Random process and simple sampling Monte Carlo simulation

Arjaree Thongon

Random or Stochastic processes

- The process THAT you cannot predict from and observation of a single event, how the next will come out
- Examples:
  - Coin: the only prediction about outcome -50% the coin will land on its tail
  - Dice: In large number of throws – probability for each face appearing is 1/6

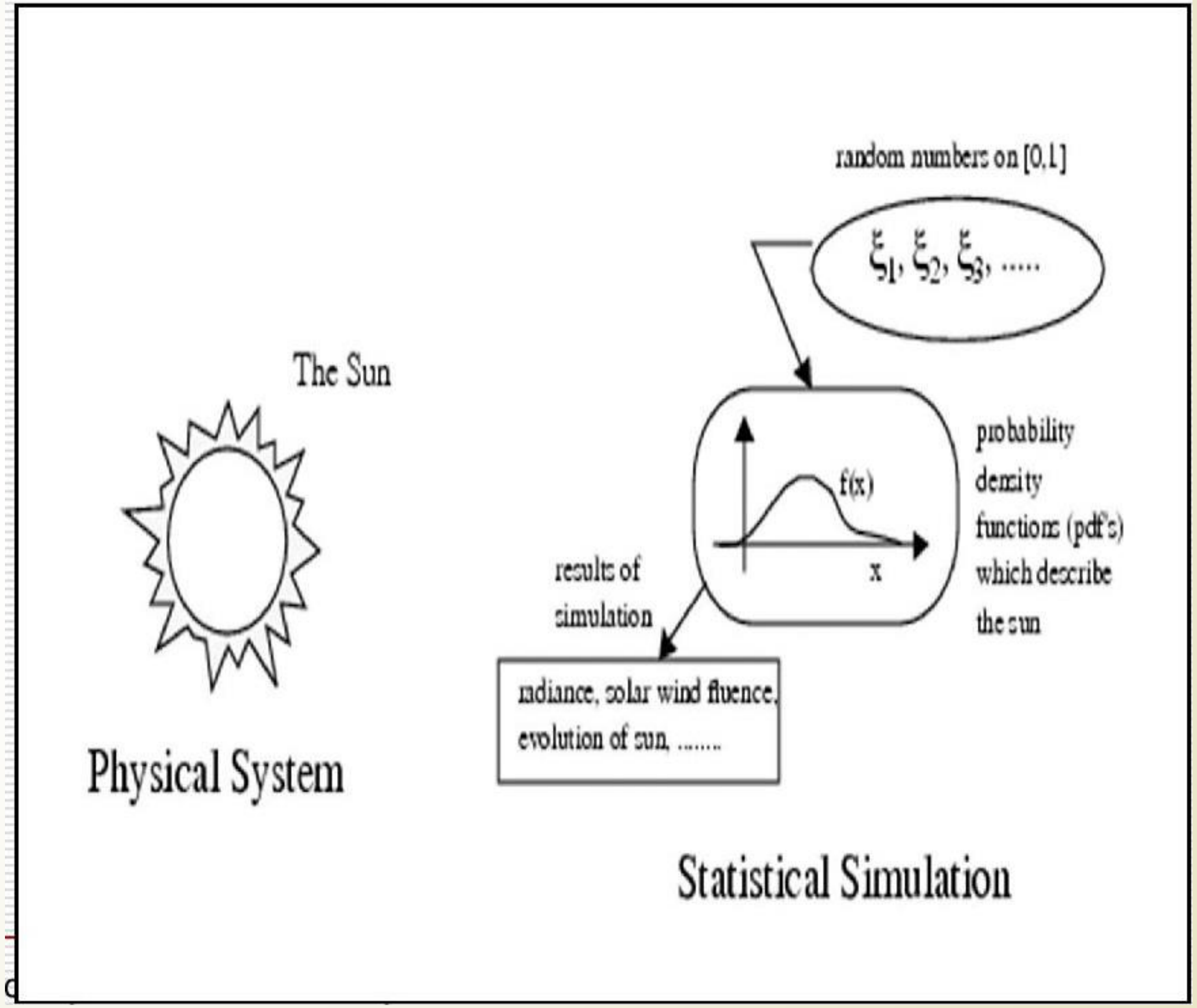
Question: What is the most probable number for the sum of two dice?

