Introduction to Monte Carlo Method

Andrzej Palczewski

and

Jan Palczewski

Brief history

In 1947, Stanislaw Ulam suggested to John von Neumann that the newly developed ENIAC computer would give them the means to carry out calculations based on statistical sampling.

Their coworker Nicholas Metropolis dubbed the numerical technique ***the Monte Carlo method*** partly inspired by Ulam's anecdotes of his gambling uncle who "just had to go to Monte Carlo".

N. Metropolis, S. Ulam – The Monte Carlo method, *Journal of American Statistical Association*, 44 (1949).

The idea to use Monte Carlo in finance comes from Phelim Boyle, who used it in 1977 to option pricing.

P. Boyle – Options: a Monte Carlo approach, *Journal of Financial Economics*, 4 (1977).

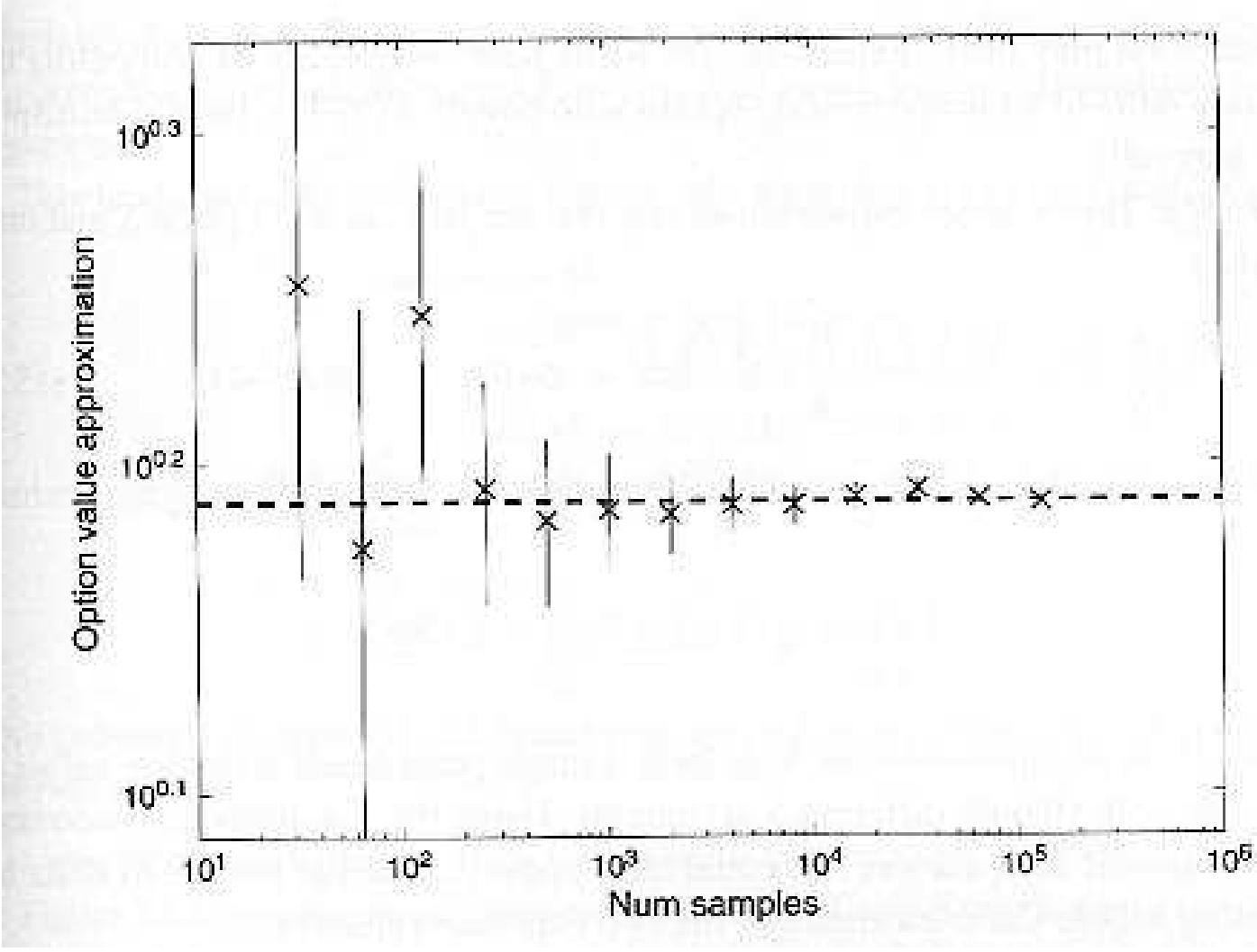
Monte Carlo put into action

Consider a European-style option with the payoff $h(S_T)$ at the moment T. Assume that under a risk-neutral measure the stock price S_t at $t \geq 0$ is given by

