Owen (1995) shows that the sequence $(\boldsymbol{X}_i)$ obtained inherits the equidistribution property of $(\boldsymbol{A}_i)$ and the individual points in it are uniformly distributed on $[0,1)^s$. Figure 2.4 presents examples of Scrambled Nets on $[0,1)^2$ with $n = 15$ and 25 runs, constructed using the software program provided by Art Owen at www-stat.stanford.edu/owen.

Figure 2.4: Example of 15 points of a Scrambled (3,0,1,2)-Net in base 5 (left panel), and 25 points of a Scrambled (0,2,2)-Net in base 5 (right panel).

2.2.3 Sobol′ Sequences

Sobol′ (1967) introduced the construction of quasi-random sequences of points that have low star discrepancy (see page 15). To introduce the construction of the Sobol′ Sequence consider working in one-dimension. To generate a sequence of values $x^1, x^2 ...$ with $0 < x^i < 1$, first we need to construct a set of *direction numbers* $v_1, v_2, ...$ Each v_i is a binary fraction that can be written $v_i = \frac{m_i}{2^i}$, where m_i is an odd integer such that $0 < m_i < 2^i$. To obtain m_i the construction starts by choosing a primitive polynomial in the field $\mathbb{Z}_2$, i.e. one may choose $P = x^d + a_1 x^{d-1} + ... + a_{d-1} x + 1$

where each a_i is 0 or 1 and P is an arbitrary chosen primitive polynomial of degree d in $\mathbb{Z}_2$. Then, the m_i's can be calculated recurrently as

$$m_i = 2a_1 m_{i-1} \oplus 2^2 a_2 m_{i-2} \oplus ... \oplus 2^{d-1} a_{d-1} m_{i-d+1} \oplus 2^d m_{i-d} \oplus m_{i-d}$$

where each term is expressed in base 2 and $\oplus$ denotes a bit-by-bit exclusive-or operation, i.e

$$0 \oplus 0 = 0,\ 0 \oplus 1 = 1 \oplus 0 = 1,\ 1 \oplus 1 = 0.$$

When using a primitive polynomial of degree d, the initial values $m_1, ..., m_d$ can be arbitrarily chosen provided that each m_i is odd and $m_i < 2^i$, $i = 1, ..., d$.

Example: If we choose the primitive polynomial $x^3 + x + 1$ and the initial values $m_1 = 1$, $m_2 = 3$, $m_3 = 7$, m_i's are calculated as follows:

$$m_i = 4m_{i-2} \oplus 8m_{i-3} \oplus m_{i-3}.$$

Then

$m_4 = 12 \oplus 8 \oplus 1 = 1100 \oplus 1000 \oplus 0001 = 0101 = 0 \times 2^3 + 1 \times 2^2 + 0 \times 2 + 1 \times 2^0 = 5$

$m_5 = 28 \oplus 24 \oplus 3 = 11100 \oplus 11000 \oplus 00011 = 00111 = 7$

$m_6 = 20 \oplus 56 \oplus 7 = 010100 \oplus 111000 \oplus 000111 = 43$

and

$v_1 = \frac{m_1}{2^1} = \frac{1}{2^1} = 0.1$ in binary

$v_2 = \frac{m_2}{2^2} = \frac{3}{2^2} = 0.11$ in binary

$v_3 = \frac{m_3}{2^3} = \frac{7}{2^3} = 0.111$ in binary

$v_4 = \frac{m_4}{2^4} = \frac{5}{2^4} = 0.0101$ in binary, and so on.

In order to generate the sequence $x^1, x^2, ...$, Sobol$'$ (1967) proposed using

$$x^n = b_1 v_1 \oplus b_2 v_2 \oplus \cdots$$

and

$$x^{n+1} = x^n \oplus v_c$$

where $\cdots b_3 b_2 b_1$ is the binary representation of n and b_c is the rightmost zero-bit in the binary representation of n.

Returning to the previous example, the first few values of x are thus generated as follows. To start the recurrence, take $x^0 = 0$.

$$
\begin{aligned}
Initialization: \; x^0 &= 0 \\
n &= 0 \text{ in binary so} \\
c &= 1 \\
Step\,1: \; x^1 &= x^0 \oplus v_1 \\
&= 0.0 \oplus 0.1 \text{ in binary} \\
&= 0.1 \text{ in binary} \\
&= \frac{1}{2} \\
n &= 1 \text{ in binary so} \\
c &= 2 \\
Step\,2: \; x^2 &= x^1 \oplus v_2 \\
&= 0.10 \oplus 0.11 \text{ in binary} \\
&= 0.01 \text{ in binary} \\
&= \frac{1}{4} \\
n &= 10 \text{ in binary so} \\
c &= 1
\end{aligned}
$$

$$
\begin{aligned}
Step\,3:\ x^3 &= x^2 \oplus v_1 \\
&= 0.01 \oplus 0.10 \ \text{ in binary} \\
&= 0.11 \ \text{ in binary} \\
&= \frac{3}{4} \\
n &= 11 \ \text{ in binary so} \\
c &= 3
\end{aligned}
$$

and so on.

