Ambient Science, 2016: Vol. 03(2); 01-03
DOI:10.21276/ambi.2016.03.2.ga01

Published by: National Cave Research and Protection Organization, India

ambient SCIENCE
Vol. 3(2): 01-03
Year 2016

GENERAL ARTICLE

Systematic Numbers for Monte Carlo Integration

Mojtaba Moradi

Faculty of Technology and Engineering, East of Guilan, University of Guilan, Iran

Key words: Quasi-Random Numbers, Halton Sequence.

MSC 2010: 65C10, 65C05, 68U20, 11K45

Abstract

The generation of quasi-random numbers is one of the most important problems in the Monte Carlo method. In this paper, the author introduces a simple and fast algorithm to generate a new class of systematic numbers. An ideal case in which one can apply these numbers is the Monte Carlo integration.

Introduction:

The Monte Carlo method needs a good random number generator and the efficiency of the Monte Carlo method is dependent on its quality. It is clear that computers cannot generate truly random numbers. To apply the Monte Carlo method on a computer, a pseudo-random or quasi-random number generator is used to approximate the random numbers (Faure & Lemieux, 2009; Hellekalek, 1998; Lemieux, 2009). A pseudo-random number generator is a mathematical method for generating a deterministic sequence that has the properties of random numbers. It is known that pseudo-random number generators cannot achieve optimal discrepancy (slight deviation from the uniform distribution) (Morokoff & Caflisch, 1994); hence, researchers have searched for new methods of generating low-discrepancy sequences. Common sequences include the Sobol, Halton, and Faure, to name a few (Niederreiter, 2010). Low-discrepancy sequences, also called quasi-random numbers, are those having points that are distributed in a way that approximates a uniform distribution as closely as possible. In fact, low discrepancy sequences are totally deterministic; thus, the popular phrase "quasi-random number" can be misleading. Quasi-random numbers are no longer independent but do have a high degree of uniformity, which is often expressed in terms of their discrepancies. The class of Monte-Carlo methods that use low-discrepancy sequences is known as "quasi-Monte Carlo methods". Estimation of an integral is the main application of a quasi-Monte Carlo method (Kocis & Whiten, 1997).

The present study introduces a new, simple, and fast algorithm to generate numbers that are uniformly distributed over the (0, 1) and which we call "systematic numbers" (SN). This new sequence of numbers improves the exactitude of simulated results for test-integrals.

Systematic Numbers :

Most methods for generating random numbers are based on the recursive equation:

$$u_{n+1}=f(u_n),\ u\in U$$

where f is the transition function and maps finite set u_n into u_{n+1} and u_o is the seed that is usually determined by the user. In this work, to generate n quasi-random numbers on $(0,1)$, we set:

$$u_k=(u_o+\frac{k}{n})-[u_o+\frac{k}{n}],\ k=0,1,2,.....n-1$$

Or

$$u_k=(u_{k-n}+\frac{1}{n})-[u_{k-n}+\frac{1}{n}],\ k=0,1,2,.....n-1$$

where u_o is selected randomly or is constant for $(0,1)$ and $[.]$ denotes the integer part function. We can write the following algorithm to generate these n systematic numbers.

Algorithm 1

1. Set $u_o=\frac{\pi}{10}$ (u_o can also be obtained using computer time)
2. Get n
3. $k=1$
4. $u_k=(u_o+\frac{k}{n})-[u_o+\frac{k}{n}]$
5. $k=k+1$
6. If $k<1$, then go to 4
7. End

The histogram of 100 systematic numbers generated by this algorithm and 100 random numbers generated by the Rand function from MATLAB is shown in Fig.-1.

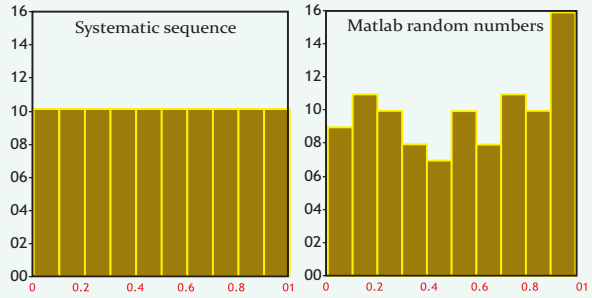

Figure-1: Histograms of 100 points generated by SN and 100 random numbers generated by MATLAB.

Author: mmoradi@guilan.ac.ir

ISSN- 2348 5191 (Print) & 2348 8980 (Electronic)

Ambient Science, 2016: Vol. 03(2); Online
DOI:10.21276/ambi.2016.03.2.ga01

This algorithm divides the interval (0, 1) into equal parts and selects a point in each part. This trick generates exactly uniform random numbers on the (0, 1). Using the same method, we can generalize the algorithm to *m*-dimensional.

It is well known that the Monte Carlo integration is more effective for multidimensional integrals, so it is necessary to use multidimensional quasi-random numbers (Sobol & Asotsky, 2003). The following algorithm generates the 2D systematic numbers.

