Available online at www.sciencedirect.com
ScienceDirect
Simulation Modelling Practice and Theory 16 (2008) 50–64
SIMULATION MODELLING PRACTICE AND THEORY
www.elsevier.com/locate/simpat

# Response surface methodology for constrained simulation optimization: An overview

Jack P.C. Kleijnen *

*Department of Information Systems/Center for Economic Research, Tilburg University, Postbox 90153, 5000 LE Tilburg, The Netherlands*

Received 28 June 2007; received in revised form 25 September 2007; accepted 7 October 2007
Available online 18 October 2007

**Abstract**

This article summarizes 'generalized response surface methodology' (GRSM), extending Box and Wilson's 'response surface methodology' (RSM). GRSM allows multiple random responses, selecting one response as goal and the other responses as constrained variables. Both GRSM and RSM estimate local gradients to search for the optimum. These gradients are based on local first-order polynomial approximations. GRSM combines these gradients with Mathematical Programming findings to estimate a better search direction than the steepest ascent direction used by RSM. Moreover, these gradients are used in a bootstrap procedure for testing whether the estimated solution is indeed optimal. The focus of this paper is the optimization of simulated (not real) systems.


*Keywords:* RSM; Mathematical Programming; Bootstrap

## 1. Introduction

The importance of *optimizing* engineered systems (man-made artefacts) is emphasized in the 2006 NSF panel reported in Oden [39]. That report also emphasizes the crucial role of *simulation* in engineering science. An essential characteristic of simulation – compared with Mathematical Programming – is that in simulation the objective function (which is the function to be minimized or maximized) is not known explicitly; actually, this function is defined by the complicated simulation model (computer code, computer program). The simplest optimization problem has no constraints on the inputs or outputs of the system (either real or simulated). The simulation model may be either deterministic or random (stochastic, discrete event). In this article, however, I will focus on random simulation, which (by definition) uses pseudo-random numbers (PRNs) — the PRNs (say) $\mathbf{r}$ are independently and uniformly distributed on the interval $[0, 1]$. Nevertheless, I expect that the methodology for optimization of random simulation models — known

* Tel.: +31 13 466 2029; fax: +31 13 466 3069.
*E-mail address:* kleijnen@uvt.nl
*URL:* http://center.uvt.nl/staff/kleijnen/

1569-190X/$ - see front matter 
doi:10.1016/j.simpat.2007.10.001

as *simulation optimization* — is also relevant for optimization of deterministic simulation models and real-world systems.

The *simplest optimization* concerns the minimization of the expected value of a single (univariate) simulation output — with no input or output constraints; the inputs (decision variables, factors) are continuous variables. An academic example is an $(s, Q)$ inventory-management simulation aimed at minimization of the expected value of the total inventory costs (the sum of inventory carrying, reorder, and out-of-stock costs), and the decision variables are the reorder level $s$ and the order quantity $Q$. (Implicit input constraints are that these two decision variables must be non-negative.) More complicated examples are inventory-production simulation models in logistics and operations management. Furthermore, telecommunications often uses queueing simulation models that try to minimize a given deterministic function of either the expected customer delay or the server blocking probability (such a probability may be formulated as the expected value of a binary variable using the indicator function).

In *practice*, however, simulation models have *multiple outputs*. Examples are many practical inventory models that require the inventory system to satisfy a minimum service rate (or fill rate), because the out-of-stock costs are hard to quantify (an example is [21]). Simulation software facilitates the collection of multiple outputs. I shall formalize this type of problems in Section 3.

There are *many methods* for simulation optimization. Recent discussions of simulation optimization are Bates et al. [7], Fu et al. [16], Henderson and Nelson [19], Kenett and Steinberg [24], Kleijnen [30], and Tekin and Sabuncuoglu [48]; also see the recent proceedings of the yearly Winter Simulation Conferences.

Many of these methods treat the simulation model as a 'black box'; i.e., they observe only the inputs and outputs of the simulation model. In this article, I focus on *response surface methodology* (RSM). RSM is a method that is often applied in real-life experiments; see Section 2 and also the software called 'Design-Expert' (see www.statease.com). (Some authors outside random simulation speak of RSM, but mean 'what-if' analysis — not sequential or iterative optimization; see, e.g., Downing et al. [14], Olivi [40] and Rikards and Auzins [42].) Google gave more than two million hits for the term 'Response Surface Methodology', on September 19, 2007. Unfortunately, RSM (unlike some other search heuristics) has not yet been implemented as an add-on to any of the commercial simulation software packages.

*Generalized response surface methodology* (GRSM) is a novel general-purpose heuristic based on Box and Wilson's [11] classic RSM. Both GRSM and RSM estimate local gradients to search for the optimum. These gradient estimates use locally fitted first-order polynomials (alternative estimators of gradients outside RSM are discussed in Boente and Rodriguez [10], Fu [15] and Tekin and Sabuncuoglu [48]). GRSM, however, combines these gradients with Mathematical Programming findings to estimate a better search direction than the steepest ascent direction used by RSM. Like Mathematical Programming, GRSM allows multiple responses — selecting one response as goal and the other responses as constrained variables. The (deterministic) input variables may also be subjected to constraints. Finally, GRSM uses the estimated gradients in a bootstrap procedure for testing whether the estimated solution is indeed optimal.

In this article, I focus on *expensive* simulation, which (by definition) takes much computer time to compute a single realization of the time path of the simulated system. For example, 36–160 hours of computer time were needed to simulate a crash model at Ford Motor Company; see the panel discussion on optimization in Simpson et al. [45]. This panel also reports the example of a (so-called 'cooling') problem with 12 inputs, 10 constraints, and 1 objective function. For such expensive simulations, many simulation optimization methods are unpractical; e.g., OptQuest (an add-on to several software products for random simulation; e.g., Arena, ProModel, and Simul8) requires relatively many simulation replicates and input combinations (points, scenarios); see the inventory simulation in Kleijnen and Wan [33]. The mathematical and statistical computations by GRSM itself are negligible — compared with the computer time required by the 'expensive' simulation runs.

In this article, I summarize previous work that I did on classic RSM, and recent work on GRSM. That recent work is scattered over several papers, referenced in the following sections. More details are given in Kleijnen [30].

The remainder of this article is organized as follows. Section 2 summarizes classic RSM, and the adapted steepest ascent (ASA) search direction developed by Kleijnen et al. [31]. Section 3 summarizes GRSM for simulation with multiple responses, developed by Angün et al. [2]. Section 4 summarizes a procedure for testing whether an estimated optimum is truly optimal — using the Karush–Kuhn–Tucker (KKT) conditions —

developed by Bettonvil et al. [9]. Section 5 presents conclusions. Nearly 50 references enable the readers to study further details. The article contains several paragraphs starting with 'Note:', which upon first reading may be skipped.

## 2. Classic RSM: single output and no constraints

Box and Wilson [11] searches for the combination of continuous inputs that minimizes the single output of a *real-world* system (or maximizes that output; i.e., simply add a minus sign to the output). Numerous applications are given in Myers et al.'s [36] excellent survey of the period 1966–1988. Recent textbooks are Khuri and Cornell [26] and Myers and Montgomery [35]. Recent software is Stat-Ease's 'Design-Ease' and 'Design-Expert'.

This RSM has also been applied to *simulation*; see Bartz-Beielstein [5], Irizarry et al. [20], Kleijnen [28], Rosen et al. [43], and Yang and Tseng [49]. A case study of RSM for deterministic simulation is Ben-Gal and Bukchin [8]. RSM is also discussed in Law [34], which is a classic discrete-event simulation textbook, and in survey articles such as Barton and Meckesheimer [4] and Tekin and Sabuncuoglu [48].

