**Martin Potužník**
Charles University, Prague

**Peter Hinow**
Dresden University of Technology

# Deterministic patterns in pseudorandom point sets

Elaborated during the Workshop on Simulation
Dresden, September 23rd – October 2nd 1997

**ABSTRACT** : Many processes in applied sciences can be studied using stochastic simulation. Vital to these methods are often normally distributed random numbers. For the generation of such normally distributed random numbers several techniques can be used, for instance the Box-Muller-Algorithm or von Neumann's rejection method. It has been pointed out by Ripley [1] that the Box-Muller algorithm depends sensitively on the independence of the incoming uniformly distributed random numbers. Pairs of succesive normals tend to form spirals if, for instance, a linear congruency method is used for generating the input random numbers. In this paper we investigate some of these patterns and suggest ways to avoid such disturbing phenomena.

We would like to thank The Institute of Mathematical Stochastics at the Dresden University of Technology for the realization of the Workshop on simulation. Especially we are indebted to Dr. Peter Neumann for lecturing at this workshop. We also thank Dr. Aleksander Janicki (Wroclaw University of Technology) and Dr. Jiří Bok (Charles University, Prague) for useful suggestions and help.

## Introduction

Many problems in sciences like physics, engineering and economics are far too complex to be studied analytically. In these cases, computer simulation has become a vital tool for their numerical solution. Essential for all simulation methods are random number generators. However, it has been criticized by Ripley [1], that the statistic properties of such random number generators are often not understood well enough. For instance, when applying the Box-Muller algorithm to pairs of consecutive uniformly distributed random numbers generated by the linear congruency method, disturbing geometric patterns can be observed, which suggest, that the normally distributed random numbers are not "random" enough. Surely, simulations using such random numbers are not reliable. It was our task to reproduce such patterns and to propose ways for avoiding them.

## Theoretical aspects

To solve the problem described above we developed a `PASCAL` program, which generates normally distributed random numbers using different algorithms. The algorithms we investigated were the following:

- **Box-Muller algorithm** (1958) which generates a pair of independent exactly $N(0,1)$-disrtibuted random numbers $(z_1, z_2)$ using a pair $(u_1, u_2)$ of uniformly distributed random numbers $\in [0,1)^2$ according to:
$$z_1 = \sqrt{-2\ln u_1}\cos 2\pi u_2,$$
$$z_2 = \sqrt{-2\ln u_1}\sin 2\pi u_2.$$
- **Central limit theorem**, which generates one approximately $N(0,1)$-distributed random variable $(z)$ using $n$ (in our case 100) uniformly distributed random numbers $\in [0,1)$ $(u_i)$ according to:
$$z = \sum_{i=1}^{n} \frac{(u_i - \frac{1}{2})}{\sqrt{\frac{n}{12}}}.$$
- **Von Neumann's rejection method**, which generates exactly $N(0,1)$-distributed random variable $(z)$ using two uniformly distributed random numbers $(u_1, u_2)$ the way, that $z = u_1$ only if $u_2 < h(u_1)$, where $h$ is the desired density function – in our case the standard normal density function, i.e.
$$h(u) = \frac{1}{\sqrt{2\pi}}\exp\left[\frac{-u^2}{2}\right].$$

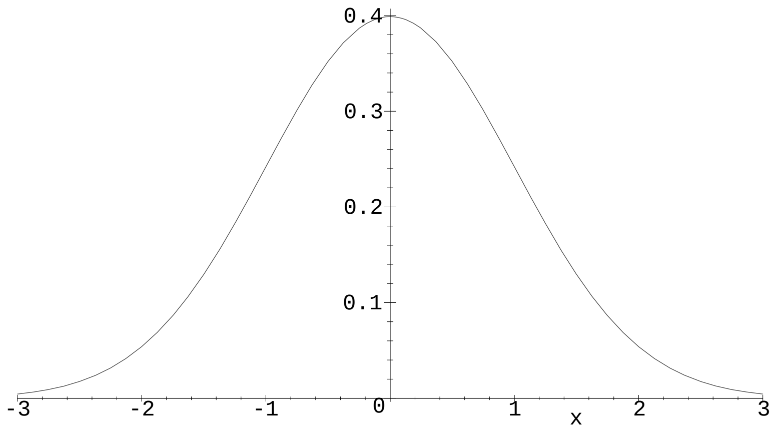


Fig.1: Density function of Gaussian normal distribution

All these methods transform uniformly distributed random numbers into normally distributed ones. It can be chosen, which generator for uniformly distributed random numbers is used:

