# ACI650 - Modelos y Simulación

## Pseudorandom Number Generator (PRNG)


Mario González

*Facultad de Ingeniería y Ciencias Ambientales*

Centro de Investigación, Estudios y Desarrollo de Ingeniería (CIEDI)

UNIVERSIDAD DE LAS AMÉRICAS

Laureate International Universities®


March 21, 2016

# Learning Objectives

- Random and pseudorandom numbers
- Pseudorandom number generation
- Properties of the pseudorandom numbers
- Application of pseudorandom numbers

# The Need for random numbers

- Many statistical models rely on random numbers: Monte Carlo method.
- Stochastic simulations require a random component in the models.
- Games: random behavior a computer controlled character (scenario).
- Cryptography: data encryption.

# Random number generation

- The basic building block of stochastic simulations is the ability to generate random numbers on a physical device or in computer.
- A random number generator (RNG) is a computational or physical device designed to generate a sequence of numbers that are or appear random.
- By random, it means that they do not exhibit any discernible pattern, no matter how much effort we put into finding one.
- Generation of random numbers on a computer is complicated by the fact that computer programs are inherently deterministic.
- Even if the output of computer program may look random, it is obtained by executing the steps of some algorithm and thus is totally predictable.

# Pseudo random number generators (PRNG) I

- There are two fundamentally different classes of methods to generate random numbers.
- **True random numbers** are generated using some physical phenomenon which is random:
  - Classical examples include tossing a coin or throwing dice.
  - Modern methods utilize quantum effects, thermal noise in electric circuits, the timing of radioactive decay, etc.
- **Pseudo random numbers** are generated by computer programs.
  - While these methods are normally fast and resource effective, a challenge with this approach is that computer programs are inherently deterministic and therefore cannot produce "truly random" output.

# Pseudo random number generators (PRNG) II

## A pseudo random number generator (PRNG)

is an algorithm which outputs a sequence of numbers that can be used as a replacement for an independent and identically distributed (i.i.d.) sequence of “true random numbers”.

- There is no such thing as a single random number.
- We talk of a sequence of random numbers that follow a specific (probability) distribution, theoretical or empirical.

# Properties of pseudo random numbers

Desirable attributes of a PRNG are:

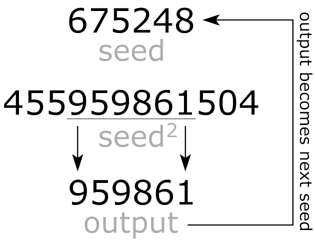
- Uniformity
- Independence
- Efficiency
- Replicability
- Long cycle length (period)

# Pi ($\pi$) as a PRNG

- **Pi** day is observed in on March 14 (3,14).
- Let's generate a sequence of random number obtained from the first 1000 Pi numbers (**algorithm**).
- We will use the date of the system as **seed**.
- A random seed (or seed state, or just seed) is a number (or vector) used to initialize a pseudorandom number generator.
- Why do we need random number generators, if $\pi$, $e$ and other irrational numbers are sources of non repeating digits?

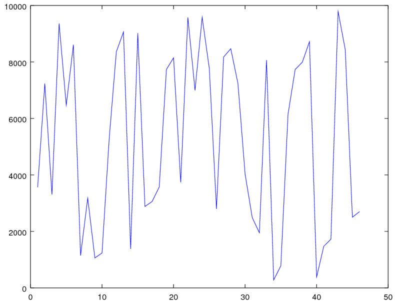
# Middle-square method I

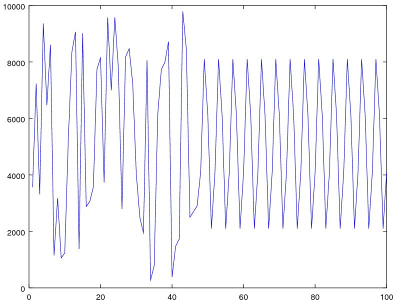
1. The middle-square method was proposed by J. von Neumann (1940s).
2. As an example, from a six digit seed, the seed is squared, and the resulting value has its middle six digits as the output value.
3. The output value in the step 2 is used as the next seed for the sequence.
4. It is not a good method, given its period is usually very short and the output sequence almost always converge to zero.



# Middle-square method II

Middle-square generator for 4-digit numbers starting from 3567. **Left:** first 46 numbers. **Right:** first 100 numbers.




See Generating uniform random numbers - examples.