$$S_T = S_0 \exp\left((r - \frac{1}{2}\sigma^2)T + \sigma W_T \right).$$

for $i = 1$ to M
    compute an $\mathcal{N}(0,1)$ sample ξ
    set $S = S_0 \exp\left((r - \frac{1}{2}\sigma^2)T + \sigma\sqrt{T}\xi \right)$
    set $V_i = e^{-rT} h(S)$
end
set $a_M = \frac{1}{M} \sum_{i=1}^{M} V_i$

Confidence intervals v. number of samples



European put option

$$S_0 = 4,\ K = 5,\ \sigma = 0.3,\ r = 0.04,\ T = 1,\ M = 10\,000.$$

Plain Monte Carlo

[1]	"Mean"	"1.02421105149"	
[1]	"Variance"	"0.700745604547"	
[1]	"Standard deviation"	"0.837105491887"	
[1]	"Confidence interval"	"1.00780378385"	"1.04061831913"

Antithetic Variates

[1]	"Mean"	"1.01995595996"	
[1]	"Variance"	"0.019493094166"	
[1]	"Standard deviation"	"0.139617671396"	
[1]	"Confidence interval"	"1.01721945360"	"1.02269246632"

Arithmetic average Asian options

$$M = 10\,000$$

Control variate:

```
Standard deviation of the option price equals 0.255346
The 99% confidence interval is [4.04609, 4.05048]

Time: 5.277 sec
```

Plain Monte Carlo:

```
Standard deviation of the option price equals 6.0897
The 99% confidence interval is [3.95807, 4.05729]

Time:  5.226 sec
```

To obtain the same precision as in the control variate case requires 23^2 times more runs and the run time grows to **2765 seconds** ($\approx$ 45 minutes).

Pseudo-random number generation

The standard approach is:

- simulate a sample from a uniform distribution on $[0, 1]$,
- transform it into a desired distribution (1D);
- use more sophisticated transformations to obtain multidimensional distributions.

Congruential generator

Define a sequence (s_i) recurrently

$$s_{i+1} = (as_i + b) \mod M.$$

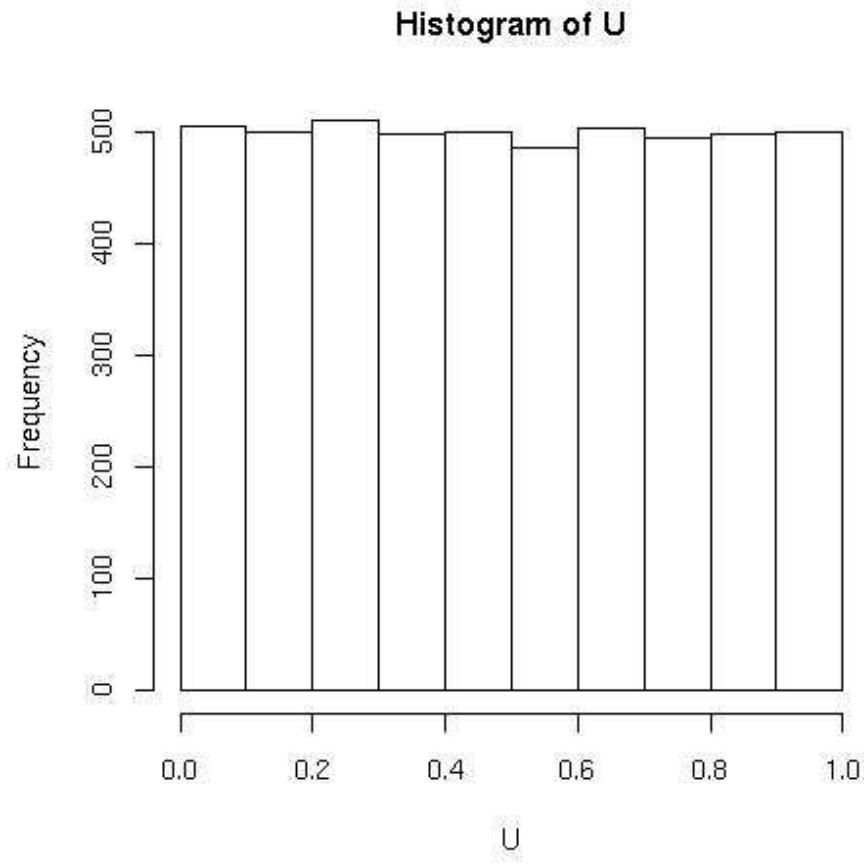
Then

$$U_i = \frac{s_i}{M}$$

form an infinite sample from a uniform distribution on $[0, 1]$.

