Response Surface Methodology

Kleijnen, Jack P.C.

Publication date:
2014

No. 2014-013

RESPONSE SURFACE METHODOLOGY

By

Jack P.C. Kleijnen

14 February, 2014

ISSN 0924-7815
ISSN 2213-9532

manuscript No.
(will be inserted by the editor)

Response Surface Methodology

Jack P.C. Kleijnen

Tilburg University (e-mail: Kleijnen@TilburgUniversity.edu)

The date of receipt and acceptance will be inserted by the editor

Abstract This chapter first summarizes Response Surface Methodology (RSM), which started with Box and Wilson's article in 1951 on RSM for real, non-simulated systems. RSM is a stepwise heuristic that uses first-order polynomials to approximate the response surface locally. An estimated polynomial metamodel gives an estimated local gradient, which RSM uses in steepest ascent (or descent) to decide on the next local experiment. When RSM approaches the optimum, the latest first-order polynomial is replaced by a second-order polynomial. The fitted second-order polynomial enables the estimation of the optimum. Furthermore, this chapter focuses on simulated systems, which may violate the assumptions of constant variance and independence. The chapter also summarizes a variant of RSM that is proven to converge to the true optimum, under specific conditions. The chapter presents an adapted steepest ascent that is scale-independent. Moreover, the chapter generalizes RSM to multiple random responses, selecting one response as the goal variable and the other responses as the constrained variables. This generalized RSM is combined with mathematical programming to estimate a better search direction than the steepest ascent direction. To test whether the estimated solution is indeed optimal, bootstrapping may be used. Finally, the chapter discusses robust optimization of the decision variables, while accounting for uncertainties in the environmental variables.

Keywords: simulation; optimization; regression, robustness; risk
JEL: C0, C1, C9

1 Introduction

RSM is one of the more popular methods in simulation optimization. Originally, RSM was developed for the optimization of *real* (physical) systems;

see the classic article by Box and Wilson (1951), the overview of RSM during the period 1966-1988 by Myers et al. (1989), the recent third edition of thee popular handbook by Myers et al. (2009), and recent RSM software on the Web. RSM in *simulation* was discussed in the monograph by Kleijnen (1975). One of the first case-studies on RSM in simulation is Van den Bogaard and Kleijnen (1977), who simulated a computer center with two servers and three priority classes—namely, small, medium, and large jobs—for which they estimated the 90% quantiles of the waiting times per class, for different class limits, and applied RSM to find the optimal class limits. RSM in simulation is also discussed by Barton and Meckesheimer (2006), Bartz-Beielstein (2006), Law (2014), and Rosen et al. (2008). Google gave more than two million results for the term "Response Surface Methodology", on February 4, 2014. Unfortunately, RSM (unlike some other search heuristics) has not yet been implemented as an add-on to commercial simulation software.

RSM treats the simulation model as a *black box*; i.e., RSM observes the input/output (I/O) of the simulation model, but not the internal variables and specific functions implied by the simulation's computer modules. RSM is a *sequential* heuristic; i.e., it uses a sequence of local experiments that may lead to the optimum input combination. Note that an input combination is also called a point or a scenario. RSM uses design of experiments (DOE) and the concomitant linear regression analysis. Though RSM is only a heuristic, it has gained a good track record—as we shall indicate in the next sections. Moreover, practitioners may not be interested in convergence proofs, because realistic experiments take so much time that large sample sizes are impossible; practitioners may be more interested in finding better solutions than the current one (a French expression claims that "the best is the enemy of the better").

More specifically, RSM uses a sequence of local *metamodels* (approximations) that are first-order polynomials in the inputs; once the optimum seems close, RSM augments the latest first-order polynomial to a second-order polynomial. Notice that terminology may be misleading, because many authors use the term "response surface' instead of "metamodel"; an example is Rikards and Auzins (2002). Note that RSM may be combined with white-box methods that give estimated gradients, which may improve RSM; see Qu and Fu (2012).

We focus on *discrete-event simulation*, though RSM has also been applied to deterministic simulation and the simulation of sets of random nonlinear difference equations. By definition, random simulation (including discrete event simulation) uses *pseudorandom numbers* (PRN). We do not discuss deterministic simulation with numeric noise.

Furthermore, we assume that the inputs are continuous variables. We focus on simulation that has a *mean* (expected value) as "the" response (output). Nevertheless, we have also applied RSM to outputs that are quantiles; see again Van den Bogaard and Kleijnen (1977). The response may also be a probability (e.g., the overflow probability in a queueing system),

which we may formulate as the expected value of a binary variable, using the indicator function.

The optimum solution for the decision variables may turn out to be inferior when ignoring uncertainties in the non-controllable environmental variables; i.e., these uncertainties create a risk. Besides "robust optimization" originated by Taguchi for the design of products, Ben-Tal developed robust optimization in mathematical programming with uncertain coefficients; see Taguchi (1987) and the update in Myers et al. (2009, pp. 483-555), and Ben-Tal and Nemirovski (1998) and the update in Yanikoglu et al. (2013). We shall discuss both approaches.

We assume that RSM starts after the important factors (inputs) and their experimental area have been identified; i.e., before RSM starts we may need to use *factor screening* to identify the really important factors among the many conceivably important factors. A recent survey of various screening methods is Shi et al. (2014). However, Chang et al. (2014) combine RSM with screening in a single method (called STRONG-S); we point out that RSM without a preceding screening phase may imply the simulation of extremely many combinations of simulation inputs, as we shall see in Section 2.

The remainder of this chapter is organized as follows. Section 2 summarizes classic RSM, developed for real-life experimentation. Section 3 adapts this RSM to the needs of simulation. Section 4 presents the adapted steepest descent (ASD) search direction, developed by Kleijnen et al. (2004). Section 5 summarizes GRSM for simulation with multiple responses, developed by Angün et al. (2009). Section 6 summarizes a procedure for testing whether an estimated optimum is truly optimal—using the Karush-Kuhn-Tucker (KKT) conditions—developed by Bettonvil et al. (2009). Section 7 discusses robust optimization. Section 8 presents conclusions. The chapter ends with many references, enabling further study of details. This chapter updates and extends Kleijnen (2008) Paragraphs starting with "Note:" may be skipped in a first reading.

2 RSM: Basics

As we have already said, the basics of RSM were developed in real-life experiments for the simplest type of optimization problems; namely, minimize the expected value of a single output, with continuous inputs and without any constraints:

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

where $E(w_0|\mathbf{z})$ is the goal or objective output, which is to be minimized through the choice of the input combinations $\mathbf{z} = (z_1, \ldots, z_k)'$ where z_j ($j = 1, \ldots k$) denotes the j^{th} "original" input; i.e., the inputs are not standardized such that they lie between -1 and 1.

Note: We use z for the original input and x for the standardized input; RSM is scale dependent as we shall see. For the output we use w, which

may be a mnemonic referring to "waiting time" which is an output of many simulation models.

RSM has the following *characteristic*s, which we shall detail next.

(i) RSM is an *optimization heuristic* that tries to estimate the input combination that minimizes a given goal function; see again (1). Because RSM is a heuristic, success is not guaranteed.

(ii) RSM is a *stepwise* (multi-stage) method; see the steps below.

(iii) In each step, RSM fits a local first-order *polynomial* regression (meta)model—except for the last step, in which RSM fits a second-order polynomial.

(iv) To fit (estimate, calibrate) these first-order polynomials, RSM uses I/O data obtained through so-called *resolution-III (R-III) designs*; for the second-order polynomial, RSM uses a *central composite design (CCD)*; we shall define these R-III designs and CCD later in this section.

(v) Each step—except the last one—selects the direction for changing the inputs through the *gradient* implied by the first-order polynomial fitted in that 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.

(vi) 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 subregion: does that region have a unique minimum, a saddle point, or a ridge with stationary points?

More specifically, RSM consists of the following *steps*; also see Figure 1 in Section 5, which gives an example with a single goal function and two constrained random outputs; these constraints vanish in classic RSM.

RSM must be initialized by the analysts who select a *starting point*. They may select the input combination currently used in practice if the system already exists. Otherwise, they should use intuition and prior knowledge—as in many other heuristics.