# Linear congruential generators I

- The linear congruential generator (LCG) was proposed D.H. Lehmer in 1948. The form of the generator is

$$X_n = (aX_{n-1} + c) \; mod \; m$$

- LCG pseudo-code:

**input:**
$m > 1$ (the modulus)
$a \in \{1, 2, ..., m-1\}$ (the multiplier)
$c \in \{0, 1, ..., m-1\}$ (the increment)
$X_0 \in \{0, 1, ..., m-1\}$ (the seed)
**output:**
a sequence $X_1, X_2, X_3, ...$ of pseud random numbers

```
1: for n = 1, 2, 3, ... do
2:     X_n ← (aX_{n−1} + c)modm
3:     output X_n
4: end for
```

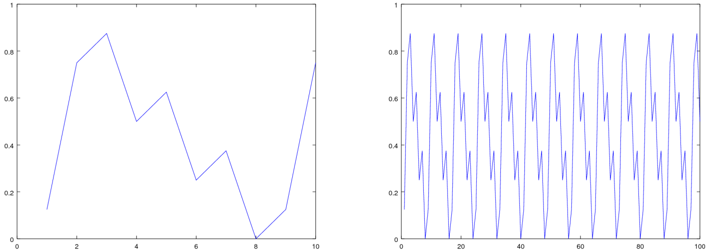
# Linear congruential generators II

- **Theorem:** The LCG has a period of $m$ if and only if
  I. $m$ and $c$ are relatively prime
  II. $a - 1$ is divisible by every prime factor of $m$
  III. if $m$ is a multiple of 4, then $a - 1$ is a multiple of 4.
- In the situation of the theorem, the period length does not depend on the seed $X_0$ and usually this parameter is left to be chosen by the user of the PRNG.
- In most cases the goal is to simulate the continuous uniform distribution $U(0, 1)$. Therefore the integers $X_n$ are rescaled to

$$U_n = X_n/m$$

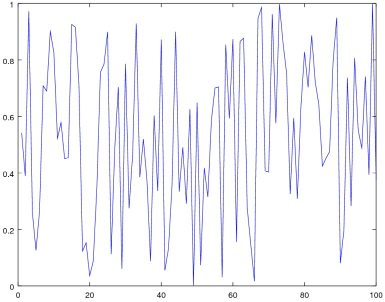
# Linear congruential generators III

Example: For parameters $m = 8, a = 5, c = 1$ and seed $X_0 = 0$.
Test it for $n = 10$ (left), and $n = 100$ (right).



# Linear congruential generators IV

Example: Let $m = 2^{32}$, $a = 1103515245$ and $c = 12345$. Since the only prime factor of $m$ is 2 and $c$ is odd, the values $m$ and $c$ are relatively prime and condition (i) of the theorem is satisfied. Similarly, condition (ii) is satisfied, since $a - 1$ is even and thus divisible by 2. Finally, since $m$ is a multiple of $4$, we have to check condition (iii) but, since $a - 1 = 1103515244 = 275878811 \cdot 4$, this condition also holds. Therefore the LCG with these parameters $m$, $a$ and $c$ has period $2^{32}$ for every seed $X_0$.



# Quality of pseudo random number generators I

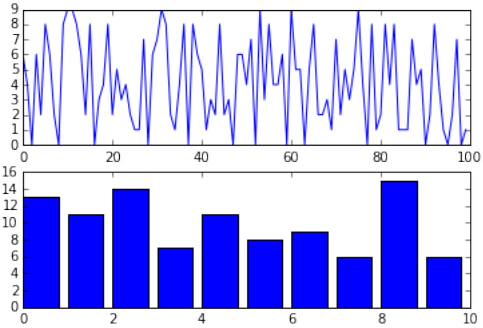
- No PRNG can produce a perfect result, but the random number generators used in practice, for example the Mersenne Twister algorithm are good enough for most purposes.
- We have seen that the output of the LCG is eventually periodic.
- The **period length** is a measure for the quality of a PRNG.
  - Most PRNGs used in practice have a period length which is much larger than the amount of random numbers a computer program could ever use in a reasonable time.
  - Periodicity of the output is not a big problem in practical applications of PRNGs.

# Quality of pseudo random number generators II

- **Distribution of samples:**
  - Frequency test: 0s, 1s, 2s, 3s, $m-1$s., have a uniform frequency.
  - Serial test: For a 2-dimensional series: 00s, 01s, 02s, etc.
  - The poker test for independence is based on the frequency in which certain digits are repeated in a series of numbers: aaaaa, aaaab, aaabb, etc.
  - Uniformity of the output can be tested using statistical tests like the chi-squared test or the Kolmogorov-Smirnov test.
- **Independence of samples:**
  - An easy way to visualize the dependence between pairs of consecutive samples is a scatter plot of the points $(X_i, Xi+1)$ for $i = 1, 2, ..., N-1$.

