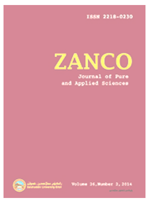
ZANCO Journal of Pure and Applied Sciences

The official scientific journal of Salahaddin University-Erbil

ZJPAS (2019), 31 (1); 32-39

http://dx.doi.org/10.21271/ZJPAS.31.1.5



Comparing Halton and Sobol Sequences in Integral Evaluation

Nadia A. Mohammed

Department of Mathematics, College of Basic Education, University of Duhok, Kurdistan Region, Iraq

ARTICLE INFO

Article History:
Received: 27/09/2016
Accepted: 22/01/2019
Published:27/02/2019

Keywords:
Halton sequence
Monte Carlo
Quasi-Monte Carlo Sobol sequence

***Corresponding Author:**
Nadia A. Mohammed
nadia.mohammed@uod.ac

ABSTRACT

Halton and Sobol sequences are two of the most popular number sets used in quasi-Monte Carlo methods. These sequences are effectively used instead of pseudo random numbers in the evaluation of integrals. In this paper, the two sequences are compared in terms of the size of the number sets and dimensionality. The comparison is implemented with matlab programming for evaluating numerical integrals. The absolute error, which is the absolute difference between the exact and estimated errors, is plotted against dimensions for different functions. The practical results show that, except the first dimension, Sobol sequence is better than Halton sequence. The results also show that Sobol sequence outputs are more stable.

1. INTRODUCTION

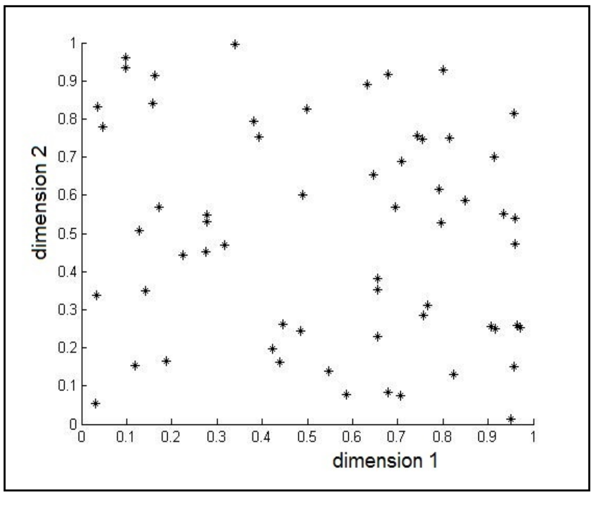
Monte Carlo (MC) method is a numerical method for solving mathematical problems by the simulation of random numbers. The most common applications of MC method are the evaluation of integrals, mathematical finance, tree search, and simulation in several branches of science (Landau & Binder 2005) (Levy 2010).

In numerical integral application, integrals are sometimes not solved analytically. Whereas some numerical approximations can be obtained for such integrals using MC methods. The MC methods are based on computer generation sequences of pseudo-random numbers. These sequences behave as if they are truly random although they are produced by deterministic algorithms. The convergence rate of approximating an integral with MC methods using a set of random samples of size *n* is $O(n^{-1/2})$ (Serre 2010).

Quasi-Monte Carlo (QMC) methods, also called low-discrepancy methods, use algorithms that reduce the discrepancy of generated random numbers. The low-discrepancy sequences cover the space better than pseudo-random sequences by reducing gaps between the points. In computer programming, the generated sequences are matrices of size *n*-by-*s*, where *n* is the number of points and *s* is the dimension of the hypercube being sampled. In Figure 1, dimension 1 is plotted against dimension 2 for both MC and QMC to show the discrepancy of generated random sequences. Quasi-random numbers appear to cover the area more uniformly than a pseudo-random numbers. The advantage of using low-discrepancy sequences is a faster rate of convergence, it achieves a convergence rate of $O(n^{-1}(\log n)^s)$ (Serre 2010). There are several high-dimensional sequences for use in QMC: Halton, Faure, and Sobol sequences. Other important sequences are Niederreiter and generalized Faure sequences (Krykova 2003).

