NE 155
Introduction to Numerical Simulations in Radiation Transport
Lecture 33: Random Number Generators

R. N. Slaybaugh
(well, Richard Vasques!)

April 13, 2016

Outline

1. Pseudo-random number sequence vs. *truly* random number sequence
2. Linear congruential pseudo-random number generators
3. Define the following:
   - Seed
   - Stride
   - Period

Notes derived from Paul Wilson and K. Ming Leung

Random Sequences

There are two types of random sequences: *truly random* and *pseudo-random*.

- "Truly random" is not a very well-define concept; truly random sequences are undesirable
  - Can be generated by sampling a truly random physical process (e.g. radioactive decay)
- A computer is a deterministic machine and so can never produce truly random sequences
  - It can only represent a finite number of numbers; eventually the numbers have to repeat themselves
- The generators of these "random" numbers are known as pseudo-random number generators
  - An algorithm is needed to generate sequences of numbers that appear to be random

Desired Characteristics

1. Numbers should be distributed uniformly between 0 and 1 without large "gaps"
2. Sequence of numbers generated should be as independent from each other as possible
3. Mean or average of the numbers generated should be as close to 1/2 as possible
4. The variance should be as close to 1/12 as possible
5. Should have little cyclic variations, i.e. free from the following problems:
   - autocorrelation between numbers
   - numbers successively higher of lower than adjacent numbers
   - several numbers above the mean followed by several numbers below the mean

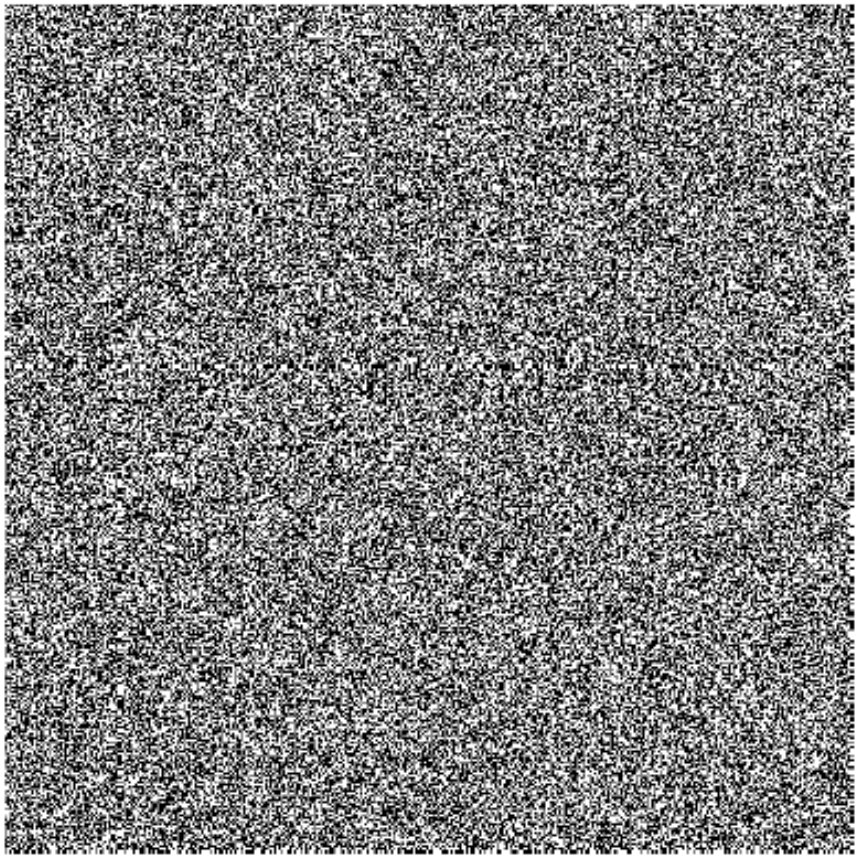
Desired Characteristics (continued)

6. Numbers should have a long cycle
7. Numbers should be replicable, i.e. the same starting condition should yield the same sequence of numbers (for debugging and comparison reasons)
8. The routine that generates these numbers should be extremely fast but should require very little memory
9. The generator should be portable

