Proceedings of the 1989 Winter Simulation Conference
E.A. MacNair, K.J. Musselman, P. Heidelberger (eds.)

IMPLEMENTATION OF QUASI-RANDOM GENERATORS AND THEIR USE IN DISCRETE EVENT SIMULATION

Teresa R. Davenport and Russell C.H. Cheng
School of Mathematics
University of Wales, Cardiff
Senghennydd Road, Cardiff CF2 4AG, U.K.

ABSTRACT

One of the most recent constructions of quasi-random sequences is due to Niederreiter (1988). These sequences, which possess the lowest known discrepancy of all such sequences, have not yet been implemented as practical computer code. In this paper we gather together relevant results and theorems presented by Niederreiter (1988) to produce a concrete construction of such a sequence. An algorithm and practical routines for the generation of the sequence are presented, together with an unusual practical application from the area of discrete event simulation.

We consider the estimation of daily gas demand and conclude that significant gains can be made by selecting a quasi-random sequence in preference to the traditional approach of using crude Monte Carlo.

The Sobol' (1967) and Faure (1982) sequences have played an important role in the development of this, the most recent sequence. They have only marginally higher discrepancies, and so for comparison simulation results are also included for these sequences.

1. INTRODUCTION

Quasi-random sequences first appeared in the 1930's with the publication of the Van der Corput sequence (1935) followed by, amongst others, the Roth (1954), Halton (1960), Sobol' (1967) and Faure (1982) sequences. Niederreiter (1978, 1987, 1988) has recently published several papers on this and related topics. In particular he describes a construction (Niederreiter 1988) which produces sequences with the lowest known discrepancy bounds to date. However, the construction is described in general terms only. Our construction theorem makes use of this general theory to give a specific algorithm and practical implementation.

Before describing this we give a brief outline of the main features of quasi-random sequences. A more detailed account is given by Niederreiter (1978). Quasi-random sequences were specifically designed for the purpose of numerical quadrature for which they produce low error bounds. We may assume the numerical quadrature problem to be the evaluation of the s-dimensional integral, $\int_{I^s} f(t)dt$, where the integration is taken over the unit hypercube I^s. Let $x_1, x_2, \ldots, x_N$ be a set of points distributed in the unit hypercube. Then the integral can be approximated by the quadrature formula,

$$\frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{1.1}$$

The Koksma-Hlawka inequality,

$$\left| \int_{I^s} f(t)dt - \frac{1}{N} \sum_{i=1}^{N} f(x_i) \right| \leq V(f) D_N$$

provides an upper bound on the absolute error, where V(f) is the total bounded variation of f, reflecting the regularity of f, and D_N is the discrepancy of the sequence of points $x_1, \ldots x_N \in I^s$ measuring the uniformity of their distribution in the unit hypercube. The variation will not be discussed further since the influences on the integration error are clearly independent from each other, and it is only the discrepancy which reflects the distribution of the points. A detailed discussion of the variation can be found in Niederreiter (1978).

The discrepancy, a measure of good spacing or uniformity of the points $x_1, \ldots, x_N \in I^s$ can be defined in the following way:

$$D_N = \sup_J \left| \frac{A(J;N)}{N} - Vol(J) \right|$$

where

$$J = \prod_{i=1}^{s} [0,u_i] , \qquad 0 \leq u_i \leq 1$$

A(J;N) is a count of the number of points $x_k \in J$, k=1, ..., N and Vol(J) is the volume of J.

The lower the discrepancy the more uniformly the points are distributed. A quasi-random sequence is designed to have low discrepancy. It is this feature of these sequences which makes their use appear attractive in areas other than numerical integration, such as discrete event simulation. A requirement in discrete event simulation is often that of independence amongst the input variables. However, the low discrepancy of these sequences is due to a specific property known as the net property (Niederreiter 1987), a result of which is that successive terms of the sequence will not be independent. Section 3 describes a method which overcomes this problem.

Since the Van de Corput sequence, quasi-random sequences have been developed with progressively lower discrepancies. A result by Halton (1960) shows that a sequence of N points in I^s can be found for which

$$D_N = O\left[\frac{(\log N)^d}{N}\right] \quad (1.2)$$

where

$$d = s \quad \text{if } N \text{ arbitrary}$$
$$d = s-1 \quad \text{if } N \text{ fixed}$$

This provides a means by which sequences can be compared in the sense that each sequence will have a unique implied constant.

Successive improvements have been made since the Halton construction by Sobol' (1967), Faure (1982), Niederreiter (1987) and Niederreiter (1988). Halton's result impressively shows that a sequence can always be constructed that will do better asymptotically than crude Monte Carlo where the order of convergence is known to be only $O[N^{-\frac{1}{2}}]$.

However, by considering varying values of N for different dimensions, it becomes clear that for dimensions greater than 2, N has to become impractically large for most applications, before Halton's result demonstrates the superiority of a quasi-random sequence compared to crude Monte Carlo. (e.g. for s=5, we need $N \approx 10^{12}$ before the methods are comparable.)

Since the application presented in this paper can be interpreted as a one-dimensional problem this feature of Halton's result is not significant in this case. However, many applications will be of higher dimension, and the crude Monte Carlo method may therefore appear preferable to a quasi-random sequence. It must be stressed however that Halton's result encompasses all definitions of discrepancy (see Niederreiter 1978) and therefore the upper bound on the discrepancy may not be the best possible in any given application. In practice quasi-random sequences may therefore perform considerably better than (1.2) suggests when N is still small. Provided therefore that the dimension is not too large it would seem that quasi-random sequences may be a worthwhile alternative to crude Monte Carlo. Fox (1986) has discussed this aspect using a selected integral and concludes that Sobol's method is preferable to Faure's for dimensions up to 6 but Faure's method is preferable for larger dimensions and that both are preferable to crude Monte Carlo.

The next section includes a description of the background to the construction of a quasi-random sequence as suggested by Niederreiter (1988), and details of the implementation of the sequence. Section 3 discusses a practical application to the estimation of daily gas demand. The appendix contains computer listings of the subroutines required for the generator.

2. CONSTRUCTION OF THE SEQUENCE

Niederreiter (1988) describes general methods for constructing quasi-random sequences. The following theorem gives theoretical details of how a specific sequence may be generated; it is based on the suggestions in §6 of that paper.