The numbers s_i have the following properties:

1. $s_i \in \{0, 1, 2, \ldots, M-1\}$
2. s_i are periodic with period $< M$. Indeed, since at least two values in $\{s_0, s_1, \ldots, s_M\}$ must be identical, therefore $s_i = s_{i+p}$ for some $p \leq M$.



5000 uniform variates

$a = 1229$, $b = 1$, $M = 2048$, $s_0 = 1$

Are the numbers U_i obtained with the congruential linear generator uniformly distributed? It seems so, as the histogram is quite flat.

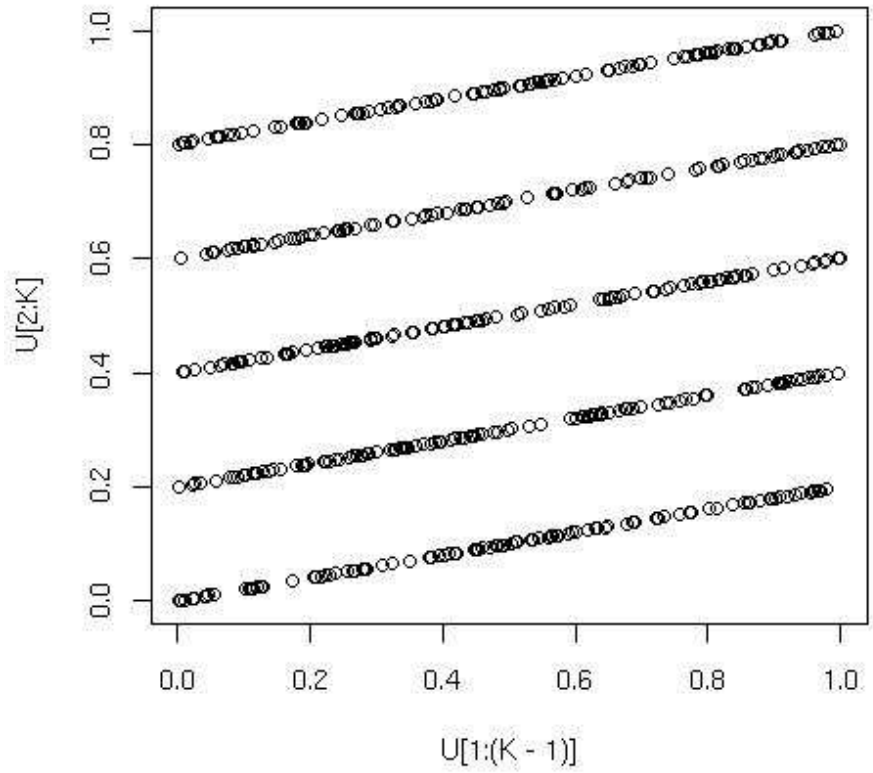
Does this really means, that the numbers U_i are good uniform deviates?

There is a question whether they are **independent**. To study this property we consider vectors built of U_i's:

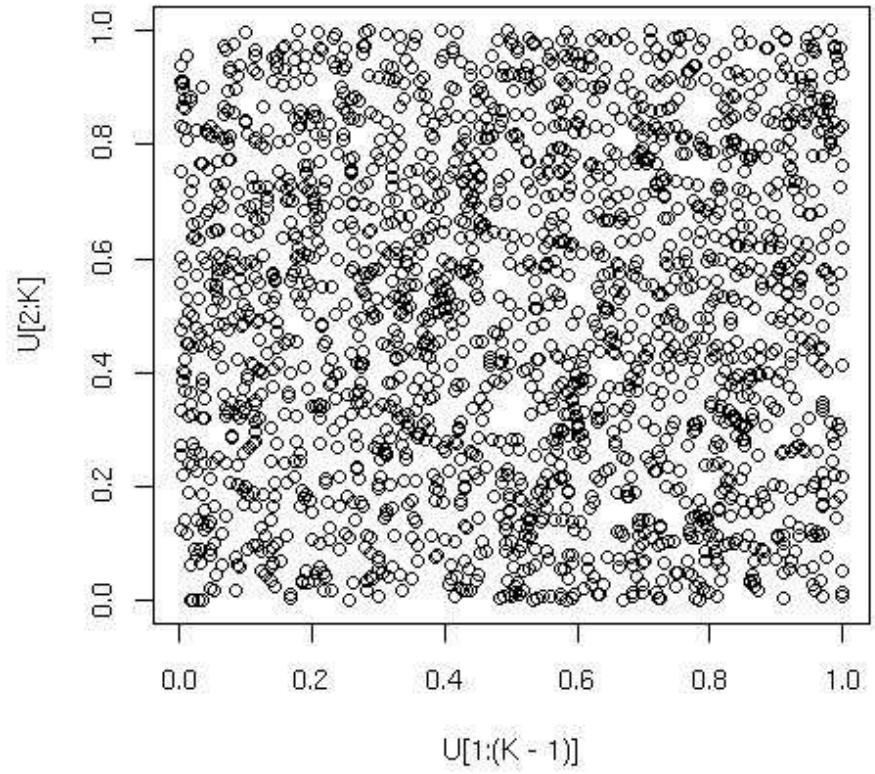
$$(U_i, U_{i+1}, \ldots, U_{i+m-1}) \in [0,1]^m$$

and analyze them with respect to distribution.
If U_i's are good uniform deviates, above random vectors are uniformly distributed over $[0,1]^m$.
It should be stressed that it is very difficult to assess random number generators!

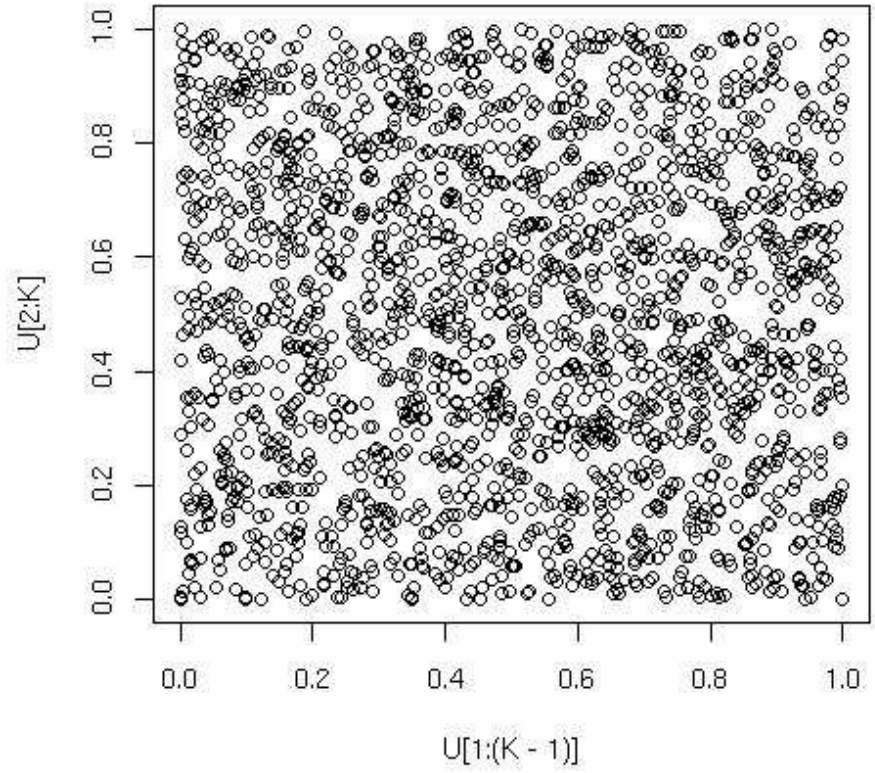
The plot of pairs (U_i, U_{i+1}) for the linear congruential generator with $a = 1229$, $b = 1$, $M = 2048$.



