Struct Multidisc Optim (2008) 36:457–475
DOI 10.1007/s00158-007-0189-0

RESEARCH PAPER

Numerical integration in statistical decision-theoretic methods for robust design optimization

S. C. Kugele · M. W. Trosset · L. T. Watson

Received: 22 December 2006 / Revised: 15 July 2007 / Accepted: 3 September 2007 / Published online: 25 January 2008

Abstract The Bayes principle from statistical decision theory provides a conceptual framework for quantifying uncertainties that arise in robust design optimization. The difficulty with exploiting this framework is computational, as it leads to objective and constraint functions that must be evaluated by numerical integration. Using a prototypical robust design optimization problem, this study explores the computational cost of multidimensional integration (computing expectation) and its interplay with optimization algorithms. It concludes that straightforward application of standard off-the-shelf optimization software to robust design is prohibitively expensive, necessitating adaptive strategies and the use of surrogates.

Keywords Bayes principle · Robust design optimization · Multidimensional integration

S. C. Kugele (✉)
Department of Computer Science,
Virginia Polytechnic Institute and State University,
Blacksburg, VA 24061, USA
e-mail: skugele@vt.edu

M. W. Trosset
Department of Statistics, Indiana University,
Bloomington, IN 47405, USA

L. T. Watson
Departments of Computer Science and Mathematics,
Virginia Polytechnic Institute and State University,
Blacksburg, VA 24061, USA

1 Introduction

Engineers increasingly rely on computer simulation to develop new products and to understand emerging technologies. In practice, this process is permeated with uncertainty: manufactured products deviate from designed products; actual products must perform under a variety of operating conditions. Most of the computational tools developed for design optimization ignore or abuse the issue of uncertainty, whereas traditional methods for managing uncertainty are often prohibitively expensive. The goal of the work described in this paper is the development of tractable computational tools that address these realities.

This paper addresses the problem of developing rigorous, computationally tractable methods for robust design, i.e., design optimization of complex, simulation-based engineered systems in the presence of uncertainty about manufacturing and operating conditions. Statistical decision theory, specifically the Bayes principle, provides a conceptual framework for quantifying these uncertainties. The difficulty with exploiting this framework is computational.

Robust design optimization (RDO) requires the simultaneous manipulation of design variables and noise variables and is an especially challenging class of multidisciplinary design optimization (MDO) (Sobieszczanski-Sobieski and Haftka 1997).

Burgee and Watson (1997) stated that, given the expense of calculating multiple disciplinary results from high-accuracy analyses, obtaining high-fidelity function values for more than 10–100 design points will be prohibitively expensive (regardless of the dimensionality of the design space). This realization has resulted in a great deal of MDO research focused on substituting

low-cost approximations, also known as *surrogates*, for the objective function and constraint functions. Various techniques have been explored for this purpose including classical response surface approximations (RS) (Unal et al. 1996), Bayesian estimators collectively known as DACE (Sacks et al. 1989), and variable-complexity modeling (VCM) (Unger et al. 1992).

Furthermore, the management of uncertainty using the Bayes principle requires numerical integration (i.e., calculating expectations), incurring an additional computational expense. Given an already expensive system, which must be analyzed at various design points, the efficient calculation of these expectations becomes critical for mitigating computational expense. Rather than focusing on the asymptotic behavior of various numerical integrators, the focus must be shifted to what can be obtained on a limited budget.

The application of statistical decision theory to robust design has been infrequently attempted and lies at the frontier of current engineering practice; the goal is to extend that frontier by developing more efficient computational methods. Inspired by the success of surrogate-based methods for design optimization, the ultimate goal is to develop surrogate-based methods for integration that use variable fidelity data, adaptive Newton–Cotes formulas, and parallel function evaluation. By facilitating modern techniques for decision making under uncertainty, this research should help advance efforts to design robust complex systems using computer simulation. The potential benefits include increased confidence in analysis tools; reductions in design cycle time, risk, and cost; increasingly robust designs; and improved system performance with ensured reliability. This paper is a small first step toward applying statistical decision theory directly to robust design.

The remainder of this paper is organized as follows. Section 2 provides background on RDO and explores the application of the Bayes principle for managing uncertainty. Section 3 provides necessary background in numerical integration, including adaptive quadrature, Monte Carlo, and quasi-Monte Carlo methods. The latter methods will be relevant for problems with more than approximately ten noise variables. Section 4 describes two variants of a prototypical robust design problem. Section 5 presents results obtained from applying two gradient-based optimization algorithms to the prototypical robust design problems, using four different numerical integration schemes. Section 6 contains concluding remarks, including discussion of adaptive optimization strategies and surrogate-based integration schemes that will be investigated in future work.

2 Robust design

An example of the class of problems addressed in this work was described in 1991 by Welch and Sacks, two statisticians who pioneered the design and analysis of computer experiments (DACE):

> 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.

Although their formulation and proposed solution of the quality improvement problem is modern, the problem itself predates the engineering community's use of computer models. To motivate the present approach to this problem, and the more general *robust design* problem, it is useful to briefly summarize the contributions of G. Taguchi (see Roy 1990).

2.1 Taguchi methods for quality engineering

Taguchi envisioned a three-stage process whereby engineers could *design* quality into their products. The first stage, *systems design*, determines the feasible region for subsequent optimization. The second stage, *parameter design* or *robust design*, optimizes the objective function that operationalizes one's notion of quality. This is the stage of particular relevance to the present work. The third stage, *tolerance design*, fine tunes the (approximately) optimal design obtained in the second stage.

Taguchi argued that one should design a product in such a way as to make its performance insensitive to variation in variables beyond the designer's control. His methods for robust design distinguish two types of inputs to a system: "control parameters" (or "control factors") are the inputs that can be easily controlled or manipulated by the designer, hence the inputs that constitute optimization variables a; "noise variables" (or "noise factors") are the inputs that are difficult or expensive to control, hence the inputs b to whose variation product performance is desired to be insensitive. For example, a might specify the design of a photocopier and b might specify the environment in which it must operate.

Although Taguchi's philosophical contributions to robust design are of fundamental importance, the efficacy of his methods is extremely controversial. These

controversies were surveyed in the panel discussion edited by Nair (1992); the focus here is on a specific objection to his method of optimization. This method was inspired by classical experimental design. The control parameters a are systematically varied according to an orthogonal array, the "control array" or "inner array." At each value of a, the noise variables are varied according to a second orthogonal array, the "noise array" or "outer array," and data from the "replications" of the quality characteristic across the noise array are used to estimate a *signal-to-noise ratio* (SNR). One obtains an array of estimated SNRs, which is then analyzed by standard analysis-of-variance techniques to identify values of a that produce robust performance.

Thus, Taguchi methods attempt to optimize an objective by specifying a priori all of the values of a at which the objective will be evaluated. (Data analysis is sometimes supplemented by performing one or more confirmatory experiments, but this is not a fundamental part of the optimization strategy.) Thus, Taguchi approach violates a fundamental tenet of numerical optimization—that one should avoid doing too much work until one nears a solution. In his defense, Taguchi was concerned primarily with situations in which sequential experimentation may not be possible, as when an entire manufacturing facility must be dedicated to performing the experiment. In modern engineering design optimization, however, performance is often assessed by computer simulation and the logistic necessity of one-shot experiments disappears. Hence, the concern here is *not* with Taguchi methods but with formulations of robust design that permit sequential experimentation.

2.2 Decision-theoretic formulations of robust design

Using ideas from statistical decision theory, the problem of robust design can be formulated as an optimization problem. Consider objective functions of the form $f : A \times B \to \Re$, where

- $a \in A$ represents decision variables, inputs (designs) controlled by the engineer.
- $b \in B$ represents uncertainty, inputs not controlled by the engineer.
- $f(a; b)$ quantifies the loss that accrues from design a when conditions b obtain.

The (unattainable) goal is to find $a^* \in A$ such that, for every $b \in B$,

$$f\left(a^*; b\right) \leq f(a; b) \qquad \forall\, a \in A.$$

Example airfoil shape design Suppose that f is the drag coefficient of an airfoil, whose shape is specified by a. Then, b might specify:

1. Manufacturing errors ϵ that perturb the design. The desired airfoil shape is a but the manufactured airfoil shape is $a - \epsilon$. The problem is to find a design that will minimize the drag of the manufactured airfoils.
2. Mach numbers $M \in [0.7, 0.8]$. The problem is to design an airfoil that performs well over a range of different Mach numbers.

The unsolvable problem of finding $a^* \in A$ that simultaneously minimizes $f(a; b)$ for each $b \in B$ is the central problem of statistical decision theory: find a decision rule that simultaneously minimizes risk for every possible state of nature. A standard way of negotiating this problem is to replace each $f(a; \cdot)$ with a real valued attribute of it, e.g., a minimax principle

$$\min_{a \in A} \phi(a), \quad \text{where } \phi(a) = \sup_{b \in B} f(a; b),$$

or a Bayes principle

$$\min_{a \in A} \phi(a), \quad \text{where } \phi(a) = \int_B f(a; b)\, p(b)\, db, \tag{1}$$

where p denotes a probability density function on B.

The minimax principle is extremely conservative. It seeks to protect the decision maker against the worst-case scenario. This work adopts the Bayes principle, which seeks to minimize average loss in a sense that can be customized (via the choice of p) to the application. This formulation of the quality control problem was first proposed by Welch et al. (1990), although their suggestion appears to have had little effect on engineering practice.

Example airfoil shape design (*continued*) Let $a \in A \subset \Re^k$ denote the design.

1. Let $b = \epsilon \in \Re^k$ denote the manufacturing error and let $f(a; b) = f(a - \epsilon)$ denote the drag coefficient of the manufactured airfoil. The problem is to minimize

$$\phi(a) = \int_{\Re^k} f(a - \epsilon)\, p(\epsilon)\, d\epsilon = \left[f * p\right](a).$$

In this case, p might be chosen to approximate the probability distribution of the random manufacturing errors that perturb the design.

2. Let $b = M \in [0.7, 0.8]$ denote the Mach number and let $f(a; M)$ denote the drag coefficient at Mach M. The problem is to minimize

$$\phi(a) = \int_{0.7}^{0.8} f(a; M)\, p(M)\, dM.$$

In this case, p is a weight function that quantifies the value placed on performance at different speeds. This problem was studied by Huyse and Lewis (2001).