Let F_B be a finite field of prime power order B and let $S_B = \{0,1,\ldots, B-1\}$ be the set of digits in base B. The integer n−1 can be written as a number in base B and we denote this by

$$n-1 \equiv b_{d_n} b_{d_{n-1}} \cdots b_o$$

where $b_j \in S_B$, $j = 0,1,\ldots,d_n$. Define also $p_i(x)$, $i = 1,2,\ldots,s$, to be s monic irreducible polynomials belonging to $F_B[x]$, the field of polynomials over F_B, and let e_i be the degree of $p_i(x)$. If we raise a polynomial p(x) to power ℓ and write this as

$$[p(x)]^\ell = x^p + t_{p-1} x^{p-1} + \ldots + t_o$$

then the impulse response sequence corresponding to $[p(x)]^\ell$ is defined to be the initial values $\nu_o = 0$, $\nu_1 = 0$, ..., $\nu_{p-2} = 0$, $\nu_{p-1} = 1$ and the linear recurrence relationship

$$\nu_{p+m} = t_{p-1}\, \nu_{p-1+m} + \ldots + t_o \nu_m, \qquad m = 0,1,\ldots$$

Theorem

The s-dimensional quasi-random sequence $x_n^{(1)}\ x_n^{(2)}\ \ldots\ x_n^{(s)} \in I^s$, $(n = 1,2,\ldots)$ can be generated from $x_n^{(i)} = 0\ .\ a_{1n}^{(i)}\ a_{2n}^{(i)}\ \ldots\ldots$

where

$$a_{jn}^{(i)} = \sum_{r=0}^{d_n} c_{jr}^{(i)} b_r \quad \in S_B, \quad j = 1, 2, \ldots$$

and

$$c_{jr}^{(i)} = \nu_{q+r} \in F_B, \quad r = 0, \ldots, d_n$$

and where ν_{q+r} are elements of the impulse response sequence corresponding to powers ℓ of the monic irreducible polynomial $p_i(x) \in F_B[x]$.

If $e_i = 1$ then $q = 0$ and $\ell = j$ for all j.

If $e_i = 2$ then $q = 0$ and $\ell = (j+1)/2$ for j odd.

If $e_i = 2$ then $q = 1$ and $\ell = j/2$ for j even.

Proof

The theorem is in essence a synopsis of an implementation outlined by Niederreiter (1988). The sequence is defined in equation (4) of that paper. The $a_{jn}^{(i)}$ and b_r of our theorem are precisely the $x_{nj}^{(i)}$ and $a_r(n)$ respectively, defined in the equation immediately preceding (4), where we have selected identity mappings for the bijections λ_{ij} and ψ_r. The $c_{jr}^{(i)}$ of the theorem is unchanged from the $c_{jr}^{(i)}$ as given in the definition of $x_{nj}^{(i)}$. These $c_{jr}^{(i)}$ have to be calculated from equation (7). We have based this calculation on equation (19) in which the $g_{ij}(x)$ of (6) have been set equal to 1. With this choice of $g_{ij}(x)$ the $a^{(i)}(j,k,r)$ of (19) are effectively replaced by the ν's of the impulse response sequence. This allows the $c_{jr}^{(i)}$ to be defined in terms of the ν's directly. We give this calculation explicitly in the theorem, except that we have replaced the (q+1) and u of equation (7) by ℓ and q respectively.

□

To implement the theorem, concrete choices of the base B and the s monic irreducible polynomials are needed.

For a fixed dimension s, the constant C_s which appears in the well known upper bound (see Faure 1982) on the discrepancy:

$$D_N \leqslant C_s \frac{(\log N)^s}{N} + O\left[\frac{(\log N)^{s-1}}{N}\right]$$

depends on both the choice of the base B and the s monic irreducible polynomials. A natural choice would therefore be to select the base and the polynomials so as to minimise C_s. For a given dimension and a value of B, the minimum value of C_s for that particular B is obtained by selecting the s monic irreducible polynomials $p_i(x) \in F_B[x]$, $i = 1, 2, \ldots s$, with degree as small as possible. Applying this criterion to the selection of the polynomials, C_s is then minimised with respect to B. Table 1 gives optimal values of B for dimensions s=1 to 40.

There are exactly B monic irreducible polynomials of degree 1 belonging to $F_B[x]$, of the form $(x + \alpha)$, $\alpha = 0, 1, \ldots, B-1$. Therefore, if $s \leqslant B$ the selection of the $p_i(x)$, $i=1, \ldots, s$ is straightforward. Table 1 illustrates that there are only two cases where $s > B$ for s=1 to 40. For these cases (s=4, B=3 and s=14, B=13), B polynomials are selected with degree 1 and one polynomial is selected with degree 2 ($x^2 + 1$ and $x^2 + 2$ for s = 4 and s = 14 respectively). The values of C_s obtained in this way (as given in Table 1) are the smallest for all quasi-random sequences.

The appendix contains computer listings of the subroutines, written in Fortran 77, which are required to produce the generator. There are 7 subroutines in total; NIEDPOLY is a data block defining the optimal base B for dimensions 1 to 40, and the monic irreducible polynomials required for each dimension; PPFIELD computes addition and multiplication tables for the prime power field of order B; POLYGEN generates all polynomials in the field of order B of degree not greater than n, where $B=p^n$; MATMULT performs matrix multiplication; MATADD performs matrix addition; NIEDSETUP initialises variables and arrays required by the generator; and finally NIEDGEN generates an s-dimensional quasi-random vector.

NIEDTEST is included to illustrate the correct use of the subroutines, and simply generates and displays an s-dimensional sequence of length N.

NIEDSETUP requires three user supplied input parameters DIMEN, NMAX and ERROR; the dimension of the quasi-random vector, the maximum number of calls to be made to the generator, and a flag set to true if either the dimension lies outside the range of 1 to 40 or NMAX exceeds $e^{50 \log_e B}$.

NIEDGEN requires only an array input QUASI which on return from NIEDGEN contains one s-dimensional quasi-random vector. To initialise the generator one call to NIEDSETUP is required at the start of the program followed by repeated calls to NIEDGEN to produce the s-dimensional quasi-random vector.