### 2.1. RSM outline

As I have already stated in Section 1, the simplest optimization problem concerns the minimization of the expected value of a single simulation output:

$$\min_{\mathbf{z}} E(w_0|\mathbf{z}) \tag{1}$$

where $E(w_0|\mathbf{z})$ is the goal (or objective) output of the random simulation model, which is to be minimized through the choice of the simulation input $\mathbf{z}$; $\mathbf{z} = (z_1, \ldots, z_k)'$ where $z_j$ $(j = 1, \ldots, k)$ denotes the $j$th original (non-coded, non-standardized) input of the simulation program.

Classic RSM (applied to real-world or simulated systems) has the following *characteristics*.

- RSM is an *optimization heuristic* that tries to estimate the input combination that minimizes a given goal function; see (1) above. Because RSM is a heuristic, success is not guaranteed (see below).
- RSM is a *stepwise* (multi-stage) method.
- In these steps, RSM uses local first-order and second-order *polynomial* regression (meta)models (response surfaces). RSM assumes that these models have *white noise*; i.e., the regression residuals $e$ are normally, independently, and identically distributed (NIID) with zero mean and constant variance in the local experimental area: $e \sim \text{NIID}(0, \sigma^2)$ (when moving to a new local area in a next step, the variance may change). Consequently, ordinary least squares (OLS) gives the best linear unbiased estimator (BLUE); see Kleijnen [29].
- To fit (estimate, calibrate) these first-order polynomials, RSM uses *classic designs* of resolution III; for second-order polynomials, RSM usually applies a central composite design (CCD); details on these designs are given in Kleijnen [30] and Myers and Montgomery [35].
- To determine in which direction the inputs will be changed in a next step, RSM uses the *gradient* that is implied by the first-order polynomial fitted in the current step. This gradient is used in the mathematical (not statistical) technique of *steepest descent* (or steepest ascent, in case the output is to be maximized, not minimized; details follow below).
- In the final step, RSM takes the *derivatives* of the (locally fitted) second-order polynomial to estimate the optimum input combination. RSM may also apply the mathematical technique of *canonical analysis* to this polynomial, to examine the shape of the optimal (sub)region: does that region have a unique minimum, a saddle point, or a ridge (stationary points)?

More specifically, classic RSM consists of the following eight *steps* (also see Fig. 1, which gives an example with a single goal function and two constrained random outputs; these constraints vanish in classic RSM).

(1) The analysts begin by selecting a *starting point*. They may select the input combination currently used in practice if the simulated system already exists. Otherwise, they should use intuition and prior knowledge (as in many other heuristics).
(2) For the *neighborhood* of this starting point, the analysts explore the input/output (I/O) behavior of the simulated system. This behavior is approximated through a local first-order polynomial (as the Taylor series expansion suggests). Hence the analysts need to estimate the intercept $\beta_0$ and the $k$ main (first-order) effects $\beta_j$ ($j = 1, \ldots, k$). Therefore they use a resolution-III design. Unfortunately, there are no general guidelines to determine the appropriate *size* of this local area; again, intuition and prior knowledge are important. (Finite differencing — which replaces the resolution-III design by a less efficient one-factor-at-a-time design — also faces the problem of selecting an appropriate size for the local area; the optimal size depends on the unknown variance and second-order derivatives; see Brekelmans et al. [12] and Zazanis and Suri [50].)
(3) To decide on the next subarea to be explored by simulation, the analysts follow the *steepest descent path*, which uses the local gradient. For example, if the estimated first-order effects are such that $\hat{\beta}_1 \gg \hat{\beta}_2 > 0$, then they obviously decrease input $z_1$ much more than input $z_2$. Unfortunately, the steepest descent method is *scale dependent*; i.e., linear transformations of the inputs affect the search direction (see Myers and Montgomery [35, pp. 218–220]). Fortunately, Kleijnen et al. [31,32] developed a scale-independent variant, which I shall present at the end of this section.
(4) Unfortunately, the steepest descent technique does not quantify the *step size* along its path. The analysts may therefore try some intuitively selected value for the step size. If that value yields an inferior simulation output (significantly higher instead of lower output), then they may reduce the step size. Otherwise, they take one more step in the current steepest descent direction. (There are more sophisticated mathematical procedures for selecting step sizes; see Safizadeh and Signorile [44], and Section 3.)
(5) After a number of steps along the steepest descent path, the simulation output will *deteriorate* (increase instead of decrease), because the first-order polynomial is only a local approximation of the implicit I/O function defined by the simulation model itself. When this deterioration happens, the analysts simulate the $n > k$ factor combinations specified by a *resolution-III design* centered around the best point found so far. (The analysts may use the same coded design as in Step 2, but translate that design into different values for the original variables; the best combination found so far, may be one of the corner points of the design; see Fig. 1.) Next the analysts estimate the first-order effects in the new local polynomial approximation. And so their search continues.
(6) However, it is intuitively clear that a *plane* (implied by the most recent local first-order polynomial) cannot adequately represent a *hill top* (when searching for the maximum; the analogue holds for the minimum). So in the neighborhood of the optimum, a first-order polynomial shows serious *lack-of-fit*. A popular and simple diagnostic measure is the coefficient of determination $R^2$. A related diagnostic tests whether all estimated first-order effects (and hence the gradient) are zero. Instead of these diagnostics, the analysts might use 'cross-validation', which is explained in Kleijnen [30] including more references. If the most recently fitted first-order polynomial turns out to be inadequate, then the analysts fit a *second-order polynomial*. To estimate this metamodel, they simulate the combinations specified by a CCD.
(7) From this second-order polynomial, the analysts estimate the optimal values of the decision variables by straightforward *differentiation* or by more sophisticated *canonical analysis*; see Myers and Montgomery [35, p. 208].
(8) If time permits, then the analysts may try to escape from a possible local minimum and *restart* their search from a different initial local area — which brings them back to Step 1; also see Nicolai and Dekker [38].

### 2.2. Adapted steepest descent

In Step 3, I mentioned a variant of steepest descent. This so-called *adapted steepest descent* (ASD) accounts for the covariances between the elements of the estimated gradient $\hat{\beta}_{-0} = (\hat{\beta}_1, \ldots, \hat{\beta}_k)'$, where the subscript $-0$ means that the intercept $\hat{\beta}_0$ of the estimated first-order polynomial vanishes in the estimated gradient; i.e., $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_{-0})'$.

This $\mathbf{cov}(\hat{\beta}_{-0})$, which follows from the (classic) white noise assumption is

$$\mathbf{cov}(\hat{\beta}) = \sigma_w^2(\mathbf{Z}'\mathbf{Z})^{-1} = \sigma_w^2 \begin{pmatrix} a & \mathbf{b}' \\ \mathbf{b} & \mathbf{C} \end{pmatrix} \tag{2}$$

where

$\sigma_w^2$ denotes the variance of the (goal) simulation output $w$
$\mathbf{Z}$ is the $N \times (1+k)$ matrix of explanatory regression variables including the column with $N$ one's
$N = \sum_{i=1}^{n} m_i$ is the total number of simulation runs
$n$ is the number of different simulated input combinations
$m_i$ is the number of identically and independently distributed (IID) replicates for combination $i$
$a$ is a scalar
$\mathbf{b}$ is a $k$-dimensional vector
$\mathbf{C}$ is a $k \times k$ matrix such that $\mathbf{cov}(\hat{\beta}_{-0}) = \sigma_w^2\mathbf{C}$.

