Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

ADAPTATION OF THE UOBYQA ALGORITHM FOR NOISY FUNCTIONS

Geng Deng

Department of Mathematics
University of Wisconsin
480 Lincoln Dr.
Madison, WI 53706, U.S.A.

Michael C. Ferris

Computer Sciences Department
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706, U.S.A.

ABSTRACT

In many real-world optimization problems, the objective function may come from a simulation evaluation so that it is (a) subject to various levels of noise, (b) not differentiable, and (c) computationally hard to evaluate. In this paper, we modify Powell's UOBYQA algorithm to handle those real-world simulation problems. Our modifications apply Bayesian techniques to guide appropriate sampling strategies to estimate the objective function. We aim to make the underlying UOBYQA algorithm proceed efficiently while simultaneously controlling the amount of computational effort.

1 INTRODUCTION

Powell's UOBYQA (Unconstrained Optimization BY Quadratic Approximation) (Powell 2002) is a derivative-free algorithm designed for (small scale) unconstrained optimization. The general structure of UOBYQA follows a *model-based* approach, which constructs a series of local quadratic models that approximate the objective function. The algorithm generates its iterates using a trust region framework (Nocedal and Wright 1999). It differs from a classical trust region method in that it creates quadratic models by interpolating a set of sample points instead of using the gradient and Hessian values of the objective function (thus making it a derivative-free tool). Besides UOBYQA, other model-based software includes WEDGE (Marazzi and Nocedal 2002) and NEWUOA (Powell 2004).

We develop a variant of the original UOBYQA, called noisy UOBYQA, that is adapted for noisy optimization problems. These are very common in real-world applications, in particular when the objective function is an associated measurement of an experimental simulation. In this situation, the objective value may be difficult to obtain and the inaccuracy of the objective function often complicates the optimization process. For example, the derivative value is typically unavailable, thus many standard algorithms are not applicable.

We consider the stochastic problem in the following parametric form:

$$\min_{x\in\mathbb{R}^n} F(x) = \mathbb{E}[f(x,\omega(x))]. \tag{1}$$

The underlying function $F(\cdot)$ is unknown and must be estimated. The function $f(\cdot,\omega(x))$ is a sample response function, which is affected by a random factor $\omega(x)$ at x. We normally assume the noise term $\omega(x)$ has mean zero and is independently distributed. As a special case, we consider $\omega(x)$ as 'white noise' in our paper and use the additive form: $f(x,\omega(x)) = F(x)+\omega(x)$.

When noise is present, UOBYQA may behave precariously. For example, a subproblem that minimizes the quadratic model within a trust region may generate poor solutions. The idea of our modification is to control the random error by averaging multiple evaluations per point, helping the algorithm to proceed appropriately. Therefore, we will face the primary issue of handling the tradeoff between two objectives: **having an efficient algorithm versus minimizing computational effort**. On the one hand, we sample more replications to increase the accuracy of estimation of the underlying function F and hence the accuracy of the algorithm; on the other hand, we want to use as little computation as possible. In our modifications, we apply Bayesian techniques to establish 'rules' of evaluating appropriateness of performing a step in UOBYQA. Only when such rules are satisfied, do we proceed with this step of the algorithm.

The noisy UOBYQA algorithm shares similarities to the *Response Surface Methodology* (RSM) (Box and Draper 1986). Both of the methods construct a series of models to the simulation response during the optimization process. However, many features implemented in UOBYQA, such as the quadratic model update, and trust region control, have advantages over the classical RSM approach.

The remainder of the paper is arranged as follows. In Section 2, we give a brief outline of the deterministic UOBYQA algorithm. In Section 3, we present the noisy UOBYQA algorithm, including the following modifications: stabilizing the quadratic models, comparing two candidate points and a new termination criterion. We also show how to optimally allocate computational resources in these procedures. In Section 4, we apply the new noisy UOBYQA algorithm to several numerical examples and compare it with other noisy algorithms. Finally, we design and solve a real simulation model.

2 THE UOBYQA ALGORITHM

We first outline the structure of the standard UOBYQA in the noiseless case. In the following description, we denote the deterministic objective function as $f(\cdot)$.

