RANDOM NUMBER GENERATION

Quantitative Risk Management project work

Silvia Baldisserotto, Maysa Jahanbani,
Claudia Pesci, Phan Ho Tan Phat,
Andrea Venuta

Università degli Studi di Firenze - Finance and Risk Management

RANDOM NUMBERS

- Computer-generated numbers are pseudo-random: deterministic and predictable
- Quasi-random numbers prevent potential lack of equidistributedness
- **Definition.** (sample) a sequence of number is called a sample from the distribution F if the numbers are independent realizations of a random variable with distribution function F
- Uniform deviates: samples from $\sim \mathcal{U}[0, 1]$
- Normal deviates: samples from $\sim \mathcal{N}(0, 1)$
- Drawing uniform deviates is the basis of random number generation

- N_0 is chosen arbitrarily
- $N_i = (aN_{i-1} + b) \bmod M$ for $i > 0$

Requirements:

1. Large period: small set of numbers makes the outcome easier to predict (choose M as large as possible)
2. Statistical tests to verify that the distribution is the intended one
   - Comparison of sample mean and variance $\mu,\ \sigma^2$ with desired values
   - Correlation between sample values
   - Quality of approximation of the distribution
3. Distribution in higher dimensional spaces: lattice structure

- Sequences of random numbers can be arranged in m-dimensional vectors
- The vectors lie on a number of parallel $(m-1)$-dimensional hyperplanes
- The ideal condition is that the number of parallel hyperplanes is maximized: number of hyperplanes is a measure of equidistributedness
- Family of parallel lines in the (U_{i-1}, U_i)-plane

$$z_0 U_{i-1} + z_1 U_i = c + \frac{z_1 b}{M} \quad \text{where} \quad c := N_{i-1} \frac{z_0 + a z_1}{M} - z_1 k$$

for each tuple (z_0, z_1) and for all cs.

immagini qui

- Inversion and transformation methods generate numbers distributed according to an arbitrary distribution from uniformly distributed samples

```
function [ rn ] = LCG( x )

  if(nargin == 0)
    x = 1;
  end

  rn = zeros(x,1);

  for i = 1:x
    rn(i) = LCGstep();
  end

end
```

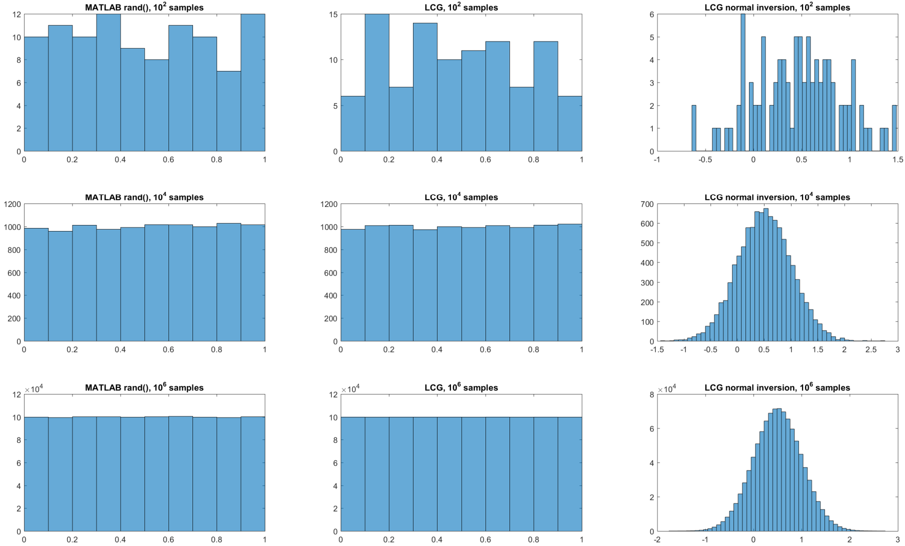
```
function [ rnStep ] = LCGstep()

  persistent seed;
  M = 244944;
  a = 1597;
  b = 51749;

  if(isempty(seed))
    seed = 0;
  end

  seed = mod(seed * a + b, M);

  rnStep = seed / M;

end
```

IMPLEMENTATIONS - BOX-MULLER METHOD

```
function [ Z ] = BoxMuller( x )

  if(nargin == 0)
    x = 1;
  end

  U = rand(x, 2);

  theta = 2 .* pi .* U(:, 2);
  rho   = sqrt( -2 .* log( U(:, 1) ) );

  Z = [ rho .* cos(theta), rho .* sin(theta) ];

end
```

```
function [ Z ] = Marsaglia( x )

  if(nargin == 0)
    x = 1;
  end

  Z = zeros(x,2);

  for i = 1 : x
    W = 1;  V = [ 1, 1 ];
    while not (W < 1)
      V = 2 * rand(1, 2) - 1;
      W = V(1) .^ 2 + V(2) .^ 2;
    end

    Z(i, :) = V .* sqrt(-2 * log(W) / W);
  end
end
```

PLOTS - UNIVARIATE METHODS



PLOTS - BIVARIATE METHODS