NIEDTEST illustrates the simplicity of implementing the generator in a simulation program, i.e. the s-dimensional quasi-random vector is generated once for every call to NIEDGEN. This in essence therefore replaces s calls to a pseudo-random number generator. However, the dimension and maximum number of calls to be made to the generator (though this maximum does not have to be reached) must be decided before the simulation. Thus more care is required in planning the simulation. This of course should not be thought of as a disadvantage.

As implied by result (1.2) the dimension should be kept as small as possible, for maximum benefit to be gained from using a quasi-random sequence. Cheng and Davenport discuss the problem of dimensionality in the context of stratified sampling. However, methods proposed in that paper to reduce the dimensionality can equally well be applied to quasi-random sequences.

Table 1 : Optimal Base for Dimensions 1–40

s	C_s	B	s	C_s	B
1	0.721	2	21	0.548×10^{-8}	23
2	0.260	2	22	0.873×10^{-9}	23
3	0.126	3	23	0.133×10^{-9}	23
4	0.086	3	24	0.837×10^{-10}	25
5	0.025	5	25	0.125×10^{-10}	25
6	0.019	7	26	0.776×10^{-11}	27
7	0.004	7	27	0.113×10^{-11}	27
8	0.003	9	28	0.697×10^{-12}	29
9	0.605×10^{-3}	9	29	0.100×10^{-12}	29
10	0.428×10^{-3}	11	30	0.610×10^{-13}	31
11	0.812×10^{-4}	11	31	0.859×10^{-14}	31
12	0.560×10^{-4}	13	32	0.667×10^{-14}	32
13	0.101×10^{-4}	13	33	0.121×10^{-13}	37
14	0.219×10^{-4}	13	34	0.178×10^{-14}	37
15	0.442×10^{-5}	17	35	0.253×10^{-15}	37
16	0.780×10^{-6}	17	36	0.351×10^{-16}	37
17	0.130×10^{-6}	17	37	0.473×10^{-17}	37
18	0.847×10^{-7}	19	38	0.117×10^{-16}	41
19	0.136×10^{-7}	19	39	0.162×10^{-17}	41
20	0.328×10^{-7}	23	40	0.218×10^{-18}	41

3. A PRACTICAL APPLICATION

We consider an application to the estimation of daily gas demand. This was described by Cheng (1984) in the context of applying the antithetic variate method, where details and further references concerning the model are given. We give only a brief outline.

The quantity of interest in the simulation is the cumulative daily gas demanded over 28 daily threshhold levels θ_k ($k = 1, \ldots, 28$) and can be defined as

$$V_k = \sum_{i=1}^{365} \text{Max}\,(d_i - \theta_k, 0), \quad k = 1, \ldots, 28$$

where $d_i = \mu_i + u_i$, $u_i = 0.47u_{i-1} + 122.7\epsilon_i$, $\epsilon_i \sim N(0,1)$, and μ_i are a sequence of calculated quantities dependent on factors such as temperature, chill factor, holidays etc. Tables of θ_k ($k = 1, \ldots, 28$) and μ_i ($i = 1, \ldots, 365$) are given by Cheng (1984).

Traditionally, discrete event simulations have been approached statistically using crude Monte Carlo simulation. In order to utilise a quasi-random sequence, we wish to view the estimation of daily gas demand as the evaluation of an integral. Strictly speaking the problem is a 365-dimensional one (366 for leap years!), there being one dimension for each ϵ_i used in the generation of the u_i sequence. However, the u_i are not very strongly correlated and hence neither are the d_i; thus V_k can be regarded as being the sum of nearly independent quantities. Moreover, if the μ_i are taken to be equal and u_i did not depend on u_i-1, then each d_i depends on one ϵ_i only, and consequently the d_i are independent. The problem can then be viewed as a 1-dimensional one. If we consider d to be a function of $X \sim U(0,1)$, i.e. $d = d(X)$, and define

$$g(X) = \begin{cases} d - \theta & \text{if } d - \theta > 0 \\ 0 & \text{otherwise} \end{cases}$$

then the estimation of daily gas demand can be thought of as the evaluation of the following integral:

$$\frac{E(V_k)}{365} = \int_0^1 g(x)dx$$

The V_k themselves will behave like the sum (1.1) used to estimate an integral, and it would seem attractive to generate the ϵ_i from a quasi-random sequence rather than by crude Monte Carlo.

It should be stressed that, provided the ϵ_i can be regarded as being independent, there is no approximation in replacing crude Monte Carlo by a quasi-random sequence. The above argument, which approximates the problem as the estimation of an integral, merely suggests that it is worthwhile replacing crude Monte Carlo by a quasi-random sequence when the d_i are independent. Lack of independence between the d_i will only weaken the variance reduction but will not invalidate the simulation.

Table 2: Results of 100 blocks of gas demand simulation; block size = 20

Threshold level k	Crude Monte Carlo $\hat{v}_k$	Crude Monte Carlo $\hat{var}(\hat{v}_k)$	Sobol' $\hat{v}_k$	Sobol' $var(\hat{v}_k)$	Faure $\hat{v}_k$	Faure $Var(\hat{v}_k)$	Niederreiter $\hat{v}_k$	Niederreiter $Var(\hat{v}_k)$
2	4.8	0.3	4.7	0.028	4.8	0.016	4.7	0.016
4	6.8	0.4	6.7	0.045	6.7	0.015	6.7	0.019
6	9.2	0.6	9.1	0.060	9.2	0.017	9.1	0.023
8	12.2	0.7	12.1	0.071	12.2	0.024	12.1	0.031
10	16.0	0.8	15.9	0.088	15.9	0.033	15.9	0.038
12	23.4	1.1	23.2	0.101	23.3	0.042	23.2	0.045
14	38.3	1.6	38.1	0.120	38.1	0.081	38.1	0.099
16	64.2	2.7	63.9	0.22	64.0	0.19	63.9	0.18
18	110.2	5.8	109.8	0.23	109.8	0.21	109.8	0.23
20	182.7	10.6	182.1	0.25	182.2	0.16	182.1	0.17
22	554.3	22.8	553.1	0.51	553.3	0.23	553.2	0.28
24	1800.8	57.5	1800.1	0.77	1800.2	0.50	1800.1	0.49
26	4096.2	94.4	4095.2	0.33	4095.3	0.12	4095.2	0.13
28	6787.7	98.1	6786.8	0.301	6786.8	0.059	6786.8	0.066
CPU time (secs)	12.6		18.5		215.9		198.9	