*Note*: $\mathbf{Z}$'s first column corresponds with the intercept $\beta_0$. Furthermore, $\mathbf{Z}$ is determined by a resolution-III design (so $n > k$), transformed into the original values of the inputs in the local area. To save computer time, only the center of the local area may be replicated (the center is added to the resolution-III design). By definition, the $m_i$ replicates use the same input combination $\mathbf{z}_i$ $(i = 1, \ldots, n)$.

The variance $\sigma_w^2$ in (2) may be estimated through the mean squared residuals (MSR):

$$\hat{\sigma}_w^2 = \frac{\sum_{i=1}^{n}\sum_{r=1}^{m_i}(w_{i;r} - \hat{y}_i)^2}{N - (k+1)} \tag{3}$$

where $\hat{y}_i = \mathbf{z}_i'\hat{\beta}$; also see Myers and Montgomery [35].

It is easy to prove that the predictor variance $\mathrm{var}(\hat{y}|\mathbf{z})$ increases as $\mathbf{z}$ (point to be predicted) moves further away from the local area where the gradient is estimated. The point with the minimum predictor variance is the point $-\mathbf{C}^{-1}\mathbf{b}$ (with $\mathbf{C}$ and $\mathbf{b}$ defined below Eq. (2)).

The new point to be simulated is

$$\mathbf{d} = -\mathbf{C}^{-1}\mathbf{b} - \lambda\mathbf{C}^{-1}\hat{\beta}_{-0} \tag{4}$$

where

$-\mathbf{C}^{-1}\mathbf{b}$ is the point where the local search starts (point with minimum variance locally)
$\lambda$ is the step size
$\mathbf{C}^{-1}\hat{\beta}_{-0}$ is the (classic) steepest descent direction (namely, $\hat{\beta}_{-0}$) adapted for $\mathbf{cov}(\hat{\beta}_{-0})$

Accounting for $\mathbf{cov}(\hat{\beta}_{-0})$ gives a scale-independent search direction, which is an important characteristic for both practitioners and researchers. This search direction in general performs better than steepest descent; experimental results are given in Kleijnen et al. [31,32].

## 3. GRSM: multiple outputs and constraints

In practice, simulation models have *multiple* responses (multivariate output). Several approaches to solve the resulting issues are surveyed by Rosen et al. [43]. Furthermore, the RSM literature also offers several approaches for such situations; see the surveys in Angün [1], Khuri [25], Ng et al. [37], and Osborne et al. [41]. However, I find these approaches less attractive than the following approach, which I call GRSM.

### 3.1. GRSM outline

GRSM assumes that one simulation output should be minimized, while all the other outputs must satisfy given constraints. GRSM has the following characteristics:

- GRSM generalizes the steepest descent search direction (applied in classic RSM) through the 'affine scaling search direction' and borrowing ideas from *interior point* methods (a variation on Karmarkar's algorithm) in Mathematical Programming; see Barnes [3]. This search direction moves faster to the optimum than steepest descent, since the GRSM search avoids creeping along the boundary of the feasible area (this feasible area is determined by the constraints on the random outputs and the deterministic inputs; see below). Moreover, this search tries to stay inside the feasible area, so the simulation program does not crash. Finally, this search direction is scale independent; see Angün et al. [2].
- GRSM uses its search direction *iteratively* (as classic RSM does). Because this heuristic is developed for expensive simulation experiments, the search should quickly reach a neighborhood of the true optimum.
- Though GRSM has been developed for *random* simulations, it can be easily adapted for *deterministic* simulations and *real-world* (non-simulated) systems (analogous to classic RSM).

Formally, GRSM extends the classic RSM problem formulated in (1) to the following *constrained non-linear random optimization problem*:

$$\min_{\mathbf{z}} E(w_0|\mathbf{z}) \tag{5}$$

such that the other (say) $(r-1)$ random outputs satisfy the constraints

$$E(w_{h'}|\mathbf{z}) \geqslant a_{h'} \quad \text{with } h' = 1,\ldots,r-1 \tag{6}$$

and the $k$ deterministic inputs satisfy the *box constraints*

$$l_j \leqslant z_j \leqslant u_j \quad \text{with } j = 1,\ldots,k. \tag{7}$$

An example is the following inventory simulation. The sum of the expected inventory carrying costs and ordering costs should be minimized. The expected service percentage should be at least (say) 90%, so $a_1 = 0.9$ in (6). Both the reorder quantity $z_1(= Q)$ and the reorder level $z_2(= s)$ should be non-negative, so $z_1 \geqslant 0$ and $z_2 \geqslant 0$; see (7). (Note the similarity of the constraints on the random outputs and the deterministic inputs.) Stricter input constraints may be formulated; e.g., the reorder level should at least cover the expected demand during the expected order lead time. Input constraints more complicated than these box constraints (namely, geometry constraints) are discussed in Stinstra et al. [47]. A simple linear budget constraint for the inputs is used in Dengiz et al. [13]. Input constraints resulting from output constraints are discussed in Ng et al. [37].

Analogously to the first steps of classic RSM, GRSM locally approximates the multivariate I/O function by $r$ univariate first-order polynomials augmented with white noise

$$\mathbf{y}_h = \mathbf{Z}\beta_h + \mathbf{e}_h \quad \text{with } h = 0,\ldots r-1. \tag{8}$$

Like RSM, GRSM assumes that *locally* the white noise assumption holds for the linear regression metamodels in (8). The following OLS estimators are then the BLUEs:

$$\hat{\beta}_h = (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{w}_h \quad \text{with } h = 0,\ldots,r-1. \tag{9}$$

Then the vector $\hat{\beta}_0$ (OLS estimator for first-order polynomial approximation of goal function) and the goal function (5) result in

$$\min_{\mathbf{z}} \hat{\beta}_{0;-0}\mathbf{z} \tag{10}$$

where $\hat{\beta}_{0;-0}$ denotes the OLS estimate of the local regression parameters for the goal output (which explains the first subscript 0) excluding the intercept (which explains the second subscript $-0$); so, $\hat{\beta}_{0;-0} = (\hat{\beta}_{0;1},\ldots,\hat{\beta}_{0,k})'$ is the *estimated local gradient* of the goal function.

The $(r-1)$ estimates $\hat{\beta}_{h'}$ in (9) combined with the original output constraints (6) give

$$\hat{\beta}'_{h';-0}\mathbf{z} \geqslant c_{h'} \quad \text{with } h' = 1,\ldots,r-1 \tag{11}$$

where $\hat{\beta}_{h';-0} = (\hat{\beta}_{h';1},\ldots,\hat{\beta}_{h',k})'$ denotes the estimated local gradient of constraint function $h'$, and $c_{h'} = a_{h'} - \hat{\beta}_{h';0}$ denotes the modified right-hand side of this constraint function. The box constraints in (7) remain unchanged.

