SILESIAN UNIVERSITY OF TECHNOLOGY PUBLISHING HOUSE

SCIENTIFIC PAPERS OF SILESIAN UNIVERSITY OF TECHNOLOGY 2022
ORGANIZATION AND MANAGEMENT SERIES NO. 167

RANDOMNESS TESTING OF THE RANDOM NUMBER GENERATORS USING DIEHARDER TOOL

Piotr P. JÓŹWIAK[1*], Ireneusz J. JÓŹWIAK[2], Krzysztof JUSZCZYSZYN[3], Tomasz MAŁACHOWSKI[4]

[1] Wroclaw University of Science and Technology; piotr.jozwiak@pwr.edu.pl, ORCID: 0000-0002-5325-3728
[2] General T. Kościuszko Military University of Land Forces in Wrocław; ireneusz.jozwiak@awl.edu.pl, ORCID: 0000-0002-2160-7077
[3] Wroclaw University of Science and Technology; krzysztof.juszczyszyn@pwr.edu.pl, ORCID: 0000-0002-9326-6734
[4] Wroclaw University of Science and Technology; malachowski.tomasz@outlook.com, ORCID: 0000-0002-4626-7902
* Correspondence author

Purpose: The aim of the research is to determine whether the Dieharder battery of statistical tests suite is able to demonstrate the superiority of the true random number generator over pseudorandom number generators.
Design/methodology/approach: Based on a number of random number sequences obtained from different generators, the randomness of these sequences was tested and the results obtained were compared between different classes of random number generators.
Findings: The research indicated that we are not able to determine in a positive sense the quality of a given generator on the basis of statistical testing with a Dieharder battery, but only able to determine whether there are no grounds to reject the generator as non-random. Statistical testing only has the character of a negative criterion.
Originality/value: The research carried out provides an answer to the question of whether statistical randomness testing with a battery of Dieharder tests can provide information about the level of randomness of a given generator in relation to another generator, when both random sequences have passed the tests. The results of the research indicate that additional quality criteria should be taken into account when selecting a random number generator that passes the statistical tests in order to unambiguously answer which generator is better.

Keywords: strategy of randomness testing, random number generator, quantum cryptography, Dieharder.

Category of the paper: Research paper.

1. Introduction

One of the many characteristics of the world around us is randomness. Randomness is understood as the lack of connection between events or the lack of predictability of these events and their causality. What for a human may be a random event, from the mathematics point of view may not. Encryption is a good example of this.

Randomness is fundamental in many algorithms, not only in computer science. However, good quality random numbers are essential for developing secure cryptography methods.

However, testing random number generators, due to its non-deterministic nature, is a complex issue (Kałuski, 2012). Therefore, the Dieharder tool (Brown, 2022) was proposed to determine whether the tested generator has the characteristics of a good quality random number generator. In this paper, an analysis of randomness testing of different random number generators will be conducted. The study uses the Dieharder tool used in randomness testing of pseudorandom and random number generators. Based on the generators built there, as well as statistical tests, testing of the quality of the random numbers generated was carried out. The aim of this paper is to try to determine whether the battery of Dieharder tests will indicate the superiority of truly random generators over pseudo-random generators. This includes a new class of quantum generators that fundamentally deliver unpredictability and randomness according to the Copenhagen interpretation of quantum mechanics.

2. Characteristics of the used sources of randomness

The literature distinguishes two basic classes of random number generators (L'Ecuyer, 2021):

- Pseudo random number generators – PRNG.
- True random number generators - TRNG.

Pseudo random numbers can be obtained from mathematical algorithms. Generators of this type require an initial entropy seed to be supplied to the generator input (Marsaglia, 2003). The next state is generated deterministically by applying a mathematical function. These are the so-called LCG (Linear Congruential Generator) generators. Generators of this type have the serious drawback of a limited number of states. After a certain number of steps of the generator the internal state loops and the generator repeats its work. This type of behaviour makes it possible to predict each subsequent state of the generator, especially if the attacker knows the seed used to run the generator. Despite these drawbacks, due to the high speed of these generators, they also find application in cryptography.

The second group of number generators are true random number generators. They differ significantly from pseudorandom number generators. These numbers can be obtained from various sources of physical phenomena by using magnetic fields, light intensity, sound waves, or quantum phenomena (Jian, et al., 2011). A good example of generating truly random numbers is a generator using the decay of radioactive elements. However, these generators are slow. They are chosen when the quality of the numbers generated is more important than the quantity.

