Journal of Engineering and Applied Sciences
MEDWELL PUBLICATIONS

A Review of Pseudo-Random Numbers Generation Techniques

Emeralda Gustria Sesari, Burhanuddin Dirgantoro, Casi Setianingsih and Tito Waluyo Purboyo
Department of Computer Engineering, Faculty of Electrical Engineering, Telkom University, Bandung, Indonesia

Key words: Pseudo-random numbers, pseudo-random numbers generator, random generation test, applications, especially

Corresponding Author:
Emeralda Gustria Sesari
Department of Computer Engineering, Faculty of Electrical Engineering, Telkom University, Bandung, Indonesia

Page No.: 3107-3111
Volume: 15, Issue 16, 2020
ISSN: 1816-949x
Journal of Engineering and Applied Sciences

Abstract: Random numbers have many applications, including in the case of cryptography and randomization of a test item. In fact, random numbers that are true random are very difficult to generate, especially by using computers. Therefore, the solution is by generating a pseudo-random numbers generator. This study discusses the pseudo-random numbers generation techniques and the test on the uniformity and independence of the sequence. The purpose of this study is to observe the development of pseudo-random numbers generator from the simplest and well-known form to the complex one.

INTRODUCTION

Random Number Generators (RNG) have an important role in simulations of computer such as to model the inherent randomness of components[1].

Random number generators are classified in two boar classes: Hardware RNGs and Algorithmic RNGs. The beginning of developing random numbers generation is by using a physical device which is relying on external sources to produce random numbers. Nowadays, people use a computer program to produce a random or disordered sequence. In choosing which type of RNG is appropriate, should depend on the application of the random numbers itself.

This study only focuses on algorithmic RNGs, especially on Pseudo Random Number Generators (PRNG). There are seven methods of PRNGs and several random number generator tests, to investigate the uniformity and independence.

Literature review

Linear Congruential Method (LCM): Linear congruential method generators have a function as shown below:

$$f(x) = (ax+c) \bmod m$$

Where:
a = Multiplier
c = Increment
m = The modulus

A sequence of integers X_1, X_2, ..., is generated by following function:

$$X_{i+1} = (aX_i+c) \bmod m, f\ 0, 1$$

The output of the generator will be affected by the selection of a, c, m and X_0. The cycle length of LCM can be maximized by following these three conditions[2]:

- Increment c is relatively prime to m
- a 1 is a multiple of every prime dividing m
- a 1 is a multiple of 4 when m is a multiple of 4

After the random integers are being generated from 0 to m 1, they can be converted to random numbers by:

$$R_i = \frac{x_i}{m},\ f\ 1, 2$$

MATERIALS AND METHODS

Random Numbers: Random numbers is a series of independent random numbers with a specified distribution. Random numbers are symbolized by U and the value is from 0-1 then expressed in U(0,1). There are many ways in order to get random numbers, either with the help of random numbers table, computer or using random numbers generator.

Properties of random numbers: Random numbers have three characteristics. Random numbers have to have a uniform distribution, random numbers that will be taken should have an equal probability and each random number is independent. Random numbers must be drawn separately from uniform distributions with Cumulative Distribution Function (CDF) and Probability Density Function (PDF):

$$f(x.) = \begin{cases} 1, & 0 \le x \le 1 \\ 0, & \text{otherwise} \end{cases}$$

$$F(x) = \begin{cases} 0, & x < 0 \\ x, & 0 \le x \le 1 \\ 1, & x > 1 \end{cases}$$

$$E(X) = \frac{1}{2},\ V(X) = \frac{1}{12}$$

Applications of random numbers: Numbers that are "chosen at random" are useful in many types of applications. Such as sampling, numerical analysis, simulation, decision making, computer programming, cryptography, aesthetics and recreation.

In sampling, a random sample is required to represent all possible cases on examining particular behaviour. Random numbers are also used to solve complicated numerical problems. For simulation, random numbers are used to make things that simulated by computer even more realistic. This application often used in the study of nuclear physics about random collisions. In regard to decision making, many people make their decision by randomly flipping a coin or by guessing, etc. In computer

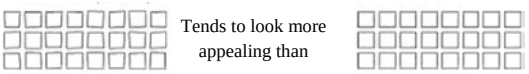

Fig. 1: Random numbers for aesthetics

programming, random value is a good source of data to test the effectiveness of certain algorithms. Random values are important to make a completely "unbiased" decision. For cryptography, unbiased data or bits is crucial for securing communications. A touch of randomness also makes graphics and music that is generated by computer seem more lively. A pattern like.