36 possibilities

6 times – for 7

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Here comes Monte Carlo simulation!



Applications for MC simulation

- Stochastic processes
- Complex systems (science)
- Numerical integration
- Risk management
- Financial planning
- …

How do we do that?

- You let the computer to throw “the coin” and record the outcome
- You need a program that randomly generates a variable
- … with relevant probability distribution

Random Number Generators (RNG)

- There are no true random number geneators but pseudo RNG!!!
- Reason: computers have only a limited number of bits to represent a number
- It means: the sequence of random numbers will repeat itself (period of the generator)

Good Random Number Generators

- Equal probability for any number inside interval [a,b]
- Yet independent of the previous number
- Long period
- Produce the same sequence if started with same initial
- Conditions
- fast

Linear Congruent Method for RNG

- Generates a random sequence of numbers $\{x_1,x_2,\ldots,x_k\}$ of length M over the interval $[0,M\text{-}1]$

$$X_i = (ax_{i\text{-}1} + c) \bmod M$$
$$= \mathrm{mod}\,(ax_{i\text{-}1} + c\,,\,M)$$

- The starting value X_0 is called "seed"
- Coefficients a and c should be chosen very carefully

NOTE: mod(b,d) = b - int(b/d)*d

Example: *a=4, c=1, M=9,* $x_0=3$

$$X_i = (ax_{i-1} + c) \bmod M$$
$$= \bmod\ (ax_{i-1} + c\ ,\ M)$$

- We get

  $x_1 = 4$

  $x_2 = 8$

  $x_3 = 6$

  x_4 - x_9 = 7, 2, 0, 1, 5, 3

Data interval: 0-8, i.e. [0,M-1]

Period: 9 (Maximum) i.e. M numbers (then repeat)

Random Numbers on interval [A,B]

- Scale results from x_i on $[0,M\text{-}1]$ to y_i on $[0,1]$

$$y_i = x_i / (M-1)$$

- Scale results from x_i on $[0,1]$ to y_i on $[A,B]$

$$y_i = A + (B - A)\, x_i$$

Magic numbers for Linear Congruent Method

- M (length of the sequence) is quite large
- Generally, the last number before overflow

  (for 32 bit machines $M = 2^{31} - 1 \approx 2*10^9$)
- Good "magic" number for linear congruent method:

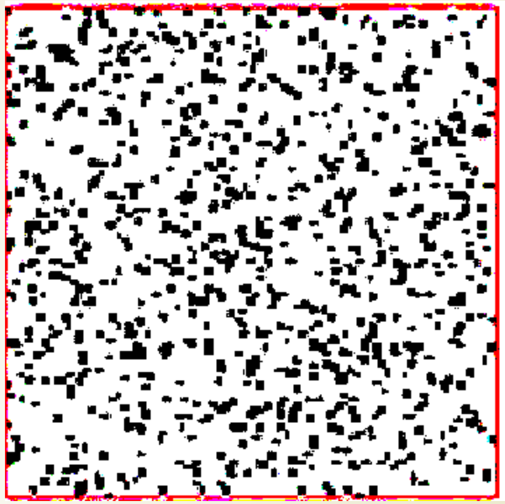
$$x_i = \mathrm{mod}\,(ax_{i-1} + c\,,\, M)$$

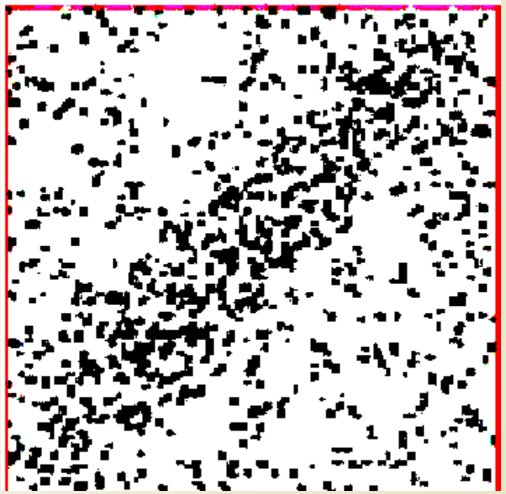
a = 16807, c = 0, M = 2^{31} -1 = 2147483647

How can we be check the RNG?