Algorithm 2

1. Get n as the number of points.

2. Set $u_{11}=\frac{\pi}{3}$ $j=0$, $k=0$, $sn=round(\sqrt{n})$, $i=1$

3. $u_{i1}=u_{11}+\frac{j}{sn}[u_{11}+\frac{j}{sn}]$, $u_{i2}=u_{11}+\frac{k}{sn}[u_{11}+\frac{k}{sn}]$,

4. $k=k+1$ 5. if $k==sn$, then: $k=0$, $j=j+1$

6. $i=i+1$ 7. if $i<=sn^2$ 1, goto 3 8. End

Figure 2 shows the 1024 points generated by this algorithm.

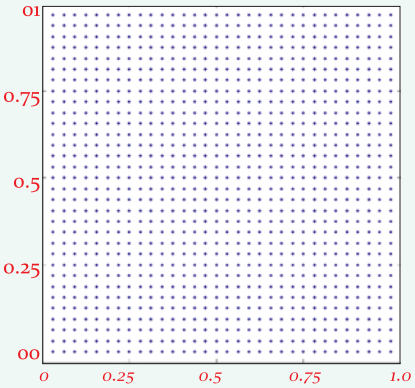

Figure-2: 1024 points generated by Algorithm 2

Test integrals:

To compare the efficiency of the SN introduced in this paper, we perform empirical comparisons based on different types of test functions. Consider test-function $f(x)$ where the integral $I(f) = \int_{[0,1]^s} f(x)dx$ can be computed exactly. Here, we use the Monte Carlo approximation

$$I(f) \approx \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} f(x_n)$$

and compare the results from SN with quasi-random number generators and MATLAB random numbers.

Most researchers use a large number of test-functions in comparative studies involving quasi-random numbers. A very good explanationof these different functions has been provided by B. Owen (Owen, 2003). Consider three classes of test function:

$$f_1(x)=\prod_{j=1}^{s}|4x_j-2|,\ f_2(x)=\prod_{j=1}^{s}1+c(x_j-0.5),\ f_3(x)=\prod_{j=1}^{s}\frac{\pi}{2}sin(\pi x_j)$$

where $x=(x_1,x_2,....x_s)$. We want to calculate the values of:

$$I_1=\int_0^1\int_0^1....\int_0^1\prod_{j=1}^{s}|4x_j-2|\,dx_s...dx_2\,dx_1=(\int_0^1|4x_1-2|\,dx_1)...(\int_0^1|4x_s-2|\,dx_s)$$

$$I_2=\int_0^1\int_0^1....\int_0^1\prod_{j=1}^{s}1+c(x_j-0.5)dx_s^i...dx_2\,dx_1=(\int_0^1 1+c(x_1-0.5)dx_1)...$$

$$(\int_0^1 1+c(x_s-0.5)dx_s)$$

$$I_3=\int_0^1\int_0^1....\int_0^1\prod_{j=1}^{s}|\frac{\pi}{2}sin(\pi x_j)dx_s...dx_2\,dx_1=(\int_0^1\frac{\pi}{2}sin(\pi x_1)dx_1)...$$

$$(\int_0^1\frac{\pi}{2}sin(\pi x_s)dx_s)$$

The exact values of I_1, I_2, and I_3 are equal to 1.

Here, we consider the same combinations (c, s) for f_2 that have been used by Sobol & Asotsky (2003), that is, (0.1, 120), (0.25, 96), and (1, 150). We tested the quality and efficiency of our SN method and compared the results with those from the Halton sequence, scrambled Halton sequence (Kocis & Whiten, 1997) and MATLAB random numbers. The results are summarized in Tables 1 to 3. It can be seen that the SN introduced in this paper provided the best performance.

Conclusion:

This paper proposed a new and simple method to generate n systematic numbers. Empirical study shows that the new generator provides better estimations of the Monte Carlo integration.

References:

Faure, H., & Lemieux, C. (2009): Generalized Halton sequences in 2008: A comparative study. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 19(4): #15.

Hellekalek, P. (1998): Good random number generators are (not so) easy to find. *Mathem. Comp. Simul.*, 46(5-6):485-505.

Kocis, L. & Whiten, W.J. (1997): Computational Investigations of Low-discrepancy Sequences. *ACM Trans. Math. Softw.*,23(2): 266-294.

Lemieux, C. (2009): **Monte carlo and quasi-monte carlo sampling**. Springer Science & Business Media.

Morokoff, W. & Caflisch, R. (1994): Quasi-Random Sequences and Their Discrepancies. *SIAM J. Sci. Comp.*, 15(6):1251-1279.

Niederreiter, H. (2010): **Quasi-Monte Carlo Methods**. John Wiley & Sons, Ltd.

Owen, A.B. (2003): The dimension distribution and quadrature test functions. *Stat. Sinica*, 13(1), 118.

Sobol', I.M. & Asotsky, D.I. (2003): One more experiment on estimating high-dimensional integrals by quasi-Monte Carlo methods. *Math. Comput. Simul.*, 62(36):255-263.