2.3 Optimization under uncertainty

This work is concerned with solving optimization problems of the general form (1). Having formulated the quality improvement problem as a special case of (1), Welch and Sacks (1991) posed the following question: "Why not simply plug the ultimate objective function into a numerical optimizer?" The answer is computational expense. In typical engineering applications, each evaluation of f is expensive; it follows that numerical integration of f is *very* expensive. Optimization under uncertainty is expensive in the sense that one cannot afford enough evaluations of the objective function

$$\phi(a) = \int_B f(a; b)\, p(b)\, db \tag{2}$$

to rely on traditional methods for numerical optimization. Because of this expense, the following nontraditional perspectives inform the present work.

- Traditional research in numerical optimization has emphasized convergence analysis. Performance has been assessed by measuring the expense (number of function evaluations, CPU time) required to solve the problem to within a specified tolerance. In contrast, ask the following question: What can be accomplished with a limited budget for evaluating f? The stance here is that future advances will come from clever heuristics and numerical experimentation, not convergence analysis.
- Practical methods tend to emphasize crude approximations, not exact solutions.
- If a problem is hard to solve in the absence of uncertainty, then it can only be harder to solve in the presence of uncertainty. For example, consider the rather mundane problem of measuring a physical quantity μ when the measurements X_i are corrupted by error. Suppose that $X_i \sim \text{Normal}(\mu, 1)$. Then, like it or not, constructing a 0.95-level confidence interval of length 0.2 requires more than 384 observations.

Believing that (1) could not be solved by traditional algorithms for numerical optimization, Welch and Sacks (1991) proposed "a system for quality improvement via computer experiments" to compete with Taguchi methods. Within the engineering community, their system has been popularized as DACE. As commonly practiced, DACE is the following recipe for minimizing a computationally expensive function $\psi : A \to \Re$.

1. Choose $a_1, \ldots, a_N \in A$ at which to evaluate ψ.
2. Compute $\psi(a_1), \ldots, \psi(a_N)$.
3. Construct a surrogate objective function $\hat{\psi}$. This might be accomplished by regression or (more often in DACE) by interpolation.
4. Minimize $\hat{\psi}$.

Welch and Sacks (1991) were concerned with $\psi(a) = \phi(a)$. Instead of Taguchi's inner and outer arrays, they envisioned a single "combined" array. Sacks and Welch commented that "the single experimental array for both control and noise factors will usually require far fewer observations than Taguchi's crossed arrays, even when interactions between the control factors are included." Thus, just like Taguchi methods, DACE relies on a single experiment: all evaluations of the objective function are made before optimization commences.

Curiously, the engineering community has embraced DACE *not* as it was intended – as a formulation of robust design and as an alternative to Taguchi methods – but as an alternative to iterative methods for the numerical optimization of computationally expensive objective functions. The introduction of DACE inaugurated a new area of research, *surrogate-based optimization*, to which the next phase of this project will be devoted. The initial phase of the project has concentrated on surrogate-based integration, described in the following sections. DACE was proposed for the purpose of minimizing $\psi(a) = \phi(a)$, but DACE has been used primarily to minimize expensive $\psi(a) = f(a; b)$ for fixed values of b. The recent surge of interest in optimization under uncertainty refocuses attention on the original problem for which DACE was proposed.

3 Numerical integration

If the robust design problem is formulated using the Bayes principle, then numerical integration is the key to optimization under uncertainty. This section reviews several important methods for numerical integration.

3.1 Adaptive quadrature

To approximate the definite integral

$$\int_S f(x)\,dx,$$

the inner loop of an adaptive quadrature algorithm uses formulas like

$$\int_T f(x)\,dx \approx \sum_{i=0}^{n} c_i f(x_i), \tag{3}$$

where $T \subset S$, the c_i, and the x_i may be fixed, adaptively determined, or even stochastic. Gauss chose the c_i and x_i so that (3) is exact when h is a polynomial of as high a degree as possible (Natanson 1965). Such formulas are optimal in many different regards and are ideally suited to automatic computation. In the context of large-scale engineering design, however, they have two significant drawbacks. First, the error in (3) is not easily estimated or controlled, especially when $f(x_i)$ is replaced by a surrogate or when f is only piecewise smooth. Second, the points x_i are extremely special and not easily reused in nested or adaptive formulas.

A different type of formula, also of form (3), is somewhat better than the Gaussian type with respect to the aforementioned drawbacks. These *Newton–Cotes* formulas fix the x_i, then choose the c_i to make (3) exact over some vector space of dimension $n+1$ (Hildebrand 1956). Good choices for the x_i and the matching subspace result in practical error estimates that are amenable to nested and adaptive integration algorithms. These adaptive strategies (Kahaner et al. 1989; Piessens 1983) permit efficient reuse of the x_i and the $f(x_i)$, and directly support analysis of the effect of replacing f with a surrogate. The latter feature is crucial: the ability to control the quality of the integral estimate directly supports the use of variable fidelity data values.

While the Newton–Cotes formulas provide a clear advantages over Gaussian formulas with respect to error control and ease of node selection, they also suffer from a number of disadvantages when applied to problems of higher dimension. In particular, the computational expense associated with using multivariate Newton–Cotes formulas becomes prohibitive as the dimensionality d increases. To illustrate this, consider the iterated integral

$$\int_{a_d}^{b_d}\int_{a_{d-1}}^{b_{d-1}}\cdots\int_{a_1}^{b_1} f(x_1,\ldots,x_d)dx_1,\ldots,dx_d.$$

A multivariate numerical integration formula for an integral of this form can be derived by applying univariate quadrature in an iterative fashion over each dimension of the multivariate integral. This method of derivation is referred to in the literature as a *product rule* and results in a cubature formula of the form

$$\sum_{i_d=1}^{N_d}\sum_{i_{d-1}=1}^{N_{d-1}}\cdots\sum_{i_1=1}^{N_1} A_{i_1}A_{i_2}\cdots A_{i_d} f(x_{i_1},\ldots,x_{i_d}),$$

where the A_{i_j} are coefficients from the univariate formulas, typically chosen so that the formula will be exact over some class of functions.

From this construction, it is immediately seen that the resulting cubature formula will require

$$N^* = \prod_{i=1}^{d} N_i$$

function evaluations, where N_i is the number of nodes sampled in the ith univariate quadrature formula. In other words, Newton–Cotes cubature formulas require the construction of a grid of N^* nodes. An immediate consequence of this grid structure is that refining the grid by doubling the number of nodes in each dimension will increase the number of required function evaluations by a factor of 2^d. More importantly, as d increases, the corresponding increase in computational effort will render this method infeasible. Therefore, in practice, stochastic integration methods called Monte Carlo methods are often employed to overcome this *curse of dimensionality*.

3.2 Monte Carlo integration

Given a Lebesgue integrable function $f(x)$, the integral of $f(x)$ can be viewed as an average or expectation of $f(x)$. Specifically, the value of the definite integral of a function $f(x)$ is equal to the product of the expected value $E[f(x)]$ (with respect to a uniformly distributed random variable x) and the volume of the region of integration.

Given a sequence of points $\{x_n\}$ sampled from a uniform random distribution in a region $\mathcal{R} \subset \Re^d$, approximate $E\left[f(x)\right]$ by

$$E_N\left[f(x)\right] = \frac{1}{N}\sum_{n=1}^{N} f(x_n).$$

Furthermore, with $V(\mathcal{R})$ denoting the volume of $\mathcal{R}$, the integral

$$I\left[f(x)\right] = \int_{\mathcal{R}} f(x)\,dx = E\left[f(x)\right]\,V(\mathcal{R})$$

can be approximated by

$$I_N\left[f(x)\right] = E_N\left[f(x)\right]\,V(\mathcal{R}).$$

From the strong law of large numbers,

$$\lim_{N\to\infty} E_N\big[f(x)\big] = E\big[f(x)\big];$$

therefore,

$$\lim_{N\to\infty} I_N\big[f(x)\big] = I\big[f(x)\big].$$

For simplicity, henceforth, assume that $\mathcal{R} = I^d = [0,1]^d$ (the d-dimensional unit cube). The value of the integral becomes $E[f(x)]$.

The rate at which the Monte Carlo method converges to the true integral value can be derived by utilizing the central limit theorem, as suggested by several authors, including Caflisch (1998) and Krommer and Überhuber (1998). Take $d = 1$, let $\epsilon_N[f(x)]$ denote the Monte Carlo integration error, where

$$\epsilon_N\big[f(x)\big] = I_N\big[f(x)\big] - I\big[f(x)\big],$$

and let $\sigma^2 = \sigma[f(x)]^2$ denote the variance of f, where

$$\sigma^2 = \int_{I^d} (f(x) - I[f(x)])^2 dx.$$

The central limit theorem states that if $X_1, X_2, \ldots, X_N$ is a sequence of independent and identically distributed random variables, each having mean μ and variance σ^2, then the distribution of

$$\frac{X_1 + \cdots + X_N - N\mu}{\sigma\sqrt{N}}$$

tends to the standard normal as $N \to \infty$. That is, for $-\infty < a < \infty$,

$$P\left\{-a \le \frac{X_1 + \cdots + X_N - N\mu}{\sigma\sqrt{N}} \le a\right\} \to \frac{1}{\sqrt{2\pi}}\int_{-a}^{a} e^{-x^2/2} dx$$

as $N \to \infty$.

In the case of Monte Carlo integration, $f(X_1)$, $f(X_2), \ldots, f(X_N)$ is a sequence of independent identically distributed random variables with mean $E[f(x)]$. Moreover, notice that

$$\begin{aligned}\frac{\sqrt{N}}{\sigma}\epsilon_N\big[f(x)\big] &= \frac{\sqrt{N}\big(I_N\big[f(x)\big] - I\big[f(x)\big]\big)}{\sigma} \\ &= \frac{\sqrt{N}\left(\frac{f(X_1)+\cdots+f(X_N)}{N}\right)}{\sigma} - \frac{NI\big[f(x)\big]}{\sigma\sqrt{N}} \\ &= \frac{f(X_1) + \cdots + f(X_N) - NI\big[f(x)\big]}{\sigma\sqrt{N}}.\end{aligned}$$

Letting $\mathcal{R} = I^d = [0,1]^d$, $\mu = E[f(x)] = I[f(x)]$. Therefore,

$$\frac{\sqrt{N}}{\sigma}\epsilon_N\big[f(x)\big] = \frac{f(X_1) + \cdots + f(X_N) - N\mu}{\sigma\sqrt{N}},$$

and applying the central limit theorem to the above equation gives

$$P\left\{\big|\epsilon_N\big[f(x)\big]\big| \le \frac{\sigma a}{\sqrt{N}}\right\} \to \frac{1}{\sqrt{2\pi}}\int_{-a}^{a} e^{-x^2/2} dx$$

as $N \to \infty$.

