DOI: 10.18372/2225-5036.30.18613

METHODS OF CHOOSING A RANDOM NUMBER GENERATOR FOR MODELING STOCHASTIC PROCESSES

Yurii Shcherbyna[1], Nadiia Kazakova[2], Oleksii Fraze-Frazenko[3], Oleg Domaskin[4]

[1]National University «Odesa Law Academy»
[2]Odessa State Environmental University
[3, 4]Odessa National Economic University

Yuriy SHCHERBINA, candidate of technical sciences, associate professor
Date and place of birth: 1953, Vinnytsia region. Teplytsky district, village Petrashivka.
Education: Riga Higher Military Engineering School of the Missile Forces, 1975.
Position: associate professor of the Department of Information Technologies since 2021. *Scientific interests:* protection of automated information and communication systems, modeling of information flows in telecommunication systems, software of data processing systems.
Publications: more than 90 scientific publications, including scientific articles, monographs, textbooks, theses and materials of reports at conferences.
E-mail: shcherbinayura53@gmail.com.
Orcid ID: 0000-0003-3885-6747.

Nadiya KAZAKOVA, doctor of technical sciences, associate professor
Date and place of birth: 1979, Odesa, Ukraine.
Education: Odesa National Academy of Communications named after OS Popov, 2001.
Position: Head of the Department of Information Technologies from 2021.
Scientific interests: methods and means of technical information protection, protection of state secrets, design of complex information protection systems, methods and models of information protection, technical channels of information leakage.
Publications: more than 200 scientific publications, including scientific articles, monographs, textbooks, theses and materials of reports at conferences.
E-mail: kaz2003@ukr.net.
Orcid ID: 0000-0003-3968-4094.

Oleksiy FRAZE-FRAZENKO, candidate of technical sciences, associate professor
Date and place of birth: 1982, Lviv, Ukraine.
Education: Kharkiv Military University, 2004.
Position: associate professor of the Department of Information Technologies since 2019.
Scientific interests: methods and means of technical information protection, protection of state secrets, design of complex information protection systems, technical channels of information leakage.
Publications: more than 80 scientific publications, including scientific articles, monographs, textbooks, theses, and conference materials.
E-mail: frazenko@gmail.com.
Orcid ID: 0000-0002-2288-8253.

Oleg DOMASKIN, candidate of technical sciences, associate professor
Date and place of birth: 1968, Odesa, Ukraine.
Education: Odesa Polytechnic National University, 1995.
Position: Head of the Information Technology Center since 2013.
Scientific interests: methods and means of technical protection of information and communication systems, design of complex information protection systems, mathematical methods and models of information protection.
Publications: more than 60 scientific publications, including scientific articles, monographs, textbooks, theses and materials of reports at conferences.
E-mail: o.domaskin@oneu.edu.ua.
Orcid ID: 0000-0001-7756-9631.

Abstract. *Modern computer modeling is an important stage in the design of control systems for the distribution of information flows in computer networks and in modern control systems for complex technological processes. The core of any computer model is a source of randomness, which should generate a uniformly distributed stream of random integers or real numbers. In addition to the uniformity of distribution, such a source must meet the requirements of economic use of computing system resources. An analysis of simple arithmetic generators is given and, based on it, it is shown that generators such as the Fibonacci sequence generator with a delay and the Xorshift generator proposed by J. Marsaglia are suitable as a generator for the needs of modeling stochastic processes, which are an alternative to the random number generators built into existing programming environment. On the basis of the conducted research, it was concluded that any unevenness of the numbers at the output of the generator chosen as a source of randomness significantly affects the quality of the process to be modeled, and because of this, the numerical flows from such generators should be additionally processed by methods extraction of that part of them that provides maximum randomness. The method of performing such extraction by "slicing" the input stream, the criteria used in this, and the results of its experimental research for the Xorshift128 generator are presented. A conclusion is made about the advantages of using simple and economical generators in a heap with post-processing procedures performed at the level of integers or real numbers. The results of the evaluation of the Xorshift generator, taking into account the methods described in the work, are given, and a conclusion is made about the feasibility of its use for the needs of modeling stochastic processes.*

Keywords: *Mersenne twister generator, Xorshift generator, inverse function method, Monte Carlo method, Pearson chi-square test, numerical flow post-processing, algorithm, method, nonlinear system, stability, forecasting, information technology.*

Introduction

Today, it is difficult to imagine computer simulation without the use of pseudorandom number generators. In the design of complex technical systems, the use of such generators makes it possible to perform their optimization and experimental evaluation.

