Random processes and Monte Carlo Simulation

- Applications of Monte Carlo simulations.
- Random number generators.

Random or Stochastic Processes

Random or Stochastic processes are those in which you cannot predict the outcome of the upcoming event from the outcome of the current event. For example:

Random or Stochastic Processes

Random or Stochastic processes are those in which you cannot predict the outcome of the upcoming event from the outcome of the current event. For example:

- Coin toss: the only prediction about the outcome: 50% of the events will end up as tail being up.

Random or Stochastic Processes

Random or Stochastic processes are those in which you cannot predict the outcome of the upcoming event from the outcome of the current event. For example:

- Coin toss: the only prediction about the outcome: 50% of the events will end up as tail being up.
- Dice: In a large number of throws, the probability of getting a given face is $\frac{1}{6}$.

Applications of Monte Carlo simulations

- Stochastic processes.

Applications of Monte Carlo simulations

- Stochastic processes.
- Complex systems (science).

Applications of Monte Carlo simulations

- Stochastic processes.
- Complex systems (science).
- Numerical integration.

Applications of Monte Carlo simulations

- Stochastic processes.
- Complex systems (science).
- Numerical integration.
- Risk management.

Applications of Monte Carlo simulations

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

Applications of Monte Carlo simulations

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

Applications of Monte Carlo simulations

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

How does one do Monte Carlo simulations

- Let the computer throw "the coin" and record the outcome.

How does one do Monte Carlo simulations

- Let the computer throw "the coin" and record the outcome.
- Need a program that generates a variable with random value.

How does one do Monte Carlo simulations

- Let the computer throw "the coin" and record the outcome.
- Need a program that generates a variable with random value.
- Often need a program that generates a random variable with a given probability distribution.

Random numbers

Sources of random numbers:

Random numbers

Sources of random numbers:

- Tables – "A million random digits with 100,000 normal deviates" by RAND.

Random numbers

Sources of random numbers:

- Tables – "A million random digits with 100,000 normal deviates" by RAND.
- Hardware – external sources of random numbers which are generated by from a physical process.

Random numbers

Sources of random numbers:

- Tables – "A million random digits with 100,000 normal deviates" by RAND.
- Hardware – external sources of random numbers which are generated by from a physical process.
- Software – source of pseudo random numbers.

Software – random number generators

- There are NO true random number generators – only pseudo random number generators!!!

Software – random number generators

- There are NO true random number generators – only pseudo random number generators!!!
- This is because computers have only a limited number of bits to represent a number.

Software – random number generators

- There are NO true random number generators – only pseudo random number generators!!!
- This is because computers have only a limited number of bits to represent a number.
- It implies that no matter which pseudo random number generator you use – it will always repeat itself (period of the generator).

Good random number generators

Important issues:

- Randomness.

Good random number generators

Important issues:

- Randomness.
- Distribution of the numbers.

Good random number generators

Important issues:

- Randomness.
- Distribution of the numbers.
- Long period.

Good random number generators

Important issues:

- Randomness.
- Distribution of the numbers.
- Long period.
- Produce the same sequence if started with the same seed.

Good random number generators

Important issues:

- Randomness.
- Distribution of the numbers.
- Long period.
- Produce the same sequence if started with the same seed.
- Fast.

Two basic techniques for RNG

The standard method of generating pseudorandom numbers use modular reduction in congruential relationships:

- Congruential methods.

Two basic techniques for RNG

The standard method of generating pseudorandom numbers use modular reduction in congruential relationships:

- Congruential methods.
- Feedback shift register methods

Linear congruent method for RNG

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

$$x_i = \mod (ax_{i-1} + c, M) = \text{remainder}\left(\frac{ax_{i-1} + c}{M}\right)$$

Linear congruent method for RNG

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

$$x_i = \quad \mod (ax_{i-1} + c, M) = \text{remainder}\left(\frac{ax_{i-1} + c}{M}\right)$$

- Starting value of x_0 is called "seed"

Linear congruent method for RNG

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

$$x_i = \quad \mod (ax_{i-1} + c, M) = \text{remainder}\left(\frac{ax_{i-1} + c}{M}\right)$$

- Starting value of x_0 is called "seed"
- Coefficients a and c should be chosen very carefully.

Linear congruent method for RNG

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

$$x_i = \quad \mod (ax_{i-1} + c, M) = \text{remainder}\left(\frac{ax_{i-1} + c}{M}\right)$$

- Starting value of x_0 is called "seed"
- Coefficients a and c should be chosen very carefully.