The stochastic input consists of a stream of normal random variates ϵ_i (i = 1, ... 365) which can be generated by the inverse distribution function method,

$$\epsilon_i = F^{-1}(X) , \qquad X \sim U(0,1).$$

For crude Monte Carlo, X is generated using a pseudo-random number generator. For the application of a quasi-random sequence, X is generated using the appropriate quasi-random number generator, i.e. the Faure, Sobol' or Niederreiter generator. (The Sobol' and Faure generators are given by Bratley and Fox 1986, and Fox 1986). The quantities to be estimated are the means of the V_k, and to do this a set of L runs are made from which one estimate of each of the $\overline{V}_k$ (k = 1, ..., 28) is produced. With a quasi-random sequence, the simulation is structured so that each of the L runs is computed simultaneously. Thus $X_1^{(i)}, \ldots, X_L^{(i)}$ are generated in a block from successive terms of the quasi-random sequence to give L values of d_i, for a fixed i. This means that successive daily demands in a given year will be generated from every L'th number in the quasi-random sequence, thereby breaking the requirement that they be generated from mutually independent ϵ_i. (Section 1 discusses briefly the dependence between terms of a quasi-random sequence). To overcome this problem, the $X_1^{(i)}$, ..., $X_L^{(i)}$ are randomly permuted. It can be shown that this reduces the correlation to O(1/L) between pairs of X's. Thus, though we do not have complete independence, we have an approximation to it which is sufficiently accurate for practical applications such as this. To produce an estimate of the variability of the $\overline{V}_k$ the simulation is replicated N times.

Table 2 contains results from the simulation taking L = 20 and N = 100.

All three quasi-random sequences have resulted in substantial improvements compared to crude Monte Carlo, particularly for higher values of k. Both the Niederreiter and Faure sequences have performed comparably, and seem to result in marginally better improvements then the Sobol' sequence. However this must be looked at alongside the timings given at the bottom of Table 2. It is clear that the Sobol' generator is significantly faster than both the Faure and Niederreiter generators. However, all three generators, considering speed as well, have performed significantly better than crude Monte Carlo. We conclude therefore that quasi-random sequences are an important and attractive alternative to the pseudo-random number generator used in crude Monte Carlo.