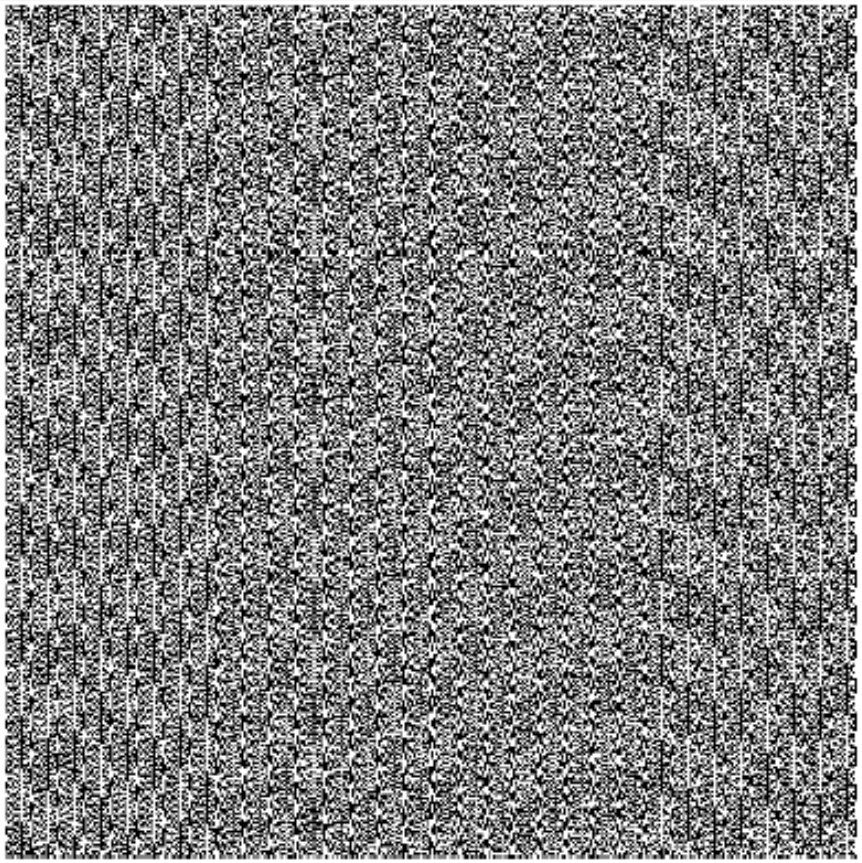
Testing Randomness

- Not a trivial pursuit
- NIST defines a set of 15 tests, based in part on different analyses of the bit sequence:
  - Frequency Test: Monobit
  - Frequency Test: Block
  - Runs Test
  - Test for the Longest Runs of Ones in a Block
  - Binary Matrix Rank Test
  - Discrete Fourier Transform (Spectral Test)
  - Non-Overlapping Template Matching Test
  - Overlapping Template Matching Test
  - Maurer's Universal Statistical Test
  - Linear Complexity Test
  - Serial Test
  - Approximate Entropy Test
  - Cumulative Sums Test
  - Random Excursions Test
  - Random Excursions Variant Test

Visual Comparison



True Random Number Generator
(Random.org)



Windows PHP `rand()` function

Linear Congruential PRNG

- Simple, fast, robust, well-understood
- According to Lehmer, 1949:

$$s_{i+1} = [s_i \times g + c] \text{mod } p$$
$$r_{i+1} = \frac{s_{i+1}}{p}$$

  where s_i, g, c, p are integers, and r_i are real.
- Often, $p = 2^m$ for integer m
- Selection of g, c, p (or m) are important. Moreover:
  - $c = 0$: "multiplicative" LCPRNG
  - $c \neq 0$: "mixed" LCPRNG

LCPRNG Parameters

- Seed s_0
  - The first number in sequence of integers
  - Generally unimportant, but must be known for the sequence to be reproducible
  - Can be changed to force a different result
- Period
  - The number of integers in the sequence before the sequence repeats
  - Generally bad to repeat sequence

Typical Parameters

Code(Author)	2^m	period	g	c
Racer (KAPL)	2^{47}	2^{45}	84,000,335,758,957	0
RCP (BAPL)	2^{48}	2^{48}	2^9+1	59,482,192,516,946
MORSE (ORNL)	2^{47}	2^{45}	5^{15}	0
MCNP (LANL)	2^{48}	2^{46}	5^{19}	0
VIM (ANL)	2^{48}	2^{46}	5^{19}	0
RANF (CRAY)	2^{48}	2^{46}	44485709377909	0

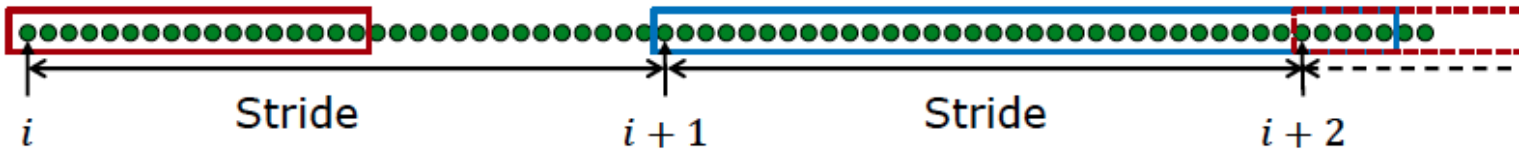
Note: 2^{45} random numbers can be generated in a few hours