Plots:

- 2D figure, where x_i and y_i are from two random sequences (check correlation between 2 sequences)
- 3D figure (x_i, y_i, z_i)
- 2D figure for correlation (x_i, x_{i+1})





How can we check the RNG?

- Uniformity
  - Equal fractions of random numbers should fall into equal “area” in space.
- Uncorrelated sequence
  - Any subsequence of random numbers should not be correlated with any other subsequence of random numbers.
- Long period
  - The repetition should occur only after the generation of a very large set of random numbers.
- Efficiency
  - Can be implemented in a high level language and consume less than 1% of overall CPU time in any applications.

How can we check the RNG?: Example-Randu

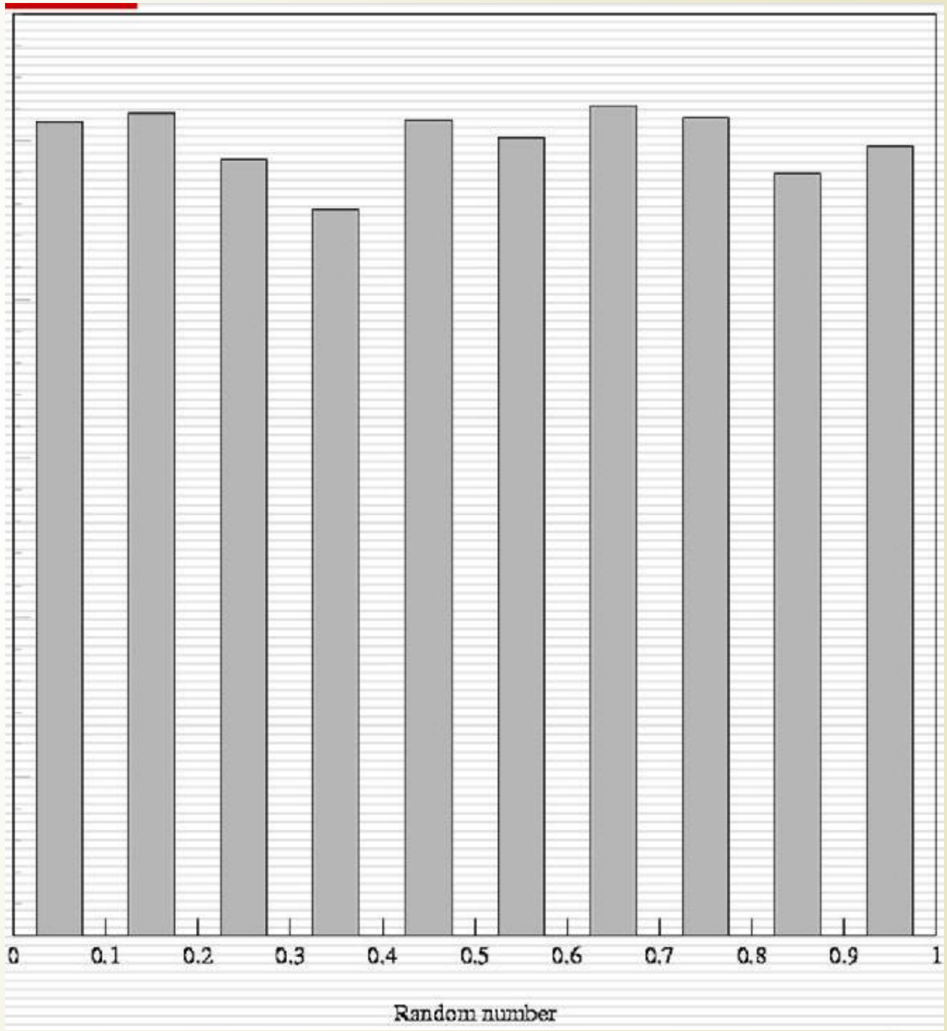
- Plot the histogram of the random number with the interval [0,1] using the formula