Now the $(r-1)$ $k$-dimensional vectors $\hat{\beta}_{h';-0}$ in (11) are collected in the $(r-1)\times k$ matrix called $\mathbf{B}$. Likewise, the $(r-1)$ elements $c_{h'}$ are collected in the vector $\mathbf{c}$. And the $k$-dimensional vectors with the non-negative *slack variables* $\mathbf{s}$, $\mathbf{r}$, and $\mathbf{v}$ are introduced. Altogether this gives

$$\begin{aligned} &\text{minimize} && \hat{\beta}'_{0;-0}\mathbf{z} \\ &\text{subject to} && \mathbf{B}\mathbf{z}-\mathbf{s}=\mathbf{c} \\ &&& \mathbf{z}+\mathbf{r}=\mathbf{u} \\ &&& \mathbf{z}-\mathbf{v}=\mathbf{l}. \end{aligned} \tag{12}$$

This (local) optimization problem is *linear* in the decision variables $\mathbf{z}$. GRSM does not solve this linear programming (LP) problem, but uses this problem only to derive the following novel *search direction* $\mathbf{d}$:

$$\mathbf{d}=-\left(\mathbf{B}'\mathbf{S}^{-2}\mathbf{B}+\mathbf{R}^{-2}+\mathbf{V}^{-2}\right)^{-1}\hat{\beta}_{0;-0} \tag{13}$$

where $\mathbf{S}$, $\mathbf{R}$, and $\mathbf{V}$ are diagonal matrixes with as main-diagonal elements the current estimated slack vectors $\mathbf{s}$, $\mathbf{r}$, and $\mathbf{v}$ in (12); the last factor in (13) (namely, $\hat{\beta}_{0;-0}$) is the estimated classic steepest ascent direction. (As mentioned above, GRSM's search direction is scale independent.)

*Note*: As the value of a slack variable in (13) decreases (so the corresponding constraint gets tighter), the GRSM search direction deviates more from the steepest descent direction. Possible singularity of the various matrices in (13) is discussed in Angün [1].

Following the search direction (or path) defined by (13), GRSM must decide on the *step size* (say) $\lambda$ along this path. An explicit step size assuming that the local metamodel (11) holds *globally* is

$$\lambda=0.8\min_{h'}\left[\frac{c_{h'}-\hat{\beta}'_{h';-0}\mathbf{z}_c}{\hat{\beta}'_{h';-0}\mathbf{d}}\right] \tag{14}$$

where the factor 0.8 is chosen to decrease the probability that the local metamodel is misleading when applied globally; $\mathbf{z}_c$ denotes the current input combination, so the new combination becomes $\mathbf{z}_c+\lambda\mathbf{d}$. Obviously, the box constraints (7) for the deterministic inputs hold globally, so it is easy to check whether the solution in (14) violates these constraints.

Analogously to classic RSM, GRSM proceeds *stepwise*; i.e., after each step along the search path, the following hypotheses are tested:

(1) $w_0(\mathbf{z}_c+\lambda\mathbf{d})$ (simulation output of new combination) is *no improvement* over $w_0(\mathbf{z}_c)$ (output of old combination); i.e., this step increases the goal output $w_0$ (pessimistic null-hypothesis):

$$H_0: E[w_0(\mathbf{z}_c+\lambda\mathbf{d})] \geqslant E[w_0(\mathbf{z}_c)]. \tag{15}$$

(2) This step is *feasible*; i.e., the new solution satisfies the $(r-1)$ constraints in (6)

$$H_0: E(w_{h'}|\mathbf{z}) \geqslant a_{h'} \quad \text{with } h'=1,\ldots,r-1. \tag{16}$$

To test these hypotheses, I propose the following simple statistical procedures (more complicated parametric bootstrapping is used by Angün [1], which permits non-normality and tests the relative improvement $w_0(\mathbf{z}_c+\lambda\mathbf{d})/w_0(\mathbf{z}_c)$ and the relative slacks $s_{h'}(\mathbf{z}_c+\lambda\mathbf{d})/s_{h'}(\mathbf{z}_c)$).

*Sub 1*: To test (15), the classic Student $t$ statistic may be applied. A paired $t$ statistic may be applied if common random numbers (CRN) are used to obtain the two simulation outputs $w_0(\mathbf{z}_c+\lambda\mathbf{d})$ and $w_0(\mathbf{z}_c)$. To estimate the standard error of their difference, $m\geqslant 2$ replicates suffice because the $t$ statistic has $m-1$ degrees of freedom. The hypothesis is rejected if significant improvement is observed.

*Sub 2*: Again a $t$ statistic with $m-1$ degrees of freedom may be applied. Because $r-1$ hypotheses are implied by (16), Bonferroni's inequality may be used (i.e., divide the classic $\alpha$ value by the number of tests). Recent references on this inequality are given in Gordon [18].

Actually, a better solution may lie in between $\mathbf{z}_c$ (old combination) and $\mathbf{z}_c+\lambda\mathbf{d}$ (new combination at 'maximum' step size). Therefore GRSM uses *binary search*; i.e., it simulates a combination that lies halfway

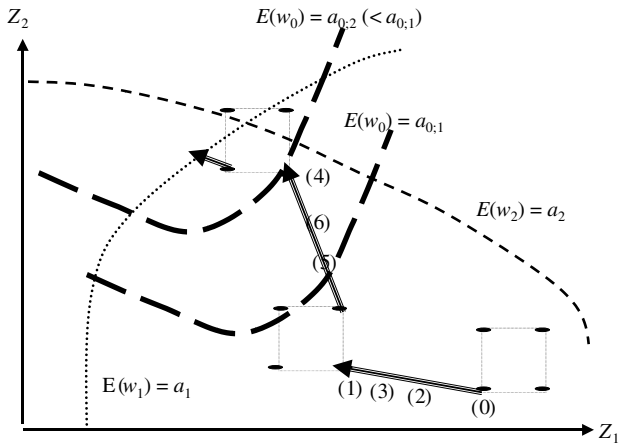


Fig. 1. GRSM illustration.

these two combinations (and is still on the search path). This halving of the step size may be applied a number of times.

Next, GRSM proceeds analogously to classic RSM. So around the best combination found so far, it selects a new local area. Again a resolution-III design selects new simulation input combinations. Only the new center may be replicated $m > 1$ times. And $r$ first-order polynomials are fitted, which gives a *new* search direction, and so on.

### 3.2. GRSM illustration

Fig. 1 illustrates GRSM, and deserves the following comments. There are two inputs; see the two axes labeled $z_1$ and $z_2$. There is one goal function that is to be minimized; the figure shows only two contour plots (iso-costs functions), namely $E(w_0) = a_{0;1}$ and $E(w_0) = a_{0;2}$ with $a_{0;2} < a_{0;1}$. There are two constrained random outputs; see $E(w_1) = a_1$ and $E(w_2) = a_2$. The search starts in the lower-right local area, where a $2^2$ design is executed; see the four elongated points. This design and the (not shown) replicates give a search direction; see the arrow leaving from point (0). The maximum step size along this path takes the search from point (0) to point (1). The binary search takes the search back to point (2), and next to point (3). Because the best point so far turns out to be point (1), the $2^2$ design is simulated at the local area with this point as one of its points. This design gives a new search direction, which avoids the boundary. The maximum step size now takes the search to point (4). The binary search takes the search back to point (5), and next to point (6). Because the best point so far is now point (4), the $2^2$ design is simulated at the local area with this point as one of its points. A new search direction is estimated, etc. (the remaining search is not displayed).

Angün [1] applies GRSM to two examples, namely Bashyam and Fu's [6] inventory simulation with a service-level constraint so the solution is unknown, and a test function with known solution (most test functions in simulation optimization are unconstrained; see, e.g., the seven functions in Nicolai and Dekker [38]). The results of these examples are encouraging, as GRSM found solutions that were both feasible and gave drastically lower goal functions.

## 4. Testing an estimated optimum: KKT conditions

By definition, it is uncertain whether the optimum estimated by a heuristic (such as GRSM) is close enough to the true optimum. In *deterministic* Non-linear Mathematical Programming, the *Karush–Kuhn–Tucker* (KKT) first-order necessary optimality conditions have been derived; see, e.g., Gill et al. [17]. In this section, I first present the basic idea behind the KKT conditions; next, I explain how to test these conditions in random simulation.