APPENDIX

```
PROGRAM NIEDTEST

C   PROGRAM ILLUSTRATES THE CORRECT USE OF THE GENERATING
C   SUBROUTINES NIEDSETUP AND NIEDGEN, IE. ONE CALL TO
C   NIEDSETUP TO INITIALISE THE GENERATOR FOLLOWED BY
C   REPEATED CALLS TO NIEDGEN.


      INTEGER B,IS
      DOUBLE PRECISION QUASI(40)
      LOGICAL ERROR(2)

      WRITE(6,*) 'DIMENSION OF THE SEQUENCE ?'
      READ*,IS
      WRITE(6,*) 'LENGTH OF THE SEQUENCE ?'
      READ*,N
      NMAX=N
      CALL NIEDSETUP(IS,NMAX,ERROR)
      IF (ERROR(1)) PRINT*,'DIMENSION NOT ALLOWED: DIMENSION =',IS
      IF (ERROR(2)) PRINT*,'TOO MANY CALLS TO THE GENERATOR:',NMAX
      IF (ERROR(1).OR.ERROR(2)) THEN
           PRINT*,'PROGRAM ABORTED'
           STOP
      ENDIF
      DO 10 I=1,N
           CALL NIEDGEN(QUASI)
           WRITE(6,100),(QUASI(J),J=1,IS)
10    CONTINUE
100   FORMAT(45F8.4)
      END

BLOCK DATA NIEDPOLY

C   POLY(I,J) CONTAINS THE POLYNOMIALS FOR THE OPTIMAL BASE I
C   BASE(I) CONTAINS THE OPTIMAL BASE FOR ALL 40 DIMENSIONS,
C             I=1,..,40

      INTEGER BASE(40),POLY(45,45)
      COMMON /BLK1/BASE,POLY

      DATA (BASE(I),I=1,40)/2,2,3,3,5,7,7,9,9,11,11,13,13,13,17,17,17
     1            ,19,19,23,23,23,23,25,25,27,27,29,29,31,31
     1            ,32,37,37,37,37,37,41,41,41/

      DATA (POLY(2,I),I=1,2)/2,3/
      DATA (POLY(3,I),I=1,4)/3,4,5,10/
      DATA (POLY(4,I),I=1,4)/4,5,6,7/
      DATA (POLY(5,I),I=1,5)/5,6,7,8,9/
      DATA (POLY(7,I),I=1,7)/7,8,9,10,11,12,13/
      DATA (POLY(9,I),I=1,9)/9,10,11,12,13,14,15,16,17/
      DATA (POLY(11,I),I=1,11)/11,12,13,14,15,16,17,18,19,20,21/
      DATA (POLY(13,I),I=1,14)/13,14,15,16,17,18,19,20,21,22,23,24,25
     1                ,171/
      DATA (POLY(17,I),I=1,17)/17,18,19,20,21,22,23,24,25,26,27,28,29
     1                ,30,31,32,33/
      DATA (POLY(19,I),I=1,19)/19,20,21,22,23,24,25,26,27,28,29,30,31
     1                ,32,33,34,35,36,37/
      DATA (POLY(23,I),I=1,23)/23,24,25,26,27,28,29,30,31,32,33,34,35
     1                ,36,37,38,39,40,41,42,43,44,45/
      DATA (POLY(25,I),I=1,25)/25,26,27,28,29,30,31,32,33,34,35,36,37
     1         ,38,39,40,41,42,43,44,45,46,47,48,49/
      DATA (POLY(27,I),I=1,27)/27,28,29,30,31,32,33,34,35,36,37,38,39
     1            ,40,41,42,43,44,45,46,47,48,49,50,51,52
     1            ,53/
      DATA (POLY(29,I),I=1,29)/29,30,31,32,33,34,35,36,37,38,39,40,41
     1            ,42,43,44,45,46,47,48,49,50,51,52,53,54
     1            ,55,56,57/
      DATA (POLY(31,I),I=1,31)/31,32,33,34,35,36,37,38,39,40,41,42,43
     1            ,44,45,46,47,48,49,50,51,52,53,54,55,56
     1            ,57,58,59,60,61/
      DATA (POLY(32,I),I=1,32)/32,33,34,35,36,37,38,39,40,41,42,43,44
     1            ,45,46,47,48,49,50,51,52,53,54,55,56,57
     1            ,58,59,60,61,62,63/
      DATA (POLY(37,I),I=1,37)/37,38,39,40,41,42,43,44,45,46,47,48,49
     1            ,50,51,52,53,54,55,56,57,58,59,60,61,62
     1            ,63,64,65,66,67,68,69,70,71,72,73/
      DATA (POLY(41,I),I=1,41)/41,42,43,44,45,46,47,48,49,50,51,52,53
     1            ,54,55,56,57,58,59,60,61,62,63,64,65,66
     1            ,67,68,69,70,71,72,73,74,75,76,77,78,79
     1            ,80,81/

      END
```

```
SUBROUTINE NIEDSETUP(DIMEN,NMAX,ERROR)

C   THE SUBROUTINE CALCULATES THE IMPULSE RESPONSE SEQUENCE
C   CORRESPONDING TO EACH OF THE POLYNOMIALS POLY(DIMEN,I)
C   I=1,...,DIMEN RAISED TO A POWER J ; J=1,...,DMAX+1.
C   AN(L,M,N) CONTAINS THE IMPULSE RESPONCE SEQUENCE
C   CORRESPONDING TO THE POLYNOMIAL POLY(DIMEN,L) RAISED TO THE
C   POWER M, OF LENGTH N=1,...,DMAX+1.
C      IT CHECKS THAT DIMEN AND NMAX ARE REASONABLE INPUTS
C   AND CALCULATES AN(L,M,N) FOR THESE GIVEN VALUES.
C
C         VARIABLES
C
C   AN(I,J,K) : INITIALLY CONTAINS THE COEFFICIENTS OF POLYNOMIALS
C               P(X)**J, J=1,,,DMAX+1, FOR I=1,,,DIMEN   POLYNOMIALS
C               IN ITS FINAL FORM AN(I,J,K) CONTAINS THE INPULSE
C               RESPONSE SEQUENCE CORRESPONDING TO P(X)**J
C   B         : OPTIMAL BASE
C   BP        : REAL B
C   BASE(40)  : OPTIMAL BASES FOR ALL 40 DIMENSIONS
C   COEFF(I,J): COEFFICIENTS OF POLYNOMIALS I=1,,,DIMEN; P(X)
C   DEG(I)    : DEGREE OF POLYNOMIAL I
C   DI(J)     : REPRESENTION OF NCALL IN BASE B
C   DIMEN     : DIMENSION OF SEQUENCE
C   DMAX      : UPPER BOUND ON NUMBER OF DIGITS IN BASE B
C               REPRESENTATION OF NMAX
C   ERROR     : SET TO FALSE IF EITHER DIMEN<1 OR DIMEN>40 OR
C               DMAX>50.
C   FIELDDADD(I,J)  : ADDITION TABLE FOR FIELD OF PRIME POWER
C                     ORDER B.
C   FIELDMLT(I,J)   : MULTIPLICATION TABLE FOR FIELD OF PRIME POWER
C                     ORDER B.
C   NCALL     : CURRENT NUMBER OF CALLS TO NIEDGEN.
C               INITIALISED TO 1
C   NEWV      : "IMPULSE" TO IMPULSE RESPONSE SEQUENCE
C              ( REF: INTODUCTION TO FINITE FIELDS AND
C                THEIR APPLICATIONS :- R. LIDL AND
C                H. NIEDERREITER. CAMBRIDGE UNIVERSITY
C                PRESS, 1986 )
C   NMAX      : USER SPECIFIED MAXIMUM NUMBER OF CALLS TO NIEDGEN
C   NOD       : NUMBER OF DIGITS IN BASE B REPRESENTATION OF NCALL
C   POLY(I,J) : J=1,... POLYNOMIALS FOR BASE I
C   RECBP(I,J): MULTIPLICATION TABLE

      INTEGER DIMEN,POLY(45,45),COEFF(45,5),DEG(45)
      INTEGER AN(45,0:51,0:51),DMAX,B,BP,DI(0:51)
      INTEGER TEMP(51),V(51),NEWV,NOD,NCALL,NMAX,U
      INTEGER BASE(40),FIELDADD(50,50),FIELDMLT(50,50)
      INTEGER X1,X2,X3,X4,X5,X6,X7
      INTEGER BB
      DOUBLE PRECISION RECBP(45,51)
      LOGICAL ERROR(2)
      COMMON /NIED/IS,B,NCALL,AN,NOD,BP,DMAX
      COMMON /NIED2/DEG,DI
      COMMON /NIED3/RECBP
      COMMON /NIED4/FIELDADD,FIELDMLT
      COMMON /BLK1/BASE,POLY

C   -------------------------------------------------
C   -----    INITIALISE VARIABLES AND PERFORM ERROR ----
C   -----    CHECKS ON USER SUPPLIED INPUT          ----
C   -----    IF ERROR THEN RETURN TO CALLING PROGRAM ---