Note that

$$\mod (b, M) = b - \text{int}(b/M) * M$$

Example of linear congruent method

$$x_i = \mod (ax_{i-1} + c, M)$$
$$\mod (b, M) = b - \text{int}(b/M) * M$$

Example of linear congruent method

$$x_i = \mod (ax_{i-1} + c, M)$$
$$\mod (b, M) = b - \text{int}(b/M) * M$$

$a = 4, c = 1, M = 9, x_1 = 3$ leads to the following sequence:

$$x_2 = 4$$
$$x_3 = 8$$
$$x_4 = 6$$
$$x_{5-10} = 7, 2, 0, 1, 5, 3$$

Example of linear congruent method

$$x_i = \mod (ax_{i-1} + c, M)$$
$$\mod (b, M) = b - \text{int}(b/M) * M$$

$a = 4, c = 1, M = 9, x_1 = 3$ leads to the following sequence:

$$x_2 = 4$$
$$x_3 = 8$$
$$x_4 = 6$$
$$x_{5-10} = 7, 2, 0, 1, 5, 3$$

interval:$0 - 8$ i.e. $[0, M - 1]$
Period: 9 i.e. M numbers (then repeat).

Magic numbers for linear congruent method

- M (length of the sequence) is quite large.

Magic numbers for linear congruent method

- M (length of the sequence) is quite large.
- No overflow. (For 32 bit machines $M \leq 2^{32}$).

Magic numbers for linear congruent method

- M (length of the sequence) is quite large.
- No overflow. (For 32 bit machines $M \leq 2^{32}$).
- Good magic numbers for linear congruent method:

$$a = 16,807 \quad c = 0 \quad M = 2,147,483,647$$

$$a = 1,664,525 \quad c = 1,013,904,223 \quad M = 2,147,483,648$$

Magic numbers for linear congruent method

- M (length of the sequence) is quite large.
- No overflow. (For 32 bit machines $M \leq 2^{32}$).
- Good magic numbers for linear congruent method:

$$a = 16,807 \quad c = 0 \quad M = 2,147,483,647$$

$$a = 1,664,525 \quad c = 1,013,904,223 \quad M = 2,147,483,648$$

- For $c = 0$ called "Multiplicative congruential generator".

Random numbers on interval $[A, B]$

- Scale results from x_i on $[0, M-1]$ to y_i on $[0, 1]$.

$$y_i = \frac{x_i}{M-1}$$

Random numbers on interval $[A, B]$

- Scale results from x_i on $[0, M-1]$ to y_i on $[0, 1]$.

$$y_i = \frac{x_i}{M-1}$$

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

$$z_i = A + (B - A) * y_i$$

Feedback shift register generator

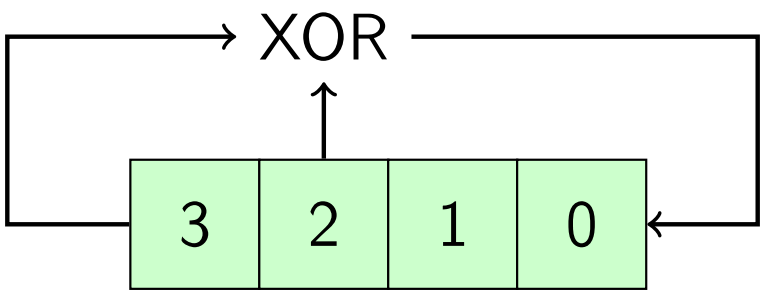
Simple shift register where the vacated bit is filled with the exclusive-or of two other bits in the shift register.

Feedback shift register generator

Simple shift register where the vacated bit is filled with the exclusive-or of two other bits in the shift register.
4 bit shift-register pseudorandom number generator:

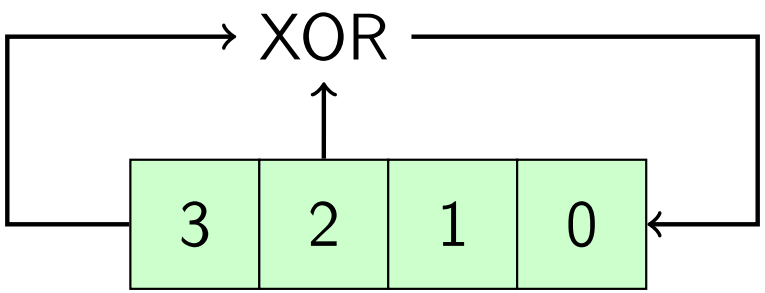
Feedback shift register generator

Simple shift register where the vacated bit is filled with the exclusive-or of two other bits in the shift register.
4 bit shift-register pseudorandom number generator:



Feedback shift register generator

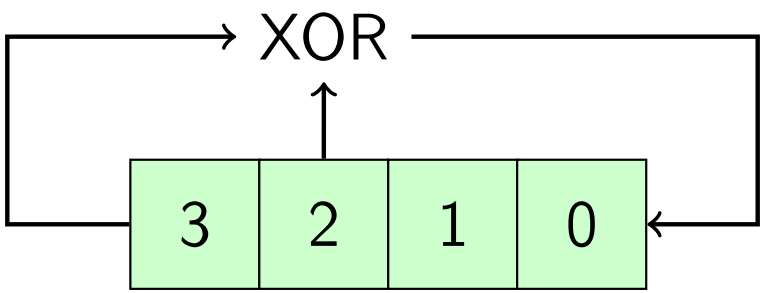
Simple shift register where the vacated bit is filled with the exclusive-or of two other bits in the shift register.
4 bit shift-register pseudorandom number generator:



- Bits 3 and 2 are combined by exclusive-or.

Feedback shift register generator

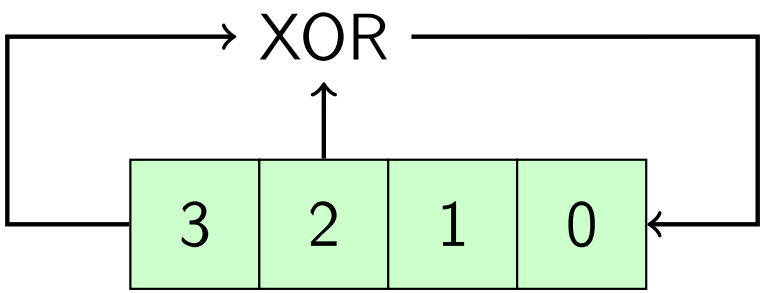
Simple shift register where the vacated bit is filled with the exclusive-or of two other bits in the shift register.
4 bit shift-register pseudorandom number generator:



- Bits 3 and 2 are combined by exclusive-or.
- The register is shifted 1 step to the left.

Feedback shift register generator

Simple shift register where the vacated bit is filled with the exclusive-or of two other bits in the shift register.
4 bit shift-register pseudorandom number generator:



- Bits 3 and 2 are combined by exclusive-or.
- The register is shifted 1 step to the left.
- The result of the exclusive-or is entered into bit 0.

4bit shift register PRNG

Here is the pattern of bits, starting with 0001:

0001
0010
0100
1001
0011
0110
1101
1010
0101
1011
0111
1111
1110
1100
1000
0001

Commonly used PRNG

- Most commonly used is Mersenne Twister (which is a generalized feedback shift register method).

Commonly used PRNG

- Most commonly used is Mersenne Twister (which is a generalized feedback shift register method).
- The commonly used version of Mersenne Twister, MT19937.

Commonly used PRNG

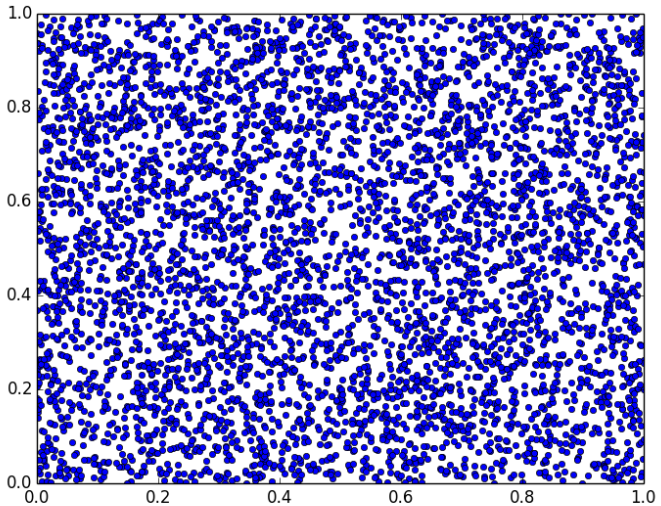
- Most commonly used is Mersenne Twister (which is a generalized feedback shift register method).
- The commonly used version of Mersenne Twister, MT19937.
- It has a period of $2^{19937} - 1$.

Commonly used PRNG

- Most commonly used is Mersenne Twister (which is a generalized feedback shift register method).
- The commonly used version of Mersenne Twister, MT19937.
- It has a period of $2^{19937} - 1$.
- Implemented in numpy.

