Methods for Generating Random Orthogonal Matrices [1]

Alan Genz

ABSTRACT Random orthogonal matrices are used to randomize integration methods for n-dimensional integrals over spherically symmetric integration regions. Currently available methods for the generation of random orthogonal matrices are reviewed, and some methods for the generation of quasi-random orthogonal matrices are proposed. These methods all have $O(n^3)$ time complexity. Some new methods to generate both random and quasi-random orthogonal matrices will be described and analyzed. The new methods use products of butterfly matrices, and have time complexity $O(\log(n)n^2)$. The use of these methods will be illustrated with results from the numerical computation of high-dimensional integrals from a computational finance application.

1 Introduction

The motivation for this work comes from methods for the numerical computation of high-dimensional integrals in the form

$$I(f) = \frac{1}{(2\pi)^{\frac{n}{2}}} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} e^{-\frac{\mathbf{x}^t\mathbf{x}}{2}} f(\mathbf{x}) d\mathbf{x},$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$. Integrals in this form often appear in the application areas of computational physics, statistics and finance. Recently, Genz and Monahan (1998, and Monahan and Genz, 1997) developed a family of stochastic integration rules (or *formulas*) for this type of problem that are based on a spherical-radial transformation. Let $\mathbf{x} = r\mathbf{u}$, with $\mathbf{u}^t\mathbf{u} = 1$, so

$$\begin{aligned} I(f) &= \frac{1}{(2\pi)^{\frac{n}{2}}} \int\limits_{U_n} \int_{0}^{\infty} e^{-\frac{r^2}{2}} r^{n-1} f(r\mathbf{u}) dr d\mathbf{u} \\ &= \frac{1}{(2\pi)^{\frac{n}{2}}} \int\limits_{U_n} (\frac{1}{2}) \int_{-\infty}^{\infty} e^{-\frac{r^2}{2}} |r|^{n-1} f(r\mathbf{u}) dr d\mathbf{u}, \end{aligned}$$

with $U_n = \{\mathbf{u} : \mathbf{u}^t\mathbf{u} = 1\}$.

The stochastic integration rules developed by Genz and Monahan for $I(f)$ in this form were constructed by randomizing products of deterministic *radial* rules and *spherical surface* rules. The radial rules have the form

$$R_{\boldsymbol{\rho}}(h) = \sum_{i=0}^{m} w_i \frac{h(\rho_i) + h(-\rho_i)}{2} \approx T(h) = \frac{1}{(2\pi)^{\frac{n}{2}}} \int_{-\infty}^{\infty} e^{-\frac{r^2}{2}} |r|^{n-1} h(r) dr,$$

[1] To appear in the Proceedings of the MCQMC98 Meeting.

with $\rho_0 = 0$ and $w_i = T(\prod_{j=0,\neq i}^{m} \frac{r^2-\rho_j^2}{\rho_i^2-\rho_j^2})$, so that such a rule has polynomial degree $2m+1$. Genz and Monahan showed how to determine joint probability densities for the radial rule points ρ_i so that averages of random rules $R_{\boldsymbol{\rho}}(h)$ can be used to provide unbiased degree $2m+1$ estimates for $T(h)$.

The spherical surface rules have the form

$$S(s) = \sum_{j=1}^{p} \tilde{w}_j s(\mathbf{u}_j) \approx U(s) = \int_{U_n} s(\mathbf{u}) d\mathbf{u}.$$

Efficient spherical rules are given in the book Stroud (1972) and the book and paper by Mysovskikh (1980, 1981). The polynomial integrating properties of a spherical surface rule in this form are not changed if the integration rule points $\mathbf{u}_j$ are transformed using an $n \times n$ orthogonal matrix Q. If Q is chosen randomly with Haar distribution (see Stewart, 1980) and $S(s)$ has degree d , then averages of the rules

$$S_Q(s) = \sum_{j=1}^{p} \tilde{w}_j s(Q\mathbf{u}_j),$$

provide unbiased degree d estimates for spherical surface integrals.

If the radial and spherical surface rules are combined in product form, then the results are spherical-radial rules for $I(f)$ defined by

$$SR_{Q,\boldsymbol{\rho}}(f) = \frac{1}{2} \sum_{j=1}^{p} \tilde{w}_j \sum_{i=1}^{m} w_i \frac{f(-\rho_i Q\mathbf{u}_j) + f(\rho_i Q\mathbf{u}_j)}{2}.$$

Such a rule has degree d if the S and R rules both have degree d. A stochastic spherical-radial rule has the form

$$\bar{SR}_d(f) = \frac{1}{N} \sum_{k=1}^{N} SR_{Q_k,\boldsymbol{\rho}_k}(f).$$

If the orthogonal matrices Q_k and radial point vectors $\boldsymbol{\rho}_k$ are randomly generated using the correct densities, then $\hat{SR}_d(f)$ is an unbiased degree d stochastic rule for $I(f)$, with the Monte Carlo standard error estimate

$$\hat{\sigma}_E(f) = \Big(\frac{1}{N(N-1)} \sum_{k=1}^{N} \big(SR_{Q_k,\boldsymbol{\rho}_k}(f) - \bar{SR}_d(f) \big)^2 \Big)^{\frac{1}{2}}.$$

The cost of using $\bar{SR}$ rules for high-dimensional applications problems is dominated by the time needed to compute the $1+2Npm$ f values and to generate N random Q matrices for $\bar{SR}$. We assume that the time to compute one f value is as least $O(n)$ because f has n inputs. The most easily implemented SR rules have degree $d \leq 5$. The degree 1 rules (with $w_0 = 0$ and $m = p = 1$) are just simple symmetric Monte Carlo rules that do not require Q's. The most efficient degree 3 rules have $m = 1$ and $p = n$, so the time needed for the f values for each degree 3 SR rule is at least $O(n^2)$. The most efficient degree 5 rules have $m = 2$ and $p = 2n(n-1)$, so the time needed fo the f values for each degree 5 SR rule is at least $O(n^3)$. Standard methods for generating random orthogonal matrices require $O(n^3)$ time for each Q, so the Q generation time will dominate the rule cost for large n for the degree 3 rules and, for some problems, the Q generation time could be a significant part of the rule cost for degree 5 rules.

The remainder of this paper focuses on the problem of efficient generation of random orthogonal matrices. In the next section standard methods are reviewed and some algorithmic improvements are described. In Section 3, a new $O(n\log(n)$ method is described. In the final section, some methods for generating quasi-random orthogonal are considered, and an example application problem from computational finance is used to illustrate the use the methods that have been developed.

