*INFORMS Journal on Computing*
Vol. 25, No. 2, Spring 2013, pp. 230–243
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.1120.0498




# Stochastic Trust-Region Response-Surface Method (STRONG)—A New Response-Surface Framework for Simulation Optimization

Kuo-Hao Chang
Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 30013, Taiwan, chang@mx.nthu.edu.tw

L. Jeff Hong
Department of Industrial Engineering and Logistics Management, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, hongl@ust.hk

Hong Wan
School of Industrial Engineering, Purdue University, West Lafayette, Indiana 47906, hwan@purdue.edu

Response surface methodology (RSM) is a widely used method for simulation optimization. Its strategy is to explore small subregions of the decision space in succession instead of attempting to explore the entire decision space in a single attempt. This method is especially suitable for complex stochastic systems where little knowledge is available. Although RSM is popular in practice, its current applications in simulation optimization treat simulation experiments the same as real experiments. However, the unique properties of simulation experiments make traditional RSM inappropriate in two important aspects: (1) It is not automated; human involvement is required at each step of the search process; (2) RSM is a heuristic procedure without convergence guarantee; the quality of the final solution cannot be quantified. We propose the stochastic trust-region response-surface method (STRONG) for simulation optimization in attempts to solve these problems. STRONG combines RSM with the classic trust-region method developed for deterministic optimization to eliminate the need for human intervention and to achieve the desired convergence properties. The numerical study shows that STRONG can outperform the existing methodologies, especially for problems that have grossly noisy response surfaces, and its computational advantage becomes more obvious when the dimension of the problem increases.

*Key words*: response surface methodology; trust region method; simulation optimization; black-box method
*History*: Accepted by Marvin Nakayama, Area Editor for Simulation; received March 2010; revised February 2011, June 2011, November 2011, December 2011; accepted December 2011. Published online in *Articles in Advance* April 11, 2012.

## 1. Introduction

Stochastic optimization refers to the minimization (or maximization) of a function in the presence of randomness; this optimization in practice has wide applications. For example, a financial manager may want to design an optimal portfolio to maximize the expected total profits with stochastic asset prices; a production manager may want to decide on an optimal production plan to minimize the expected inventory cost with stochastic customer demands; and an engineer may want to determine the safety parameters so as to optimize the design of a vehicle with stochastic operational conditions.

Among the stochastic optimization problems, some applications have (deterministic) closed-form objective functions and can be solved in minutes by traditional linear or nonlinear optimization techniques, such as portfolio optimization. Others, however, may have little knowledge of the structure of objective functions and require real experiments that may take days or even months to solve, for example, the vehicle collision tests. Statistical methods such as metamodel-based strategies that require only running experiments on solutions are designed to solve these types of problem (Barton and Meckesheimer 2006). Simulation optimization problems are in between those mentioned. The objective function in these problems can be evaluated only by stochastic simulation. A stochastic simulation model is a simplified representation of the actual system. A credible simulation model accounts for details that are important to system performance. In contrast with real experiments, the system's mechanisms or dynamics to be simulated are usually known because discrete-event simulation is a

bottom-up approach, but there is no closed form for the objective function.

Much literature has discussed the methodology development of simulation optimization, for example, Banks (1998), Fu (2002), Tekin and Sabuncuoglu (2004), and Fu (2006). Stochastic approximation (SA) (Kiefer and Wolfowitz 1952) is among the most popular and well-studied methods. One significant advantage of SA is that it is conceptually easy to implement. Moreover, it has a provable convergence guarantee under regularity conditions (Kushner and Yin 1997). However, the convergence of SA is typically slow in practice, and the performance of the algorithm is highly sensitive to the setting of the gain sequence (Andradóttir 2007), which is hard to tune without prior knowledge of the objective function. Some SA variants with improved computational performance have been discussed, for example, in Ljung et al. (1992), Yin and Zhu (1992), and Spall (2000). Many metaheuristics are also developed for black-box problems. For example, the Nelder-Mead simplex method is a direct-search-based method that has been widely used in practice (Barton and Ivey 1996); others include genetic algorithm, tabu search, and scatter search. Different metaheuristics suit different problems. Their disadvantages are that they may require excessive computing time; moreover, they usually have no convergence guarantees (Spall 2003, Sakalauskas and Krarup 2006, Bartz-Beielstein et al. 2010).

Currently, metamodel-based statistical methods have been widely applied in simulation optimization (Barton and Meckesheimer 2006). The most popular one in real-life experiments may be the response surface methodology (RSM). As stated in Myers et al. (2009, p. 1), "RSM is a collection of statistical and mathematical techniques useful for developing, improving, and optimizing processes." The fundamental strategy of RSM is to sequentially approximate the underlying response surface by low-order polynomials within subregions of the domain. RSM was originally proposed as a tool to optimize operating conditions for a chemical process (Box and Wilson 1951) but later evolved to be a general heuristic approach for complex systems (Kleijnen 1998, Wu and Hamada 2000). One of the most significant advantages of RSM is that it applies well-studied statistical techniques, such as the design of experiments and regression analysis, in its framework so that it can enjoy computational efficiency when solving higher-dimensional problems. Moreover, the local metamodel approximation approach requires little prior knowledge about the system. The sequential approach also makes it fit for computer experiments. Over the past three decades, RSM has become one of the most popular heuristics for simulation optimization (Biles and Swain 1979, Hood and Welch 1993, Kleijnen 2008). Full, in-depth coverage of RSM can be found in many classic texts such as Khuri and Cornell (1996) and Myers et al. (2009). Angün et al. (2009) further generalize the framework of RSM, enabling it to deal with stochastic constraints in case of multiple stochastic responses.

Although RSM has significant advantages, its current application in simulation optimization treats simulation experiments the same as physical experiments. However, they have unique properties that are different from physical experiments. First, simulation experiments are usually faster and less expensive. One simulation run often takes only seconds or minutes, but one physical experiment may require days or even months. Second, after the simulation model is constructed, the simulation experiment has minimal monetary costs, but the physical experiment can still remain expensive. The following issues then become relevant for the methodology of simulation optimization: (1) *Large-sample property*. The algorithm is desired to possess asymptotical convergence; that is, when infinite computational effort is allowed, the algorithm should achieve the true optimum of the original problem of interest. Traditional RSM and other metamodel-based algorithms are all heuristic (Barton and Meckesheimer 2006); the quality of the final solution is not guaranteed. (2) *Automation*. In simulation optimization, the number of iterations can be very large, and thus it is impractical for the algorithm to stop and ask for human input at each iteration, as required in traditional RSM (note also that the experimenter may have no idea about the location of the optimum). Moreover, the simulation experiments are conducted on computers, which can facilitate the automation. An automated version of RSM is proposed in Neddermeijer et al. (2000) and Nicolai et al. (2004). As far as we know, the convergence issue is yet to be addressed. Furthermore, in simulation studies, because the inner workings of simulation model are often known, deriving an effective gradient estimator before experimenting is possible. If this is so, the use of gradient information can significantly help the algorithm to achieve better performance, which is not allowed in traditional RSM. These gaps motivate us to develop a new RSM-based framework that is suitable for simulation experiments and is capable of handling problems to which existing methods are not applicable.

We propose the stochastic trust-region response-surface method for unconstrained simulation optimization. For convenience, we refer to it as STRONG. As an improved response surface method, STRONG combines the advantages of traditional RSM with the trust-region method (TRM) (Conn et al. 2000) developed for nonlinear deterministic optimization. The introduction of trust-region concepts helps the

algorithm to achieve the desired convergence and automation properties.

The basic idea of STRONG is described as follows. Similar as traditional RSM, STRONG takes a sequential strategy in search of the optimal solution. At each iteration, a subregion is defined, called "trust region," where a local model, either a first-order or a second-order polynomial, is constructed to approximate the underlying response surface. A linear or quadratic deterministic optimization problem is solved within the trust region. Based on quality of the solution and fidelity of the local model, STRONG will automatically determine whether the new solution should be accepted, select the appropriate local model, and update the size of the trust region for the next iteration.

STRONG is a general framework in which the metamodel construction can be based on an existing gradient estimator (i.e., a white-box approach) or a black-box gradient estimation. When there is no gradient information, STRONG applies the design of experiments (DOE) and regression analysis to construct the metamodel, as traditional RSM does. On the other hand, for simulation models that can provide a gradient estimate, STRONG takes advantage of the gradient information to enhance the computational performance. We will show that STRONG can achieve convergence based on the white-box framework. For the black-box framework, we also show that the convergence analysis is similarly applicable under the assumption that the underlying response surface is quadratic, and it approximates well when the trust region is small but the underlying response surface is not quadratic. Details will be presented in §§4 and 5.

The trust-region concept has been applied in stochastic and simulation optimizations (e.g., Bastin et al. 2006, Deng and Ferris 2009). These approaches are essentially based on the *sample average approximation approach* (SAA), also called *sample path method* or *Monte Carlo sampling approach*, which is a well-recognized method in simulation optimization (Spall 2003, Fu 2002). The basic idea is to generate enough sample paths and then to approximate the expected value function by the sample average function. By fixing a sequence of sample paths, the stochastic problem is converted to a deterministic problem and a proper deterministic method; for example, the TRM can be applied to solve it. Although these methods may seem close to STRONG, there are several crucial differences. First, the distribution of sample paths in STRONG is allowed to vary depending on location; in SAA, however, a fixed set of sample paths is typically used across the decision space. As a result, for problems where the sample path is not identically distributed in the decision space, SAA may not be applicable. Second, STRONG is a new RSM-based algorithm that takes advantage of many well-established statistical tools, such as the DOE technique and regression analysis, and its computational advantage becomes more obvious when the dimension of the problem increases (Kleijnen et al. 2005, Sanchez 2008). By contrast, SAA generates a sequence of sample paths, treats the problem as a deterministic problem, and applies the existing deterministic approach to solve the problem. For large-scale problems, this may require generating and storing a large number of sample paths and thus may not be efficient. Thirdly, for problems where the gradient estimator can be derived, such as infinitesimal perturbation analysis (IPA) or likelihood ratio/score function (LR/SF) (Fu 2006), STRONG allows use of the existing gradient estimator to improve the overall efficiency, which is not possible in SAA. Section 6 will demonstrate the computational advantages of STRONG in which both the DOE and the trust-region updating techniques are employed.