Arithmetic generators of pseudo-random numbers (PRN), built on the basis of finite state machines and capable of creating sequences of numbers that can be used as truly random, occupy a special place. Despite the fact that they give periodic sequences, their periods are extremely large and they satisfy basic randomness tests.

Analysis of existing studies

Given that such generators can be easily implemented using simple and fast software procedures, they find their place in modeling using such methods as inverse function methods and Monte Carlo [1].

The main requirement for PRS generators intended for modeling stochastic processes is their symmetry at the binary level - the equal probability of zeros and ones in their composition. The absence of such equal probability is called bias. Any PRN generators, regardless of whether they are built on the basis of real physical processes or on the basis of software algorithms, have a fixed bias. To overcome this drawback, such generators usually consist of two parts: the first is a source of randomness, and the second is a corrector that compresses the original stream of symbols to ensure the highest possible probability (unbiasedness) of the original process [2]. In other words, the Random Source is always asymmetric and it is important to be able to effectively ensure the selection of unshifted bits from its source sequence. [3].

An additional requirement for computer modeling is the ability to repeatedly reproduce the implementation of a random process, which can only be provided by an algorithmic generator.

The vast majority of PRS generators built into the most common programming environments, which include the MT generator known as the Mersenne twister (MT) [4], the Xorshift generator [5] and the Linear congruent generator (LCG) [6], do not pass the distribution uniformity check and, without additional post-processing of the original sequence, do not provide satisfactory modeling. With this in mind, every time creating a computer model of a stochastic process, the developers must perform tests of the selected generator for its compliance with the set requirements.

Purpose and statement of the task

Taking into account the fact that common implementations of PRN generators do not meet the conditions put forward for modern computer modeling, and, in addition to the fact that their use requires a significant amount of computing resources of the system, the task of the work is to substantiate the principles of creating such a generator, which would be effective in terms of specific modeling requirements.

The main part of the study

The idea of computer modeling arose almost immediately after the appearance of electronic computing systems. Immediately, with this, the search for random number generators suitable for simulation needs began. Many outstanding mathematicians worked on this problem and proposed several simple algorithms for obtaining random numbers by arithmetic methods. Their analysis can be found in D. Knuth's work [7] " The Art of Computer Programming". The composition of such algorithms should include the method of mean squares, linear congruence algorithms, Fibonacci algorithms and some others. All of them produced sequences of numbers that seemed random, but in terms of the uniformity of their distribution, they did not meet the requirements of modeling. Many attempts were made to improve them, but they did not have significant success.

It is certainly possible to create an arithmetic algorithm that would ensure an even distribution of numbers at the output of the generator, but this problem is solved at the expense of its extreme complication. Examples of such algorithms are almost all known cryptographic generators used in block cipher systems such as DES and the like.

Since the requirements for generators used in simulation are not as high as the requirements for cryptographic generators, the uniformity of the distribution of numbers at their outputs is usually achieved by additional post-processing procedures. At the moment, the following methods are used to ensure it [8]:

1. Ad hoc simple correctors;
2. Whitening with hash functions;
3. Extractor algorithms;
4. Resilient functions.

Considering the fact that one of the requirements for computer modeling remains the saving of computing resources, advantages are given to simple correctors and extractors. The essence of their work is the readiness to sacrifice part of the bits of the original sequence to increase the randomness of the original stream of numbers.

An example of such a post-processing method is the algorithm developed by von Neumann in [9]. He proposed combining each subsequent pair of bits received from independent sources according to the principle: if the bits match (00 or 11), then such bits are canceled, the bit combination 01 corresponds to the 0th output bit, and the 10th output bit corresponds to the combination. The maximum efficiency of such an algorithm is an average of 4 input bits per 1 output bit.