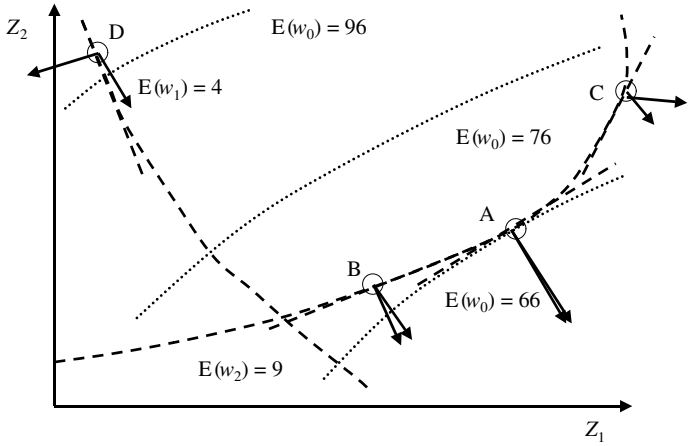


Fig. 2. A constrained non-linear random optimization problem.

### 4.1. The KKT conditions

Fig. 2 illustrates the same type of problem as the one in Fig. 1. In the present example there is again a goal function $E(w_0)$, for which three contour plots are displayed corresponding to the values 66, 76, and 96; also see (5). There are two constrained simulation outputs, namely $E(w_1) \geqslant 4$ and $E(w_2) \geqslant 9$; also see (6). The figure displays the boundaries of the feasible area, which is determined by $E(w_1) = 4$ and $E(w_2) = 9$. The optimum combination is point A. Points B and C lie on the boundary $E(w_2) = 9$; point D lies on the boundary $E(w_1) = 4$. Obviously, point D is far away from the optimum combination A. The figure also displays the (local) gradients at these four points for the goal function and the *binding constraint*; i.e., the constraint with a zero slack value in (6). These gradients are perpendicular to the local tangent lines; those lines are shown only for the binding constraint — not for the goal function.

Let $\mathbf{z}^0$ denote a local minimizer for the deterministic variant of our problem. The KKT conditions for $\mathbf{z}^0$ are then

$$\begin{aligned} &\beta_{0;-0} = \sum_{h \in A(\mathbf{z}^0)} \lambda_h^0 \beta_{h;-0} \\ &\lambda_h^0 \geqslant 0 \\ &h \in A(\mathbf{z}^0) \end{aligned} \tag{17}$$

where $\beta_{0;-0}$ denotes the $k$-dimensional vector with the gradient of the goal function (see (10)); $A(\mathbf{z}^0)$ is the index set with the indices of the constraints that are binding at $\mathbf{z}^0$; $\lambda_h^0$ is the Lagrangian multiplier for binding constraint $h$; $\beta_{h;-0}$ is the gradient of the output in that binding constraint (in Fig. 2, there is only one binding constraint at the points A–D). The KKT conditions imply that the gradient of the objective can be expressed as a non-negative linear combination of the gradients of the binding constraints, at $\mathbf{z}^0$.

Fig. 2 shows that point A satisfies (17); point B has two gradients that point in different but similar directions — and so does point C. Point D, however, has two gradients that point in completely different directions.

*Note*: If the optimum occurs *inside* the feasible area, then there are no binding constraints. The KKT conditions then reduce to the condition that the goal gradient is zero. Classic RSM tests for a zero gradient, estimated from a second-order polynomial; see Step 7 in Section 2. This test may use a classic $F$-test (see Section 2 and Myers and Montgomery [35]). In this article, I do not consider such situations any further.

### 4.2. Testing the KKT conditions

In *random* simulation the gradients must be *estimated*. Moreover, to check which constraints are binding, the slacks of the constraints must be estimated. This estimation turns the KKT conditions (17) into a problem of non-linear statistics.

Angün [1] presents an asymptotic test to check whether the optimum has indeed been found. Bettonvil et al. [9] derives an alternative, bootstrap test. I focus on the latter test, because it is simpler (the former test uses the so-called Delta method and a generalized form of the so-called Wald statistic) and it allows a small number of replicates (as is the case in expensive simulation). Both tests assume a problem like the one formulated in the preceding section; i.e., there is one random simulation output to be minimized and there are $r-1$ constrained random simulation outputs; see (5) and (6).

*Note*: Bettonvil et al. [9] assumes that estimated gradients are indeed available. RSM and GRSM do give estimated gradients; most metaheuristics (e.g., OptQuest) do not estimate gradients. However, when these latter metaheuristics are applied, then — at the presumed end of the search — a local experiment may be performed to estimate the gradients and use these gradients as a stopping criterion — instead of using rather arbitrary criteria such as a prefixed computer budget! Results are given in Kleijnen and Wan [33].

*Note*: Whenever a metaheuristic is used to estimate gradients while treating the simulation model as a *black box*, the analysts better not change one factor at a time followed by some type of finite differencing (such an approach is proposed in, e.g., Spall [46] and Zazanis and Suri [50]). Instead, the analysts should use classic designs to fit first-order or second-order polynomials locally; e.g., the tangent lines in Fig. 2 may be interpreted as first-order polynomials. To fit such polynomials, classic RSM uses highly efficient resolution-III designs and a CCD (see Section 2 and also Joshi et al. [22]). These designs give more accurate estimators for the gradient than one-at-a-time designs. Obviously, the estimated gradient is biased if second-order effects are important and yet a first-order polynomial is fitted.

As in classic RSM, let us assume *locally constant (co)variances* for each of the $r$ simulation outputs; i.e., when moving to a new local area, the (co)variances may change. For example, the points A–D in Fig. 2 do not have the same variance for the goal output. Under these assumptions, OLS applied per univariate simulation output gives the BLUE, $\hat{\beta}_h (h = 0, 1, \ldots, r-1)$ defined in (9). These OLS estimators then have the following estimated covariance matrix:

$$\widehat{\mathbf{cov}}(\hat{\beta}_h, \hat{\beta}_{h'}) = \widehat{\mathbf{cov}}(w_h, w_{h'}) \otimes (\mathbf{Z}'\mathbf{Z})^{-1} (h, h' = 0, \ldots, r-1) \tag{18}$$

where $\otimes$ is the well-known 'Kronecker product' operator and $\widehat{\mathbf{cov}}(w_h, w_{h'})$ is an $r \times r$ matrix with the classic estimators of the (co)variances based on the $m$ replicates at the local center

$$\widehat{\mathbf{cov}}(w_h, w_{h'}) = (\hat{\sigma}_{h;h'}) = \left( \sum_{l=1}^{m} (w_{h;l} - \bar{w}_h)(w_{h';l} - \bar{w}_{h'}) \right) \frac{1}{m-1}. \tag{19}$$

The Kronecker product implies that $\widehat{\mathbf{cov}}(\hat{\beta}_h, \hat{\beta}_{h'})$ is an $rq \times rq$ matrix with $q$ denoting the number of regression parameters (e.g., $q = 1 + k$ in a first-order polynomial), formed from the $r \times r$ matrix $\widehat{\mathbf{cov}}(w_h, w_{h'})$ by multiplying each of its elements by the entire $q \times q$ matrix $(\mathbf{Z}'\mathbf{Z})^{-1}$ (in Eq. (2), $\mathbf{Z}$ was an $N \times (1 + k)$ matrix). The matrix $\widehat{\mathbf{cov}}(w_h, w_{h'})$ is singular if $m \leqslant r$; e.g., the case study in Kleijnen [27] has $r = 2$ response types and $k = 14$ inputs so $m \geqslant 3$ replicates of the center point are required. Of course, the higher $m$ is, the higher is the power of the tests that use these replicates.