For the *neighborhood* of this starting point, RSM explores the I/O behavior of the observed system. RSM approximates this behavior through a local first-order polynomial—as Taylor's series expansion suggests—augmented with additive *white noise* e:

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j z_j + e. \tag{2}$$

We denote the regression parameters by $\beta = (\beta_0, \beta_1, ..., \beta_k)'$ with the intercept β_0 and the k first-order or "main" effects β_j $(j = 1, \ldots, k)$. White noise means that e is normally, independently, and identically distributed (NIID) with zero mean and a constant variance (say) σ^2 in the local experimental area: $e \sim NIID(0, \sigma^2)$. When the next step moves to a new local area, the variance may change. Obviously, (2) is a linear regression model with white

Combi. i	**1**	**2**	**3**	**4 = 1.2**	**5 = 1.3**	**6 = 2.3**	**7 = 1.2.3**
1	-	-	-	+	+	+	-
2	+	-	-	-	-	+	+
3	-	+	-	-	+	-	+
4	+	+	-	+	-	-	-
5	-	-	+	+	-	-	+
6	+	-	+	-	+	-	-
7	-	+	+	-	-	+	-
8	+	+	+	+	+	+	+

Table 1 A fractional factorial design for seven factors

noise, so *least squares* (LS) gives the best linear unbiased estimator (BLUE) of β:

$$\widehat{\beta} = (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{w} \tag{3}$$

where $\mathbf{Z}$ denotes the $N \times (k+1)$ matrix that is determined by the R-III design and the m_i replications of point i, and $\mathbf{w}$ denotes the vector with N outputs; obviously, $N = \sum_{i=1}^{n} m_i$ so $\mathbf{Z}$ has m_i identical rows where the first element of each row is 1 and corresponds with the intercept β_0.

To select the n input combinations needed to estimate these $k+1$ effects, RSM uses a R-III design with $n \geq k+1$. In classic R-III designs this n is a multiple of four; e.g., if $k = 7$, then Table 1 gives a 2^{7-4} fractional factorial design in the standardized inputs x_j so - means -1 and + means 1; the last column has the so-called generator $\mathbf{7 = 1.2.3}$, which means $x_{i;7} = x_{i;1}x_{i;2}x_{i;3}$; the generators in the columns 4, 5, and 6 are defined analogously. The design is "balanced"; i.e., each column has $n/2$ values equal to -1. Moreover, all pairs of columns are orthogonal. If n is not a power of 2, then the R-III design is called a Plackett-Burman design, and is tabulated in many publications and provided by DOE software; e.g., Kleijnen (2014) gives the design for $k = 11$ inputs and $n = 12$ combinations.

Unfortunately, there are no general guidelines for determining the appropriate *size* of the local area in each step; again, intuition and prior knowledge are important. Chang et al. (2013), however, decide on the size of the local area, using a so-called trust region; also see Section 3.

To decide on the next subarea, RSM follows the *steepest descent path*; e.g., if the estimated first-order effects are such that $\widehat{\beta}_1 >> \widehat{\beta}_2 > 0$, then RSM decreases 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 et al. (2009). In Section 4 we shall present a scale-independent variant.

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 output that is significantly higher instead of lower, then they may reduce the step size. Otherwise, they take one more step in the current steepest descent direction. In Section 5 we shall present a more sophisticated mathematical procedure for selecting the step size.

After a number of steps along the steepest descent path, the output will increase instead of decrease: the first-order polynomial is only a local approximation of the true I/O function. When such deterioration occurs, RSM observes the $n > k$ combinations specified by a R-III design centered around the best point found so far; i.e., RSM may use the same design as in step 2 (see Table 1), but translate the standardized inputs x_j into different values for the original inputs z_j. The best combination found so far, may be one of the corner points of the design; see again Figure 1. Next RSM estimates the first-order effects in the new local polynomial approximation, using LS; see (3). And so the search continues.

It is intuitively clear that the *plane* implied by the most recent local first-order polynomial cannot adequately represent a *hill top* when searching to maximize a function or—equivalently—minimize (1). 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 uses an F statistic to test whether all estimated first-order effects—and hence the gradient—are zero. These diagnostics are given by all modern regression software packages. Instead of these diagnostics, RSM might use "cross-validation". More details including references are given by Kleijnen (2014).

If the most recently fitted first-order polynomial turns out to be inadequate, then RSM fits a *second-order polynomial*:

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j z_j + \sum_{j=1}^{k} \sum_{j'=k}^{k} \beta_{j;j'} z_j z_{j'} + e. \tag{4}$$

To estimate the $q = 1 + 2k + k(k-1)/2$ coefficients of this polynomial, RSM uses a CCD with $n > q$ combinations More precisely, a CCD starts with a *resolution-V* (R-V) design, which by definition gives unbiased estimators of the intercept, the k first-order effects, and the $k(k-1)/2$ cross-products or two-factor interactions in (4). For example, if $k = 7$. then the 2^{7-4} design in Table 1 is replaced by a 2^{7-1} design with the generator $\mathbf{7 = 1.2.3.4.5.6}$ so 64 combinations are used to estimate 29 effects. A CCD augments this R-V design such that the purely quadratic effects $\beta_{j;j}$ can also be estimated. More specifically, a CCD adds the *central* point and $2k$ *axial* points that form a *star design*, where—for the standardized factors—the central point is $(0, \ldots 0)'$ and the "positive" axial point for factor j is $x_j = c$ and all other $(k-1)$ factors are fixed at the center so $x_{j'} = 0$ with $j' = 1, \ldots, k$ and $j' \neq j$; the "negative" axial point for factor j is $x_j = -c$ and $x_{j'} = 0$. Kleijnen (2014) gives a detailed discussion of CCDs, including more efficient so-called "saturated" designs; by definition, the latter designs have $n = q$. Using this fitted second-order polynomial, RSM estimates the optimal values of the inputs by straightforward *differentiation* or by more sophisticated *canonical analysis*; see Myers et al. (2009).

If time permits, then RSM may try to escape from a possible local minimum and *restart* the search from a different initial local area—which brings RSM back to its initial step.

We recommend not to eliminate inputs that have non-significant effects in a first-order polynomial fitted within a local experimental area: these inputs may have significant effects in a next experimental area. A related issue is the determination of the number of replications. Indeed, determining whether the signal-to-noise ratio is big enough, is a moot issue in metamodeling. Recently, this issue is examined in Kriging metamodeling by Ankenman et al. (2010). For the time being, we recommend estimating the true mean response for a given input combination such that a relative precision of (say) 10% has a 90% probability, using the method detailed in Law (2014).

The Taylor series argument suggests that a higher-order polynomial is more accurate than a lower-order polynomial. A statistical counterargument, however, is that *overfitting* gives less accurate estimators of the polynomial coefficients. Consequently, the higher-order polynomial may give a predictor $\widehat{y}$ with lower bias but higher variance such that its mean squared error (MSE) increases. Moreover, a higher-order polynomial requires the simulation of more input combinations.

3 RSM in simulation

We consider the following two characteristics of simulation experiments:

(i) The constant variance assumption does not hold.

(ii) The independence assumption does not hold if common random numbers (CRN) are applied.

Many simulation experiments concern queueing systems; examples are supply chains and telecommunication networks. The simplest queueing model is the so-called M/M/1 model, which has one server and exponential arrival and service times so it is a Markov chain. It is well known that as the traffic rate of the M/M/1 model increases, the mean steady-state waiting time increases and the variance increases even more; consequently, the assumption of a constant variance does not hold.

CRN are often applied in simulation experiments, because it is the default option in simulation software such as Arena; moreover, CRN are a simple and intuitive variance reduction technique that gives more accurate estimators of the regression parameters—except for the intercept β_0. The outputs of all input combinations that use CRN are statistically dependent or correlated. CRN are related to "blocking" in real-life experiments. In simulation experiments, we may use blocking when combining CRN and antithetic random numbers through the so-called Schruben-Margolin strategy; this strategy is detailed by Chih (2013).

The preceding two characteristics imply that the ordinary LS (OLS) does not give the BLUE. Weighted LS (WLS) does give the BLUE, but assumes known response variances. Generalized LS (GLS) gives the BLUE but assumes known response variances and covariances. We therefore recommend

the following simple estimator, which is detailed by Kleijnen (2015): assuming a constant number of replications m (as is usual in CRN), we compute the OLS per replication replacing $\mathbf{w}$ in (3) by $\mathbf{w}_r$ to get $\widehat{\beta}_r$ (r = 1, ..., m). Replication r then gives an estimate of the steepest descent direction if a first-order polynomial is used or the optimum input combination if a second-order polynomial is used. Together, the m replications give an estimate of the accuracy of this estimated direction or optimum; if the analysts find this accuracy too low, then they may simulate additional replications. We have not yet any experience with this simple sequential approach.