In [10] Trevisen gave a formal definition of the sequence transformation at the output of the PRN generator. He introduced the concept of minimum entropy, which characterizes the unevenness of the distribution of the quantity X in the range $\{0,1\}^n$ where n is the binary combination at the output of the source. It is assumed that the entropy will be maximum if all the initial combinations are equally likely. Otherwise, the entropy will be lower. If the entropy of the output flow is at least k, then for each $x \in \{0,1\}^n$ the condition $\Pr[X = x] \le 2^{-k}$ is fulfilled. The extractor must perform a transformation of the stream X into a nearly uniform stream. To quantify the output flow, the concept of statistical difference ϵ between two random variables X and Y in the range $\{0,1\}^n$ is introduced, which is defined as:

$$|p[T(X) = 1]| - |p[T(Y) = 1]| \le \epsilon. \qquad (1)$$

In the general case, the (k, ϵ)-extractor transforms a stream of random variables X into an almost uniform stream according to the rule:

$$Ext: \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m, \qquad (2)$$

when the random variable X has minimum entropy k, and $\{0,1\}^t$ is a set of t-bit binary combinations forming a uniformly distributed variable U_t. The general principle of operation of the randomness extractor is shown (fig. 1).

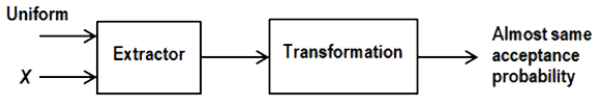

Fig. 1. The mechanism of the extractor

In order to be able to compare the methods of correcting the numerical flow, the concept of displacement of the output distribution of numbers is introduced in [11] to assess the unevenness of the distribution of numbers at the output of the PRN generator. The binary sequence of numbers $x_1, x_2, \dots x_i$ is treated as non-uniform with offset. Taking into account that the bits x_i are independent, the offset value is defined as:

$$e = \frac{1}{2}\big(P(x_i = 1) - P(x_i = 0)\big), \qquad (3)$$

where $P(x_i = 1) = \frac{1}{2} + e$ і $P(x_i = 0) = \frac{1}{2} - e$.

If the number at the output of the PRN generator can take one of m values, then for a good generator the probability of each of them should be equal to $1/m$. In the case of a binary sequence, this value should be equal to 0.5, and the value of e should approach zero.

The general characteristics of the methods used to create generators of initial unshifted values [1] can be formulated as follows:

– they do not use all bits of the original sequence;

– the bit selection algorithm must be implemented in an effective way, from the point of view of saving computing resources;

- for such generators, there must be a mathematical justification of their properties.

The first step in choosing a generator for creating a computer model should be to choose a PRN generator. It is desirable that it be simple enough. A good example of a simple generator can be a generator built on the basis of using Fibonacci numbers [7]. His work can be described by the following expression:

$$X_n = (X_{n-24} + X_{n-55}) \bmod m, \quad n \ge 5, \qquad (4)$$

where m is an even number, and $X_0, \dots, X_{54}$ are arbitrary integers, and not all of them are even. The length of the sequence period at the output of such a generator is $2^{q-1}(2^{55} - 1)$, where q is the bit rate of the microprocessor register, and q. Other experimentally determined good coefficients for X_i are given in [7].

Given the need to initially fill the generator's memory with the 55 seed numbers and the fact that 24 and 55 are Fibonacci numbers, its output sequences are called delayed Fibonacci sequences.

Such generators were used at the end of the 20th century, they were considered the best sources of randomness, but they did not pass the new tests created at that time, which indicates the exhaustion of the means of creating such a generator capable of ensuring the necessary uniformity of the distribution of output numbers without their additional processing. The Fibonacci sequence generator works faster than other similar generators and has the longest repetition period. Moreover, it allows implementation of floating-point operation modes.

At one time, George Marsaglia gave a mathematical justification to the vast majority of iterative generators described by D. Knuth in [7], which could be used as a source of randomness, after appropriate refinement [12]. Based on this analysis, he proposed an efficient generator called Xorshift [13], built using simple shift and addition operations. Its iterative generator uses a set of numbers Z,, the inverse function f over the set Z,, and an initial number $z_0 \in Z$.

The numbers at the output of such a generator are formed according to the principle:

$$f(z), f^2(z), f^{32}(z), \dots, \qquad (5)$$

where $f^2(z)$ means $f(f(z))$, $f^3(z)$ means $f(f^2(z))$ etc. Usually, the set Z is the set of all possible 32-bit numbers that are tuples $x_1, x_2, \dots, x_m$, and f is a function that transforms the current tuple in the next.