As we have mentioned, UOBYQA is essentially in the framework of a model-based approach. Starting the algorithm requires an initial trial point x_0 and an initial trust region radius Δ_0. At each iteration k, the derivative estimate of the objective function f is contained in a quadratic model

$$Q_k(x_k+s) = c_Q + g_Q^T s + \frac{1}{2} s^T G_Q s, \tag{2}$$

which is constructed by interpolating a set of well-positioned points $\mathcal{I} = \{y^1, y^2, \ldots, y^L\}$,

$$Q_k(y^i) = f(y^i), \quad i = 1, 2, \ldots, L.$$

The point x_k acts as the center of the trust region, and is the best point in the set $\mathcal{I}$. The interpolative model is expected to well approximate f around the base point x_k, such that the parameters c_Q, g_Q and G_Q approximate the Taylor expansion coefficients of f around x_k. To ensure a unique quadratic interpolator, the number of interpolating points should satisfy

$$L = \frac{1}{2}(n+1)(n+2). \tag{3}$$

As in a classical trust region method, a new promising point is found from a subproblem:

$$\min_{s \in \mathbb{R}^n} Q_k(x_k+s), \quad \text{subject to } \|s\|_2 \le \Delta_k. \tag{4}$$

The solution s^* of (4) is accepted (or not) by evaluating the "degree of agreement" between f and Q_k:

$$\gamma = \frac{f(x_k) - f(x_k+s^*)}{Q_k(x_k) - Q_k(x_k+s^*)}. \tag{5}$$

If the ratio γ is large enough and the new point maintains the "good geometry" of the set $\mathcal{I}$, then the point is accepted into $\mathcal{I}$. Otherwise, we either improve the geometry of $\mathcal{I}$ or update the trust region radius. We always keep track of the best point in the set $\mathcal{I}$. Whenever a new point x^+ enters, it is compared with $f(x_k)$ to determine the best point, which becomes the next iterate $x_{k+1} \leftarrow \arg\min\{f(x_k), f(x^+)\}$.

The following outline of the UOBYQA algorithm contains only the key points regarding the modifications; details can be found in (Powell 2002).

The UOBYQA Algorithm

A starting point x_0, an initial trust region radius Δ_0, and a terminating trust region radius Δ_{end} are given.

1. Generate initial trial points in the interpolation set $\mathcal{I}$. Let iterate x_1 be the best point in $\mathcal{I}$.
2. For iterations $k = 1, 2, \ldots$
   (a) Construct a quadratic model $Q_k(x)$ of the form (2) that interpolates points in $\mathcal{I}$. Solve the trust region subproblem (4).
   (b) Evaluate the function at the new point $x_k + s^*$ and compute the agreement ratio γ in (5).
   (c) Test whether the point is acceptable in the set $\mathcal{I}$. If not, improve the quality $\mathcal{I}$ or update the trust radius Δ_k. If a point is added to the set $\mathcal{I}$, another element in $\mathcal{I}$ should be removed.
   (d) When a new point x^+ is added, determine the best point in $\mathcal{I}$.
   (e) If $\Delta_k \le \Delta_{end}$, terminate the loop.
3. Evaluate and return the final solution point.

3 MODIFICATIONS

In the following subsections, we will present our modifications to the UOBYQA algorithm to deal with problematic points of the algorithm in the noisy case.

3.1 Reducing Quadratic Model Variance

When there is uncertainty in the objective function, the existence of noise can cause erroneous estimations of coefficients of the quadratic model $Q(x)$, say c_Q, g_Q, G_Q, and as a result, generate an unstable solution $x_k + s^*$. To reduce the variance of the quadratic model, we consider generating multiple function values for points $y^j, j = 1, 2, \ldots, L$ in the set $\mathcal{I}$ and use the averaged function value for the interpolation process.

In UOBYQA, the quadratic function is constructed as a linear combination of Lagrange functions $l_j(x)$,

$$Q_k(x) = \sum_{j=1}^{L} f(y^j) l_j(x), \quad x \in \mathbb{R}^n, \tag{6}$$

where L is the cardinality of the set $\mathcal{I}$, as defined in (3). Each $l_j(x)$ is a quadratic polynomial from $\mathbb{R}^n$ to $\mathbb{R}$

$$l_j(x_k+s) = c_j + g_j^T s + \frac{1}{2} s^T G_j s, \quad j = 1,2,\ldots,L,$$

that has the property $l_j(y^i) = \delta_{ij},\ i = 1,2,\ldots,L$, where δ_{ij} is 1 if $i = j$ and 0 otherwise. It follows from (2) and (6) that the parameters of Q_k are derived as

$$\begin{array}{l} c_Q = \sum_{j=1}^L f(y^j) c_j,\ g_Q = \sum_{j=1}^L f(y^j) g_j, \\ \text{and } G_Q = \sum_{j=1}^L f(y^j) G_j. \end{array} \tag{7}$$

Note that the parameters c_j, g_j, and G_j in each Lagrange function l_j are uniquely determined when the y^j are given, regardless of the objective function $f(\cdot)$.

When we have r_j function evaluations for the point y^j, we can compute more accurate estimates of c_Q, g_Q, and G_Q using mean values in (7) in place of $f(y^j)$. We employ Bayesian tools to analytically quantify the distributions of the parameters in (7), which helps us determine the appropriate number of evaluations r_j.