In other words, the Monte Carlo method converges at the rate $\mathcal{O}(N^{-1/2})$ – known as the "$n^{-1/2}$ law" – where the constant affecting this term is the variance of the function $f(x)$ from its expected value $I[f(x)]$. Furthermore, because this error is probabilistically distributed as a normal random variable, one can bind the error to a particular range, with some probability that the error will exceed that bound.

One can use the converse of the central limit theorem to determine the number of function evaluations needed to arrive at a particular error estimate for a given confidence level c. Let $s(c)$ be a confidence function that maps a confidence level to a value of a; then, for confidence level

$$c = \frac{1}{\sqrt{2\pi}}\int_{-s(c)}^{s(c)} e^{-x^2/2} dx,$$

at least $N = \epsilon_N^{-2}\big(\sigma s(c)\big)^2$ function evaluations are required. Because the variance σ of the function will likely be unknown in practice, somehow it must be estimated (see, e.g., Caflisch 1998).

How does the convergence rate for Monte Carlo compare to the convergence rate of the Newton–Cotes cubature formulas? Caflisch (1998) states that grid-based cubature methods converge to the true integral value at a rate of $\mathcal{O}(N^{-k/d})$, where k is the order of the method and d is the number of dimensions. In contrast to the "$n^{-1/2}$ law" that characterizes the convergence rate of the Monte Carlo method, the dimensional dependence of the Newton–Cotes methods is clear. [While the Monte Carlo method appears to operate independently of dimension, Davis and Rabinowitz (1984) make the observation that the variance seems to increase with the number of dimensions.] However, the Monte Carlo method is extremely slow to converge for all d, and worse than this, the efficacy of additional points in reducing the error diminishes as N increases: reducing the error by a factor of ten requires one hundred times as many function evaluations. Nevertheless, due to the *curse of dimensionality*, for each Newton–Cotes formula of finite order k, there will exist dimension d for which the Monte Carlo method will be superior.

Consider a Newton–Cotes product rule formula constructed from repeated application of the one-dimensional composite Simpson's rule. The one-dimensional composite Simpson's rule has order $k=4$

because its error term is $\mathcal{O}(h^4) = \mathcal{O}(N^{-4})$, where $h = (b-a)/N$ and the one-dimensional region of integration is the interval $[a, b]$. Therefore, for arbitrary dimension d, the product Simpson's rule has convergence rate $\mathcal{O}(N^{-4/d})$, which suggests that, for $d > 8$, the Monte Carlo method will converge more rapidly.

From the probabilistic error formula demonstrated previously, the convergence rate depends on the number of function evaluations N and on a constant determined by the variance. As a result, efforts to improve the Monte Carlo method have focused on reducing the variance of the function by modifying the way in which the sampling points are chosen. Numerous techniques exist for reducing the variance. Among these, a technique known as *importance sampling* is one of the most widely used in practice. Importance sampling reduces the variance by rewriting the integral to include a probability density function $p(x) > 0$, where

$$\int_{I^d} p(x)dx = 1,$$

so that

$$\int_{I^d} \frac{f(x)}{p(x)} p(x)dx = I\left[f(x)\right] \approx I_N\left[f(x)\right] = \hat{E}_p\left[\frac{f(x)}{p(x)}\right],$$

where

$$\hat{E}_p\left[\frac{f(x)}{p(x)}\right] = \frac{1}{N}\sum_{n=1}^{N} \frac{f(x_n)}{p(x_n)}.$$

The function is evaluated at points chosen with respect to the probability density function $p(x)$, instead of with respect to the uniform random distribution used in the basic Monte Carlo method, and the resulting variance is given by

$$\sigma_p\left[f(x)\right]^2 = \int_{I^d} \left(\frac{f(x)}{p(x)} - I\left[f(x)\right]\right)^2 p(x)dx.$$

Ideally, $p(x) \approx f(x)/I[f(x)]$ so that $\sigma_p[f(x)]^2 \approx 0$; however, this requires the value $I[f(x)]$ of the integral (the original problem). Instead, $p(x)$ is chosen so that $p(x)/f(x)$ is approximately constant. O'Leary (2004) states that the intuition behind importance sampling is to evaluate the function in regions where $|f(x)|$ is largest and waste less time sampling from regions that contribute little to the value of the integral.

In practice, it is nontrivial to choose an appropriate $p(x)$. If a probability density function is included in the original integral, then choose $p(x)$ to be the original probability density function. Using this approach, the modified integral will take less computational resources than the original integral (Krommer and Überhuber 1998). On the other hand, if a probability density function is not included in the original integral, one method for determining $p(x)$ would be to divide the region of integration $\mathcal{R}$ into smaller regions, sample $f(x)$ at the center x_i of each of these subregions (mesh point), and then set $p(x)$ to be constant over each of these subregions with a value proportional to the magnitude of the sampled function value at the mesh point. With a probability $p(x_i)$ for each mesh point x_i, sample $Np(x_i)$ points from region i, where N is the total number of points to be sampled (O'Leary 2004).

The Monte Carlo method is robust in its simplicity. It allows for simple implementations, provides dimensional independence, and places no continuity requirements on the integrand. On the other hand, the asymptotic error bounds are probabilistic, introducing a small probability that the error bounds will be grossly inaccurate; convergence is slow and decreasing; and the method does not take advantage of smoothness when the integrand *is* smooth (improvements on the basic Monte Carlo method such as stratified sampling and antithetic variates do take advantage of integrand smoothness). The quasi-Monte Carlo method, which will be considered next, attempts to address some of these fundamental difficulties.

3.3 Quasi-Monte Carlo integration

Traditional Monte Carlo integration techniques use pseudorandom sequences to generate the needed sampling points. A side effect of this stochastic sampling mechanism is that the sampled points tend to conglomerate, leaving some regions unsampled while oversampling in others; this has the undesirable consequence of slowing convergence. Deterministic sequences, called quasirandom sequences, can be substituted for the pseudorandom sequences used in Monte Carlo integration, forming the basis for the *quasi-Monte Carlo* integration techniques.

In contrast to pseudorandom sequences, quasirandom sequences have the desirable property of correlating the sampled points with the goal of maximizing the "uniformity" of the sampled points. The uniformity of a particular sequence can be quantified by its discrepancy, where sequences with low discrepancies are closer to uniformity than sequences with high discrepancies. For this reason, quasirandom sequences are often referred to as low-discrepancy sequences.

Let $\mathcal{B}$ denote a family of Lebesgue measurable subsets of $[0, 1]^d$. Let $S_N = \left\{x_i^{(N)}\right\}_{i=1}^{N}$ be a sequence of points in $[0, 1]^d$. The discrepancy of S_N with respect to $\mathcal{B}$, or *the general discrepancy*, is defined as

$$D(\mathcal{B}; S_N) = \sup_{B\in\mathcal{B}} \left|\frac{A(B; S_N)}{N} - \lambda_d(B)\right|,$$

where $A(B; S_N)$ is the number of points from S_N that are also elements of B and $\lambda_d(B)$ is the d-dimensional Lebesgue measure of B. Note that $A(B; S_N)/N \in [0, 1]$ and $\lambda_d(B) \in [0, 1]$, which implies $D(\mathcal{B}; S_N) \in [0, 1]$.

For theoretical reasons, it is useful to define two variations of the general discrepancy. Let $\mathcal{J}$ denote the family of Lebesgue measurable subsets of $[0, 1)^d$ of the form $\prod_{i=1}^{d}[u_i, v_i)$, and let $\mathcal{J}^*$ denote the family of Lebesgue measurable subsets of $[0, 1)^d$ of the form $\prod_{i=1}^{d}[0, v_i)$. The *discrepancy* of the sequence S_N is defined as

$$D_N(S_N) = D(\mathcal{J}; S_N),$$

and the *star discrepancy* of the sequence S_N is defined as

$$D_N^*(S_N) = D(\mathcal{J}^*; S_N).$$

A sequence $S_N = \{x_i^{(N)}\}_{i=1}^{\infty}$ is considered uniformly distributed, or equidistributed, if and only if

$$\lim_{N\to\infty} D_N(S_N) = 0,$$

or equivalently

$$\lim_{N\to\infty} D_N^*(S_N) = 0.$$

It can be shown (see Niederreiter 1992) that for any $S_N = \{x_i^{(N)}\}_{i=1}^{N}$ in $[0, 1]^d$,

$$D_N^*(S_N) \le D_N(S_N) \le 2^d D_N^*(S_N).$$

Using the star discrepancy and the Koksma–Hlawka theorem, it is possible to obtain an upper bound on the quasi-Monte Carlo integration error. The Koksma–Hlawka Theorem states that, for any sequence of points $S_N = \{x_i^{(N)}\}_{i=1}^{N}$ in $[0, 1)^d$ and any function f with bounded variation $\mathrm{Var}[f]$ in the sense of Hardy–Krause (see Niederreiter 1992) on $[0, 1]^d$, the integration error $\epsilon[f] = |I_N[f(x)] - I[f(x)]|$ satisfies the bound

$$\epsilon[f] \le \mathrm{Var}[f] D_N^*.$$

Following the convention of Caflisch (1998), quasirandom will denote sequences S_N that satisfy

$$D_N^*(S_N) \le D_N(S_N) \le c\frac{(\log N)^k}{N},$$

where c and k are independent of N but may depend on the dimension d. In particular, sequences have been constructed (e.g., Halton, Hammersley, Sobol) that satisfy

$$D_N^*(S_N) = \mathcal{O}\left(\frac{(\log N)^d}{N}\right).$$

Therefore, using the attained discrepancies, a conservative upper bound on the integration error is $\mathcal{O}\big((\log N)^d N^{-1}\big)$.

There is an open, though widely believed, conjecture that

$$D_N^*(S_N) = \Omega\left(\frac{(\log N)^{d-1}}{N}\right),$$

where the constant depends only on the dimensionality. Several of the sequence constructions to date have attained this order for the star discrepancy.