If f is a mutually unique function over Z, and the initial number z, from which the generator starts its work, is randomly selected from the set Z, the stream of initial random values $f(z)$ will also be uniformly distributed.

The formation of the original sequence of numbers must be ensured by the use of transformations of the type $f(z), f^2(z), \ldots$ by means of uniform random selections of the number z from Z. Such a sequence will have all signs of randomness.

In search of a computationally efficient PRN generator, George Marsaglia proposed a large number of modifications to the Xorshift generator. Basically, such a generator is a certain number of linear feedback registers (LFSR), the configuration of which is determined by the choice of generating polynomials of relatively low degree.

The architecture of generators of the Xorshift type is focused on the use of a 32- or 64-bit integer as an element of a vector space in a binary field modulo 2. It is the use of elementary computer operations (addition and shifts) that ensures the simplicity and efficiency of implementing the necessary transformations over numbers in the vector space linear space.

The essence of the Xorshift algorithm is that it uses the set of all nonzero 1×32 binary vectors from Z, and f is treated as a linear transformation over Z represented by a nondegenerate binary matrix T of size 32×32. Taking this into account, we can state that the sequence of numbers at the output of the generator will look like yT, yT^2, $yT^3, \ldots, ,$ only if the order of T is equal to $2^{32} - 1$ in the group of non-degenerate binary matrices of size 32×32 and the sequence has a period of $2^{32} - 1$ [14].

In [13], Marsaglia showed that a simple method of forming the matrix product yT can be implemented if the order:

$$T = (I + L^a)(I + R^b)(I + L^c), \qquad (6)$$

where L is the matrix that affects the left shift by one. In the C language, this operation looks like $y \text{ ^ } = (y \ll 1)$. Accordingly, the matrix yL^a realizes the shift $y \text{ ^ } = (y \ll a)$. Given that matrix R is a transposed matrix L, its use implements a right shift by one unit $y \text{ ^ } = (y \gg 1)$. This means that (6), for a random 32-bit number from Z, makes it possible to obtain each subsequent number in the sequence yT, yT^2, yT^3, ... Thus, to obtain the maximum period, matrices that implement the three types of shifts described above $y \text{ ^ } = y \ll 13$; $y \text{ ^ } = y \ll 17$; $y \text{ ^ } = y <$ 5 ..

The implementation of the described algorithm in the C language may look like this:

```
x = 123456789
y = 362436069
z = 521288629
w = 88675123

t = x ^ ((x << 11) & 0xFFFFFFFF);
x = y;
y = z;

z = w;
w = (w ^ (w >> 19)) ^ (t ^ (t >> 8));
```

After starting the generator, the initial numbers x, y, z and w are set, which determine the internal state of the generator. Each subsequent number $w = \{N_0, N_1, \ldots, N_{31}\}$ is formed as a combination of bits of x and the previous value of w as shown in the following code snippet. After that, the $y \to x$, $z \to y$ and $w \to z$ are shifted. According to Marsaglia, it is precisely these shifts that increase the "randomness" of lower order numbers. He substantiated in detail the choice of three numbers $[a, b, c], a < c$ for which the binary matrix of type (6) has the period $[2^{32} - 1]$, and also listed combinations of numbers $[a, b, c]$, which are the best, with from the point of view of minimal use of computing resources. Thus, its generator, Xorshift, due to its simplicity and unpretentiousness to resource costs, can be considered as one of the main candidates for use as a source of randomness in computer simulations.

The main problem of modeling is that regardless of the method of formation of the model of the original stochastic process, the unevenness of the numbers at the output of the PRN generator, which is a source of randomness, is completely transferred to the process, which is the goal of modeling. Although the requirements for the accuracy of modeling in comparison with the real process do not exceed 5÷10 percent, the latest modification of the generator proposed by Marsaglia does not meet such requirements (fig. 2).

