COMMUN. STATIST.-THEORY METH., 20(2), 477-495 (1991)

A SYSTEM FOR QUALITY IMPROVEMENT VIA COMPUTER EXPERIMENTS

William J. Welch

Department of Statistics
and Actuarial Science
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada

Jerome Sacks

Department of Statistics
University of Illinois
Champaign, Illinois 61820

Key Words and Phrases: circuit design; computer code; off-line quality control; parameter design; response surface; stochastic process, Taguchi method.

ABSTRACT

Many products are now routinely designed with the aid of computer models. Given the inputs—designable engineering parameters and parameters representing manufacturing-process conditions—the model generates the product's quality characteristics. The quality improvement problem is to choose the designable engineering parameters such that the quality characteristics are uniformly good in the presence of variability in processing conditions. This article summarizes recent work to develop an efficient, systematic approach to quality improvement via such computer models. Using relatively few runs of the computationally expensive computer model, our approach builds approximating functions to be used during product-design optimization. We contrast several approximation strategies. We also discuss how to choose a loss to optimize when there are multiple, conflicting quality characteristics. Applications in the design of electronic circuits are given.

1. INTRODUCTION

Taguchi's methods for improving products and processes (Taguchi and Wu 1979; Taguchi 1986) have created considerable interest in the statistical design of experiments as applied to engineering design. His parameter design methodology searches for levels of easy to control engineering parameters that make the product or process perform well in the presence of variability from uncontrollable (or expensive to control) factors. Thus, quality is improved by designing quality "into" the product or process, rather than by on-line inspection. Taguchi's objective of finding such robust engineering designs has been widely accepted, yet his statistical techniques for reaching this goal have not won such universal acclaim.

This article is concerned with techniques for parameter design via computer experiments. Many products are now routinely designed with the aid of computer models. These models or "codes" complement or sometimes largely replace physical experiments, so reducing the cost of experimentation and, perhaps more importantly, speeding up product development. The design of electronic circuits is one important area where computer models are in widespread use. Finite element analysis of mechanical components is another increasingly common application area.

Although computer simulation is invariably less expensive than physical experimentation, these codes can be computationally demanding. Taguchi's crossing of inner and outer arrays in parameter design typically requires a large experimental plan. (We shall refer to experimental *plans* rather than experimental *designs* to avoid confusion with the engineering design procedure.) In his example of designing a Wheatstone bridge circuit via a mathematical function (Taguchi 1986, Chapter 6), crossing two 36-run experimental plans leads to 1296 runs. Of course, the mathematical model used is trivial, and a run has virtually zero cost, but this just raises the question, "Why not simply plug the ultimate objective function into a numerical optimizer?" A more realistic, though still modest, circuit simulation problem might take several minutes on a workstation for one run of the simulator: 1296 runs at 2 minutes

per run equals 43 hours of computing time. Finite-element codes are usually much more expensive. Somewhat smaller crossed-array plans are no doubt possible, but the crossing of arrays almost ensures that a fairly large, possibly prohibitive, number of runs will be required.

Even though some input parameters may represent factors that vary randomly in manufacturing, computer models of this type are typically deterministic. Running the code again with the same inputs will produce the same outputs. Given the absence of random error, it is not obvious that methods for planning and analyzing physical experiments are relevant to computer experiments. Nonetheless, the distinction is often ignored.

This article summarizes recent work by us and co-workers on parameter design via computer experiments: Bernardo, Sacks, Welch, Liu, and Nazaret (1989); Buck, Sacks, Welch, Liu, and Nazaret (1989); Welch, Yu, Kang, and Sacks (1990); and Yu, Kang, Sacks, and Welch (1989). The methods we use require relatively few runs of the simulation code. They can also incorporate experimental planning and analysis strategies developed specifically for deterministic computer experiments (Sacks, Welch, Mitchell, and Wynn, 1989). The common theme is that we approximate the relationships between the quality characteristics (computer-model outputs) of interest and the code's input parameters. In circuit-design applications, the outputs might be the circuit's bandwidth and gain. Typical inputs include designable transistor dimensions and various parameters to represent processing variation. In manufacturing, the process-variation parameters are random, but they can be manipulated at will during the simulation experiment. Thus, at this stage the distinction between controllable and uncontrollable inputs is ignored. A single plan for both types of inputs typically reduces the number of runs relative to crossed-array approaches. Then, we fit approximating functions to the data from the experiment. These approximations predict each output as a function of all input parameters. They act as computationally cheap surrogates for the simulation code during engineering design optimization.