It is worth emphasizing that the difference between the Monte Carlo and quasi-Monte Carlo methods lies in the way the points are sampled, not in the way the integral estimate is calculated. Substituting low-discrepancy sequences for pseudorandom sequences produces a more rapid asymptotic convergence rate, as was shown above, but at the cost of introducing an explicit dimensional dependence reflected in the constant c_d and the exponent in the error bound. In addition, not only does quasi-Monte Carlo fail to take advantage of smooth integrands, but it may actually lose some of its effectiveness when faced with discontinuous integrands (see Caflisch 1998 for a more detailed treatment of the topic of quasi-Monte Carlo and smoothness).

4 Examples

The purpose of this paper is to consider how standard methods perform when applied to problems of robust design. Toward that end, this section formulates two variants of a prototypical problem. This problem includes both a robust objective function and a reliability constraint, i.e., a constraint that the probability of failure not exceed a specified tolerance. The subject of *reliability-based design* concerns techniques for minimizing an objective function (typically not robust) subject to such constraints. Because the probability of failure is an integral, the issues related to numerical integration that are studied herein are also germane to reliability-based design.

4.1 Prototype robust design problem 1

Let $d = (d_1, d_2)^t$ denote the design variables, with design bounds $d_1 \in [-10, 10]$ and $d_2 \in [0, 10]$. Let $X = (X_1, X_2)^t$ denote a random vector representing

uncertain physical parameters or operating conditions, and assume that $X_1 \sim$ Uniform$(-1, 1)$ and $X_2 \sim$ Uniform$(-3/4, 3/4)$. Define a "coupled analysis" $y(d, X)$ by the following algorithm:

$y_2 \leftarrow 5$
Do until convergence:
$\{y_1 \leftarrow d_1^2 + d_2 - y_2/5 + X_1;$
$y_2 \leftarrow d_1 - d_2^2 + (1/2)e^{-y_1^2} + X_2\}$
Return (y_1, y_2)

Let

$$f(d, X) = \begin{cases} 1, & \text{if product } d \text{ fails under } X, \\ 0, & \text{otherwise,} \end{cases}$$

where failure means $y_1(d, X) < 8$ or $y_2(d, X) > 5$. The probability of failure is $E[f(d, X)]$, and $\tau > 0$ denotes the probability of failure that is tolerable.

Ignoring the possibility of failure, let

$$L(d, X) = (d_1 + 1)^2 + 10d_2^2 + y_1(d, X)$$

denote the loss incurred from successfully operating product (design) d under condition X. The corresponding risk of operating d is given by

$$R(d) = E[L(d, X)] = (d_1 + 1)^2 + 10d_2^2 + E[y_1(d, X)].$$

The optimization problem is

$$\min_d R(d) \quad \text{subject to} \quad E[f(d, X)] \le \tau,$$
$$d_1 \in [-10, 10],$$
$$d_2 \in [0, 10].$$

Figure 1 shows the contours for the objective function $R(d)$ and the contours for the expected value $E[f(d, X)]$ in the stochastic constraint. Observe that the constraint contains large flat regions where all designs fail with probability one or succeed with probability one. As a consequence, gradient-based optimizers will generally fail given an infeasible design in these flat regions due to a zero-constraint gradient. A further consequence is that, within these flat regions, all Monte Carlo-based integrators will be exact after only one function evaluation; therefore, for these regions, Monte Carlo will outperform its Newton–Cotes counterparts in terms of cost. Problem 1 contains two local minima: 25.861 at $d \approx (3.104, 0)^t$ and 12.642 at $d \approx (-2.904, 0)^t$.

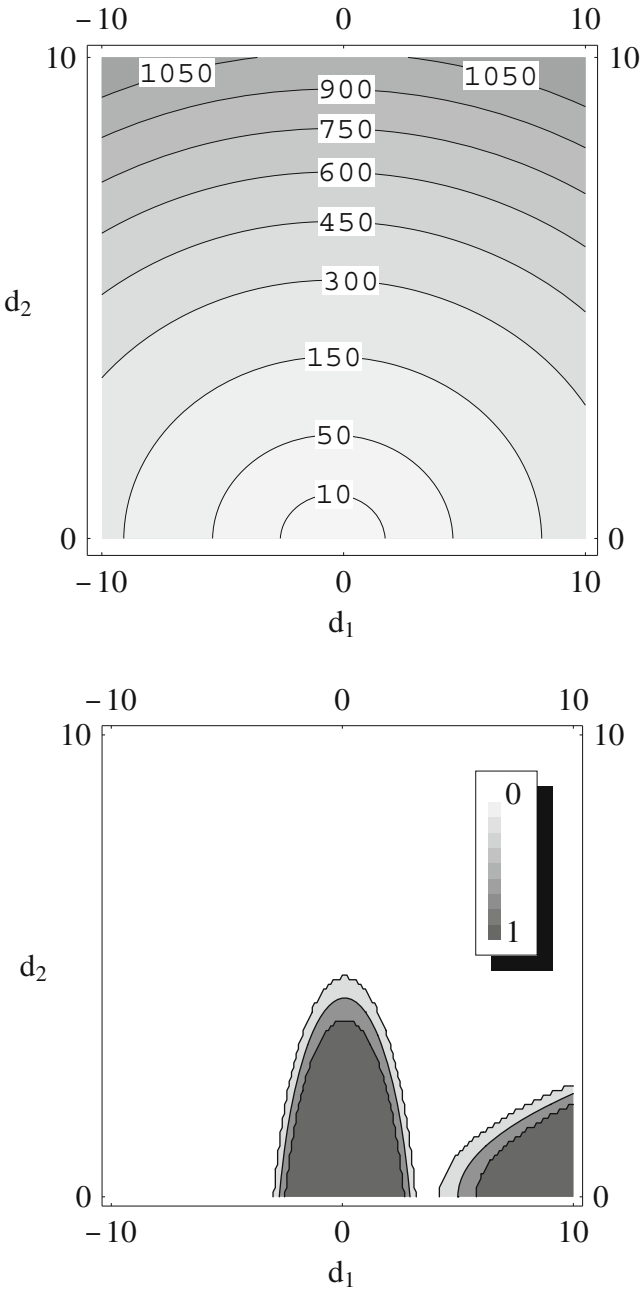

Fig. 1 For problem 1, $R(d)$ contours (*top*) and $E[f(d, X)]$ contours (*bottom*)

4.2 Prototype robust design problem 2

Motivated by the difficulties encountered with problem 1, a second robust design problem was developed with a modified constraint that has no flat regions. This was accomplished by changing the distribution of the noise variables from uniform to normal, which removes the problematic flat regions of problem 1, though at the expense of introducing more difficult integrands.

Let $d = (d_1, d_2)^t$ denote the design variables, with design bounds $d_1 \in [-10, 10]$ and $d_2 \in [0, 10]$. Let $X = (X_1, X_2)^t$ denote a random vector representing uncertain physical parameters or operating conditions, and

assume that $X_1 \sim N(0, 1)$ and $X_2 \sim N(0, 1)$. Define a "coupled analysis" $y(d, X)$ by the following algorithm:

$y_2 \leftarrow 5$
Do until convergence:
$\{y_1 \leftarrow d_1^2 + d_2 - y_2/5 + X_1;$
$y_2 \leftarrow d_1 - d_2^2 + (1/2)e^{-y_1^2} + 40X_2\}$
Return (y_1, y_2)

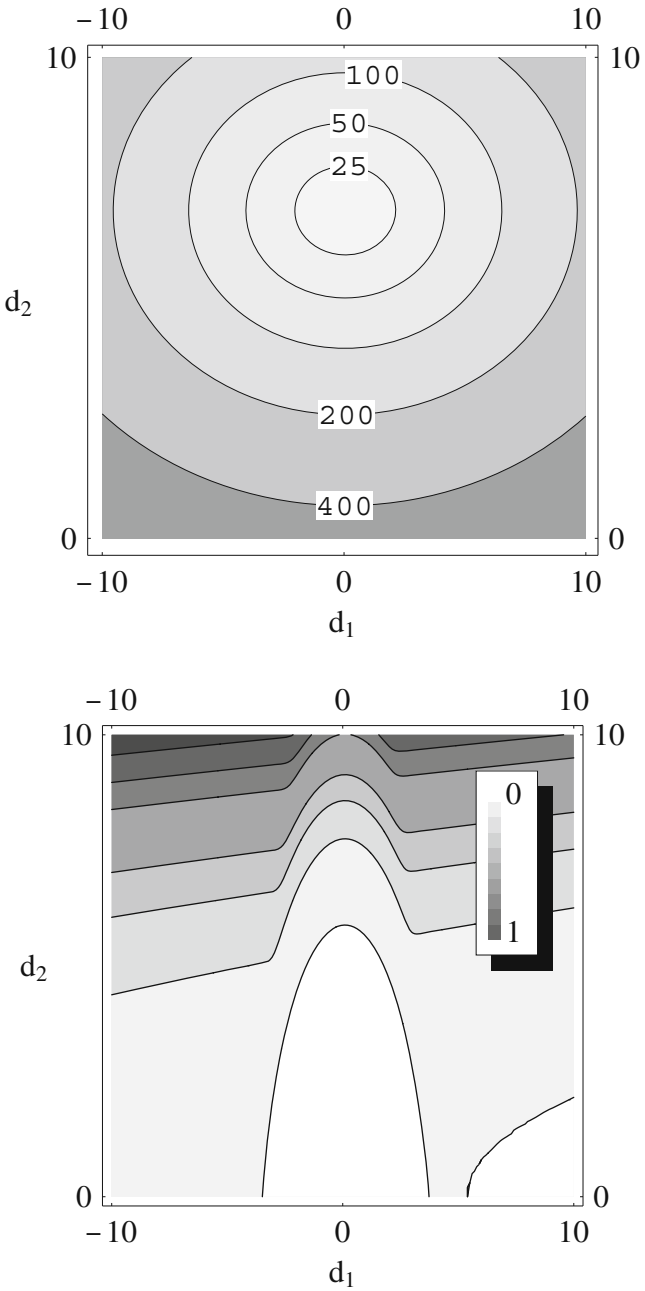

Fig. 2 Problem 2, $R(d)$ contours (*top*) and $E[f(d, X)]$ contours (*bottom*)

Let

$$f(d, X) = \begin{cases} 1, & \text{if product } d \text{ fails under } X, \\ 0, & \text{otherwise,} \end{cases}$$

where failure means $y_1(d, X) > 30$ and $y_2(d, X) < -80$. The probability of failure is $E[f(d, X)]$, and $\tau > 0$ denotes the probability of failure that is tolerable.