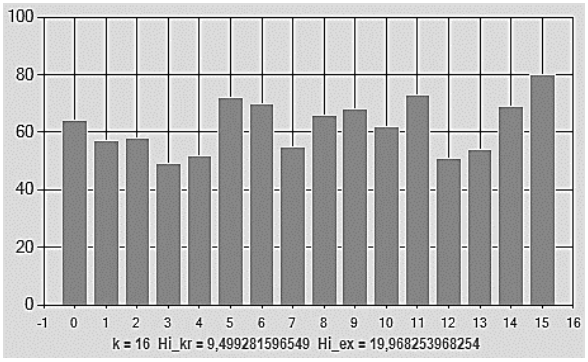

Fig. 2. Histogram of the distribution of PRN obtained using the function Xorshift128

It should be noted here that we are not talking about a binary output stream, but about a stream converted to a sequence of integers or real numbers. So, Figure 1 shows a 16-segment histogram of a sequence of 1000 positive random 4-byte numbers at the output of the Xorshift128 generator.

Studies conducted using the χ^2-Pearson test [14] show that the uniformity index, taking into account 15% accuracy, usually significantly exceeds the critical permissible value.

In favor of the Xorshift128 generator, the fact that due to the use of logical and bit operations provides "whitening" and "mixing" of the higher and lower digits of the generated numbers, just as it is done in cryptographic generators.

Despite the positive properties described above, the Xorshift generator in its known modifications requires additional post-processing. Applying the number selection method described in [15], [16] to the sequence of numbers at its output, the shift in their distribution uniformity can be reduced. The essence of this method is that, given the type of distribution, the sample size N, and the number of intervals k, the number of numbers falling into each interval of the histogram is calculated. For uniform distribution, these values must coincide and be equal to the value of N_i/k. The mathematical expectation of the value $m_i$falling into each segment of the histogram

$x_{min} \le x_i < x_{max}$ is equal to the value $m_i = (x_{min} + x_{max})/2$, and the sum of the numbers S_i, which should fall into the i-th interval, will be approximately equal to the value of $S_i^* = N_i^* m_i$. If the sum of the numbers that actually fell into the i-th segment of the histogram S_i, it will differ from the expected value S_i^* every time. Thus, the "extra" numbers that fall into each i-th segment of the histogram will be filtered out, and the displacement of the distribution of the output flow will decrease, which is confirmed by tests, as can it be seen (fig. 3)

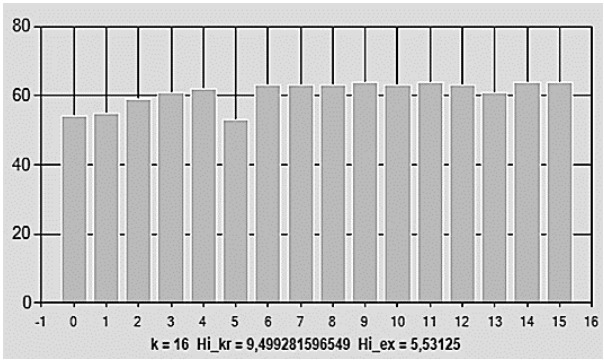

Fig. 3. Histogram of the distribution of numbers from the output of the Xorshift128 generator after post-processing

Tests of the described generators and their post-processing method using the χ^2-criterion show that in most cases, without additional post-processing, the χ^2 indicator exceeds the critical value ($\chi^2 < \chi^2_{кр}$). At the same time, performing additional processing of the original numerical stream provides a source of randomness suitable for modeling stochastic processes.

Conclusions. The problem of computer modeling is that even a sufficiently acceptable bit sequence, from the point of view of distribution uniformity, does not preserve this uniformity when transformed into a numerical stream of integers or real numbers. Because of this, it is necessary to look for ways of such an additional transformation of numerical sequences that would take into account exactly this aspect of the problem.

The analysis of known simple arithmetic generators of pseudo-random numbers has shown that, if good post-processing methods are applied to them, they can be successfully used to simulate stochastic processes, and are an alternative to generators built into most known programming environments. Today, developers' efforts to create modern post-processing methods are focused on their wide use in various scientific fields, including cryptographic algorithms. Given that the requirements for the uniformity of the distribution in the numerical sequences intended for the needs of modeling differ from the requirements for the sequences obtained at the output of cryptographic generators, economic PVC generators and post-processing methods based on a simple extraction of "the most random" part of the original numerical stream. This problem is best solved at the level of the numerical flow, which is directly used in the simulation process.

References

[1]. Smile Markovski, Danilo Gligoroski, Ljupco Kocarev. Unbiased Random Sequences from Quasigroup String Transformations. Part of the Lecture Notes in Computer Science book series (LNSC, volume 3557). URL: https:// www.iacr.org /archive/fse2005/35570161 /35570161.pdf.