We have developed these techniques largely in response to electrical-engineering applications like those described throughout this article, but the

approach seems to be quite general. [We should also mention that various quality improvement methods have been developed in the electrical-engineering literature; see Brayton, Hachtel, and Sangiovanni-Vincentelli (1981) for instance.]

After formulating the problem more rigorously in Section 2, we will give a fuller account of our approach in Section 3. Modeling the computer code in order to replace it by an approximating function is central to this approach; in Section 4 we contrast some modeling strategies. Section 5 discusses experimental planning of the runs used for model fitting. Section 6 deals with various possible objective functions at the engineering-design optimization stage. In particular, there are practical difficulties in minimizing squared-error loss (or maximizing Taguchi's signal-to-noise ratios) when there are multiple, conflicting quality characteristics. Finally, Section 7 makes some concluding comments.

2. FORMULATION OF THE PARAMETER DESIGN PROBLEM

Let $y_1, \ldots, y_q$ denote the q critical quality characteristics produced by the simulator. Each y_k is a function of d input parameters, the d-dimensional vector $x = (x_1, \ldots, x_d)$, all other inputs to the simulator being fixed. In simulation, an input parameter may be varied to represent *controllable* (designable) adjustment and/or *uncontrollable* (random) variation. To distinguish the controllable and uncontrollable components, we write $x_j = c_j + u_j$. A parameter with no controllable adjustment has a fixed c_j, which can be ignored. Similarly, if there is no random variation, then $u_j = 0$.

Ideally, we would be able to specify a loss function $l(y_1, \ldots, y_q)$ that measures the economic loss attached to a product with characteristics $y_1, \ldots, y_q$. For fixed values of $c = (c_1, \ldots, c_d)$, we can define an expected loss with respect to the distribution f of $u = (u_1, \ldots, u_d)$:

$$L(c) = \int l[y_1(c+u), \ldots, y_q(c+u)] f(u)\, du. \tag{1}$$

This expected loss is then minimized as a function of c. For example, if there is a single quality characteristic y with target T, squared-error loss leads to minimization of $\int [y(c+u) - T]^2 f(u)\, du$.

There are obvious practical difficulties in specifying the loss function $l(y_1, \ldots, y_q)$ and the distribution $f(u)$, and any method will have to face these issues. Once the expected loss is specified, the problem of minimizing $L(c)$ in (1) is one of numerical optimization because $y_1, \ldots, y_q$ are deterministic functions. They are commonly solutions to differential equations and are usually very expensive to solve. Plugging $L(c)$ into an optimizer is likely to require a prohibitive number of function evaluations. Our approach, at least formally, takes the above formulation, but optimizes through inexpensive predictors of $y_k(x)$.

3. A SYSTEM FOR QUALITY IMPROVEMENT

Bernardo et al. (1989), Buck et al. (1989), Welch et al. (1990), and Yu et al. (1989) discussed strategies for optimization of electronic-circuit designs via computer experiments. There are some differences amongst these methods, which will be elaborated on in Sections 4 and 6, but there is an underlying common theme. Based on a sequence of experiments with relatively few runs of the simulator, we build computationally cheap approximating models of the relationships between inputs and outputs. These models replace the simulator for the optimization problem outlined above in Section 2. As we shall see, the optimization may also be approached in a less formal way, via graphical techniques. The generic strategy involves seven steps.

1. Postulate a model for each output $y_k(x)$ as a function of all inputs x. As will be described in Section 4, possibilities include polynomial approximations and models treating each output as the realization of a stochastic process.

2. Plan an initial experiment of n sets of x vectors, and run the simulator at these input vectors to generate the quality-characteristic outputs.

3. Use the data to fit the models (e.g., to estimate the unknown coefficients in polynomial models), and build approximating functions, $\hat{y}_k(x)$. These will be used to predict $y_k(x)$ at new x vectors. Before proceeding further, the adequacy of the predictors should be assessed and, if necessary, improved. For example, in Yu et al. (1989) the prediction of a CMOS comparator circuit's gain was improved by separately modeling the gains at the circuit's two stages and then combining the two gains.