The remainder of this article is organized as follows. In §2, we present the main concepts of STRONG. Then, in §3, we detail the algorithm of STRONG. The convergence analysis of STRONG is presented in §4, and in §5, we discuss the DOE-based gradient and Hessian estimators for black-box problems. In §6, we demonstrate the numerical performances of DOE-based STRONG, and we finish with conclusions and future research in §7.

## 2. Problem Definition

Consider the following simulation optimization problem:

$$\min_{x\in\mathbb{R}^p} \mathrm{E}[G(x)], \tag{1}$$

where $x$ is the vector of continuous decision variables, and $G(x)$ is the stochastic response evaluated at $x$. We assume that $G(x)$ may be obtained by running simulation experiments at $x$.

Let $g(x) = \mathrm{E}[G(x)]$ and $\sigma^2(x) = \mathrm{Var}[G(x)]$. Following the convention of traditional RSM literature, we assume that $G(x)$ follows a normal distribution with mean $g(x)$ and variance $\sigma^2(x)$. Furthermore, we assume that $\sup_{x\in\mathbb{R}^p} \sigma^2(x) < \infty$. Because the optimization is over $\mathbb{R}^p$, which is infinitely large, it is more realistic to allow $G(x)$ to have unequal variances. Note that our formulation in Equation (1) does not include quantiles. However, the proposed STRONG framework may also be applicable to quantile-based problems when some conditions hold.

RSM is a popular choice to solve problem (1). A typical RSM algorithm consists of two stages. In stage I, it constructs a first-order polynomial (in $x$), uses the model to find a better solution in a *region of interest*, and repeats the process. Once a first-order polynomial is no longer appropriate, it switches to stage II, where a second-order (quadratic) polynomial

(in $x$) is constructed to predict the optimal solution. Several statistical tests have been designed to determine the transition between the two stages and the optimality of the solution (Myers et al. 2009). This method has several problems. First, a region of interest needs to be specified at every iteration, which typically requires human involvement. Second, there is no (convergence) guarantee on the optimality of the solution found by the method.

STRONG solves these problems by introducing the concept of "trust region" (Conn et al. 2000). A trust region at a solution $x'$ with a radius $\Delta > 0$ is defined as $\mathcal{B}(x', \Delta) = \{x \in \mathbb{R}^p: \|x - x'\| \le \Delta\}$, where $\|\cdot\|$ denotes the Euclidian norm. At any iteration of the algorithm, for example iteration $k$, the current solution is denoted as $x_k$ and the trust-region radius as $\Delta_k$. Then the trust region at iteration $k$ is $\mathcal{B}(x_k, \Delta_k)$. STRONG uses the trust region as the region of interest and automatically adjusts it, thus avoiding the need for human involvement. Furthermore, the trust region also determines the use of the first- or second-order polynomials and helps the algorithms converge to the set of optimal solutions as the simulation effort goes to infinity.

In this research, the decision variables are assumed properly coded, for example, in the way described in (Myers et al. 2009, p. 23). The advantage of coded variables is that they are "effective for determining the relative size of factor effects" (Montgomery 2005). Note that classic RSM uses the steepest descent method to search for the improved solution; therefore in general it is scale dependent. Kleijnen et al. (2004) proposed a scale-independent gradient estimator that can yield better search directions. This estimator can also be incorporated in STRONG to improve its computational efficiency. In this paper, however, we avoid this complication and simply assume that the steepest descent method is used.

To prove convergence of the STRONG algorithm presented in §3, we will need the following assumption, which is concerned with the functional behavior of the underlying response surface. Our proof closely follows that for the deterministic trust-region method (Conn et al. 2000), though there are significant differences between them. In Assumption 1, we use $\nabla g(x)$ and $H(x)$ to denote the gradient and Hessian at $x$, and "w.p.1" to denote "with probability one."

Assumption 1. *The objective function $g(x)$ is bounded below, twice differentiable, and there exist two positive constants, $\alpha_1$ and $\beta_1$, such that the gradient $\|\nabla g(x)\| \le \alpha_1$ and the Hessian $\|H(x)\| \le \beta_1$, for all $x \in \mathbb{R}^p$.*

Assumption 1 provides some regularity conditions for problem (1). It requires that $g(x)$ has uniformly bounded gradient and Hessian.

## 3. The STRONG Algorithm

STRONG has two stages and an *inner loop*. In stage I, it constructs and optimizes a first-order polynomial, and in stage II it constructs and optimizes a second-order polynomial. The transitions between stages I and II depend on the size of the trust region. If $\Delta_k$ is large, which implies that a first-order polynomial may suffice to find an improved solution, a first-order polynomial will be used to estimate the underlying response surface; otherwise a second-order polynomial will be used for a better fitting. In both stages I and II (which we call the *outer loop*), the algorithm uses a fixed number of observations to construct the models. However, the noise in the estimates may cause a poor fitting of the surface. Therefore when the algorithm fails to generate a good solution in stage II, it will enter an *inner loop* where additional simulation effort is allocated to reduce error in function and gradient estimates.

At any iteration of the algorithm, for example iteration $k$, STRONG conducts the following four steps:

*Step* 1. Construct a local model $r_k(x)$ around the center point $x_k$.

*Step* 2. Solve $x_k^* \in \arg\min\{r_k(x): x \in \mathcal{B}(x_k, \Delta_k)\}$.

*Step* 3. Simulate several observations at $x_k^*$ and estimate $g(x_k^*)$.

*Step* 4. Conduct so-called ratio-comparison (RC) and sufficient-reduction (SR) tests to examine the quality of $x_k^*$ and to update $x_{k+1}$ and the size of the trust region $\Delta_{k+1}$.

These four steps and the sample-allocation scheme in the *inner loop* are cornerstones of STRONG. In the rest of this section, we give a more detailed description of these steps. We use $k_i$ to denote the $i$th *inner loop* of the $k$th iteration when the algorithm enters the *inner loop*. The full algorithm is provided in Figures 1 and 2 in the online supplement (available at http://dx.doi.org/10.1287/ijoc.1120.0498).

### 3.1. Estimation of Gradients and Hessians

At any iteration of the *outer loop* of STRONG, for example, iteration $k$, there is a center point $x_k$. Then several simulation replications are taken to estimate $g(x_k)$, $\nabla g(x_k)$, and $H(x_k)$. Based on the size of the trust region, the algorithm decides to construct a first- or second-order polynomial,

$$r_k(x) = \hat{g}_k(x_k) + \hat{\nabla} g_k^T(x_k)(x - x_k) \quad \text{and}$$
$$r_k(x) = \hat{g}_k(x_k) + \hat{\nabla} g_k^T(x_k)(x - x_k) + \tfrac{1}{2}(x - x_k)^T \hat{H}_k(x_k)(x - x_k),$$

respectively. If the algorithm is in the $i$th *inner loop* of $k$th iteration, it constructs a second-order polynomial

at the center point $x_k$, which is

$$r_{k_i}(x) = \hat{g}_{k_i}(x_k) + \hat{\nabla} g_{k_i}^T(x_k)(x - x_k) + \tfrac{1}{2}(x - x_k)^T \hat{H}_{k_i}(x_k)(x - x_k),$$

where $\hat{g}_{k_i}(x_k)$, $\hat{\nabla} g_{k_i}(x_k)$, and $\hat{H}_{k_i}(x_k)$ are the estimates of the value, gradient, and Hessian of the center point $x_k$ at the $i$th *inner loop* of the $k$th iteration.

To construct these models, we need the estimates of $g(x_k)$, $\nabla g(x_k)$, and $H(x_k)$. To simplify the presentation, we suppose that there is a random vector $D(x)$ at any $x$ such that $\nabla g(x) = \mathrm{E}[D(x)]$. Note that many methods, including infinitesimal perturbation analysis (Ho and Cao 1991, Cao 1994) and the likelihood ratio or score function method (Glynn 1990, Rubinstein and Shapiro 1993), can be used to derive $D(x)$. Fu (2006) provided many practical examples, including the stochastic activity network and the inventory system, among others, in which the unbiased gradient estimators can be derived. Spall (2003) also has in-depth discussions about gradient estimation and the associated regularity conditions for them to be unbiased. In §5, we discuss how to use linear regression to simultaneously estimate $\nabla g(x)$ and $H(x)$ when $D(x)$ is not readily available. In this section and succeeding sections, however, we assume that $D(x)$ is available.

Let $G_1(x), \ldots, G_n(x)$ and $D_1(x), \ldots, D_m(x)$ denote independent and identically distributed (i.i.d.) random variables of $G(x)$ and $D(x)$, respectively. We let the averages

$$\bar{G}(x, n) = \frac{1}{n}\sum_{i=1}^{n} G_i(x) \quad \text{and} \quad \bar{D}(x, m) = \frac{1}{m}\sum_{i=1}^{m} D_i(x). \tag{2}$$

Suppose that the center point $x_k$ has $n_k$ and $m_k$ (or $n_{k_i}$ and $m_{k_i}$, if in the *inner loop*) replications of $G(x_k)$ and $D(x_k)$. Then we have the estimators $\hat{g}_k(x_k) = \bar{G}(x_k, n_k)$, and $\hat{\nabla} g_k(x_k) = \bar{D}(x_k, m_k)$ (or $\hat{g}_{k_i}(x_k) = \bar{G}(x_k, n_{k_i})$, and $\hat{\nabla} g_{k_i}(x_k) = \bar{D}(x_k, m_{k_i})$, if in the *inner loop*) as the estimates of $g(x_k)$ and $\nabla g(x_k)$.