Rolling dice, spinning roulette wheels, etc. are a fascinating thing to encounter. It takes some time to finally, Fig. 1 out the result.

Random number generator: Bruce Scheiner used these three characteristics as a definition of random number generator:

- The result (output) looks random, meaning that it passes the statistical test of randomness
- Unpredictable, meaning that even if the algorithm and the previous random numbers are known, the next random numbers must not be easily computable
- Can not be reliably reproduced, meaning that if it executed twice with the same exact input, then the result is completely different

Random number generators that comply three requirements above are called True Random Number Generators (TRNG). But those are hard to be generated, specifically by using computer. So, the other alternative is by generating pseudo-random numbers.

What is meant by random in pseudo-random number is that the numbers are hard to predict. Pseudo-random numbers are generally produced by mathematical formulas and usually generated random numbers can be repeated periodically. Such random numbers generator is called Pseudo-Random Number Generators (PRNG).

Quadratic congruential method: LCM leads to reasonably good random numbers. The sequence produced by:

$$X_{i+1} = \left((aX_i)\ \text{mod} + (m+1) + c\right) \text{mod}\ m$$

is less random. LCM can be generalized to quadratic congruential method:

$$X_{i+1} = \left(dX_i^2 + aX_i + c\right)\ \text{mod}\ m$$

where a, c and d has to be obtained in necessary and sufficient conditions. This sequence has a term of the maximum length m.

Fibonacci: Fibonacci sequence is a sequence that X_{i+1} depends on more than one of the preceding values:

$$X_{i+1} = \left(X_i + X_{i-1}\right) \bmod m$$

This generator was talked in early 1950s. Result of some generator testing shown that the sequence produced by Fibonacci sequence is not satisfactorily random. We may also consider generators of the form shown below:

$$X_{i+1} = \left(X_i + X_{i-k}\right) \bmod m$$

where, x is a comparatively large value. This form was introduced by Green, Smith and Klem, etc.

Multiple recursive generators: Multiple recursive generators are based on a generalization of linear congruential generators:

$$X_{i+1} = \left(aX_{i-1} +, ..., a_k X_{i-k} c\right) \bmod m$$

where, k = permanent integer. Therefore, the ith of series relies on the previous k.

Inversive congruential sequences: This method is suggested by Eichenauer and Lehn.

$$X_{i+1} = \left(aX_i^{-1} + c\right) \bmod m$$

Here p is prime and X_1 ranges over {0, 1, ..., p, 1, ∞}. Inverses are defined by 01 ∞∝ 0. Therefore, definition of $0^1\infty$ could simply be replaced by 0^10 for purposes of implementation.

Blum-Blum-Shub Generator: Blum *et al*.[3] linear Congruential Method is vulnerable to attacks if they are used to generate keys in a cryptosystem. BBS generator is a random bit generator and has a very strong cryptographic properties. BBS generator has the following form:

$$X_{i+1} = X_i^2 \bmod M$$

where, M = result of two big distinct primes. The outcome bit is either least significant of X_{i-1} or the parity of X_{i-1}.

Mersenne-Twister: In 1994, TT800 generator was invented by Matsumoto and Kurita using binary operations. Then, Matsumoto and Nishimura[4] fixed the utilization of binary operations and developed Mersenne-Twister. They perform on $N_2\{0, 1\}$, so, variable x is expressed by a vector ω bits (e.g., 32 bits):

$$X_{i+n} = X_{i+m} \oplus \left(X_i^{upp} \middle| X_{i+1}^{low}\right) A$$

where, n>m are constant integers, X_1^{upp} aims the upper ωr(r) bits of X_2 and A is ω×ω matrix of N_2.| is the operation where $X_1^{upp} \middle| X_{i-1}^{low}$ appends the upper ωr bits X_2 with the lower r bits X_{i-1}. After being multiplied with matrix A, Φ (exclusive-or) do an addition between the result of multiplication with bit $X_{1, m}$ and bit modulo two.

Random generation tests: Random generation tests are meant to check if the sequence produced by generator is an independent and identically distributed sequence. There are two categories of tests: first is testing for uniformity and the second is testing for independence.

The basic ideas of testing a random number generator are by using hypothesis. For an example, use the test in uniformity. There are two hypothesis, the first one is random number generator is uniformly distributed which is denoted by H_0 or known as null hypothesis. While alternative hypothesis is when the random numbers generated by the generator is not uniformly distributed (H_1).