4. Plot the predicted surfaces. Using the methods described in Section 4 it is possible to decompose $\hat{y}_k(x)$ into main effects, two-factor interactions, and so on. Particularly when there are conflicting objectives, visualization of the input-output relationships can help to establish trade-offs. Even if model assessment indicated that the predictors are of low accuracy, the plots may suggest promising subregions of the x space for the next experiment.

5. Estimate the expected-loss function and perform a tentative optimization. We simply replace $y_k(c+u)$ in the expected loss (1) by the predictor $\hat{y}_k(x)$, where $x = c + u$. This gives an estimated expected loss, $\hat{L}(c)$. Monte Carlo is a straightforward way to compute the integral in the $\hat{L}(c)$ analog of (1); this is usually computationally feasible because $\hat{y}_k(x)$ is cheap to compute. Denote the resulting "optimal" c by c^*.

6. If necessary, reduce the size of the input space. If Step 3 indicated that the predictor is not accurate enough yet, reduce the size of the x region to a promising subregion based on the plots in Step 4 and the tentative optimization in Step 5, and return to Step 2 to collect new data on the subregion. Otherwise continue to Step 7.

7. Conduct a confirmation experiment to evaluate c^* found in Step 5. A single Monte-Carlo integration might be carried out to evaluate the expected loss $L(c^*)$ using (1) by running the computer code. Alternatively, as described in Yu et al. (1989), a small computer experiment could be

conducted, fixing c at c^* and only varying u. Again, this requires relatively few simulator runs. Fixing c makes the space of $x = c^* + u$ smaller, and the predictor $\hat{y}_k$ is likely to be more accurate. This predictor can then be used in (1) to estimate the expected loss. If the confirmation experiment demonstrates that c^* is unsatisfactory, then we have to return to an earlier step. This might involve reducing the x space and collecting more data, as outlined in Step 6.

Within this common framework, different classes of predictors $\hat{y}_k(x)$ and various objective functions can be substituted in a modular way. The following sections elaborate on these alternatives.

4. PREDICTION

Building predictors based on fitting polynomial (e.g., second order) models by least squares is well known and computationally straightforward, at least when the dimension d of the x space is modest. If the output function $y_k(x)$ is simple, which is most likely to occur when the x ranges are small, then accurate predictions can result. For example, Welch et al. (1990) modeled the skew of a clock-driver circuit as a function of six transistor widths and five levels of manufacturing noise. A model with linear and quadratic terms for the widths, a five-level qualitative factor for the manufacturing noise, and some two-factor interactions led to a highly accurate predictor.

There are a number of philosophical and practical objections to least-squares fitting of polynomials, however. The usual statistical assumptions implicit in fitting a regression model by least squares and making inferences are inappropriate. A deterministic output deviates from the regression model according to *systematic* error and not the white-noise error usually assumed. Related to this is the fact that the least-squares predictor need not interpolate the observed values of an output, which are known exactly. Last, and perhaps most important, we have often found that polynomials are not flexible enough to model complex input-output relationships. In an example described below,

a second-order polynomial has 2 to 3 times the prediction error of the approach we now outline.

Modeling and prediction of deterministic outputs from computer codes was discussed in Sacks, Schiller, and Welch (1989) and in Sacks, Welch, Mitchell, and Wynn (1989). These methods have some advantages relative to polynomial models fitted by least squares. The resulting predictors are flexible, they interpolate the data, and in a number of applications they have led to more-accurate predictions. On the negative side, fitting these models is computationally more expensive than least squares, though computationally efficient algorithms are being developed by us and other teams.

This framework treats an output of interest, $y(x)$, as a realization from a stochastic process or random function on x:

$$Y(x) = \beta + Z(x). \tag{2}$$

The unknown constant β can be replaced by a regression model (e.g., low-order polynomial) in x, but we have found in applications that this usually offers little advantage. The random process $Z(x)$ is assumed to have mean zero and covariance

$$\sigma^2 R(w, x) \tag{3}$$

between $Z(w)$ and $Z(x)$ at two vector-valued inputs w and x. Here, σ^2 is the process variance and $R(w, x)$ is the correlation function. The idea is that a deterministic output, though actually generated quite differently, may resemble a sample path of a (suitably chosen) stochastic process $Z(x)$.