In the Bayesian framework, the unknown mean $\mu(y^j)$ and variance $\sigma^2(y^j)$ of $f(y^j,\omega)$ are considered as random variables, whose distributions are inferred by Bayes' rule. By assuming a non-informative prior distribution, we can estimate the joint posterior distributions of $\mu(y^j)$ and $1/\sigma^2(y^j)$ as

$$\begin{array}{l} \frac{1}{\sigma^2(y^j)} | X \sim Gamma((r_j-1)/2, \hat{\sigma}^2(y^j)(r_j-1)/2), \\ \mu(y^j) | \sigma^2(y^j), X \sim N(\bar{\mu}(y^j), \sigma^2(y^j)/r_j). \end{array} \tag{8}$$

Here we denote the sample mean and sample variance of the data by $\bar{\mu}(y^j)$ and $\hat{\sigma}^2(y^j)$. The gamma distribution $Gamma(\alpha,\beta)$ has mean α/β and variance α/β^2. We use a non-informative prior since there is no prior information on the real distribution.

The distribution of the mean value $\mu(y^j)$ is of most interest to us. When the sample size is large, we can replace the variance $\sigma^2(y^j)$ with the sample variance $\hat{\sigma}^2(y^j)$ in (8), and can asymptotically derive the posterior distribution of $\mu(y^j)|X$ as

$$\mu(y^j)|X \sim N(\bar{\mu}(y^j), \hat{\sigma}^2(y^j)/r_j). \tag{9}$$

According to (7), we treat the c_Q, g_Q and G_Q as random variables from a Bayesian perspective. They all follow normal distributions whose means are estimated using (9) as (all of these are posterior estimates)

$$\begin{array}{ll} \mathbb{E}[c_Q] & = \mathbb{E}\left[\sum_{j=1}^L f(y^j,\omega) c_j\right] \\ & = \mathbb{E}\left[\sum_{j=1}^L \mu(y^j) c_j\right] \\ & = \sum_{j=1}^L \bar{\mu}(y^j) c_j, \\ \mathbb{E}[g_Q] & = \sum_{j=1}^L \bar{\mu}(y^j) g_j, \\ \mathbb{E}[G_Q] & = \sum_{j=1}^L \bar{\mu}(y^j) G_j \end{array} \tag{10}$$

and the variances are estimated as

$$\begin{array}{ll} var(c_Q) & = var\left(\sum_{j=1}^L \mu(y^j) c_j\right) \\ & = \sum_{j=1}^L c_j^2 \hat{\sigma}^2(y^j)/r_j, \\ var(g_Q(i')) & = \sum_{j=1}^L g_j^2(i') \hat{\sigma}^2(y^j)/r_j, \\ var(G_Q(i',j')) & = \sum_{j=1}^L G_j^2(i',j') \hat{\sigma}^2(y^j)/r_j, \\ & i',j' = 1,\ldots,n. \end{array} \tag{11}$$

As r_j increases to infinity, the variance decreases to zero. A more precise computation involving the use of the Student's t-distribution was found to perform similarly.

To increase the stability of a quadratic model, we want to quantify how the randomness in the coefficients c_Q, g_Q, and G_Q affects the solution of the subproblem (4). Suppose we solve N_t trial subproblems whose quadratic model coefficients are extracted from their posterior distributions, we expect that solutions $s^{*(i)}, i = 1,2,\ldots,N_t$ have a small overall variance (see Figure 1). Therefore, we introduce a criterion that constrains the standard deviation of solutions in each coordinate direction, requiring them to be smaller than a threshold value β times the trust region radius:

$$\max_{j=1}^n std([s^{*(1)}(j), s^{*(2)}(j), \ldots, s^{*(N_t)}(j)]) \le \beta \Delta_k. \tag{12}$$

Increasing r_j should help reduce the volatility of coefficients, thus reduce the standard deviations of solutions. Therefore, satisfying the above criterion necessitates a sufficiently large r_j for the point y^j.

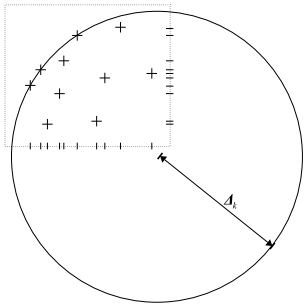

Figure 1: The Trial Solutions within a Trust Region are Projected to Each Coordinate Direction and the Standard Deviations of the Projected Values are Evaluated.