General Properties of LCPRNG

- The scheme is very simple, reasonably fast, and requires little memory
- Uniform deviates generated do not continuously fill up the line segment from 0 to 1. They are discrete and can only assume values from the set $0, 1/p, 2/p, ..., (p-1)/p$. The smallest possible gap size is $1/p$
- One can generate at most p distinct numbers before repeating the same sequence of numbers, i.e. the maximum possible cycle length is p
- The modulus operation with p can be conducted efficiently if p is an integer power of 2 by saving only the rightmost digits. For example if $p = 2m$, then we save only the m rightmost digits

Using PRN sequences in MC Transport

- Stride
  - The first random number for history i and history $i+1$ are separated by the stride



  - Most PRNGs have formula to efficiently skip ahead
    - Useful in parallel implementations to spread histories over different processors
  - Most histories use only a fraction of the stride
  - Some may exceed the stride
    - Usually OK since numbers will be used to sample very different physical processes
- Summary
  - The linear congruential pseudo-random number generator is useful for MC sampling
  - The seed, period and stride are interesting to characterize the quality of the PRNG

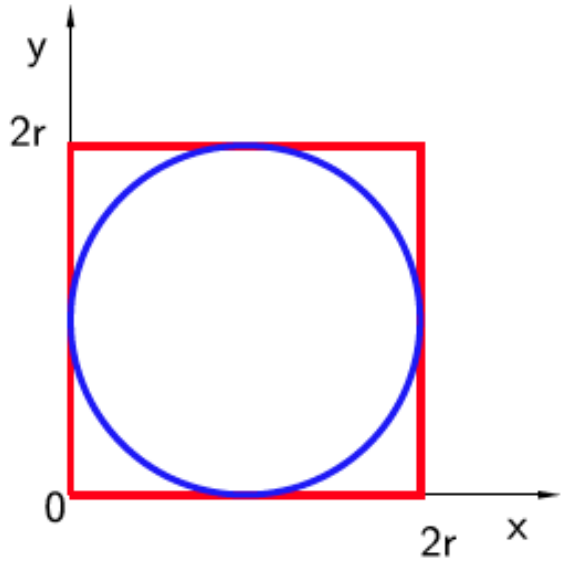
Toy Model

Consider the LCPRNG having parameters: $c = 0$, $g = 13$, and $p = 2^6 = 64$.

- Using $s_0 = 1$ as seed:
  - The following sequence of 16 pseudo-random integers is generated: 1, 13, 41, 21, 17, 29, 57, 37, 33, 45, 9, 53, 49, 61, 25, 5
  - After that the sequence repeats itself (the cycle length is 16)
  - Starting with 1, every 4th integer appears in the sequence (the gap size is $4/64 = 1/16 = 0.0625$)
- Using $s_0 = 2$:
  - The cycle length is 8
  - The gap size is $8/64 = 0.125$
- etc ...

A Simple Example of MC Simulation

Consider a circle of radius r inside a square square of side $2r$:



- Choose N random points in the Square
- Let N' be the number of points in the square that are inside the circle
- The ratio N'/N must be proportional to [Area Circle]/[Area Square]

Therefore:

$$\frac{N'}{N} \approx \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

and so

$$\pi \approx 4\frac{N'}{N}.$$

This suggests a way of computing the value of π with the help of a PRNG.

Estimating π

The algorithm for computing π using MC simulation is:

1. Initialize counters N and N' to 0.
2. Go through the following loop N times:
   (a) Generate a random variate u.
   (b) Let $x = 2u$.
   (c) Generate another random variate u.
   (d) Let $y = 2u$.
   (e) Increment N' by 1 only if $(x-1)^2 + (y-1)^2 < 1$.
   (f) Increment N by 1 and repeat.
3. Value of π is given by $4N'/N$.

Other PRNGs

- Middle-square method
  - $s_{k+1} =$ middle digits of s_k^2
- Quadratic-congruential
  - $s_{k+1} = [as_k^2 + bs_k + c] \text{mod } p$
- Modified middle-square
  - $s_{k+1} = [s_k(s_k + 1)] \text{mod } p$
- Generalized Additive
  - $s_k = [a_1 s_{k-1} + a_2 s_{k-2} + \ldots + a_i s_{k-i}] \text{mod } p$