$$x_{n+1} = x_n/m$$

 from RANDU

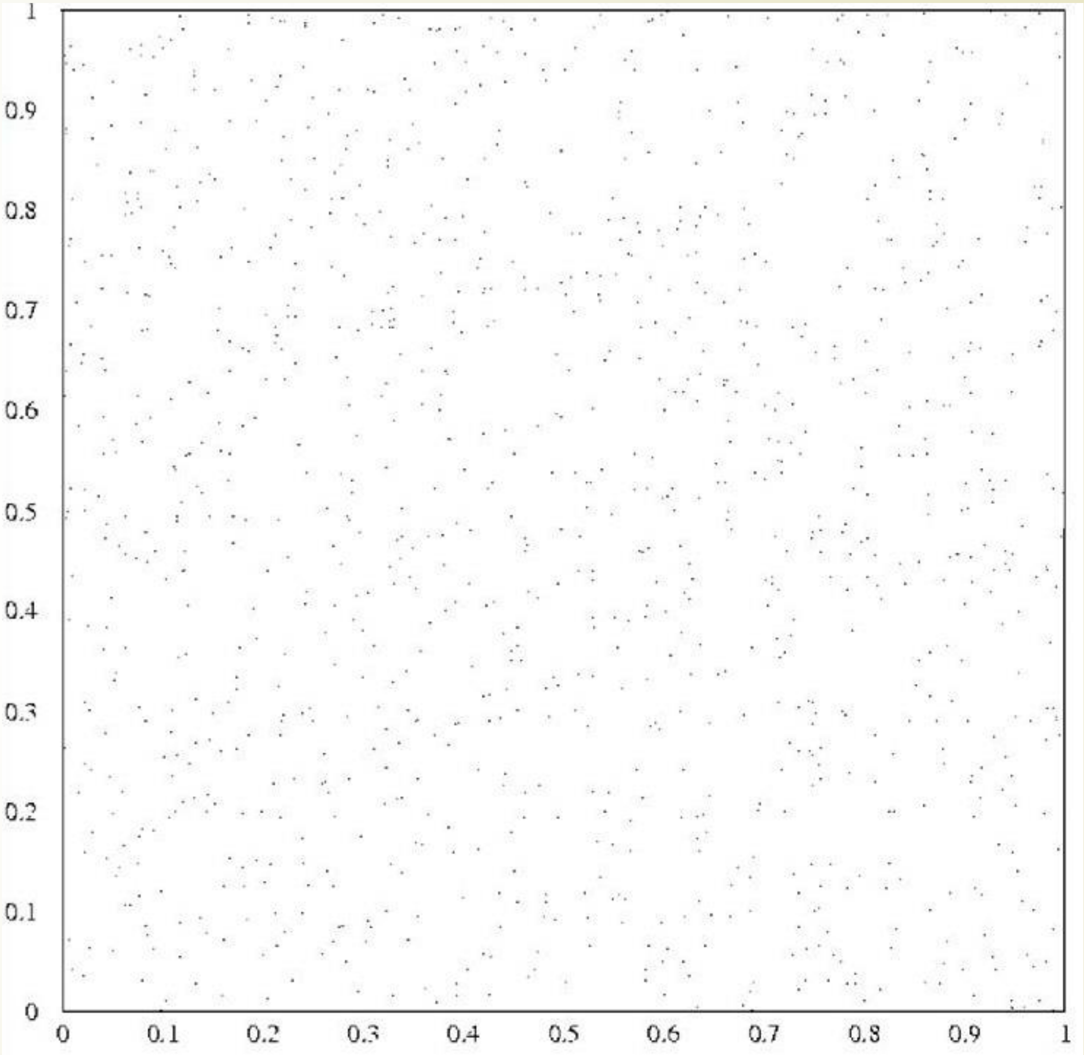
- To check if it is uniform

- **Look ok!**



How can we check the RNG?: Example-Randu