[2]. Dichtl, M.: Bad and good ways of post-processing biased physical random numbers. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 127-152. Springer, Heidelberg (2007) URL: https://link.springer.com/content/pdf/10.1007/978-3-540-74619-5_9.pdf.

[3]. Smile Markovski, Danilo Gligoroski Ljupco Kocarev. Unbiased Random Sequences from Quasigroup String Transformations. International Workshop on Fast Software Encryption FSE 2005: Fast Software Encryption pp 163-180 URL: https://www.iacr.org/archive/fse2005/35570161/35570161.pdf.

[4]. Makoto Matsumoto and Takuji Nishimura. 1998. Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-Random Number Generator. ACM Trans. Model. Comput. Simul. 8, 1 (Jan. 1998), 3-30. https://doi.org/10.1145/272991.272995 URL: https://dl.acm.org/doi/pdf/10.1145/272991.272995.

[5]. François Panneton and Pierre L'Ecuyer. 2005. On the Xorshift Random Number Generators. ACM Trans. Model. Comput. Simul. 15, 4 (Oct. 2005), 346-361. https://doi.org/10.1145/1113316.1113319. URL: https://web.archive.org/web/20210126143346/http://www.iro.umontreal.ca/~lecuyer/myftp/papers/xorshift.pdf.

[6]. Niederreiter H. Quasi-Monte Carlo methods and pseudo-random number. doi: https://doi.org/10.1090/S0002-9904-1978-14532-7. URL: https://www.ams.org/journals/bull/1978-84-06/S0002-9904-1978-14532-7/S0002-9904-1978-14532-7.pdf.

[7]. D. E. Knuth. The Art of Computer Programming, Volume 2: Seminumerical Algorithms, 3rd. ed., Boston, Mass, USA: Addison-Wesley, Longman Publishing, Addison-Wesley, Reading, Mass, 1998. URL: http://publ.lib.ru/ARCHIVES/K / KNUT_Donal%27d_ Ervin/ Knut_D.E._I skusstvo _ programmirovaniya._T.2. (2001). [djv-fax].zip.

[8]. Yurii Shcherbyna, Nadiia Kazakova, Oleksii Fraze-Frazenko. The Mersenne Twister Output Stream Postprocessing. INTERNATIONAL SCIENTIFIC AND PRACTICAL CONFERENCE. September 2021. pp 277-285 URL: https://ceur-ws.org/Vol-3200/paper39.pdf.

[9]. J. von Neumann. Various techniques for use in connection with random digits. Applied Math Series, Notes by G. E. Forsythe, in National Bureau of Standards, Vol. 12, 36-38, 1951. URL: https://mcnp.lanl.gov/pdf_files/nbs_vonneumann.pdf.

[10]. Trevisan L. Extractors and Pseudorandom Ge nerators 1999. Journal of the ACM. URL: http://theory.stanford.edu/~trevisan/pubs/extractor-full.pdf.

[11]. Siew-Hwee Kwok, Yen-Ling Ee, Guanhan Chew, Kanghong Zheng, Khoongming Khoo, Chik-How Tan. A Comparison of Post-Processing Techniques for Biased Random Number Generators. WISTP 2011: Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication. pp 175–190. URL: https://link.springer.com/content/pdf/10.1007/978-3-540-74619-5_9.pdf.

[12]. George Marsaglia. Random Number Generators. 2003. DOI 10.22237/jmasm/1051747320 URL: https://digitalcommons.wayne.edu/jmasm/vol2/iss1/2/.

[13]. George Marsaglia. Xorshift RNGs. 2003. DOI: 10.18637/jss.v008.i14. URL: https: // www.researchgate.net/publication/5142825_Xorshift_RNGs.

[14]. Shcherbyna, Y. Using the Xorshift generator to simulate stochastic processes // Przetwarzanie, transmitsja i bezpieczeństwo informacji // monografia / Yurii SHCHERBYNA, Nadiia KAZAKOVA, Oleksii FRAZE-FRAZENKO; Bielsko - Biała: Wydawnictwo Naukowe Akademii Techniczno-Humanistycznej w Bielsku-Białej, 2022. pp 313-144. ISBN 978-83-67652-00-1 DOI: https://doi.org/10.53052/9788367652001.

