COMPARISON OF THE RESULTS OBTAINED BY PSEUDO RANDOM NUMBER GENERATOR BASED ON IRRATIONAL NUMBERS

PhD. Dimitrievska Ristovska V., Prof. PhD. Bakeva V.
Faculty of Computer Science and Engineering- Ss. Cyril and Methodius University, Skopje, Macedonia,

vesna.dimitrievska.ristovska@finki.ukim.mk

verica.bakeva@finki.ukim.mk,

Abstract: *Pseudo-random number generators* (*PRNG*) *based on irrational numbers are proposed elsewhere. They generate random numbers using digits of real numbers which decimal expansions neither terminate nor become periodic and practically their decimal expansion has infinite period. Using that algorithm, we generate sequences of random numbers and then we check their randomness with statistical tests from Diehard battery. Our main idea is to check is there a difference in the randomness of the generated sequences if digits of any irrational non- transcendental number* (*like* $\sqrt{2}, \sqrt{3}, \sqrt{5}, \ldots$) *are used versus the case when digits of a transcendental number* (*like* π *or e*) *are used. In our experiments we use about* $3 \cdot 10^7$ *digits of a given non-periodic irrational or transcendental number. Many experiments were done and all generated sequences by proposed PRNG based on irrational numbers passed the Diehard tests very well. We may conclude that there is not a significant difference in the randomness of the generated sequences in the both cases* (*irrational non-transcendental versus irrational transcendental number*).

Keywords: PRNG, IRRATIONAL NUMBERS, TRANSCENDENTAL NUMBERS, STATISTICAL TESTS, DIEHARD BATERY, RANDOMNESS

1. Introduction

Pseudo-random number generator (PRNG) is an algorithm which generates a long sequence of numbers $r_1, r_2, \ldots$ which are elements of a given set of numbers and the distribution of generated numbers $r_1, r_2, \ldots$ is supposed to be uniform.

A sequence of obtained random numbers $r_1, r_2, \ldots$ should have two important properties: uniformity (i.e., they are equally probable everywhere) and independence (i.e., the current value of a random variable does not depend on the previous values).

In practice, we cannot construct an ideal PRNG, since the way we are building the mechanism is not a random one, but in fact it is completely determined by an initial value. This affects the uniformity and independence of the produced sequences and $r_1, r_2, \ldots$ and that is why the word "pseudo" is used and we have to measure the randomness of the obtained sequences.

A good random number generator should have some additional qualities as large period and small order computational complexity.

This paper is organized as follows. In Section 2 we give a background and overview of related works. In Section 3 we explain basic ideas for construction of our generator of pseudo random numbers [2], basic principles for usage of statistical tests from Diehard battery and then we present the algorithm for the generator. The obtained results are given in Section 4. In Section 5, some conclusions are made.

2. Background and overview of related works

In this section we present some historical facts recall on L'Ecuyer (2017) in [4] about PRNGs which use irrational numbers.

The inspiration for using successive digits of π, *e* or any other transcendental number in order to generate a random number sequence is an old idea.

For example, Metropolis et al. (1950) in [6] succeeded to compute 2000 decimals of π and *e* and confirmed that these sequences pass elementary statistical tests. This testing was extended to the first 10000 decimals by Pathria (1962) (in [7]) and to 100000 decimals by Esmenjaud-Bonnardel (1965) (in [3]), and all of these sequences very well pass elementary statistical tests.

Till now, many sequences of digits of π have been obtained and tested and many papers have discussed this idea. The world record in 2016 was 22 459 157 718 361 decimal digits of π, computed in about four months by Peter Trueb using an algorithm of Chudnovsky and Chudnovsky (1989) (in [1]), Bellard's formula, and the Y-Cruncher multi-threaded software (Yee, 2017) (in [9]). However, to give good reason that the successive digits of π (or any other given irrational number) in a given base *b* can be taken as random sequence, it should be good to know that this sequence of digits is uniformly distributed in base *b*, i.e., that each of the *b* possible digits appears with frequency 1/*b* (on average) in the infinite sequence. For π, practical counting over several digits suggests that this is true, but there is no known proof of it.