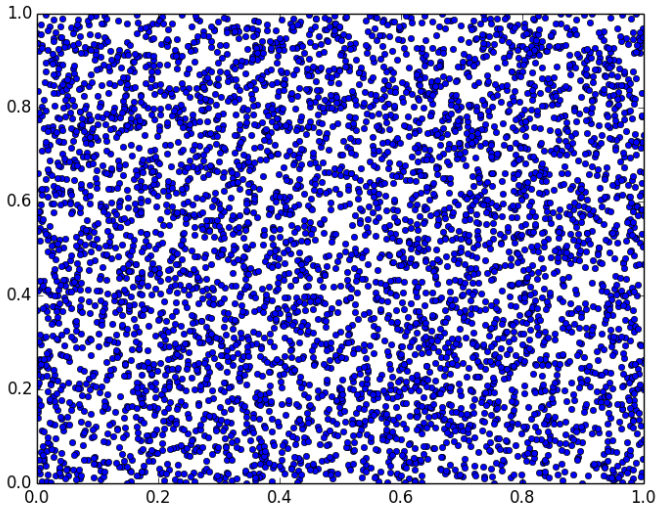
How do we check the RNG?

- 2D plot, x_i and y_i from two random sequences (parking lot test).



How do we check the RNG?

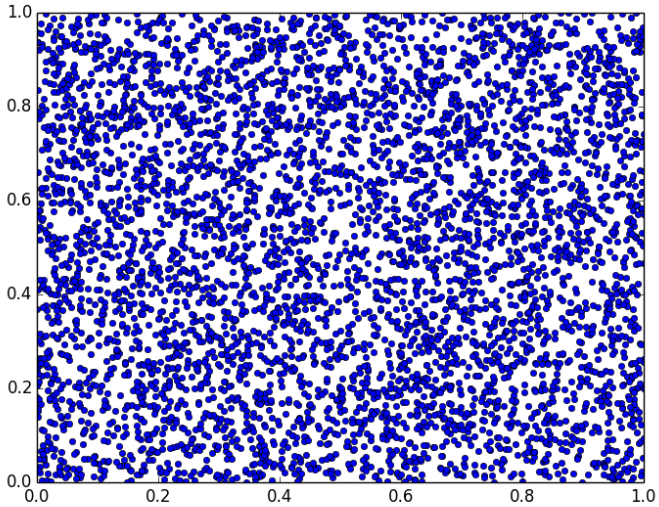
- 2D plot, x_i and y_i from two random sequences (parking lot test).



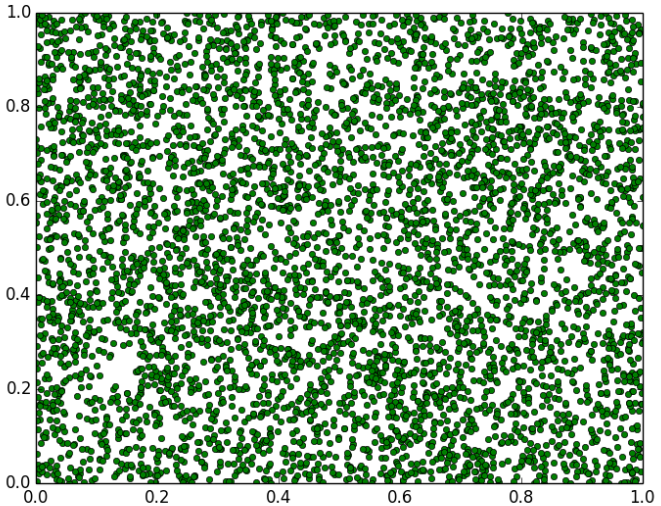
- Plot 3D figure (x_i, y_i, z_i)

How do we check the RNG?

- 2D plot, x_i and y_i from two random sequences (parking lot test).



- Plot 3D figure (x_i, y_i, z_i)
- Plot correlation (x_i, x_{i+k})



How can we check the RNG?

Examples of other assessments:

- Uniformity – A random sequence should contain numbers distributed in the unit interval with equal probability.

How can we check the RNG?

Examples of other assessments:

- Uniformity – A random sequence should contain numbers distributed in the unit interval with equal probability.
- k-th moment:

$$< x^k >= \frac{1}{N} \sum_{i=1}^{N} x_i^k = \frac{1}{k+1}$$

How can we check the RNG?