There are several publications that compare MC and QMC methods (Owen 2008). QMC method have been successfully used for multivariate integration of high dimensions, and were significantly more efficient than MC method (Sloan & Woźniakowski 1998) (Kuo & Sloan 2013). One disadvantage of QMC integration is that its error is difficult to be estimated, unlike MC integration (Tuffin 2008) (Jank 2005). Also it is stated that the problem of QMC methods is that their convergence is not independent of dimensionality (Frey 2008). Therefore, the accuracy of high-dimensional computations will not be guaranteed to be better than the MC. Lemieux mentioned within the drawbacks of QMC that the dimension needs to be small and the number of elements needs to be large in order to enhance the efficiency of QMC over the regular MC (Lemieux 2009). Morokoff and Caflisch remark that the advantage of the QMC method is greater if the integrand is smooth, and the number of dimensions *s* of the integral is small (Morokoff & Caflisch 1995).

The aim of this paper is to implement a practical comparison between two of the most popular sequences used in QMC methods, Halton and Sobol sequences. The comparison concentrates on the effect of the size of random number sets and the number of dimensions on the accuracy of the integral evaluation. The approach is to use the two matlab functions, *haltonset* and *sobolset*, with some of more general parameters for generating the two sequences and apply them on the simulation of mathematical integrals. To show the contribution of the dimensions in the error of the numerical integration, 60 dimensions are used separately in evaluating one-dimensional integral functions. The integral is applied on



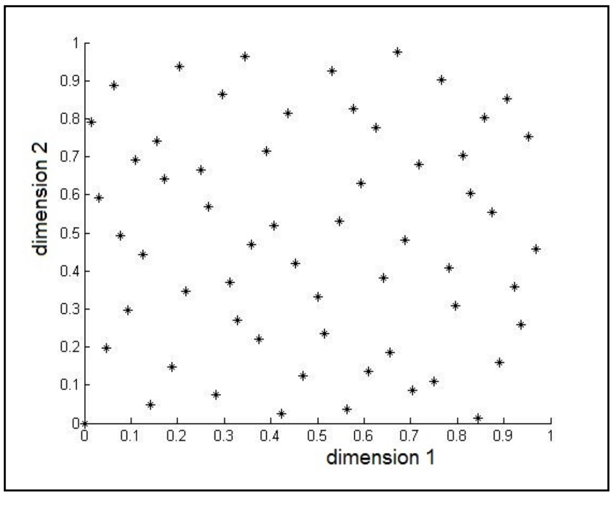


Figure 1: Two dimensional plot of pseudo-random numbers (left) and quasi-random numbers (right)

two different functions and the absolute errors are depicted to be investigated visually.

The rest of the paper is organized as follows. Section 2, and section 3 give the mathematical background for Halton and Sobol sequences. Section 4 explains the method of integral evaluation using sample technique and shows the error calculations. Section 5 presents the algorithms and programs used to achieve the goal of this paper. Section 6 presents the results and discusses the outputs of the programs. Section 6 concludes the paper.

2. HALTON SEQUENCES

The theoretical frameworks of Halton and Sobol sequences are presented in several references. The interested reader can cosult (Caflisch 1998), (Dalal et al. 2008), and (McLeish 2011). The following short description is based on (Veach 1997). Halton low-discrepancy sequences are inspired by Van der Corput sequence introduced originally for one dimension and using the binary number system. The main contribution of Van der Corput in low-discrepancy sequences is that the coefficients of the digit expansion of an increasing integer *k* in base *b* can be used to define a low-discrepancy sequence. The Halton sequence is a generalization of the one-dimensional Van der Corput sequence to higher dimensions. Each dimension is represented in a different prime base *b* (e.g., 2, 3, 5, 7, . . .). In one dimension, the radical inverse sequence $x_i = \emptyset_b(i)$ is obtained by first writing the base-*b* expansion of *k*:

$$k = \sum_{i=0}^{\infty} a_i b^i,$$

where $a_i \in \{0, \dots, b-1\}$. Reflecting the digits around the decimal point gives:

$$\emptyset_b(i) = \sum_{i=0}^{\infty} a_i b^{-i-1}$$

The special case when $b = 2$ is the sequence:

$$\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \cdots$$

To obtain a low-discrepancy sequence in several dimensions, we use a different radical inverse sequence in each dimension:

$$x_i = (\emptyset_{b_1}(i), \emptyset_{b_2}(i), \dots, \emptyset_{b_s}(i)),$$

where the bases b_i are all relatively prime. To get Halton sequence, b_i are chosen to be the first s primes:

$$x_i = (\emptyset_2(i), \emptyset_3(i), \emptyset_5(i), \dots, \emptyset_{p_s}(i))$$

Even though standard Halton sequences perform very well in low dimensions, correlation problems have been noted between sequences generated from higher primes.

3. SOBOL SEQUENCES

Sobol sequences (Sobol' 1967) also inspired by Van der Corput sequence with different approach. Here, we give a short description of generating Sobol sequences based on (Dalal et al. 2008) and (Bratley & Fox 1988). To generate one sequence (i.e., one dimension) of *N*-bit low-discrepancy Sobol numbers, we choose positive odd integers m_i, where $i=0, 1, \dots, N-1$ and define *N* direction vectors c_i:

$$c_i = \frac{m_i}{2^i} = 0.c_{i1}c_{i2}c_{i3}\cdots$$

where c_{ij} denote the binary expansion of c_i. Now, construct a primitive polynomial of degree *d* with coefficients a_i chosen from {0, 1}:

$$p(x) = x^d + a_1 x^{d-1} + \cdots + a_{d-1}x + 1$$

These coefficients a_i are used to calculate each direction vector c_i as:

$$c_i = a_1 c_{i-1} \oplus a_2 c_{i-2} \oplus a_1 c_{i-1}$$
$$\cdots \oplus a_d c_{i-d+1} \oplus c_{i-d} \oplus [c_{i-d} \gg d]$$

where $\oplus$ is bitwise exclusive-or (XOR), and the last term is c_{i-d} right-shifted by *d* bits.

A one-dimensional *N*-bit wide Sobol sequence $x_1, x_2, x_3, \ldots$ can be generated based on this set of direction vectors. The n^{th} term of this sequence is generated with $n = b_N b_{N-1} \ldots b_2 b_1$ in binary. Then,

$$x_n = b_1 c_1 \oplus b_2 c_2 \oplus \cdots \oplus b_{n-1} c_{N-1} \oplus b_N c_N$$

To generalize this procedure to *s* dimensions, it is sufficient to choose any *s* different primitive polynomials and calculate *s* different sets of direction vectors as explained above, and then generate each x_n using the corresponding set of direction vectors.

4. INTEGRAL EVALUATION

Monte Carlo approximation transforms the problem of integration into numerical method by calculating the average of the functions over random numbers. This is one of the MC methods for evaluating integrals which is called sample-mean method. To find the integral of a function over an *s*-dimensional unit cube, this method approximates the solution to the average of the function at a randomly set of points $x_1, \ldots, x_n$:

$$\int_{[0,1]^s} f(u)\,du \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i),$$

where each x_i is a vector of *s* elements. In matlab, the QMC functions, *haltonset* and *sobolset*, can be used to produce quasi random sequences in the form of *n*-by-*s* matrices, where *n* is the number of points in each dimension *s* of the hypercube being sampled. Many trials are implemented for testing these functions with different integrals and different sets of *n* and *s*. To explain the results, the integrals are evaluated with *n*=10000 and *n*=100000 with each value of *s*, where *s*=1, 2, …, 60. The functions which are used for the following explanations are:

$$f(x) = \sin(2x) \qquad (1)$$

$$f(x) = e^{-x} \qquad (2)$$

The general form of the integral is

$$I = \int_a^b f(x)\,dx$$

With sample-mean method, the *n* random numbers are taken from $x \in [0,1]$ and the integral is estimated as

$$I_{est} = (b-a)\frac{1}{n}\sum_1^n f(x_i)$$

If the interval of integration is taken between a=0 and b=1, the integral is simplified to

$$I_{est} = \frac{1}{n}\sum_1^n f(x_i)$$

To illustrate the results of evaluating the integrals, matlab programming is used to simulate the MC sample-mean method. The two functions, *haltonset* and *sobolset*, are used to generate 60 dimensions used to evaluate the one-dimensional integrals of functions (1) and (2) in the interval [0, 1]. With each dimension, the one-dimensional integral is evaluated and the absolute error is found for the sake of accuracy comparison. The absolute error is the absolute difference between the exact solution of the integral I_{exc}, and I_{est}. The integral evaluation is implemented for *n*=10000 and *n*=100000 to show the effect of changing the size of random number sets on the results of the integral evaluation.

The mean absolute error (MAE) is used to illustrate the effect of increasing the number of samples used for evaluating the integrals on decreasing the error of evaluation. The mean absolute error is the average of the absolute differences between the exact and estimated integrals. It is given by:

$$MAE = \frac{1}{n}\sum_1^n |I_{est} - I_{exc}|$$

5. ALGORITHMS AND PROGRAMS

Algorithm 1 shows a general method for estimating the absolute error for the function f(x) in the interval [a, b] for any of Halton or Sobol sequences with different values of dimensions. Program 1 uses the a matlab program to plot the relation between the absolute error and the dimensions for the function *sin(2x)* in the interval [0, 1]. Algorithm 2 lists the steps of calculating MAE for different values of samples for a selected dimension. Program 2 applies the algorithm to plot the relation between the number of samples and the MAE for the function *exp(-x)* in the interval [0, 1]

Alogorithm 1: Absolute error for interval [a,b]

```
1- SET n to the number of samples
2- SET s to number of dimensions
3- SET $I_{exc}$ to the exact solution of the integral
4- Use haltonset or sobolset to generate n×s matrix
5- FOR dim=1 to s
       FOR i=1 to n
           $y[i]=f(x_i)$
       END FOR
       $I_{est}$=(b-a)*AVERAGE(y)
       Error[dim]=| $I_{est}$ - $I_{exc}$|
6- END FOR
```

Program 1: Plotting absolute error vs dimensions for interval=[0,1]

```
% function y=sin(2x)
% a=0, b=1, therefore interval is ignored
exact=0.70807;
dim=60;          % for 60 dimensions
n=100000;        % number of samples
rng default
h=haltonset(dim);
halt=net(h, n);
s=sobolset(dim);
sobol=net(s, n);
for d=1:dim
xHalton=halt(:, d); % get x vector
yHalton=sin(2*xHalton); % find f(x)
    xSobol=sobol(:, d);% get x vector
    ySobol=sin(2*xSobol); % find f(x)
    yyHalton(d)=mean(yHalton);
    yySobol(d)=mean(ySobol);
end
errorHalton=abs(yyHalton-exact); % absolute
                      % error for Halton sequence
errorSobol=abs(yySobol-exact); % absolute
                      % error for Sobol sequence
plot(1:dim, errorHalton, ':.k','linewidth',1)
hold all
plot(1:dim, errorSobol, '-k','linewidth',1)
axis([0 dim 0 0.0004])
grid ON
xlabel('Dimension')
ylabel('Absolute Error')
```