- **Linear congruency method.** Recursive method, $a, c \in \mathbf{N}, m \in \mathbf{N}$ (modulus), generating uniformly distributed random numbers from $\{0, 1, \ldots, m-1\}$ according to:
$$u_i = (au_{i-1} + c) \bmod m.$$
- **Quadratic congruency method**. Recursive method, $a, b, c \in \mathbf{N}, b \neq 0, m \in \mathbf{N}$ (modulus), generating uniformly distributed random numbers from $\{0, 1, \ldots, m-1\}$ according to:
$$u_i = (bu_{i-1}^2 + au_{i-1} + c) \bmod m.$$
- **Inversive congruency method**. Recursive method, $a, c \in \mathbf{N}, m \in \mathbf{N}$ (modulus), $m \geq 5$ generating uniformly distributed random numbers from $\{0, 1, \ldots, m-1\}$ according to:
$$u_i = (au_{i-1}^{-1} + c) \bmod m,$$
where $u^{-1}$ is the inverse element of $u$ in the Galois field $GF(m)$ if $u \not\equiv 0\,(mod\,m)$, else $u^{-1} = 0$. In our program $m$ is a prime, because otherwise the computation of $u^{-1}$ becomes too complicated and therefore requires too much computation time.
- **Fibonacci algorithm**, recursive method, $m \in \mathbf{N}$ (modulus), generating uniformly distributed random numbers from $\{0, 1, \ldots, m-1\}$ according to:
$$u_{i+1} = (u_{i-1} + u_i) \bmod m.$$
- **Wichmann & Hill algorithm**, which is actually a linear combination of three linear congruency methods, using specific constants, see Wichmann & Hill [3].
- **PASCAL's internal random number generator**.

## Features of the program

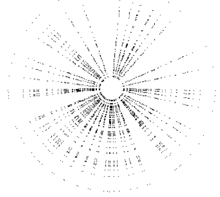
To avoid a loss of precision, we did not generate uniformly distributed random numbers as `real` or `double` numbers, but as `longint` numbers. These are transformed into `real` numbers only for the translation into normally distributed random numbers.

We made it possible to choose constants for the linear congruency method, since this method allows also bigger constants. You can use some small, not really suitable constants, as well as "well prooved" constants from literature, e.g. from Janicki [4], Ripley [1] or Matloff [2].

## Observations

We first drew our attention to the Box-Muller algorithm and compared visually the influence of different uniformly distributed random numbers on the quality of the normally distributed random number generator. Here is, what we observed:

**Linear congruency methods** produce with some constants (e.g. m=541 or m=31218 [J.Bok]) a sparse, but strongly regular pattern, see `Figs.2 and 3`. With other constants (e.g. m=1013, "Ripley" or "Matloff"), the cycling period becomes longer, however spiral patterns appear in various shapes, see `Figs.4 and 5`. When using the "Janicki" constants, patterns show up very late.



`Fig. 2: Box-Muller algorithm, linear congruency method, m=541`

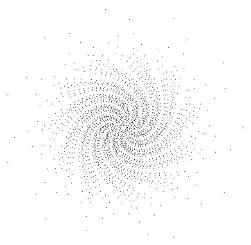


Fig. 3: Box-Muller algorithm, linear congruency method, J.Bok constants

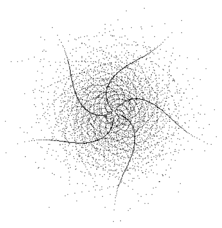


Fig. 4: Box-Muller algorithm, linear congruency method, m=1013

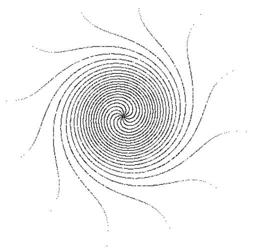


Fig. 5: Box-Muller algorithm, linear congruency method, Ripley constants

Concerning **quadratic congruency method**, we used only one set of constants (m=1013), since the square of $u$ is becoming rather huge already for small values of $m$. This generator actually behaves in a very similar way as the linear congruency method, this holds especially for the patterns obtained, see `Fig.6`.

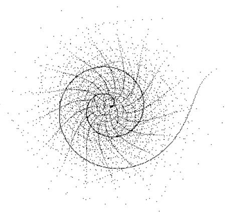


Fig. 6: Box-Muller algorithm, quadratic congruency method

Concerning **inversive congruency method**, this method seems to be rather good, but with the time you will also obtain some (not so distinct) pattern.