C   -------------------------------------------------
      IP=0
      IS=DIMEN
      B=BASE(IS)
      RB=REAL(B)
      NCALL=B**IP+1
      BP=B**(IP+1)
      FIRSTN=B**IP
      NOD=1
      DMAX=NINT(LOG(REAL(NMAX+FIRSTN))/LOG(REAL(B)))+1
      ERROR(1)=.FALSE.
      ERROR(2)=.FALSE.
      IF ((DIMEN.GT.40).OR.(DIMEN.LT.1)) ERROR(1)=.TRUE.
      ERROR(2)=DMAX.GT.50
      IF ((ERROR(1)).OR.(ERROR(2))) RETURN
      DI(0)=-1
C   ----------------------------------------------------
C   -----    CALCULATE RESULTS OF MULTIPLICATIONS      ----
C   -----    REQUIRED BY NIEDGEN, SAVING ON FUTURE     ----
C   -----    COMPUTATIONAL TIME                 ----
C   ----------------------------------------------------
      DO 3 I=1,DMAX-1
         DI(I)=0
         DO 4 J=1,B
            RECBP(J,I)=(J-1)*(RB**(-I))
4     CONTINUE
3     CONTINUE
      CALL PPFIELD(FIELDADD,FIELDMLT,B)
C   ----------------------------------------------------
C   -----    FOR EACH OF THE IS POLYNOMIALS; P(X),     ----
C   -----    COMPUTE P(X)**J, J=1,,,DMAX+1             ----
C   ----------------------------------------------------
      DO 1 I=1,IS
         M=0
         J=POLY(B,I)
         DO WHILE (J.GE.B)
```

```
         J=J/B
         M=M+1
      ENDDO
      DEG(I)=M
      J=POLY(B,I)
      AN(I,0,1)=1
      IF (DEG(I).EQ.1) THEN
         INTV=1
         LOOPE=DMAX
         COEFF(I,1)=MOD(J,B)
         DO 20 J=1,DMAX+1
            AN(I,J,1)=1
            X1=AN(I,J-1,J)+1
            X2=COEFF(I,1)+1
            AN(I,J,J+1)=FIELDMLT(X1,X2)
20       CONTINUE
         DO 30 J=1,DMAX+1
            DO 40 K=J+1,DMAX+1
               X1=AN(I,K-1,J+1)+1
               X2=COEFF(I,1)+1
               X3=AN(I,K-1,J)+1
               AN(I,K,J+1)=FIELDADD(X1,(FIELDMLT(X2,X3)+1))
40          CONTINUE
30       CONTINUE
      ELSE
      IF (DEG(I).EQ.2) THEN
         INTV=2
         LOOPE=DMAX+1
         COEFF(I,1)=MOD(J,B)
         COEFF(I,2)=(J-(B*B+COEFF(I,1)))/B
         AN(I,0,1)=1
         AN(I,0,0)=1
         IC=3
         DO 60 J=1,DMAX+1
            AN(I,J,0)=0
            AN(I,J,1)=1
            X1=AN(I,J-1,IC-2)+1
            X2=COEFF(I,1)+1
            AN(I,J,IC)=FIELDMLT(X1,X2)
            AN(I,J,IC+1)=0
            IC=IC+2
60        CONTINUE
         DO 75 J=1,DMAX+1
            X1=AN(I,J-1,1)+1
            X2=COEFF(I,2)+1
            X3=AN(I,J-1,2)+1
            X4=COEFF(I,1)+1
            X5=FIELDMLT(X1,X2)+1
            X6=FIELDMLT(X3,X4)+1
            AN(I,J,2)=FIELDADD(X5,X6)
75        CONTINUE
         DO 70 J=INTV,LOOPE
            DO 80 K=(J-1),J
               DO 90 L=J,DMAX+1
                  X1=AN(I,L-1,J+K-1)+1
                  X2=COEFF(I,2)+1
                  X3=AN(I,L-1,J+K-2)+1
                  X4=COEFF(I,1)+1
                  X5=AN(I,L-1,J+K)+1
                  X6=FIELDMLT(X1,X2)+1
                  X7=FIELDMLT(X3,X4)+1
                  ITEMP=FIELDADD(X6,X7)+1
                  AN(I,L,J+K)=FIELDADD(ITEMP,X5)
90             CONTINUE
80          CONTINUE
70       CONTINUE
      ELSE
         WRITE(6,*),'POLYNOMIAL HAS DEGREE>2.'
         WRITE(6,*),'PROGRAM ABORTED'
         STOP
      ENDIF
      ENDIF
      K=INTV
C  ---------------------------------------------------------------
C  COMPUTE THE IMPULSE RESPONSE SEQUENCE FOR P(X)**J, IE. THE
C  V'S GIVEN IN SECTION 2.
C  ---------------------------------------------------------------
    DO 100 I1=1,DMAX+1
      DO 110 II=1,K-1
         TEMP(II)=0
110   CONTINUE
      TEMP(K)=1
      DO 120 II=1,DMAX+1
         V(II)=TEMP(1)
         NEWV=0
         KK=K+1
         DO 130 JJ=1,K-1
            X1=NEWV+1
            X2=TEMP(JJ)+1
            X3=AN(I,I1,KK)+1
            NEWV=FIELDADD(X1,(FIELDMLT(X2,X3)+1))
            TEMP(JJ)=TEMP(JJ+1)
            KK=KK-1
130      CONTINUE
         X1=NEWV+1
         X2=TEMP(K)+1
         X3=AN(I,I1,2)+1
         TEMP(K)=FIELDADD(X1,(FIELDMLT(X2,X3)+1))
120   CONTINUE
      DO 140 II=1,DMAX+1
         AN(I,I1,II)=V(II)
140   CONTINUE
   IF (K.LT.DMAX) K=K+INTV
100 CONTINUE
1  CONTINUE
   RETURN
   END
```

```
SUBROUTINE NIEDGEN(QUASI)

C   A CALL TO NIEDGEN GENERATES ONE QUASI-RANDOM VECTOR
C   QUASI OF DIMENSION IS.
C   ALL COEFFICIENTS REQUIRED IN THE CALCULATION OF AN ELEMENT
C   OF THE VECTOR HAVE BEEN COMPUTED IN NIEDSETUP, AND STORED
C   IN AN(I,J,K). THEREFORE NIEDGEN SIMPLY SELECTS APPROPRIATE
C   VALUES AND CALCULATES THE AJN(I) AND XN OF THE THEOREM.
C
C          VARIABLES
C
C   XN       : A QUASI-RANDOM NUMBER
C   A        : AJN(I) OF THEOREM
C   EOBCALC  : =TRUE WHEN NCALL CALCULATED IN BASE B
C   QUASI(I): VECTOR OF QUASI-RANDOM NUMBERS
C   ZERO     : =TRUE WHEN IMPULSE RESPONSE SEQUENCE ELEMENTS HAVE
C               BEEN SELECTED FOR POLYNOMIAL P(X)**J
C
    INTEGER C,DI(0:51),BP,NOD,A
    INTEGER AN(45,0:51,0:51),B,DEG(45),DMAX,X1
    INTEGER FIELDADD(50,50),FIELDMLT(50,50)
    LOGICAL ZERO,EOBCALC
    DOUBLE PRECISION RECBP(45,51)
    DOUBLE PRECISION XN,QUASI(40)
    LOGICAL ERROR(2)
    COMMON /NIED/IS,B,NCALL,AN,NOD,BP,DMAX
    COMMON /NIED2/DEG,DI
    COMMON /NIED3/RECBP
    COMMON /NIED4/FIELDADD,FIELDMLT
    IF (NOD.GT.DMAX) THEN
       PRINT*,'NUMBER OF CALLS ON GENERATOR EXCEEDS SPECIFIED
   1 NUMBER'
       STOP
    ENDIF
C   ----------------------------------------------------------
C   NCALL IN BASE B  .......
C   ----------------------------------------------------------
    EOBCALC=.FALSE.
    J=-1
    DO WHILE (.NOT.EOBCALC)
       EOBCALC=.TRUE.
       J=J+1
       DI(J)=DI(J)+1
       IF (DI(J).EQ.B) THEN
          DI(J)=0
          EOBCALC=.FALSE.
       ENDIF
    ENDDO
C   ----------------------------------------------------------
C   COMPUTE QUASI-RANDOM VECTOR
C   ----------------------------------------------------------
    DO 38 LOOP=1,IS
       XN=0
       ZERO=.FALSE.
       J=1
        IF (DEG(LOOP).EQ.1) THEN
           DO WHILE (.NOT.ZERO)
              A=0