Another reason for replicating the center point is that this point is used to test whether a constraint is binding in the current local area (see Eq. (20) below). The center point is more representative of the local behavior than the points of the resolution-III design. Classic RSM also uses replication of the center point when using a CCD (for estimating a second-order polynomial).

Let us further assume that the $r$-variate simulation output is *multivariate Gaussian*. Then (as in classic RSM) the *validity* of the local metamodel may be tested through the classic lack-of-fit $F$-statistic; see Myers and Montgomery [35]. This test assumes that no CRN are applied. In GRSM there are multiple simulation responses, so this classic test is combined with *Bonferroni's inequality*; i.e., the classic type-I error rate $\alpha$ is replaced by the 'experimentwise' or 'familywise' error rate $\alpha/r$. If CRN are applied, then other validation statistics may be computed; see Kleijnen [30].

If the metamodel is rejected, then there are two options: (i) Decrease the local area; e.g., halve each factor's range. (ii) Increase the order of the polynomial; e.g., switch from a first-order to a second-order polynomial. I do not explore these options further in this overview.

Testing the KKT conditions in random simulation implies testing the following three null-hypotheses, denoted by the superscripts (1)–(3):

(1) The current solution is feasible and at least one constraint is binding; see (6):

$$H_0^{(1)}: E(w_{h'}|\mathbf{d}=\mathbf{0}) = a_{h'} \quad \text{with } h' = 1,\ldots,r-1 \tag{20}$$

where $\mathbf{d}=\mathbf{0}$ corresponds with the center of the local area expressed in the coded (standardized) inputs.

(2) The expected value of the estimated goal gradient may be expressed as the expected value of a *linear* combination of the estimated gradients of the simulation outputs in the binding constraints; i.e., in (17) the deterministic quantities are replaced by their random estimators

$$H_0^{(2)}: E(\hat{\beta}_{0;-0}) = E\left(\sum_{h\in A(\mathbf{z}^0)} \hat{\lambda}_h^0 \hat{\beta}_h\right). \tag{21}$$

(3) The Lagrangian multipliers in (21) are non-negative

$$H_0^{(3)}: E(\hat{\lambda}) \geqslant \mathbf{0}. \tag{22}$$

Each of these three hypotheses requires multiple tests, so Bonferroni's inequality is applied. Moreover, these three hypotheses are tested sequentially, so it is hard to control the final type-I and type-II error probabilities. However, classic RSM has the same type of problems, and nevertheless it has acquired a track record in practice. The numerical results for the GRSM are also promising (see below).

*Sub 1*: To test the hypothesis (20), the classic $t$ statistic may be used

$$t_{m-1}^{(h')} = \frac{\bar{w}_{h'}(\mathbf{d}=\mathbf{0},\mathbf{r}_0) - a_{h'}}{\sqrt{\hat{\sigma}_{h';h'}/m}} \quad \text{with } h' = 1,\ldots,r-1 \tag{23}$$

where $\mathbf{r}_0$ is the vector with the seeds of the PRN streams (this vector may consist of a single element); both the numerator and the denominator are based on the $m$ replicates at the local center point; see (19).

*Note*: To save simulation runs, a local experiment should start at its center point including replicates. If it turns out that either no constraint is binding or at least one constraint is violated, then the other hypotheses need not be tested so the remainder of the local design is not simulated.

The $t$ statistic in (23) may give the following three different results:

- The statistic is *significantly positive*; i.e., the constraint for output $h'$ is not binding. If none of the $(r-1)$ constraints is binding, then the optimal solution is not yet found (assuming that at the optimum at least one constraint is binding; otherwise, classic RSM applies). The search for better solutions continues (see again Section 3).
- The statistic is *significantly negative*; i.e., the current local area does not give feasible solutions so the optimal solution is not yet found. The search should back-up into the feasible area.
- The statistic is *non-significant*; i.e., the current local area gives feasible solutions, and the constraint for output $h'$ is binding. The index of this gradient is then included in $A(\mathbf{z}^0)$; see (21). And the KKT test proceeds as follows.

*Sub 2*: The hypothesis (21) is that the expected value of the goal gradient may be expressed as the expected value of a *linear* combination of the estimated gradients of the binding constraints. To estimate this linear combination, OLS may be applied with as explanatory variables the estimated gradients of the (say) $J$ binding constraints (so the explanatory variables become random). The latter gradients are collected in the $k \times J$ matrix $\widehat{\mathbf{B}}_{J;-0}$. The parameters estimated through OLS are $\widehat{\lambda}$. Let $\widehat{\beta}_{0;-0}$ denote the OLS estimator of the goal gradient; i.e.,

$$\widehat{\hat{\beta}}_{0;-0} = \widehat{\mathbf{B}}_{J;-0}\left(\widehat{\mathbf{B}}'_{J;-0}\widehat{\mathbf{B}}_{J;-0}\right)^{-1}\widehat{\mathbf{B}}'_{J;-0}\widehat{\beta}_{0;-0} = \widehat{\mathbf{B}}_{J;-0}\widehat{\lambda} \tag{24}$$

with $\widehat{\lambda} = (\widehat{\mathbf{B}}'_{J;-0}\widehat{\mathbf{B}}_{J;-0})^{-1}\widehat{\mathbf{B}}'_{J;-0}\widehat{\beta}_{0;-0}$.

The *validity* of this linear approximation may be quantified through the $k$-dimensional vector of its residuals

$$\hat{\mathbf{e}}\left(\hat{\tilde{\beta}}_{0;-0}\right) = \hat{\tilde{\beta}}_{0;-0} - \hat{\beta}_{0;-0}. \tag{25}$$

The hypothesis (21) implies $E(\hat{\mathbf{e}}(\hat{\tilde{\beta}}_{0;-0})) = \mathbf{0}$. This hypothesis involves the product of multivariates, so standard tests do not apply. Therefore *bootstrapping* may be used. Distribution-free bootstrapping does not apply because in expensive simulation only the center point is replicated a few times. Instead, *parametric bootstrapping* should be used; i.e., a specific distribution type is assumed and its parameters are estimated from the simulation's I/O data at hand (so the bootstrap is called 'data driven'; it is also known as a 'Monte Carlo' method). Like in classic RSM, the standard type of distribution assumed for testing the KKT conditions is the Gaussian distribution.

Fig. 3 shows the following three layers of models:

(1) The simulation model, which is treated as a black box. Like in Fig. 2, only two simulation inputs and three simulation outputs are pictured.
(2) The regression metamodel, which uses the simulation I/O data $(\mathbf{Z}, \mathbf{w})$ as input and estimates the gradients of the goal response ($\hat{\beta}_{0;-0}$) and of the constrained responses including the binding constraints collected in $\hat{\mathbf{B}}_{J;-0}$. The regression analysis also estimates $\widehat{\mathbf{cov}}(\hat{\beta}_{0;-0}, \hat{\mathbf{B}}_{J;-0})$ (covariance matrix of these estimated gradients); the vec operator in Fig. 3 is explained below.
(3) The bootstrap model, which uses the regression output ($\hat{\beta}_{0;-0}$, $\hat{\mathbf{B}}_{J;-0}$, $\widehat{\mathbf{cov}}(\hat{\beta}_{0;-0}, \hat{\mathbf{B}}_{J;-0})$) as parameters of the multivariate normal distribution of its output $\hat{\beta}^*_{0;-0}$ and $\hat{\mathbf{B}}^*_{J;-0}$, where the superscript $*$ denotes bootstrapped values.

