DS Journal of Modeling and Simulation
Volume 3 Issue 1, 45-50, Jan - Mar 2025
ISSN: 2584-1564 / https://doi.org/10.59232/MS-V3I1P104

Original Article

On Parameter Selection Criteria for Linear Congruential Generators

Cyril Oseremen Odijie

Department of Statistics, University of Benin, Benin City, Nigeria.

cyril.odijie@uniben.edu

Received: 05 January 2025; Revised: 07 February 2025; Accepted: 14 March 2025; Published: 31 March 2025

Abstract - Pseudo-Random Number Generators (PRNGs), especially Linear Congruential Generators (LCGs), have been widely studied. Their desirability lies in the ability to construct one with a long period, possibly a full period. This helps to ensure that the sequence of random integers generated satisfies desirable properties of randomness and independence to a large extent. Hull and Dobell, in 1962, had given three criteria or conditions that certain parameters of LCGs should satisfy to guarantee a full period. However, implementing the criteria has eluded many users. This short paper puts clear meanings to each condition and gives several implementable examples.

Keywords - Full-period, Hull-Dobell theorem, Linear congruential generators, Pseudorandom number generator.

1. Introduction

The Linear Congruential Generator (LCG) of the additive form is given as [1, 2]

$$X_{n+1} = (aX_n + c) \bmod m, \quad n = 0,1,2,\Lambda\ ,m-1 \tag{1}$$

Where,

X_n is the sequence of pseudo-random numbers to be generated,
a, satisfying $0 < a < m$ is called the multiplier,
c, satisfying $0 \le c < m$ is called the increment, and
X_0, satisfying $0 \le X_0 < m$ is called the seed, the starting value.
If $c = 0$, the multiplicative form of the LCG is obtained [3].

The recursive formula in Equation (1) may also be written in a "lagged" form as: [4]

$$X_i = (aX_{i-1} + c) \bmod m, \quad i = 1,2,\Lambda\ ,m \tag{2}$$

Where each of the parameters retains the same meaning as before. Notice that the index n in Equation (1) is replaced by the lagged version, $i-1$, such that the range of values for the index "shifts up" by 1.

In simple terms, the modulo operation (mod) in both equations may be written to find the remainder after dividing y by x. It stems from the fact that any two positive integers x and y, possess the "divisibility" relation, $y = qx + r$, where q is called the quotient and r is the remainder. x is said to be divisible by y, in this case, if the remainder $r = 0$. Note that $y \bmod x = y$ if $y < x$ and the relation is written as $y = 0 \cdot x + r$, in this case.

The sequence of pseudo-random numbers generated is expected to possess desirable properties of randomness, uniformity and some sort of independence. Of course, randomness and independence can only be mimicked (which is the reason for the prefix "pseudo- ") as the sequence results from a deterministic recursion. Hence, a long and full period LCG is important to ensure the resulting sequence is not easily predictable, as the reverse will be fatal [5].

The criteria for a full period (see definition of period in Appendix 1) m to be guaranteed are as follows [6].

Criterion 1. m and c are coprime (or relatively prime).
Criterion 2. $a-1$ is divisible by all prime factors of m.
Criterion 3. $a-1$ is divisible by 4 if m is divisible by 4.

The three criteria or requirements are sometimes called the Hull-Dobell theorem, and they are necessary and sufficient conditions to achieve a full period for the LCG. A definition of, and remarks on, the period of LCG is given in Appendix 1.

There exists a copious amount of studies regarding random number generation [7-9], some of which have found their implementation in software packages that involve the use of random numbers from statistics to gaming. While some are too "technical" for the ordinary user to implement, others are in a "ready-made" inaccessible software environment. It is, therefore, common among some practitioners to resort to selecting the parameters of the LCG using the "trial and error" method. This can be frustrating and can lead to poor sequences being generated.

2. Motivation

Some users of LCG have experienced difficulty in selecting good parameters needed to achieve a long and full-period LCG using the Hull-Dobell theorem or criteria due to a poor understanding of their interpretations and implications. For example, Park and Miller [10] had specifically stated that "good ones [random number generators] are hard to find." This short paper, therefore, attempts to give more practical meaning to each criterion or requirement in simple terms with several practical examples.

3. Materials and Methods

The following subsections clearly and concisely explain each of the three criteria.

3.1. M and C are Coprimes or Relatively Prime

This criterion appears to be the most misunderstood. m and c are coprime or relatively prime if their Greatest Common Divisor (GCD) is 1. This does not necessarily mean that m and c must be primes themselves, as is commonly mistaken. For example, $m = 2^4$ and $c = 7$. Note that for m = powers-of-two modulus types, it is better to choose c as a prime number since power-of-two numbers are not primes and, as such, would have divisors greater than 1 (e.g. 2) and prime numbers cannot divide power-of-two numbers. Also, $m = 31$ they are coprime, and both are coincidentally prime in this case. Note that a prime number has only two factors, being 1 and itself.