However, the property of uniform distribution of the digits of any given irrational number is not sufficient; we need to have the uniform distribution of the pairs, triplets, and so on.

In the latest years, there are some trials to design a PRNG using digits of any irrational number since irrational numbers have decimal expansions that neither terminate nor become periodic, practically their decimal expansion has infinite period. In [8], Rogers and al. (2015) proposed an algorithm for pseudo-random 5-digit numbers using the digits of π and made some visual and statistical analyses for goodness of proposed generator.

In [2], the author proposed a new algorithm for generating pseudo-random numbers using digits of any irrational number. The randomness of obtained sequences of numbers is checked by some statistical tests and the test results are very well.

In this our paper, the main work is: using algorithm proposed in [2], to check is there a difference in the randomness of the generated sequences, if digits of any irrational non- transcendental number (like $\sqrt{2}, \sqrt{3}, \sqrt{5}, \ldots$ or the golden ratio $\varphi = \frac{1+\sqrt{5}}{2}$) are used versus the case when digits of a transcendental number (like π or *e*) are used.

3. PRNG based on digits of irrational numbers

3.1 The main idea in the algorithm– *n*-tuples

We will explain the ideas for designing a PRNG using digits of an irrational number ([2]), by example digits of π. In this example, we will use sequential *n*-tuples, for example 10-tuples of digits from the decimal expansion of π. Using some database we will take *l*

digits of π. In the next step, decimal 10-digit number from every 10-tuple is generated.

If we take the obtained 10-digit *number* (which is obtained directly from the 10-tuple) then its value is in the range $0 <= number <10^{10}$ and we need to check if the *number* is greater than *max* (maximal allowed generated number). If it is true, we have to omit the obtained *number* and continue with checking the next 10-tuple. The 10-tuples are taken without overlapping.

But, we will improve the previous idea if we scale the obtained 10-digit *number* (which is obtained directly from the 10-tuple) from the range $0 <= number <10^{10}$ in the range $0 <= number < 2^{32}$. In this way, there is no need to check if the scaled number is greater than *max*. Note that if the sequence of generated number is uniform then the scaled sequence will be uniform on the set $\{0,1,\ldots,max\}$.

These steps from the last idea will be repeated until we obtain l numbers.

3.2 Input parameters and the algorithm

In order to produce different sequences, each time when we started generating of a new sequence, we must initialize the beginning pointer to an arbitrary digit of the chosen irrational number. The position of the beginning pointer will be an input parameter. Also, the input parameter will be the length n of digits (n-tuples) for generating of each number in the sequence (in the previous example, we choose $n = 10$). Let stress that we compute the digits of any irrational number using package *Mathematica*.

Algorithm

[1] Choose an irrational number which digits will be used for generating random numbers.

[2] Set the length n of digits for generating of each number in the sequence, the position s of the beginning pointer (the first digit where the generating starts), the length l of generated sequence and the maximum *max* of the generated numbers.

[3] Let *counter*=0.

[4] Until *counter* <=l do

[4.1] Use slice size of n digits to generate a number r.

[4.2] Scale $r \leftarrow \left[\frac{r}{10^n-1} \cdot max\right]$.

[4.3] Put pointer position $s \leftarrow s + n$.

[4.4] *counter* = *counter* + 1.

We will notice that software realization of this algorithm and many experiments were done using package *Mathematica*.

3.5. Diehard tests

Nowadays there are a lot of tests for randomness and all of them measure the difference between the generated pseudo-random sequences and the theoretically supposed ideal random sequence. We say that a PRNG passes a test if the random sequences produced by that PRNG pass the test with a probability near to 1. We can classify PRNGs depending of the tests they have passed. So, for obtaining a better classification we should have many different tests.

Over several years, George Marsaglia [5] has developed Diehard tests as a battery of statistical tests for measuring the quality of a random number generator. This battery was published in 1995. It consists of 15 statistical tests, and it is a comprehensive set of statistical tests for PRNG and serves as some kind of litmus for checking and certification of PRNG. If a PRNG passes Diehard statistical tests, then it can be used in deeper scientific researches.