Ambient Science, 2016: Vol. 03(2); Online
DOI:10.21276/ambi.2016.03.2.ga01

Table-1: Approximation of $f_1(x)$ by SN, Halton sequence, S.Halton, and MATLAB random numbers

		N=1000		N=100,000	
		Value Estimated	Absolute Error	Value Estimated	Absolute Error
S=20	SN	0.999999999999993	6.6613381477e-15	1.000000000000022	2.22044604926e-14
	Matlab	1.028331439360769	2.8331439361e-02	0.995788337638417	4.2116623616e-03
	Halton	1.000109492474432	1.0949247443e-04	0.999946427917515	5.3572082485e-05
	S.Halton	1.001254242199163	1.2542421992e-03	0.999989756426128	1.02435738726-05
S=30	SN	1.000000000000013	1.3322676295e-14	0.999999999999929	7.10542735760e-14
	Matlab	1.024640850457242	2.4640850457e-02	0.999347548255679	6.52451744321e-04
	Halton	1.60153593811404	6.015359381e-01	1.623653216752387	6.236532167526-01
	S.Halton	0.434126363785991	5.6587363621e-01	0.435678536399603	5.643214636006-01
S=40	SN	1.000000000000018	1.7763568394e-14	1.000000000000057	5.684341886086-14
	Matlab	1.013982768993721	1.3982768994e-02	1.001720175880462	1.720175880466-03
	Halton	1.60782162765345	6.0782162765e-01	1.623680501642655	6.236805016436-01
	S.Halton	0.432093504513543	5.6790649549e-01	0.435623832978769	5.643761670216-01
S=50	SN	0.999999999999991	8.6597395920e-15	1.000000000000022	2.22044604925e-14
	Matlab	0.995698645151500	4.3013548486e-03	1.057488369027108	5.748836902716-02
	Halton	1.623646054340081	6.2364605434e-01	1.610678575743374	6.106785757436-01
	S.Halton	0.43562537317642	5.6437462682e-01	0.426071248325463	5.739287516756-01

Table 2. Approximation of $f_2(x)$ by SN, Halton sequence, S.Halton, and MATLAB random numbers

		N=1000		N=100,000	
		Value Estimated	Absolute Error	Value Estimated	Absolute Error
S=96	SN	0.999522134262453	4.7786573756e-04	1.000034734255789	3.473425578936-05
	Matlab	0.993843290667582	6.1567093324e-03	0.999192911010441	8.070889895586-04
	Halton	1.089144226083116	8.9144226083e-02	1.101326495555152	1.013264955556-01
	S.Halton	0.976196204050666	2.3803795949e-02	0.972759433848584	0.027240566156-02
S=120	SN	0.999812226104974	1.8777389503e-04	1.00001525254471	1.525254470966-05
	Matlab	1.001770650427579	1.7706504276e-03	0.999614599961894	3.854000381056-04
	Halton	1.034614310640393	3.4614310640e-02	1.040519285533055	4.051928553316-02
	S.Halton	0.990856133027420	9.1438669725e-03	0.989105719950683	1.089428004936-02
S=150	SN	0.999291116505083	7.0888349492e-04	1.000146513710361	1.465137103616-04
	Matlab	0.964558500984803	3.5441499015e-02	0.988995475910523	1.100452408956-02
	Halton	1.314443051137544	3.1444305114e-01	1.404764983506390	4.047649835066-01
	S.Halton	0.905627280392289	9.4372719608e-02	0.891290297714902	1.087097022856-04

Table 3.Approximation of $f_3(x)$ by SN, Halton sequence, S.Halton, and MATLAB random numbers

		N=1000		N=100,000	
		Value Estimated	Absolute Error	Value Estimated	Absolute Error
S=25	SN	1.000003097671240	3.0976712402e-06	0.999999999232465	7.67535368595e-10
	Matlab	0.955371573799878	4.4628426200e-02	1.001902903198451	1.902903198456-03
	Halton	1.003066902925894	3.0669029259e-03	1.000047435915923	4.743591592306-05
	S.Halton	1.002213224917115	2.2132249171e-03	0.999972470323350	2.752967664976-05
S=50	SN	1.000005652592921	5.6525929211e-06	0.999999998691862	1.308138375406-09
	Matlab	0.979808974202336	2.0191025798e-02	1.001764870381329	1.764870381336-03
	Halton	0.460205777238901	5.3979422276e-01	0.457456718945169	5.425432810556-01
	S.Halton	1.501470386876410	5.0147038688e-01	1.479691270700559	4.796912707016-01
S=100	SN	1.000005868700544	5.8687005442e-06	0.999999997836444	2.163555890006-09
	Matlab	0.966817291185195	3.3182708815e-02	1.009222481323536	9.222481323546-03
	Halton	0.458767713030401	5.4123228697e-01	0.457439117772408	5.425608822286-01
	S.Halton	1.518004041967035	5.1800404197e-01	1.479178036798521	4.791780367996-01
S=200	SN	1.000006308209266	6.3082092663e-06	0.999999996725595	3.274405413486-09