Ignoring the possibility of failure, let

$$L(d, X) = d_1^2 + 10(d_2 - 7)^2 + y_1(d, X)$$

denote the loss incurred from successfully operating product (design) d under condition X. The corresponding risk of operating d is given by

$$R(d) = E[L(d, X)] = d_1^2 + 10(d_2 - 7)^2 + E[y_1(d, X)].$$

The optimization problem is

$$\min_d R(d) \quad \text{subject to} \quad E[f(d, X)] \le \tau,$$
$$d_1 \in [-10, 10],$$
$$d_2 \in [0, 10].$$

Figure 2 shows the contours for the objective function $R(d)$ and the contours for the expected value $E[f(d, X)]$ in the stochastic constraint. Problem 2 has two local minima: 25.240 at $d \approx (0.055, 5.881)^t$, and 416.669 at $d \approx (8.024, 1.636)^t$.

5 Numerical results

In this section, the accuracy, efficiency, and suitability of various well known numerical integration techniques for solving problems 1 and 2 will be examined. Section 5.1 examines the performance of these numerical integration techniques when used to evaluate the integrands contained in the test problems. Section 5.2 examines the interaction between the output of the numerical integrators and several standard optimization algorithms. Experiments were performed using a Sun Blade 2000 workstation running Sun OS 5.8, and all programs were compiled using the Sun WorkShop 6 update 2 Fortran 95 compiler.

5.1 Experimental comparison of numerical integrators

Four integration subroutines were programmed for the purpose of determining their empirical rate of asymptotic convergence and their accuracy using a small

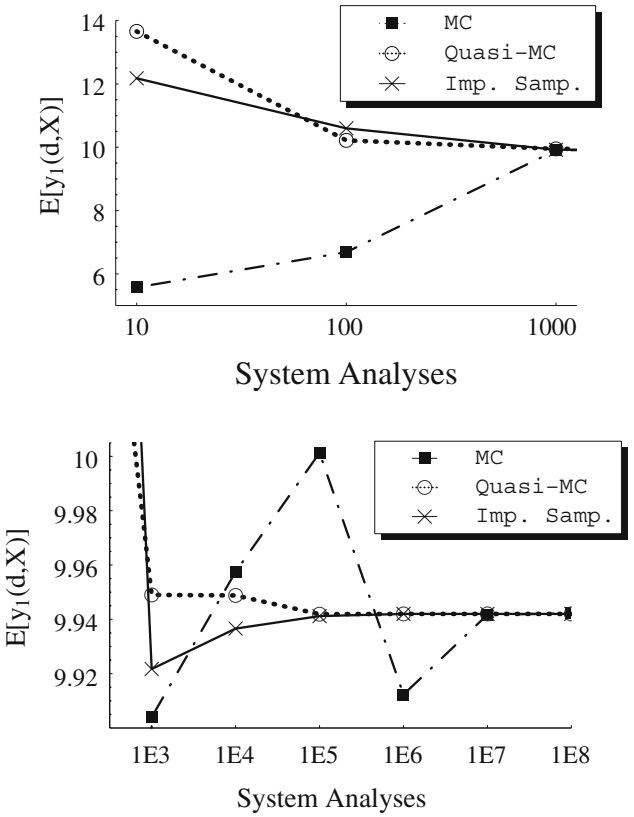

Fig. 3 Comparison of Monte Carlo, importance sampling, and quasi-Monte Carlo integrators. The *horizontal axis* represents the number of system analyses used to compute the expected value $E[y_1(d, X)]$ from problem 2 given a fixed design $d = (0, 5)^t$, and the *vertical axis* represents the computed value of $E[y_1(d, X)]$ from problem 2. The *top figure* depicts the performance of the integrators using up to 1,000 system analyses, and the *bottom figure* depicts the performance of the integrators using greater than or equal to 1,000 system analyses

number of function evaluations ($\leq 10^3$). Two of the integration subroutines were chosen to be stochastic and two deterministic.

The stochastic integration subroutines used were a basic Monte Carlo implementation and an importance sampling Monte Carlo implementation. The probability density function for the importance sampling subroutine was constructed using the techniques described in O'Leary (2004) and Beichl and Sullivan (1999).

The deterministic integration subroutines used in this paper were a quasi-Monte Carlo implementation based on the multidimensional Sobol' deterministic sequence (Bratley and Fox 1988) and a Newton–Cotes integrator based on a product rule of one-dimensional adaptive Newton–Cotes integrators. [The Newton–Cotes integrator actually contains two separate product rule integrators. The first, which is used to integrate the continuous function $y_1(d, X)$ in the objective function, is based on an eight-panel (nine-point) adaptive Newton–Cotes rule, and the second, which is used to evaluate the discontinuous function $f(d, X)$, is based on a two-panel (three-point) adaptive Newton–Cotes rule.]

Figures 3 and 4 compare the accuracy of the Monte Carlo, quasi-Monte Carlo, and importance sampling integrators when used to evaluate $E[y_1(d, X)]$ from problem 2 for fixed $d = (0, 5)^t$. The asymptotic convergence rates of the three Monte Carlo-based methods, shown in Fig. 3, coincide with the theoretical expectations of their relative performances considered in

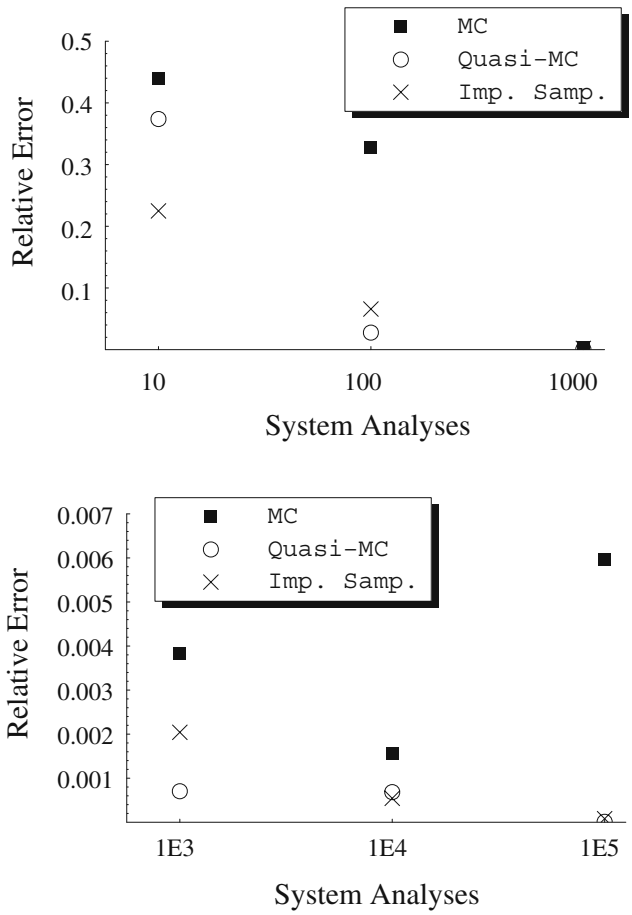

Fig. 4 Error comparison of Monte Carlo, importance sampling, and quasi-Monte Carlo integrators. The *horizontal axis* represents the number of system analyses used to compute the expected value $E[y_1(d, X)]$ from problem 2 given a fixed design $d = (0, 5)^t$, and the *vertical axis* represents the relative error of the computed expectation from the "true" value of $E[y_1(d, X)]$, which is 9.9420. The *top figure* depicts the performance of the integrators using up to 1,000 system analyses, and the *bottom figure* depicts the performance of the integrators using greater than or equal to 1,000 system analyses

Section 3. The unpredictability of the basic Monte Carlo method is apparent in Figs. 3 and 4 because its relative error is three times greater for 10^5 function evaluations than for 10^4 function evaluations. Importance sampling and quasi-Monte Carlo obtain a relative error less than 10^{-3} after 10^4 function evaluations, whereas Monte Carlo requires 10^7 function evaluations for the same level of accuracy. By contrast, the Newton–Cotes eight-panel integrator obtains a relative error less than 10^{-4} after only 1,089 function evaluations. A systematic study of eight-panel Newton–Cotes accuracy is summarized in Table 1 of the Appendix.

Figures 5 and 6 compare the accuracy of the Monte Carlo, quasi-Monte Carlo, and importance sampling integrators when used to evaluate $E[f(d, X)]$ from problem 2 given an initial design of $d = (0, 5)^t$. The discontinuous nature of $f(d, X)$ introduces the theoretical possibility that the quasi-Monte Carlo and Newton–Cotes integrators will perform poorly; however, the empirical evidence suggests that quasi-Monte Carlo and the two-panel Newton–Cotes integrators continue to perform at least as well as the simple Monte Carlo approach in spite of discontinuities present in $f(d, X)$. Caflisch (1998) makes a similar observation regarding the loss of quasi-Monte Carlo effectiveness, stating that computational experience almost always demonstrates the superiority of the quasi-Monte Carlo method relative to stochastic Monte Carlo. All methods converged more slowly to the true value of $E[f(d, X)]$ than to the true value of $E[y_1(d, X)]$; however, the