              ZERO=.TRUE.
              DO 33 IR=1,NOD
                 C=AN(LOOP,J,IR)
                 X1=FIELDMLT((DI(IR-1)+1),C+1)+1
                 A=FIELDADD(A+1,X1)
                 IF (C.NE.0) ZERO=.FALSE.
 33             CONTINUE
              XN=XN+RECBP((A+1),J)
              J=J+1
              ENDDO
        ELSE
           DO WHILE (.NOT.ZERO)
              A=0
              ZERO=.TRUE.
              IF (MOD(J,2).EQ.0) THEN
                 U=1
                 IQ1=J/2
              ELSE
                 U=0
                 IQ1=(J+1)/2
              ENDIF
              DO 34 IR=1,NOD
                 I2=U+IR
                 C=AN(LOOP,IQ1,I2)
                 X1=FIELDMLT((DI(IR-1)+1),C+1)+1
                 A=FIELDADD(A+1,X1)
                 IF (C.NE.0) ZERO=.FALSE.
 34             CONTINUE
              XN=XN+RECBP((A+1),J)
              IF (U.EQ.0) ZERO=.FALSE.
```

```
          J=J+1
          ENDDO
       ENDIF
    QUASI(LOOP)=XN
 38   CONTINUE
    IF ((NCALL).EQ.BP) THEN
       BP=BP*B
       NOD=NOD+1
    ENDIF
    NCALL=NCALL+1
    RETURN
    END

 SUBROUTINE PPFIELD(FIELDADD,FIELDMLT,BASE)

 C   SUBROUTINE GENERATES ADDITION AND MULTIPLICATION TABLES FOR A
 C   PRIME POWER FIELD OF ORDER BASE,  CHARACTERISTIC P, ORDER B

    INTEGER N,P,BASE,IRRPOLY(5,10)
    INTEGER POLYN,A(50,10,10),SUMA(50,10,10),OLDA(50,10,10)
    INTEGER POLY(0:50,0:50),FIELD(50,10,10)
    INTEGER A2(50,10,10)
    INTEGER TEST(50,10,10),FIELDADD(50,50),FIELDMLT(50,50)
    INTEGER COUNT,SIZE
    LOGICAL FOUNDIT
    COMMON /BLK2/P

    DATA (IRRPOLY(1,J),J=1,3)/1,0,1/
    DATA (IRRPOLY(2,J),J=1,3)/2,0,1/
    DATA (IRRPOLY(3,J),J=1,4)/1,2,0,1/
    DATA (IRRPOLY(4,J),J=1,6)/1,0,1,0,0,1/

    IF (BASE.EQ.9) THEN
       POLYN=1
       P=3
       N=2
    ELSE
       IF (BASE.EQ.25) THEN
          POLYN=2
          P=5
          N=2
       ELSE
          IF (BASE.EQ.27) THEN
             POLYN=3
             P=3
             N=3
          ELSE
             IF (BASE.EQ.32) THEN
                POLYN=4
                P=2
                N=5
             ELSE
                POLYN=1
                P=BASE
                N=1
             ENDIF
          ENDIF
       ENDIF
    ENDIF

 C   ---------------------------------------------------------------
 C   CALCULATE COMPANION {A(1, , )} aND IDENTITY {A(0, , )} mATRICES
 C   ---------------------------------------------------------------
    A2(1,1,N) = P-IRRPOLY(POLYN,1)
    DO 10 I=1,N-1
       A2(1,I+1,I)=1
       A2(1,I+1,N) = MOD(P-IRRPOLY(POLYN,I+1),P)
       A2(0,I,I)=1
 10   CONTINUE
    A2(0,N,N)=1
    DO 11 I=1,N-2
       CALL MATMULT(A2,A2,A2,N,I+1,1,I)
 11   CONTINUE

 C   --------------------------------------------
 C   GENERATE ALL POLYNOMIALS WITH DEGREE<N IN FP
 C   --------------------------------------------

    CALL POLYGEN(POLY,N,P)

 C   ---------------------------------------------------
 C   COMPUTE ELEMENTS OF THE FIELD FQ, WHERE Q=P**N
 C   A'= (A0 I) + (A1 A) + (A2 A**2) + ... + (AN-1 A**N-1)
 C   ---------------------------------------------------

    DO 20 I=1,BASE
       DO 50 J=1,N
          DO 30 L1=1,N
             DO 40 L2=1,N
                A(J-1,L1,L2)=MOD((A2(J-1,L1,L2)*POLY(I,J)),P)
 40          CONTINUE
 30       CONTINUE
          CALL MATADD(FIELD,FIELD,A,N,I,I,(J-1))
 50    CONTINUE
 20 CONTINUE
    SIZE=N**2
    DO 12 I=1,BASE
       DO 13 J=1,BASE
```

```
 C   ----------------------
 C   ADDITION TABLE ...
 C   ----------------------

          CALL MATADD(TEST,FIELD,FIELD,N,1,I,J)
          L1=1
          FOUNDIT=.FALSE.
          DO WHILE (.NOT.FOUNDIT)
             COUNT=0
             DO 14 L2=1,N
               DO 15 L3=1,N
                  IF (TEST(1,L2,L3).EQ.FIELD(L1,L
     1             2,L3)) COUNT=COUNT+1
 15            CONTINUE
 14          CONTINUE
             IF (COUNT.EQ.SIZE) THEN
                FOUNDIT=.TRUE.
                FIELDADD(I,J)=L1-1
             ENDIF
             L1=L1+1
          ENDDO

 C   --------------------------
 C   MULTIPLICATION TABLE ...
 C   --------------------------

          CALL MATMULT(TEST,FIELD,FIELD,N,1,I,J)
          L1=1
          FOUNDIT=.FALSE.
          DO WHILE (.NOT.FOUNDIT)
             COUNT=0
             DO 16 L2=1,N
                DO 17 L3=1,N
                   IF (TEST(1,L2,L3).EQ.FIELD(L1,L
     1             2,L3)) COUNT=COUNT+1
 17             CONTINUE
 16          CONTINUE
             IF (COUNT.EQ.SIZE) THEN
                FOUNDIT=.TRUE.
                FIELDMLT(I,J)=L1-1
             ENDIF
             L1=L1+1
          ENDDO
 13     CONTINUE
 12   CONTINUE
    END

 SUBROUTINE POLYGEN(POLY,N,P)

 C   GENERATES ALL POLYNOMIALS IN A FIELD OF ORDER NPOLY OF
 C   OF DEGREE<N

    INTEGER I,J,K,N,P
    INTEGER POLY(0:50,0:50)

    NPOLY=P**N
    DO 10 I=1,NPOLY
       DO 20 J=1,N
          INTG=INT((I-1)/(P**(J-1)))
          POLY(I,J)=MOD(INTG,P)
 20    CONTINUE
 10 CONTINUE
 100   FORMAT(10I4)
    RETURN
    END

 SUBROUTINE MATMULT(A,A1,A2,N,I,J,K)

 C   MATRIX MULTIPLICATION: A=A1*A2, IN MODULO P
 C       WHERE A,A1,A2 ARE N*N MATRICES AND
 C            I,J,K INDEX THE CORRECT MATRIX.

    COMMON /BLK2/P
    INTEGER A(50,10,10),A2(50,10,10)
    INTEGER A1(50,10,10)
    INTEGER P

       DO 40 L1=1,N
       DO 50 L2=1,N
       A(I,L1,L2)=0
       DO 60 L3=1,N
          A(I,L1,L2)=A(I,L1,L2)+A1(J,L1,L3)*A2(K,L3,L2)
 60      CONTINUE
       A(I,L1,L2)=MOD(A(I,L1,L2),P)
 50      CONTINUE
 40      CONTINUE
 100   FORMAT(10I4)
    RETURN
    END

 SUBROUTINE MATADD(A,B,C,N,I,J,K)

 C   MATRIX ADDITION: A=B+C, WHERE A,B,C ARE N*N MARICES
 C   AND I,J,K INDEX THE CORRECT MATRIX.

    COMMON /BLK2/P
    INTEGER A(50,10,10),B(50,10,10),C(50,10,10)
    INTEGER P
```

```
      DO 10 L1=1,N
         DO 20 L2=1,N
            ITEMP=B(J,L1,L2)+C(K,L1,L2)
            A(I,L1,L2)=MOD(ITEMP,P)
20          CONTINUE
10       CONTINUE
      RETURN
      END
```