Examples of other assessments:

- Uniformity – A random sequence should contain numbers distributed in the unit interval with equal probability.
- k-th moment:

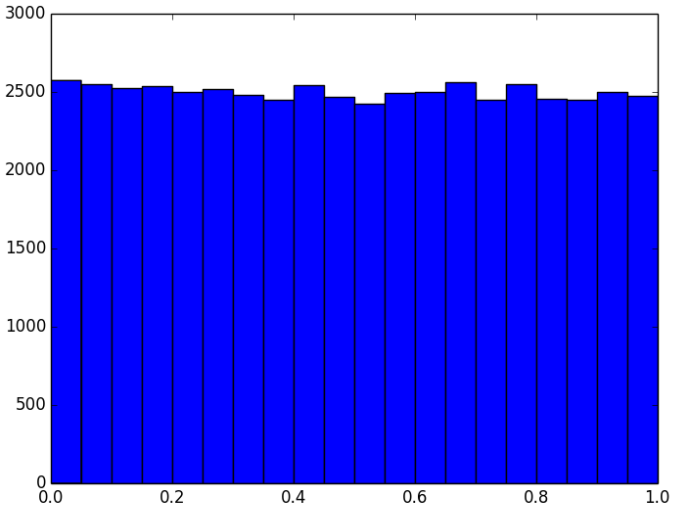
$$< x^k >= \frac{1}{N} \sum_{i=1}^{N} x_i^k = \frac{1}{k+1}$$

- Near neighbour correlation:

$$\frac{1}{N} \sum_{i=1}^{N} x_i x_{i+k} \approx \frac{1}{4}$$

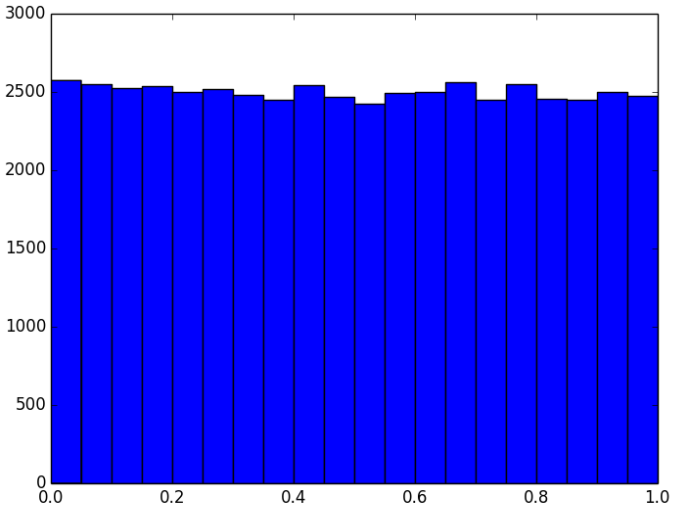
Examples of other assessments

- Uniformity (50000 random numbers)



Examples of other assessments

- Uniformity (50000 random numbers)

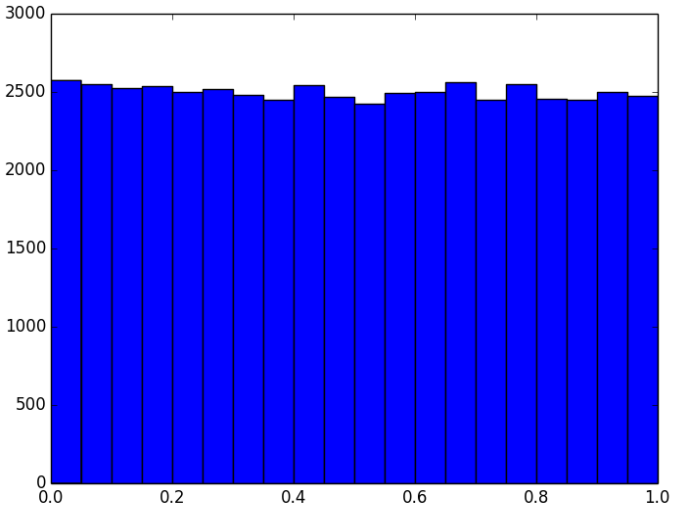


- 4th moment: (50000 random numbers)

$$< x^4 >= 0.1988$$

Examples of other assessments

- Uniformity (50000 random numbers)



- 4th moment: (50000 random numbers)

$$< x^4 >= 0.1988$$

- near neighbor correlation: (50000 random numbers) $= 0.2478$

Test suites for RNG

Good test suites exist – TestU01 – which can be used to uncover problems in random number generators.

Dont try to invent your own random number generator – unless you know what you are doing. This is very tricky business!!!