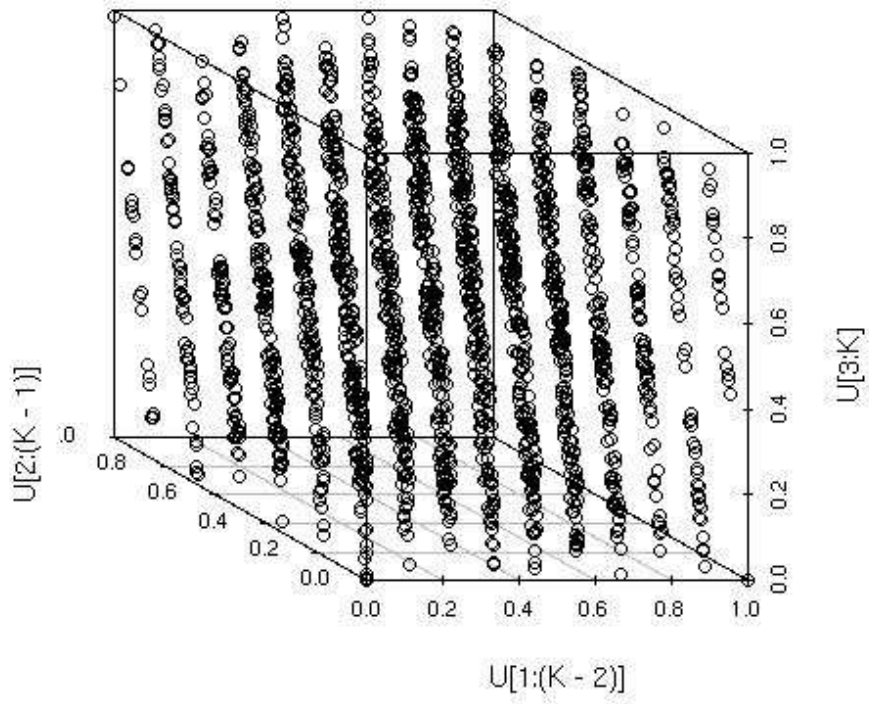
Good congruential generator with $a = 1597$, $b = 51749$ and $M = 244944$



Deceptively good congruential generator with
$a = 2^{16} + 3$, $b = 0$, $M = 2^{31}$



Apparently not so good congruential generator with
$a = 2^{16} + 3$, $b = 0$, $M = 2^{31}$



In general, Marsaglia showed that the m-tuples $(U_i, \ldots, U_{i+m-1})$ generated with linear congruential generators lie on relatively low number of hyperplanes in $\mathbb{R}^m$.

An alternative way to generate uniform deviates is by using **Fibonacci generators**.

Fibonacci generators

The original Fibonacci recursion motivates the following general approach to generate pseudo-random numbers

$$s_i = s_{i-n} \text{ op } s_{i-k}.$$

Here $0 < k < n$ are the lags and **op** can be one of the following operators:

- $+$ addition $\mod M$,
- $-$ subtraction $\mod M$,
- $*$ multiplication $\mod M$.

To initialize (seed) these generators we have to use another generator to supply first n numbers $s_0, s_1, \ldots, s_{n-1}$. In addition, in subtraction generators we have to control that if $s_i < 0$ for some i the result has to be shifted $s_i := s_i + M$.

What are ”good generators”?

Good generators are those generators that pass a large number of statistical tests, see, e.g.,

P. L’Ecuyer and R. Simard – TestU01: A C Library for Empirical Testing of Random Number Generators, *ACM Transactions on Mathematical Software*, Vol. 33, article 22, 2007

TestU01 is a software library, implemented in C, and offering a collection of utilities for the empirical statistical testing of uniform random number generators.

These tests are freely accessible on the web page http://www.iro.umontreal.ca/ simardr/testu01/tu01.html

”Minimal Standard” generators

W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery – *Numerical Recipes in C*
offers a number of portable random number generators, which passed all new theoretical tests, and have been used successfully.

The simplest of these generators, called *ran0*, is a standard congruential generator

$$s_{i+1} = a s_i \mod M,$$

with $a = 7^5 = 16807$ and $M = 2^{31} - 1$, is a basis for more advanced generators *ran1* and *ran2*. There are also better generators like *ran3* and *ran4*.
Of these generators only *ran3* possesses sufficiently good properties to be used in financial calculations.

The Mersenne Twister

The Mersenne Twister of Matsumoto and Nishimura which appeared in in late 90-ties is now mostly used in financial simulations.
It has a period of $2^{19937} - 1$ and the correlation effect is not observed for this generator up to dimension 625.
Mersenne Twister is now implemented in most commercial packages. In particular, it is a standard generator in Matlab, Octave (open-source version of Matlab), R-project, S-plus.

Generation of non-uniformly distributed random deviates

- Idea 1: Discrete distributions
- Idea 2: Inversion of the distribution function
- Idea 3: Transformation of random variables

Idea 1: Discrete distributions

Coin toss distribution: $\mathbb{P}(X = 1) = 0.5$, $\mathbb{P}(X = 0) = 0.5$

(1) Generate $U \sim \mathcal{U}(0, 1)$
(2) If $U \leq 0.5$ then Z = 1, otherwise Z = 0

Z has a coin toss distribution.