Frequency test: The first random generation test is a frequency test. Frequency test is one of uniformity test. There are two different methods to do frequency test, Kolmogorov-Smirnov and the chi-square test. Those tests above measure agreement between sample distribution and theoretical uniform distribution.

Kolmogorov-Smirnov test compares the CDF F(x) (uniform distribution) to the empirical CDF $S_N(X)$ of the N samples observations:

$$F\left(x\right) = x,\ 0 \le x \le 1$$

$$S_N\left(x\right) = \frac{\text{number of } R_1, R_2, ..., R_N \le x}{N}$$

If N becomes bigger, $S_N(x)$ should approach F(x). This test build upon the statistic:

$$D = \max\left|F\left(x\right) - S_N\left(x\right)\right|$$

The Chi-square test have the same point of view as Kolmogorov-Smirnov test but uses different method. Chi-square looks at the deviation from the expected value.

$$x_0^2 = \sum_{i=1}^{n} \frac{(0_i - E_i)^2}{E_i}$$

Where:

n = The intervals
0_1 = The total of samples
E_i = Intended total of samples. If the sample size is N

$$E_i = \frac{N}{n}$$

Runs tests: There are three kinds of tests in runs test: runs up and runs down, runs above and runs below the mean and length of runs.

Runs up and runs down test examines the independence from the arrangement of number in a sequence. A run is obtained from a series of identical events proceeded and followed by a distinct event. For example in a sequence of coin tossing may have:

HH THT THHT THH

There are seven runs in this sequence, first with a length two, second and third with a length one. Fourth, fifth, sixth and seventh with a length two. If is the number of runs in a sequence, mean and variance of a is:

$$\mu_a = \frac{2N-1}{3}$$

Runs above and below the mean test assure that the sequence is random. n_1 and n_2 are total individual surveillances above and below the mean, b is the number of runs. The mean and variance are:

$$\mu_b = \frac{2n_1n_2}{N} + \frac{1}{2}$$

$$\sigma_b^2 = \frac{2n_1n_2(2n_1n_2-N)}{N^2(N-1)}$$

If n_1 or n_2 is larger than 20, b is normally distributed:

$$Z_0 = \frac{b-(2n_1n_2/N)-1/2}{\left[\frac{2n_1n_2(2n_1n_2-N)}{N^2(N-1)}\right]^{1/2}}$$

Hypothesis of independence is said to be true if:

$$-z_{\alpha/2} \le Z_0 \le z_{\alpha/2}$$

where the stage of significance is denoted by α. The other runs test is to test whether a length of runs is random or not where Y_1 is a total runs of length i in a sequence of N. The value for runs up and down for i≤N 2 is:

$$E(Y_i) = \frac{2}{(i+3)}\left[N(i^2+3i+1)-(i^3+3i^2-i-4)\right]$$

and

$$E(Y_i) = \frac{2}{N!}$$

for i N 1. The value for runs above and below the mean is approximated by:

$$E(Y_i) = \frac{Nu_i'}{E(I)},\ N>20$$

where u'_i and E(i) are given by:

$$u_i' = \left(\frac{n1}{N}\right)^i\left(\frac{n2}{N}\right) + \left(\frac{n1}{N}\right)\left(\frac{n2}{N}\right)^i$$

$$E(I) = \frac{n_1}{n_2} + \frac{n_2}{n_1},\ N>20$$

Total amount of runs (off all length) in length N:

$$E(A) = \frac{N}{E(I)},\ N>20$$

Tests for auto-correlation: This test are related in reliance between every random numbers or m numbers starting with th number in a sequence:

$$R_i, R_{i+m}, R_{i+2m}, ..., R_{i+(M+1)m}$$

where i|(M|1)m≤ and N is the number of values in sequence. The distribution of auto-correlation ρ_{im} is denoted as ρ_{im}:

$$\hat{\rho}_{im} = \frac{1}{M+1}\left[\sum_{k=0}^{M} R_{i+km}R_{(k+1)m}\right]-0.25$$

$$\sigma_{\hat{\rho}_{im}} = \frac{\sqrt{13M+7}}{12(M+1)}$$

Hypothesis of independence is said to be true if:

$$-z_{\alpha/2} \le Z_0 \le z_{\alpha/2}$$

where:

$$Z_0 = \frac{\hat{\rho}_{im}}{\sigma_{\hat{\rho}im}}$$

RESULTS AND DISCUSSION

This study will discuss some of the methods and properties contained in pseudo-random number