The correlation $R(w, x)$ should reflect the characteristics of the computer-code output: a correlation function with some derivatives would be appropriate for a smoothly varying output, for instance. The circuit simulation examples in Bernardo et al. (1989) and Buck et al. (1989) employed

$$R(w, x) = \Pi_{j=1}^{d} \exp(-\theta_j |w_j - x_j|^{p_j}), \tag{4}$$

where $\theta_j \geq 0$ and $0 < p_j \leq 2$. This family is a product of stationary one-dimensional correlations. The case $p_1 = \cdots = p_d = 2$ gives a process with

infinitely differentiable paths (mean square sense) and is useful when the response is analytic.

Maximum likelihood estimation of the unknown parameters in this model—β in (2), σ^2 in (3), and $\theta_1, \ldots, \theta_d$ and $p_1, \ldots, p_d$ in (4)—was outlined by Sacks, Welch, Mitchell, and Wynn (1989).

This model leads to a best linear unbiased predictor (BLUP) which exploits the correlations between the observed values of y in the experimental plan and the (random) $Y(x)$ at an untried x. According to the correlation function (4), $Y(x)$ will have the largest correlations with those outputs at points close to x in the experimental plan. Thus, the predictor, which is a linear combination of the observed outputs, tends to assign greater weight to nearby observations than those far away. It turns out the BLUP is a generalized least squares fit of β (or the regression-type trend) in model (2), plus an interpolation of the residuals. Again, the formal derivation and computational details can be found in Sacks, Welch, Mitchell, and Wynn (1989).

In quality improvement applications, the vector of inputs, x, is often high dimensional. For instance, the voltage-shifter example in Buck et al. (1989) started out with 23 input parameters. Expert engineering advice enabled a reduction to 14 factors, but, in general, it is important to reduce dimension and identify active factors on the basis of data. Welch, Buck, Sacks, Wynn, Mitchell, and Morris (1989) described such a screening algorithm based on the stochastic-process model (2). In two examples, 14-dimensional and 20-dimensional input vectors were reduced to the important factors, and nonlinear and interaction effects were discovered, with just 30–75 runs of the code. Moreover, the same data led to a useful predictor without further runs, an important consideration if the code is computationally complex.

A recent application in which we have been involved illustrates the advantages of this predictor relative to fitting polynomials by least squares. There are at least 10 outputs of varying degrees of interest in the example. We concentrate here on one output, a time delay. This delay should not exceed 6.7 ns (nanoseconds), with smaller delays being even better. There are 11

inputs, comprising various device dimensions, capacitances, and so on. Applying a modified version of the maximum-likelihood algorithm in Welch et al. (1989) to fit the stochastic-process model (2) indicates that x_5, x_9, and x_{10} are inactive.

We may assess the accuracy of the resulting best linear unbiased predictor by cross validation. Let $x^{(i)}$ denote the x vector for run i of the experiment, and let $\hat{y}_{-i}(x^{(i)})$ be the predictor of $y(x^{(i)})$ based on all the data *except* the observation $y(x^{(i)})$. Then we can define an empirical root mean squared error (ERMSE) at the 75 observations by

$$\cdot \left\{ \frac{1}{75} \sum [\hat{y}_{-i}(x^{(i)}) - y(x^{(i)})]^2 \right\}^{\frac{1}{2}}. \tag{5}$$

For the time-delay output the ERMSE is about 0.047 ns, relative to a data range of about 5.1 to 10.7 ns.

This predictor may be compared with a polynomial fitted by least squares. With 75 runs it is not possible to fit a full second-order model, so we initially fit a first-order model and apply backward elimination until $t > 2$ for all t statistics. This identifies x_3, x_4, x_7, x_8, x_9, and x_{11}. A second-order model in these inputs, followed by further backward elimination, leads to a model with 14 terms. Its ERMSE from (5) is 0.12 ns, 2 to 3 times that of the predictor based on the stochastic-process model (2). The ERMSE from the regression model is too large to permit its use in optimization with any confidence. In fact, the regression model erroneously predicts that the "optimal" design (found using the stochastic-process predictor) would violate the constraint of 6.7 ns on the delay time by a nontrivial margin.