Assumption 2. *The estimators of $g(x)$ and $\nabla g(x)$ satisfy $\sup_{x\in\mathbb{R}^p} |\bar{G}(x, n) - g(x)| \to 0$ w.p.1 as $n \to \infty$ and $\sup_{x\in\mathbb{R}^p} \|\bar{D}(x, m) - \nabla g(x)\| \to 0$ w.p.1 as $m \to \infty$.*

Assumption 2 provides some regularity conditions on the estimators of both $g(x)$ and $\nabla g(x)$. It requires that both $\bar{G}(x, n)$ and $\bar{D}(x, m)$ follow uniform law of large numbers (ULLN). Notably, for any strongly consistent estimator, it is possible to extend it to ULLN as long as the conditions given in Andrews (1992) are satisfied.

The Hessian matrix $H(x_k)$ is often difficult to estimate directly. In this paper, we suggest using the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method as in deterministic nonlinear optimization (Nocedal and Wright 1999) shown as follows:

$$\hat{H}_k = \hat{H}_{k-1} - \frac{\hat{H}_{k-1} s_{k-1} s_{k-1}^T \hat{H}_{k-1}}{s_{k-1}^T \hat{H}_{k-1} s_{k-1}} + \frac{y_{k-1} y_{k-1}^T}{y_{k-1}^T s_{k-1}}, \tag{3}$$

where $\hat{H}_k = \hat{H}_k(x_k)$, $\hat{H}_{k-1} = \hat{H}_{k-1}(x_{k-1})$, $y_{k-1} = \hat{\nabla} g_k(x_k) - \hat{\nabla} g_{k-1}(x_{k-1})$, $s_{k-1} = x_k - x_{k-1}$. The initial $\hat{H}_0(x_0)$ can be either estimated by using the finite-difference method at $x_0$ or simply set as an identity matrix. Furthermore, we impose an upper bound on the norm of $\hat{H}_k$. We let

$$\hat{H}_k := \kappa \cdot \hat{H}_k / \|\hat{H}_k\|$$

if $\hat{H}_k$ calculated by Equation (3) has a norm larger than a large positive constant $\kappa$. In the *inner loop*, because the Hessian matrix will not affect the algorithm convergence, $\hat{H}_{k_i}$ is simply set as $\hat{H}_k$.

It is remarkable that the convergence of STRONG does not require the convergence of $\hat{H}(x)$. The BFGS formula introduced in Equation (3) approximates the Hessian matrix well in deterministic cases and it helps the BFGS algorithm achieve a superlinear rate of convergence (Nocedal and Wright 1999). Therefore, we also suggest using it in STRONG.

### 3.2. Estimation of a Better Solution

Once the local model is constructed, we need to use it to find a better solution within the trust region. In this section, we discuss this step. Without loss of generality, we consider only the *outer loop*, where the model is either first- or second-order. If the algorithm is in the *inner loop*, it is the same as the second-order polynomial of the *outer loop*, except for the notation.

At iteration $k$, we have the current center point $x_k$, a trust region $\mathcal{B}(x_k, \Delta_k)$, and a constructed response model $r_k(x)$ that is either first- or second-order. Ideally, we may set

$$x_k^* \in \arg\min\{r_k(x)\colon x \in \mathcal{B}(x_k, \Delta_k)\},$$

which is the best solution within the trust region predicted by the model $r_k(x)$. However, seeking an exactly optimal solution within the trust region may be time consuming and is certainly not necessary for STRONG to converge. Instead, we follow the deterministic TRM to find a Cauchy point $x_k^*$ that is a nearly optimal solution (Nocedal and Wright 1999). The Cauchy point calculation is as follows:

Cauchy point calculation:

*Step* 1. Find the steepest descent direction $d_k = \arg\min\{\hat{g}_k(x_k) + \hat{\nabla} g_k^T(x_k) d\colon \|d\| \le \Delta_k\}$.

*Step* 2. Choose a step size $\tau_k = \arg\min\{r_k(\tau d_k): \tau > 0, \|\tau d_k\| \le \Delta_k\}$.

*Step* 3. Let $x_k^* = x_k + \tau_k d_k$.

In the following lemma, we show that the Cauchy point provides a sufficient reduction on the model surface, which is critical to prove the convergence of STRONG. The proof of the lemma is provided in the Online Supplement.

Lemma 1. *For any iteration $k$, if the algorithm is in stage I, stage II, or the ith inner loop, respectively,*

$$r_k(x_k) - r_k(x_k^*) \ge \zeta_k := \|\hat{\nabla} g_k(x_k)\| \cdot \Delta_k,$$

$$r_k(x_k) - r_k(x_k^*) \ge \zeta_k := \frac{1}{2}\|\hat{\nabla} g_k(x_k)\| \cdot \min\left\{\frac{\|\hat{\nabla} g_k(x_k)\|}{\|\hat{H}_k(x_k)\|}, \Delta_k\right\},$$

$$r_{k_i}(x_k) - r_{k_i}(x_{k_i}^*) \ge \zeta_{k_i} := \frac{1}{2}\|\hat{\nabla} g_{k_i}(x_k)\| \min\left\{\frac{\|\hat{\nabla} g_{k_i}(x_k)\|}{\|\hat{H}_{k_i}(x_k)\|}, \Delta_{k_i}\right\}.$$

### 3.3. Acceptance/Rejection of the New Solution

There are two tests in the algorithm to decide whether the new solution should be accepted; namely, the ratio-comparison (RC) test and the sufficient-reduction (SR) test. The RC test compares the "observed reduction" with the "predicted reduction" given by the model and decides whether the model in the trust region is trustworthy. The SR test decides whether the observed reduction is statistically significant. For every new solution, the RC test is conducted first, followed by the SR test if it passes the RC test. In the deterministic TRM, there is only the RC test. In the stochastic settings, however, we also need the SR test to ensure that the observed reduction is not entirely caused by noise in the estimates.

**3.3.1. Ratio-Comparison Test.** We first consider the *outer loop*. At iteration $k$, once a Cauchy point $x_k^*$ is identified, $n_0$ simulated observations will be allocated to estimate $g(x_k^*)$. We let $\hat{g}_k(x_k^*) = \bar{G}(x_k^*, n_0)$. Note that although $n_0$ replications are always allocated to any new solutions $x_k^*$ identified in the *outer loop*, $x_k$ may have more than $n_0$ replications; i.e., $n_k$ may be larger than $n_0$ because $x_k$ may have been chosen by an *inner loop* that allocates more replications in the previous iteration to improve the estimate of $g(x_k)$. In this case, $n_k$ replications will be used for $x_k$.

The RC test computes

$$\rho_k = \frac{\hat{g}_k(x_k) - \hat{g}_k(x_k^*)}{r_k(x_k) - r_k(x_k^*)}.$$

Let $0 < \eta_0 < \eta_1 < 1$, for example, we may set $\eta_0 = 1/4$, $\eta_1 = 3/4$. If the ratio is large ($\rho_k \ge \eta_1$), which implies that the new observed solution is significantly better than the current one, the new solution will be accepted. If the ratio is moderate ($\eta_0 \le \rho_k < \eta_1$), which implies that the observed reduction has fair agreements with the predicted reduction, the procedure will also accept the new solution. If $\rho_k$ is close to zero or negative ($\rho_k < \eta_0$), which implies that the observed reduction does not agree with the predicted reduction, the new solution will be rejected. This RC test is similar to the one in the deterministic TRM.

The same idea applies to the *inner loop*, except that $x_k$ and $x_{k_i}^*$ have $n_{k_i}$ and $n_{k_i}^*$ replications, respectively. The details of how sample size changes in the *inner loop* will be discussed later in §3.4. Let $\hat{g}_{k_i}(x_k) = \bar{G}(x_k, n_{k_i})$ and $\hat{g}_{k_i}(x_{k_i}^*) = \bar{G}(x_{k_i}^*, n_{k_i}^*)$. In the *inner loop*, we use

$$\rho_{k_i} = \frac{\hat{g}_{k_i}(x_k) - \hat{g}_{k_i}(x_{k_i}^*)}{r_{k_i}(x_k) - r_{k_i}(x_{k_i}^*)}$$

to conduct the RC test.

**3.3.2. Sufficient-Reduction Test.** The RC test does not take into consideration that the observed reduction is estimated and that the ratio is subject to sampling error. To ensure that the reduction is sufficient and statistically significant, an SR test is further conducted. For any iteration $k$ of the *outer loop*, the new solution $x_k^*$ is said to yield *sufficient reduction* if $g(x_k) - g(x_k^*) > \eta_0^2\zeta_k$, and for any $i$th *inner loop* of iteration $k$, the new solution $x_{k_i}^*$ is said to yield *sufficient reduction* if $g(x_k) - g(x_{k_i}^*) > \eta_0^2\zeta_{k_i}$, where both $\zeta_k$ and $\zeta_{k_i}$ are defined in Lemma 1.

Then the SR test is defined as

$$H_0\colon\ g(x_k) - g(x_k^*) \le \eta_0^2\zeta_k \quad \text{versus}$$
$$H_1\colon\ g(x_k) - g(x_k^*) > \eta_0^2\zeta_k$$

in the *outer loop* and

$$H_0\colon\ g(x_k) - g(x_{k_i}^*) \le \eta_0^2\zeta_{k_i} \quad \text{versus}$$
$$H_1\colon\ g(x_k) - g(x_{k_i}^*) > \eta_0^2\zeta_{k_i}$$

in the *inner loop*. In both tests, the type I error is set as $\alpha_k$. If $H_0$ is rejected, we then conclude that the new solution yields a sufficient reduction.