Sequentially allocating computational resources: A new resource allocation question arises (based on r_j function evaluations at site y^j) about which site to assign new function replications in order to satisfy the constraint (12) with the minimum total number of function evaluations. In solving the subproblem (4), we know that only g_Q and G_Q matter

for determining the solution s^* from the definition of $Q_k(x)$ in (2). Instead of satisfying the constraint directly, we aim to control the variance of the most volatile coefficient. We minimize the corresponding ratio of the standard deviation to the expected value

$$\max_{i',j'} \left(\frac{std(g_Q(i'))}{|E[g_Q(i')]|}, \frac{std(G_Q(i',j'))}{|E[G_Q(i',j')]|} \right), \quad i',j' = 1,\ldots,n. \tag{13}$$

We propose a fast but sub-optimal strategy here, one that assigns additional computational resources sequentially. We assume that when an additional batch of r replications at site y^j are newly produced, the sample mean $\bar{\mu}(y^j)$ and variance $\hat{\sigma}^2(y^j)$ remain invariant. Under this assumption, the posterior variance of $\mu(y^j)$ (see (9)) changes from $\hat{\sigma}^2(y^j)/r_j \to \hat{\sigma}^2(y^j)/(r_j+r)$.

First, let $\phi(\vec{r})$ (here $\vec{r} = [r_1, r_2, \ldots, r_L]$) denote the largest quantity in (13):

$$\begin{aligned}
\phi(\vec{r}) &= \max_{i',j'} \left(\frac{std(g_Q(i'))}{E[g_Q(i')]}, \frac{std(G_Q(i',j'))}{E[G_Q(i',j')]} \right) \\
&= \max_{i',j'} \left(\frac{\sqrt{\sum_{j=1}^{L} g_j^2(i')\hat{\sigma}^2(y^j)/r_j}}{\sum_{j=1}^{L} \bar{\mu}(y^j) g_j(i')}, \frac{\sqrt{\sum_{j=1}^{L} G_j^2(i',j')\hat{\sigma}^2(y^j)/r_j}}{\sum_{j=1}^{L} \bar{\mu}(y^j) G_j(i',j')} \right), \\
& i',j' = 1,\ldots,n.
\end{aligned} \tag{14}$$

To achieve a good allocation scheme, we invest our new resources in the point y^j in order to most sharply decrease the quantity $\phi(\vec{r})$. After assigning r samples to the point y^j, we obtain a new vector $\vec{r} + re_j$, where e_j is the standard unit vector in $\mathbb{R}^L$ with 1 on the jth component. We select the site y^j with the index:

$$\arg\max_j \phi(\vec{r} + re_j). \tag{15}$$

The best index is determined by comparing the L different possible options.

Procedure to stabilize the quadratic model:
Given initial sample size r_0, batch size r, and the threshold value β.

1. Generate r_0 function evaluations for each point for pre-estimations of sample mean and sample variance. Set $r_j \leftarrow r_0, j = 1,2,\ldots,L$.
2. Determine the largest quantity $\phi(\vec{r})$ which corresponds to the most volatile coefficient.
3. Select the site for further evaluations using (15).
4. Evaluate r function replications on the selected site and update the sample mean and sample variance.
5. Repeat Steps 2-4, until constraint (12) is satisfied.

3.2 Selecting the Best Point

If x^+ is a new point entering the interpolation set $\mathcal{I}$, we encounter the problem of selecting the best point in $\mathcal{I}$. Since x_k is known to be the best in the previous set, the question becomes how we determine the order of x_k and x^+. When there is noise in the function output, we need more precise estimations of the underlying function value to make the correct decision.

Without loss of generality, the problem is equivalent to solving a discrete-event optimization problem:

$$\arg\min_{j\in\{1,2\}} \mathbb{E}[f(y^j, \omega)]. \tag{16}$$

An extension of the problem is to select the best point among K ($K \geq 2$) points. This general problem is often regarded 'selecting the best system', which is considered one of the principal problems in discrete-event simulation studies. Kim and Nelson (2003) provide a good survey on IZ-R&S procedures.

In our case, since there are only two points involved, we implement a simplified variant of a ranking and selection procedure $\mathcal{OCBA}$ (Chen, Chen, and Yucesan 1999). Suppose we have replicated r_j function values for y^j, let $\mu(y^j)$ and $\sigma^2(y^j)$ be the unknown mean and variance of the output at y^j. The point evidenced by a smaller sample mean is selected:

$$\text{Choose point} \begin{cases} 1, & \text{if } \bar{\mu}(y^1) \leq \bar{\mu}(y^2); \\ 2, & \text{otherwise.} \end{cases}$$

We will quantify our decision-making by the so-called *probability of correct selection* (*PCS*):

$$PCS = Pr(\text{select } y^{(1)} \,|\, \mu(y^{(1)}) \leq \mu(y^{(2)})). \tag{17}$$

Here the notation $\mu(y^{(1)}) \leq \mu(y^{(2)})$ reveals the underlying order of the means. In $\mathcal{OCBA}$, Bayesian methods are applied to estimate the 'posterior' *PCS*. The idea is to construct a posterior distribution for the underlying mean $\mu(y^j)$ using observed data and estimate the *PCS* via joint distributions. We increase the number of samples until the desired accuracy is achieved

$$PCS \geq 1 - \alpha, \tag{18}$$

where α represents a significance level.