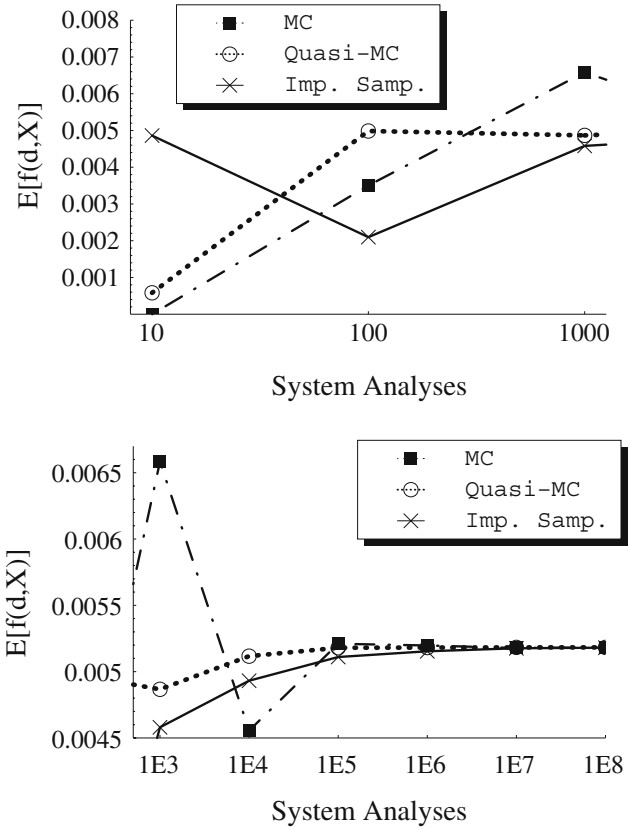

Fig. 5 Comparison of Monte Carlo, importance sampling, and quasi-Monte Carlo integrators. The *horizontal axis* represents the number of system analyses used to compute the expected value $E[f(d, X)]$ from problem 2 given a fixed design $d = (0, 5)^t$, and the *vertical axis* represents the computed value of $E[f(d, X)]$ from problem 2. The *top figure* depicts the performance of the integrators using up to 1,000 system analyses, and the *bottom figure* depicts the performance of the integrators using greater than or equal to 1,000 system analyses

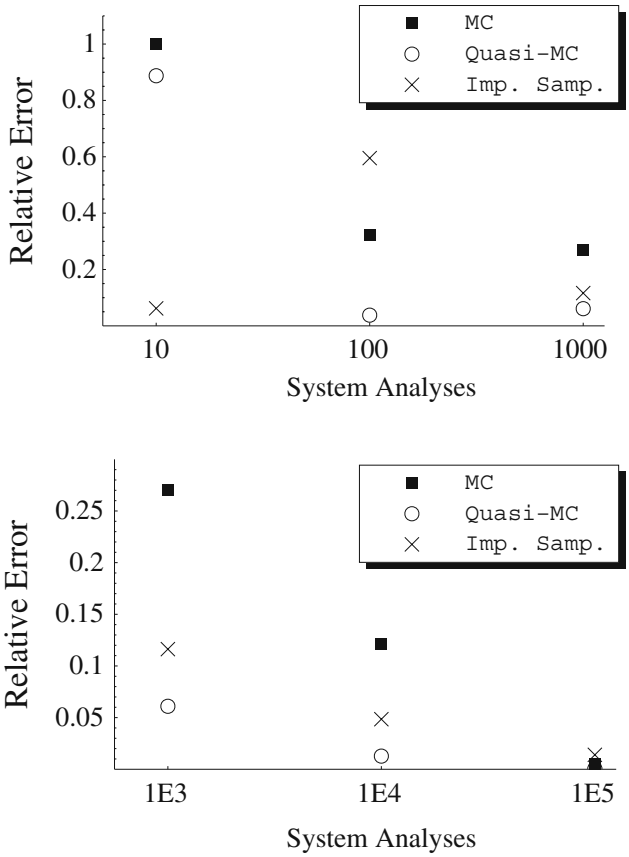

Fig. 6 Error comparison of Monte Carlo, importance sampling, and quasi-Monte Carlo integrators. The *horizontal axis* represents the number of system analyses used to compute the expected value $E[f(d, X)]$ from problem 2 given a fixed design $d = (0, 5)^t$, and the *vertical axis* represents the relative error of the computed expectation from the "true" value of $E[f(d, X)]$, which is $5.1828 \cdot 10^{-3}$. The *top figure* depicts the performance of the integrators using up to 1,000 system analyses, and the *bottom figure* depicts the performance of the integrators using greater than or equal to 1,000 system analyses

disparity between the convergence rate of Monte Carlo and the other methods was also smaller. A systematic study of two-panel Newton–Cotes accuracy is summarized in Table 2 of the Appendix.

5.2 Experimental comparison of gradient-based optimizers

Two optimizers, a sequential quadratic programming (SQP) algorithm and a modified method of feasible directions (MMFD) algorithm, were used in the experiments. Implementations of both algorithms are included in version 4.0 of the Design Optimization Tools (DOT) software distributed by Vanderplaats Research & Development. See Bazaraa et al. (1993) for a description of SQP and MMFD. See Haim et al. (1998) for a comparison of DOT to other optimizers for solving MDO problems. DOT's default parameters were adjusted to facilitate optimization runs using crude expected value calculations. The relative and absolute finite difference step sizes were increased from their default values to 0.05 because experience with problems 1 and 2 demonstrated the need for a larger finite difference step size when low-accuracy expected value calculations are used. The number of consecutive iterations for which the convergence criteria must be met was increased from two to three, decreasing the chances of spurious convergence when crude objective function values are used.

The results of the optimizer trials can be seen in Tables 3 and 4 of the Appendix. For the stochastic integrators (Monte Carlo and importance sampling), each row represents the average of ten separate optimizer trials. A positive number in the failure column indicates that one or more of the trials terminated due to a failure condition reported by DOT. Failure resulted either from an inability to produce a feasible design or from a zero constraint gradient in the infeasible region. Statistics from a failed trial were not used in the calculated row averages. The feasibility of a given final design was evaluated using a Newton–Cotes integrator with a requested error of 10^{-12} and a constraint tolerance of 10^{-3}.

5.2.1 Problem 1 optimizer results

The SQP optimizer was completely unable to cope with the flat infeasible regions of problem 1 regardless of the supplied initial design. Ironically, the only SQP trials that did not fail were those corresponding to low-accuracy, stochastic expected value calculations. In these cases, the SQP optimizer tended to be less ambitious around the boundary between feasibility and infeasibility (perhaps due to the extreme variability in the low-accuracy stochastic results), so the optimizer was less likely to jump into the infeasible region and get trapped by a zero-gradient condition.

By contrast, the MMFD optimizer never terminated on a failure condition given a feasible initial design, though, given an infeasible initial design, the MMFD optimizer also failed due to a zero-constraint gradient. Furthermore, due to the mostly flat feasible region, all of the integrators resulted in termination a small distance from a local optimum point; however, when the final designs were evaluated for feasibility using a high-accuracy integrator, it was found that the stochastic methods produced infeasible designs with a high probability. It is interesting to note that increasing the number of function evaluations did not translate into a proportional increase in the number of feasible designs produced by a stochastic method; on the contrary, after a certain threshold number of function evaluations, the additional accuracy seemed to actually hinder the ability of the optimizer to produce feasible designs. A summary of the optimizer results for problem 1 given the initial designs $d = (-10, 10)^t$ appears in Table 3 of the Appendix. (Similar results appear in the appendix of Kugele et al. (2006) for the initial design $d = (10, 6)^t$.)

5.2.2 Problem 2 optimizer results

Table 4 shows the superiority of the deterministic integrators for solving problem 2 when combined with either optimizer, with respect to avoiding failure and returning feasible designs near a locally optimal point.

The majority of failures occurred due to an inability to find the feasible region, perhaps resulting from premature convergence in the infeasible region or some other reason that prevented the optimizers from making progress. It is interesting to note that, even though the constraint was designed to have no flat regions and, thus, prevent failures due to zero constraint gradients, this failure condition was not completely eliminated in problem 2 when using Monte Carlo methods with a small number of function evaluations (≤ 100).

When combined with stochastic integrators, the SQP optimizer often exhibited extraordinarily erratic behavior, which can be seen in Fig. 7. Increasing the number of function evaluations did not eliminate the erratic

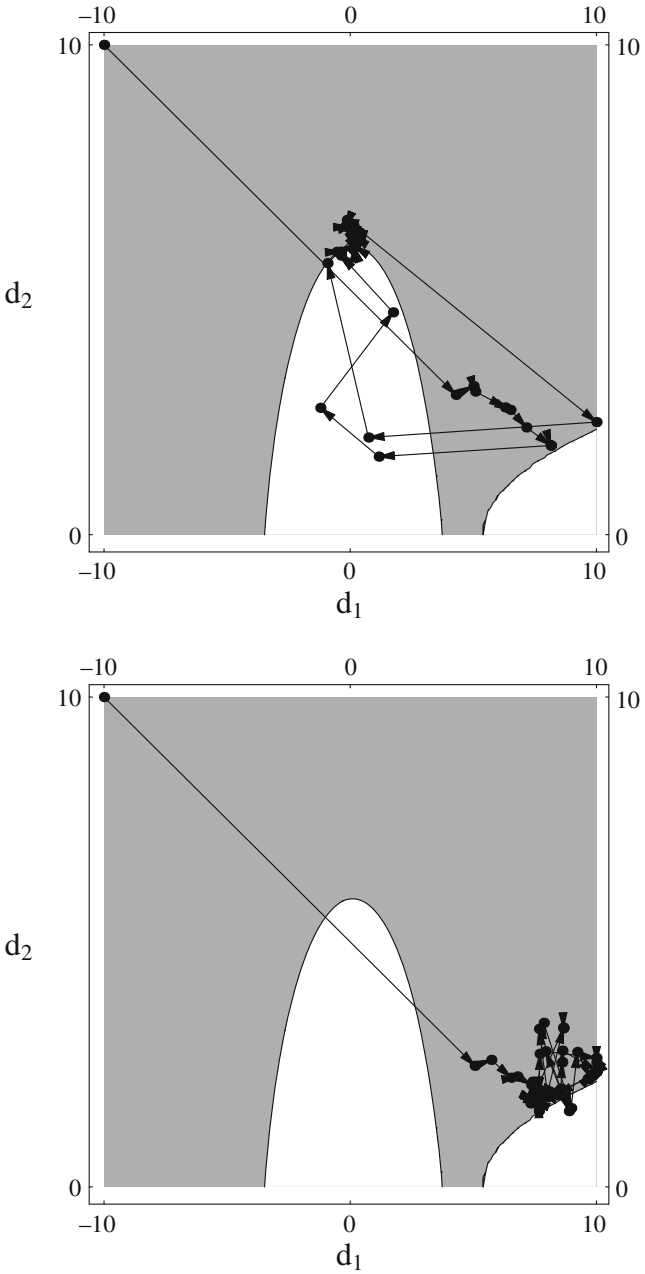

Fig. 7 Intermediate designs visited by the DOT SQP optimizer for problem 2 using Monte Carlo with 10^5 function evaluations (*top*) and importance sampling with 10^3 function evaluations (*bottom*)

intermediate designs; however, it did reduce the area over which the intermediate designs exhibited erratic jumps. The MMFD optimizer did not exhibit similar behavior.