In general, hypothesis testing implies that the null-hypothesis $H_0$ is rejected only in case of strong counterevidence; in our case, $H_0$ is rejected if the new solution is "much" lower than the original solution. For the SR test in the *outer loop*, we let $S^2(x_k, n_k)$ and $S^2(x_k^*, n_0)$ denote the heterogeneous variances of $x_k$ and $x_k^*$ (the so-called Behrens-Fisher problem), and we let

$$S_k^2 = \frac{S^2(x_k, n_k)}{n_k} + \frac{S^2(x_k^*, n_0)}{n_0},$$

$$df = S_k^4 \cdot \left[\frac{(S^2(x_k, n_k)/n_k)^2}{n_k - 1} + \frac{(S^2(x_k^*, n_0)/n_0)^2}{n_0 - 1}\right]^{-1}.$$

We then compute

$$t^* = \frac{\hat{g}_k(x_k) - \hat{g}_k(x_k^*) - \eta_0^2 \zeta_k}{S_k}$$

and reject $H_0$ if $t^* > t_{1-\alpha_k, df}$ (Montgomery 2005).

For the SR test in the *inner loop*, we can similarly compute the test statistics. Therefore we omit the details, except to note that the type I error is set to $\alpha_k$ for all *inner loops* of iteration $k$. To ensure the convergence of STRONG, we require that the sequence of $\{\alpha_k\}$ satisfies $\sum_{k=1}^{\infty} \alpha_k < \infty$.

A solution that passes the SR test becomes the center point of the next iteration, i.e., $x_{k+1} = x_k^*$ in the *outer loop* and $x_{k+1} = x_{k_i}^*$ in the *inner loop*. If a solution is rejected by the SR test, the center point remains until a satisfactory solution is found.

**3.3.3. Updating the Size of Trust Region.** To implement STRONG, the user needs to select an initial size of the trust region, denoted as $\Delta_0 > 0$, and a threshold $\tilde{\Delta} > 0$; typically $\Delta_0 > \tilde{\Delta}$. In any iteration, such as iteration $k$, if $\Delta_k > \tilde{\Delta}$, STRONG uses a first-order polynomial to fit the response surface; otherwise, a second-order polynomial is used.

STRONG automatically updates the size of the trust region based on the results of the RC and SR tests. Let $0 < \gamma_1 < 1 < \gamma_2$, for example, $\gamma_1 = 1/2$ and $\gamma_2 = 2$. In stage I, if $x_k^*$ fails either the RC or SR test, the center point remains and the trust region shrinks, i.e., $x_{k+1} = x_k$ and $\Delta_{k+1} = \gamma_1 \Delta_k$; if $\rho_k \geq \eta_1$ and $x_k^*$ passes the SR test, the center point then moves to the new solution and the trust region enlarges, i.e., $x_{k+1} = x_k^*$ and $\Delta_{k+1} = \gamma_2 \Delta_k$. If $\eta_0 \leq \rho_k < \eta_1$ and $x_k^*$ passes the SR test, then the center point will move to the new solution and the trust region remains; i.e., $x_{k+1} = x_k^*$ and $\Delta_{k+1} = \Delta_k$.

Once the size of the trust region falls below $\tilde{\Delta}$, the procedure goes to stage II. Based on the same concept, the size of the trust region is updated as in stage I. However, because the trust region is typically small in stage II, STRONG is specially careful about further shrinking the trust region. Specifically, if the new solution fails either in the RC test or in the SR test in stage II, the trust region does not shrink. Instead, the algorithm goes to the *inner loop* to collect more information. In the *inner loop*, the size of the trust region is updated based only on the RC test, and the SR test is used only to determine whether the *inner loop* should be stopped. Once a satisfactory solution is found in the *inner loop*, such as $x_{k_i}^*$, STRONG terminates the *inner loop*, moves to the new solution, i.e., $x_{k+1} = x_{k_i}^*$, and resumes the size of the trust region at iteration $k$ *before* entering the *inner loop*, so $\Delta_{k+1} = \Delta_k$. Note that this mechanism also implies that $\Delta_k \geq \gamma_1 \tilde{\Delta}$ for any iteration $k$.

### 3.4. Inner Loop

Stages I and II both may find no satisfactory solution that can pass the RC and SR tests. The *inner loop* is designed so that it can always find a satisfactory solution, thus avoiding the algorithm getting stuck at a suboptimal solution (i.e., a solution that is neither locally nor globally optimal).

Two mechanisms in the *inner loop* help the algorithm find a satisfactory solution. First, the quality of the local model will be continuously improved as the *inner loop* continues. The algorithm will increase the sample sizes of the value and gradient estimators to improve their precision. In this way, the model can achieve the desired precision to yield a satisfactory solution. Second, the sample sizes of the current and new solutions are also increased to reduce the sampling error. This allows the algorithm to correctly determine the acceptance or rejection of a new solution and the size of the next trust region.

Let $n_{k_i}^*$ and $n_{k_i}$ denote the sample sizes for estimating the value of the new solution $x_{k_i}^*$ and the center point $x_k$ in the $i$th *inner loop* of the $k$th iteration of STRONG. To prevent the trust region from shrinking to zero before the algorithm finds a satisfactory solution, we require $n_{k_i}^*$ to satisfy the following inequality

$$n_{k_i}^* \geq (\lceil 1/\gamma_1^4 \rceil + 1) \cdot n_{k_{i-1}}^*. \tag{4}$$

Because the center point may have obtained samples from previous iterations, we let

$$n_{k_i} = \max\{n_{k_{i-1}}, n_{k_i}^*\}. \tag{5}$$

For the sample size of the gradient estimator in the $i$th *inner loop* of the $k$th iteration, we require

$$m_{k_i} \geq (\lceil 1/\gamma_1^2 \rceil + 1) \cdot m_{k_{i-1}}. \tag{6}$$

Equations (4)–(6) are developed to help the algorithm achieve convergence (see proof of Lemma 2).

## 4. Convergence Analysis

The *inner loop* is mainly used to ensure the convergence of STRONG. We analyze it in detail. Note that $\hat{g}_{k_i}(x_k) = \bar{G}(x_k, n_{k_i})$ and $\hat{\nabla} g_{k_i}(x_k) = \bar{D}(x_k, m_{k_i})$. By Assumption 2 and Equations (4)–(6), we can prove Lemma 2, which states that the estimation errors in both the value and gradient of the center point $x_k$ are bounded by $\Delta_{k_i}^2$ and $\Delta_{k_i}$, respectively, whenever the *inner loop* counter $i$ is sufficiently large. Note that Lemmas 2–4 are for the *inner loop*, in which the center point $x_k$ is fixed, and $\{\Delta_{k_i}, i = 0, 1, 2, \ldots\}$ is a deterministic sequence (because $\Delta_{k_i}$ shrinks by $\gamma_1$ for each additional *inner loop*). We use "i.o." to denote "infinitely often," which refers to the limit when $i \to \infty$.

Lemma 2. *Suppose that Assumptions* 1 *and* 2 *hold. Then, for any* $x_k \in \mathbb{R}^p$ *and given* $k$,

$$\Pr\{|\hat{g}_{k_i}(x_k) - g(x_k)| > \Delta_{k_i}^2 \ i.o.\} = 0, \tag{7}$$

$$\Pr\{\|\hat{\nabla} g_{k_i}^T(x_k) - \nabla g^T(x_k)\| > \Delta_{k_i} \ i.o.\} = 0. \tag{8}$$

In the next lemma, we show that the difference between the metamodel prediction and the observed value at the new solution is bounded by an error term that is of order of $\Delta_{k_i}^2$.

Lemma 3. *Suppose that Assumptions* 1 *and* 2 *hold. Then, for any* $x_k \in \mathbb{R}^p$ *and given* $k$,

$$\Pr\{|r_{k_i}(x_{k_i}^*) - \hat{g}_{k_i}(x_{k_i}^*)| > c \cdot \Delta_{k_i}^2 \ i.o.\} = 0 \tag{9}$$

*for some constant* $c > 0$.

In the next lemma, we show that the *inner loop* at any iteration $k$ can always find a satisfactory solution, one that can pass both the RC test and the SR test, if the center point $x_k$ is not a stationary point, i.e., if $\|\nabla g(x_k)\| > 0$.

Lemma 4. *Suppose that Assumptions* 1 *and* 2 *hold. Then, for any* $x_k \in \mathbb{R}^p$ *and given* $k$, *if* $\|\nabla g(x_k)\| > 0$, *the algorithm can always find a new satisfactory solution in iteration* $k$.

After analyzing the *inner loop*, we now show the convergence of STRONG by analyzing the *outer loop.* Lemmas 5 and 6 and Theorem 1 are for the *outer loop.* To present the convergence proof, we need some additional notations. For every iteration $k$, we let $\hat{\nabla} g_k'(x_k)$, $\hat{H}_k'(x_k)$, and $\Delta_k'$ denote the estimates of gradient and Hessian at $x_k$ and the trust-region size *at the end of iteration* $k$ before the algorithm moves to iteration $k+1$. If STRONG finds a satisfactory solution in the *outer loop*, i.e., $x_{k+1} = x_k^*$, then $\hat{\nabla} g_k'(x_k) = \hat{\nabla} g_k(x_k)$ and $\Delta_k' = \Delta_k$; if it finds a satisfactory solution in the *inner loop*, such as the $i^{th}$ *inner loop*, i.e., $x_{k+1} = x_{k_i}^*$, then $\hat{\nabla} g_k'(x_k) = \hat{\nabla} g_{k_i}(x_k)$ and $\Delta_k' = \Delta_{k_i}$.

Suppose the algorithm yields an infinite (random) sequence of solutions $\{x_k\}_{k=0}^{\infty}$. Because $\Delta_k'$ denotes the trust-region size when the *inner loop* at iteration $k$ is terminated, and because each iteration $k$ may have a different number of passes through the *inner loop*, $\{\Delta_k', k \geq 0\}$ is a random sequence. Lemma 5 indicates that the size of the trust region is bounded away from zero almost surely if $\|\hat{\nabla} g_k'(x_k)\|$ is bounded away from zero.

Lemma 5. *Suppose that Assumptions* 1 *and* 2 *hold. Then, w.p.*1, $\liminf_{k\to\infty} \Delta_k' > 0$ *if* $\liminf_{k\to\infty} \|\hat{\nabla} g_k'(x_k)\| > 0$.