Alogorithm 2: Mean absolute error for interval [a,b]

```
1- SET s to a selected dimension
2- SET $I_{exc}$ to the exact solution of the integral
3- SET k=1
4- FOR n=10000 to 100000, steps of 10000
       Use haltonset or sobolset to generate n×s matrix
       FOR d=1 to s
           FOR i=1 to n
               $y[i]=f(x_i)$
           END FOR
           yy[d]=AVERAGE(y)
       END FOR
       $I_{est}$=(b-a)*AVERAGE(yy)
       Error[k]=| $I_{est}$ - $I_{exc}$|
       k=k+1
5- END FOR
```

Program 2: Plotting MAE vs number of samples for interval [0,1]

```
% function y=exp(-x)
% a=0, b=1, therefore interval is ignored
exact=0.63212;  % exact solution of the function
dim=3;  % dimension=3
rng default
for n=10000:10000:100000  % samples
    h=haltonset(dim);
    halt=net(h, n);
    s=sobolset(dim);
    sobol=net(s, n);
    for d=1:dim
        xHalton=halt(:, d);   % get x vector
        xSobol=sobol(:, d);  % get x vector
        yHalton=exp(-xHalton);  % find f(x)
        ySobol=exp(-xSobol);  % find f(x)
        yyHalton(d)=mean(yHalton);
        yySobol(d)=mean(ySobol);
  end
```

```
maeHalton(n/10000)=mean(abs(yyHalton-
exact));  % MAE for Halton sequence
maeSobol(n/10000)=mean(abs(yySobol-exact));
          % MAE for Sobol sequence
  end
  plot(1:10,maeHalton, ':k', 'linewidth', 2)
  hold all
  plot(1:10, maeSobol, '-r', 'linewidth', 1)
  axis([1 10 0 0.001])
  grid ON
  xlabel('Nx1000')
  ylabel('MAE')
```

6. RESULTS AND DISCUSSION

The results of evaluating integrals of functions (1) and (2) are shown in Figure 2 and Figure 3 respectively. The figures show the plot of absolute error of estimated integrals using Halton and Sobol sequences as a function to the dimension of the sequences.

Visual inspection of the outputs shows that the absolute errors are same when s=1 for the two sequences. For s>1, the performance of Sobol sequence is better than that of Halton sequence for all the values of the dimensions. The outputs show that the average of the absolute error increases with the increasing of the dimension for Halton sequence, while it is stable with Sobol sequence. The outputs also show that when n is changed from 10000 to 100000, the absolute error is decreased in both sequences.

The number of samples used for the evaluation affects the accuracy of integral. To show that, Program 2, listed in the previous section, is executed with different values of samples n. The values of n are taken as 10000,

