Chapter 4:
(0,1) Random Number Generation

Refer to Text Book:
• “Simulation Modeling and ARENA”, Manuel Rossetti, Ch. 2
• “Operations Research: Applications and Algorithms” By Wayne L. Winston, Ch. 21
• “Operations Research: An Introduction” By Hamdi Taha, Ch. 16
LEARNING OBJECTIVES

▪ To be able to describe and use linear congruential pseudorandom number generation methods

▪ To be able to define and use key terms in pseudorandom number generation methods such as streams, seeds, and period.

▪ To be able to explain the key issues in pseudorandom number testing.
Review Last Lecture

- **Steps for Simulation Modeling**
  How to conduct a complete simulation modeling analysis?

- **Applications on Simulation Modeling**
  The Manufacturing model:
  - how to formulate the problem statement.
  - how to define goals of the simulation
  - how to determine the missing information
  - what are the data needed.
Today’s Lecture Plan

- Idea of Random Number Generators
- Pseudo-Random Numbers
- Linear congruential generator (LCG)
  - Definitions
  - Conditions for LCG Full Cycle
  - Examples
- Random Streams
Random Number Generation

- Generating any random number from any distribution depends on U[0,1].
Pseudo-Random Numbers

- Random means Not Known for certain
- The random numbers used in a simulation are not really random!
- You can get all the numbers in advance
Pseudo-Random Numbers

- *pseudo-random numbers*

Definition:

A sequence of *pseudo-random* numbers, \( U(i) \), is a deterministic sequence of numbers in \([0,1]\) having the same relevant statistical properties as a sequence of truly random \( U(0,1) \) numbers. (Ripley 1987)
Pseudo-Random Numbers

- *linear congruential generator (LCG)*
  - a recursive algorithm for producing a sequence of pseudorandom numbers.
  - Each new pseudorandom number from the algorithm depends on the previous pseudorandom number.
  - There must be a starting value called the *seed*
Definition 2.2 (LCG) An LCG defines a sequence of integers, $R_0, R_1, \ldots$ between 0 and $m - 1$ according to the following recursive relationship, where $i = 0, 1, 2, \ldots$:

$$R_{i+1} = (aR_i + c) \mod m \quad \text{for } i = 0, 1, 2, \ldots$$  \hspace{1cm} (2.1)

where $R_0$ is called the seed of the sequence, $a$ is called the constant multiplier, $c$ is called the increment, and $m$ is called the modulus. $(m, a, c, R_0)$ are integers with $a > 0$, $c \geq 0$, $m > a$, $m > c$, $m > R_0$, and $0 \leq R_i \leq m - 1$.

To compute the corresponding pseudorandom uniform number, we use

$$U_i = \frac{R_i}{m}$$  \hspace{1cm} (2.2)
Linear congruential generator (LCG)

- choice of the parameters of the LCG: seed, constant multiplier, increment, and modulus, that is, the will determine the properties of the sequences

- properly chosen parameters, an LCG can be made to produce pseudorandom numbers look like real random.
Linear congruential generator (LCG)

**EXAMPLE**
Consider an LCG with parameters \((m = 8, a = 5, c = 1, R_0 = 5)\). Compute the first nine values for \(R_i\) and \(U_i\) from the defined sequence.

- how to compute using the mod operator. The mod operator is defined as

\[
z = y \mod m = y - m \left\lfloor \frac{y}{m} \right\rfloor
\]

where \([\cdot]\) is the floor operator,
Linear congruential generator (LCG)

**EXAMPLE**

\[ z = y \mod m \]
\[ = y - m \left\lfloor \frac{y}{m} \right\rfloor \]

\[ z = 17 \mod 3 \]
\[ = 17 - 3 \left\lfloor \frac{17}{3} \right\rfloor \]
\[ = 17 - \lfloor 5.66 \rfloor \]
\[ = 17 - 3 \times 5 = 2 \]

In our example

- \( m = 8 \)
- \( a = 5 \)
- \( c = 1 \)
- \( R_0 = 5 \)

\[ R_1 = (5R_0 + 1) \mod 8 = \]
\[ R_2 = (5R_1 + 1) \mod 8 = \]
\[ R_3 = (5R_2 + 1) \mod 8 = \]
\[ R_4 = (5R_3 + 1) \mod 8 = \]
EXAMPLE

\[ R_1 = (5R_0 + 1) \mod 8 = 26 \mod 8 = 2 \Rightarrow U_1 = 0.25 \]
\[ R_2 = (5R_1 + 1) \mod 8 = 11 \mod 8 = 3 \Rightarrow U_2 = 0.375 \]
\[ R_3 = (5R_2 + 1) \mod 8 = 16 \mod 8 = 0 \Rightarrow U_3 = 0.0 \]
\[ R_4 = (5R_3 + 1) \mod 8 = 1 \mod 8 = 1 \Rightarrow U_4 = 0.125 \]
\[ R_5 = 6 \Rightarrow U_5 = 0.75 \]
\[ R_6 = 7 \Rightarrow U_6 = 0.875 \]
\[ R_7 = 4 \Rightarrow U_7 = 0.5 \]
\[ R_8 = 5 \Rightarrow U_8 = 0.625 \]
\[ R_9 = 2 \Rightarrow U_9 = 0.25 \]
Linear congruential generator (LCG)