[15]. The Mersenne Twister Output Stream Postprocessing. Shcherbyna, Y., Kazakova, N., Fraze-Frazenko, O. CEUR Workshop https://ceur-ws.org/Vol-3200/paper39.pdf., 2021, 3200, pp. 265-273.

[16]. Yevseiev, S., Melenti, Y., Voitko, O., Hrebeniuk, V., Korchenko, A., Mykus, S., Milov, O., Prokopenko, O., Sievierinov, O., Chopenko, D. Development of a Concept for Building a Critical Infrastructure Facilities Security System (2021) Eastern-European Journal of Enterprise Technologies, 3, pp. 63-83. DOI: 10.15587/1729-4061.2021.233533.

УДК 621.391.25

Щербина Ю., Казакова Н., Фразе-Фразенко О., Домаскін О. Методи вибору генератора випадкових чисел для моделювання стохастичних процесів

***Анотація**. Сучасне комп'ютерне моделювання – це важливий етап проектування систем управління розподіленням інформаційних потоків в обчислювальних мережах та у сучасних системах управління складними технологічних процесами. Ядром будь-якої комп'ютерної моделі є джерело випадковості, яке повинно формувати рівномірно розподілений потік випадкових цілих або дійсних чисел. Крім рівномірності розподілення, таке джерело повинно задовольняти вимогам економічного використання ресурсів обчислювальної системи. Наведено аналіз простих арифметичних генераторів і, на його основі, показано, що у якості генератора для потреб моделювання стохастичних процесів підходять такі генератори, як генератор послідовності Фібоначчі з запізненням та запропонований Дж. Марсальєю генератор Xorshift, які є альтернативою генераторам випадкових чисел, вбудованих в існуючі середовища програмування. На основі проведених досліджень зроблено висновок про те, що будь-яка нерівномірність чисел на виході генератора, обраного у якості джерела випадковості, суттєво впливає на якість процесу, який підлягає моделюванню, і, через це, числові потоки від таких генераторів мають бути додатково оброблені методами екстракції тої їх частини, яка забезпечує максимальну випадковість. Наведено методику виконання такої екстракції шляхом "проріжування" вхідного потоку, критерії, які при цьому використовуються, та результати його експериментального дослідження для генератора Xorshift128. Зроблено висновок про переваги використання простих і економічних генераторів в купі з процедурами постоброблення, що виконується на рівні цілих або дійсних чисел. Наведено результати оцінки роботи Xorshift генератора з урахування описаних в роботі методик та зроблено висновок про доцільність його використання для потреб моделювання стохастичних процесів.*

***Ключові слова**: моделювання, лінійний конгруентний генератор, генератор Вихор Мерсенна, Xorshift генератор, метод зворотної функції, метод Монте-Карло, критерій хі-квадрат Пірсона, постоброблення числового потоку, алгоритм, метод, персональні дані, особиста інформація, фазовий портрет, нелінійна система, стійкість, запізнення, конфіденційність, прогнозування, інформаційні технології.*

Щербина Юрій Володимирович, кандидат технічних наук, доцент, доцент кафедри інформаційних технологій Національного університету «Одеська юридична академія».

Yurii Shcherbina, candidate of technical sciences, associate professor, associate professor of the department of information technologies of the National University "Odesa Law Academy".

Казакова Надія Феліксівна, доктор технічних наук, професор, завідувачка кафедри інформаційних технологій Одеського державного екологічного університету.

Nadiia Kazakova, doctor of technical sciences, professor, head of the department of information technologies of the Odesa State Environmental University.

Фразе-Фразенко Олексій Олексійович, кандидат технічних наук, доцент, доцент кафедри інформаційних технологій Одеського державного екологічного університету.

Oleksii Fraze-Frazenko, candidate of technical sciences, associate professor, associate professor of the department of information technologies of the Odesa State Environmental University.

Домаскін Олег Михайлович, кандидат технічних наук, доцент, Керівник центру інформаційних технологій Одеського національного економічного університету

Oleh Domaskin, candidate of technical sciences, associate professor, Head of the information technology center in Odesa National Economic University.

Отримано *2 березня 2024 року*, затверджено редколегією *1 квітня 2024 року*