Table 1: Review of the methods and characteristics used in random number generator

Methods	Researchers	Discussion
Probability Density Function (PDF)	Johnathan Mun	In mathematics, a probability density function represents a continuous probability distribution that is defined via its integration property. The probability of interval |a,b| is specified by $\int_a^0 f(x)dx$. The total integral of the function must be 1.0
Cumulative Distribution Function (CDF)	Johnathan Mun	The cumulative distribution function is defined as P(x) P(X≤x) Probability of X is less than or similar to x. The limits has the following characteristics: lim∞i(x) 0 and lim∞i(x) 1. The probability of interval |x, ∞| is specified by $\int_{\infty}^{x} f(x)dx$
Unpredictable	Lagarias[5], L'Ecuyer and Proulx[6] and Ritter[7]	The numbers are said to be random if unpredictable by looking at the previously generated numbers, even with unlimited resources. Random numbers generator is considered as unpredictable if the prediction time of the next output is super-polynomial or the probability of correct prediction in polynomial time is equal to randomly guessing a value
Kolmogorov complexity	M. Hutter, W. Merkle and P. Vitanyi	The Kolmogorov complexity of x⊂{0, 1} famous as algorithmic information theory as well is highly applied in computer science and a majority of other scientific studies. The Kolmogorov complexity of an object is the minimal number of bits needed to successfully portray the object. It measures the source of computation needed to specify an object
Cryptographic properties	Bellare *et al*.[8]	Randomness is an important ingredient for cryptography. For example is randombits that are needed not only for generating keys but are also part of steps in cryptographic algorithms. Random bits will be generated by pseudo-random generator. The safety of the generated random bits relies on the excellence of the generator

generators. Such as the properties of random numbers generally PDF, CDF and level of unpredictability. Also Kolmogorov complexity that is discussed on Kolmogorov-Smirnov test and cryptographic properties in random numbers (Table 1).

CONCLUSION

We have reviewed pseudo-random number generation techniques in this study. Pseudo-random number generation have some methods based on its formula such as, Linear Congruential Method (LCM), Quadratic Congruential Method, Fibonacci Sequence, Multiple Recursive Generators, Inversive Congruential Sequences[3] and Mersenne-Twister. Meanwhile, the random number generation test is to test the distribution and independence of the sequence. There are frequency test, runs test and test for auto-correlation.

The variety of random number generation techniques can develop the use of random numbers, especially, those which generated by computers. Each method has its own disadvantages and advantages, so that, it can be adjusted to the purpose of generating such random numbers.

RECOMMENDATIONS

Future research includes developing the function, advantages and disadvantages for each pseudo-random numbers generator. By reviewing some of the methods in this study, it is hoped that there will be other effective methods that can generate random numbers better.

REFERENCES

01. Law, A.M. and W. Kelton, 2000. Simulation Modeling and Analysis. 3rd Edn., McGraw-Hill, New York, USA.,.
02. Knuth, D.E., 1997. The Art of Computer Programming Volume 2: Seminumerical Algorithms. 3rd Edn., Addison-Wesley, Boston, Massachusetts, USA., ISBN-13: 978-0201896848, Pages: 784.
03. Blum, L., M. Blum and M. Shub, 1986. A simple unpredictable pseudo-random number generator. SIAM. J. Comput., 15: 364-383.
04. Matsumoto, M. and T. Nishimura, 1998. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM. Trans. Model. Comput. Simul. (TOMACS.), 8: 3-30.
05. Lagarias, J.C., 1993. Pseudorandom numbers. Stat. Sci., 8: 31-39.
06. L'Ecuyer, P. and R. Proulx, 1989. About polynomial-time unpredictable generators. Proceedings of the International 21st Conference on Winter Simulation (WSC'89), December 4-6, 1989, ACM, Washington, D.C., USA., pp: 467-476.
07. Ritter, T., 1991. The efficient generation of cryptographic confusion sequences. Cryptologia, 15: 81-139.
08. Bellare, M., S. Goldwasser and D. Micciancio, 1997. Pseudo-random number generation within cryptographic algorithms: The DDS case. Proceedings of the Annual International Cryptology Conference (CRYPTO'97), August 17-21, 1997, Springer, Berlin, Germany, pp: 277-291.