We perform a similar analysis using Bayesian inference as in Section 3.1. Without loss of generality, suppose we observe $\bar{\mu}(y^1) \leq \bar{\mu}(y^2)$ and select the first point. The *PCS*

corresponds to the probability of the event $\{\mu(y^1) \le \mu(y^2)\}$,

$$\begin{aligned} PCS &= Pr(\mu(y^1) \le \mu(y^2)) \\ &\sim Pr(\mu(y^1) \le \mu(y^2)|X) \\ &= Pr(\mu(y^1)|X - \mu(y^2)|X \le 0). \end{aligned} \tag{19}$$

The approximate *PSC* in (19) is a tail probability of a normal distribution:

$$PCS \sim Pr\left(N\left(\bar{\mu}(y^1) - \bar{\mu}(y^2), \frac{\hat{\sigma}^2(y^1)}{r_1} + \frac{\hat{\sigma}^2(y^2)}{r_2}\right) \le 0\right). \tag{20}$$

Sequentially allocating computational resources We consider a sequential allocation strategy to assign new computational resources to the points y^1 and y^2, so that we can use the least number of function evaluations to satisfy the rule (18). The change of sample mean and sample variance follows the assumptions in Section (3.1).

To determine the potentially better point, we compare the derivative value:

$$\max_{j\in\{1,2\}} \frac{\partial}{\partial r_j} Pr\left(N\left(\bar{\mu}(y^1) - \bar{\mu}(y^2), \frac{\hat{\sigma}^2(y^1)}{r_1} + \frac{\hat{\sigma}^2(y^2)}{r_2}\right) \le 0\right). \tag{21}$$

It is not hard to find that, since the mean of the joint distribution $\bar{\mu}(y^1) - \bar{\mu}(y^2)$ is unchanged, we desire the largest decrease in the variance $\psi(\vec{r}) := \hat{\sigma}^2(y^1)/r_1 + \hat{\sigma}^2(y^2)/r_2$. Therefore, the problem (21) becomes

$$\min_{j\in\{1,2\}} \psi(\vec{r} + re_j). \tag{22}$$

This determines the index of the potentially better point.

Procedure to select the best point in $\mathcal{I}$:
Given initial sample size r_0, batch size r, and a significance parameter α.

1. Evaluate r_0 function evaluations for each point for pre-estimations of sample mean and sample variance. Set $r_j \leftarrow r_0, j = 1,2$.
2. Select the point to carry out further replications using (22).
3. Evaluate r further function replications on the point selected and update the sample mean and sample variance.
4. Repeat Steps 2 and 3 until the constraint (18) is satisfied.

3.3 New Termination Criterion

In UOBYQA and other model-based optimization algorithms, a test on the norm of the gradient $\|g_k\|$ or the trust region size Δ_k is typically treated as termination criteria; i.e., $\Delta_k \le \Delta_{end} = 10^{-12}$. However, these criteria are not suitable for noisy cases. When Δ_k gets smaller, the estimated value of G_Q, which is a gauge for 'curvature' of the quadratic model, will approach zero. This will inevitably require much more function evaluations in later iterations to reduce the variance of the quadratic model in order to retain accuracy.

The new termination criterion is designed to relax the value of Δ_{end}, such that the algorithm terminates much earlier. We specify a parameter N_{max} that controls the maximum number of replications per site in the algorithm. N_{max} represents the amount of computing we are willing to spend at any site. In each iteration k, we will check that any point x on the edge of the subregion $\{x| \|x - x_b\| \le \Delta_k\}$ is 'separable' from the center x_k, given the control N_{max} on the number of replications per site. Here 'separable' means that one point is better than the other with high accuracy ($PCS \ge 1-\alpha$).

The difficulty occurs in estimations of sample mean $\bar{\mu}(x)$ and sample variance $\hat{\sigma}^2(x)$ of an edge point x, which demand additional function evaluations. We simplify this procedure by performing the separability test using $Q_k(\cdot)$ instead of the original $F(\cdot)$, because Q_k is a good surrogate model of the underlying mean function F when Δ_k is small. We can (a) approximate $\bar{\mu}(x)$ with $Q_k(x)$ and (b) approximate $\hat{\sigma}^2(x)$ with $\hat{\sigma}^2(x_k)$. The second approximation is valid because the variances of function output at two points are very close within a small trust region.