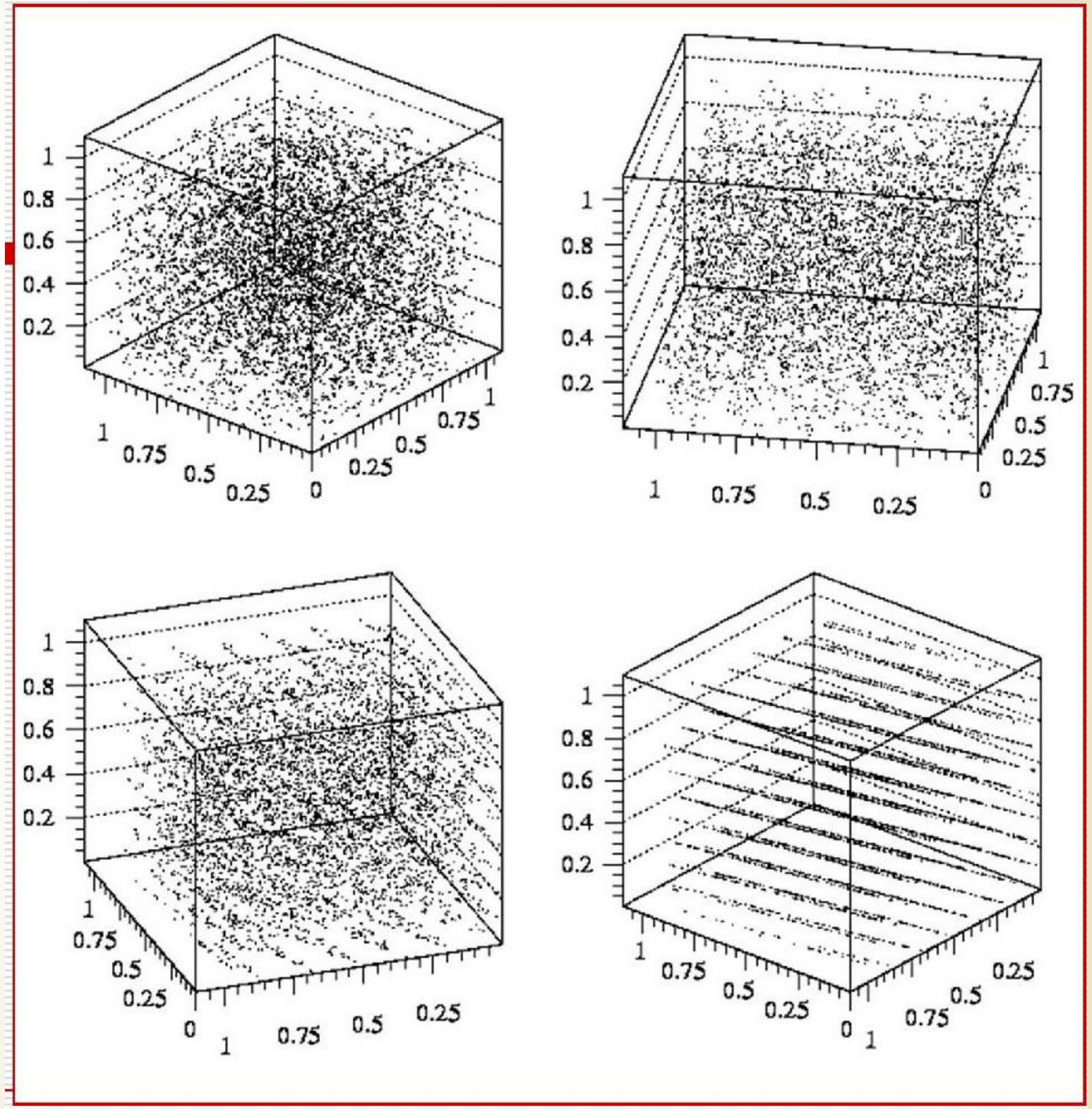
- To check the correlation
- Plot the coordinate (x_{n+1}, x_n) to form 2D distribution area
- **Still lool ok!**