There are many more practical quantum phenomena which can be considered as good candidate processes for the truly non-deterministic randomness generation. An in-detail study can be found in (Jacak, Jóźwiak et al., 2021) mainly focusing on quantum shot noise as well as on quantum optics. It is possible to consider quantum phenomena in nano-plasmonics as well, yet of a lesser practical significance (Jacak, 2020). Furthermore quantum entanglement independently of its technical implementations can bring important advantages in efficient randomness testing (Jacak, J., Jacak, W. et al., 2020), which is of critical role for implementing Quantum Key Distribution (Jacak, M., Jacak, J. et al., 2016; Jacak, Mielniczuk et al., 2015).

Hardware randomness generators should be equipped with internal testing devices. However, this type of approach is not trivial when using phenomena of the surrounding nature. Therefore, the most common choice is an empirical approach (Knuth, 1997), in particular statistical randomness testing. The most important implementations of the tests are the NIST (Rukhin et al., 2010) and Dieharder (Hotoleanu et al., 2010; Brown, 2022; Suresh et al., 2013; Vascova et al., 2010) batteries. In this paper, a set of Dieharder tests was chosen to study generators. Note that in addition to empirical implementations of the tests, methods for quantum generator properties are detailed.

Three algorithmic pseudo-random number generators were used for the study:

- *rand()* - C/C++ library function used to generate a random number. Does not allow the initial entropy to be given as a seed. Used as a known very weak source of random numbers,
- *ran1* (Class Ran1, 2022) - a generator developed by Park and Miller (Park et al., 1988) with the Bays-Durham shuffling algorithm (Bays et al., 1976),
- *random256-glibc2* generator from GNU C library.

Another pseudorandom number generator uses a virtual device on a unix machine. This solution is based on generating random numbers based on interrupts occurring in the system from device drivers and other events in the operating system:

- */dev/urandom* - random number generator using system events to obtain.

A triple pendulum has been used as an example of a generator using a physical phenomenon. Such a implementation becomes a chaotic system when tilted appropriately (Stachowiak et al., 2006; Botha et al., 2013). The generator is an implementation based on the work of (Nouar et al., 2020) and (Awrejcewicz et al., 1999). Three masses are sequentially connected to each other by inextensible rods, and the first one is additionally attached to a fixed point in space.

This is a physical phenomenon whose behavioural description we are able to calculate. In the literature this effect is known as the "Butterfly Effect":

- *Triple pendulum* (Małachowski, 2021).

As a counterbalance to the pseudo-random generators discussed earlier, a sequence of numbers generated by a quantum random number generator called:

- *ANU QRNG* (Małachowski, 2021).

Unlike previous generators, this generator is fully non-deterministic. This generator was developed at the Australian National University (ANU). The generation of random numbers is carried out on the basis of phenomena occurring in a vacuum (Botha et al., 2013). As the authors of this generator point out, the definition of a vacuum in classical terms differs from the quantum definition. In classical physics, a vacuum is considered as a space that is empty of matter or photons. Quantum physics however says that same space resembles a sea of virtual particles appearing and disappearing all the time. These particles produce a magnetic field that causes minute changes in phase and amplitude at all frequencies of the waves passing through the field (ANU QRNG, 2022). The researchers, by using a laser, are able to read these differences, allowing for a high-throughput quantum generator. The authors of the solution provide an API in which quantum random number strings can be obtained on the project website (ANU QRNG 2022). For the research in this paper, the mentioned API was used to obtain test samples.

3. Statistical testing of randomness with Dieharder test suite

Empirical randomness testing uses various types of statistical methods to test for randomness based on hypothesis testing. Generally, in this type of testing, the null hypothesis H_0 is set, which generally reads: *the sequence under test is random*. Directly related to the definition of the null hypothesis is the alternative hypothesis H_1, which takes the opposite claim to H_0, namely that: *the sequence under test is not random*. Initially, the position is taken that the null hypothesis is true and, on the basis of a given statistical test, an attempt is made to show that it is not. If the test is confirmatory then the null hypothesis is rejected and the alternative hypothesis is accepted as valid for the given random sequence under test. Otherwise, there is no basis for rejecting the null hypothesis H_0 (Bobrowski, 1986).

Directly connected with hypothesis testing is the determination of the significance level α for a given test. For the adopted significance level α there is a critical area R_α. If the value of the K statistic does not belong to the critical area R_α, we have no grounds to reject the hypothesis of randomness of the sample. Otherwise, we reject the null hypothesis and accept the alternative one that the sample is not random. A significance level of $\alpha = 0.005$ was assumed in the conducted study.