To gain insight into the relationships between the important inputs and the responses we like to visualize the fitted model. Specifically, we:

1. Decompose each response function $y(x)$ into an overall average, main effects, and two-factor interactions.

2. Estimate these effects by replacing the unknown $y(x)$ by $\hat{y}(x)$.

3. Plot the estimated effects.

This decomposition was suggested by Sacks, Welch, Mitchell, and Wynn (1989) and exploited graphically by Welch et al. (1989). Thus, the estimated overall average of the response $y(x)$ is

$$\hat{\mu}_0 = \int \hat{y}(x)\Pi_{h=1}^{d} dx_h,$$

the estimated main effect of input factor x_i (averaged over the other factors) is the function

$$\hat{\mu}_i(x_i) = \int \hat{y}(x)\Pi_{h\neq i} dx_h - \hat{\mu}_0,$$

and the estimated interaction effect of x_i and x_j is the function

$$\hat{\mu}_{ij}(x_i, x_j) = \int \hat{y}(x)\Pi_{h\neq i,j} dx_h - \hat{\mu}_i(x_i) - \hat{\mu}_j(x_j) - \hat{\mu}_0.$$

We do not consider higher order interactions because of obvious difficulties in plotting. With product correlation functions like (4) the integration is just a product of one-dimensional integrals.

We illustrate these visualization techniques using the time-delay example discussed earlier in this section and the stochastic-process model (2). Figure 1 shows the estimated main effects of $x_1, \ldots, x_4$, x_6, x_7, x_8, and x_{11}, the factors identified as having some effect. In these plots the estimated overall mean, $\hat{\mu}_0$, has been added in. It is apparent that x_3, x_8, and x_{11} have the most important main effects; the remaining factors produce a blur of small effects. Contour plots of the estimated interactions, $\hat{\mu}_{ij}(x_i, x_j)$, for all pairs x_i and x_j from $\{x_1, \ldots, x_4, x_6, x_7, x_8, x_{11}\}$ show that the estimated interaction effect of x_3 and x_{11} is important. This estimated interaction has contours as large as ± 0.2; all other interactions are apparently unimportant. As in physical experiments, when a two-factor interaction is detected it is useful to look at the joint effect of the two factors—the sum of the two main effects and the interaction. It is estimated here by integrating the predictor with respect to all factors except x_3 and x_{11}. Figure 2 shows the contour plot. With pictures like these for all important outputs, the engineer can see the input-output relationships and identify any trade-offs between conflicting objectives.

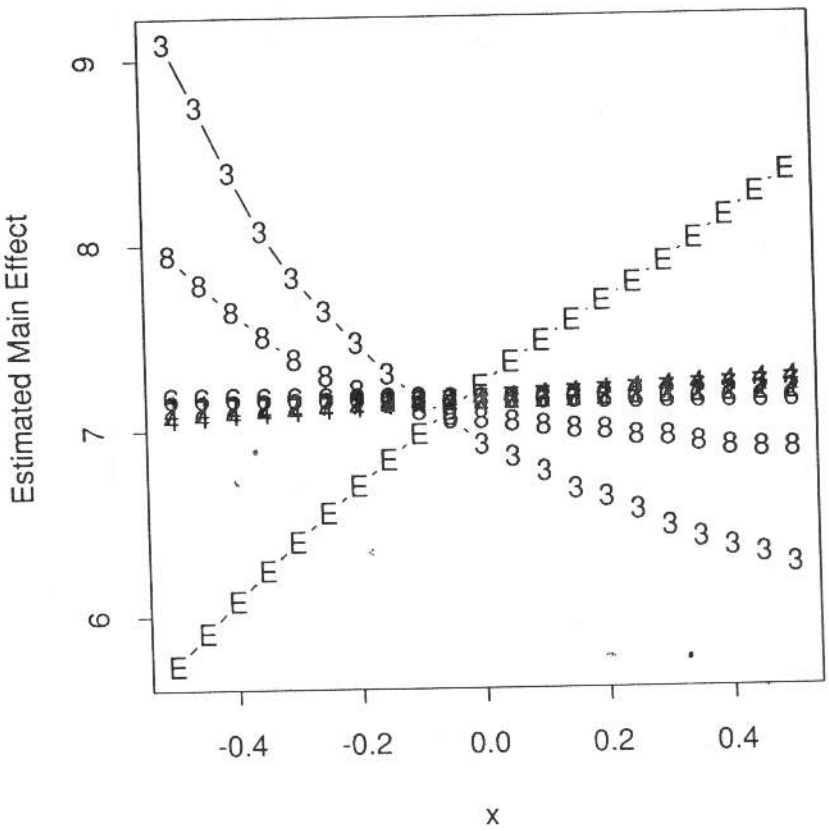

FIG. 1: Estimated main effects of $x_1, \ldots, x_4$, x_6, x_7, x_8, and x_{11} (plotting symbols 1,...,4, 6, 7, 8, and E, respectively) on time delay (ns).