The Diehard battery consists of Birthday Spacings Test, Overlapping 5-Permutation Test (OPERM-5), Binary rank tests, 31× 31 Binary Matrix, 32 × 32 Binary Matrix, 6 × 8 Binary Matrix, Bitstream Test, Test OQSO (Overlapping quadruples sparse occupancy test), Test DNA, Count the 1's Test for specific bytes, Parking test, Minimum Distance Test, 3D Spheres Test, Squeeze Test, Overlapping Sums Test, Runs Test and Craps Test.

We will note that the most of the tests in Diehard return a p-value, which should be uniform on [0,1) if the input file contains truly independent random bits. Those p-values are obtained by $p = F(X)$, where F is the assumed cumulative distribution function of the sample (random variable X) – often normal. But that assumed F is just an asymptotic approximation, for which the fit will be worst in the tails. Therefore $p < 0.025$ or $p > 0.975$ means that the PRNG has "failed the test at the 0.05 level".

4. Results obtained from Diehard tests and discussion

Diehard tests have requirements with precise format of the numbers whose randomness they test. Explicitly, the file of the numbers should be a binary file of a hexadecimal integer nonnegative numbers with approximately 11 MB size. There should be ten numbers in each row, about 2 870 000 numbers in the file and the maximum number in the file should be $max = 2^{32} - 1$.

As we mentioned previously, some of Diehard tests give p-value, and some of them are performed several times and the result from these tests is the ratio of the number of passed tests and the total number of tests. Therefore, we presented the results in separated tables depends on the kind of test output.

In the next tables we will present some of the obtained results of Diehard tests applied to the sequences generated by our proposed algorithm in [2].

In Table 1, we present the percentage of passed Diehard test for 14 sequences generated by our PRNG using different irrational numbers or same irrational number with different initial pointer s or with different initial length n. The bold line in the table separates the sequences obtained from the digits of non-transcendent irrational numbers from them obtained from the digits of transcendent irrational numbers. Note that almost all sequences pass more than 90% of the Diehard test. Exception is only the sequence obtained using the digits from the sin 1, where the percentage of passed tests is between 80%and 90%, but it is satisfactory.

Table 1 *Success of Diehard tests*

Irrational number	Seq. number	n	s	Time for sequence generation (in sec.)	Success of Diehard tests
φ	Seq. 1	10	1	630	91 %
$\sqrt{2}$	Seq. 2a	10	2	670	99 %
$\sqrt{2}$	Seq. 2b	10	1	678	98 %
$\sqrt{3}$	Seq. 3a	10	6	636	97 %
$\sqrt{3}$	Seq. 3b	12	2	653	92 %
$\sqrt{7}$	Seq. 4a	10	6	654	99 %
$\sqrt{7}$	Seq. 4b	10	3	662	91 %
sin 1	Seq. 5a	10	2	351	85 %
sin 1	Seq. 5b	10	4	478	87 %
sin 1	Seq. 5c	9	4	529	86 %
π	Seq. 6	10	8	485	94 %
e	Seq. 7a	10	2	586	90 %
e	Seq. 7b	10	1	577	94%
ln 2	Seq. 8	10	3	627	95 %

In Table 2 and Table 3, we present the results from Diehard tests obtained from sequences generated from the digits of non-

transcendent numbers. In Table 2, we give the results of Diehard test which output is *p*-value and in Table 3, the results when the output is the ratio of the number of passed tests and the total number of tests. The red (bold) values in the tables mean that the sequence does not pass the corresponding test.

