23. PROBLEM OF RANDOM NUMBER GENERATION AND SOLUTIONS

Kotova K.A.

Belarusian State University of Informatics and Radioelectronics
Minsk, Republic of Belarus

Kaspiarovich N.G. – Senior Lecturer

The general information about random number generation is presented in the paper. The reason why it is a problem is explained and the methods used for the solution are examined. The advantages and disadvantages of each method are provided.

Random numbers are used in many spheres of people's life, particularly in game industry for roulette imitation, in science for hypothesis testing and creating complicated physical, technological, social and economic systems, in lotteries, in cryptography and information security for data encryption and decryption, in bank payments and trade systems [1].

The approaches to random number generation comprise hardware, software and their combination [2]. The first method generates numbers, using something random such as the movements of a mouse pointer on a screen, or the pauses in keystrokes, or it can be based on a real physical process with certain characteristics such as quantum-based devices (Samsung cellphone with quantum-based random number generator can serve as an example). It should be mentioned that, although the time and material required for this method are rather costly, it provides a higher degree of randomness which means the generated sequence consists of true random numbers.

The software approach is focused on special mathematical transformations, often in a recurrent form, which makes it possible to obtain deterministic numerical sequences with characteristics close to truly random processes. These sequences are called pseudo-random. The term comes from the fact that a random variable is a parameter that, as a result of an experiment, takes one of the many values, and these values cannot be accurately predicted. However, computer systems are deterministic. They are characterized by a strictly defined set of states. The number of such states can be very large, but finite. The aforementioned feature means that the sequence of random numbers generated by a computer is just a simulation, or a pseudo-random process [1].

The main differences between a pseudo-random number generator (PRNG) and a random number generator (RNG) are that the PRNG, firstly, uses a single initial value, secondly, it has a predictable relationship between the members of the sequence, and thirdly, there is a certain period of the generated sequence, after which the generator loops, while the RNG always generates a random number, having a high-quality random value at the beginning [1].

Despite the fact that hardware generators promise better statistical characteristics and a higher degree of randomness, they still inherent a number of disadvantages: potentially high time and material costs for design, installation and configuration compared to software PRNG; slower random number generation than with software implementation of PRNG; the impossibility of reproducing previously generated sequences of numbers, hence PRNGs are widely used.

The first algorithmic method for obtaining uniformly distributed pseudo-random numbers was proposed by John von Neumann in 1946. It is called **the middle-square method**. The main idea is as follows: the method starts with an n-digit number called the seed, which becomes the first element of the pseudorandom sequence, designated x_0. Squaring x_0 yields a number with $2n$ digits (possibly including leading zeros). The middle *n* digits of the squared value are selected, and the resulting number is called x_1. The process can be repeated indefinitely, each time squaring x_{i+1} and taking the middle digits as the value of x_{i+1}. If all goes well, the sequence of x values looks random, with no obvious pattern or ordering principle.

The main disadvantages including the presence of a correlation between the elements, a short period, the problem which was referred to as the "zero mechanism" (once the middle digits became zero,

the generator would continue to produce zero outputs) suggest that this method is unreliable and interesting only in the historical aspect. And, this opinion is widespread. But, there are also ways to troubleshoot these problems. In particular, a new implementation of the method using the Weyl sequence [3] overcomes the "zero mechanism", keeps the generator, running through a long period, prevents repeating cycles. Modern 64-bit computing architecture provides fast processing speed and the fact that the method is nonlinear gives the generator an advantage in quality of data over linear-based generators.

Linear congruential (LCG) method [2] was proposed by D.H. Lehmer in 1949. The main idea is represented in calculating the members of a linear recurrent sequence by the absolute value (modulo) of some natural number m, using formula 1:

$$x_{k+1} = (ax_k + c)\ mod\ m \tag{1}$$

where a – multiplier, c – increment, m – some integer constants.

The result depends on the choice of the starting number (seed) x_0 and the parameters above, poor combination of which may generate a sequence with a small period.