# Quality of PRNG: a closer look

- Let's get back to our Pi PRNG:
  - Pi_PRNG(100, 68), Pi_PRNG(sequenceLen, seed)
  - Observed frequencies for $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$ are $o_i = [13, 11, 14, 7, 11, 8, 9, 6, 15, 6]$
  - Expected frequency: $e_i = 1/10$



# Chi-squared Test I

- Designed for testing discrete distributions, large samples
- General test: can be used for testing any distribution
  - uniform random number generators
  - random variate generators
- The statistical test is

$$\sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} < \chi^2_{[1-\alpha,k-1]}$$

- Components:
  - $k$ is the number of bins in the histogram
  - $o_i$ is the number of observed values in bin $i$ in the histogram
  - $e_i$ is the number of expected values in bin $i$ in the histogram
- The test

# Chi-squared Test II

- if the sum is less than $\chi^2_{[1-\alpha,k-1]}$, then the hypothesis that the observations come from the specified distribution cannot be rejected at a level of significance $\alpha$

- For the Pi_PRNG example we have:

| Value | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Observed | 13 | 11 | 14 | 7 | 11 | 8 | 9 | 6 | 15 | 6 |
| Expected | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| $\frac{(o_i-e_i)^2}{e_i}$ | 0.9 | 0.1 | 1.6 | 0.9 | 0.1 | 0.4 | 0.1 | 1.6 | 2.5 | 1.6 |

$$\sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} < \chi^2_{[1-0{,}05,10-1]} = 9{,}8 < 16{,}919$$

- So we cannot reject the null hypothesis, and we can conclude that the data is probably uniformly distributed in $[0, 9]$.

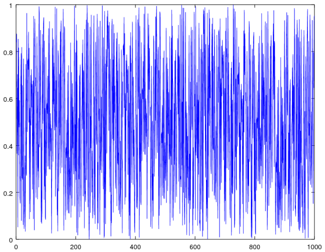
# Chi-squared Test III

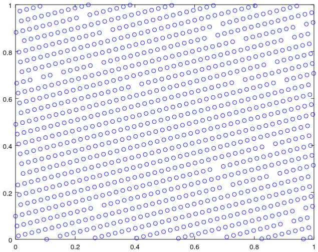
- Data must be a random sample of the population
  - to which you wish to generalize claims
- Data must be reported in raw frequencies (not percentages)
- Measured variables must be independent
- Values/categories on independent and dependent variables
  - must be mutually exclusive and exhaustive
- Observed frequencies cannot be too small
- Use Chi-square test only when observations are independent:
  - no category or response is influenced by another
- Chi-square is an approximate test of the probability of getting
  - the frequencies you've actually observed if the null hypothesis were true
  - based on the expectation that within any category, sample frequencies are distributed according the expected population

# Independence of samples I

## 2D Visual Check of Overlapping Pairs

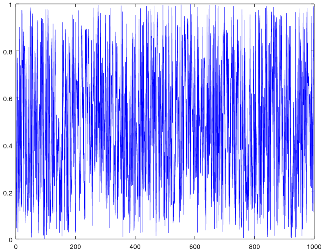
- An easy way to visualize the dependence between pairs of consecutive samples is a scatter plot of the points $(X_i, Xi+1)$ for $i = 1, 2, ..., N-1$.
- 2D Visual Check for LCG(401,101,1024,7,1000), LCG(a,c,m,seed,n)

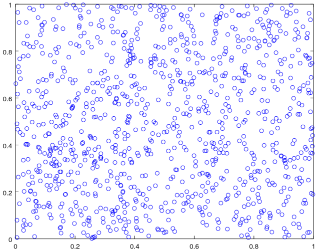


# Independence of samples II

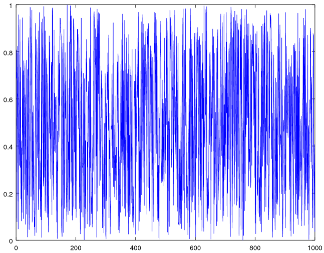
- 2D Visual Check for LCG(1103515245,12345,$2^{32}$,7,1000), LCG(a,c,m,seed,n)

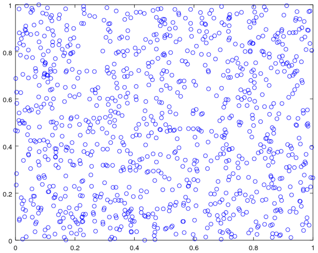


# Independence of samples III

- 2D Visual Check for Mersenne Twistter




Check this presentation for Testing Random Number Generators

# Sources and Resources

- Tests for Random Numbers.
- Voss, J. (2013). An introduction to statistical computing: a simulation-based approach. John Wiley & Sons.
- Low-discrepancy sequence, Wikipedia.
- Monte Carlo integration, Wikipedia.
- Generating uniform random numbers examples.