REFERENCES

Bratley, P. and Fox, B.L. (1986). ***Implementing Sobol's quasi-random sequence generator.*** Tech. Rep. Université de Montréal, Montreal, Quebec, Canada.

Cheng, R.C.H. (1984). ***Antithetic variate methods for simulations of processes with peaks and troughs.*** Eur. J. Opl. Res. 15, 227-236.

Cheng, R.C.H. and Davenport, T. ***The problem of dimensionality in stratified sampling.***. (to be published in Management Science).

Faure, H. (1982). ***Discrépance de suites associées à un système de numération (en dimension s).*** Acta Arithmetica XLI, 337-351.

Fox, B.L. (1986). ***Algorithm 647: Implementation and relative efficiency of quasi-random sequence generators.*** ACM Transactions on Mathematical Software, Vol. 12, No. 4, 362-376.

Halton, J.H. (1960). ***On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals.*** Numer. Math. 2, 84-90.

Niederreiter, H. (1978). ***Quasi Monte Carlo methods and pseudo random numbers.*** Bull. Amer. Math. Soc. 84, 957-1041.

__________ (1987). ***Point sets and sequences with small discrepancy.*** Mh. Math 104, 273-337.

__________ (1988). ***Low-discrepancy and low-dispersion sequences.*** J. of Number Theory, Vol. 30, No. 1, 51-70.

Roth, K.F. (1954). ***On Irregularities of distribution.*** Mathematika 1, 73-79.

Sobol, I.M. (1967). ***The distribution of points in a cube and the approximate evaluation of integrals.*** USSR Comput. Maths. Maths. Phys. 7, 86-112.

Van der Corput, J.G. (1935). ***Verteilungsfunktionen.*** I, II, Nederl. Akad. Wetensch. Proc. 38, 813-821, 1058-1066.

AUTHORS' BIOGRAPHIES

RUSSELL C.H. CHENG obtained a B.A. from Cambridge University, England, in 1968, and the diploma in Mathematical Statistics in 1969. He obtained his Ph.D. in 1972 from Bath University working on computer simulation models of industrial chemical plants in association with ICI. He joined the Mathematics Department of the University of Wales Institute of Science and Technology in 1972 and was appointed Reader in 1988. His main fields of interest include: computer generation of random variates, variance reduction methods, parametric estimation methods, applications of Markov decision processes to industrial processes and more recently: ship simulation; and he has published a number of articles in all these fields.

School of Mathematics,
University of Wales College of Cardiff,
Senghennydd Road,
Cardiff CF2 4AG,
Great Britain.

TERESA R. DAVENPORT received her B.Sc.(Hons) degree in Statistics with Management Science Techniques in 1986 from the University of Wales. She became interested in discrete event simulation and was awarded a University of Wales Research Studentship to study variance reduction techniques. She is currently enrolled in the doctoral program of the School of Mathematics, University of Wales, Cardiff.