The advantages of LCG are its simplicity and high-rate performance. It is incorporated while solving problems of modeling and mathematical statistics, in lottery or entertainment industry, however, they cannot be recommended to be used for cryptographic purposes, since cryptanalysts have learned how to restore the entire PRNG sequence from several values.

Further development of the LCG idea is represented in linking the next element output of the sequence with two instead of one previous elements.

The most famous example of a sequence in which the next element depends on the two preceding, is the Fibonacci sequence [2]. The Lagged Fibonacci generator **(LFG)** uses formula 2:

$$S \equiv S_{n-j} * S_{n-k} (mod\ m), 0 < j < k \tag{2}$$

the operator $*$ can be either addition (+), subtraction (−), multiplication (×) or X-OR ($\oplus$).

The period of such a generator, as a rule, is greater than that of an LCG one with the same value of the modulus n. It provides high performance, but is not cryptographically secure. LFGs are used as auxiliary blocks of cryptographically stable PRNGs. They also serve as the basis for some crypto algorithms.

Mersenne Twister (MT) was developed in 1997 by Japanese scientists Makoto Matsumoto and Takuji Nishimura. It is based on the properties of simple Mersenne numbers and is famous for its period length which is the number of values before the sequence repeats, and which is rather massive ($2^{19937-1}$) and hence solves the problem of a hidden pattern [4]. Besides, MT was faster than other statistically valid generators at the time of its introduction. However, the method has been critisized for using up too many bits for the states which has a strong effect on the overall performance. Also, it is not cryptographically stable but is widely used and implemented in many programming languages. Its popularity is proven by the fact that due to the increase in processor performance 64-bit and 128-bit versions of MT were invented. They were proposed by Takuji Nishimura in 2000 and 2006 respectively.

Blum Blum Shub (BBS) generator [5] was created by Lenore Blum, Manuel Blum and Michael Shub in 1986. It uses the form of formula 3:

$$x_{n+1} = x_n^2 (mod\ m) \tag{3}$$

where x_0 – a random seed, the value of m – equal to $p * q$ (Blum integer), the values of p and q – both congruent to $3\ mod\ 4$.

For each step, we extract some of the information from x_{n+1} (typically the least significant bit). The most interesting fact about this method for practical purposes is that, if there is the need to get the n number of the sequence, it is possible to calculate all the previous $n-1$ numbers. The x_n can be immediately obtained by formula 4:

$$x_n = x_0^{(2^n) mod ((p-1)*(q-1)) mod\ m} \tag{4}$$

The security of the method involves the difficulty in factorizing m. It is said that if m is large enough, it does not even need to be kept in secret: as long as m is not factorized, no one can predict the exit. The problem of the method is that it is slow, however, it is the strongest proven random number generator.

It should be noted that around 1970 an American research mathematician, an expert on pseudo-random number generators, Robert R. Coveyou published an article entitled “Random Number Generation Is Too Important to Be Left to Chance”. And, the issue of random number generation is still topical and the development of new, more advanced methods is still relevant.

References:

Introduction to Random Number Generation [Electronic resource]. – Mode of access: https://www.learncpp.com/cpp-tutorial/introduction-to-random-number-generation/. – Date of access: 02.03.2023.

Types of generators. Linear Congruential method. Lagged Fibonacci generator [Electronic resource]. – Mode of access: https://books.ifmo.ru/file/pdf/2474.pdf. – Date of access: 04.03.2023.

Middle-Square Weyl Sequence [Electronic resource]. – Mode of access: https://arxiv.org/pdf/1704.00358.pdf. – Date of access: 04.03.2023.

Mersenne Twister generator [Electronic resource]. – Mode of access: http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/ARTICLES/mt.pdf. – Date of access: 07.03.2023.

Blum Blum Shub generator [Electronic resource]. – Mode of access: https://cyberleninka.ru/article/n/realizatsiya-generatora-blum-blum-shub-i-issledovanie-ego-osnovnyh-harakteristik/viewer. – Date of access: 09.03.2023.