More specifically, the bootstrap procedure consists of the following four steps:

(1) Use the *Monte Carlo* method to sample $\text{vec}(\hat{\beta}^*_{0;-0}, \hat{\mathbf{B}}^*_{J;-0})$, which is a $(k + kJ)$-dimensional vector formed by 'stapling' (stacking) the $k$-dimensional goal gradient vector and the $J$ $k$-dimensional vectors of the $k \times J$ matrix $\hat{\mathbf{B}}^*_{J;-0}$:

$$\text{vec}\left(\hat{\beta}^*_{0;-0}, \hat{\mathbf{B}}^*_{J;-0}\right) \sim N\left(\text{vec}\left(\hat{\beta}_{0;-0}, \hat{\mathbf{B}}_{J;-0}\right), \widehat{\mathbf{cov}}\left(\text{vec}\left(\hat{\beta}_{0;-0}, \hat{\mathbf{B}}_{J;-0}\right)\right)\right) \tag{26}$$

where $\widehat{\mathbf{cov}}(\text{vec}(\hat{\beta}_{0;-0}, \hat{\mathbf{B}}_{J;-0}))$ is the $(k + kJ) \times (k + kJ)$ matrix computed through (18).
(2) Use the bootstrap values resulting from Step 1, to compute the OLS estimate of the bootstrapped goal gradient using the bootstrapped gradients of the binding constraints as explanatory variables; i.e., use (24) adding the superscript $*$ to all random variables resulting in $\hat{\beta}^*_{0;-0}$ and $\hat{\lambda}^*$.

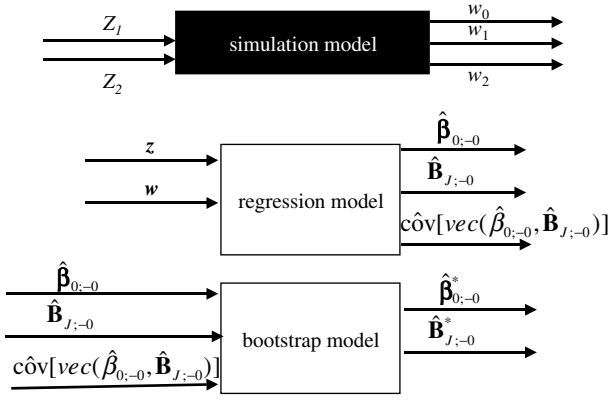


Fig. 3. I/O of simulation, regression, and bootstrap models.

(3) Use $\widehat{\hat{\beta}}^*_{0;-0}$ from Step 2 and $\hat{\beta}^*_{0;-0}$ from Step 1 to compute the *bootstrap residual* $\hat{\mathbf{e}}(\widehat{\hat{\beta}}^*_{0;-0}) = \widehat{\hat{\beta}}^*_{0;-0} - \hat{\beta}^*_{0;-0}$ (analogous to Eq. (25)). Furthermore, determine whether any of the *bootstrapped Lagrangian multipliers* $\hat{\lambda}^*$ (found in Step 2) is negative; i.e., augment a counter (say) $c^*$ with the value 1 if this event occurs.

(4) *Repeat* the preceding three steps (say) 1000 times (this is known as the 'bootstrap sample size'). This gives the estimated density function (EDF) of the bootstrapped residuals per input $j (j = 1, \ldots, k) \hat{\mathbf{e}}(\widehat{\hat{\beta}}^*_{0;-0;j})$ and the final value of the counter $c^*$. Reject the hypothesis in (21) if this EDF implies a two-sided $(1 - \alpha/(2k))$ confidence interval that does not cover the value 0 (the factor $k$ is explained by Bonferroni's inequality). Reject the hypothesis in (22) if the fraction $c^*/1000$ is significantly higher than 50% (if the true Lagrangian multiplier is only 'slightly' larger than zero, then 'nearly' 50% of the bootstrapped values is negative). To test the latter fraction, the binomial distribution may be approximated through the normal distribution with mean 0.50 and variance $(0.50 \times 0.50)/1000 = 0.00025$.

The numerical examples in Bettonvil et al. [9] give the following encouraging results:

(1) The classic $t$ test for zero slacks and the classic lack-of-fit $F$-test perform as expected.

(2) The new bootstrap tests give observed type-I error rates close to the prespecified (nominal) rates; the type-II error rate (complement of the power) decreases as the input combination tested moves farther away from the true optimum (see the points A–D in Fig. 2).

## 5. Conclusions

In this overview, I first summarized classic RSM, assuming a single response variable. I added the adapted steepest ascent (ASA) search direction, which improves the classic steepest ascent.

Next, I summarized GRSM for simulation with multivariate responses, assuming that one response is to be minimized while all the other responses must meet given constraints. Moreover, the (deterministic) inputs must satisfy given box constraints.

Finally, I summarized a procedure for testing whether an estimated optimum is truly optimal — using the KKT conditions. This procedure combines classic tests and bootstrapped tests.

I hope that this overview will stimulate other researchers to further investigate GRSM; e.g., they may apply GRSM and competing heuristics to a variety of practical simulation models. (Currently, some colleagues and I are applying both GRSM and OptQuest to the test function in Angün [1] and the call-center simulation in Kelton et al. [23].)

## Acknowledgements

Ebru Angün (Galatasaray University, Istanbul) commented on a previous version of this article; as a Ph.D. student at Tilburg University she did much of the research underlying this overview. I also thank my Tilburg colleagues Dick den Hertog and Gül Gürkan, who shared my supervision and coaching of Ebru's Ph.D. research. Two anonymous referees also provided useful comments.

## References

[1] M.E. Angün, Black Box Simulation Optimization: Generalized Response Surface Methodology, CentER Dissertation series, Tilburg University, Tilburg, The Netherlands, 2004.

[2] E. Angün, D. den Hertog, G. Gürkan, J.P.C. Kleijnen, Response surface methodology revisited, in: E. Yucesan, C.H. Chen, J.L. Snowdon, J.M. Charnes (Eds.), Proceedings of the 2002 Winter Simulation Conference, 2002, pp. 377–383.

[3] E.R. Barnes, A variation on Karmarkar's algorithm for solving linear programming problems, Mathematical Programming 36 (1986) 174–182.

[4] R.R. Barton, M. Meckesheimer, Metamodel-based simulation optimizationHandbooks in Operations Research and Management Science, vol. 13, Elsevier/North Holland, 2006, pp. 535–574.

[5] T. Bartz-Beielstein, Experimental Research in Evolutionary Computation; The New Experimentalism, Springer, Berlin, 2006.

[6] S. Bashyam, M.C. Fu, Optimization of $(s, S)$ inventory systems with random lead times and a service level constraint, Management Science 44 (1998) 243–256.