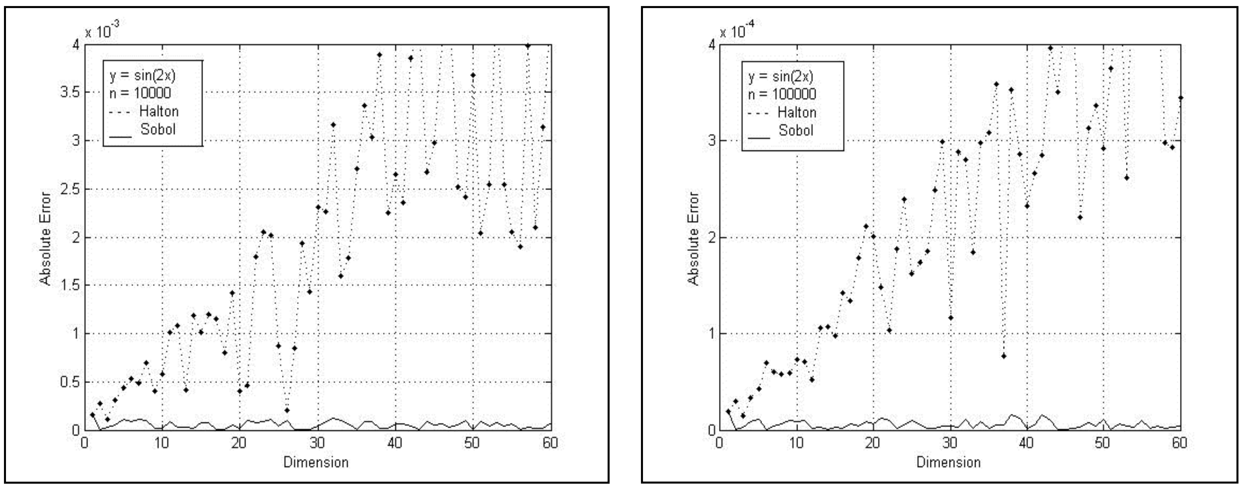


Figure 2: Absolute error of integral (1) with n=10000 (left) and n=100000 (right)

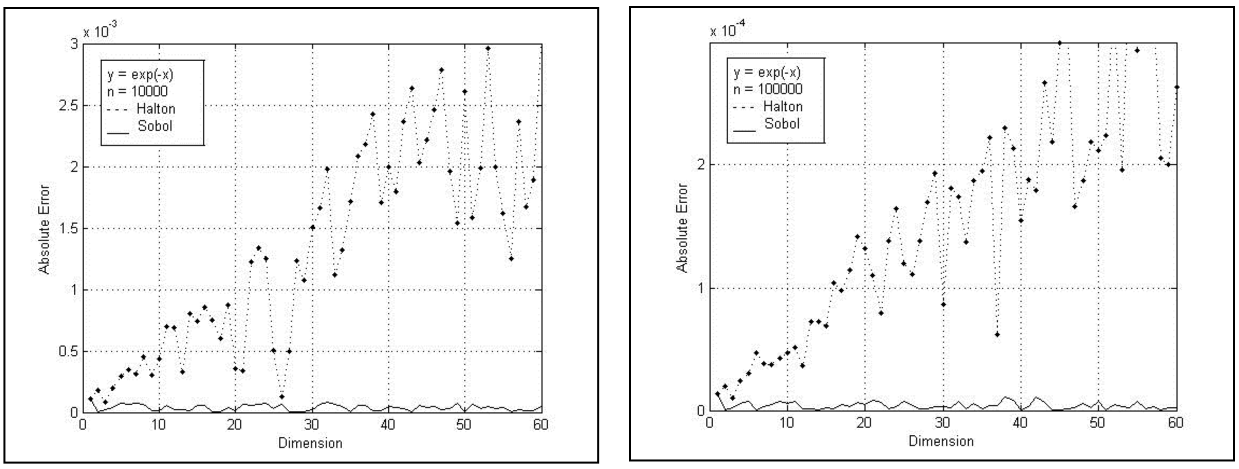


Figure 3: Absolute error of integral (2) with n=10000 (left) and n=100000 (right)

20000, …, 100000 with two values of the dimension, 3 and 60. These values are sufficient for revealing the effect of increasing n on the error with small and large values of the dimension. Figure 4 is the output of the program for integral (2) and it shows that the absolute error is decreased in both sequences with increasing n. The figure also shows that when d=3, the difference of MAE in the two sequences is much less than that when d is increased to 60. The MAE of Halton sequence produces very bad performance with large values of d and low values of n.

To find the reasons of the difference in accuracy of the results, it is important to see how the discrepancy is changed with changing dimension values in both sequences. To do so, the sequence numbers of several pairs of dimensions are plotted and investigated. All the testing of outputs show that Sobol sequence fill the space better than Halton sequence, leading to improve the results of integral estimation. Figure 5 shows examples of the relation between pair of dimensions of the two sequences. In high dimensions, Halton sequence tends to be uniform in diagonal arrangement leaving large spaces in the square unit. This bad distribution will increase the error in estimating the value of the integral. Sobol sequences maintains good distribution of the random number on all dimensions.