Table 2: *Results from Diehard tests applied on the sequences generated by PGNG when irrational number is non-transcendental. Obtained p-values are presented*

Seq. name	Seq. 1	Seq. 2a	Seq. 2b	Seq. 3a	Seq. 3b	Seq. 4a	Seq. 4b
Irr. number	φ	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{3}$	$\sqrt{3}$	$\sqrt{7}$	$\sqrt{7}$
	p-value						
Birthday Spacings test	0.10	0.87	0.37	0.40	0.23	0.15	0.34
OPERM-5	0.15 0.14	0.82 0.12	0.08 0.57	0.91 0.80	0.29 0.70	0.31 0.72	0.93 0.49
Binary-31 test	0.34	0.85	0.78	0.32	0.68	0.32	0.73
Binary-32 test	0.79	0.36	0.32	0.86	0.64	0.72	0.46
Binary-6x8 test	0.89	0.971	0.54	0.82	0.83	0.57	0.09
Count-Stream test	0.55 0.68	0.91 0.28	0.64 0.68	0.24 0.23	0.36 0.28	0.52 0.50	0.23 0.70
Parking test	0.48	0.30	0.73	0.30	0.18	0.38	**0.977**
Minim. Distance test	**0.99**	0.09	0.82	0.57	**0.004**	0.51	0.32
3D Spheres	0.88	0.96	0.74	0.92	0.28	0.47	0.15
Sqeeze test	**0.004**	0.71	0.14	0.45	0.53	0.96	0.17
O-SUM test	0.42	0.31	0.50	0.03	0.30	0.48	0.26
Run test	0.12 0.80 0.39 0.86	0.85 0.69 0.19 0.27	0.96 0.14 0.34 0.61	0.12 **0.02** 0.82 0.40	0.47 0.68 0.78 0.40	0.80 0.52 0.87 0.91	0.65 0.31 0.82 0.56
Craps test	0.49 0.81	0.92 0.24	0.26 0.41	0.24 0.03	0.39 0.92	0.75 0.30	**0.99** 0.24

Table 3: *Results from Diehard tests applied on the sequences generated by PGNG when irrational number is non-transcendental. No. of passed tests / No. of total tests are presented*

Seq. name	Seq. 1	Seq. 2a	Seq. 2b	Seq. 3a	Seq. 3b	Seq. 4a	Seq. 4b
Irr. number	φ	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{3}$	$\sqrt{3}$	$\sqrt{7}$	$\sqrt{7}$
	No. of passed tests / total tests						
Bit stream test	20/20	19/20	20/20	19/20	20/20	20/20	20/20
OPSO	23/23	22/23	18/23	21/23	17/23	21/23	22/23
OQSO	26/28	28/28	28/28	28/28	25/28	25/28	28/28
DNA test	31/31	28/31	28/31	31/31	30/31	30/31	31/31
Count Bytes test	23/25	23/25	22/25	23/25	24/25	24/25	24/25

From the last two tables, we can conclude that the generated sequences passed almost all Diehard tests.

In Table 4 and Table 5, we present the results from Diehard tests obtained from sequences generated from the digits of transcendent irrational numbers.

Table 4: *Results from Diehard tests applied on the sequences generated by PGNG when irrational number is transcendental. Obtained p-values are presented*

Seq. name	Seq. 5a	Seq. 5b	Seq. 5c	Seq. 6	Seq. 7a	Seq. 7b	Seq. 8
Irr. number	sin 1	sin 1	sin 1	π	e	e	ln 2
	p-value						
Birthday Spacings test	0.40	0.05	0.17	0.56	0.03	0.08	0.24
OPERM-5	**0.98** 0.94	0.05 0.73	0.20 **0.99**	0.68 0.32	0.95 0.27	0.19 **0.99**	0.57 0.49
Binary-31 test	0.33	0.66	0.41	0.77	0.49	0.81	**0.99**
Binary-32 test	0.60	0.61	0.62	0.65	0.72	0.32	0.64
Binary-6x8 test	0.18	0.22	0.89	0.55	0.31	0.16	0.14
Count-Stream test	0.40 0.26	**0.01** 0.34	0.87 0.03	0.43 0.31	0.90 0.45	0.80 0.51	0.80 0.04
Parking test	**0.006**	0.19	**0.98**	0.71	0.27	0.34	0.69
Minim. Distance test	0.92	0.03	0.55	**0.01**	0.35	0.86	0.72
3D Spheres	0.49	0.09	**0.99**	0.86	0.78	0.46	0.15
Sqeeze test	0.14	0.84	0.61	0.27	0.87	0.66	0.85
O-SUM test	**0.99**	**0.003**	0.36	0.31	0.61	0.52	0.82
Run test	0.53 0.08 0.97 0.52	0.25 0.93 0.39 **0.013**	0.76 0.34 0.84 0.21	0.10 0.61 0.13 0.73	0.42 0.74 0.97 0.48	0.36 0.28 0.39 0.71	0.25 0.65 0.44 0.51
Craps test	0.15 0.35	0.05 0.58	0.71 0.60	0.45 0.67	**0.99** **0.99**	0.86 0.83	0.46 0.64