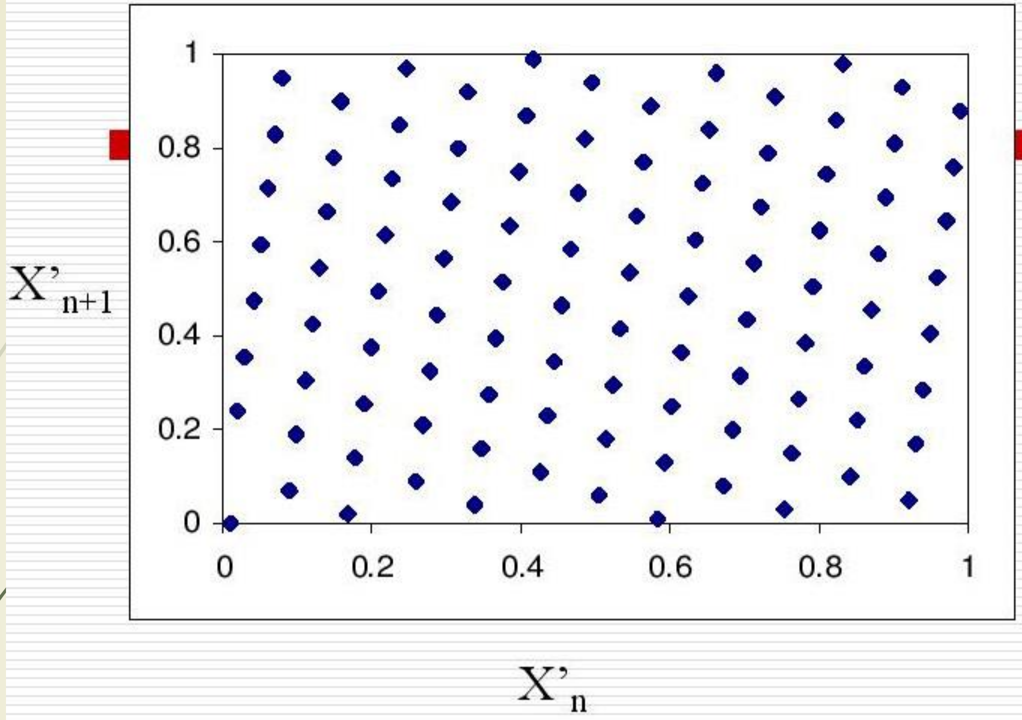
How can we check the RNG?: Example-Randu

BUT

Problem found when observe at right angles.