5. PLANNING OF EXPERIMENTS

So far we have avoided the issue of how to choose the experimental plan (Step 2 in Section 3). In the absence of random error, criteria like D optimality (e.g., St. John and Draper 1975) are inappropriate. Welch et al. (1990) optimized the experimental plan using an integrated mean squared error criterion based on systematic departure from an assumed polynomial model. Sacks, Schiller, and Welch (1989) used a similar criterion but based on the predictor from the

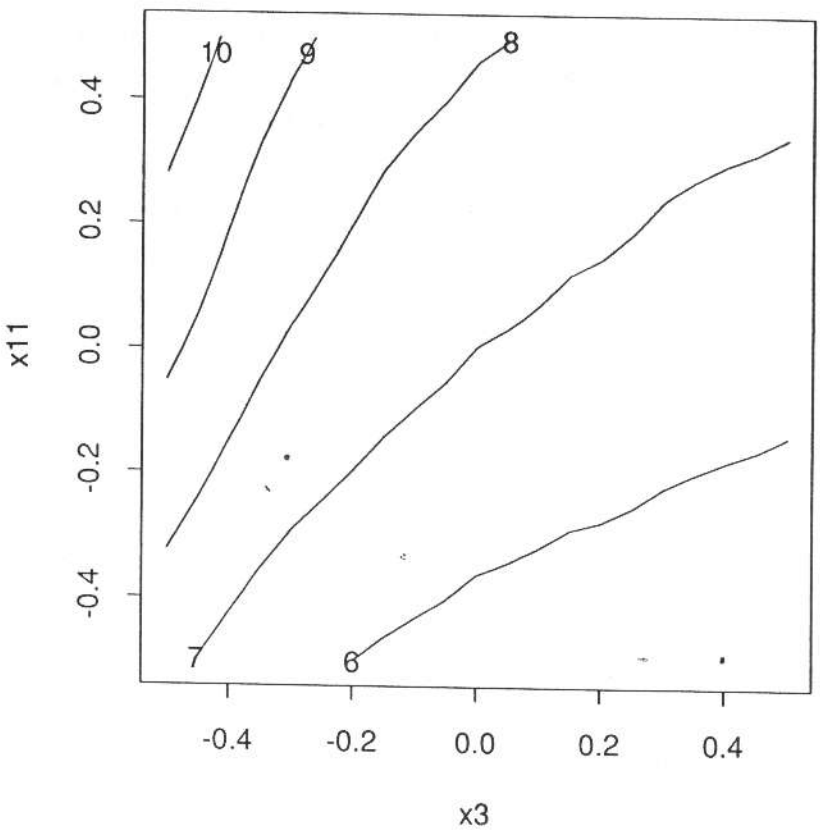

FIG. 2: Estimated joint effect of x_3 and x_{11} on time delay (ns).

stochastic-process model (2) rather than a polynomial fitted by least squares. Many quality improvement computer experiments are too large for optimal planning of experiments in a reasonable amount of computing time, however. The example in Buck et al. (1989) has 14 dimensions and 75 runs at the first stage. Finding an optimal design requires optimization over $14 \times 75 = 1050$ coordinates, an expensive undertaking.

We now typically use Latin hypercube experimental plans (McKay, Conover, and Beckman 1979), which have some attractive properties for com-

puter experiments. First, they are simple to generate, even for many input variables. Secondly, they cover the experimental region fairly uniformly. Work going back to Box and Draper (1959) on fitting polynomials suggests that uniform coverage is desirable when systematic error is important. Similarly, optimal experiments for the predictor from the stochastic-process model (2) tend to infiltrate the entire space, at least for low-dimensional x where such plans can be computed. Thirdly, if only a few factors turn out to be important, the experimental plan projected down onto these factors is still a Latin hypercube, and so is still fairly uniform.

The question of sample size is also difficult. Ongoing work is investigating the relationship between the number of runs, dimension of x, and prediction accuracy. At present, to keep down the computational cost from running the simulator and from fitting the correlation parameters in the model (2), we typically take 50–100 runs for each experiment in the sequential strategy. For a high-dimensional x space, a first experiment of this size will often result in inaccurate prediction and a non-definitive optimization in Step 5 of Section 3. We then rely on the sequential reduction of the experimental region (Step 6) to improve accuracy—as the region shrinks, the response relationship will usually become less complex and easier to predict.

