Notes on generating Sobol' sequences
Stephen Joe and Frances Y. Kuo
August 2008

1 Original implementation

The algorithm for generating Sobol’ sequences is clearly explained in [2]. Here we give a brief outline of the details. To generate the \( j \)th component of the points in a Sobol’ sequence, we need to choose a primitive polynomial of some degree \( s_j \) in the field \( \mathbb{Z}_2 \),
\[
x^{s_j} + a_{1,j} x^{s_j-1} + a_{2,j} x^{s_j-2} + \cdots + a_{s_j-1,j} x + 1,
\]
where the coefficients \( a_{1,j}, a_{2,j}, \ldots, a_{s_j-1,j} \) are either 0 or 1. We define a sequence of positive integers \( \{m_{1,j}, m_{2,j}, \ldots\} \) by the recurrence relation
\[
m_{k,j} := 2a_{1,j} m_{k-1,j} \oplus 2^2 a_{2,j} m_{k-2,j} \oplus \cdots \oplus 2^{s_j-1} a_{s_j-1,j} m_{k-s_j+1,j} \oplus 2^{s_j} m_{k-s_j,j} \oplus m_{k-s_j,j},
\]
where \( \oplus \) is the bit-by-bit exclusive-or operator. The initial values \( m_{1,j}, m_{2,j}, \ldots, m_{s_j,j} \) can be chosen freely provided that each \( m_{k,j}, 1 \leq k \leq s_j \), is odd and less than \( 2^k \). The so-called direction numbers \( \{v_{1,j}, v_{2,j}, \ldots\} \) are defined by
\[
v_{k,j} := \frac{m_{k,j}}{2^k}.
\]
(With a slight abuse of terminology, we also refer to the numbers \( m_{k,j} \) as direction numbers.) Then \( x_{i,j} \), the \( j \)th component of the \( i \)th point in a Sobol’ sequence, is given by
\[
x_{i,j} := i_1 v_{1,j} \oplus i_2 v_{2,j} \oplus \cdots,
\]
where \( i_k \) is the \( k \)th digit from the right when \( i \) is written in binary \( i = (\ldots i_3 i_2 i_1)_2 \). Here and elsewhere in this article, we use the notation \( (\cdot)_2 \) to denote the binary representation of numbers.

For example, with \( s_j = 3 \), \( a_{1,j} = 0 \), and \( a_{2,j} = 1 \), we have the primitive polynomial \( x^3 + x + 1 \). Starting with \( m_{1,j} = 1 \), \( m_{2,j} = 3 \), and \( m_{3,j} = 7 \), we use the recurrence (2) to obtain \( m_{4,j} = 5 \), \( m_{5,j} = 7 \), etc. This leads to the direction numbers
\[
v_{1,j} = (0.1)_2, \ v_{2,j} = (0.11)_2, \ v_{3,j} = (0.111)_2, \ v_{4,j} = (0.0101)_2, \ v_{5,j} = (0.00111)_2, \ldots.
\]
Following (3), the \( j \)th components of the first few points are given by
\[
\begin{align*}
0 &= (0)_2, & x_{0,j} &= 0, \\
1 &= (1)_2, & x_{1,j} &= (0.1)_2 = 0.5, \\
2 &= (10)_2, & x_{2,j} &= (0.11)_2 = 0.75, \\
3 &= (11)_2, & x_{3,j} &= (0.1)_2 \oplus (0.11)_2 = (0.01)_2 = 0.25, \\
4 &= (100)_2, & x_{4,j} &= (0.111)_2 = 0.875, \\
5 &= (101)_2, & x_{5,j} &= (0.1)_2 \oplus (0.111)_2 = (0.011)_2 = 0.375,
\end{align*}
\]
2 Gray code implementation

The formula (3) corresponds to the original implementation of Sobol’. A more efficient Gray code implementation proposed by Antonov and Saleev [1] can be used in practice, see also [2].

The (binary-reflected) Gray code of an integer $i$ is defined as

$$\text{gray}(i) := i \oplus \left\lfloor \frac{i}{2} \right\rfloor = (\ldots i_3i_2i_1)_2 \oplus (\ldots i_4i_3i_2)_2.$$ 

It has the property that the binary representations of $\text{gray}(i)$ and $\text{gray}(i - 1)$ differ in only one position, namely, the index of the first 0 digit from the right in the binary representation of $i - 1$.

$$
\begin{array}{ccc}
i & \text{gray}(i) \\
0 &=& (0000)_2 \\
1 &=& (0001)_2 \\
2 &=& (0010)_2 \\
3 &=& (0011)_2 \\
4 &=& (0100)_2 \\
5 &=& (0101)_2 \\
6 &=& (0110)_2 \\
7 &=& (0111)_2 \\
8 &=& (1000)_2 \\
9 &=& (1001)_2 \\
10 &=& (1010)_2 \\
11 &=& (1011)_2 \\
12 &=& (1100)_2 \\
13 &=& (1101)_2 \\
14 &=& (1110)_2 \\
15 &=& (1111)_2 \\
\end{array}
$$

Observe from the table that Gray code is simply a reordering of the nonnegative integers within every block of $2^m$ numbers for $m = 0, 1, \ldots$.

Instead of (3), we generate the Sobol’ points using

$$\bar{x}_{i,j} := g_{i,1} v_{1,j} \oplus g_{i,2} v_{2,j} \oplus \cdots,$$  \hspace{1cm} (4)

where $g_{i,k}$ is the $k$th digit from the right of the Gray code of $i$ in binary, i.e., $\text{gray}(i) = (\ldots g_{i,3}g_{i,2}g_{i,1})_2$. Equivalently, since $\text{gray}(i)$ and $\text{gray}(i - 1)$ differ in one known position, we can generate the points recursively using

$$\bar{x}_{0,j} := 0 \quad \text{and} \quad \bar{x}_{i,j} := \bar{x}_{i-1,j} \oplus v_{c_i-1,j},$$ \hspace{1cm} (5)

where $c_i$ is the index of the first 0 digit from the right in the binary representation of $i = (\ldots i_3i_2i_1)_2$. We have $c_0 = 1$, $c_1 = 2$, $c_2 = 1$, $c_3 = 3$, $c_4 = 1$, $c_5 = 2$, etc.

With the Gray code implementation, we simply obtain the points in a different order, while still preserving their uniformity properties. This is because every block of $2^m$ points for $m = 0, 1, \ldots$ is the same as the original implementation. We stress that (4) and (5) generate exactly the same sequence; (4) allows one to start from any position in the sequence, while (5) is recursive and is more computationally efficient.
3 Primitive polynomials and direction numbers

Following the convention established in [2], we identify the coefficients of a primitive polynomial (1) with an integer

\[ a_j := (a_{1,j} a_{2,j} \ldots a_{s_j-1,j})_2, \]

so that each primitive polynomial is uniquely specified by its degree \( s_j \) together with the number \( a_j \). For example, from \( s_j = 7 \) and \( a_j = 28 = (011100)_2 \) we obtain the polynomial

\[ x^7 + x^5 + x^4 + x^3 + 1. \]

The primitive polynomials and direction numbers obtained based on various search criteria (see [3, 4]) can be downloaded as text files from our web page

http://www.maths.unsw.edu.au/~fkuo/sobol/

The files will be updated frequently as the parameters for higher dimensions become available. Our target dimension is 21201.

4 Skipping initial points?

It has been recommended by some that the Sobol’ sequence tends to perform better if an initial portion of the sequence is dropped: the number of points skipped is the largest power of 2 smaller than the number of points to be used. However, we are less persuaded by such recommendation ourselves.

References