**Notes to consider on LCG:**

- The $U_i$ are simple fractions involving $m = 8$.
- Certainly, this sequence does not appear very random. (pseudo-random) … Why?
- The $U_i$ can only take one of the rational values:

$$0, \frac{1}{m}, \frac{2}{m}, \frac{3}{m}, \ldots, \frac{(m-1)}{m} \quad \text{since } 0 \leq R_i \leq m - 1$$

- if $m$ is small, there will be big gaps on the interval $[0, 1)$
- if $m$ is large, then the $U_i$ will be distributed on $[0, 1)$.
Linear congruential generator (LCG)

**Cycle of LCG:**

- **Definition:** a sequence generates the same value as a previously generated value, then the sequence *cycle*.

- **Definition:** The length of the cycle is called the *period* of the LCG.

- **Definition:** the LCG is said to achieve its full period if the cycle length is equals to \( m \).

- LCG has a long cycle for good choices of parameters \( a, m, c \).

- Most computers (32-bit) has value for

\[
m = 2^{31} - 1 = 2,147,483,647
\]

represents the largest integer number.
Linear congruential generator (LCG)

**Theorem:** (LCG Full Period Conditions)
An LCG has full period if and only if the following three conditions hold:

1. The only positive integer that (exactly) divides both $m$ and $c$ is 1 (i.e., $c$ and $m$ have no common factors other than 1).

2. If $q$ is a prime number that divides $m$ then $q$ should divide $(a - 1)$ (i.e., $(a - 1)$ is a multiple of every prime number that divides $m$).

3. If 4 divides $m$, then 4 should divide $(a - 1)$ (i.e., $(a - 1)$ is a multiple of 4 if $m$ is a multiple of 4).
Linear congruential generator (LCG)

**Example**: (LCG Full Period Conditions)

To apply the theorem, you must check if each of the three conditions holds for the generator.

\[ m = 8 \ , \ a = 5 \ , \ c = 1 \]

Cond-1. \( c \) and \( m \) have no common factors other than 1: factors of \( m = 8 \) are \((1, 2, 4, 8)\), since \( c = 1 \) (with factor 1) condition 1 is true.

Cond-2. \((a - 1)\) is a multiple of every prime number that divides \( m \): The first few prime numbers are \((1, 2, 3, 5, 7)\).
**Linear congruential generator (LCG)**

**Example:** (LCG Full Period Conditions)

To apply the theorem, you must check if each of the three conditions holds for the generator.

\[ m = 8 \ , \ a = 5 \ , \ c = 1 \]

Cond. 2: \((a - 1)\) is a multiple of every prime number that divides \(m\).

The prime numbers, \(q\), that divide \(m = 8\) are \((q = 1, 2)\). Since \(a = 5\) and \((a - 1) = 4\), clearly \(q = 1\) divides 4 and \(q = 2\) divides 4. Thus, condition 2 is true.

Cond. 3: If 4 divides \(m\), then 4 should divide \((a - 1)\).

Since \(m = 8\), clearly 4 divides \(m\). Also, 4 divides \((a - 1) = 4\). Thus, condition 3 holds.
**Linear congruential generator (LCG)**

**Random Stream**

**Definition** (Random Number Stream): The subsequence of random numbers generated from a given seed is called a random number stream.

- A seed, e.g. \( R_1 = 2 \), defines a starting place in the cycle and thus a sequence.
- Small period easy to remember the random number streams
- With large \( m \) hard to remember the stream.
Linear congruential generator (LCG)

Random Stream

Seed 1
Seed 2
Seed 3
Linear congruential generator (LCG)

**Random Stream**

- choose to divide the entire sequence so that the number of non-overlapping random numbers in each stream is quite large
- computer simulation languages come with a default set of streams that divide the “circle” up into independent sets of random numbers
Linear congruential generator (LCG)

**Random Stream**

- The streams are only independent if you do not use up all the random numbers within the subsequence.
- To insure independence in the simulation, you can associate a specific stream with specific random processes in the model. For example:

2. Service time: stream 2.
3. Demand type: stream 3.
Linear congruential generator (LCG)

**Exercise**
Analyze the following LCG:
\[ X(i) = (11 \times X(i - 1) + 5) \pmod{16}, \quad X_0 = 1 \]
- What is the maximum possible period length for this generator?
- Does this generator achieve the maximum possible period length? Justify your answer.