It is worth mentioning that there are other

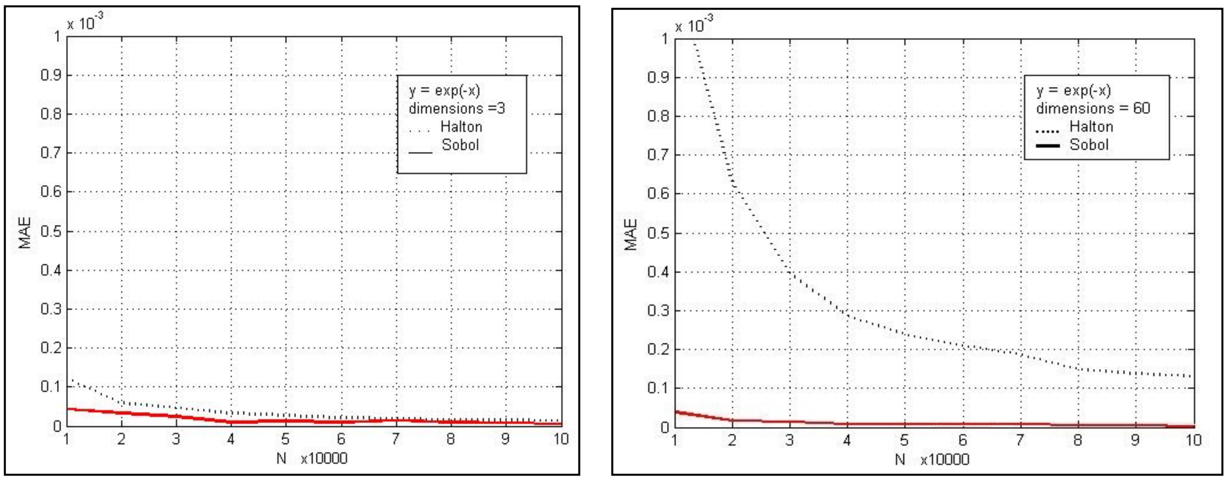


Figure 4: Mean absolute error for d=3 (left) and d=60 (right)

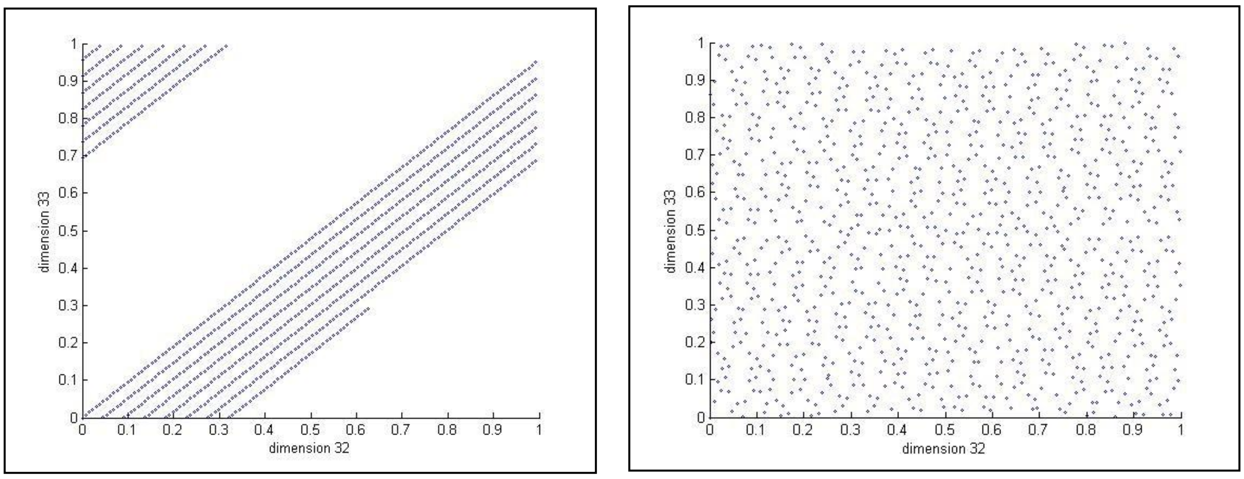


Figure 5: Discrepancy plot of Halton sequences (left) and Sobol sequences (right)

parameters, such as '*leap*' and '*skip*', which can be used with *haltonset* and *sobolset* in matlab to change the properties of the produced sets of the numbers (Kocis & Whiten 1997). These parameters are not used in this work.