To allow easier analysis of results, each statistical test provides a result in the form of a number called P_{value}. Each P_{value} is the probability that an ideal random number generator would generate a sequence of numbers less random than the sequence being tested. Note that if P_{value} for the test equals 0, then the sequence of numbers appears to be completely non-random. This is because we need to perform many statistical tests in order to accurately test a sequence of numbers against the null hypothesis of randomness. Each statistical test examining different characteristics provides a non-normal value of the statistic. Only the calculation of P_{value} introduces a standardised measure for the entire set of statistical tests.

A positive pass of a given statistical test is taken to mean that the inequality $P_{value} > \alpha$ must be satisfied. In addition, in order to consider that a given sequence does not show basis for rejecting the hypothesis of its randomness, the sequence must obtain positive results for all tests in the battery.

The Dieharder statistical test battery (Brown, 2022) was developed by Robert G. Brown based on the Diehard test battery proposed by George Marsaglia (Marsaglia, 1996). The author has improved and developed the basic battery with additional statistical tests. The current version 3.31.1 has 31 implemented tests listed in table 1. Not all tests are considered reliable, therefore the author of this paper marks 4 tests as not recommended in randomness testing. In this paper the non-recommended tests have been omitted.

4. Description of research method

Test samples with sizes close to 5GB were obtained from each of the six generators. Each test from the Dieharder battery was repeated a hundred times in five iterations, obtaining multiple P_{value} for each test of a given generator. This fulfils the requirement for multiple string testing to minimise the possibility of confusion. Finally, the Kolmogorov-Smirnov statistic is calculated. This test is designed to verify that for a given significance level $\alpha = 0.005$ the obtained distribution of P_{value} values is consistent with the theoretical uniform distribution. Application of this test allows to obtain an unambiguous answer whether a given generator can be considered random. The Python library scipy.stats and the kstest test were used to calculate the statistic. The library easily allows to calculate the Kolmogorov-Smirnov consistency test providing the result in the form of normalized P_{value}.

In addition, the final value calculated for the tested generators will answer the question of whether the battery of Dieharder tests will be able to show that the quantum random number generator is superior to algorithmic generators. If this is the case, then the Kolmogorov-Smirnov test should show a better distribution of P_{value} derived from individual statistical tests for this generator relative to deterministic generators.

Table 1.
Randomnes test results obtained using the Dieharder test battery

No.	Test name	Test status	Random number generators test results					
			rand()	**ran1**	**random256 glibc**	**/dev/ urandom**	**triple pendulum**	**ANU QRNG**
1.	Diehard Birthdays	Good	passed	passed	passed	passed	passed	passed
2.	Diehard OPERM5	Good	passed	passed	passed	passed	passed	passed
3.	Diehard 32x32 Binary Rank	Good	passed	passed	passed	passed	passed	passed
4.	Diehard 6x8 Binary Rank	Good	**failed**	passed	passed	passed	passed	passed
5.	Diehard Bitstream	Good	**failed**	passed	passed	passed	passed	passed
6.	Diehard OPSO	Suspected	skipped					
7.	Diehard OQSO	Suspected	skipped					
8.	Diehard DNA	Suspected	skipped					
9.	Diehard Count the 1s (stream)	Good	**failed**	passed	passed	passed	passed	passed
10.	Diehard Count the 1s Test (byte)	Good	**failed**	passed	passed	**failed**	passed	passed
11.	Diehard Parking Lot	Good	passed	passed	passed	passed	passed	passed
12.	Diehard Minimum Distance (2d Circle)	Good	passed	passed	passed	passed	passed	passed
13.	Diehard 3d Sphere (Minimum Distance)	Good	passed	passed	passed	passed	passed	passed
14.	Diehard Squeeze	Good	passed	passed	passed	passed	passed	passed
15.	Diehard Sums	Bad	skipped					
16.	Diehard Craps	Good	passed	passed	passed	passed	passed	passed
17.	Marsaglia and Tsang GCD	Good	**failed**	passed	passed	passed	passed	passed
18.	STS Monobit	Good	passed	passed	passed	passed	passed	passed
19.	STS Runs	Good	passed	passed	passed	passed	passed	passed
20.	STS Serial (Generalized)	Good	passed	passed	passed	passed	passed	passed
21.	Diehards Runs	Good	passed	passed	**failed**	passed	passed	passed
22.	RGB Generalized Minimum Distance	Good	**failed**	passed	passed	passed	passed	passed
23.	RGB Permutations	Good	passed	passed	passed	passed	passed	passed
24.	RGB Lagged Sum	Good	passed	passed	passed	passed	passed	passed
25.	RGB Kolmogorov-Smirnov	Good	passed	passed	passed	passed	passed	passed
26.	Byte Distribution	Good	**failed**	passed	passed	passed	passed	passed
27.	DAB DCT	Good	**failed**	passed	passed	passed	passed	passed
28.	DAB Fill Tree	Good	passed	passed	passed	passed	passed	passed
29.	DAB Fill Tree 2	Good	**failed**	passed	passed	passed	passed	passed
30.	DAB Monobit 2	Good	**failed**	passed	passed	passed	passed	passed