3.2. A - 1 is Divisible by all Prime Factors of M

This requirement is straightforward. Prime factors of m are the numbers that divide m completely and are, at the same time, prime numbers. Suppose we use the first example in subsection 2.1 for m = 16 $a = 5$. Then,

- The only prime factor of 16 is 2 (since the other factors, 1, 4, 8 and 16, are not primes)
- $a-1 = 5-1 = 4$ is divisible by 2, which is the only prime factor of 16 in this case.

However, in the second example in subsection 2.1, $m = 31$ has 31 as the only prime factor, being a prime number itself. Hence, this criterion cannot be verified since whatever value a is chosen within the interval $0 < a < m$, $a-1$

cannot be divisible by 31 as $a-1$ it is less than 31 in this case. It is, therefore common to set $a=1$ for a prime modulus since $a-1=0$ is divisible by any number. However, a little "trick" still works (that is, it guarantees full period for prime moduli) – choose $a=m+1$ if m is a prime modulus. This ensures that $a-1=m$, it is divisible by the modulus m, which is the only prime factor in that case.

Although this trick meets the second criterion as needed, a it does not fall within its interval of definition, namely, 0 < a < m. Prime number moduli are commonly used in the multiplicative form of LCG, i.e. when c = 0 [3] These types usually have periods $\leq m-1$, which are considered as the full period possible for this form by some authors (see e.g. [4]).

3.3. A - 1 is Divisible by 4 if M is Divisible by 4

This appears to be the most straightforward criterion and is very clear to understand. Using the example in subsection 2.2, we can see that it is divisible by 4, and so is $a-1=4$. Now, all power-of-two numbers are divisible by 4. The focus is, therefore, on choosing the values of a and c properly such that m (the power-of-two modulus type in this case) is coprime with c and $a-1$ is divisible by 4.

4. Results and Discussion

4.1. Summary of Results from the Method Discussed

The points made so far for achieving full-period LCGs are summarized in Table 1. Note that the choice X_0 is trivial since it only affects the order of appearance of the terms of the sequence X_n and has nothing to do with the period, which is our main point of interest in this paper.

Table 1. Summary of parameter selection criteria guide for full-period LCG guarantee

Nature of m	Choice(s) of a	Choice(s) of c	X_0
Power-of-two	$0<a<m: a-1 \bmod 4=0$, i.e. $a=1$ or $a=4i+1=5, 9, 13, \Lambda$ where $i \geq 1$ is an integer	$c=1$, a prime or a Mersenne prime (i.e. 2^n-1, where n is a prime)	Any number in $\{0, 1, 2, \Lambda, m-1\}$
Prime	$a=1$ or $a=m+1$ (the trick! Works, but violates $0<a<m$)	$c<m: m \bmod c \neq 0$ i.e. any number less than m and does not divide m	

4.2. Practical Examples and Empirical Analysis

Following the parameter selection guide in the foregoing, some good choices of parameters for a full-period LCG in quick manually computable examples include:

4.2.1. Power-of-Two Modulus Type

- $m=8,\ a=1,\ c=3,\ \ X_0=4,$
- $m=16,\ a=5,\ c=5,\ \ X_0=2$
- $m=32,\ a=9,\ c=7,\ \ X_0=3.$

4.2.2. Prime Modulus Type

- $m=7,\ a=8,\ c=2,\ \ X_0=5,$ (the "trick" example).
- $m=11,\ a=1,\ c=7,\ \ X_0=10,$
- $m=19,\ a=1,\ c=13,\ \ X_0=8.$

The first example $m = 8,\ a = 1,\ c = 3,\ \ X_0 = 4,$ is demonstrated as follows:

Using Equation (1), the recursive formula is given as:

$X_{n+1} = (X_n + 3) \bmod 8, \quad n = 0, 1, 2, \Lambda\ , 7.$
For $n = 0$,
$X_1 = (X_0 + 3) \bmod 8 = (4 + 3) \bmod 8 = 7 \bmod 8 = 7.$
For $n = 1$,
$X_2 = (X_1 + 3) \bmod 8 = (7 + 3) \bmod 8 = 10 \bmod 8 = 2.$
For $n = 2$,
$X_3 = (X_2 + 3) \bmod 8 = (2 + 3) \bmod 8 = 5 \bmod 8 = 5.$
For $n = 3$,
$X_4 = (X_3 + 3) \bmod 8 = (5 + 3) \bmod 8 = 8 \bmod 8 = 0.$
For $n = 4$,
$X_5 = (X_4 + 3) \bmod 8 = (0 + 3) \bmod 8 = 3 \bmod 8 = 3.$
For $n = 5$,
$X_6 = (X_5 + 3) \bmod 8 = (3 + 3) \bmod 8 = 6 \bmod 8 = 6.$
For $n = 6$,
$X_7 = (X_6 + 3) \bmod 8 = (6 + 3) \bmod 8 = 9 \bmod 8 = 1.$