Actually, if we have $m_i > 1$ (i = 1, ..., n) replications, then we can further explore the statistical properties of the LS estimator of β through *bootstrapping*. The classic textbook on bootstrapping in general is Efron and Tibshirani (1993); many more references are given by Kleijnen (2014). To explain *distribution-free* bootstrapping in RSM, we first consider the case of no CRN. We resample—with replacement—the m_i simulation outputs $w_{i;r}$ (r = 1, ..., m_i) for input combination i (i = 1, ..., n), and obtain the bootstrapped simulation outputs $w^*_{i;r}$ where the superscript $*$ is the general symbol for bootstrapped observations. Resampling for each input combination i, we obtain the bootstrapped I/O data $(\mathbf{Z}, \mathbf{w}^*)$ with the N-dimensional vector $\mathbf{w}^* = (w^*_{1;1}, \ldots, w^*_{1;m_1}, ..., w^*_{n;1}, ..., w^*_{nm_n})^T$ (so $N = \sum_{i=1}^{n} m_i$) and the "original" $N \times q$ matrix of input combinations $\mathbf{Z}$. Next we consider the case of CRN. The n elements of $\mathbf{w}_r = (w_{1;r}, \ldots, w_{n;r})^T$ (r = 1, ..., m) are then not identically and independently distributed (IID), so we resample the m IID vectors $\mathbf{w}_r$. This case also gives $(\mathbf{Z}, \mathbf{w}^*)$. In both cases (CRN or no CRN) we use $(\mathbf{Z}, \mathbf{w}^*)$ to compute the bootstrapped regression parameter estimator $\widehat{\beta}^*$replacing $\mathbf{w}$ in (3) by $\mathbf{w}^*$. This bootstrapping we repeat (say) B times; popular choices are $B = 100$ or $B = 1{,}000$. This bootstrap sample gives $\widehat{\beta}^*_b$ ($b = .1, ..., B$) where we replace $\mathbf{w}$ in (3) by $\mathbf{w}^*_b$. Sorting these $\widehat{\beta}^*_b$ in ascending order gives the estimated density function (EDF) of $\widehat{\beta}^*$. We can also use $\widehat{\beta}^*_b$ to derive the corresponding estimated steepest ascent direction and optimum.

Note: Instead of distribution-free bootstrapping we can apply parametric bootstrapping, which assumes a specific type of distribution; e.g., a Gaussian distribution (also see Section 6). Parametric bootstrapping may be attractive if m_i is small and no CRN are used; e.g., the n means and n variances can be estimated if the weak condition $m_i > 1$ holds. If CRN are used, then the covariance matrix $cov(w_r, w_{r'})$ with r, r' = 1, ..., m needs to be estimated; this estimation requires $m > n$, as proven by Dykstra (1970). So parametric bootstrapping may require fewer replications, but it may violate the assumed distribution for the simulation outputs.

Chang et al. (2013) develop STRONG, which is a completely automated variant of RSM combined with so-called *trust regions*. They prove that STRONG converges to the true optimum. Originally, trust regions were developed by Conn et al. (2000) for deterministic nonlinear optimization. The trust region is a subregion in which the objective function is approxi-

mated; if an adequate approximation is found within the trust region, then the region is expanded; else the region is contracted. In STRONG these trust regions replace the "local" regions of classic RSM. Chang et al. derive tests to decide whether trust regions should be expanded or shrunken in the various steps of STRONG, and how much these areas should change. If necessary, the trust region is small and a second-order polynomial is used. Next, Chang et al. (2014) combine STRONG with screening, and call this STRONG-S; they apply this method to several test functions with multiple local minima. Contrary to the Taylor series argument, STRONG may have a relatively large trust region that does not require a second-order polynomial metamodel but only a first-order polynomial metamodel.

4 Adapted steepest descent (ASD)

So-called *adapted steepest descent* (ASD) accounts for the covariances between the elements of the estimated gradient $\widehat{\beta}_{-0} = (\widehat{\beta}_1, \ldots, \widehat{\beta}_k)'$ where the subscript -0 means that the intercept $\widehat{\beta}_0$ of the estimated first-order polynomial vanishes in the estimated gradient; i.e., $\widehat{\beta} = (\widehat{\beta}_0, \widehat{\beta}_{-0})'$ with $\widehat{\beta}$ defined in (3). Obviously, the white-noise assumption implies

$$\mathbf{cov}(\widehat{\beta}) = \sigma_w^2(\mathbf{Z}'\mathbf{Z})^{-1} = \sigma_w^2 \begin{pmatrix} a & \mathbf{b}' \\ \mathbf{b} & \mathbf{C} \end{pmatrix} \tag{5}$$

where

σ_w^2 denotes the variance of the 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 IID replications for combination i;

a is a scalar;

$\mathbf{b}$ is a k-dimensional vector;

$\mathbf{C}$ is a $k \times k$ matrix such that $\mathbf{cov}(\widehat{\beta}_{-0}) = \sigma_w^2\mathbf{C}$.

We estimate the variance σ_w^2 in (5) through the mean squared residuals (MSR):

$$\widehat{\sigma}_w^2 = \frac{\sum_{i=1}^{n}\sum_{r=1}^{m_i}(w_{i;r} - \widehat{y}_i)^2}{N - (k+1)} \tag{6}$$

where $\widehat{y}_i = \mathbf{z}_i'\widehat{\beta}$; also see Kleijnen (2015).

We can easily prove that the predictor variance $var(\widehat{y}|\mathbf{z})$ increases as $\mathbf{z}$—the point to be predicted—moves further away from the local area where the gradient is estimated. The point with the minimum predictor variance is $-\mathbf{C}^{-1}\mathbf{b}$ where $\mathbf{C}$ and $\mathbf{b}$ were defined below (5). ASD means that the new point to be simulated is

$$\mathbf{d} = -\mathbf{C}^{-1}\mathbf{b} - \lambda\mathbf{C}^{-1}\widehat{\beta}_{-0} \tag{7}$$

where

$-\mathbf{C}^{-1}\mathbf{b}$ is the point where the local search starts; namely, the point with minimum local variance;

λ is the step size;

$\mathbf{C}^{-1}\widehat{\beta}_{-0}$ is the classic steepest descent direction $\widehat{\beta}_{-0}$ adapted for $\mathbf{cov}(\widehat{\beta}_{-0})$.

Accounting for $\mathbf{cov}(\widehat{\beta}_{-0})$ gives a scale-independent search direction. Kleijnen et al. (2004, 2006) present experimental results showing that ASD performs "better" than classic steepest descent; i.e., the angle between the search direction based on the true β_{-0} and the search direction estimated in ASD is smaller; e.g., in one example this angle reduces from 89.87 for classic steepest descent to 1.83 for ASD.

5 Multiple responses: Generalized RSM

In practice, simulation models have *multiple* responses (multivariate output); e.g., many realistic inventory models require the inventory system to result in a minimum service rate or fill rate—because the out-of-stock costs are hard to quantify—and to minimize the sum of all the other inventory costs. Simulation software facilitates the collection of multiple outputs. There are several approaches to solve the resulting issues; see the survey by Rosen et al. (2008). The RSM literature also offers several approaches for such situations; see the surveys in Angün (2004), Khuri (1996), and Ng et al. (2007).

A novel heuristic is *Generalized RSM* (GRSM) that combines the estimated gradients of RSM with a search procedure in mathematical programming. GRSM selects one response as the goal variable and the other responses as constrained variables; moreover, the deterministic input variables may also be subjected to constraints. GRSM generalizes the steepest descent search direction through the *affine scaling search direction*, borrowing ideas from *interior point* methods (a variation on Karmarkar's algorithm) in mathematical programming; see Barnes (1986). This search direction moves faster to the optimum than steepest descent, because the former avoids creeping along the boundary of the feasible area, which is determined by the constraints on the random outputs and the deterministic inputs. 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. (2009).

Formally, GRSM extends the classic RSM problem in (1) to the following *constrained nonlinear random optimization problem*:

$$\min_{\mathbf{z}} E(w_0|\mathbf{z}) \tag{8}$$

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

$$E(w_{h'}|\mathbf{z}) \geq a_{h'} \text{ with } h' = 1, \ldots, r-1, \tag{9}$$

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

$$l_j \leq z_j \leq u_j \text{ with } j = 1, \ldots, k. \tag{10}$$

An example is an inventory simulation, in which the sum of the expected inventory carrying costs and ordering costs should be minimized while the expected service percentage should be at least (say) 90% so $a_1 = 0.9$ in (9); both the reorder quantity $z_1 = Q$ and the reorder level $z_2 = s$ should be non-negative so $z_1 \geq 0$ and $z_2 \geq 0$ in (10). Note that more complicated input constraints than (10)—namely, linear budget constraints—feature in a call-center simulation by Kelton et al. (2007); input constraints resulting from output constraints are discussed in Ng et al. (2007).