Table 5: *Results from tests in Diehard battery, when irrational number is transcendental. Results are given with No. of passed tests / No. of total tests.*

Seq. name	Seq. 5a	Seq. 5b	Seq. 5c	Seq. 6	Seq. 7a	Seq. 7b	Seq. 8
Irr. number	sin 1	sin 1	sin 1	π	e	e	ln 2
	No. of passed tests / total tests						
Bit stream test	19/20	19/20	19/20	16/20	18/20	20/20	19/20
OPSO	19/23	21/23	20/23	20/23	21/23	21/23	23/23
OQSO	27/28	28/28	25/28	28/28	27/28	27/28	24/28
DNA test	30/31	30/31	29/31	31/31	30/31	26/31	31/31
Count Bytes test	20/25	24/25	23/25	25/25	23/25	23/25	24/25

Analyzing the results from Table 4 and Table 5, we can conclude that the sequences generated from the digits of transcendent irrational numbers also passed almost all Diehard tests. As we concluded from Table 1, exception is only the sequence obtained using the digits from the sin 1, where the results are a little

bit worse, but these sequences also passed more of the considered tests.

5. Conclusions

In this paper, we generate sequences using PRNG based on digits of different irrational numbers. Our goal is to check if the kind of irrational numbers (non-transcendent or transcendent) has influence to the randomness of generated sequences. Our expectations were that the transcendent numbers will give better results than non-transcendent ones, but the results reject our expectations. They confirm that almost all tested irrational numbers are good for using in our PRNGs and there is not a significant difference in the randomness of the generated sequences in the both cases.

Acknowledgements

This work was partially financed by the Faculty of Computer Science and Engineering at the "Ss. Cyril and Methodius" University in Skopje.

References

1. Chudnovsky D., G. Chudnovsky The computation of classical constants, in: Proceedings of the National Academy of Sciences of the USA 86, 1989, pp. 8178--8182

2. Dimitrievska Ristovska, V. Pseudo random generator based on irrational numbers, in: Trajanov D., V. Bakeva (eds.): ICT-Innovations 2017, Data Driven Innovations, Conference Web Proceedings, 2017, pp. 105--113

3. Esmenjaud-Bonnardel M. ´Etude Statistique des D´ecimales de Pi, Revue Francaise de Techniques Informatiques 8 (4), 1965, pp. 295--306

4. L'Ecuyer P. History of uniform random number generation, Chan W. K. V., D'Ambrogio A., Zacharewicz G., Mustafee N., Wainer G., and Page E., eds.: DIRO, GERAD, and CIRRELT, Proceedings of the 2017 Winter Simulation Conference, 2017

5. Marsaglia G., W. W. Tsang Some difficult-to-pass tests of randomness, Journal of Statistical Software, Volume 7, Issue 3, 2002

6. Metropolis N., G. Reitwiesner, J. von Neumann Statistical Treatment of Values of First 2000 Decimals Digits of E and Pi Calculated on the ENIAC, Mathematical Tables and Other Aids to Computation 4, 1950

7. Pathria R. K. A Statistical study of randomness among the first 10000 digits of Pi, Mathematics of Computation 16, 1962, pp. 188--197

8. Rogers I., G. Harrell, J. Wang Using Pi digits to generate random numbers: A Visual and statistical analysis, Int'l Conf. Scientific computing | CSC'15 |

9. Yee A. J. Y-Cruncher—A Multi-Threaded Pi-Program, 2017