Therefore, the sequence of pseudorandom integers generated is 4, 7, 2, 5, 0, 3, 6, and 1, 8 in number (without repetition), indicating a full period.

The reader is encouraged to try to deliberately violate these guidelines or rules and observe the impact on the period of the sequence so generated. For instance, try $m = 8$, $a = 1$, $c = 2$, $X_0 = 4$ and observe a period of 4 (that is, half the possible full period) with the sequence generated is 4, 6, 0, 2, 4, 6, 0, 2. The sequence repeats itself after the fourth step. This is because $m = 8$, and $c = 2$ are not relatively primes as 2 divides 8 completely. Also, try $m = 11$, $a = 7$, $c = 1$, $X_0 = 2$ and observe a period of 10 (i.e. $m - 1$). In this case, the sequence only has the seed repeated. For a more extended, but not entire, period of $m - 1 = 100$, consider $m = 101$, $a = 11\, c = 13$ and $X_0 = 5$ then contrast with when $a = 1$ (instead of 11) for the same example, which now yields a full period of 101. For longer periods, such as those found in some known programming languages and domains like FORTRAN, C++, etc., see e.g. [4, 11] for tables of good parameter (multiplier) choices.

5. Conclusion

In summary, a full-period LCG generates pseudorandom integers, $X_n \in \{0, 1, 2, ..., m - 1\}$ with period = m, while those not having full period have cycles that repeat before the possible full period of m, that is, they generate integers, $X_n \in \{0, 1, 2, ..., m'\}$ where $m' < m - 1$. Having a full period is desirable for any LCG to ensure that it meets the basic properties of randomness, uniformity and independence to a great extent. This paper clearly explains how to carefully select the parameters of a linear congruential generator to achieve that end. Furthermore, an example code written in MATLAB® to construct a full-period ($m = 2^{20}$ = 1,048,576) LCG is given in Appendix 2. The reader can copy and paste it into an M-file in MATLAB®, if possible, to try each of the examples given in this text (and any other) by changing the parameter values. There are other forms of random number generators besides the LCG, from the earliest Lehmer generator and Linear-feedback shift register to the more recent Mersenne Twister and Xorshift, among others [12-15]. These generators have their own advantages, limitations, and areas of misconceptions that may require further elucidation in future research.

References

[1] W.E. Thomson, “A Modified Congruence Method of Generating Pseudo-random Numbers,” *The Computer Journal*, vol. 1, no. 2, 1958. [CrossRef] [Google Scholar] [Publisher Link]

[2] A. Rotenberg, “A New Pseudo-Random Number Generator,” *Jouranl of the ACM*, vol. 7, no. 1, pp. 75-77, 1960. [CrossRef] [Google Scholar] [Publisher Link]

[3] Pierre L’Ecuyer, “Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure,” *Mathematics of Computation*, vol. 68, no. 225, pp. 249-260, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[4] G.L. Steele Jr., and Sebastiano Vigna, “Computationally Easy, Spectrally Good Multipliers for Congruential Pseudorandom Number Generators,” *Software: Practice and Experience*, vol. 52, no. 2, pp. 443-458, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[5] Pierre L'Ecuyer, “History of Uniform Random Number Generation,” *Winter Simulation Conference*, Las Vegas, NV, USA, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[6] T.E. Hull, and A.R. Dobell, “Random Number Generators,” *SIAM Review*, vol. 4, no. 3, pp. 230-254, 1962. [CrossRef] [Google Scholar] [Publisher Link]

[7] G.S. Fishman, and L.R. More III, “An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus 2^{31}-1,” *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 1, pp. 24-45, 1986. [CrossRef] [Google Scholar] [Publisher Link]

[8] Pierre L'Ecuyer, François Blouin, and Raymond Couture, “A Search for Good Multiple Recursive Random Number Generators”, *ACM Transactions on Modeling and Computer Simulation*, vol. 3, no. 2, pp. 87-98, 1993. [CrossRef] [Google Scholar] [Publisher Link]