Analogously to the first steps of classic RSM based on (2), GRSM locally approximates the multivariate I/O function by r univariate first-order polynomials augmented with white noise:

$$\mathbf{y}_h = \mathbf{Z}\beta_h + e_h \text{ with } h = 0, \ldots r-1. \tag{11}$$

Like RSM, GRSM assumes that locally the white noise assumption holds for (11), so the BLUEs are the following LS estimators:

$$\hat{\beta}_h = (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{w}_h \text{ with } h = 0, \ldots, r-1. \tag{12}$$

The vector $\hat{\beta}_0$ (LS estimator of first-order polynomial approximation of goal function) and the goal function (8) result in

$$\min_{\mathbf{z}} \widehat{\beta}_{0;-0}\mathbf{z} \tag{13}$$

where $\widehat{\beta}_{0;-0} = (\widehat{\beta}_{0;1}, \ldots, \widehat{\beta}_{0,k})'$ is the LS estimate of the local gradient of the goal function. The $(r-1)$ estimates $\hat{\beta}_{h'}$ in (12) combined with the original output constraints (9) give

$$\widehat{\beta}'_{h';-0}\mathbf{z} \geq c_{h'} \text{ with } h' = 1, \ldots, r-1 \tag{14}$$

where $\widehat{\beta}_{h';-0} = (\widehat{\beta}_{h';1}, \ldots, \widehat{\beta}_{h',k})'$ is the estimated local gradient of constraint function h', and $c_{h'} = a_{h'} - \widehat{\beta}_{h';0}$ is the modified right-hand side of this constraint function. The box constraints (10) remain unchanged.

Now the k-dimensional vectors $\widehat{\beta}_{h';-0}$ $(h' = 1, \ldots, r-1)$ in (14) are collected in the $(r-1) \times k$ matrix (say) $\mathbf{B}$. Likewise, we collect the $(r-1)$ elements $c_{h'}$ in the vector $\mathbf{c}$. We define $\mathbf{l}$ as the vector with the k elements l_j, and $\mathbf{u}$ as the vector with the elements u_j. Finally, we introduce the k-dimensional vectors with the non-negative *slack variables* $\mathbf{s}$, $\mathbf{r}$, and $\mathbf{v}$, to get the following equivalent problem formulation:

$$\begin{array}{ll} \text{minimize} & \widehat{\beta}'_{0;-0}\mathbf{z} \\ \text{subject to} & \mathbf{Bz} - \mathbf{s} = \mathbf{c} \\ & \mathbf{z} + \mathbf{r} = \mathbf{u} \\ & \mathbf{z} - \mathbf{v} = \mathbf{l}. \end{array} \tag{15}$$

This optimization problem is linear in the inputs $\mathbf{z}$. GRSM uses this problem to derive the following *search direction*:

$$\mathbf{d} = -\left(\mathbf{B}'\mathbf{S}^{-2}\mathbf{B} + \mathbf{R}^{-2} + \mathbf{V}^{-2}\right)^{-1} \widehat{\beta}_{0;-0} \tag{16}$$

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 (15). Note that $\widehat{\beta}_{0;-0}$ in (16) is the estimated classic steepest ascent direction. As the value of a slack variable in (16) 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 (16) is discussed by Angün (2004).

Following the path defined by (16), GRSM must decide on the *step size* (say) λ along this path, and chooses

$$\lambda = 0.8 \min_{h'} \left[\frac{c_{h'} - \widehat{\beta}'_{h';-0}\mathbf{z}_c}{\widehat{\beta}'_{h';-0}\mathbf{d}} \right] \tag{17}$$

where the factor 0.8 is chosen to decrease the probability that the local metamodel (14) is misleading when applied globally; $\mathbf{z}_c$ denotes the current input combination, so the new combination becomes $\mathbf{z}_c + \lambda\mathbf{d}$. The box constraints (10) for the deterministic inputs hold globally, so it is easy to check whether the new combination $\mathbf{z}_c + \lambda\mathbf{d}$ violates these constraints.

Analogously to classic RSM, GRSM proceeds *stepwise*. After each step along the search path, GRSM tests the following two hypotheses, denoted by the superscripts (1) and (2):

1. Pessimistic null-hypothesis: $w_0(\mathbf{z}_c + \lambda\mathbf{d})$ (output of new combination) is *no improvement* over $w_0(\mathbf{z}_c)$ (output of old combination):

$$H_0^{(1)} : E[w_0(\mathbf{z}_c + \lambda\mathbf{d})] \geq E[w_0(\mathbf{z}_c)]. \tag{18}$$

2. This step is *feasible*; i.e., $w_{h'}(\mathbf{z}_c + \lambda\mathbf{d})$ satisfies the $(r-1)$ constraints in (9):

$$H_0^{(2)} : E[w_{h'}(\mathbf{z}_c + \lambda\mathbf{d})] \geq a_{h'} \text{with } h' = 1, \ldots, r-1. \tag{19}$$

To test these hypotheses, we may apply the following simple statistical procedures (more complicated parametric bootstrapping is used by Angün 2004, permitting non-normality and testing the relative improvement $w_0(\mathbf{z}_c + \lambda\mathbf{d})/w_0(\mathbf{z}_c)$ and the relative slacks $s_{h'}(\mathbf{z}_c + \lambda\mathbf{d})/s_{h'}(\mathbf{z}_c)$). To test (18), we apply the paired Student t_{m-1} statistic if CRN are used; we reject the hypothesis if significant improvement is observed. To test (19), we again apply a t_{m-1} statistic; because we test multiple hypotheses, we apply Bonferroni's inequality so we divide the classic α value by the number of tests $(r-1)$.

Actually, a better solution may lie somewhere 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 between these two combinations—and is still on the search path. This halving of the stepsize may be applied several times.

Next, GRSM proceeds analogously to classic RSM; i.e., around the best combination found so far, GRSM selects a new local area. Again a R-III design specifies the new simulation input combinations, and r first-order polynomials are fitted, which gives a *new* search direction, etc. Note that we might use the m replications $\widehat{\beta}_r$ to estimate the accuracy of the search direction; to test the accuracy of the estimated optimum, we present a test in the next section.

Figure 1 illustrates GRSM. This figure shows two inputs; see the two axes labeled z_1 and z_2. The goal function is to be minimized; the figure also shows two contour plots or iso-costs functions: $E(w_0) = a_{0;1}$ 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 with a 2^2 design; see the four elongated points. Together with the replications that are not shown, the I/O data 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 used to select four points to be simulated in this local area; this point is one of the four points. This design gives a new search direction, which indeed 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 in the local area with this point as one of its points. A new search direction is estimated, etc.

Angün (2004) applies GRSM to two more examples; namely, Bashyam and Fu (1998)'s inventory simulation with a service-level constraint so the solution is unknown, and a test function with known solution. Her results for these examples are encouraging, as GRSM found solutions that were both feasible and gave much lower values for the goal functions.

6 Testing an estimated optimum in GRSM: KKT conditions

Obviously, it is uncertain whether the optimum estimated by a heuristic such as GRSM is close enough to the true optimum. In *deterministic* nonlinear mathematical programming, the first-order necessary optimality conditions are known as the *Karush-Kuhn-Tucker* (KKT) conditions; see Gill et al. (2000). First we present the basic idea behind these conditions; next, we explain how to test these conditions in simulation.