We did not observe any patterns when using **Fibonacci algorithm** and **Wichmann & Hill algorithm**, see `Fig.7`.

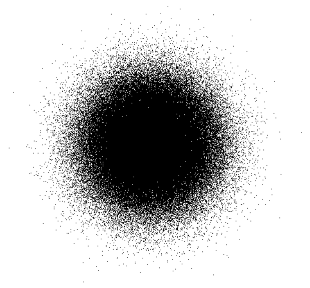


Fig. 7: Box-Muller algorithm, Wichmann & Hill algorithm

**PASCAL's internal random number generator** seems to work satisfactory, with the time you will also obtain some pattern, however not a spiral. We observed some similarity in patterns produced by PASCAL's internal random number generator and the inversive congruency method.

The idea to transform uniformly distributed random numbers into normally distributed random numbers using the central limit theorem can be recommended only in a few cases. If the incoming random numbers are produced by a linear congruency method, it is necessary to chose a sufficiently large modulus, in practice $\geq 100000$ . Using quadratic congruency method, a ragged geometric pattern appears, see `Fig.8`. Since 100 consecutive random numbers are needed for the generation of a single normally distributed random number, the required computation time must also be taken into consideration. Here the inversive congruency method as well as the Wichmann & Hill algorithm fail to prove being suitable, just PASCAL's internal random number generator works quick enough, while showing no pattern.

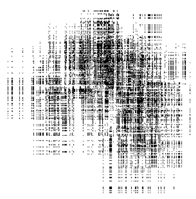


Fig. 8: central limit theorem, quadratic congruency method

Von Neumann's rejection method reacts rather sensibly on the quality of the incoming uniformly distributed random numbers. Nearly all linear congruency methods as well as the quadratic congruency method and the inversive congruency method cause the generator to cycle after only 100..200 random numbers. Sure enough, this excludes this method for most simulation porblems. As an example for the deterministic pattern we refer to `Fig.9`. In contrary to this behaviour, von Neumann's rejection method in combination with Wichmann & Hill algorithm or PASCAL's internal random number generator seems to be among the best of all proposed methods, see `Fig.10`.

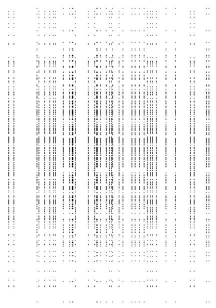


Fig. 9: von Neumann's rejection method, linear congruency method

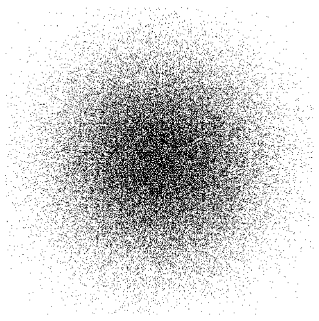


Fig. 10: von Neumann's rejection method, Wichmann & Hill algorithm

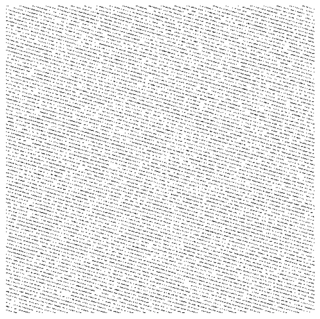


Fig. 11: uniformly distributed random numbers generated using linear congruency method

## Conclusions

Satisfactory results can be obtained by using a suitable combination of an uniformly distributed random number generator and a transformation method. What concerns the reasons for the observed spirals, we suggest that they are the images of linear patterns produced by the generator for the uniformly distributed random numbers (see `Fig.11`) under a continuous and differentiable mapping, in our case of course the Box-Muller algorithm. Using a discontinuous transformation mapping like von Neumann's rejection method helps to avoid such disturbing phenomena.

## References

[1] Ripley, B. D., Uses and abuses of statistical simulation, Mathematical Programming **42** (1988), p. 53-68

[2] Matloff, N. S., Probability modelling and computer simulation, PWS-Kent Publishing Company, Boston 1988

[3] Wichmann, B. A., Hill, I. D., Building a random number generator, Byte **12** (3) (1987), p. 127-128

[4] Janicki, A., Weron, A., Simulation and chaotic behavior of $\alpha$-stable stochastic processes, Dekker, New York 1994