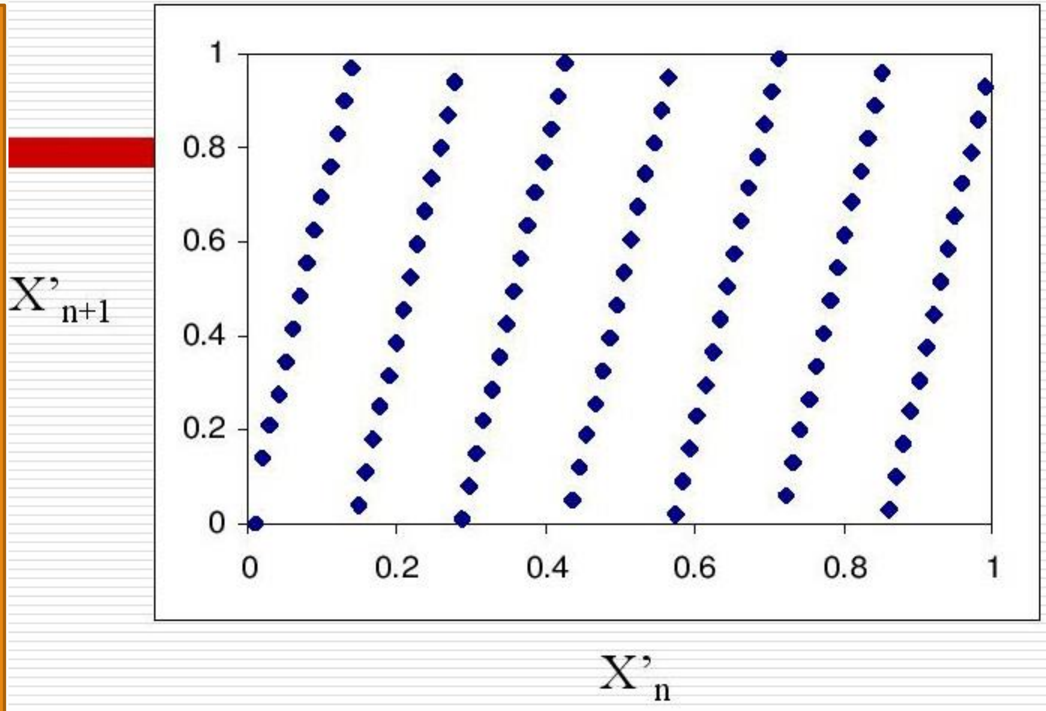
Linear Congruent Generator (LCG) Example