Figure 2 illustrates the same type of problem as the one in Figure 1. More precisely, Figure 2 shows a goal function $E(w_0)$ with the three contour plots that correspond with the values 66, 76, and 96; also see (8). The figure also shows two constrained simulation outputs, namely $E(w_1) \geq 4$ and $E(w_2) \geq$

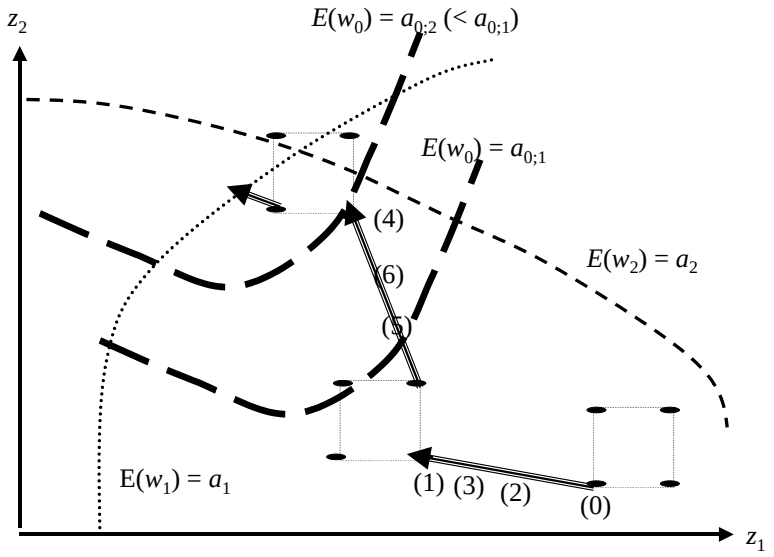

Fig. 1 GRSM illustration with two inputs, two contour plots for the goal output, two constraints for the other outputs, three local areas, three search directions, and six steps in these directions

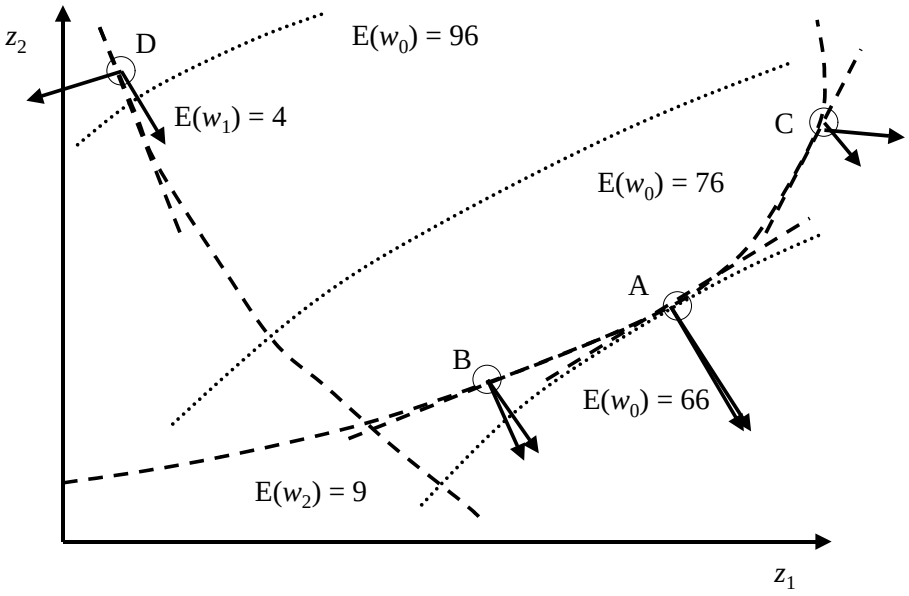

Fig. 2 A constrained nonlinear random optimization problem: three contour plots with goal values 66, 76, and 96; two other outputs with lower bounds 4 and 9; . optimal point A; points B and C on bound 9; point D on bound 4; local gradients at A through D for goal function and binding constraint, perpendicular to local tangent lines for binding constraint

9; also see (9). 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. The points B and C lie on the boundary $E(w_2) = 9$; point D lies on the boundary $E(w_1) = 4$. Obviously, the points A and D lie far away from each other. The figure also displays the local gradients at these four points for the goal function and for the *binding constraint*; i.e., the constraint with a zero slack value in (9). 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

$$\begin{array}{l} \beta_{0;-0} = \sum\limits_{h \in A(\mathbf{z}^0)} \lambda_h^0 \beta_{h;-0} \\ \lambda_h^0 \geq 0 \\ h \in A\left(\mathbf{z}^0\right) \end{array} \tag{20}$$

where $\beta_{0;-0}$ denotes the k-dimensional vector with the gradient of the goal function, as we saw in (13); $A\left(\mathbf{z}^0\right)$ is the index set with the indices of the constraints that are binding at $\mathbf{z}^0$; λ_h^0 is the Lagrangian multiplier for binding constraint h; $\beta_{h;-0}$ is the gradient of the output in that binding constraint; Figure 2 has only one binding constraint at the points A through D. Altogether, (20) implies that the gradient of the objective is a nonnegative linear combination of the gradients of the binding constraints, at $\mathbf{z}^0$. Figure 2 shows that A satisfies (20); B has two gradients that point in different but similar directions—and so does C—whereas D has two gradients that point in completely different directions.

Note: If the optimum occurs *inside* the feasible area, then there are no binding constraints so the KKT conditions reduce to the condition that the goal gradient is zero. Classic RSM includes tests for a zero gradient estimated from a second-order polynomial; see Section 2.

In simulation we must *estimate* the gradients; moreover, to check which constraints are binding, we must estimate the slacks of the constraints. This estimation changes the KKT conditions into a problem of nonlinear statistics. Angün (2004) presents an asymptotic test, but Bettonvil et al. (2009) derive a small-sample bootstrap test. We present the latter test, because it is simpler and it allows expensive simulation. Readers not interested in the technical details of this bootstrapping should skip the remainder of this section.

As in classic RSM, we assume *locally constant (co)variances* for each of the r simulation outputs (when moving to a new local area, the (co)variances may change; e.g., the points A through D in Figure 2 do not have the same variance for the goal output). LS per univariate simulation output gives $\hat{\beta}_h$ $(h = 0, 1, \ldots, r-1)$ defined in (12). These estimators have the following estimated covariance matrix:

$$\widehat{\mathbf{cov}}(\hat{\beta}_h, \hat{\beta}_{h'}) = \widehat{\mathbf{cov}}(w_h, w_{h'}) \otimes (\mathbf{Z}'\mathbf{Z})^{-1} \ (h, h' = 0, \ldots, r-1) \tag{21}$$

where $\otimes$ denotes the Kronecker product 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 replications at the local center:

$$\widehat{\mathbf{cov}}(w_h, w_{h'}) = (\widehat{\sigma}_{h;h'}) = (\sum_{l=1}^{m} (w_{h;l} - \overline{w}_h)(w_{h';l} - \overline{w}_{h'})) \frac{1}{m-1}. \quad (22)$$

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); this matrix is 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 $(\mathbf{Z}'\mathbf{Z})^{-1}$ (e.g., $\mathbf{Z}$ is an $N \times (1+k)$ matrix in equation 5). The matrix $\widehat{\mathbf{cov}}(w_h, w_{h'})$ is singular if $m \leq r$; e.g., the case study in Kleijnen (1993) has $r = 2$ response types and $k = 14$ inputs so $m \geq 3$ replications of the center point are required. Of course, the higher m is, the higher is the power of the tests that use these replications. Bettonvil et al. (2009) do not consider cases where all n local points are replicated and CRN are possibly used; these cases require further research.

In classic RSM we assume that the output is Gaussian, and now we assume that the r-variate simulation output is *multivariate Gaussian*. We use the center point to test whether a constraint is binding in the current local area, because this point is more representative of the local behavior than the points of the R-III design. (To save simulation runs, we should start a local experiment at its center point including replications; if it turns out that either no constraint is binding or at least one constraint is violated, then we need not test the other hypotheses so we do not simulate the remainder of the local design.) Actually, we test the following three null-hypotheses, denoted by the superscripts (1) through (3):

1. The current solution is feasible and at least one constraint is binding; see (9):

$$H_0^{(1)} : E(w_{h'} | \mathbf{d} = \mathbf{0}) = a_{h'} \text{ with } h' = 1, \ldots, r-1 \quad (23)$$

   where $\mathbf{d} = \mathbf{0}$ corresponds with the center of the local area expressed in the 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 (20) we replace the deterministic quantities by their random estimators:

$$H_0^{(2)} : E(\widehat{\beta}_{0;-0}) = E\left(\sum_{h \in A(\mathbf{z}^0)} \widehat{\lambda}_h^0 \widehat{\beta}_h \right). \quad (24)$$

3. The Lagrangian multipliers in (24) are non-negative:

$$H_0^{(3)} : E(\widehat{\lambda}) \geq \mathbf{0}. \quad (25)$$

Each of these three hypotheses requires multiple tests, so we apply Bonferroni's inequality. Moreover, we test these three hypotheses sequentially, so it is hard to control the final type-I and type-II error probabilities (classic RSM has the same type of problem, but has nevertheless acquired a track record in practice).