2 $O(n^3)$ Random Orthogonal Matrix Methods

Standard methods for generating random orthogonal matrices with Haar distribution are based on the method of Heiberger (1978). With this method, an $n \times n$ matrix X is first generated with entries $x_{ij} \sim Normal(0,1)$. Then a QR factorization ($X = QR$) is computed. This method provides a random Q with correct distribution. Stewart (1980) and Anderson, Olkin and Underhill (1987) refined the basic Heiberger algorithm by developing methods that construct Q directly.

The Stewart (1980) algorithm constructs Q as a product of reflectors.

$$Q = (I - \beta\mathbf{x}_1\mathbf{x}_1^t)\cdots(I - \beta\mathbf{x}_{n-1}\mathbf{x}_{n-1}^t),$$

where $\mathbf{x}_k$ is in the form $\mathbf{x}_k = [0,\ldots,0,*,\ldots,*]^t$, with k-1 0's. and $n-k+1$ nonzero entries that are essentially scaled Normal random numbers. The time to determine Q in product form is only $O(n^2)$, but the time to determine the columns of Q is $O(n^3)$. Let S be an $n \times p$ matrix with columns that are the integration rule points, so that the product QS is a matrix with columns of transformed points. The computation of QS requires $O(pn^2)$ time.

The Anderson, Olkin and Underhill (1987) algorithm constructs Q as a product of rotators.

$$Q = G_{1,2}G_{1,3}\cdots G_{1,n}G_{2,3}\cdots G_{2,n}\cdots G_{n-1,n}D,$$

where D is a diagonal matrix with random $\pm$ entries and the $G_{j,i}$'s are Givens rotators (see Golub and Van Loan, 1996, pp. 208–217) with appropriately chosen angles. This algorithm has the same time complexity as the Stewart algorithm, but one complication with this algorithm is the generation of the angles, which require Beta random variables. A more direct algorithm can be developed by considering the first column of Q, which can be written as

$$Q\mathbf{e}_1 = \pm \begin{bmatrix} \cos(\theta_2)\cos(\theta_3)\cdots\cos(\theta_{n-1})\cos(\theta_n) \\ \sin(\theta_2)\cos(\theta_3)\cdots\cos(\theta_{n-1})\cos(\theta_n) \\ \ddots \\ \sin(\theta_{n-1})\cos(\theta_n) \\ \sin(\theta_n) \end{bmatrix}.$$

This vector is a uniform random vector on U_n, the surface of the n-sphere, so an alternate method for generating the θ's for $G_{1,2}, G_{1,3}, \ldots, G_{1,n}$ is to first generate $\mathbf{x}$ with $x_i \sim Normal(0,1)$, set $\mathbf{u} = \mathbf{x}/||\mathbf{x}||$ and extract the θ's from $\mathbf{u}$ by taking appropriate ratios of components: $\tan(\theta_2) = u_2/u_1, \tan(\theta_3) = \sin(\theta_2)u_3/u_2, \ldots, \tan(\theta_n) = \sin(\theta_{n-1})u_n/u_{n-1}$. The θ's for the other k values can be generated in a similar manner, with a $\mathbf{u}$ vector of length $n-k+1$ needed for $G_{1,k+1}, G_{1,k+2}, \ldots, G_{1,n}$.

A more direct method for generating the $\mathbf{u}$ components, which avoids the use of Normal random numbers, is to use the transformation described in the book by Fang and Wang (1994, pp. 167–169) and paper by Fang and Li (1997). The same transformation will be used in the next section, and for generation of quasi-random Q's, so it is given here. Let $\mathbf{u}$ be defined by

$$u_{n-2i+2} = \sqrt{1 - r_{n-2i}^{\frac{2}{n-2i}}} \prod_{k=1}^{i-1} r_{n-2k}^{\frac{1}{n-2k}} \sin(2\pi r_{n-2i+1})$$

$$u_{n-2i+1} = \sqrt{1 - r_{n-2i}^{\frac{2}{n-2i}}} \prod_{k=1}^{i-1} r_{n-2k}^{\frac{1}{n-2k}} \cos(2\pi r_{n-2i+1})$$

for $i = 1, 2, \ldots l$, where $l = \lfloor \frac{n}{2} \rfloor - 1$, and ending with

$$u_2 = \prod_{k=1}^{l} r_{n-2k}^{\frac{1}{n-2k}} sin(2\pi r_1) \text{ and } u_1 = \prod_{k=1}^{l} r_{n-2k}^{\frac{1}{n-2k}} cos(2\pi r_1),$$

if n is even, or ending with

$$u_3 = (2r_2 - 1) \prod_{k=1}^{l} r_{n-2k}^{\frac{1}{n-2k}}, \;\; u_2 = 2\sqrt{r_2(1-r_2)} \prod_{k=1}^{l} r_{n-2k}^{\frac{1}{n-2k}} sin(2\pi r_1)$$

$$\text{and } u_1 = 2\sqrt{r_2(1-r_2)} \prod_{k=1}^{l} r_{n-2k}^{\frac{1}{n-2k}} cos(2\pi r_1),$$

if n is odd. Fang and Wang show $\mathbf{u}$ is uniform on U_n, if $r_i \sim Uniform(0,1)$, for $i = 1, 2, \ldots, n-1$.

The use of the methods that have been described in this section, for generating random Q's, will be illustrated with a simple integration rule for $U(s)$. Define $S_Q(s) = \frac{|U_n|}{2n}\sum_{j=1}^{n}(s(Q\mathbf{e}_j) + s(-Q\mathbf{e}_j))$, where the points $\mathbf{e}_j$ are the standard Euclidean basis vectors. This rule has degree 3 (Stroud, 1972). Also define

$R_Q(s) = \frac{S_Q(s}{U(s)}$, $\bar{R}(s) = \frac{1}{N}\sum_{i=1}^{N} R_{Q_i}(s)$ and $E(s) = |\bar{R}(s) - 1|$ using random Q_i's. In Table 1, and in the all of the other tables in this paper, test results were produced using double precision with a 433 MHz DEC Alpha workstation, for selected n values. The time, in seconds, given for each n, is the approximate computation time for one Q (determined by averaging the times for N Q's).

