CPSC 531:
System Modeling and Simulation

Carey Williamson
Department of Computer Science
University of Calgary
Fall 2017

Outline

- Random number generation
  - Properties of random numbers
  - Linear Congruential Generator
  - Seed selection and random streams

Random Number Generators (RNG's)

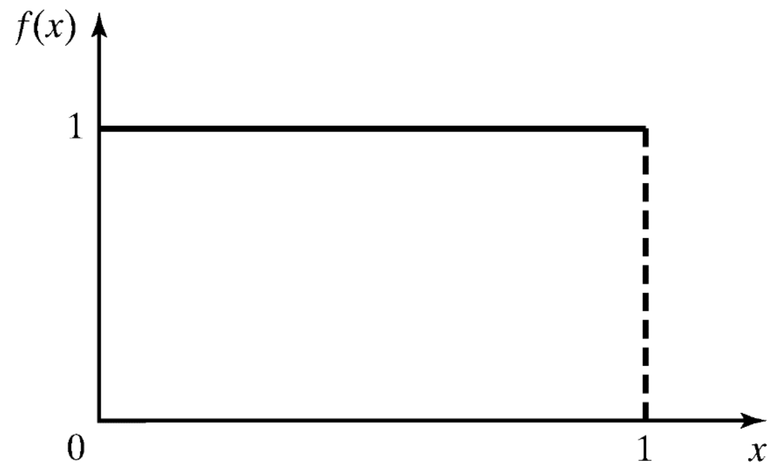
- Requirements
  - Sequence generated has uniform distribution (continuous) between 0 and 1
  - The numbers in the sequence are independent of each other
- RNG's in computer simulation are pseudorandom
  - Each number in the sequence is determined by one or several of its predecessors
  - Statistical tests can be used to determine how well the requirements of uniformity and independence are met

Properties of Random Numbers

- Two important statistical properties:
  - Uniformity
  - Independence

- Random numbers, x_1, x_2, x_3, …, must be independently drawn from a uniform distribution with PDF:

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



Desirable Properties

- Uniformity and independence
- Should be able to reproduce a given sequence of random numbers
  - Helps program debugging
  - Helpful when comparing alternative system design
- Should have provision to generate several streams of random numbers
- Computationally efficient

A Sample Generator

$$x_n = 5x_{n-1} + 1 \mod 16$$

- Starting with $x_0 = 5$:

$$x_1 = 5(5) + 1 \mod 16 = 26 \mod 16 = 10$$

- The first 32 numbers obtained by the above procedure
  10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5, 10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5.
- By dividing x's by 16:
  0.6250, 0.1875, 0.0000, 0.0625, 0.3750, 0.9375, 0.7500, 0.8125, 0.1250, 0.6875, 0.5000, 0.5625, 0.8750, 0.4375, 0.2500, 0.3125, 0.6250, 0.1875, 0.0000, 0.0625, 0.3750, 0.9375, 0.7500, 0.8125, 0.1250, 0.6875, 0.5000, 0.5625, 0.8750, 0.4375, 0.2500, 0.3125.

Linear Congruential Generator (LCG)

- Commonly used algorithm
- A sequence of integers $x_1, x_2, \ldots$ between 0 and $m-1$ is generated according to
    - $x_i = (a\, x_{i-1} + c) \bmod m$
    - where a and c are constants, m is the modulus and x_0 is the seed (or starting value)
- Random numbers $u_1, u_2, \ldots$ are given by $u_i = x_i/m$
- The sequence can be reproduced if the seed is known

Linear Congruential Generator (LCG)

- Example
  - $x_n = 7\ x_{n-1} + 3$ mod 10, $x_0 = 3$
  - sequence: 3, 4, 1, 0, 3, 4, 1, …
- Example
  - $x_n = 4\ x_{n-1} + 2$ mod 9, x_0 = 3
  - sequence: 3, 5, 4, 0, 2, 1, 6, 8, 7, 3, 5, 4, …

Properties of LCG

- Can have at most m distinct integers in the sequence
  - As soon as any number in the sequence is repeated, the whole sequence is repeated
  - **Period**: number of distinct integers generated before repetition occurs