In fact, we enumerate $2n$ edge points (of $\{x| \|x - x_k\| < \Delta_k\}$) which are in standard coordinate directions from x_k, and check their separability from x_k. The $2n$ points consist of a representative set of edge points. The stopping criterion is met when a portion of the representative point set, i.e., 80% of the $2n$ points, are separable from x_k.

In practice, what we first calculate is a least 'separable' distance d. By observing the posterior distribution with the approximations:

$$\begin{aligned} &\mu(x) - \mu(x_k)|X \\ &\sim N(F(x) - F(x_k), \hat{\sigma}^2(x) + \hat{\sigma}^2(x_k)/N_{max}) \\ &\approx N(Q_k(x) - Q_k(x_k), 2\hat{\sigma}^2(x_k)/N_{max}). \end{aligned}$$

We compute d, satisfying

$$Pr(N(d, 2\hat{\sigma}^2(x_k)/N_{max}) \ge 0) \ge 1 - \alpha,$$

via an inverse cdf function evaluation. Then, we say x and x_k are 'separable' if and only if the difference $|Q_k(x) - Q_k(x_b)| \ge d$.

4 NUMERICAL RESULTS

4.1 Numerical Functions

We tested the noisy UOBYQA algorithm on several numerical examples and compared it with two other noisy optimization tools, NOMADm (Nonlinear Optimization for Mixed vAriables and Derivatives in Matlab) (Abramson 2006) and SPSA (Simultaneous Perturbation Stochastic Approximation) (Spall 2003). NOMADm is a pattern search algorithm that implements ranking and selection; and SPSA is a line search algorithm that applies simultaneous perturbation for gradient estimation.

The test function we employed was the well-known extended Rosenbrock function:

$$F(x) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2.$$

The function has a global optima at the point $(1,1,\ldots,1)'_n$, at which the optimal objective value attains 0. A noisy optimization problem was formulated by adding a white noise term to the objective (Section 1). We also applied our algorithm to the ARGLINC function with very similar results.

We assume the random term ω was independent of x and followed a normal distribution with mean zero and variance σ^2, indicating the levels of noise. The starting point x_0 was $(-1.2,1,-1.2,1,\ldots,-1.2,1)'_n$ and the initial trust region radius Δ_0 was set to 2.

Table 1 presents the details about a single-run of noisy UOBYQA to the two-dimensional Rosenbrock function. The variance σ^2 was set to 0.01, which was moderately noisy from our tested noise levels. We used the following default setting of parameters: initial sampling number $r_0 = 3$, which is small but enough to generate sample mean and sample variance; the significance level $\alpha = 0.2$, the threshold value $\beta = 0.4$, the number of trail solutions $N_t = 20$, and the maximum number of evaluations per point $N_{max} = 60$. The new termination criterion we introduced terminated the algorithm earlier, at iteration 81. In fact, we did not observe noticeable improvement in terms of objective value (0.0017 to 0.0016) between the iteration 80 and 120, because the presence of noise deteriorated the accuracy in these iterations. In order to keep the algorithm running and maintaining correct decisions, a large number of function evaluations were necessary. It was therefore preferable to stop the algorithm early.

In the earlier iterations, the algorithm used relatively few function replications. This implied that when noisy effects were small compared to the large function values, the basic operation of the method was unchanged.

In Table 3, we further compare numerical results of UOBYQA, NOMADm, and SPSA. We tested the Rosenbrock function of dimension 2 and 10. Increasing the dimension significantly increased computational burden. To solve a 10-dimensional problem, we need 20000 function evaluations in total. The different scales of noise variance σ^2 were specified to be 0.001, 0.01, 0.1, and 1. We constrained the algorithm with various total number of function evaluations. The parameter setting was the same as before, except that we used different N_{max} when the total number of function evaluations or the variance changed. A practically useful formula for N_{max} was

$$N_{max} = \frac{\text{Max FN}}{\text{Iteration \# (n)}} \cdot \delta(\sigma^2).$$

Here the iteration # represented the estimated number of iterations that the algorithm should maintain for a given dimension. This is a rough estimate and the actual number of iterations varies in different problems. $\delta(\sigma^2)$ was an adjustment value associated with the variance σ^2. The algorithm used a relatively smaller N_{max} when the variance was small. We provide the following suggested values in Table 2.

Table 1: The Performance of Noisy UOBYQA for the Rosenbrock Function, with $n = 2$ and $\sigma^2 = 0.01$.