Table 1: Test results for Stewart algorithm with $N = 25$.

n	Q Time	$E(x_1^2)$	$E(x_1^4)$	$E(x_1^6)$	$E(x_1^8)$
22	0.001	$1.914{\cdot}10^{-16}$	$1.532{\cdot}10^{-2}$	$3.642{\cdot}10^{-2}$	$6.049{\cdot}10^{-2}$
43	0.005	$4.286{\cdot}10^{-17}$	$5.598{\cdot}10^{-3}$	$1.634{\cdot}10^{-2}$	$3.316{\cdot}10^{-2}$
87	0.024	$1.937{\cdot}10^{-16}$	$3.188{\cdot}10^{-3}$	$8.911{\cdot}10^{-3}$	$1.772{\cdot}10^{-2}$
173	0.141	$8.820{\cdot}10^{-17}$	$2.545{\cdot}10^{-3}$	$7.390{\cdot}10^{-3}$	$1.515{\cdot}10^{-2}$
347	0.956	$4.980{\cdot}10^{-17}$	$6.645{\cdot}10^{-4}$	$2.045{\cdot}10^{-3}$	$4.645{\cdot}10^{-3}$
693	8.068	$4.875{\cdot}10^{-17}$	$5.362{\cdot}10^{-4}$	$1.688{\cdot}10^{-3}$	$3.992{\cdot}10^{-3}$

The rule S_Q has degree 3, so the $E(x_1^2)$ results should be zero, and they are to within machine precision. A similar set of results was produced using the Anderson, Olkin and Underhill (1987) method, except the times were somewhat longer.

Table 2: Test results for modified Stewart algorithm with $N = 25$.

n	Q Time	$E(x_1^2)$	$E(x_1^4)$	$E(x_1^6)$	$E(x_1^8)$
22	0.001	$2.855{\cdot}10^{-17}$	$5.163{\cdot}10^{-2}$	$2.311{\cdot}10^{-1}$	$7.101{\cdot}10^{-1}$
43	0.005	$2.106{\cdot}10^{-17}$	$6.909{\cdot}10^{-2}$	$6.170{\cdot}10^{-1}$	$3.750{\cdot}10^{0}$
87	0.020	$3.152{\cdot}10^{-17}$	$7.202{\cdot}10^{-2}$	$1.345{\cdot}10^{0}$	$1.728{\cdot}10^{1}$
173	0.101	$2.855{\cdot}10^{-17}$	$8.533{\cdot}10^{-2}$	$3.547{\cdot}10^{0}$	$9.884{\cdot}10^{1}$
347	0.554	$3.623{\cdot}10^{-17}$	$5.834{\cdot}10^{-2}$	$5.031{\cdot}10^{0}$	$2.927{\cdot}10^{2}$
693	3.657	$3.557{\cdot}10^{-17}$	$4.969{\cdot}10^{-2}$	$8.596{\cdot}10^{0}$	$1.023{\cdot}10^{3}$

Table 3: Test results for Butterfly algorithm with $N = 25$.

n	Q Time	$E(x_1^2)$	$E(x_1^4)$	$E(x_1^6)$	$E(x_1^8)$
22	0.001	$5.289{\cdot}10^{-17}$	$1.082{\cdot}10^{-2}$	$3.033{\cdot}10^{-2}$	$6.008{\cdot}10^{-2}$
43	0.004	$6.448{\cdot}10^{-17}$	$1.168{\cdot}10^{-2}$	$3.731{\cdot}10^{-2}$	$9.009{\cdot}10^{-2}$
87	0.017	$9.684{\cdot}10^{-17}$	$3.654{\cdot}10^{-3}$	$1.100{\cdot}10^{-2}$	$2.763{\cdot}10^{-2}$
173	0.076	$6.907{\cdot}10^{-17}$	$2.440{\cdot}10^{-3}$	$7.299{\cdot}10^{-3}$	$1.530{\cdot}10^{-2}$
347	0.346	$4.905{\cdot}10^{-17}$	$1.482{\cdot}10^{-3}$	$4.562{\cdot}10^{-3}$	$1.005{\cdot}10^{-2}$
693	1.846	$6.844{\cdot}10^{-17}$	$5.982{\cdot}10^{-4}$	$1.618{\cdot}10^{-3}$	$3.525{\cdot}10^{-3}$

In an attempt to reduce the times, a simple $O(\log(n)n^2)$ method was developed. For this method a random Q was construct from a product of reflectors and random permutation matrices (P_i) in the form $Q = (I - \beta\mathbf{x}_{k_1}\mathbf{x}_{k_1}^t)P_1 \cdots (I - \beta\mathbf{x}_{k_m}\mathbf{x}_{k_m}^t)P_m$. The $\mathbf{x}_{k_i}$ vectors were chosen in the form $[*, \ldots, *]^t$, with components computed as modified $Normal(0,1)$ random numbers, as in the $O(n^3)$ Stewart algorithm, and m chosen to be $3\ln(n)$. It was expected that this algorithm would provide similar results when compared with the Stewart algorithm, but with shorter times. The computation times were shorter, but the large errors in the last columns indicate bias. As a preview to the next section, results are also given for computations using an $O(\log(n)n^2)$ Butterfly matrix algorithm.

3 Butterfly Orthogonal Matrices

Matrices with butterfly structure have traditionally been used for Fast Fourier Transforms. Recently, Parker (1995) used butterfly matrices to precondition linear systems. The butterfly orthogonal matrices that will be used to randomize integration rules are most easily described for the cases where n is a power of two. Let $c_i = \cos(\theta_i)$, $s_i = \sin(\theta_i)$, for $i = 1, \ldots, n-1$ be given, assume $n = 2^k$, with $k > 0$, and define $Q^{(n)}$ by

$$Q^{(n)} =$$

$$\begin{bmatrix} c_1 & -s_1 & 0 & 0 & \ldots & 0 & 0 \\ s_1 & c_1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & c_3 & -s_3 & \ldots & 0 & 0 \\ 0 & 0 & s_3 & c_3 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & c_{n-1} & -s_{n-1} \\ 0 & 0 & 0 & 0 & \ldots & s_{n-1} & c_{n-1} \end{bmatrix} \begin{bmatrix} c_2 & 0 & -s_2 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & c_2 & 0 & -s_2 & 0 & \ldots & 0 & 0 & 0 \\ s_2 & 0 & c_2 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & s_2 & 0 & c_2 & 0 & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & c_{n-2} & 0 & -s_{n-2} & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 & c_{n-2} & 0 & -s_{n-2} \\ 0 & 0 & 0 & \ldots & 0 & s_{n-2} & 0 & c_{n-2} & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 & s_{n-2} & 0 & c_{n-2} \end{bmatrix}$$