Let $k_j$, $j = 1, 2, \ldots$, denote a subsequence of $k = 1, 2, \ldots$. In the next lemma, we show that if there exists a subsequence of $x_k$ whose gradient estimates converge to zero, the actual gradients of this subsequence will also converge to zero. It turns out that, to prove convergence of the gradients, we need only to analyze the convergence of gradient estimates.

Lemma 6. *Suppose that Assumptions* 1 *and* 2 *hold. If there is a subsequence of* $\{x_k\}_{k=0}^{\infty}$, *denoted as* $\{x_{k_j}\}_{j=0}^{\infty}$, *such that* $\lim_{j\to\infty} \|\hat{\nabla} g_{k_j}'(x_{k_j})\| = 0$, *then* $\lim_{j\to\infty} \|\nabla g(x_{k_j})\| = 0$ *w.p.*1.

With these lemmas, we have the following theorem on the convergence of STRONG.

Theorem 1. *Suppose that Assumptions* 1 *and* 2 *hold. If STRONG has infinitely many successful iterations, then* $\liminf_{k\to\infty} \|\nabla g(x_k)\| = 0$ *w.p.*1.

The conclusion of Theorem 1 is a typical convergence condition for a nonlinear optimization algorithm; see, for instance, page 46 of Nocedal and Wright (1999). It shows that there exists at least a subsequence of solutions of which the true gradient goes to zero w.p.1. Furthermore, based on the similar argument in Theorem 6.4.6 in Conn et al. (2000), we believe that the conclusion of Theorem 1 can be extended to a stronger result: $\lim_{k\to\infty} \|\nabla g(x_k)\| = 0$ w.p.1.

## 5. Black-Box Estimation of Gradient and Hessian Matrix

In §3, we assumed a strongly consistent gradient estimator and suggested using the BFGS method to obtain estimates of the Hessian matrix. In this section, we discuss a situation where no gradient estimator exists, and the simulation model is considered as a black box. We suggest the use of DOE and regression analysis for obtaining estimates of the gradient and Hessian matrix.

The DOE approach is widely adopted in classical RSM. It has several advantages. First, when the local model is a second-order polynomial (stage II or the *inner loop*), the linear and quadratic terms give the gradient and Hessian matrix estimations, respectively. In other words, the regression analysis can estimate the gradient and Hessian matrix simultaneously. Second, DOE has been demonstrated to enjoy computational advantages compared to random sampling (Montgomery 2005, Sanchez 2008). This efficiency gain is more obvious when the dimension of the problem increases. In this section, we will show how to apply the black-box estimation to STRONG and also analyze the convergence of STRONG. The numerical evaluations in §6 shows the efficiency gain of incorporating experimental designs.

The DOE approach works as follows: at each *outer loop* or *inner loop*, there is a center point. We use DOE techniques to determine several input combinations

(design points) to run simulation experiments. The collected observations will be used to estimate the gradient (and Hessian, if necessary) at the center point. Without distinguishing *outer loops* and *inner loops*, we let $x$ denote the center point, $n$ the number of replications at $x$, and $m$ the number of design points. Let $X_m$ denote the design matrix with only the main effect columns, and $\tilde{X}_m$ the design matrix with the main effect, quadratic, and interaction columns. Note that $X_m$ is used to estimate gradient and $\widetilde{X}_m$ to estimate both gradient and Hessian simultaneously. Let

$$Y_m = (y_1, \ldots, y_m)^T - \bar{G}(x, n)$$

be the centralized response vector, where $y_j$ is the output of the simulation experiment conducted at the design point $x_j$. We then estimate $\nabla g(x)$ and $H(x)$ as follows:

$$\hat{\nabla} g(x) = (X_m^T X_m)^{-1} X_m^T Y_m \quad \text{(estimating only the gradient)}, \tag{10}$$

$$\begin{bmatrix} \hat{\nabla} g(x) \\ \hat{H}(x) \end{bmatrix} = (\tilde{X}_m^T \tilde{X}_m)^{-1} \tilde{X}_m^T Y_m \quad \text{(estimating the gradient and Hessian)}, \tag{11}$$

where the gradient and Hessian are estimated by applying the ordinary least squares (OLS) method.

To estimate the gradient of the center point through Equation (10), we need at least $p$ design points, where $p$ is the dimension of the decision space. To estimate both the gradient and Hessian matrix of the center point through Equation (11), we need at least $p(p+1)/2$ design points. When the problem dimension is large, it may be too costly to estimate both the gradient and Hessian. We may then estimate only the gradient and use the BFGS method to construct the Hessian estimate. This will not affect the convergence of STRONG, because the convergence depends only on the consistency of the gradient estimator. In the rest of this section, we show that the gradient estimator of Equation (10) is strongly consistent under certain conditions.

As described in §3, when the algorithm cannot find a satisfactory solution in stage II, it enters the *inner loop* where the local model will be improved. When the DOE approach is applied in the *inner loop*, each *inner loop* will add some new design points, and all design points from the inner loops at the same iteration will be accumulated. All of these points are used to estimate the gradient of the center point. To ensure the convergence of STRONG, we need the DOE-based gradient estimator to be strongly consistent as the number of design points goes to infinity. Theorem 2 gives a set of sufficient conditions under which the OLS-based gradient estimator is strongly consistent.

Theorem 2 (Lai et al. 1979). *Suppose that* $y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i$ $(i = 1, 2, \ldots)$, *the random variables* $\varepsilon_1, \varepsilon_2, \ldots$ *are independent with* $\mathrm{E}(\varepsilon_i) = 0$ $\forall i$, *and* $\sup_i \mathrm{E}(\varepsilon_i^2) < \infty$. *Let* $X_m$ *denote the design matrix* $\{x_{ij}\}_{1 \le i \le m, 1 \le j \le p}$, $Y_n = (y_1, y_2, \ldots, y_m)^T$ *the response vector, and* $\beta = (\beta_1, \beta_2, \ldots, \beta_p)^T$. *For* $m \ge q$, *the least squares estimate* $(X_m^T X_m)^{-1} X_m^T Y_m \to \beta$ *w.p.*1. *if* $(X_m^T X_m)^{-1} \to 0$ *as* $m \to \infty$.

Remark 1. For the response surface whose variances are nonhomogeneous, WLS (weighted least squares) may be used to replace OLS (Kleijnen 2008, p. 91) to obtain lower variances for the metamodel parameters. Note, however, that using WLS requires estimating the variance-covariance matrix, which may be computationally demanding when the problem is of large scale. Also, because RSM and STRONG fit models only locally, OLS seems better than WLS.

Note that when the number of design points increases in the *inner loop*, eventually we will have enough to fit a quadratic surface. Theorem 2 basically shows that the gradient and Hessian estimators of Equation (11) are strongly consistent as $m \to \infty$ if the underlying response surface is indeed quadratic (so $\mathrm{E}(\varepsilon_i) = 0$, see Theorem 2). Although STRONG allows general nonlinear response surfaces, the underlying response surface inside of a trust region can be very well approximated by a quadratic function when the trust region is sufficiently small. Note that if the number of design points goes to infinity in the *inner loop*, the trust-region size also shrinks to zero. Therefore the sufficient conditions of Theorem 2 are reasonable when applied to analyze the strong consistency of the OLS-based gradient estimators. Furthermore, the strong consistency of the OLS-based gradient estimator can also be extended to uniform convergence under some additional regularity conditions (Andrews 1992).

By Theorem 2, to ensure the strong consistency of the OLS-based gradient estimator, we need the main-effect design matrix $X_m$ to satisfy $(X_m^T X_m)^{-1} \to 0$ as $m \to \infty$. In the following lemma, we show that the requirement is satisfied if we use an orthogonal design to estimate the main effects. All columns in $X_m$ are mutually orthogonal, or equivalently, $X_m^T X_m = mI_{m \times m}$ is a diagonal matrix ($I_{m \times m}$ is a $m \times m$ identity matrix). In main-effect orthogonal designs, the main effects of each variable (i.e., the gradient estimate of each dimension) are being assessed independently of each other.

Lemma 7. *If* $X_m$ *is an orthogonal design matrix, then* $(X_m^T X_m)^{-1} = (1/m) I_{m \times m} \to 0$ *as* $m \to \infty$.

Theorem 2 and Lemma 7 suggest that the main-effect orthogonal designs can be used to achieve strong consistency for the OLS gradient estimator.

To illustrate how to construct the design matrix in the *inner loop*, let $Q_{m_{k_i}}$ be the newly generated main-effect orthogonal design within the current trust region $\Delta_{k_i}$, and the design matrix $X_{m_{k_i}} = [X_{m_{k_{i-1}}}, Q_{m_{k_i}}]^T$ includes all the design points up to the *inner loop* $k_i$. It is clear that if both $X_{m_{k_{i-1}}}$ and $Q_{m_{k_i}}$ are orthogonal, then $X_{m_{k_i}}$ is also orthogonal. The OLS-based gradient estimator is then strongly consistent as $m \to \infty$. With Theorem 2 and Lemma 7, we can also show that Lemma 2 holds for STRONG with black-box gradient estimation by applying the same argument to each dimension of the gradient estimator.

In the numerical experiments reported in §6, we use orthogonal designs (resolution III fractional factorial designs in stage I and central composite designs in stage II and the *inner loop*) to ensure strong consistency of the gradient estimators (Montgomery 2005, Myers et al. 2009). Note that the main-effect designs in central composite designs are orthogonal, thereby fulfilling the requirement. We observed that STRONG works well with these designs. It is also worth mentioning that STRONG can incorporate a wide variety of experimental designs into its framework. A sufficient condition for the convergence to hold is that in the *inner loops*, the main-effect design in the second-order design needs to be orthogonal.

## 6. Numerical Evaluation

In this section, we use several examples to help us understand the properties of the proposed STRONG algorithm and to compare it with other optimization algorithms.