To generalize this procedure to s dimensions, Sobol′ shows that in order to obtain $O(\log^s N)$ discrepancy, where N represents the number of points, it suffices to choose s distinct primitive polynomials, calculate s sets of *direction numbers* and then generate each component x_i^n of the quasi-random vector separately. Figure 2.5 presents graphs of 15 and 25-point Sobol′ sequences in $[0,1)^2$.

Figure 2.5: Example of 15-point Sobol′ Sequence (left panel), and 25-point Sobol′ Sequence (right panel).

Several other methods for producing low-discrepancy sequences have been proposed by Halton, Faure, Niederreiter. In our comparisons beginning in Chapter 3, we have used the Sobol′ and Niederreiter Sequences whose description follows.

2.2.4 Niederreiter Sequences

Niederreiter (1988) proposed a new method of generating quasi-Monte Carlo sequences. Let $\triangle(N)$ denote $N \times D_N^*$, where D_N^* is the star discrepancy. It is believed that the best possible bound for the discrepancy of the first N terms of a sequence of points in $[0,1)^s$ is of the form

$$\triangle(N) \leq C_s(\log N)^s + O((\log N)^{s-1})$$

for all $N \geq 2$. The methods proposed by Niederreiter yield sequences with the lowest C_s currently known. Niederreiter provides a method of constructing (t,s)-sequences in any base b with $b \geq 2$ but the construction of sequences for prime bases is much simpler to implement. In our comparison, we used Niederreiter sequences with base $b = 2$ whose implementation is much faster than for other bases given the binary nature of the computers and the fact that the construction of such sequences involves operations in the field F_2 whose elements are bits 0 or 1.

As in the construction of the Sobol′ sequences, we focus on the one-dimensional case. To generalize to s dimensions it suffices to choose s distinct primitive polynomials and generate each dimension, x_i^n, of the quasi-random vector separately. For now, our aim is to generate a sequence $x_1, x_2, ..., 0 \leq x_n < 1$, with low-discrepancy over the unit interval.

To generate x_n, we let $n-1 = a_{R-1}a_{R-2}...a_1a_0$ be the *base-b representation* of $n-1$ (where R represents the maximum number of base-b digits allowed by convention).

Then x_n will be given as a base b fraction of the form

$$x_n = 0.d_1 d_2 ... d_R.$$

In practice, x_n is obtained by calculating an integer Q_n whose base-b representation is $Q_n = d_1 d_2 ... d_R$ followed by taking $x_n = \frac{Q_n}{b^R}$. The Q_n's are recurrently constructed as

$$Q_n = a_0 C_0 \oplus a_1 C_1 \oplus ... \oplus a_{R-1} C_{R-1}$$

$$Q_{n+1} = Q_n \oplus C_r$$

where $\oplus$ represents a bit-by-bit exclusive-or operation. To start the recurrence, Q_1 is taken to be 0 and $C_r = c_{1r} c_{2r} ... c_{Rr}$ $(r \leq R)$ where the c_{jr}'s $(1 \leq j \leq R)$ are constructed using the following algorithm:

(1) Choose a primitive polynomial $p(x)$ with coefficients in F_2 of degree $e \geq 1$. Set $j \leftarrow 0$, $q \leftarrow -1$ and $u \leftarrow e$.

(2) Increment j. If $u = e$, go to step (3); otherwise, go to step (4).

(3) Increment q and set $u \leftarrow 0$. Calculate $b(x) = p(x)^{q+1} = x^m - b_{m-1} x^{m-1} - \cdots - b_0$, a polynomial of degree $m = e(q+1)$, and then calculate the elements $v_i = \oplus_{k=1}^{m} b_{m-k} v_{i-k}$ for $m \leq i \leq R + e - 2$ and $v_i = 0, v_{m-1} = 1$ for $0 \leq i \leq R + e - 2$.

(4) For $0 \leq r \leq R-1$ set $c_{jr} \leftarrow v_{r+u}$. Increment u. If $j < R$ go to step (2); otherwise stop.

To generalize this procedure to the s dimensional case, it suffices to take different polynomials for each coordinate and calculate different c_{jr}^i's, hence different x_n^i's for each coordinate. Figure 2.6 presents the matrix of 2-dimensional projections of a 31-point Niederreiter sequence with points in $[0, 1)^6$.

Figure 2.6: Example of 2-dimensional projections of a 6-dimensional 31-point Niederreiter Sequence.

2.2.5 The Good Lattice Point Sets

The Good Lattice Point design is another example of a low-discrepancy point set introduced in the literature of numerical integration by Korobov (1959). It was motivated by the desire to find good sets of evaluation points for numerical computation of multi-dimensional integrals.

The experimental domain is taken to be $C^s = [0, 1)^s$. The construction of Good Lattice Point sets involves using a *generating vector* $(n; h_1, h_2, ..., h_s)$ with integral components satisfying $1 \leq h_i < n$, where $h_i \neq h_j$ for $(i \neq j), s < n$ and the greatest