- Problem: Instead of continuous, the u_i's can only take on discrete values $0, 1/m, 2/m, \ldots, (m-1)/m$
  - Solution: m should be selected to be very large in order to achieve the effect of a continuous distribution (typically, $m > 10^9$)
  - Approximation appears to be of little consequence

Characteristics of a Good Generator

- Maximum Density
  - Such that the values assumed by $x_i, i = 1,2, \ldots$ leave no large gaps on $[0,1]$

- Maximum Period
  - To achieve maximum density and avoid cycling
  - Achieve by: proper choice of a, c, m, and x_0

- Most digital computers use a binary representation of numbers
  - Speed and efficiency are aided by a modulus, m, to be (or close to) a power of 2

Types of LCG

- Mixed LCG
  - $c > 0$
  - Example:

$$x_n = (2^{34} + 1)x_{n-1} + 1 \mod 2^{35}$$

- Multiplicative LCG
  - $c = 0$
  - Example:

$$x_n = 5x_{n-1} \bmod 2^5$$

  - Generally performs as well as mixed LCG

Example

$$x_n = 5x_{n-1} \bmod 2^5$$

- Using a seed of $x_0 = 1$:

  5, 25, 29, 17, 21, 9, 13, 1, 5,…

  Period = 8
- With $x_0 = 2$:

  10, 18, 26, 2, 10,…

  Period is only 4

Note: Full period is a nice property but uniformity and independence are more important

Example RNGs

- A currently popular multiplicative LCG is:

$$x_n = 7^5 x_{n-1} \bmod (2^{31} - 1)$$

  - 2^{31}-1 is a prime number and 7^5 is a primitive root of it $\longrightarrow$ Full period of 2^{31}-2.

- This generator has been extensively analyzed and shown to be good

See the following book for advanced RNGs:

Numerical Recipes: The Art of Scientific Computing
http://www.nr.com/

Tips for Seed Selection

- Seed selection
  - Any value in the sequence can be used to “seed” the generator
- Do not use random seeds: such as the time of day
  - Cannot reproduce. Cannot guarantee non-overlap.
- Do not use zero:
  - Fine for mixed LCGs
  - But multiplicative LCGs will be stuck at zero
- Avoid even values:
  - For multiplicative LCG with modulus $m=2^k$, the seed should be odd
- Do not use successive seeds
  - May result in strong correlation

Better to avoid generators that have too many conditions on seed values or whose performance (period and randomness) depends upon the seed value.

Random-Number Streams

- Multi-stream simulations: need more than one random stream
  - Do not subdivide one stream: the sub-streams may be correlated
  - Use non-overlapping streams
- A random-number stream:
  - Refers to a starting seed taken from the sequence of random numbers $x_0, x_1, \ldots$
- A single random-number generator with k streams can act like k distinct virtual random-number generators
  - Choose the seeds for each stream to be far apart
  - To have streams that are b values apart, stream i could be defined by starting seed:

$$S_i = x_{b(i-1)}$$

Older generators: $b = 10^5$; Newer generators: $b = 10^{37}$

Myths About Random-Number Generation (1 of 3)

- A complex set of operations leads to random results.
  It is better to use simple operations that can be analytically evaluated for randomness.

- Random numbers are unpredictable.
  Easy to compute the parameters, a, c, and m from a few numbers => LCGs are unsuitable for cryptographic applications

Myths (2 of 3)

- Some seeds are better than others. May be true for some generators.

$$x_n = (9806x_{n-1} + 1) \bmod (2^{17} - 1)$$

  - Works correctly for all seeds except x_0 = 37911
  - Stuck at x_n= 37911 forever
  - Such generators should be avoided
  - Any non-zero seed in the valid range should produce an equally good sequence
  - Generators whose period or randomness depends upon the seed should not be used, since an unsuspecting user may not remember to follow all the guidelines

- Accurate implementation is not important.
  - RNGs must be implemented without any overflow or truncation
    For example:

$$x_n = 1103515245 x_{n-1} + 12345 \bmod 2^{31}$$

  - Straightforward multiplication above may produce overflow.