[9] Hui-Chin Tang, “An Analysis of Linear Congruential Random Number Generators When Multiplier Restrictions Exist,” *European Journal of Operational Research*, vol. 182, no. 2, pp. 820-828, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[10] S.K. Park, and K.W. Miller, “Random Number Generators: Good Ones are Hard to Find,” *Communications of the ACM*, vol. 31, no. 10, pp. 1192-1209, 1969. [CrossRef] [Google Scholar] [Publisher Link]

[11] W.H. Payne, J.R. Rabung, and T.P. Bogyo, “Coding the Lehmer Pseudo-Random Number Generator,” *Communications of the ACM*, vol. 12, no. 2, pp. 85-86, 1969. [CrossRef] [Google Scholar] [Publisher Link]

[12] Derrick H. Lehmer, “Mathematical Methods in Large-Scale Computing Units,” *2nd Symposium on Large-Scale Digital Calculating Machinery*, pp. 141-146, 1951. [Google Scholar]

[13] Robert C. Tausworthe, “Random Numbers Generated by Linear Recurrence Modulo Two,” *Mathematics of Computation*, vol. 19, no. 90, pp. 201-209, 1965. [CrossRef] [Google Scholar] [Publisher Link]

[14] Makoto Matsumoto, and Takuji Nishimura, “Mersenne Twister: A623-dimensionally Equidistributed Uniform Pseudo-Random Number Generator,” *ACM Transactions on Modelling and Computer Simulation*, vol. 8, no. 1, pp. 3-30, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[15] George Marsaglia, “Xorshift RNGs,” *Journal of Statistical Software*, vol. 8, no. 14, pp. 1-6, 2003. [CrossRef] [Google Scholar] [Publisher Link]

Appendix 1

Definition (Period of LCG). The period of a Linear Congruential Generator (LCG) is the number of random integers generated before the cycle is repeated.

Remark 1: The period of a linear congruential generator is not the highest integer generated before the cycle is repeated, as it is easily confused.

Remark 2: Uniform pseudo-random numbers $U_i \in [0, 1]$ are obtained by the formula:

$$U_i = \frac{X_i}{m},\ i = 0,1,2,\Lambda\ ,m-1$$

Which can be tested for uniformity, randomness and independence by using standard methods such as comparing the mean and variance of the sequence U_i with the theoretical mean and variance of the uniform distribution (i.e. 1/2 and 1/12, respectively) and plotting a correlogram U_i. If the biases of the mean and variances are small, then the test for uniformity is passed, and if the plotted lags in the correlogram fall within the bounds, then the test for randomness and independence are passed, all to a large extent (usually 95% by default). The trivial value of $U_i = 0$ may be expunged from the sequence altogether by expunging $X_i = 0$ from the sequence of pseudorandom integers.

Appendix 2

A code is written in MATLAB® for executing a long full-period LCG with a detailed report.

```
X = zeros; %To store the sequence of pseudo-random integers to be generated.
U = zeros; %To store the pseudo-random uniform numbers obtained from U.

%Choose parameters
m = 2^20;            %power-of-two type.
a = 2^14 + 1;        %satisfying 0 < a < m : (a - 1) is divisible by 4.
c = 2^11 - 1;        %a Mersenne Prime.
Xo = 2^10;           %any number between 0 and m (inclusive) would do.

X(1) = Xo            %X(1) because MATLAB indexes start from 1 and not 0.
U(1) = X(1)/m        %The first uniform pseudo-random number obtained from Xo.

%The loop or recurrence formula:
for n = 1:m-1
  X(n+1) = mod((a*X(n) + c), m);
  U(n)=X(n)/m;
end

%Display the results in a table after some preamble talks. Comment or delete the
%next 9 lines of executable codes to avoid longer runtime of displaying all the
%results if the period is large, as in this example.

fprintf(['The LCG of the form: Xn+1 = (aXn + c) mod m, with\nm = %.0f,\n'...
'a = %.0f,\nc = %.0f,\nXo = %.0f,\nhas the following results:\n'], m, a, c, X(1))
disp('====================')
disp('i     X(i)    U(i)  ')
disp('--------------------')
for i = 1:length(X)
    fprintf('%-3.0f\t %3.0f\t %6.4f\n',i-1 , X(i), U(i))
end
disp('====================')

%Check if X has repeated elements, i.e. check for full period or not, and comment.
if length(X) ~= length(unique(X))
    fprintf('There is at least one repeated value in X.\n')
end
fprintf('The period of this LCG is %.0f', length(unique(X)))
if m == length(unique(X))
    fprintf(', which is full\n')
else
    fprintf(', which is not full (since m = %.0f).\n', m)
end
```