6. OPTIMIZATION CRITERIA

We now consider some alternatives for the loss (1) and its estimate optimized in Step 5 of Section 3.

6.1. Quadratic loss

Taguchi (1986, Chapter 2) advocated minimizing the mean squared deviation of a quality characteristic from its target. With a single characteristic of interest, the loss function in Step 5 of our strategy in Section 3 is easy to implement: see the examples in Welch et al. (1990) and Bernardo et al. (1989).

Reconciling quadratic losses from several quality characteristics is not so simple, however. The objective function is inevitably problem specific, not in

itself undesirable, but specifying the trade-offs raises obvious practical difficulties.

In the design of a voltage shifter circuit, Buck et al. (1989) considered and built predictors for four characteristics—bandwidth, gain, voltage shift, and (indirectly) ripple—which we denote by y_B, y_G, y_V, and y_R. Bandwidth and gain are larger the better characteristics, the voltage shift has a target of 5 V, and ripple is smaller the better. For a given nominal design c (the controllable component of x), they computed $\hat{y}_B(c)$ and $\hat{y}_G(c)$. (These are roughly equivalent to the means with respect to the distribution of u, the uncontrollable component of x.) They also estimated the variances of the bandwidth and the gain around their nominal values:

$$s_B^2(c) = \int [\hat{y}_B(c+u) - \hat{y}_B(c)]^2 f(u)\, du$$

and

$$s_G^2(c) = \int [\hat{y}_G(c+u) - \hat{y}_G(c)]^2 f(u)\, du.$$

Similarly, the variability of the voltage shift around the target of 5 V was predicted by

$$s_V^2(c) = \int [\hat{y}_V(c+u) - 5]^2 f(u)\, du.$$

Then, by maximizing

$$\hat{y}_B(c) + \hat{y}_G(c) - s_B(c) - s_G(c)$$

subject to

$$s_V(c) \leq 0.1$$

and

$$\hat{y}_R(c) \leq 0.01,$$

with respect to c, they attempted to make the average bandwidth and gain large while penalizing variability of these quantities and controlling voltage shift and ripple. Though this combines the outputs in an *ad hoc* way, optimization led to an engineering design with consistently good outputs, the ultimate test.

6.2. Yield

The yield is defined as the proportion of product satisfying constraints on the quality characteristics, usually simple lower and/or upper bounds on each characteristic. Thus, yield maximization provides a much more straightforward way of dealing with several responses. On the negative side, it does not reward reduction in variability once the constraints are met.

Maximizing the yield is equivalent to minimizing the expected loss (1) if we define the loss as $l(y_1, \ldots, y_q) = 1$ if at least one output does not satisfy its constraint, and 0 otherwise. Thus, yield maximization is easy to implement within our framework; see Yu et al. (1989) for two examples.

Estimating the yield by substituting $\hat{y}_k(x)$ for $y_k(x)$, the approach taken by Yu et al. (1989), can sometimes lead to substantial error even if the predictors are fairly good. Consider a single response $y(x) = y(c+u)$ that is satisfactory if it exceeds the bound b and the marginal yield with respect to this output. If $\hat{y}(x)$ is close to b, then even a small error in $\hat{y}(x)$ can lead to predicting $y(x) > b$ when, in fact, $y(x) < b$, or vice versa.

Assigning a loss of 0 or 1 ignores uncertainty of prediction. If uncertainty is taken into account, then a prediction very close to the boundary might be assigned a loss of 0.5 to indicate that we really do not know which side of b the true output will fall. A formalization of this idea is as follows. Denote the standard error attached to $\hat{y}(x)$ by $s[\hat{y}(x)]$, and assume that $\hat{y}(x)$ has a normal distribution with mean $y(x)$ and standard deviation $s[\hat{y}(x)]$. Then estimate the loss $l[y(x)]$ by

$$\Phi\left(\frac{b - \hat{y}(x)}{s[\hat{y}(x)]}\right), \tag{6}$$

where $\Phi(a)$ is the probability that a standard normal random variable is less than a. These probabilities are then averaged (rather than 0's and 1's) in the computation of the expected loss (1).

The stochastic process model outlined in Section 4 leads to a predictor for which a standard error can be computed. Furthermore, as illustrated by a circuit-simulation example in Sacks, Welch, Mitchell, and Wynn (1989), these

standard errors often give a realistic idea of the magnitude of the actual error, and the predictors $\hat{y}(x)$ are often found to be approximately normal.