5. Results

The test results based on P_{value} for all iterations are presented in table 1. It is impossible to present the exact P_{value} in a concise form, so the final judgment based on the calculated P_{value} is presented. A test was considered to have failed if at least one iteration of the test failed.

The generators *rand()*, *random256-glibc2* and */dev/urandom* were qualified as not random for a given level of significance. No basis was found to reject the hypothesis of randomness for the generators *ran1*, *triple pendulum* and *ANU QRNG*.

Table 2 shows the results of calculating the final Kolmogorov-Smirnov consistency test for the obtained P_{values}.

Table 2.
Kolmogorov-Smirnov consistency test for the obtained P_{values}.

Generator	**Kolmogorov-Smirnov uniformity test** P_{value}	**Result**
rand()	$6.18 \cdot 10^{-23}$	**failed**
ran1	0.40815	passed
random256-glibc2	0.19808	passed
/dev/urandom	0.06175	passed
triple pendulum	0.19757	passed
ANU QRNG	0.03800	passed

6. Summary and discussion

The obtained results confirm that algorithmic pseudorandom number generators are generally of low quality. This was particularly evident for the *rand()* generator from the standard C/C++ library. The *random256-glibc2* and */dev/urandom* generators using system interrupts performed slightly better. This may seem quite unexpected, as it is a generator that does not use a specific mathematical algorithm. However, if you consider the fact that in the operating system many interrupts are executed in a cyclic manner, you can expect a certain repetitiveness that will manifest itself in the generated numbers. The only algorithmic generator that passed all the tests positively is the *ran1* generator. The tests for the *triple pendulum* generator were also positive, which confirms that the chaotic nature of this classical phenomenon is so high that statistical tests are unable to find determinism. The *ANU QRNG* quantum generator, as expected, passed the test positively.

It remained to answer the question whether, based on the obtained results, we are able to indicate which generator is a truly random generator, e.g. a quantum generator. For this purpose, the Kolmogorov-Smirnov test was performed, the results of which are presented in table 2. The value of this test determines the uniformity of the distribution of all P_{value} values. The Kolmogorov-Smirnov test itself, from the Python library scipy.stats, provides a result in

the form of a P_{value} specifying the probability that the two distributions are identical. It is expected that this value should be as high as possible. Which would indicate a good distribution of values with respect to the theoretical uniform distribution. As can be seen from the above results in table 2, the *ANU QRNG* quantum generator received a lower score than other deterministic generators. Thus, the result did not confirm the high fit of the two distributions, which does not prejudge the non-randomness of a given generator. It follows that we are not able on the basis of statistical hipotesis testing with the Dieharder battery to determine additionally the quality of a given generator, but only to determine whether there are no basis to reject the generator as non-random. Statistical testing only has the character of a negative criterion. In the absence of basis to reject the hypothesis of randomness, statistical testing does not provide a mechanism to account for how positive the tested sequence of numbers was. Thus, a quantum generator in a positive way is not distinguishable in statistical testing from deterministic generators.