Discrete distribution: $\mathbb{P}(X = a_i) = p_i$, $i = 1, 2, \ldots, n$

(1) Compute $c_k = \sum_{i=1}^{k} a_i$
(2) Generate $U \sim \mathcal{U}(0, 1)$
(3) Find smallest k such that $U \leq c_k$. Put $Z = a_k$

Z has a given discrete distribution.

Idea 2: Inversion of the distribution function

Proposition. *Let $U \sim \mathcal{U}(0,1)$ and F be a continuous and strictly increasing cumulative distribution function. Then $F^{-1}(U)$ is a sample of F.*

Theoretically this approach seems to be fine. The only thing we really need to do is to generate uniformly distributed random numbers.

Works well for: exponential distribution, uniform distribution on various intervals, Cauchy distribution.

Beasley-Springer-Moro algorithm for normal variate uses inversion of the distribution function with high accuracy (3×10^{-9}).
In interval $0.5 \leq y \leq 0.92$ the algorithm uses the formula

$$F^{-1}(y) \approx \frac{\sum_{n=0}^{3} a_n (y-0.5)^{2n+1}}{1+\sum_{n=0}^{3} b_n (y-0.5)^{2n}},$$

and for $y \geq 0.92$ the formula

$$F^{-1}(y) \approx \sum_{n=0}^{8} c_n \Big(\log\big(-\log(1-y)\big)\Big)^n.$$

Idea 3: Transformation of random variables

Proposition. *Let X be a random variable with density function f on the set $A = \{x \in \mathbb{R}^n | f(x) > 0\}$. Assume that the transformation $h : A \to B = h(A)$ is invertible and that the inverse h^{-1} is continuously differentiable. Then $Y = h(X)$ has the density*

$$y \mapsto f(h^{-1}(y)) \cdot \left| \det \left(\frac{dh^{-1}}{dy}(y) \right) \right|,$$

for all $y \in B$.

We are going to apply this result for the case where $A = [0,1]^2$ and $f(x) = 1$ for all $x \in A$ (i.e. we start with a two-dimensional uniformly distributed random variable) and choose the transformation

$$x \mapsto h(x) = \begin{pmatrix} \sqrt{-2\ln x_1}\cos(2\pi x_2), \\ \sqrt{-2\ln x_1}\sin(2\pi x_2). \end{pmatrix}$$

The inverse of this transformation is given by

$$y \mapsto h^{-1}(y) = \begin{pmatrix} \exp(-\|y\|^2/2), \\ \arctan(y_2/y_1)/2\pi. \end{pmatrix}$$

We compute the determinant of the derivative at y as follows:

$$\det\Big(\frac{dh^{-1}}{dy}(y)\Big) = -\frac{1}{2\pi}\exp\Big(-\frac{1}{2}(y_1^2+y_2^2)\Big)$$

Therefore, the density function of $Y = h(X)$, where $X \sim U([0,1]^2)$ equals

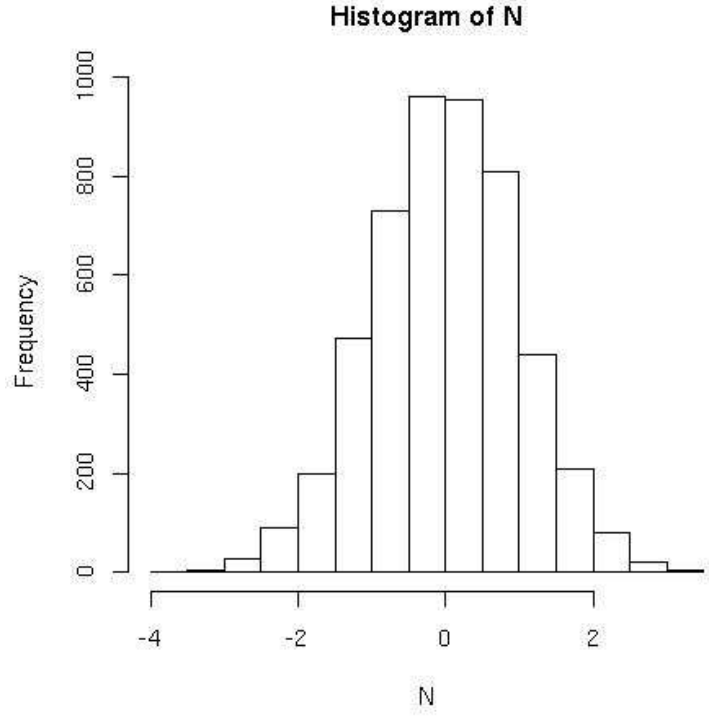
$$f\big(h^{-1}(y)\big)\cdot\Big|\det\Big(\frac{dh^{-1}}{dy}(y)\Big)\Big| = 1\cdot\frac{1}{2\pi}\exp\Big(-\frac{1}{2}(y_1^2+y_2^2)\Big).$$