### 6.1. Test Problems

The test problems used in this section are all constructed from deterministic functions with added noise. We adopt this way instead of using real simulation models because (1) the true value of the objective function is analytically available, and we can therefore compare the performance of the algorithms clearly and explicitly; (2) the settings of the test problem can be easily manipulated for investigating the strengths and limitations of the algorithm; and (3) this approach is more time efficient when the amount of numerical evaluations is substantially large (Barton and Ivey 1996).

We consider four types of problems with three dimensionalities, $p = 2, 6, 14$. The three types of problems are selected from literature in nonlinear optimization (More et al. 1981), and the fourth type is a simple quadratic function. Specifically, the first type is the extended Rosenbrock function,

$$g(x) = \sum_{i=1}^{p-1} \left[100(x_i - x_{i+1}^2)^2 + (1 - x_i)^2\right], \tag{12}$$

which is known to be a difficult problem even in deterministic settings. Note that the "extended" Rosenbrock function is different from the standard Rosenbrock function in its functional form and the number of (local) optimal solutions. The standard Rosenbrock function has only one (locally and globally) optimal solution, but the extended Rosenbrock function has more than one local optimal solutions when dimensionality is greater than or equal to four (Shang and Qiu 2006). The second type is the Freudenstein and Roth function:

$$g(x) = \sum_{i=1}^{p/2} \left[-13 + x_{2i-1} + ((5 - x_{2i})x_{2i} - 2)x_{2i}\right]^2 + \left[-29 + x_{2i-1} + ((x_{2i} + 1)x_{2i} - 14)x_{2i}\right]^2, \tag{13}$$

which is also a multimodal function (More et al. 1981). The third and fourth types are both unimodal functions. Specifically, the third type is the Beale function,

$$g(x) = \sum_{i=1}^{3} \left[1.5 - x_{2i-1}(1 - x_{2i})\right]^2 + \left[2.25 - x_{2i-1}(1 - x_{2i}^2)\right]^2 + \left[2.625 - x_{2i-1}(1 - x_{2i}^3)\right]^2, \tag{14}$$

and the fourth type is a quadratic function, i.e.,

$$g(x) = \sum_{i=1}^{p} x_i^2. \tag{15}$$

The stochastic responses of all test problems take the form of $G(x) = g(x) + \varepsilon(x)$, where $\varepsilon(x) \sim$ Normal $(0, \sigma^2(x))$. For each test problem, we consider two scenarios of variance configurations: (1) $\sigma(x) = \sigma$ for all $x \in \mathbb{R}^p$, and (2) $\sigma(x) = 0.1 \cdot g(x)$, where $\sigma(x)$ corresponds to the standard deviation of noise. The first setting assumes homogeneous response surfaces as assumed in the classic RSM; and the second setting gives heterogeneous response surfaces, which are typical in simulation experiments. Notice that in setting (2), when the initial solution is selected far away from the optimum, $g(x)$ is large; thus, the variance of response variable can be very large. The stability of algorithm can be tested in setting (2) when the response surface is grossly noisy. We summarize all 24 scenarios in Table 1.

To evaluate the effect of initial solutions, we select the initial solution in the following two ways: (1) use a fixed initial solution, $20 \cdot \mathbf{1}_p$, where $\mathbf{1}_p = [1, 1, \ldots, 1]^T$ and $p$ is the dimensionality of the problem; and (2) randomly select an initial solution from a $p$-dimensional Cartesian product $[100, -100] \times [100, -100] \times \cdots \times [100, -100]$.

### 6.2. Some Competing Algorithms

To understand the effectiveness of the DOE approach and the trust-region updating technique, we compare the performances of STRONG, SPSA (Spall 2003), NM (Nelder-Mead simplex method),

**Table 1 Twenty-Four Scenarios**

| Scenario | Dimension | Test function | Variance setting |
|---|---|---|---|
| 1 | 2 | Extended Rosenbrock | $\sigma(x) = 50$ |
| 2 | 2 | Extended Rosenbrock | $\sigma(x) = 0.1g(x)$ |
| 3 | 6 | Extended Rosenbrock | $\sigma(x) = 50$ |
| 4 | 6 | Extended Rosenbrock | $\sigma(x) = 0.1g(x)$ |
| 5 | 14 | Extended Rosenbrock | $\sigma(x) = 50$ |
| 6 | 14 | Extended Rosenbrock | $\sigma(x) = 0.1g(x)$ |
| 7 | 2 | Freudenstein and Roth | $\sigma(x) = 50$ |
| 8 | 2 | Freudenstein and Roth | $\sigma(x) = 0.1g(x)$ |
| 9 | 6 | Freudenstein and Roth | $\sigma(x) = 50$ |
| 10 | 6 | Freudenstein and Roth | $\sigma(x) = 0.1g(x)$ |
| 11 | 14 | Freudenstein and Roth | $\sigma(x) = 50$ |
| 12 | 14 | Freudenstein and Roth | $\sigma(x) = 0.1g(x)$ |
| 13 | 2 | Beale | $\sigma(x) = 50$ |
| 14 | 2 | Beale | $\sigma(x) = 0.1g(x)$ |
| 15 | 6 | Beale | $\sigma(x) = 50$ |
| 16 | 6 | Beale | $\sigma(x) = 0.1g(x)$ |
| 17 | 14 | Beale | $\sigma(x) = 50$ |
| 18 | 14 | Beale | $\sigma(x) = 0.1g(x)$ |
| 19 | 2 | Quadratic | $\sigma(x) = 50$ |
| 20 | 2 | Quadratic | $\sigma(x) = 0.1g(x)$ |
| 21 | 6 | Quadratic | $\sigma(x) = 50$ |
| 22 | 6 | Quadratic | $\sigma(x) = 0.1g(x)$ |
| 23 | 14 | Quadratic | $\sigma(x) = 50$ |
| 24 | 14 | Quadratic | $\sigma(x) = 0.1g(x)$ |

and RSM. In particular, for simultaneous perturbaton stochastic approximation (SPSA) we use an existing implementation provided at http://www.jhuapl.edu/SPSA/Pages/MATLAB.htm. NM is originally a direct-search-based heuristic method for deterministic nonlinear optimization but now it can be used to handle stochastic optimization problems when proper modifications are employed. We code it based on the `fminsearch` function in MATLAB (version 7) with modifications suggested in Barton and Ivey (1996). Lastly, the recent version of RSM (Nicolai et al. 2004) is implemented to compare with STRONG.

### 6.3. Parameter Settings

In our numerical study, the parameters are set as Table 2. Among them, $n_0$ and $n_d$ are the most important because they determine how the simulation effort is allocated. We provide a heuristic approach based on signal/noise concept on how to acquire an educated setting. Note that $n_0$ is the initial sample size of the center point. It should allow the RC and SR test to yield reasonable results. We suggest that $n_o$ can be determined based on signal/noise ratio, described as follows, where the signal equals objective value and noise equals standard error:

$$\left| \frac{\mu}{\sigma/\sqrt{n_0}} \right| \geq 2. \tag{16}$$

**Table 2 Parameter Settings for All Test Problems**

| Parameters | $n_0$ | $n_d$ | $\Delta_0$ | $\tilde{\Delta}$ | $\eta_0$ | $\eta_1$ | $\gamma_1$ | $\gamma_2$ | $\alpha_k$ |
|---|---|---|---|---|---|---|---|---|---|
| Settings | $\geq 3$ | $\geq 2$ | 2 | 1.2 | 0.01 | 0.3 | 0.9 | 1.11 | $0.5 \times 0.98^k$ |

Using Equation (16), we can estimate how many initial sample sizes ($n_0$) are needed. Note that in practice, the true $\mu$ and $\sigma$ in Equation (16) are usually unknown, and estimates must be used. We believe that the minimum sample size $n_0$ for the RC and SR test to be effective should be three.

As for $n_d$, it is the number of replications for all design points in black-box gradient estimation. To obtain a reliable local model when the response is noisy, we also suggest using Formula (16) with the replacement of $n_0$ by $n_d$. We believe that for better estimates, two is the minimal sample size of $n_d$. The optimal settings of other parameters, such as $\Delta_0$, $\tilde{\Delta}$, $\eta_0$, $\eta_1$, $\gamma_1$, $\gamma_2$, and $\alpha_k$ are problem dependent and scale dependent. We choose a standard setting as shown in Table 2. In other words, in our numerical study, the settings for $\Delta_0$, $\tilde{\Delta}$, $\eta_0$, $\eta_1$, $\gamma_1$, $\gamma_2$, and $\alpha_k$ are not fine tuned across all 24 scenarios.

### 6.4. Performance Measure

To compare the performances of different algorithms, we use "optimality gap" (OG) defined below to evaluate the performance of algorithm:

$$\frac{g(x_k^*) - g(x^*)}{g(x_0) - g(x^*)},$$

where $x_0$, $x^*$, and $x_k^*$ correspond to the initial solution, the true optima of the underlying response surface and the best solution before the algorithm terminates, respectively. Note that when the test functions have more than one local optimum, $x^*$ is defined as the local optimum that is closest to the solution that the running algorithm produces. We run 20 macroreplications for 24 scenarios for each algorithm (STRONG, SPSA, NM, RSM) with fixed initial solutions and random initial solutions. For each macroreplication, the algorithm is terminated after 4,000 observations are consumed, including the observations that are used to estimate the gradient and Hessian matrix.

### 6.5. Results

The results are summarized in Tables 3 and 4 for fixed and random initial solutions, respectively. Given 20 macroreplications, the average OG is reported, along with the associated standard deviation given in the parentheses. If at least one replication fails to converge (i.e., OG $\geq 1$), the percentage of successful replications is given in parentheses.