Spurious local convergence was also a problem for the optimizers when using stochastic integrators, especially when the integrators were restricted to small numbers of function evaluations. [Figures C and G of the appendix contained in Kugele et al. (2006) show this phenomenon in more detail.]

6 Discussion

The uncertainties that arise in robust design optimization can be managed by the tools of statistical decision theory, specifically the Bayes principle. While conceptually elegant, this formulation of robust design optimization leads to objective and constraint functions that are difficult to evaluate because they involve numerical integration. An engineer who is confronted with such a problem may attempt an obvious solution: use off-the-shelf optimization software that calls user-written objective and constraint functions that utilize a standard numerical integration technique. This work explored what an engineer might expect to accomplish with such an approach.

Two prototype robust design problems that include two design variables (d) and two uncertain quantities (X) are considered here. Algorithm performance is defined in terms of a "coupled analysis" $y(d, X)$. In actual design problems, the coupled analysis is expensive to compute; hence, quantities that depend on y should be considered expensive and the number of times that y must be evaluated is a reasonable measure of total expense.

Robust design optimization is inherently expensive. Numerical optimization requires evaluation of the objective and constraint functions at multiple designs. Each evaluation involves numerical integration. As illustrated in Section 5.1, a single accurate evaluation may require thousands of coupled analyses, far too expensive for a typical application. Thus, in specifying an optimization strategy, the engineer should anticipate that searches will be based on inaccurate information. This reality has two important implications. First, one should not expect to obtain precise solutions to robust design optimization problems. Second, to whatever extent may be possible, searches should be insensitive to errors in function evaluation.

The DOT software distributed by Vanderplaats Research & Development is widely used by engineers. In this study, DOT plays the role of canonical off-the-shelf optimization software, developed for deterministic nonlinear programming and used in circumstances (inaccurate function values) for which it was not intended. Perceived failures of DOT to solve the prototype robust design problems should be attributed

to the inherent difficulty of robust design optimization, without inferring that DOT is deficient for its intended purpose.

When using DOT for robust design optimization, one should endeavor to choose code input parameters for which DOT is relatively insensitive to inaccurate function values. Assuming that derivative information is not available, DOT will attempt to approximate derivatives by finite differencing. DOT is designed to accommodate inaccurate function values, basing its finite difference step size on an input parameter specifying the function noise level. In the context of Monte Carlo integration, this noise level is difficult to estimate accurately. Finite difference step sizes below the noise level result in useless search directions.

It is not clear a priori whether reliability constraints are better managed by SQP (DOT's SQP implementation) or by a feasible direction method (DOT's MMFD implementation). Reliability constraints are likely to induce flat infeasible regions (corresponding to designs that certainly fail) and flat feasible regions (corresponding to designs that certainly do not fail). The results suggest that MMFD outperforms SQP in such situations.

Beyond choosing code parameters that correctly account for inaccurate function values, it may be possible to adopt numerical integration strategies that facilitate optimization. By fixing the points at which the integrands are evaluated, deterministic integrators regularize the robust design problem, effectively replacing integrals with sums. Indeed, significantly better results were obtained with deterministic integrators than with stochastic integrators. However, as noted by Huyse and Lewis (2001), the potential difficulty with this approach is that solutions to the regularized problem may be unsatisfying. Huyse and Lewis recommend varying the points used by the deterministic integrator, which is, in fact, done here because the Newton–Cotes integration algorithms are adaptive.

The results beg numerous questions. Supposing that an engineer can afford only a fixed number of coupled analyses, which ones should be performed? Is it better to spend one's limited budget obtaining relatively accurate function values and rely on a relatively small number of optimization iterations, or is it better to rely on relatively inaccurate function values, thereby purchasing a larger number of optimization iterations? The results suggest that a simple answer is impossible. Performance improves as the number of coupled analyses per numerical integration increases, but relatively small numbers are sometimes adequate to obtain good approximate solutions. Such tradeoffs are problem- and algorithm-specific. It seems natural to pursue an adaptive search strategy, starting with a small number of analyses per integration, then restarting the search with a larger number after the initial search terminates, and so on, ideally reusing all the earlier obtained coupled analyses (via surrogates, e.g.). Subsequent work will study the efficacy of such strategies.

A common feature of all numerical integration schemes is that more work is required for more accurate approximations of (2). Thus, numerical integration schemes provide models of varying fidelity, suggesting that certain ideas useful in surrogate-based optimization might be adapted for the purpose of *surrogate-based integration*.

Because of the expense of high-fidelity simulation, surrogates are playing an increasingly prominent role in engineering design optimization. Unlike the problems addressed by classical approximation theory, modern large-scale engineering design problems are typified by a large number of dimensions and a relative paucity of exact/accurate function values. In this context, most of the work on surrogate construction has been motivated by the concerns of optimization. Consider a surrogate construction with a different goal: that a definite integral of the surrogate approximate a definite integral of the true underlying function. This goal may dictate radically different surrogate construction strategies than do more traditional goals of pointwise function approximation and optimization. This paper concludes with descriptions of two plausible approaches to surrogate-based integration to be studied in subsequent work.

6.1 Surrogate-based adaptive quadrature

The first approach attempts to modify traditional strategies for adaptive quadrature to exploit information about the integrand provided by surrogates. Suppose that $h : S \to \Re$ and that $s : S \to \Re$ is a surrogate for h. Consider the approximation

$$\int_S h(x)\,dx \approx \int_S s(x)\,dx.$$

This problem does not arise in traditional numerical integration: if evaluating h is no more expensive than evaluating s, then computational resources dedicated to integrating h are better spent evaluating h than constructing and evaluating s. However, what if evaluating s is substantially less expensive than evaluating h?

A natural challenge is to modify adaptive quadrature algorithms based on Newton–Cotes and Gauss–Kronrod formulas to obtain new adaptive quadrature algorithms that are exact for various families of surrogate functions, rather than for polynomials. To exploit these algorithms, it will be necessary to develop new optimization methods that adaptively sample the design parameters with the goal of efficient optimization *and* the noise variables with the goal of efficient integration. Recent work (Pérez et al. 2002) indicates that adaptive sampling is more efficient than static sampling for optimization. Respecting both goals poses interesting theoretical challenges and potentially offers greatly improved robust design strategies.

6.2 Taylor approximations

Whatever the reduction in computational expense from using surrogate-based adaptive quadrature schemes is, some robust design problems will remain prohibitively expensive. Such problems require crude approximations of the objective function (2). A thorough study of what can be accomplished with Taylor approximations – the second approach – is indicated.

Given $g : \Re^k \to \Re$, let $\hat{x} = \text{argmin}\, g(x)$ and $H = \nabla^2 g(\hat{x})$. Laplace's approximation is

$$\int_{\Re^k} \exp\left[-g(x)\right] dx \approx \frac{\exp\left[-g\left(\hat{x}\right)\right]}{\sqrt{\det(H/2\pi)}}.$$

To approximate (2), let $g(a; b) = -\log f(a; b) - \log p(b)$. Laplace's approximation applies *if* one can minimize g in b and estimate H.

Laplace's approximation has made tractable certain integrations that arise in Bayesian statistics (see Tierney and Kadane 1986; Kass et al. 1989; Wong and Li 1992), but it has not yet been applied to robust design. The difficulty of minimizing g in b and estimating H may render Laplace's approximation unsuitable for engineering problems. However, Huyse and Lewis (2001) obtained promising results for the problem of designing an airfoil that performs well over a range of different Mach numbers by replacing f with its second-order Taylor polynomial, expanded about

$$\bar{b} = \int_B b p(b)\, db.$$

This leads to the approximation

$$\phi(a) \approx f\left(a; \bar{b}\right) + c\nabla^2 f\left(a; \bar{b}\right),$$

where

$$c = \frac{1}{2}\int_B \left\| b - \bar{b} \right\|^2 p(b)\, db.$$

This approach deserves greater scrutiny.

Acknowledgements This work was supported in part by National Science Foundation Grants DMI-0422719, DMI-0355391, DMI-0355362 and Department of Energy Grant DE-FG02-06ER25720.

Appendix

Table 1 Newton–Cotes integrator results using an eight-panel adaptive product rule to evaluate $E[y_1(d, X)]$ from problem 2 given an initial design $d = (0, 5)^t$

Requested error	Estimated error	System analyses	$E[y_1(d, X)]$
1E-2	3.66E-5	1,089	9.9425029
1E-4	3.66E-5	1,089	9.9425029
1E-6	5.17E-6	1,617	9.9419921
1E-8	2.55E-8	4,113	9.9419941

The two-dimensional error estimate was obtained using the one-dimensional error estimates and a technique described in Kahaner et al. (1989)

Table 2 Newton–Cotes integrator results using a two-panel (Simpson's) adaptive product rule to evaluate $E[f(d, X)]$ from problem 2 given an initial design $d = (0, 5)^t$

Requested error	Estimated error	System analyses	$E[f]$
1E-2	-4.46E-6	1,413	5.1834225E-3
1E-4	-9.96E-9	7,741	5.1828158E-3
1E-6	3.80E-10	35,841	5.1828190E-3
1E-8	-3.57E-12	204,069	5.1828190E-3

The two dimensional error estimate was obtained using the one dimensional error estimates and a technique described in Kahaner et al. (1989)

Table 3 Optimizer statistics for problem 1 given an initial design $d = (-10, 10)^t$ for DOT MMFD and DOT SQP