Iteration (k)	FN	$F(x_k)$	Δ_k
1	1	404	2
20	78	3.56	9.8×10^{-1}
40	140	0.75	1.2×10^{-1}
60	580	0.10	4.5×10^{-2}
80	786	0.0017	5.2×10^{-3}
✓ Stops with the new termination criterion			
100	1254	0.0019	2.8×10^{-4}
120	2003	0.0016	1.1×10^{-4}
✓ Stops with the termination criterion $\Delta_k \leq 10^{-4}$			

Table 2: Suggested Values to Define N_{max}

n	2	4	7	10
Iteration #	50	200	550	1000
σ^2	0.001	0.01	0.1	1
$\delta(\sigma^2)$	2.5	3	3.5	4

As shown in Table 3, for the 2 dimensional case, our algorithms performed better than the other two algorithms. The mean value was the smallest among the peer algorithms. For the 10 dimensional case, when the variance σ^2 was large, our algorithm did not perform well. We should note a fact that making the variance 10 times larger, from our previous theoretical analysis, we potentially require 10 times more function evaluations in order to achieve the same scale of accuracy. That is why the quality of the solutions

decreased sharply as σ^2 increased. Another aspect that influences the performance may be the dimension of the problem. Powell has pointed out that in higher dimensional problems, UOBYQA may not be practically useful because the number of interpolation points L is huge. We think that noisy UOBYQA inherits the limitation in the same way.

Table 3: Apply Noisy UOBYQA to the Rosenbrock Function (Results are Based on 10 Replications of the Algorithms).

n	Noise level σ^2	Max FN	Noisy UOBYQA	NOMADm	SPSA
			Mean Error	Mean Error	Mean Error
2	0.001	200	0.14	0.61	0.42
	0.01	200	0.28	0.63	0.65
	0.1	200	0.44	0.88	0.59
	1	200	0.61	1.44	0.57
	0.001	500	0.099	0.44	0.38
	0.01	500	0.18	0.57	0.35
	0.1	500	0.32	0.77	0.43
	1	500	0.47	1.26	0.49
	0.001	1000	0.024	0.44	0.30
	0.01	1000	0.18	0.46	0.32
	0.1	1000	0.20	0.77	0.34
	1	1000	0.47	1.2	0.42
10	0.001	5000	0.042	0.63	2.3
	0.01	5000	0.42	0.89	2.6
	0.1	5000	0.97	1.24	2.9
	1	5000	7.7	1.78	3.3
	0.001	10000	0.033	0.54	2.3
	0.01	10000	0.15	0.88	2.9
	0.1	10000	0.77	1.2	3.6
	1	10000	7.2	1.66	3.9
	0.001	20000	0.022	0.32	2.1
	0.01	20000	0.12	0.45	2.6
	0.1	20000	0.53	0.50	3.4
	1	20000	5.8	1.1	3.3

4.2 Simulation Problems

In this subsection, we applied the noisy UOBYQA algorithm to solve a pricing problem via simulation. The parameters we considered were the prices of M similar goods in a store, say, $p_1, p_2, \ldots, p_M$. When a customer arrives at the store, he is sequentially exposed to the M goods by the store keeper, from the most expensive good which is of the best quality to the cheapest good. He decides to purchase the item or not after viewing it, but he will buy at most one of them. For good i, we set the probability that the customer purchases the item as

$$Prob_i = \exp(-p_i/\eta_i), \quad i = 1, 2, \ldots, M.$$

The objective of the store keeper is to determine the best combination of prices $\vec{p} = [p_1, p_2, \ldots, p_M]$ which yields the largest expected profit. Running the simulation for a period of time, if we denote the total number of customers as m, and the number of customers who purchase good i as m_i, we formulate the stochastic problem as:

$$\max_{\vec{p}} (\text{Expected Profit}), \text{ where Profit} := \sum_{i=1}^{M} \frac{m_i}{m} p_i.$$

The precise solution is obtained from:

$$\max_{\vec{p}} (\text{Expected Profit}) := \max_{\vec{p}} \sum_{i=1}^{M} \left[\left(\prod_{j=1}^{i-1} (1 - Prob_j) \right) Prob_i p_i \right]. \tag{23}$$

We used Arena 9.0 from Rockwell software to model the pricing system. We considered cases $M = 2$ and $M = 10$, where M also indicated the dimension of the variable $\vec{p}$. In computation of the $Prob_i$, the parameter η_i can reflect the goodness of the merchandize. We set $\eta_1 = 50$ and $\eta_2 = 20$ in the two-dimensional case; set $\eta_1 = 50, \eta_2 = 48, \ldots, \eta_{10} = 32$ in the ten-dimensional case.