In all 24 scenarios where the initial solution is fixed (Table 3), we can see that STRONG is remarkably successful, giving results that are better than other competing algorithms. In particular, we found that SPSA can work well for quadratic functions (unimodal), but it fails to converge in other test problems that are either multimodal, higher dimensional, or of larger variance. The results also coincide with

**Table 3 Optimality Gap for 24 Scenarios with Fixed Initial Solutions**

| | STRONG | SPSA | NM | RSM |
|---|---|---|---|---|
| Scenario | Mean (Std. dev.) | Mean (Std. dev.) | Mean (Std. dev.) | Mean (Std. dev.) |
| 1 | 2.26E–06 (1.50E–06) | (0%) | 2.40E–06 (1.48E–06) | 1.20E–03 (9.76E–04) |
| 2 | 2.36E–06 (1.16E–06) | (10%) | 1.24E–01 (3.06E–01) | 1.37E–03 (9.01E–04) |
| 3 | 8.06E–06 (4.87E–08) | (0%) | 5.09E–06 (4.40E–06) | 3.98E–04 (1.68E–04) |
| 4 | 9.02E–06 (6.81E–06) | (0%) | (10%) | 8.67E–02 (2.65E–01) |
| 5 | 7.84E–06 (2.17E–08) | (0%) | 1.36E–04 (1.89E–04) | 2.22E–04 (1.68E–10) |
| 6 | 5.13E–06 (4.72E–06) | (0%) | (10%) | 3.53E–02 (5.06E–02) |
| 7 | 8.93E–07 (9.54E–08) | (45%) | 1.31E–06 (1.61E–07) | 1.47E–05 (2.37E–05) |
| 8 | 2.45E–06 (1.26E–06) | (45%) | 3.02E–06 (1.67E–06) | 1.03E–05 (2.30E–05) |
| 9 | 3.09E–06 (1.73E–10) | (15%) | 4.81E–06 (4.53E–08) | 1.17E–05 (2.03E–05) |
| 10 | 2.76E–06 (1.29E–06) | (10%) | 9.29E–01 (2.15E–01) | 1.12E–03 (3.81E–03) |
| 11 | 4.23E–06 (1.21E–07) | (0%) | 1.71E–05 (7.15E–07) | 4.40E–06 (3.86E–11) |
| 12 | 2.93E–06 (1.08E–06) | (0%) | (35%) | 4.87E–05 (6.85E–05) |
| 13 | 2.31E–09 (1.82E–10) | (30%) | 4.64E–10 (3.03E–10) | 1.15E–03 (3.70E–03) |
| 14 | 5.82E–09 (3.25E–09) | (20%) | 1.57E–11 (1.40E–12) | 8.64E–02 (2.66E–01) |
| 15 | 3.80E–09 (2.41E–10) | (20%) | 3.08E–10 (3.31E–10) | 3.30E–04 (5.96E–04) |
| 16 | 1.06E–08 (3.61E–09) | (15%) | (40%) | 5.68E–03 (1.35E–02) |
| 17 | 7.55E–09 (6.40E–10) | (0%) | 6.18E–09 (8.85E–09) | 3.27E–05 (4.79E–11) |
| 18 | 1.24E–08 (2.43E–09) | (0%) | (20%) | 1.91E–04 (2.87E–04) |
| 19 | 5.30E–03 (4.02E–03) | 1.27E–02 (9.90E–03) | (50%) | 2.04E–02 (6.98E–02) |
| 20 | 1.16E–06 (1.62E–07) | 6.24E–06 (1.76E–06) | (35%) | 2.39E–02 (8.34E–02) |
| 21 | 3.20E–03 (2.48E–03) | 1.61E–02 (9.40E–03) | (0%) | 5.43E–02 (2.22E–01) |
| 22 | 1.49E–06 (1.44E–06) | 6.60E–06 (4.80E–06) | (0%) | 1.05E–01 (2.82E–01) |
| 23 | 3.17E–03 (1.88E–03) | 1.15E–02 (4.90E–03) | (0%) | 3.13E–03 (1.18E–03) |
| 24 | 4.48E–01 (1.80E–01) | 1.28E–05 (7.67E–06) | (0%) | 3.36E–02 (2.91E–02) |

the finding that the performance of SPSA is quite sensitive to the initial solution, especially in multimodal functions. Specifically, when the initial solution is not selected close to the local optimum, it is quite likely that SPSA will result in poor performance or even diverge. On the other hand, NM shows volatile performance over different scenarios. The results of NM are appalling when the response surface is grossly

**Table 4 Optimality Gap for 24 Scenarios with Random Initial Solutions**

| | STRONG | SPSA | NM | RSM |
|---|---|---|---|---|
| Scenario | Mean (Std. dev.) | Mean (Std. dev.) | Mean (Std. dev.) | Mean (Std. dev.) |
| 1 | 7.72E–05 (2.14E–04) | (85%) | 2.17E–05 (8.82E–05) | (95%) |
| 2 | 4.62E–05 (1.49E–04) | (80%) | (90%) | (90%) |
| 3 | 3.33E–07 (6.60E–07) | 2.09E–02 (4.02E–02) | 4.02E–06 (7.01E–06) | 2.80E–03 (3.29E–03) |
| 4 | 1.07E–01 (1.82E–01) | 7.32E–03 (2.63E–03) | (70%) | 9.25E–02 (2.51E–01) |
| 5 | 1.46E–07 (2.24E–07) | 1.08E–02 (6.24E–03) | 8.00E–04 (1.39E–03) | 4.02E–02 (1.55E–01) |
| 6 | 6.97E–01 (2.17E–01) | 8.20E–03 (3.11E–03) | (35%) | 9.12E–02 (2.25E–01) |
| 7 | 1.89E–08 (1.16E–08) | (90%) | 4.96E–04 (2.00E–03) | (90%) |
| 8 | 2.37E–05 (1.05E–04) | (70%) | 5.57E–04 (2.31E–03) | (90%) |
| 9 | 1.19E–07 (2.31E–07) | 6.70E–03 (1.87E–02) | 5.33E–07 (1.47E–06) | 1.38E–03 (4.16E–03) |
| 10 | 2.77E–08 (2.60E–08) | 1.76E–02 (4.26E–02) | 2.92E–01 (3.74E–01) | 5.54E–02 (2.13E–01) |
| 11 | 2.36E–08 (3.20E–08) | 6.25E–04 (4.60E–04) | 1.18E–07 (1.54E–07) | 2.97E–03 (4.76E–03) |
| 12 | 1.87E–08 (1.74E–08) | 9.91E–04 (1.29E–03) | (85%) | 3.81E–02 (9.60E–02) |
| 13 | 2.43E–05 (1.08E–04) | (85%) | 9.26E–08 (3.48E–07) | 1.66E–03 (6.97E–03) |
| 14 | 7.40E–08 (2.25E–07) | (80%) | 1.04E–06 (4.66E–06) | 0.0308504(1.38E–01) |
| 15 | 1.27E–04 (3.39E–04) | 4.43E–03 (1.53E–02) | 1.74E–12 (3.79E–12) | 9.41E–04 (2.78E–03) |
| 16 | 3.21E–04 (1.28E–03) | 5.28E–04 (1.05E–03) | 2.80E–01 (3.77E–01) | 7.85E–03 (3.47E–02) |
| 17 | 5.60E–05 (1.30E–04) | 1.49E–04 (2.71E–04) | 9.15E–10 (3.94E–09) | 2.21E–03 (4.82E–03) |
| 18 | 9.76E–05 (1.80E–04) | 2.78E–04 (5.00E–04) | (85%) | 1.31E–02 (2.29E–02) |
| 19 | 1.74E–03 (1.48E–03) | 3.74E–03 (6.59E–03) | 3.03E–02 (2.27E–01) | 1.59E–02 (5.23E–02) |
| 20 | 7.08E–04 (2.68E–03) | 1.38E–06 (2.39E–06) | (50%) | 6.99E–02 (2.04E–02) |
| 21 | 6.93E–04 (6.62E–04) | 1.57E–03 (1.50E–03) | 6.70E–02 (7.69E–02) | 5.21E–04 (1.95E–03) |
| 22 | 1.02E–02 (4.34E–02) | 8.44E–07 (5.39E–07) | (45%) | 2.34E–01 (3.65E–01) |
| 23 | 4.19E–04 (2.85E–04) | 1.62E–03 (8.19E–04) | 1.59E–01 (9.35E–02) | 1.24E–04 (4.38E–04) |
| 24 | 3.45E–01 (2.92E–01) | 3.47E–06 (5.70E–06) | (10%) | 2.04E–01 (3.17E–01) |

noisy, such as scenarios 2, 4, 6, 12, and 16. The reason may be attributed to the moving direction of NM being based on the rank of the stochastic responses instead of on the function values themselves (Spall 2003); therefore when the response surface is very noisy, the random error can corrupt the relative ranks of solutions, leading the algorithm to an incorrect direction and finally to failure of convergence. RSM has robust performance over all scenarios. In most scenarios, however, it is inferior to STRONG. We believe that the trust region (region of interest) featured in STRONG allows the algorithm to take adaptive steps, either more aggressive or more conservative, in the search process. Moreover, the iterative selection between first-order and second-order polynomials also grants the algorithm more flexibility to perform the optimization work.

Similar results are exhibited in Table 4, where 24 scenarios are evaluated with random initial solutions in 20 macroreplications. The standard deviation of performance measures in this table, regardless of algorithms and scenarios, is much larger than that in Table 3 because of the random initial solutions. We can see that the location of the initial solution can seriously affect the performance of SPSA and RSM and lead to large variation of performances. Again, NM performs poorly in large variance situations, exhibited in scenarios 2, 4, 6, 12 ,14 18, 20, 22, and 24.

In both Tables 3 and 4, STRONG is more stable and has smaller OGs versus other algorithms. Moreover, the computational advantage of STRONG is more obvious when the dimension of the problem increases. This demonstrates the usefulness of the DOE technique in improving algorithm efficiency, especially in higher-dimensional problems. In other numerical experiments we found that the STRONG framework with random sample designs instead of orthogonal designs had poorer performance. Because of space limitations, those results were not included here.