This is obviously the density function of the two-dimensional standard normal distribution.
We therefore obtain that $h(X)$ is 2-dimensional standard normally distributed, whenever X is uniformly distributed on $[0,1]^2$.

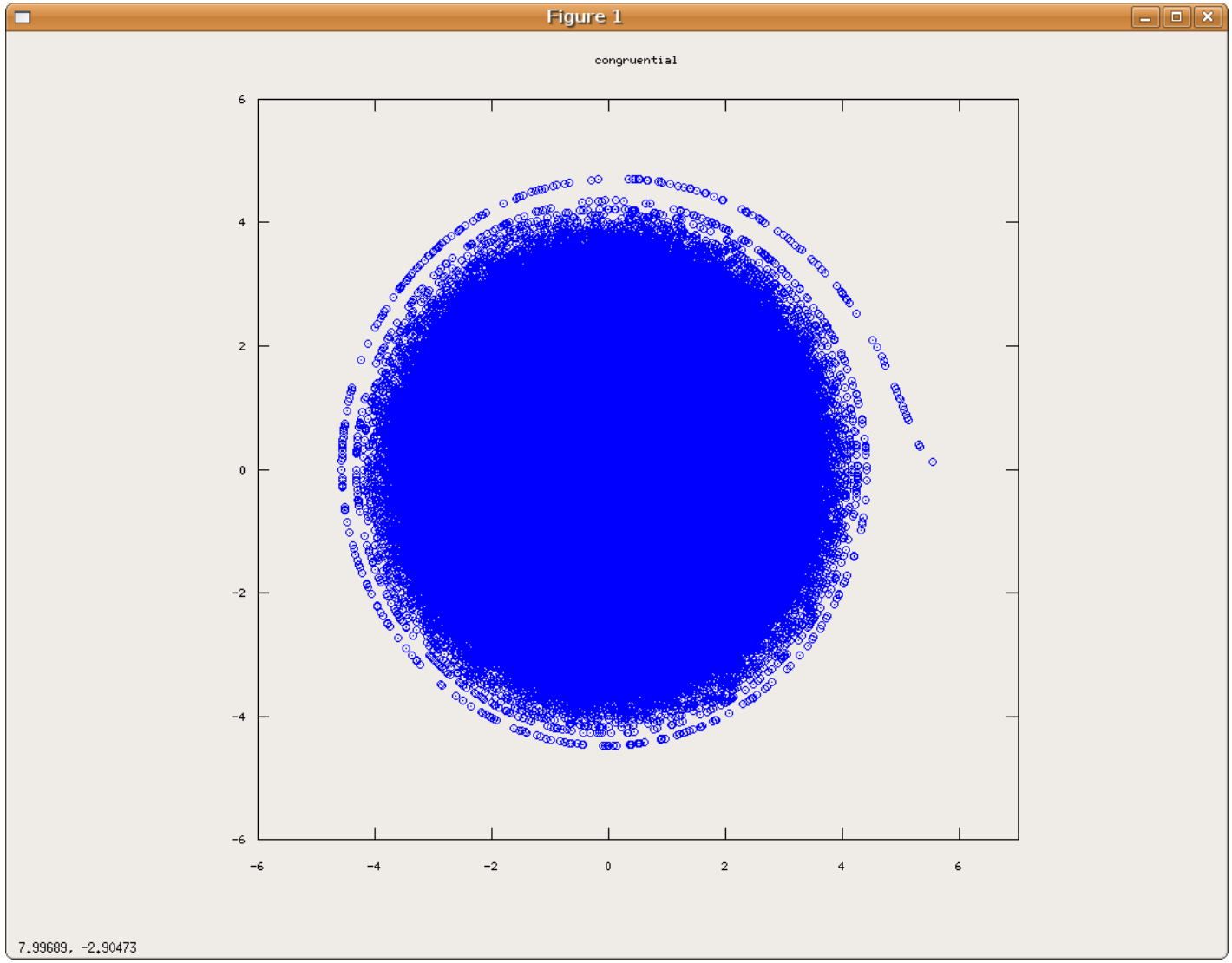
Box-Muller algorithm

Creates $Z \sim \mathcal{N}(0,1)$:

1. Generate $U_1 \sim \mathcal{U}[0,1]$ and $U_2 \sim \mathcal{U}[0,1]$,
2. $\theta = 2\pi U_2$, $\rho = \sqrt{-2\log U_1}$,
3. $Z_1 := \rho\cos\theta$ is a normal variate (as well as $Z_2 := \rho\sin\theta$).