$$\boxed{X_{n+1} = 12X_n \bmod 101}$$

Seed $X_0 = 1$

$X'_{n+1} = X_{n+1} / 101$, $X'_n = X_n / 101$



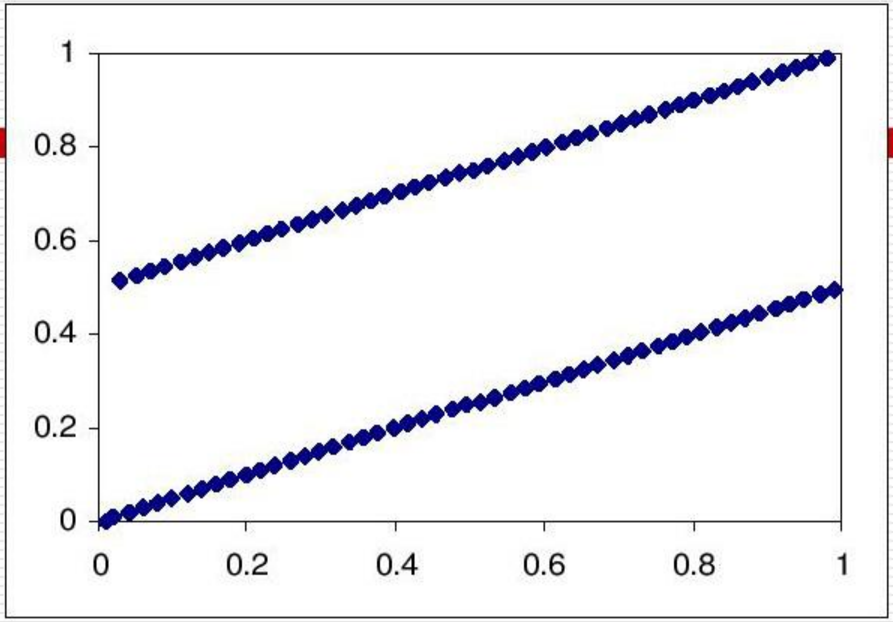
$$\boxed{X_{n+1} = 7X_n \bmod 101}$$

Seed $X_0 = 1$

$X'_{n+1} = X_{n+1} /101$, $X'_n = X_n / 101$

Linear Congruent Generator (LCG) Example

X'_{n+1}



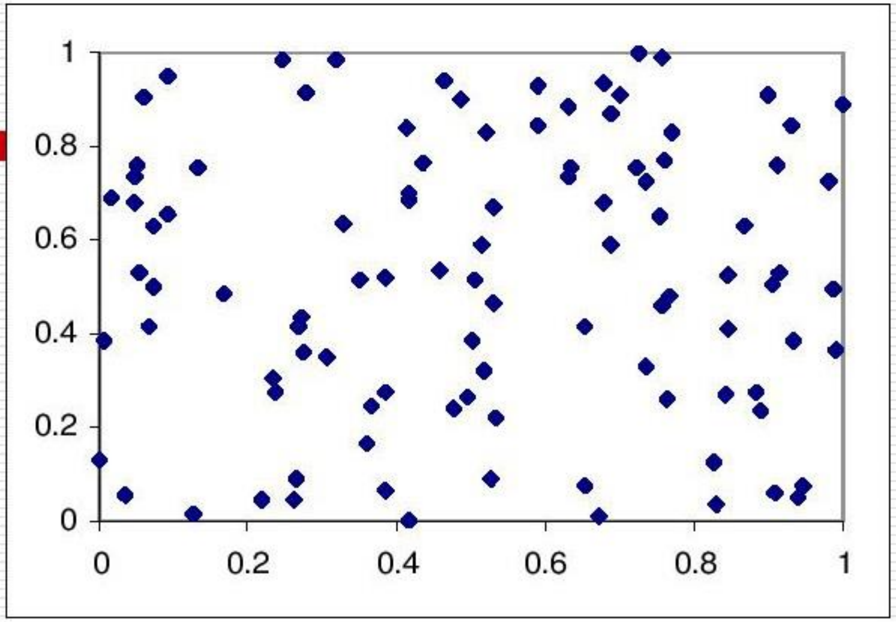
X'_n

$$\boxed{X_{n+1} = 12X_n \bmod 51}$$

Seed $X_0 = 1$

$X'_{n+1} = X_{n+1} / 51,\ X'_n = X_n / 51$

X'_{n+1}



X'_n

$$\boxed{X_{n+1} = \mathbf{7^5}\ Xn \bmod \mathbf{2^{31}\text{-}1}}$$

Seed $X_0 = 1$

$X'_{n+1} = X_{n+1} /101,\ X'_n = X_n / 101$

Long period random number generator

$$X_i = (ax_{i-1} + c) \bmod m$$

- The linear congruential sequence define by a, m, c and x_0 has period m if only if
  - c is relatively prime to m
  - $b = a - 1$ is a multiple of p,

    for every prime p dividing by m
  - b is a multiple of 4, if m is a multiple of 4

Some commonly used parameters for LCG

a	m	c	period
7^5	$2^{31} - 1$	0	$2^{31} - 2$
1664525	2^{32}	1013904223	2^{32}
69069	2^{30}	0	2^{30}
6364136223846793005	2^{64}	1	2^{64}

Seed matter : X_0

- Funny way to choose
  - Randomly taking from telephone directory.
  - Exposing a Geiger counter to radioactive source for a minute (SHIELD YOUSELF) and use the resulting count.
  - Asking a friend.
  - Asking an enemy.
  - Taking from lotto or horse racing.

Seed matter : X_0

- Better way to choose
  - From computer clock (current system time)
  - Example, in each day, most computers would have
    - Hour: $0 \le c \le 23$
    - Minute: $0 \le t_m \le 59$
    - Second: $0 \le t_s \le 59$
    - One-hundredth second: $0 \le t_{hs} \le 99$
  - We may assign a seed
  - $x_0 = 100[60(60t_h + t_m) + t_s] + t_{hs}$

Example: choosing X_0

Structure of time in Turbo C

```
struct time{
    unsigned char ti_min; /*minutes*/
    unsigned char ti_hour; /*hours*/
    unsigned char ti_hund; /*hundredths of seconds*/
    unsigned char ti_sec; /* seconds */
};
```

Example: choosing X_0

Calling time function

```
#include <stdio.h>
#include <dos.h>
int main (void)
{
    struct time t;
    gettime(&t);
    printf(“The current time is: %2d:%02d:%02d.%02d\n”, t.ti_hour, t.ti_min, t.ti_sec, t.ti_hund);
    return0;
}
```