$$\times \cdots \times \begin{bmatrix} c_{\frac{n}{2}} & 0 & \ldots & 0 & -s_{\frac{n}{2}} & 0 & \ldots & 0 & 0 \\ 0 & c_{\frac{n}{2}} & 0 & \ldots & 0 & -s_{\frac{n}{2}} & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \ldots & 0 & c_{\frac{n}{2}} & 0 & \ldots & 0 & 0 & -s_{\frac{n}{2}} \\ s_{\frac{n}{2}} & 0 & \ldots & 0 & c_{\frac{n}{2}} & 0 & \ldots & 0 & 0 \\ 0 & s_{\frac{n}{2}} & 0 & \ldots & 0 & c_{\frac{n}{2}} & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \ldots & 0 & s_{\frac{n}{2}} & 0 & \ldots & 0 & 0 & c_{\frac{n}{2}} \end{bmatrix}.$$

A recursive definition for $Q^{(n)}$, starting with $Q^{(1)} = [1]$, is

$$Q^{(2n)} = \begin{bmatrix} Q^{(n)}c_n & -Q^{(n)}s_n \\ \hat{Q}^{(n)}s_n & \hat{Q}^{(n)}c_n \end{bmatrix},$$

where $\hat{Q}^{(n)}$ has the same form as $Q^{(n)}$ except that the c_i and s_i indices are all increased by n. Example butterfly orthogonal matrices are

$$Q^{(4)} = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & c_3 & -s_3 \\ 0 & 0 & s_3 & c_3 \end{bmatrix} \begin{bmatrix} c_2 & 0 & -s_2 & 0 \\ 0 & c_2 & 0 & -s_2 \\ s_2 & 0 & c_2 & 0 \\ 0 & s_2 & 0 & c_2 \end{bmatrix} = \begin{bmatrix} c_1c_2 & -s_1c_2 & -c_1s_2 & s_1s_2 \\ s_1c_2 & c_1c_2 & -s_1s_2 & -c_1s_2 \\ c_3s_2 & -s_3s_2 & c_3c_2 & -s_3c_2 \\ s_3s_2 & c_3s_2 & s_3c_2 & c_3c_2 \end{bmatrix}$$

and

$$Q^{(8)} = \begin{bmatrix} c_1 & -s_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ s_1 & c_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & -s_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & s_3 & c_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_7 & -s_7 \\ 0 & 0 & 0 & 0 & 0 & 0 & s_7 & c_7 \end{bmatrix}$$

$$\times \begin{bmatrix} c_2 & 0 & -s_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_2 & 0 & -s_2 & 0 & 0 & 0 & 0 \\ s_2 & 0 & c_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & s_3 & 0 & c_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_6 & 0 & -s_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_6 & 0 & -s_6 \\ 0 & 0 & 0 & 0 & s_6 & 0 & c_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & s_6 & 0 & c_6 \end{bmatrix} \begin{bmatrix} c_4 & 0 & 0 & 0 & -s_4 & 0 & 0 & 0 \\ 0 & c_4 & 0 & 0 & 0 & -s_4 & 0 & 0 \\ 0 & 0 & c_4 & 0 & 0 & 0 & -s_4 & 0 \\ 0 & 0 & 0 & c_4 & 0 & 0 & 0 & -s_4 \\ s_4 & 0 & 0 & 0 & c_4 & 0 & 0 & 0 \\ 0 & s_4 & 0 & 0 & 0 & c_4 & 0 & 0 \\ 0 & 0 & s_4 & 0 & 0 & 0 & c_4 & 0 \\ 0 & 0 & 0 & s_4 & 0 & 0 & 0 & c_4 \end{bmatrix}$$

$$= \begin{bmatrix} c_1c_2c_4 & -s_1c_2c_4 & -c_1s_2c_4 & s_1s_2c_4 & -c_1c_2s_4 & s_1c_2s_4 & c_1s_2s_4 & -s_1s_2s_4 \\ s_1c_2c_4 & c_1c_2c_4 & -s_1s_2c_4 & -c_1s_2c_4 & -s_1c_2s_4 & -c_1c_2s_4 & s_1s_2s_4 & c_1s_2s_4 \\ c_3s_2c_4 & -s_3s_2c_4 & c_3c_2c_4 & -s_3c_2c_4 & -c_3s_2s_4 & s_3s_2s_4 & -c_3c_2s_4 & s_3c_2s_4 \\ s_3s_2c_4 & c_3s_2c_4 & s_3c_2c_4 & c_3c_2c_4 & -s_3s_2s_4 & -c_3s_2s_4 & -s_3c_2s_4 & -c_3c_2s_4 \\ c_5c_6s_4 & -s_5c_6s_4 & -c_5s_6s_4 & s_5s_6s_4 & c_5c_6c_4 & -s_5c_6c_4 & -c_5s_6c_4 & s_5s_6c_4 \\ s_5c_6s_4 & c_5c_6s_4 & -s_5s_6s_4 & -c_5s_6s_4 & s_5c_6c_4 & c_5c_6c_4 & -s_5s_6c_4 & -c_5s_6c_4 \\ c_7s_6s_4 & -s_7s_6s_4 & c_7c_6s_4 & -s_7c_6s_4 & c_7s_6c_4 & -s_7s_6c_4 & c_7c_6c_4 & -s_7c_6c_4 \\ s_7s_6s_4 & c_7s_6s_4 & s_7c_6s_4 & c_7c_6s_4 & s_7s_6c_4 & c_7s_6c_4 & s_7c_6c_4 & c_7c_6c_4 \end{bmatrix}.$$

The (i,j) entry, $(Q^{(n)})_{i,j}$, in $Q^{(n)}$ can be determined using the formula

$$(Q^{(n)})_{i,j} = \prod_{l=1}^{k} t_{i_l}(\theta_{2^l \lfloor \frac{i-1}{2^l} \rfloor + 2^{l-1}} + j_l \frac{\pi}{2}), \tag{1}$$

where $i-1 = i_1 + 2i_2 + \cdots + 2^{k-1}i_k$, $j-1 = j_1 + 2j_2 + \cdots + 2^{k-1}j_k$, the i_l's and j_l's are 0 or 1, $t_0(\theta) = \cos(\theta)$ and $t_1(\theta) = \sin(\theta)$.