Sub 1: To test (23), we use the classic t statistic:

$$t_{m-1}^{(h')} = \frac{\overline{w}_{h'}(\mathbf{d}=\mathbf{0}) - a_{h'}}{\sqrt{\widehat{\sigma}_{h';h'}/m}} \text{with } h' = 1, \ldots, r-1 \tag{26}$$

where both the numerator and the denominator use the m replications at the local center point; see (22). This t statistic may give the following three results:

(i) 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 5).

(ii) 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.

(iii) 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 $A\left(\mathbf{z}^0\right)$; see (24). And the KKT test proceeds as follows.

Sub 2 and 3: To estimate the *linear* combination in (24), we apply LS with as explanatory variables the estimated gradients of the (say) J binding constraints; these explanatory variables are now random. We collect these gradients in the $k \times J$ matrix $\widehat{\mathbf{B}}_{J;-0}$. The parameters estimated through LS are $\widehat{\lambda}$. Let $\widehat{\widehat{\beta}}_{0;-0}$ denote the LS estimator of the goal gradient:

$$\widehat{\widehat{\beta}}_{0;-0} = \widehat{\mathbf{B}}_{J;-0}(\widehat{\mathbf{B}}'_{J;-0}\widehat{\mathbf{B}}_{J;-0})^{-1}\widehat{\mathbf{B}}'_{J;-0}\widehat{\beta}_{0;-0} = \widehat{\mathbf{B}}_{J;-0}\widehat{\lambda} \tag{27}$$

with $\widehat{\lambda} = (\widehat{\mathbf{B}}'_{J;-0}\widehat{\mathbf{B}}_{J;-0})^{-1}\widehat{\mathbf{B}}'_{J;-0}\widehat{\beta}_{0;-0}$. To quantify the *validity* of this linear approximation, we use the k-dimensional vector of its residuals

$$\widehat{\mathbf{e}}(\widehat{\widehat{\beta}}_{0;-0}) = \widehat{\widehat{\beta}}_{0;-0} - \widehat{\beta}_{0;-0}. \tag{28}$$

Hypothesis (24) implies $E(\widehat{\mathbf{e}}(\widehat{\widehat{\beta}}_{0;-0})) = \mathbf{0}$. This hypothesis involves a product of multivariates, so standard tests do not apply and we use *bootstrapping*. We do not apply distribution-free bootstrapping, because in expensive simulation only the center point is replicated a few times. Instead, we apply *parametric bootstrapping*; i.e., we assume a Gaussian distribution (like in classic RSM), and we estimate its parameters from the simulation's I/O data. This bootstrapping consists of the following four steps, where the superscript $*$ denotes a bootstrapped value:

(i) Use the *Monte Carlo* method to sample $vec(\widehat{\beta}^*_{0;-0}, \widehat{\mathbf{B}}^*_{J;-0})$, 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 $\widehat{\mathbf{B}}^*_{J;-0}$:

$$vec(\widehat{\beta}^*_{0;-0}, \widehat{\mathbf{B}}^*_{J;-0}) \sim N(vec(\widehat{\beta}_{0;-0}, \widehat{\mathbf{B}}_{J;-0}), \widehat{\mathbf{cov}}(vec(\widehat{\beta}_{0;-0}, \widehat{\mathbf{B}}_{J;-0}))) \quad (29)$$

where $\widehat{\mathbf{cov}}(vec(\widehat{\beta}_{0;-0}, \widehat{\mathbf{B}}_{J;-0}))$ is the $(k + kJ) \times (k + kJ)$ matrix computed through (21).

(ii) Use the bootstrap values resulting from Step 1, to compute the *LS* estimate of the bootstrapped goal gradient using the bootstrapped gradients of the binding constraints as explanatory variables; i.e., use (27) adding the superscript $*$ to all random variables resulting in $\widehat{\widehat{\beta}}^*_{0;-0}$ and $\widehat{\lambda}^*$.

(iii) Use $\widehat{\widehat{\beta}}^*_{0;-0}$ from Step (ii) and $\widehat{\beta}^*_{0;-0}$ from Step (i) to compute the *bootstrap residual* $\widehat{\mathbf{e}}(\widehat{\widehat{\beta}}^*_{0;-0}) = \widehat{\widehat{\beta}}^*_{0;-0} - \widehat{\beta}^*_{0;-0}$, analogous to (28). Determine whether any of the *bootstrapped Lagrangian multipliers* $\widehat{\lambda}^*$ found in Step (ii) is negative; i.e., augment a counter (say) c^* with the value 1 if this event occurs.

(iv) *Repeat* the preceding three steps (say) 1000 times. This bootstrap sample gives the EDF of $\widehat{\mathbf{e}}(\widehat{\widehat{\beta}}^*_{0;-0;j})$—which denotes the bootstrapped residuals per input j $(j = 1, \ldots, k)$—and the final value of the counter c^*. Reject the hypothesis in (24) if this EDF implies a two-sided $(1 - \alpha/(2k))$ confidence interval that does not cover the value 0, where the factor k is explained by Bonferroni's inequality. Reject (25) 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, we approximate the binomial distribution 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. (2009) give encouraging results; i.e., the classic t test for zero slacks performs as expected and the new bootstrap tests give observed type-I error rates close to the prespecified (nominal) rates, while the type-II error rate decreases as the tested input combination is farther away from the true optimum (see A through D in Figure 2).

7 Robust optimization

Taguchi emphasizes that in practice some inputs of a product are under complete control, whereas other inputs are not; e.g., the design of a car engine is completely controlled by the engineers, but the driving style is not. Consequently, a design that allows some flexibility in its use is "better"; e.g., a car optimized only for the race circuit does not perform well on

the highways. Likewise, in simulation the optimum solution may be completely wrong when ignoring uncertainties in some inputs; e.g., the nominally optimal decision on the inventory control limits s (reorder level) and S (order-up-to level) may be completely wrong, because this solution ignores the uncertainty in the parameters assumed for the random demand and delivery time distributions. In Section 7.1, we first explain Taguchi's approach, updating and extending Dellino et al. (2010); also see Dellino et al. (2012). In Section 7.2, we briefly discuss Ben-Tal's approach, summarizing Yanıkoğlu et al. (2013).

7.1 Taguchi's robust optimization

We use Taguchi's view of the world, distinguishing between two types of inputs:

(i) decision variables, denoted by (say) d_j $(j = 1, \ldots, k)$ so $\mathbf{d} = (d_1, ..., d_k)'$, and

(ii) environmental or noise factors e_g $(g = 1, \ldots, c)$ so $\mathbf{e} = (e_1, ..., e_c)'$.

Taguchi assumes a single output, which we denote by w. He focuses on the mean μ_w and the variance σ_w^2 of this output, caused by the noise factors $\mathbf{e}$. However, we do not use Taguchi's scalar *loss function* such as the signal-to-noise or mean-to-variance ratio μ_w/σ_w^2; see Myers et al. (2009, pp. 486-488). Instead we use both μ_w and σ_w^2 to characterize the statistical distribution of the output, and we try to solve the following problem:

$$\min \mu_w \text{ such that } \sigma_w^2 \leq T \tag{30}$$

where T is some threshold; also see Myers et al. (2009, pp. 488-495). We also refer to the surveys on robust optimization by Beyer and Sendhoff (2007) and Park et al. (2006).

Taguchi's worldview has been very successful in production engineering, but statisticians have seriously criticized his statistical techniques; see the panel report by Nair (1992). Therefore Myers et al. (2009, pp. 502-506) combine Taguchi's worldview with RSM. We adapt their RSM; i.e., we assume that $\mathbf{e}$ has the mean $\boldsymbol{\mu}_\mathbf{e}$ and the covariance matrix $\boldsymbol{\Omega}_\mathbf{e}$, whereas Myers et al. assume a constant variance $\boldsymbol{\sigma}_e^2$ so $\boldsymbol{\Omega}_\mathbf{e} = \boldsymbol{\sigma}_e^2\mathbf{I}$. To find a robust solution, Myers et al. superimpose contour plots for the mean and variance of the output, whereas we use more general and flexible mathematical programming. This mathematical programming, however, requires specification of threshold values like T in (30); managers may find it hard to select specific values, so we may try different values and estimate the corresponding Pareto-optimal efficiency frontier. To estimate the variability of this Pareto frontier resulting from the various estimators, we may use bootstrapping (also see our bootstrapping in the preceding section). For details on our adaptation of Meyer et al.'s approach we refer to Dellino et al. (2010).