Box-Muller with congruential generator



Marsaglia algorithm

The Box-Muller algorithm has been improved by Marsaglia in a way that the use of trigonometric functions can be avoided. It is important, since computation of trigonometric functions is very **time-consuming**.

Algorithm: polar method (creates $Z \sim \mathcal{N}(0,1)$):

1. *repeat* generate $U_1, U_2 \sim \mathcal{U}[0,1]$;
   $V_1 = 2U_1 - 1$, $V_2 = 2U_2 - 1$;
   until $W := V_1^2 + V_2^2 < 1$.
2. $Z_1 := V_1 \sqrt{-2\ln(W)/W}$
   $Z_2 := V_2 \sqrt{-2\ln(W)/W}$
   are both normal variates.

Generation of $\mathcal{N}(\mu, \sigma^2)$ distributed pseudo-random variables

(1) Compute $Z \sim \mathcal{N}(0, 1)$

(2) $Z_1 = \mu + \sigma Z$

Z_1 is a pseudo-random number from a normal distribution $\mathcal{N}(\mu, \sigma^2)$.

How to generate a sample from a multidimensional normal distribution?

Theorem. *Let Z be a vector of p independent random variables each with a standard normal distribution $\mathcal{N}(0,1)$. There exists a matrix A such that*

$$\mu + AZ \sim \mathcal{N}(\mu, \Sigma).$$

Our aim is to find such a matrix A. We know how to generate a sequence of independent normally distributed random variables. Using matrix A we can transform it into a sequence of multidimensional normal variates.

Cholesky decomposition

The covariance matrix Σ is positive definite. One can show that there is exactly one lower-triangular matrix A with positive diagonal elements, s.t. $\Sigma = A \cdot A^T$, where A^T denotes a transpose of a matrix A. This decomposition is called the **Cholesky decomposition**.

Algorithm for generation of $\mathcal{N}(\mu, \Sigma)$ distributed pseudo-random variables:

1. Calculate the Cholesky decomposition $AA^T = \Sigma$.
2. Calculate $Z \sim \mathcal{N}(0, I)$ componentwise by $Z_i \sim \mathcal{N}(0, 1)$, $i = 1, \ldots, n$.
3. $\mu + AZ$ has the desired distribution $\sim \mathcal{N}(\mu, \Sigma)$.

PCA construction

Since Σ is symmetric positive definite matrix,

$$\Sigma = \Gamma \Lambda \Gamma^T,$$

where Γ is the matrix of d eigenvectors of Σ and Λ is the diagonal matrix of eigenvalues of Σ.

Theorem. *Let Z be a vector of d independent random variables each with a standard normal distribution $\mathcal{N}(0,1)$. Then*

$$\mu + \Gamma \Lambda^{1/2} Z \sim \mathcal{N}(\mu, \Sigma),$$

where $\Sigma = \Gamma \Lambda \Gamma^T$ is the spectral decomposition of matrix Σ.

Due to the positivity of the eigenvalues $\Lambda^{1/2}$ is well defined.

In general Cholesky decomposition and PCA construction do not give the same results:

$$\Gamma \Lambda^{1/2} \neq A.$$

t-Student distribution

d-dimensional t-Student distribution $t_d(\nu, \mu, \Sigma)$ with ν degrees of freedom, where μ – vector of localization, Σ – matrix of dispersion, can be generated using the relation

$$t_d(\nu, \mu, \Sigma) \sim \frac{\mathcal{N}(\mu, \Sigma)}{\sqrt{\chi^2_\nu / \nu}},$$

where χ^2_ν is a chi-square distribution.

Generation of a sample of variable Z with the distribution $t_d(\nu, \mu, \Sigma)$:

1. Generate $X \sim \mathcal{N}(\mu, \Sigma)$.
2. Generate $Y \sim \chi^2_\nu$, independent from X.
3. Take $Z = \frac{X}{\sqrt{Y/\nu}}$.

Random variable Z has distribution $t_d(\nu, \mu, \Sigma)$.

t-Student distribution in 2D