There are various ways to define butterfly Q's for the cases when n is not a power of two. The method chosen here is to define the Q matrices for these cases as modified submatrices of the 2^k cases. If n is not a power of two, let $k = \lceil \log_2(n) \rceil$. Then $Q^{(n)}$ is constructed by taking the product of k $n \times n$ matrices determined from the k matrices that are used for $Q^{(2^k)}$. For each matrix in the product for $Q^{(2^k)}$, the last $2^k - n$ rows and $2^k - n$ columns are deleted, and every c_i in the remaining $n \times n$ matrix, that is in the same column as a deleted s_i is replaced by 1. The effect of this rule on the explicit coefficient formula (1) is to set equal to 1 all of the cosine terms that either have a θ index greater than $n-1$ or were replaced by 1, and to set equal to 0 all of the sine terms that have a θ index greater than $n-1$ or that were associated with a cosine term that was replaced by 1. This construction process is illustrated with

$$Q^{(5)} = \begin{bmatrix} c_1 & -s_1 & 0 & 0 & 0 \\ s_1 & c_1 & 0 & 0 & 0 \\ 0 & 0 & c_3 & -s_3 & 0 \\ 0 & 0 & s_3 & c_3 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_2 & 0 & -s_2 & 0 & 0 \\ 0 & c_2 & 0 & -s_2 & 0 \\ s_2 & 0 & c_2 & 0 & 0 \\ 0 & s_2 & 0 & c_2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_4 & 0 & 0 & 0 & -s_4 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ s_4 & 0 & 0 & 0 & c_4 \end{bmatrix}$$

$$= \begin{bmatrix} c_1c_2c_4 & -s_1c_2 & -c_1s_2 & s_1s_2 & -c_1c_2s_4 \\ s_1c_2c_4 & c_1c_2 & -s_1s_2 & -c_1s_2 & -s_1c_2s_4 \\ c_3s_2c_4 & -s_3s_2 & c_3c_2 & -s_3c_2 & -c_3s_2s_4 \\ s_3s_2c_4 & c_3s_2 & s_3c_2 & c_3c_2 & -s_3s_2s_4 \\ s_4 & 0 & 0 & 0 & c_4 \end{bmatrix}$$

and

$$Q^{(6)} = \begin{bmatrix} c_1 & -s_1 & 0 & 0 & 0 & 0 \\ s_1 & c_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & -s_3 & 0 & 0 \\ 0 & 0 & s_3 & c_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_5 & -s_5 \\ 0 & 0 & 0 & 0 & s_5 & c_5 \end{bmatrix} \begin{bmatrix} c_2 & 0 & -s_2 & 0 & 0 & 0 \\ 0 & c_2 & 0 & -s_2 & 0 & 0 \\ s_2 & 0 & c_2 & 0 & 0 & 0 \\ 0 & s_2 & 0 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_4 & 0 & 0 & 0 & -s_4 & 0 \\ 0 & c_4 & 0 & 0 & 0 & -s_4 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ s_4 & 0 & 0 & 0 & c_4 & 0 \\ 0 & s_4 & 0 & 0 & 0 & c_4 \end{bmatrix}$$

$$= \begin{bmatrix} c_1c_2c_4 & -s_1c_2c_4 & -c_1s_2 & s_1s_2 & -c_1c_2s_4 & s_1c_2s_4 \\ s_1c_2c_4 & c_1c_2c_4 & -s_1s_2 & -c_1s_2 & -s_1c_2s_4 & -c_1c_2s_4 \\ c_3s_2c_4 & -s_3s_2c_4 & c_3c_2 & -s_3c_2 & -c_3s_2s_4 & s_3s_2s_4 \\ s_3s_2c_4 & c_3s_2c_4 & s_3c_2 & c_3c_2 & -s_3s_2s_4 & -c_3s_2s_4 \\ c_5s_4 & -s_3s_4 & 0 & 0 & c_5c_4 & -s_5c_4 \\ s_5s_4 & c_3s_4 & 0 & 0 & s_5c_4 & c_5c_4 \end{bmatrix}.$$

The butterfly Q's are required for the transformation of sets of spherical surface integration rule points. If S is an $n \times p$ matrix with columns that are the integration rule points, then the total time for the product QS is $O(\log(n)np)$. This follows because a) Q has $\lceil log_2(n) \rceil$ factors, and b) the product of each factor with S requires only $O(np)$ floating point operations, since there are at most two nonzero elements in each row of a factor of Q. If $S = I$ the computation of Q as a product requires $O(log(n)n^2)$ time.

If the butterfly Q's are to be used for the transformation of spherical surface integration rules, the angles for the trigonometric functions that are used to construct these matrices need to be randomly chosen so that averages of the transformed rules provide unbiased estimates for the $U(s)$. A sufficient condition for unbiasedness is that the columns of the randomly chosen Q's are uniform on U_n. Therefore, the angles need to be chosen so that the columns of the resulting Q are uniform on U_n. The entries in the first column of $Q^{(n)}$ determine a spherical coordinate system. Although this type of spherical coordinate system is not a standard one, this type of system has been analyzed by Shelupsky (1962) and is complete in the sense that any point on U_n can be represented in terms of $n-1$ angles. The actual angles are not needed for $Q^{(n)}$, but the cosines and sines of the angles are needed, and these factors can be easily determined from the tangents of the angles, which are found by taking ratios of appropriate first column components. An algorithm for generating $Q^{(n)}$ first computes a uniform random point $\mathbf{u}$ from U_n (which can be determined using a normalized Normal vector, or the Fang and Wang algorithm). Then the cosine and sine factors are computed from tangents of the angles which a determined by taking ratios of appropriate $\mathbf{u}$ coordinates. Once all of the sine and cosine factors are available, $Q^{(n)}$ is completely determined in factored form, and can easily be used to transform integration rule points.

In the cases where n is a power of two, it can be seen from the formula (1) for $(Q^{(n)})_{i,j}$ that all of the columns of $Q^{(n)}$ are rotated versions (relative to the Shelupsky spherical coordinate system) of the first column. Therefore, if the first column of $Q^{(n)}$ is uniform on U_n, then so are the other columns. In the cases where n is not a power of two, some of the columns of $Q^{(n)}$ cannot be uniform on U_n because some of the last entries in those columns are zero. One measure of this type of deficiency is what percentage of $Q^{(n)}$ entries are zero. This percentage was computed for $0 < n < 1024$ and it was found that this deficiency percentage fluctuates after starting at zero for each power of two, but the pattern of fluctuation was not determined. The highest percentage was around 14%, and the worst cases occurred at $n = 3, 5, 11, 22, 43, 87, 173, 347, 693$. Following table provides the results from the test that was described at the end of the previous section for these cases.