References

1. *ANU QRNG*. Australian National University. Available online https://qrng.anu.edu.au/, 4.03.2022.
2. Awrejcewicz, J., Kudra, G. (1999). *Nonlinear dynamics of a triple physical pendulum*. 2nd National Conference, Methods and Computer Systems in Scientific Research and Engineering Design, pp. 231-236.
3. Bays, C., Durham, S. (1976). Improving a Poor Random Number Generator. *ACM Transactions on Mathematical Software*, *Vol. 2, No. 1*.
4. Bobrowski, D. (1986). *Probabilistyka w zastosowaniach technicznych.* Warszawa: WNT.
5. Botha E., Qi, G. (2013). Analysis of the Triple Pendulum as a Hyperchaotic System. *Physics*.
6. Brown, R.G., *Dieharder: A Random Number Test Suite.* Retrieved from: https://webhome.phy.duke.edu/~rgb/General/dieharder.php, 4.03.2022.
7. *Class Ran1-Random-Number-Generator*. Available online https://cl-variates.common-lisp.dev/documentation/cl-variates-package/class-ran1--random--number--generator.html, 4.03.2022.
8. Hotoleanu, D., Cret, O., Suciu, A., Gyorfi, T., Vacariu, L. (2010). *Real-time testing of true random number generators through dynamic reconfiguration.* 13th Euromicro Conf. on Digital System Design: Architectures, Methods and Tools. IEEE, pp. 247-250.
9. Jacak, J., Jacak, W., Donderowicz, W., Jacak, L. (2020). Quantum random number generators with entanglement for public randomness testing. *Scientific Reports, vol. 10, no. 164*.

10. Jacak, M., Jacak, J., Jóźwiak, P.P., Jóźwiak, I.J. (2016). Quantum cryptography: Theoretical protocols for quantum key distribution and tests of selected commercial QKD systems in commercial fiber networks. *International Journal of Quantum Information, vol. 14, no. 2, 1630002*.
11. Jacak, M., Jóźwiak, P.P., Niemczuk, J., Jacak, J. (2021). Quantum generators of random number. *Scientific Reports, vol. 11, 16108*.
12. Jacak, M., Melniczuk, D., Jacak, J., Jóźwiak, I.J., Gruber, J., Jóźwiak, P.P. (2015). Stability assessment of QKD procedures in commercial quantum cryptography systems versus quality of dark channel. *International Journal of Quantum Information, vol. 13, no. 8, 1550064*.
13. Jacak, W. (2020). *Quantum nano-plasmonics*. Cambridge University Press, ISBN 9781108777698.
14. Jian, Y., Ren, M., Wu, E., Wu, G., Zeng, H. (2011). Two-bit quantum random number generator based on photon-number-resolving detection. *Review of Scientific Instruments*.
15. Kałuski, J. (2012). Logika podejmowania decyzji. Podejmowanie decyzji w aspektach logiki klasycznej i logiki kwantowej. Gliwice: *Zeszyty Naukowe Politechniki Śląskiej, no. 1873*, pp. 191-219.
16. Knuth, D.E. (1997). *The Art of Computer Programming, Vol. 2*. New York: Addison Wesley.
17. L'Ecuyer, P. (2004). *Random number generation*. Berlin: Center for Applied Statistics and Economics, Humboldt – Universitaet Berlin.
18. Małachowski, T. (2021). *Analiza porównawcza metod generowania liczb pseudolosowych*. (Master's thesis) Wrocław: Politechnika Wrocławska, Wydział Informatyki i Zarządzania.
19. Marsaglia, G. (1996). *Diehard: a battery of tests of randomness*. Retrieved from: http://stat.fsu.edu/geo/diehard.html, 5.03.2022.
20. Marsaglia, G. (2003). *Seeds for random number generators. Commun. ACM*. Retrieved from: https://doi.org/10.1145/769800.769827, 5.03.2022.
21. Nouar, C., Guennoun, Z. (2020). A Pseudo-Random Number Generator Using Double Pendulum. *Applied Mathematics & Information Sciences, 14*. pp. 977-984.
22. Park, S.K., Miller, K.W. (1988). Random Number Generators: Good ones are hard to find. *Computing Practices, Comm. Of the ACM*, *Vol. 31, No. 10*.
23. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S. (2010). *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.* USA: National Institute of Standards and Technology.
24. Stachowiak, T., Okada, T. (2006). A numerical analysis of chaos in the double pendulum. *Chaos, Solutions and Fractals, 29(2).* Elsevier, pp. 417-422.

25. Suresh, V.B., Antonioli, D., Burleson, W.P. (2013). *On-chip lightweight implementation of reduced NIST randomness test suite.* IEEE Int. Symposium on Hardware-Oriented Security and Trust (HOST). IEEE, pp. 93-98.
26. Symul, T., Assad, S.M., Lam, P.K. (2013). Real time demonstration of high bitrate quantum random number generation with coherent laser light. *Appl. Phys. Lett., 98*, 231103.
27. Vascowa, A., Lopez-Ongil, C., Jimenez-Horas, A., San Millan, E., Entrena, L. (2010). *Robust cryptographic ciphers with on-line statistical properties valida*tion. 16th Int. Symposium on On-line Testing Symposium (IOLTS). IEEE, pp. 208-210.