## 7. Conclusions

In this paper, we proposed STRONG, a new response-surface-based method for unconstrained simulation optimization with continuous decision variables. STRONG uses a series of local metamodel, either a first-order or second-order polynomial, to approximate the underlying complex response surface; therefore it can handle problems in which the response surface is very complex. Compared with traditional RSM, STRONG applies the trust-region updating technique to achieve the desired automation and convergence properties. The framework permits the utilization of the existing gradient information, or a wide variety of experimental designs to obtain black-box gradient estimates. Our numerical evaluations indicate the advantage of STRONG compared with other competing algorithms. Moreover, the design of experiments is found to significantly improve the efficiency of the optimization procedure.

It is noteworthy that if the objective function in Equation (1) is a quantile instead of an expected value, by Serfling (1980) and Hong (2009) there exist strongly consistent estimators of quantile and quantile gradient. If Assumption 2 is satisfied, which means that both the quantile and quantile gradient estimators satisfy the uniform strong law of large numbers, we believe that the gradient-based STRONG still converges (i.e., the conclusion of Theorem 1 holds). However, gradient estimators are typically biased (see, for instance, Serfling 1980). Therefore, the DOE-based STRONG may fail to converge because the unbiasedness of quantile estimators is necessary for Theorem 2 (Lai et al. 1979) to hold.

The limitations of STRONG can be summarized as follows. First, STRONG is an RSM-based algorithm; therefore it is scale dependent, as is classic RSM. Second, like RSM, STRONG provides a local rather than global optimum. Third, we currently use central composite designs (CCD) to construct the second-order polynomials, which can be computationally demanding for large-scale problems. Alternatives to CCD are discussed in Kleijnen (2008). In the future, we will explore the way of constructing the trust region and the appropriate experimental designs to further improve the efficiency of the STRONG framework. Moreover, the use of variance reduction techniques such as common random number (Law 2007), and generalized least squares (Myers et al. 2009) will also be investigated to improve the statistical accuracy of the gradient estimate. It is also of interest to compare the efficiency of STRONG and the traditional RSM in practical applications.

### Electronic Companion

An electronic companion to this paper is available as part of the online version at http://dx.doi.org/10.1287/ijoc.1120.0498.

### Acknowledgments

The authors thank two anonymous referees and the associate editor for their insightful comments and suggestions that have significantly improved this paper. In particular, one referee suggested the use of signal/noise ratio heuristic for determining the appropriate sample size, as discussed in §6.3. The preliminary work was reported in Chang et al. (2007) and Chang (2008). This research was partially supported by the National Science Council in Taiwan [NSC98-2218-E-007-015]; Purdue Research Foundation; Naval Postgraduate School Award [N00244-07-1-0018 and N00244-06-C-0002]; and the Hong Kong Research Grants Council [GRF 613410 and N_HKUST 626/10].

## References


Andradóttir S (2007) Simulation optimization. Banks J, ed. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice* (John Wiley & Sons, Hoboken, NJ).

Andrews DWK (1992) Generic uniform convergence. *Econometric Theory* 8:241–257.

Angün E, Kleijnen J, Hertog DD, Gurkan G (2009) Response surface methodology with stochastic constraints for expensive simulation. *J. Oper. Res. Soc.* 60(6):735–746.

Banks J, ed. (1998) *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice* (John Wiley & Sons, Hoboken, NJ).

Barton RR, Ivey JS (1996) Nelder-Mead simplex modifications for simulation optimization. *Management Sci.* 42(7):954–973.

Barton RR, Meckesheimer M (2006) Metamodel-based simulation optimization. Henderson SG, Nelson BL, eds. *Handbooks in Operations Research and Management Science*, Vol. 13 (Elsevier, Amsterdam), 535–574.

Bartz-Beielstein T, Chiarandini M, Paquete L, Preuss M (2010) *Experimental Methods for the Analysis of Optimization Algorithms* Springer-Verlag, Berlin, Heidelberg.

Bastin F, Cirillo C, Toint PL (2006) An adaptive Monte Carlo algorithm for computing mixed logit estimators. *Comput. Management Sci.* 3:55–79.

Biles WE, Swain JJ (1979) *Optimization and Industrial Experimentation* (Wiley-Interscience, New York).

Box GEP, Wilson KB (1951) On the experimental attainment of optimum conditions. *J. Royal Statist. Soc.* 13:1–38.

Cao XR (1994) *Realization Probabilities: The Dynamics of Queuing Systems* (Springer-Verlag, New York).

Chang K-H (2008) Stochastic trust-region response-surface method (STRONG)—A new response-surface-based algorithm in simulation optimization. Ph.D. thesis, Purdue University, West Lafayette, IN.

Chang K-H, Hong LJ, Wan H (2007) Stochastic trust region gradient-free method (STRONG)—A new response-surface-based algorithm in simulation optimization. Henderson SG, Biller B, Hsieh M-H, Shortle J, Tew JD, Barton RR, eds. *Proc. 2007 Winter Simulation Conf.* (IEEE, Piscataway, NJ), 346–354.

Conn AR, Gould NLM, Toint PL (2000) *Trust-Region Methods* (SIAM, Philadelphia).

Deng G, Ferris MC (2009) Variable-number sample-path optimization. *Math. Programming* 117:81–109.

Fu MC (2002) Optimization for simulation: Theory vs. practice. *INFORMS J. Comput.* 14:192–215.

Fu MC (2006) Gradient estimation. Henderson SG, Nelson BL, eds. *Handbooks in Operations Research and Management Science*, Vol. 13 (Elsevier, Amsterdam), 575–616.

Glynn PW (1990) Likelihood ratio gradient estimation for stochastic systems. *Comm. ACM* 35(10):75–84.

Ho YC, Cao XR (1991) *Discrete Event Dynamic Systems and Perturbation Analysis* (Kluwer Academic Publishers, Boston).

Hong LJ (2009) Estimating quantile sensitivities. *Oper. Res.* 57(1):118–130.

Hood SJ, Welch PD (1993) Response surface methodology and its applications in simulation. Evans, GW, Mollaghasemi, M, Russel, EC, Biles, WE, *Proc. 1993 Winter Simulation Conf.* (IEEE, Piscataway, NJ), 115–122.

Khuri AI, Cornell JA (1996) *Response Surfaces Designs and Analyses* (Marcel Dekker, New York).

Kiefer J, Wolfowitz J (1952) Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.* 23(3):462–466.

Kleijnen JPC (1998) Experimental design for sensitivity analysis, optimization, and validation of simulation models. Banks J, ed. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice* (John Wiley & Sons, New York).

Kleijnen JPC (2008) *Design and Analysis of Simulation Experiments* (Springer, New York).

Kleijnen JPC, Hertog DD, Angün E (2004) Response surface methodology's steepest ascent and step size revisited. *Eur. J. Oper. Res.* 159(1):121–131.

Kleijnen JPC, Sanches SM, Lucas TW, Cioppa TM (2005) A user's guide to the brave new world of designing simulation experiments. *INFORMS J. Comput.* 17(3):263–289.

Kushner HJ, Yin GG (1997) *Stochastic Approximation Algorithms and Applications* (Springer-Verlag, New York).

Lai TL, Robbins H, Wei CZ (1979) Strong consistency of least squares estimates in multiple regression. *J. Multivariate Anal.* 9:343–362.

Law AM (2007) *Simulation Modeling and Analysis*, 4th ed. (McGraw-Hill, Boston).

Ljung L, Pflug G, Walk H (1992) *Stochastic Approximation and Optimization of Random Systems* (Birkhäuser, Basel, Berlin).

Montgomery DC (2005) *Design and Analysis of Experiments*, 6th ed. (John Wiley & Sons, New York).

More JJ, Garbow BS, Hillstrom KE (1981) Testing unconstrained optimization software. *ACM Trans. Math. Software* 7(1):17–41.

Myers RH, Montgomery DC, Anderson-Cook CM (2009) *Response Surface Methodology-Process and Product Optimization Using Designed Experiments* (John Wiley & Sons, New York).

Neddermeijer HG, Oortmarssen GJV, Piersma N, Dekker R (2000) A framework for response surface methodology for simulation optimization. Joines JA, Barton RR, Kang K, Fishwick PA, eds. *Proc. 2000 Winter Simulation Conf.* (IEEE, Piscataway, NJ), 129–136.

Nicolai RP, Dekker R, Piersma N, Oortmarssen GJV (2004) Automated response surface methodology for stochastic optimization models with unknown variance. Ingalls RG, Rossetti MD, Smith JS, Peters BA, eds. *Proc. 2004 Winter Simulation Conf.* 491–499.

Nocedal J, Wright SJ (1999) *Numerical Optimization* (Springer, New York).

Rubinstein RY, Shapiro A (1993) *Discrete Event Systems: Sensitivity Analysis and Stochastic Approximation Using the Score Function Method* (John Wiley & Sons, New York).

Sakalauskas L, Krarup J (2006) Editorial; heuristic and stochastic methods in optimization. *Eur. J. Oper. Res.* 171(3):723–724.

Sanchez SM (2008) Better than a petaflop: The power of efficient experimental design. Mason SJ, Hill RR, Monch L, Rose O, Jefferson T, Fowler JW, eds. *Proc. 2008 Winter Simulation Conf.* (IEEE, Piscataway, NJ), 73–84.

Serfling RJ (1980) *Approximation Theorems of Mathematical Statistics* (John Wiley & Sons, New York).

Shang Y-W, Qiu Y-H (2006) A note on the extended Rosenbrock function. *Evolutionary Comput.* 14(1):119–126.

Spall JC (2000) Adaptive stochastic approximation by the simultaneous pertubation method. *IEEE Trans. Automatic Control* 45(10):1839–1853.

Spall JC (2003) *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control* (John Wiley & Sons, New York).

Tekin E, Sabuncuoglu I (2004) Simulation optimization: A comprehensive review on theory and applications. *IIE Trans.* 36(11): 1067–1081.

Wu CFJ, Hamada M (2000) *Experiments: Planning, Analysis, and Parameter Design Optimization* (John Wiley & Sons, New York).

Yin G, Zhu Y (1992) Averaging procedures in adaptive filtering: An efficient approach. *IEEE Trans. Automatic Control* 37(4): 466–475.