Table 4: Test results for Butterfly algorithm with $N = 25$, $m = 1$.

n	Q Time	$E(x_1^2)$	$E(x_1^4)$	$E(x_1^6)$	$E(x_1^8)$
22	0.001	$4.377{\cdot}10^{-17}$	$5.015{\cdot}10^{-2}$	$1.620{\cdot}10^{-1}$	$3.840{\cdot}10^{-1}$
43	0.003	$5.589{\cdot}10^{-17}$	$5.570{\cdot}10^{-2}$	$2.963{\cdot}10^{-1}$	$1.257{\cdot}10^{0}$
87	0.014	$5.315{\cdot}10^{-17}$	$4.664{\cdot}10^{-2}$	$2.787{\cdot}10^{-1}$	$1.384{\cdot}10^{0}$
173	0.061	$3.958{\cdot}10^{-17}$	$3.217{\cdot}10^{-2}$	$3.442{\cdot}10^{-1}$	$3.306{\cdot}10^{0}$
347	0.274	$6.711{\cdot}10^{-17}$	$3.138{\cdot}10^{-2}$	$3.533{\cdot}10^{-1}$	$3.734{\cdot}10^{0}$
693	1.375	$6.097{\cdot}10^{-17}$	$2.480{\cdot}10^{-2}$	$3.421{\cdot}10^{-1}$	$6.268{\cdot}10^{0}$

The test results indicate the bias expected from the deficient columns. Additional randomizations can be introduced to correct for this bias. One possible correction replaces Q with QP, where P is a random permutation matrix (a matrix whose columns are randomly permuted columns of the identity matrix), but this type of correction just spreads the bias among all columns of Q. A more effective method of reducing the bias is to replace QP with a product of several, say m, QP's, so that the final Q has the form $(QP)_1(QP)_2\cdots(QP)_m$. The motivation for this technique can be provided by a simple analysis. $(QP)_2$ already has most of its columns uniform on U_n. Multiplication of those columns by the orthogonal matrix $(QP)_1$ does not affect the uniform distribution of those columns. But the matrix multiplication $(QP)_1(QP)_2$ mixes some uniform and some nonuniform columns of $(QP)_1$ to replace nonuniform columns of $(QP)_1$ with columns which should be more uniform. The results in the Table 3 at the end of the last section were produced using $m = 2$. The results in the following table were produced using $m = 3$.

Table 5: Test results for Butterfly algorithm with $N = 25$, $m = 3$.

n	Q Time	$E(x_1^2)$	$E(x_1^4)$	$E(x_1^6)$	$E(x_1^8)$
22	0.001	$7.607 \cdot 10^{-17}$	$1.539 \cdot 10^{-2}$	$3.983 \cdot 10^{-2}$	$7.459 \cdot 10^{-2}$
43	0.005	$6.461 \cdot 10^{-17}$	$7.452 \cdot 10^{-3}$	$2.124 \cdot 10^{-2}$	$3.934 \cdot 10^{-2}$
87	0.019	$7.451 \cdot 10^{-17}$	$3.260 \cdot 10^{-3}$	$9.091 \cdot 10^{-3}$	$1.868 \cdot 10^{-2}$
173	0.088	$6.811 \cdot 10^{-17}$	$1.925 \cdot 10^{-3}$	$6.465 \cdot 10^{-3}$	$1.490 \cdot 10^{-2}$
347	0.413	$6.896 \cdot 10^{-17}$	$9.038 \cdot 10^{-4}$	$2.965 \cdot 10^{-3}$	$7.149 \cdot 10^{-3}$
693	2.306	$9.398 \cdot 10^{-17}$	$5.670 \cdot 10^{-4}$	$1.620 \cdot 10^{-3}$	$3.557 \cdot 10^{-3}$

With respect to accuracy, the results in the Tables 2 and 5 cannot be distinguished, and furthermore, they cannot be distinguished from the results in Table 1, where an $O(n^3)$ algorithm was used to compute Q. It has not be proven that a product of m random Butterfly QP factors will converge as, $m \to \infty$, to $Q \sim$ Haar distribution, but the test results suggest that this is true, and that a small m is sufficient to provide a good approximation.

4 Quasi-Random Orthogonal Matrices and SR Rules

It should be possible to generate quasi-random orthogonal matrices by replacing the random numbers in the algorithms for random orthogonal matrices by appropriately chosen quasi-random numbers. There are many ways that this could be done. For example, consider the Stewart or the Anderson, Olkin and Underhill algorithms for random orthogonal matrices. These algorithms both require approximately $\frac{n^2}{2}$ random numbers to generate one Q. Both algorithms can be implemented using Normal random numbers. A simple method for generating Normal quasi-random numbers is to use $\Phi^{-1}(Quasi(0,1))$, where $\Phi(t)$ is the standard normal distribution function and $Quasi(0,1)$ is a quasi-random number. But this requires the generation of collections of size $\frac{n^2}{2}$ of quasi-random numbers, for large n. In the experiments that are discussed later in this section, the Richtmeyer (see Davis and Rabinowitz, 1984, pp. 482-483) method was used. Another method, based on the use of NT-nets, is described in the paper by Fang and Li (1997), but this is method is not easily implemented for the high-dimensional problems considered here. With the Richtmeyer method, vectors of quasi-random numbers of size M, for large M, are easily generated. The k^{th} vector in the Richtmeyer sequence has the form $\mathbf{v}_k = (\{kp_1\}, \{kp_2\}, \ldots, \{kp_M\})$, where p_i is the i^{th} prime number, and $\{r\}$ denotes the fractional part of r. Quasi-random butterfly orthogonal matrices can be generated with a similar approach. A single quasi-random butterfly orthogonal matrix produced as a product of m quasi-random QP's requires a set of $2m(n-1)$ quasi-random (0,1) numbers, $m(n-1)$ for the Q's and $m(n-1)$ for the P's. Some test results using this method (combined with the Fang and Wang algorithm for the transformation of Quasi(0,1)'s to $\mathbf{u}$'s) are given in Table 6.