More precisely, Myers et al. fit a *second-order polynomial* for $\mathbf{d}$ that is to be optimized. To model possible effects of $\mathbf{e}$, they fit a first-order polynomial in $\mathbf{e}$. They also estimate "control-by-noise" two-factor interactions.

Altogether, they fit the following "incomplete" second-order polynomial:

$$\begin{aligned} y &= \beta_0 + \sum_{j=1}^{k} \beta_j d_j + \sum_{j=1}^{k} \sum_{j'=j}^{k} \beta_{j;j'} d_j d_{j'} + \sum_{g=1}^{c} \gamma_g e_g + \sum_{j=1}^{k} \sum_{g=1}^{c} \delta_{j;g} d_j e_g + \epsilon \\ &= \beta_0 + \boldsymbol{\beta}'\mathbf{d} + \mathbf{d}'\mathbf{B}\mathbf{d} + \boldsymbol{\gamma}'\mathbf{e} + \mathbf{d}'\boldsymbol{\Delta}\mathbf{e} + \epsilon\,, \end{aligned} \tag{31}$$

where we now use the symbol ϵ (instead of e) to denote the regression residual ϵ with $\mu_\epsilon = 0$ if this metamodel has no lack-of-fit and with constant variance σ_ϵ^2; furthermore, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_k)'$, $\mathbf{B}$ denotes the $k \times k$ symmetric matrix with main-diagonal elements $\beta_{j;j}$ and off-diagonal elements $\beta_{j;j'}/2$, $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_c)'$, and $\boldsymbol{\Delta}$ denotes the $k \times c$ matrix with interactions $\delta_{j;g}$.

Obviously, (31) implies the following regression predictor for the true mean μ_w:

$$\mu_y = \beta_0 + \boldsymbol{\beta}'\mathbf{d} + \mathbf{d}'\mathbf{B}\mathbf{d} + \boldsymbol{\gamma}'\boldsymbol{\mu}_\mathbf{e} + \mathbf{d}'\boldsymbol{\Delta}\boldsymbol{\mu}_\mathbf{e}. \tag{32}$$

And the regression predictor for the true variance σ_w^2 is

$$\sigma_y^2 = (\boldsymbol{\gamma}' + \mathbf{d}'\boldsymbol{\Delta})\boldsymbol{\Omega}_\mathbf{e}(\boldsymbol{\gamma} + \boldsymbol{\Delta}'\mathbf{d}) + \sigma_\epsilon^2 = \mathbf{l}'\boldsymbol{\Omega}_\mathbf{e}\mathbf{l} + \sigma_\epsilon^2 \tag{33}$$

where $\mathbf{l} = (\boldsymbol{\gamma} + \boldsymbol{\Delta}'\mathbf{d}) = (\partial y/\partial e_1, \ldots, \partial y/\partial e_c)'$ so $\mathbf{l}$ is the gradient with respect to $\mathbf{e}$. So, the larger the gradient's elements are, the larger σ_y^2 is—which stands to reason. Furthermore, if there are no control-by-noise interactions so $\boldsymbol{\Delta} = \mathbf{0}$, then we cannot control σ_y^2 through $\mathbf{d}$.

To enable estimation of the regression parameters in (31), we use a *crossed design*; i.e., we combine the design for $\mathbf{d}$ and the design for $\mathbf{e}$—as is usual in a Taguchian approach. To estimate the optimal $\mathbf{d}$, we use a CCD; see the discussion below (4). For the first-order polynomial in $\mathbf{e}$, we use a R-III design; see the example in Table 1. The combination of these two designs obviously enables the estimation of the two-factor interactions $\delta_{j;g}$. Note that Myers et al. (2009) and Dellino et al. (2010) also discuss designs that are more efficient than crossed designs.

To use *linear regression analysis* for the estimation of the q parameters in (say) $\boldsymbol{\zeta} = (\beta_0, ..., \delta_{k;c})'$ in (31), we reformulate (31) as

$$y = \boldsymbol{\zeta}'\mathbf{x} + \epsilon \tag{34}$$

where $\mathbf{x}$ is defined in the obvious way; e.g., the element corresponding with $\beta_{1;2}$ (interaction between d_1 and d_2) is $d_1 d_2$. Note that (34) is linear in $\boldsymbol{\zeta}$, but (31) is not linear in $\mathbf{d}$. Then (34) gives the LS estimator

$$\widehat{\boldsymbol{\zeta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{w} \tag{35}$$

where $\mathbf{X}$ is the $N \times q$ matrix of explanatory variables with $N = \sum_{i=1}^{n} m_i$ and n denoting the number of different simulated combinations of $\mathbf{d}$ and $\mathbf{e}$; $\mathbf{w}$ consists of the N simulation outputs. Obviously, the covariance matrix of $\widehat{\boldsymbol{\zeta}}$ is

$$\mathbf{cov}(\widehat{\boldsymbol{\zeta}}) = (\mathbf{X}'\mathbf{X})^{-1}\sigma_w^2\,. \tag{36}$$

The RSM metamodel (31) implies that σ_w^2 equals σ_ϵ^2. This variance is estimated by

$$MSR = \frac{(\widehat{\mathbf{y}} - \mathbf{w})'(\widehat{\mathbf{y}} - \mathbf{w})}{N - q} \tag{37}$$

where $\widehat{\mathbf{y}} = \widehat{\boldsymbol{\zeta}}' \mathbf{x}$; also see (6).

To estimate μ_y, we simply plug $\widehat{\zeta}$ defined by (35) into the right-hand side of (32), where $\mathbf{d}$ and $\boldsymbol{\mu}_\mathbf{e}$ are known. To estimate σ_y^2, we also plug $\widehat{\zeta}$ into (33), where $\boldsymbol{\Omega}_\mathbf{e}$ is known. Note that (33) involves products of unknown parameters, so it implies a *nonlinear* estimator $\widehat{\sigma}_y^2$; plugged-in estimators certainly create bias, but we ignore this bias.

Our final goal is to solve (30). We solve this constrained minimization problem through a mathematical programming solver; e.g., Matlab's "fmincon"—but a different solver might be used; see Gill et al. (2000). This gives estimates of the robust decision variables and the corresponding mean and variance.

Dellino et al. (2010) give the example of the *economic order quantity* (EOQ) for an environment with a demand rate that is uncertain but has a known distribution. This example demonstrates that if management prefers low variability of inventory costs, then they must pay a price; i.e., the expected costs increases. Furthermore, the classic EOQ assuming a known fixed demand rate and the robust EOQ do differ. More examples are referenced by Yanıkoğlu et al. (2013).

7.2 Ben-Tal's robust optimization

Ben-Tal emphasizes that the nominal solution—which ignores the uncertainty in $\mathbf{e}$—may easily violate the constraints in the given mathematical programming problem, so he derives a robust solution that gives a slightly worse value for the goal variable but increases the probability of satisfying the constraints. The mathematical challenge is to develop a computationally tractable "robust counterpart" of the original mathematical programming problem. Therefore he proposes a robust solution that is "immune" to variations within the *uncertainty set*. Yanıkoğlu et al. (2013) derive a specific uncertainty set for $\mathbf{p}$ where $\mathbf{p}$ denotes the unknown density function of $\mathbf{e}$, which is compatible with the historical data on $\mathbf{e}$. Technically, $\mathbf{p}$ belongs to this set with confidence $1 - \alpha$, assuming some phi-divergence measure such as the well-known chi-square distance. In this chapter we do not present the mathematical details of the derivation of tractable robust counterparts, but refer to Yanıkoğlu et al.

Note that Taguchians assume a specific distribution for $\mathbf{e}$, which implies $\boldsymbol{\mu}_\mathbf{e}$ and $\boldsymbol{\Omega}_\mathbf{e}$ in (32) and (33). This distribution may be estimated from historical data; Yanıkoğlu et al., however, develop an approach that uses only the original observed data on $\mathbf{e}$. Several numerical examples demonstrate the effectiveness of this novel combination of Taguchi's and Ben-Tal's approaches.

Yanıkoğlu et al. also present Myers et al. (2009, p. 512)'s real-life television example, and Shi (2011)'s simulated distribution-center example. In this chapter we focus on simulation, so we discuss only Shi's example. He has five decision variables (e.g., number of forklifts) and two environmental variables (e.g., delay probabilities of suppliers); the response is the total throughput. Shi fits an incomplete second-order polynomial; see (31). Yanıkoğlu et al. replace (30) by the following related problem:

$$\min \sigma_w^2 \text{ such that } \mu_w \leq T. \tag{38}$$

These two examples again demonstrate that in general robust solutions have better worst-case performance and also better average performance.