7. CONCLUSIONS

Halton and Sobol sets are used in matlab for the generation of quasi-Monte Carlo sequences. These sequences are compared with different size of set numbers and different dimensions. The comparison is based on evaluating numerical integrals for one-dimensional functions by matlab programming. The results show that the performance of Sobol sequence is better and more stable than Halton sequence. The results also show that Sobol sequence maintains the feature of low-discrepancy while this feature is deteriorated with Halton sequence when the dimension value is increased. These differences in discrepancy affect the results of integral evaluations by increasing the absolute error with increasing d of Halton sequence. In the future work, the properties that change the sequences, such as *'skip'* and *'leap'*, which are available in matlab, can be used for further analysis.

REFERENCES

Bratley, P. & Fox, B.L., 1988. Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 14(1), 88–100.

Caflisch, R.E., 1998. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, 7, 1–49.

Dalal, I.L., Stefan, D. & Harwayne-Gidansky, J., 2008. Low discrepancy sequences for Monte Carlo simulations on reconfigurable platforms. In *2008 International Conference on Application-Specific Systems, Architectures and Processors*. 108–113.

Frey, R., 2008. *Monte Carlo methods: with application to the pricing of interest rate derivatives*. University of St. Gallen. (Doctoral dissertation)

Jank, W., 2005. Quasi-Monte Carlo sampling to improve the efficiency of Monte Carlo EM. *Computational statistics & data analysis*, 48(4), 685–701.

Kocis, L. & Whiten, W.J., 1997. Computational Investigations of Low- Discrepancy Sequences. *ACM Transactions on Mathematical Software*, 23(2), 266–294.

Krykova, I., 2004. *Evaluating of Path-Dependent Securities with Low Discrepancy Methods*. Worcester Polytechnic Institue. (Doctoral dissertation)

Kuo, F.Y. & Sloan, I.H., 2013. High-dimensional integration : The quasi-Monte Carlo way. *Acta Numerica*, *22*, 133-288.

Landau, D.P. & Binder, K., 2005. *A Guide to Monte Carlo Simulations in Statistical Physics* 2nd ed., cambridge university press.

Lemieux, C., 2009. *Monte Carlo and Qusi-Monte Carlo Sampling*, Springer.

Levy, G., 2002. An introduction to quasi-random numbers. *Numerical Algorithms Group Ltd.*, 143.

McLeish, D.L., 2011. *Monte Carlo simulation and finance* (Vol. 276). John Wiley & Sons.

Morokoff, W.J. & Caflisch, R.E., 1995. Quasi-monte carlo integration. *Journal of computational physics*, 122(2), 218–230.

Owen, A.B., 2009. Monte Carlo and Quasi-Monte Carlo for Statistics. In *Monte Carlo and Quasi-Monte Carlo Methods*. Springer Berlin Heidelberg., 3–18.

Serre, L., 2010. *A Matlab Program for Testing Quasi-Monte Carlo Constructions*. University of Waterloo. (Master dissertation)

Sloan, I.H. & Woźniakowski, H., 1998. When Are Quasi-Monte Carlo Algorithms Efficient for High Dimensional Integrals? *Journal of Complexity*, 14(1), 1-33.

Sobol', I.M., 1967. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4), 784–802.

Tuffin, B., 2008. Randomization of Quasi-Monte Carlo methods for error estimation: survey and normal approximation. *Monte Carlo Methods and Applications*, 10(3–4), 617–628.

Veach, E., 1997. *Robust monte carlo methods for light transport simulation*. Stanford University. (Doctoral dissertation)