Table 6: Test results for Quasi-Butterfly algorithm with $N = 25$, $m = 2$.

n	Q Time	$E(x_1^2)$	$E(x_1^4)$	$E(x_1^6)$	$E(x_1^8)$
22	0.001	$4.828{\cdot}10^{-17}$	$1.094{\cdot}10^{-2}$	$2.827{\cdot}10^{-2}$	$5.159{\cdot}10^{-2}$
43	0.004	$6.582{\cdot}10^{-17}$	$8.897{\cdot}10^{-3}$	$2.513{\cdot}10^{-2}$	$5.554{\cdot}10^{-2}$
87	0.017	$7.535{\cdot}10^{-17}$	$3.567{\cdot}10^{-3}$	$1.079{\cdot}10^{-2}$	$2.452{\cdot}10^{-2}$
173	0.078	$7.883{\cdot}10^{-17}$	$2.863{\cdot}10^{-3}$	$8.838{\cdot}10^{-3}$	$2.006{\cdot}10^{-2}$
347	0.355	$5.580{\cdot}10^{-17}$	$1.324{\cdot}10^{-3}$	$3.480{\cdot}10^{-3}$	$6.827{\cdot}10^{-3}$
693	1.906	$7.225{\cdot}10^{-17}$	$5.814{\cdot}10^{-4}$	$1.761{\cdot}10^{-3}$	$3.979{\cdot}10^{-3}$

If the Q's are used for quasi-stochastic spherical-radial integration rules, a few more quasi-random points are needed for the generation of the quasi-random $\boldsymbol{\rho}$'s. Quasi-random $\boldsymbol{\rho}$'s can be generated by applying the inverses for the distribution functions obtained from the probability densities for the $\boldsymbol{\rho}$'s to the quasi-random (0,1) points.

The use of quasi-stochastic spherical-radial integration rules requires some method for error estimation. The method chosen for the tests, to be described at the end of this section, is to randomize the quasi-stochastic spherical-radial integration rules using a method described by Ökten (1998). Define a quasi-stochastic spherical-radial integration rule $QSR_d^{(k)}(f)$ by

$$QSR_d^{(k)}(f) = \frac{1}{N_Q}\sum_{j=1}^{N_Q} SR_{Q_{k,j}\boldsymbol{\rho}_{k,j}}(f).$$

The set $\{Q_{k,1}, Q_{k,2}, \ldots, Q_{k,N_Q}\}$ is a set of N_Q quasi-random orthogonal matrices. Each matrix requires a set of size $2m(n-1)$ of Quasi(0,1) points. The Ökten method of randomization requires the generation of a relatively small number l (for example, $l = 20$) of extra quasi-random points for each j. This extended set of points is uniform-randomly permuted for each k, and the first $2m(n-1)$ of these "randomized" quasi-random points are used to construct $Q_{k,j}$. A randomized quasi-stochastic spherical-radial integration rule $Q\bar{S}R_d(f)$ with sample size N_R is then defined by

$$Q\bar{S}R_d(f) = \frac{1}{N_R}\sum_{k=1}^{N_R} QSR_d^{(k)}(f).$$

The Monte Carlo standard error estimate for $Q\bar{S}R_d(f)$ is

$$\hat{\sigma}_E(f) = \Big(\frac{1}{N_R(N_R-1)}\sum_{k=1}^{N_R} \big(\ QSR_d^{(k)}(f) - Q\bar{S}R_d(f)\ \big)^2 \Big)^{\frac{1}{2}}.$$

The Monte-Carlo sample size N_R for these rules is usually fixed at some small number (say 10-20) and N_Q is increased until some desired accuracy level is attained.

A high-dimensional application problem from computational finance will now be used to illustrate the use of the methods described in this paper for producing random and quasi-random orthogonal matrices that are used to randomize spherical integration rules. The problem is the mortgage backed securities problem used by Caflisch and Morokoff (1996). Define

$$f(\mathbf{x}) = \sum_{k=1}^{n} \frac{((1-w_k(\mathbf{x})) + w_k(\mathbf{x})c_k)\prod_{j=1}^{k-1}(1-w_j(\mathbf{x}))}{\prod_{j=0}^{k-1}(1+i_k(\mathbf{x}))},$$

where $i_k(\mathbf{x}) = i_0 K_0^k e^{\sigma(x_1+x_2+\cdots+x_k)}$, $K_0 = e^{-\hat{\sigma}^2/2}$, $c_k = \sum_{j=0}^{n-k}(1+i_0)^{-1}$, $\sigma = .02$, $i_0 = .007$, and $w_k(\mathbf{x}) = K_1 + K_2 \tan^{-1}(K_3 i_k(\mathbf{x}) + K_4)$, with (K_1, K_2, K_3, K_4) given constants. $I(f)$ is the present value of a security backed by n month mortgages. Caflisch and Morokoff considered two cases: a) a "nearly linear" case, with $(K_1, K_2, K_3, K_4) = (0.01, -0.005, 10, 0.5)$, and b) a "nonlinear" case, with $(K_1, K_2, K_3, K_4) = (0.04, 0.0222, -1500, 7)$. Tests were carried out using the rules $\bar{SR}_3$, $Q\bar{SR}_3$, $\bar{SR}_5$, $Q\bar{SR}_5$. Results are given for $n = 90$ and $n = 360$. All of tthe results were computed using butterfly orthogonal matrices with $m = 2$.

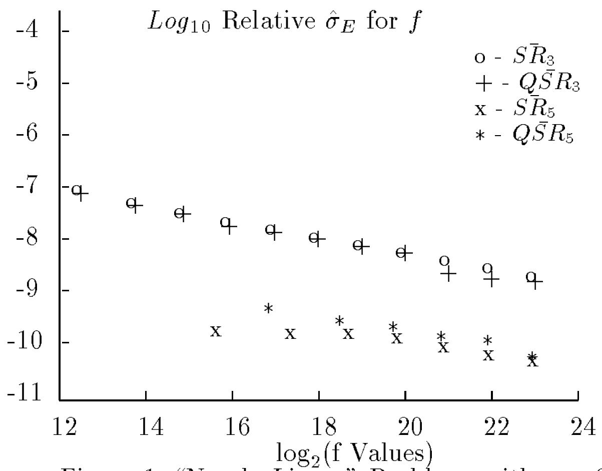

Figure 1: "Nearly Linear" Problem, with $n = 90$.