We now give some results from applying this method to the voltage-shifter circuit in Buck et al. (1989). (As described in Section 6.1, yield was not the objective in their study.) For simplicity, consider the marginal yield with respect to the voltage shift only. We take upper and lower bounds of 4.85 V and 5.15 V, and we use 100 Monte Carlo samples from the noise distribution to estimate the proportion of circuits meeting these bounds. Estimating the loss using $l[\hat{y}(x)]$ gave an estimated yield of 91% at the "optimal" circuit design. Using (6) gave an estimated yield of 80.5%, much closer to the true yield of 79% obtained by running the circuit simulator at the same 100 samples. Note that Monte Carlo error is irrelevant when comparing these figures as they are all based on the same samples.

7. CONCLUDING REMARKS

We have summarized work on the development of a system for the design of products via computer models. These methods have already proved useful in a number of integrated-circuit applications.

We are working on a software implementation of this system. This tool will help the engineer to: initially identify the important input parameters; build approximating models; visualize relationships; and proceed sequentially to a good design. These are complex problems, and complete automation of this process is unrealistic. Nonetheless, many of the statistical details can be hidden, allowing the engineer to concentrate on problem formulation, trade-offs, and so on.

ACKNOWLEDGMENTS

This research was supported by: the AT&T Affiliates Program; AFOSR and NSF through NSF DMS 86-09819 and DMS 90-01726; NSA MDA-904-89-H-2011; the Natural Sciences and Engineering Research Council of Canada; and

Cray Research, Inc. Many people have contributed to the research summarized here. In particular, we acknowledge Chona Bernardo, Robert Buck, Sung Mo Kang, Lihsin Liu, Toby Mitchell, Max Morris, William Nazaret, Henry Wynn, and Tat-Kwan Yu. Robert Buck also provided invaluable assistance with the new computations in this article.

BIBLIOGRAPHY

Bernardo, M.C., Sacks, J., Welch, W.J., Liu, L., and Nazaret, W. (1989), "Integrated circuit design optimization using a sequential strategy," Technical Report #32, Department of Statistics, University of Illinois at Urbana-Champaign.

Box, G.E.P. and Draper, N.R. (1959), "A basis for the selection of a response surface design," *Journal of the American Statistical Association*, 54, 622–654.

Brayton, R.K., Hachtel, G.D., and Sangiovanni-Vincentelli, A.L. (1981), "A survey of optimization techniques for integrated-circuit design," *Proceedings of the IEEE*, 69, 1334–1362.

Buck, R., Sacks, J., Welch, W.J., Liu, L., and Nazaret, W. (1989), "Integrated circuit design optimization using a sequential strategy: multiple performances," Technical Report #33, Department of Statistics, University of Illinois at Urbana-Champaign.

McKay, M.D., Conover, W.J., and Beckman, R.J. (1979), "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, 21, 239–245.

Sacks, J., Schiller, S.B., and Welch, W.J. (1989), "Designs for computer experiments," *Technometrics*, 31, 41–47.

Sacks, J., Welch, W.J., Mitchell, T.J., and Wynn, H.P. (1989), "Design and analysis of computer experiments" (with discussion), *Statistical Science*, 4, 409–435.

St. John, R.C. and Draper, N.R. (1975), "D-Optimality for regression designs: a review," *Technometrics*, 17, 15–23.

Taguchi, G. (1986), *Introduction to quality engineering*, Tokyo: Asian Productivity Organization.

Taguchi, G. and Wu, Y. (1979), *Introduction to off-line quality control*, Nagaya: Central Japan Quality Control Association.

Welch, W.J., Buck, R.J., Sacks, J., Wynn, H.P., Mitchell, T.J., and Morris, M.D. (1989), "Screening a large-dimensional input to a computer experiment," Technical Report #31, Department of Statistics, University of Illinois at Urbana-Champaign.

Welch, W.J., Yu, T.K., Kang, S.M., and Sacks, J. (1990), "Computer experiments for quality control by parameter design," *Journal of Quality Technology*, 22, 15–22.

Yu, T.K., Kang, S.M., Sacks, J., and Welch, W.J. (1989), "Parametric yield optimization of MOS integrated circuits by statistical modeling of circuit performances," Technical Report #27, Department of Statistics, University of Illinois at Urbana-Champaign.