Random Number Generation

Carmina Tió Sirvach Jumani Soheil Shams Zhuocheng Xie

Universität Trier

Introduction and Motivation

Since natural random numbers are not reproducible, we often work with pseudo-random numbers that are generated using mathematical procedures.

Applications of Random Number Generation

- Randomly assign samples into different groups.
- Simulate a process, approximate or estimate a value.
- Encrypted information for safety communication.

We used linear congruential generator to check the patterns while using good and bad parameters.

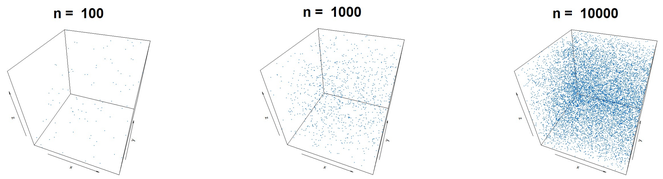


Figure 1. Good Generator

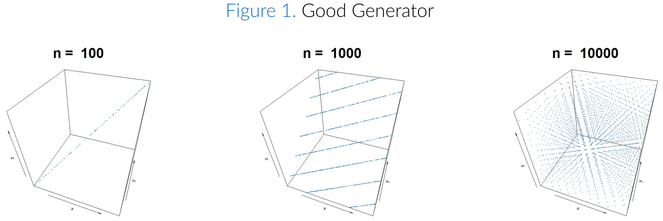


Figure 2. Bad Generator

Essential Properties of a Random Number Generator

Random numbers must be tested because the random numbers we generate are pseudo random numbers and not real, and a sequence of random numbers must be uniformly distributed and independent.

- **Uniformity:** Random number generator should be uniformly distributed and unbiased.
- **Independence:** The current value of a random variable has no relation with the previous values.
- **Reproducibility:** The same sequence should be produced with the same initial values (or seeds).
- **Randomness:** Should produce independent uniformly distributed random variables that pass all statistical tests for randomness.
- **Long Period:** A pseudo-random number sequence uses finite numbers, so the sequence must repeat itself with a finite period. This should be longer than the amount of random numbers needed for the simulation.

Methods

- **Fibonacci Generator:** The Fibonacci generator is based on the Fibonacci sequence. We get a generalization by the modulo factor m and by lagging the sequence by j and k.

$$x_n = x_{n-j} * x_{n-k} \ mod \ m$$

- **Inversive Congruential Generator:** The non-linearity of the inverse congruential method is accomplished by using the multiplicative inversion operation with respect to a defined modulus.

$$x_n = (ax_{n-1}^{-1} + c) \ mod \ m$$

- **Multiply with Carry Number Generator:** The key benefit of the MWC method is that it expresses simple computer integer arithmetic and leads to very fast generation of random number sequences with enormous periods.

$$x_n = (ax_{n-1} + c_{n-1}) \ mod \ b, \ c_n = \frac{ax_{n-1} + c_{n-1}}{b}$$

- **Combined Generator (Using K Multiplicative Congruential Generator):** The combination of more than one generator will improve both the period and the apparent randomness of the random number generators.

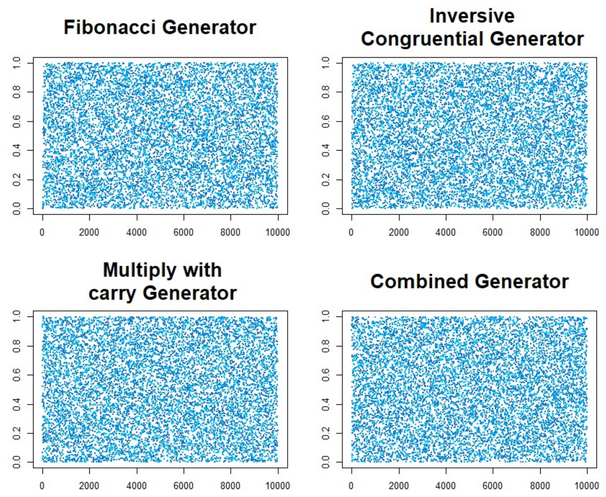
$$x_n = (a * x_{n-1}) \ mod \ m_1$$

$$y_n = (b * y_{n-1}) \ mod \ m_2$$

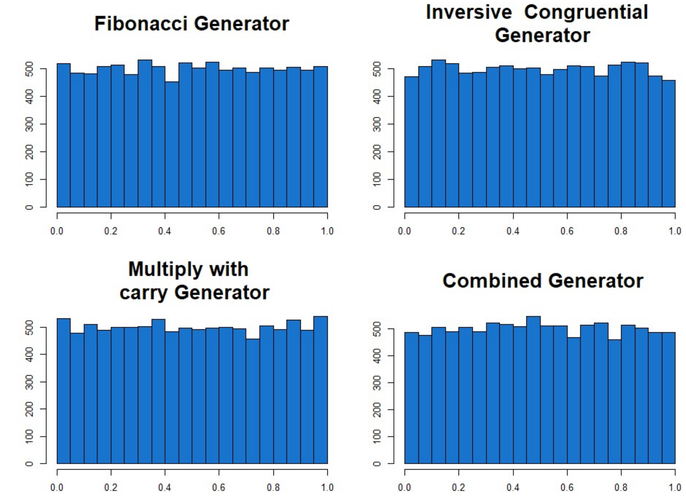
$$z_n = (x_n - y_n) \ mod \ m_1$$

$$u_n = 4.656613 * z_n * 10^{-10}$$

Results (Scatter Plots)



Results (Histograms and Tests)



Name	Serial Correlation	KS-Test	Period	Time
Fibonacci	-0.00020801	$H_0 : X \sim Unif; p > 0.5$	2^{32}	5.00
Inversive	0.000349379	$H_0 : X \sim Unif; p > 0.5$	$2^{31} - 1$	4.94
Multiply	0.002646124	$H_0 : X \sim Unif; p > 0.5$	2^{32}	5.89
Combined	0.000093411	$H_0 : X \sim Unif; p > 0.5$	10^{18}	12.97

Conclusion

Overall we can say in this study that all RNGs are performing well in terms of uniformity, randomness and independence. The Combined Generator has the longest period of RNGs and takes the longest time to generate its sequences. We could also observe that the seed of the parameters plays an important role in the performance of the RNG.

References

[1] Zhou Bo, and Qiankun Song. *Period Distribution of Inversive Pseudorandom Number Generators Over Finite Fields*. arXiv preprint arXiv:1209.1295, 2012.

[2] Marsaglia, George. *On the randomness of pi and other decimal expansions*. InterStat 5, 2005.

[3] Niederreiter, Harald. *Random Number Generation and Quasi-Monte Carlo Methods*. Vol. 63, Society for Industrial and Applied Mathematics, 1992.

[4] Gentle, James E. *Random Number Generation and Monte Carlo Methods*. Springer, 2003.

[5] Dagpunar, John. *Principles of Random Variate Generation*. Clarendon Press, 1988.