[7] R.A. Bates, R.S. Kenett, D.M. Steinberg, H.P. Wynn, Achieving robust design from computer simulations, Quality Technology and Quantitative Management 3 (2) (2006) 161–177.
[8] I. Ben-Gal, J. Bukchin, Ergonomic design of working environment via rapid prototyping tools and design of experiments, IIE Transactions 34 (4) (2002) 375–391.
[9] B. Bettonvil, E. del Castillo, J.P.C. Kleijnen, Statistical Testing of Optimality Conditions in Multiresponse Simulation-based Optimization. Working Paper, Tilburg University, Tilburg, The Netherlands, 2007.
[10] G. Boente, D. Rodriguez, Robust estimators of high order derivatives of regression functions, Statistics and Probability Letters, in press.
[11] G.E.P. Box, K.B. Wilson, On the experimental attainment of optimum conditions, Journal Royal Statistical Society, Series B 13 (1) (1951) 1–38.
[12] R. Brekelmans, L. Driessen, H. Hamers, D. den Hertog, Gradient estimation schemes for noisy functions, Journal of Optimization Theory and Applications 126 (3) (2005) 529–551.
[13] B. Dengiz, T. Bektas, A.E. Ultanir, Simulation optimization based DSS application: a diamond tool production line in industry, Simulation Modelling Practice and Theory 14 (3) (2006) 296–312.
[14] D.J. Downing, R.H. Gardner, F.O. Hoffman, An examination of response-surface methodologies for uncertainty analysis in assessment models, Technometrics 27 (2) (1985) 151–163 (Discussion: 28 (1) (1986) 91–93).
[15] M.C. Fu, Gradient EstimationHandbooks in Operations Research and Management Science, vol. 13, Elsevier/North Holland, 2006, pp. 575–616.
[16] M.C. Fu, F.W. Glover, J. April, Simulation optimization: a review, new developments, and applications, in: M.E. Kuhl, N.M. Steiger, F.B. Armstrong, J.A. Joines (Eds.), Proceedings of the 2005 Winter Simulation Conference, Institute of Electrical and Electronics Engineers, Piscataway, NJ, 2005, pp. 83–95.
[17] P.E. Gill, W. Murray, M.H. Wright, Practical Optimization, 12th ed., Academic Press, London, 2000.
[18] A.Y. Gordon, Unimprovability of the Bonferroni procedure in the class of general step-up multiple testing procedures, Statistics and Probability Letters, in press.
[19] S.G. Henderson, B.L. Nelson (Eds.), Handbooks in Operations Research and Management Science, vol. 13, Elsevier/North Holland, 2006.
[20] M. Irizarry, J.R. Wilson, J. Trevino, A flexible simulation tool for manufacturing-cell design, II: response surface analysis and case study, IIE Transactions 33 (2001) 837–846.
[21] C. Ivanescu, W. Bertrand, J. Fransoo, J.P.C. Kleijnen, Bootstrapping to solve the limited data problem in production control: an application in batch processing industries, Journal of the Operational Research Society 57 (1) (2006) 2–9.
[22] S. Joshi, H.D. Sherali, J.D. Tew, An enhanced response surface methodology (RSM) algorithm using gradient deflection and second-order search strategies, Computers and Operations Research 25 (7/8) (1998) 531–541.
[23] W.D. Kelton, R.P. Sadowski, D.T. Sturrock, Simulation with Arena, fourth ed., McGraw-Hill, Boston, 2007.
[24] R. Kenett, D. Steinberg, New frontiers in design of experiments, Quality Progress (2006) 61–65.
[25] A.I. Khuri, Multiresponse surface methodology, in: S. Ghosh, C.R. Rao (Eds.), Handbook of Statistics, vol. 13, Elsevier, Amsterdam, 1996.
[26] A.I. Khuri, J.A. Cornell, Response Surfaces: Design and Analysis, second ed., Marcel Dekker, New York, 1996.
[27] J.P.C. Kleijnen, Simulation and optimization in production planning: a case study, Decision Support Systems 9 (1993) 269–280.
[28] J.P.C. Kleijnen, Experimental design for sensitivity analysis, optimization, and validation of simulation models, in: J. Banks (Ed.), Handbook of Simulation, Wiley, New York, 1998, pp. 173–223.
[29] J.P.C. Kleijnen, White noise assumptions revisited: regression metamodels and experimental designs for simulation practice, in: L.F. Perrone, F.P. Wieland, J. Liu, B.G. Lawson, D.M. Nicol, R.M. Fujimoto, (Eds.), Proceedings of the 2006 Winter Simulation Conference, pp. 107–117.
[30] J.P.C. Kleijnen, DASE: Design and Analysis of Simulation Experiments, Springer, New York, 2008 (available October 2007).
[31] J.P.C. Kleijnen, D. den Hertog, E. Angün, Response surface methodology's steepest ascent and step size revisited, European Journal of Operational Research 159 (2004) 121–131.
[32] J.P.C. Kleijnen, D. den Hertog, E. Angün, Response surface methodology's steepest ascent and step size revisited: correction, European Journal of Operational Research 170 (2006) 664–666.
[33] J.P.C. Kleijnen, J. Wan, Optimization of simulated systems: OptQuest and alternatives, Simulation Modelling Practice and Theory 15 (2007) 354–362.
[34] A.M. Law, Simulation Modeling and Analysis, fourth ed., McGraw-Hill, Boston, 2007.
[35] R.H. Myers, D.C. Montgomery, Response Surface Methodology: Process and Product Optimization Using Designed Experiments, second ed., Wiley, New York, 2002.
[36] R.H. Myers, A.I. Khuri, W.H. Carter, Response surface methodology: 1966–1988, Technometrics 31 (2) (1989) 137–157.
[37] S.H. Ng, K. Xu, W.K. Wong, Optimization of Multiple Response Surfaces with Secondary Constraints, Working Paper, National University of Singapore, 2006.
[38] R. Nicolai, R. Dekker, Automated Response Surface Methodology for Stochastic Optimization Models with Unknown Variance, Working Paper, Erasmus University, Rotterdam, The Netherlands, 2005.
[39] J.T. Oden (Chair), Revolutionizing Engineering Science through Simulation, National Science Foundation (NSF), Blue Ribbon Panel on Simulation-based Engineering Science, 2006.
[40] L. Olivi, Response Surface Methodology, Handbook for Nuclear Reactor Safety. EUR 9600, Commission of the European Communities, Joint Research Centre, Ispra, Italy, 1984.

[41] D.M. Osborne, R.L. Armacost, J. Pet-Edwards, State of the art in multiple response surface methodology, in: Proceedings of the IEEE/SMC Conference, Orlando, FL, 1997.
[42] R. Rikards, J. Auzins, Response surface method for solution of structural identification problems, in: Fourth International Conference on Inverse Problems in Engineering, Rio de Janeiro, Brazil, 2002.
[43] S.C. Rosen, C.M. Harmonosky, M.T. Traband, Optimization of systems with multiple performance measures via simulation: survey and recommendations, Computers and Industrial Engineering, in press.
[44] M.H. Safizadeh, R. Signorile, Optimization of simulation via quasi-Newton methods, ORSA Journal on Computing 6 (4) (1994) 398–408.
[45] T.W. Simpson, A.J. Booker, D. Ghosh, A.A. Giunta, P.N. Koch, R.-J. Yang, Approximation methods in multidisciplinary analysis and optimization: a panel discussion, Structural and Multidisciplinary Optimization 27 (5) (2004) 302–313.
[46] J.C. Spall, Introduction to Stochastic Search and Optimization; Estimation, Simulation, and Control, Wiley, New York, 2003.
[47] E.D. Stinstra, H.P. Stehouwer, J. van der Heijden, Collaborative tube design optimization: an integral meta-modeling approach, in: Proceedings of the Fifth ISSMO Conference on Engineering Design Optimization, 2003.
[48] E. Tekin, I. Sabuncuoglu, Simulation optimization: a comprehensive review on theory and applications, IIE Transactions 36 (2004) 1067–1081.
[49] T. Yang, L. Tseng, Solving a multi-objective simulation model using a hybrid response surface method and lexicographical goal programming approach: a case study on integrated circuit ink-marking machines, Journal of the Operational Research Society 53 (2) (2002) 211–221.
[50] M.A. Zazanis, R. Suri, Convergence rates of finite-difference sensitivity estimates for stochastic systems, Operations Research 41 (4) (1993) 694–703.