The noisy UOBYQA algorithm and OptQuest (the optimization add-on for Arena, <www.opttek.com>) were compared on the stochastic simulation problem. (See results in Table 4.) We implemented the same parametric setting of the algorithm as in the Section 4.1, except that we assigned the initial trust region as 10, and the upper bound for replication usage as 200 and 2000 for the two cases respectively. By varying the number of customers generated in each simulation run, we can have several levels of noisy "profit value". The table shows the different levels of variances of the output. We compared with the real optimal solutions solved using the same η_i's via (23). As we can see in the table, noisy UOBYQA did a uniformly better job than OptQuest, with a higher quality of solutions. For all the tested cases, the gap to the optimal solution was reduced around 50%.

Table 4: Optimization Results of the Pricing Model (over Average Value of 10 Runs), Where the Real Solution for $M = 2$ is 23.23 and for $M = 10$ is 68.28.

Model	Estimated variance	Max FN	Noisy UOBYQA	OptQuest
M=2	0.0022	200	0.10	0.29
	0.014	200	0.25	0.39
	1.1	200	0.45	1.05
M=10	0.0098	2000	0.78	1.32
	0.093	2000	0.89	1.54
	1.1	2000	1.47	2.98

5 CONCLUSIONS

In this paper we modify Powell's UOBYQA algorithm to solve noisy optimization problems. UOBYQA belongs to the class of model-based algorithms that do not require evaluations of the gradient of the objective, therefore, this algorithm is particularly applicable to real-world simulation problems. In the noisy setting, performing more function evaluations can reduce the randomness in function responses. Our approach is motivated by applying analytical Bayesian inference to determine the appropriate number of replications and provide statistical accuracy of the algorithm. At the same time, we have attempted to control the total computational effort.

Note that our method does not assume *Common Random Numbers* (CRN) (Law and Kelton 2000) are implemented in the simulations. Therefore, we cannot generate a deterministic approximation function $\frac{1}{N}\sum_{i=1}^{N} f(\cdot,\omega_i)$ to $F(\cdot)$ by fixing a sequence of samples $\omega_i, i = 1,2,\ldots,N$. The use of CRN is a topic for future research.

The modifications can be generalized to other model-based algorithms, such as the WEDGE algorithm. Our modifications are not intended to be applied to linear model based algorithms, since linear models are more sensitive to noise than quadratic models. In a stochastic algorithm, quadratic models are robust against noise and preferable to use.

REFERENCES

Abramson, M. 2006. NOMADm: mash adaptive direct search Matlab toolbox. <http://www.afit.edu/en/ENC/Faculty/MAbramson/NOMADm.html>.

Box, E. P. G., and N. R. Draper. 1986. *Empirical model-building and response surfaces*. New York: John Wiley & Sons.

Chen, H.-C., C.-H. Chen, and E. Yucesan. 1999. An asymptotic allocation for simultaneous simulation experiments. In *Proceedings of the 1999 Winter Simulation Conference*, 359–366.

Kim, S.-H., and B. L. Nelson. 2003. Selecting the best system: theory and methods. In *Proceedings of the 2003 Winter Simulation Conference*, 101–112.

Law, A., and W. Kelton. 2000. *Simulation modeling and analysis*. Third ed. New York: McGraw-Hill.

Marazzi, M., and J. Nocedal. 2002. Wedge trust region methods for derivative free optimization. *Mathematical Programming* 91:289–305.

Nocedal, J., and S. J. Wright. 1999. *Numerical optimization*. New York: Springer.

Powell, M. J. D. 2002. UOBYQA: Unconstrained optimization by quadratic approximation. *Mathematical Programming* 92:555–582.

Powell, M. J. D. 2004. The NEWUOA software for unconstrained optimization with derivatives. *DAMTP Report 2004/NA05*, University of Cambridge.

Spall, J. C. 2003. *Introduction to stochastic search and optimization*. New York: John Wiley & Sons.

ACKNOWLEDGMENTS

This research was partially supported by Air Force Office of Scientific Research Grant FA9550-04-1-0192, and National Science Foundation Grants DMS-0427689 and IIS-0511905.

AUTHOR BIOGRAPHIES

GENG DENG is a Ph.D. student at the University of Wisconsin–Madison. He is interested in simulation-based optimization, dynamic programming, and applied statistics. He is a member of INFORMS, SIAM, and AMS. He can be reached via <geng@cs.wisc.edu>.

MICHAEL C. FERRIS is Professor of Computer Sciences and Industrial Engineering at the University of Wisconsin–Madison. His research interests are in computational algorithms and modeling, coupled with applications in economics, structural and mechanical engineering and medicine. His work emphasizes practical solution of large scale optimization and complementarity problems. He is a member of the Mathematical Programming Society, SIAM, and INFORMS. His email and web addresses are <ferris@cs.wisc.edu> and <www.cs.wisc.edu/~ferris>.