 4$, $g_3 = 60$, $g_4 = 60$, and $g_5 = 60$. For the AKG and SKO, the first 12 observations were chosen by an LHS design in order to obtain MLE estimates of the parameters of the covariance function in equation (3) which are the variance of the observation noise, λ, and the initial mean, μ^0.

We would like the objective given by equation (1), which is negative, to be as large as possible, although an objective of zero is not necessarily achievable by the simulator. For LHS, the implementation decision, $x^\star$, is chosen simply as the sampling decision with the largest (best) observation. For *AKG* in these numerical tests the implementation decision is chosen in the same way as SKO; (Huang, Allen, Notz, and Zeng 2006) gives a exponential utility maximization argument to suggest using $x^\star = \arg\max_{x^0,\ldots,x^{n-1}} \mu^n(x) - \sqrt{\Sigma^n(x,x)}$ for the implementation decision. This avoids implementation decisions with very large uncertainty that $\max_x \mu^n(x)$ may occasionally give.

Figure 1 shows the results of the the algorithms calibrating the simulator, each with 50 observations from the simulator. Each algorithm was run five times, and the five sample paths of the objective are plotted. To produce Figure 1, the estimates of the objective are obtained after the algorithm has run by going back and running the simulator at each of the implementation decisions. The Latin hypercube sampling design appears to have a steadily increasing objective. SKO and AKG have similar performances, although they both appear better than LHS. The output statistics, G_1 through G_5, are all on the scale of 0-100. Limited sample paths are available due to the time-consuming nature of the simulator, but we can visually conclude that the AKG policy is a competitive algorithm.

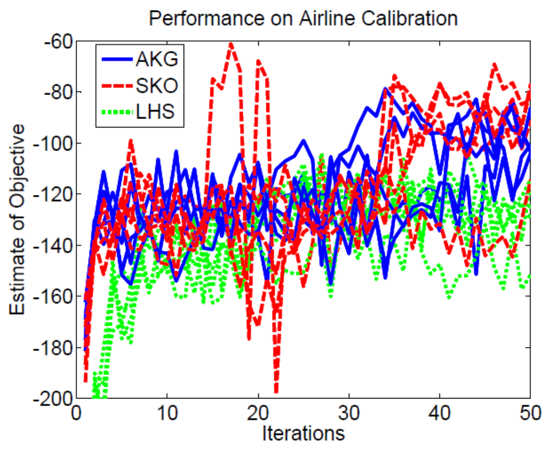

Figure 1: We show the performance of the algorithms on the calibration problem. The estimates of the objective of each algorithm are shown from five sample paths.

We now briefly examine the first sample path of the AKG algorithm by plotting where it sampled as well as how the output statistics of the implementation decisions improve as the number of sampled decisions increases. Figure 2 shows where the policy is sampling in each of the five dimensions. The final implementation decision for this sample path is $x = [20.4, 0.40, 44.5, 0.16, 51.68]^T$. Looking at the histograms in Figure 2, we see that for each dimension of x the AKG policy has explored much of the decision space, but has also spent additional time sampling near the final implementation decision.

4 CONCLUSION

We describe an adaptation of the knowledge gradient for continuous measurement choices, which is then run on test functions and later used for calibrating an industrial simulator. In both scenarios the approximate knowledge gradient is found to be a competitive policy base on the value of an additional measurement. The approximate knowledge gradient policy is a viable method with nice convergence properties that sequentially chooses different parameter settings in order to optimize continuous parameters with a limited number of noisy runs.

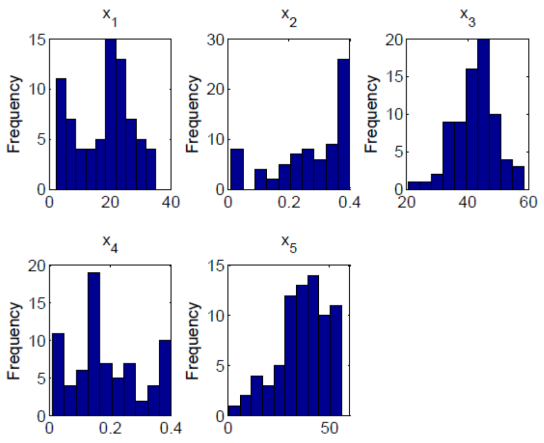

Figure 2: Frequency histograms showing where the AKG policy sampled in each dimension for a particular sample path.

REFERENCES

Cressie, N. 1990. The origins of kriging. *Mathematical Geology* 22 (3): 239–252.

Frazier, P., W. B. Powell, and S. Dayanik. 2009. The knowledge gradient policy for correlated normal beliefs. *INFORMS Journal on Computing* 21:599–613.

Huang, D., T. Allen, W. Notz, and N. Zeng. 2006. Global optimization of stochastic black-box systems via sequential kriging meta-models,. *Journal of Global Optimization* 34:441–446.

Jones, D., M. Schonlau, and W. Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* 13:455–492.

Kleijnen, J. P. 2009. Kriging metamodeling in simulation: A review. *European Journal of Operations Research* 192:707–716.

Kushner, H. J. 1964. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering* 86:97–106.

Locatelli, M. 1997, January. Bayesian algorithms for one-dimensional global optimization. *Journal of Global Optimization* 10 (1): 57–76.

Matheron, G. 1963. Principles of geostatistics. *Economic Geology* 58:1246–1266.

Meinhold, R., and N. Singpurwalla. 1983. Understanding the Kalman Filter. *The American Statistician* 37 (2): 123–127.

Mockus, J. 1993, June. Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization* 4 (4): 347–365.

Rasmussen, C., and C. Williams. 2006. *Gaussian processes for machine learning*. MIT Press.

Sacks, J., W. Welch, T. Mitchell, and H. Wynn. 1989. Design and analysis of computer experiments. *Statistical Science* 4:409–423.

Scott, W., P. Frazier, and W. Powell. 2010. The correlated knowledge gradient for maximizing expensive continuous functions with noisy observations using gaussian process regression.

Zhilinskas, A. G. 1975, January. Single-step bayesian search method for an extremum of functions of a single variable. *Cybernetics and Systems Analysis* 11 (1): 160–166.

AUTHOR BIOGRAPHIES

Warren R. Scott is a fourth year graduate student in the Operations Research and Financial Engineering Department at Princeton University. His research interests include optimal learning, simulation optimization, and optimizing noisy functions over a continuous decision space. His email address is <wscott@princeton.edu>.

Warren B. Powell is a professor in the department of Operations Research and Financial Engineering at Princeton University, where he has taught since 1981. He is director of CASTLE Laboratory which specializes in stochastic optimization for problems in transportation, energy, and health. He has authored/coauthored over 150 refereed papers and is the author of Approximate Dynamic Programming, published by John Wiley. His

email address is <powell@princeton.edu>.

Hugo P. Simão is a senior staff member at CASTLE Laboratory, in Princeton University. He holds a Ph. D. in Civil Engineering and Operations Research from Princeton University. He has worked on planning systems in the areas of transportation, distribution, logistics, and supply-chain. He has also taught courses on computer programming, basic probability and statistics, and resource and information management at the School of Engineering in Princeton. His email address is <hpsimao@princeton.edu>.