8 Conclusions

In this chapter, we started with the basics of classic RSM, which minimizes the expected value of a single response variable in real-life experiments. Next we considered simulation experiments. We then added the adapted steepest descent search direction, which improves classic steepest descent. We also summarized GRSM for simulation with multivariate responses, assuming that one response is to be minimized while all the other responses and deterministic inputs must meet given constraints. Furthermore, we presented a bootstrap procedure for testing whether the KKT conditions hold for the estimated optimum. Finally, we considered robust optimization.

Future research may study the selection of the required number of replications, and the use of replications to estimate the accuracy of the resulting estimated search direction or optimum. Bootstrapping may solve this problem, but it needs more research. Numerical evaluation of the adapted steepest descent method would benefit from more applications in practice. There is also a need for more research on the KKT testing procedure when all local points (not only the center) are replicated and CRN are used; more practical applications are also needed. In Taguchian robust optimization through RSM we may vary the threshold values, which gives different optimal solutions and a corresponding Pareto frontier; bootstrapping this frontier might enable management to make the final compromise decision—but more research and applications are needed.

References

Angün, M.E. (2004), *Black box simulation optimization: generalized response surface methodology*. CentER Dissertation series, Tilburg University, Tilburg, the Netherlands

Angün, E., D. den Hertog, G. Gürkan, and J.P.C. Kleijnen (2009), Response surface methodology with stochastic constraints for expensive simulation. Journal of the Operational Research Society, 60, (6), 735–746

Ankenman, B., Nelson, B., and Staum, J., (2010), Stochastic Kriging for Simulation Metamodeling, Operations Research, 58, No. 2, pp. 371-382

Barnes, E. R. (1986), A variation on Karmarkar's algorithm for solving linear programming problems. *Mathematical Programming*, 36, pp. 174-182

Barton, R.R. and M. Meckesheimer (2006), Metamodel-based simulation optimization. *Handbooks in Operations Research and Management Science, Volume 13*, Elsevier/North Holland, pp. 535-574

Bartz-Beielstein, T. (2006), *Experimental research in evolutionary computation; the new experimentalism*. Springer, Berlin

Bashyam, S. and M. C. Fu (1998), Optimization of (s, S) inventory systems with random lead times and a service level constraint. *Management Science*, 44, pp. 243-256

Ben-Tal, A. and A. Nemirovski (1998), Robust convex optimization. Math. Oper. Res. 23(4) 769-805

Ben-Tal, A. and A. Nemirovski (2008), Selected topics in robust convex optimization, *Mathematical Programming*, 112, no. 1, pp. 125–158

Bettonvil, B.W.M., E. del Castillo, and J.P.C. Kleijnen (2009). Statistical testing of optimality conditions in multiresponse simulation-based optimization. European Journal of Operational Research, 199(2), 448-458.

Beyer, H. and B. Sendhoff (2007), Robust optimization—a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering,* 196, nos. 33-34, pp. 3190–3218

Box, G.E.P. and K.B. Wilson (1951), On the experimental attainment of optimum conditions. *Journal Royal Statistical Society, Series B*, 13, no. 1, pp. 1-38

Chang, K-H., J. Hong, and H. Wan (2013), Stochastic Trust-Region Response-Surface Method (STRONG) – A New Response-Surface Framework for Simulation Optimization, *INFORMS Journal on Computing*, 25, no. 2, pp. 230-243

Chang, K.-H. M-K Li and H. Wan (2014), Combining STRONG with Screening Designs for Large-Scale Simulation Optimization, *IIE Transactions*, accepted.

Chih, M. (2013), A more accurate second-order polynomial metamodel using a pseudo-random number assignment strategy. Journal of the Operational Research Society, 64, pp. 198-207

Conn, A.R., N.L.M. Gould, P.L. Toint (2000), Trust-Region Methods. SIAM

Dellino, G., Kleijnen, J.P.C., & Meloni, C. (2010). Robust optimization in simulation: Taguchi and Response Surface Methodology. International Journal of Production Economics, 125(1), 52-59

Dellino, G., Kleijnen, J.P.C., & Meloni, C. (2012). Robust optimization in simulation: Taguchi and Krige combined. Informs Journal on Computing, 24(3), 471-484

Dykstra, R.L. (1970), Establishing the positive definiteness of the sample covariance matrix. The Annals of Mathematical Statistics, 41, no. 6, pp. 2153–2154

Efron, B. and R.J. Tibshirani (1993), *An introduction to the bootstrap.* Chapman & Hall, New York

Gill, P. E., W. Murray, and M. H. Wright (2000). *Practical optimization, 12th edition.* Academic Press, London

Kelton, W.D., R.P. Sadowski, and D.T. Sturrock (2007), *Simulation with Arena; fourth edition.* McGraw-Hill, Boston

Khuri, A. I. (1996), Multiresponse surface methodology. *Handbook of Statistics, volume 13*, edited by S. Ghosh and C. R. Rao, Elsevier, Amsterdam

Kleijnen, J.P.C. (1975). Statistical techniques in simulation, Part II. New York: Dekker

Kleijnen, J.P.C. (1993), Simulation and optimization in production planning: a case study. *Decision Support Systems*, 9, pp. 269-280

Kleijnen, J.P.C. (2008). Response surface methodology for constrained simulation optimization: An overview. Simulation Modelling Practice and Theory, 16, 50-64

Kleijnen, J.P.C. (2015), Design and Analysis of Simulation Experiments; Second Edition, Springer

Kleijnen, J.P.C., D. den Hertog, and E. Angün (2004), Response surface methodology's steepest ascent and step size revisited. *European Journal of Operational Research*, 159, pp. 121-131

Kleijnen, J.P.C., D. den Hertog and E. Angün (2006), Response surface methodology's steepest ascent and step size revisited: correction. *European Journal of Operational Research*, 170, pp. 664-666

Law, A.M. (2014), *Simulation modeling and analysis; fifth edition.* McGraw-Hill, Boston

Myers, R.H., A.I. Khuri, and W.H. Carter (1989), Response surface methodology: 1966-1988. *Technometrics*, 31, no., 2, pp. 137-157

Myers, R.H., D.C. Montgomery, and C.M. Anderson-Cook (2009), *Response surface methodology: process and product optimization using designed experiments; third edition.* Wiley, New York

Nair, V.N., editor (1992), Taguchi's parameter design: a panel discussion. *Technometrics*, 34, no. 2, pp. 127-161

Ng S.H., Xu K. and Wong W.K. (2007), Optimization of multiple response surfaces with secondary constraints for improving a radiography inspection process. Quality Engineering, 19, no. 1, 53-65

Park, G-J., T-H. Lee, K.H. Lee, and K-H. Hwang (2006), Robust design: an overview. *AIAA Journal*, 44, no. 1, pp. 181–191

Qu, H. and M.C. Fu (2012), ON DIRECT GRADIENT ENHANCED SIMULATION METAMODELS. Proceedings of the 2012 Winter Simulation Conference, C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

Rikards, R. and J. Auzins (2002), Response surface method for solution of structural identification problems. *Fourth International Conference on Inverse Problems in Engineering*, Rio de Janeiro, Brazil

Rosen, S.C., C.M. Harmonosky, and M.T. Traband (2008), Optimization of systems with multiple performance measures via simulation: survey and recommendations. *Computers & Industrial Engineering*, 54, no. 2, pp. 327-339

Shi, W. (2011), Design of pre-enhanced cross-docking distribution center under supply uncertainty: RSM robust optimization method. Working Paper, Huazhong University of Science & Technology, China

Shi, W., Kleijnen, J.P.C., & Liu, Z. (2014). Factor Screening For Simulation With Multiple Responses: Sequential Bifurcation. European Journal of Operational Research, accepted

Taguchi, G. (1987), *System of experimental designs, volumes 1 and 2.* UNIPUB/ Krauss International, White Plains, New York

Van den Bogaard, W. and Kleijnen, J.P.C. (1977). Minimizing waiting times using priority classes: A case study in response surface methodology. Discussion Paper FEW 77.056 (see

http://publications.uvt.nl/repository/kleijnen/publications.html)

Yanikoglu, I., D. den Hertog, J.P.C. Kleijnen (2013), Adjustable Robust Parameter Design with Unknown Distributions. CentER Discussion Paper

Acknowledgment: I thank Michael Fu for inviting me to write a chapter on RSM for this handbook and for his many comments on earlier versions.