Integrator	Failures	Reported optimum [*Mean*(σ)]	System analyses [*Mean*(σ)]	Optimizer calls [*Mean*(σ)]	Percentage feasible
DOT (MMFD)					
Monte Carlo 1E+01	0	25.05(0.39)	1.01E+03(2.81E+02)	49.50(14.03)	0.00%
Monte Carlo 1E+02	0	25.76(0.11)	1.05E+04(1.98E+03)	51.40(9.91)	20.00%
Monte Carlo 1E+03	0	25.83(0.04)	9.70E+04(1.90E+04)	47.50(9.48)	30.00%
Monte Carlo 1E+04	0	25.86(0.01)	9.12E+05(2.18E+05)	44.60(10.90)	80.00%
Monte Carlo 1E+05	0	25.86(0.00)	7.84E+06(8.00E+04)	38.20(0.40)	40.00%
MC Imp. Samp. 1E+01	0	25.50(0.65)	1.18E+03(1.80E+02)	57.90(9.02)	10.00%
MC Imp. Samp. 1E+02	0	25.79(0.08)	1.05E+04(2.90E+03)	51.30(14.48)	20.00%
MC Imp. Samp. 1E+03	0	25.81(0.00)	9.70E+04(1.20E+04)	47.50(6.02)	0.00%
MC Imp. Samp. 1E+04	0	25.85(0.00)	7.80E+05(0.00E+00)	38.00(0.00)	0.00%
MC Imp. Samp. 1E+05	0	25.86(0.00)	8.06E+06(2.37E+05)	39.30(1.19)	0.00%
Quasi-MC 1E+01	0	25.39	8.80E+02	43.00	0.00%
Quasi-MC 1E+02	0	25.87	8.80E+03	43.00	100.00%
Quasi-MC 1E+03	0	25.86	7.60E+04	37.00	100.00%
Quasi-MC 1E+04	0	25.86	7.80E+05	38.00	100.00%
Quasi-MC 1E+05	0	25.86	7.40E+06	36.00	100.00%
Newton–Cotes 1E-02	0	25.86	6.08E+04	38.00	100.00%
Newton–Cotes 1E-04	0	25.86	7.93E+04	38.00	100.00%
Newton–Cotes 1E-06	0	25.86	9.47E+04	38.00	100.00%
Newton–Cotes 1E-08	0	25.86	1.21E+05	38.00	100.00%

DOT SQP failed in all cases except Monte Carlo with less than or equal to 1,000 evaluations per integrand and importance sampling with 10 evaluations per integrand, so its rows will be omitted. All optimizer reported failures resulted from a zero-gradient value calculated for the reliability constraint when its value was infeasible

Table 4 Optimizer statistics for problem 2 given an initial design $d = (-10, 10)^t$ and using DOT MMFD and DOT SQP optimizers

Integrator	Failures	Reported optimum [*Mean*(σ)]	System analyses [*Mean*(σ)]	Optimizer calls [*Mean*(σ)]	Percentage feasible
DOT (MMFD)					
Monte Carlo 1E+01	8	66.89(21.36)	8.80E+02(3.20E+02)	43.00(16.00)	0.00%
Monte Carlo 1E+02	4	418.45(205.64)	7.23E+03(3.62E+03)	35.17(18.09)	66.67%
Monte Carlo 1E+03	7	241.90(247.35)	1.09E+05(1.88E+04)	53.67(9.39)	66.67%
Monte Carlo 1E+04	2	108.04(162.28)	1.45E+06(4.55E+05)	71.62(22.74)	0.00%
Monte Carlo 1E+05	1	114.13(165.75)	1.88E+07(5.41E+06)	93.11(27.07)	11.11%
MC Imp. Samp. 1E+01	8	216.28(102.72)	6.30E+02(7.00E+01)	30.50(3.50)	0.00%
MC Imp. Samp. 1E+02	7	164.79(197.34)	9.60E+03(7.48E+02)	47.00(3.74)	0.00%
MC Imp. Samp. 1E+03	4	107.76(174.32)	1.63E+05(4.41E+04)	80.33(22.04)	0.00%
MC Imp. Samp. 1E+04	4	90.00(146.16)	1.77E+06(3.57E+05)	87.33(17.83)	16.67%
MC Imp. Samp. 1E+05	0	64.59(118.72)	1.88E+07(5.21E+06)	93.10(26.05)	10.00%
Quasi-MC 1E+01	1	—	—	—	—
Quasi-MC 1E+02	0	59.70	1.60E+04	79.00	100.00%
Quasi-MC 1E+03	0	25.69	1.12E+05	55.00	100.00%
Quasi-MC 1E+04	0	25.12	1.58E+06	78.00	0.00%
Quasi-MC 1E+05	0	25.20	1.54E+07	76.00	100.00%
Newton–Cotes 1E-02	0	25.18	1.93E+05	73.00	100.00%
Newton–Cotes 1E-04	0	25.19	6.66E+05	77.00	100.00%
Newton–Cotes 1E-06	0	25.19	2.87E+06	77.00	100.00%
Newton–Cotes 1E-08	0	25.19	1.53E+07	77.00	100.00%
DOT(SQP)					
Monte Carlo 1E+01	8	255.27(217.06)	1.22E+03(6.00E+02)	60.00(30.00)	50.00%
Monte Carlo 1E+02	9	39.42(—)	3.60E+04(—)	179.00(—)	100.00%
Monte Carlo 1E+03	7	29.38(3.89)	2.56E+05(1.68E+05)	127.00(84.22)	66.67%
Monte Carlo 1E+04	9	29.68(—)	1.02E+06(—)	50.00(—)	100.00%
Monte Carlo 1E+05	7	293.75(186.21)	1.35E+07(3.92E+06)	66.67(19.62)	66.67%
MC Imp. Samp. 1E+01	9	31.80(—)	4.44E+03(—)	221.00(—)	0.00%
MC Imp. Samp. 1E+02	8	37.79(11.60)	2.10E+04(1.56E+04)	104.00(78.00)	100.00%
MC Imp. Samp. 1E+03	8	210.16(186.50)	1.27E+05(9.00E+03)	62.50(4.50)	0.00%
MC Imp. Samp. 1E+04	5	103.53(157.31)	1.30E+06(4.42E+05)	64.20(22.12)	40.00%
MC Imp. Samp. 1E+05	3	25.01(0.01)	2.47E+07(9.46E+06)	122.43(47.30)	0.00%
Quasi-MC 1E+01	1	—	—	—	—
Quasi-MC 1E+02	1	—	—	—	—
Quasi-MC 1E+03	0	25.63	1.22E+05	60.00	100.00%
Quasi-MC 1E+04	0	25.24	9.20E+05	45.00	100.00%
Quasi-MC 1E+05	0	25.25	9.20E+06	45.00	100.00%
Newton–Cotes 1E-02	0	25.23	8.77E+04	34.00	100.00%
Newton–Cotes 1E-04	0	25.23	3.75E+05	38.00	100.00%
Newton–Cotes 1E-06	0	25.24	1.47E+06	36.00	100.00%
Newton–Cotes 1E-08	0	25.24	7.70E+06	36.00	100.00%

References

Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: theory and algorithms. Wiley, Hoboken

Beichl I, Sullivan F (1999) The importance of importance sampling. Comput Sci Eng 1:71–73

Bratley P, Fox BL (1988) Algorithm 659: Sobol's quasirandom sequence for multivariate quadrature and optimization. ACM Trans Math Softw 14:88–100

Burgee S, Watson L (1997) The promise (and reality) of multidisciplinary design optimization. In: Biegler LT, Coleman TF, Conn AR, Santosa FN (eds) Large-Scale Optimization with Applications, Part II: Optimal Design and Control (1997). Springer-Verlag, New York, pp 301–324

Caflisch RE (1998) Monte Carlo and quasi-Monte Carlo methods. Acta Numer 7:1–49

Davis PJ, Rabinowitz P (1984) Methods of numerical integration, 2nd edn. Academic, New York

Haim D, Giunta AA, Hozwarth MM, Mason WH, Watson LT, Haftka RT (1999) Comparison of optimization software packages for an aircraft multidisciplinary design optimization problem. Des Optim 1:9–23

Hildebrand FB (1956) Introduction to numerical analysis. McGraw Hill, New York

Huyse L, Lewis RM (2001) Aerodynamic shape optimization of two-dimensional airfoils under uncertain conditions. Technical Report 01-01, Institute for Computer Applications in Science & Engineering, NASA Langley Research Center, Hampton

Kahaner D, Moler C, Nash S (1989) Numerical methods and software. Prentice-Hall, Englewood Cliffs

Kass RE, Tierney L, Kadane JB (1989) Approximate methods for assessing influence and sensitivity in Bayesian analysis. Biometrika 76:663–674

Krommer AR, Überhuber CW (1998) Computational integration. SIAM, Philadelphia

Kugele SC, Trosset MW, Watson LT (2006) The cost of numerical integration in statistical decision-theoretic methods for robust design optimization. Technical Report TR-06-27, Computer Science, Virginia Polytechnic Institute and State University, Blacksburg

Nair VN (1992) Taguchi's parameter design: a panel discussion. Technometrics 34:127–161

Natanson IP (1965) Constructive function theory, volume III: interpolation and approximation quadratures. Ungar, New York

Niederreiter H (1992) Random number generation and quasi-Monte Carlo methods. Society for Industrial and Applied Mathematics, Philadelphia

O'Leary DP (2004) Multidimensional integration: partition and conquer. Comput Sci Eng 6:58–62

Pérez VM, Renaud JE, Watson LT (2002) Adaptive experimental design for construction of response surface approximations. AIAA J 40:2495–2503

Piessens R (1983) QUADPACK: a subroutine package for automatic integration. Springer, Berlin Heidelberg New York

Roy RK (1990) A primer on the Taguchi method. Van Nostrand Reinhold, New York

Sacks J, Welch WJ, Mitchell TJ, Wynn HP (1989) Design and analysis of computer experiments. Stat Sci 4:409–435

Sobieszczanski-Sobieski J, Haftka RT (1997) Multidisciplinary aerospace design optimization: survey and recent developments. Struct Multidisc Optim 14:1–23

Tierney L, Kadane JB (1986) Accurate approximations for posterior moments and marginal densities. J Am Stat Assoc 81: 82–86

Unal R, Lepsch RA, Engelund W, Stanley D (1996) Approximation model building and multidisciplinary design optimization using response surface methods. AIAA Paper 96–4044. In: Sixth AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, pp 592–598

Unger ER, Hutchison MG, Rais-Rohani M, Haftka RT, Grossman B (1992) Variable-complexity multidisciplinary design of a transport wing. Inter J Syst Autom Res Appl (SARA) 2:87–113

Welch WJ, Sacks J (1991) A system for quality improvement via computer experiments. Commun Stat Theory Methods 20:477–495

Welch WJ, Yu TK, Kang SM, Sacks J (1990) Computer experiments for quality control by parameter design. J Qual Technol 22:15–22

Wong WH, Li B (1992) Laplace expansion for posterior densities of nonlinear functions of parameters. Biometrika 79: 393–398