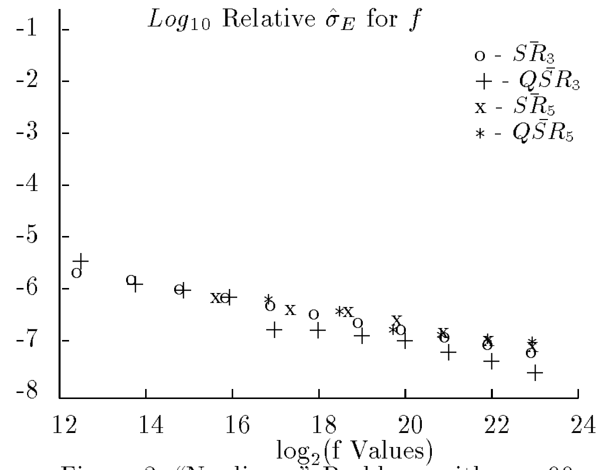

Figure 2: "Nonlinear" Problem, with $n = 90$.

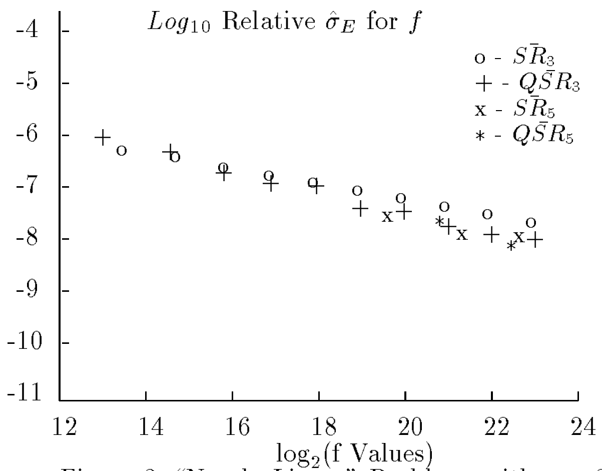

Figure 3: "Nearly Linear" Problem, with $n = 360$.

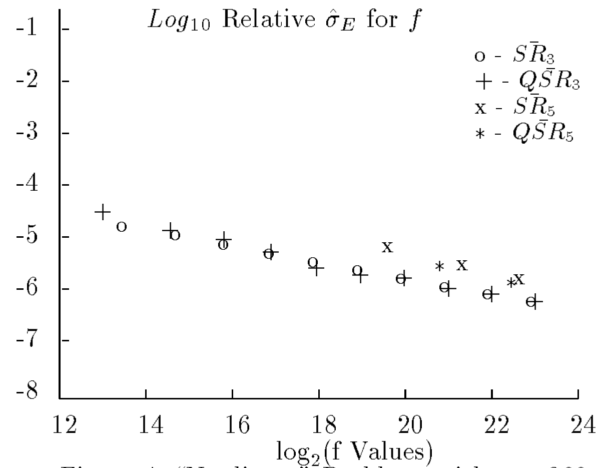

Figure 4: "Nonlinear" Problem, with $n = 360$.

The test results do not clearly demonstrate the superiority of one rule over the others for these problems. The standard errors for the quasi-stochastic degree three rule are somewhat smaller than the stochastic degree three rule standard errors in two of the four cases. But the results do demonstrate that both random and quasi-random butterfly orthogonal matrix algorithms can be feasibly implemented for use with high-dimensional integration algorithms.

5 References

Anderson, T. W., Olkin, I., and Underhill L. G. (1987) *SIAM. J. Sci. Stat. Comput.* **8**, 625–629.

Caflisch, R., and Morokoff, W. (1996), Quasi-Monte Carlo Computation of a Finance Problem, in *Proceedings of the Workshop on Quasi-Monte Carlo*

Methods and Their Applications, Statistics Research and Consultancy Unit, Hong Kong Baptist University, 15–30.

Davis, P. J., and Rabinowitz, P. (1984), *Methods of Numerical Integration*, Academic Press, New York.

Fang, K.-T., and Li, R.-Z. (1997), Some Methods for Generating Both an NT-net and the Uniform Distribution on a Stiefel Manifold and Their Applications, *Comp. Stat. & Data Anal.* **24**, 29–46.

Fang, K.-T., and Wang, Y. (1994) *Number-Theoretic Methods in Statistics*, Chapman and Hall, London, 167–170.

Genz, A., and Monahan J. (1998), Stochastic Integration Rules for Infinite Regions, *SIAM Journal on Scientific Computation*, **19**, 426–439.

Golub, G., and Van Loan, C. (1996), *Matrix Computations*, Johns Hopkins University Press, Baltimore.

Heiberger R. M. (1978), Algorithm AS 127: Generation of Random Orthogonal Matrices, *Appl. Statist.* **27**, 199–206.

Monahan, J., and Genz, A. (1997), Spherical-Radial Integration Rules for Bayesian Computation, *Journal of the American Statistical Association* **92**, 664–674.

Mysovskikh, I. P. (1980), The Approximation of Multiple Integrals by using Interpolatory Cubature Formulae, in *Quantitative Approximation*, R. A. DeVore and K. Scherer (Eds.), Academic Press, New York, 217–243.

Mysovskikh, I. P. (1981), *Interpolatory Cubature Formulas* (Russian), Izd Nauka, Moscow-Leningrad.

Ökten, G. (1998), Error Estimation for Quasi-Monte Carlo Methods, in *Monte Carlo and Quasi-Monte Carlo Methods 1996*, Springer-Verlag, New York, 353–368.

Parker, D. S. (1995), Random Butterfly Transformations with Applications in Computational Linear Algebra, UCLA Computer Science Department Technical Report CSD–950023.

Shelupsky, D. (1962), An Introduction to Spherical Coordinates, *American Mathematical Monthly* **69**, 644–646.

Siegel, A. F. and O'Brien, F. (1983), Unbiased Monte Carlo Integration Methods with Exactness for Low Order Polynomials, *SIAM. J. Sci. Stat. Comput.* **6**, 169–181.

Stewart, G. W. (1980), The Efficient Generation of Random Orthogonal Matrices with An Application to Condition Estimation, *SIAM J. Numer. Anal.* **17**, 403–409.

Stroud, A. H. (1971), *The Approximate Calculation of Multiple Integrals*, Prentice Hall, Englewood Cliffs, New Jersey.

Alan Genz
Department of Mathematics
Washington State University
Pullman, WA 99164-3113 USA

Email: AlanGenz@wsu.edu