Improving Statistical Properties of Number Sequences Generated by Multiplicative Congruential Pseudorandom Generator

Mieczysław Jessa

***Abstract*—A new method of improving the properties of number sequences produced by a multiplicative congruential pseudorandom generator (MCPG) was proposed. The characteristic feature of the method is the simultaneous usage of numbers generated by the sawtooth chaotic map, realized in a finite-state machine, and symbols produced by the same map. The period of generated sequences can be significantly longer than the period of sequences produced by a multiplicative congruential pseudorandom generator realized in the same machine. It is shown that sequences obtained with the use of the proposed method pass all statistical tests from the standard NIST statistical test suite v.1.8.**

***Index Terms*—pseudorandom generators, shuffling, combined generators, sequences of symbols, statistical properties**

I. INTRODUCTION

PSEUDORANDOM number sequences are used in many fields of science. Every programming language provides a pseudorandom number generator that produces a sequence of nonnegative integers $\{p_0, p_1, ...\}$ with integer upper bound b, and then uses $\{x_0 = p_0/b, x_1 = p_1/b, ...\}$ as an approximation of an independent and identically distributed (i.i.d.) sequence from unit interval $I = (0,\ 1)$. In almost all programming languages, numbers $\{p_0, p_1, ...\}$ are generated by a multiplicative congruential pseudorandom generator (MCPG) of the form

$$p_n = (ap_{n-1}) \bmod b \quad n = 1, 2, ... \tag{1}$$

The properties of generated sequences depend strongly on the choice of two parameters: a multiplier a and a modulus b. To obtain maximal-length sequences (m-sequences), modulus b has to be a prime number and multiplier a has to be a primitive element modulo b [1]–[3]. Because the value for b is usually determined by the number of bits used to encode numbers, the statistical properties of generated sequences depend on the choice of the multiplier. In general, the choice of a "good" a is not simple and the number of multipliers generating number sequences with good statistical properties is quite small [1], [2].

In this paper, we propose a new method of improving properties of m-sequences produced by generator (1). The method exploits a sequence of symbols produced by the sawtooth chaotic map, implemented in computer in the modular arithmetic. The sequence is used to shuffle the output stream of MCPG. The same stream is shuffled in different ways, producing different sequences. The obtained sequences are combined into a single sequence which forms the output stream. The generation of successive numbers is slightly slower but we obtain additional control parameters (degrees of freedom) which can be used for improving the statistical properties of generated sequences, including the possibility of increasing the period of the sequences. The statistical properties of output streams are verified with the use of the standard NIST statistical test suite v.1.8 [4].

This paper is organized as follows. Section II describes the method and the period of generated sequences. The results of the statistical tests from the standard NIST statistical test suite v.1.8, applied to sequences produced by the MCPG and to sequences produced by the proposed generator, are presented in Section III. Conclusions are drawn in Section IV.

M. Jessa is with the Poznan University of Technology, Faculty of Electronics and Telecommunications (e-mail: mjessa@et.put.poznan.pl).

II. THE METHOD

One of the characteristic features of many pseudorandom number generators is that numbers obtained in the iterative procedure are simultaneously the output of the generator. MacLaren and Marsaglia suggested that the output stream of linear congruential pseudorandom number generator should be shuffled by using another, perhaps simpler, generator to obtain sequences with better statistical properties [2], [3]. The first generator produces sequences which fill a table and the second one is used to read off elements from this table. Because a single pseudorandom number generator can be used to generate independent pseudorandom numbers, it can also be used to shuffle itself [2], [3]. This method, using only one generator, was applied by Gebhardt to improve the statistical properties of number sequences produced by the Fibonacci generator [5]. In 1976 Bays and Durham proposed a method of using a single generator to shuffle number sequences produced by the MCPG, known as RANDU [6]. Although shuffling can improve the statistical properties of sequences produced by the MCPG, it is insufficient to ensure that all statistical tests from the standard NIST statistical test suite v.1.8 could be passed for many a. Another approach uses combined generators. In such type of generator the output streams of two or more generators (called source generators) are combined, usually with the use of modulo 2 operation, into a single stream. The output sequence of the combined generator has significantly longer period and better statistical properties than the output sequences of the source generators. Examples of combined generators can be found, e.g., in [1], [3]. To achieve a positive result of all tests from the NIST test suite, we must use many source generators, which is numerically inefficient. In this section, we introduce a new method for generating many source streams by a single MCPG. The generator is derived from the sawtooth chaotic map implemented in a finite-state machine in the modular arithmetic. The benefit is that we can combine many source streams into a single sequence without significantly decreasing the speed of producing pseudorandom numbers.

Let S_λ denote the sawtooth map, named also the Rényi map, the Bernoulli shift, or the Bernoulli map. Map S_λ transforms the unit interval $I = [0,\ 1) \subset X,\ X \equiv R$ into itself and has the following form

$$S_\lambda(x) = \lambda x \mod 1, \tag{2}$$

where λ is a real number. Computing successive values of expression

$$s_n = \lfloor \alpha x_n \rfloor\ ,\ \alpha \geq 2\ ,\quad n = 1, 2, ..., \tag{3}$$

where α is an integer and $x_n = \lambda x_{n-1} \mod 1$, we obtain a sequence $\{s_n\}$ of integer numbers. Numbers s_n can be regarded as indices of subintervals containing x_n and obtained as the result of partitioning I into α disjoint, equal-sized subintervals $I_j,\ \ j = 0, 1, 2, ..., \alpha - 1$, covering the whole set I. Through assigning a unique number (symbol) from set $A_\alpha = \{0, 1, ..., \alpha - 1\}$ to every I_j, the macroscopic behavior of the dynamical system (S_λ, I) can be studied. This macroscopic dynamics is called symbolic dynamics. It is known that symbolic sequences may be treated as truly random sequences in many aspects [7]–[10]. Assuming integer λ and rational $x_0 = (p_0)/(q_0)$, where $0 < p_n < q_0$, we obtain that [11]

$$\begin{cases} s_n = \lfloor \alpha x_n \rfloor \\ x_n = \frac{p_n}{q_0} \qquad\qquad n = 1, 2, \ldots \ . \\ p_n = \lambda \cdot p_{n-1} \bmod q_0 \end{cases} \tag{4}$$

Because in a finite-state machine the number of bits encoding the values of all variables is limited to l, where l is finite, expression (4) can be written as

$$\begin{cases} s_n = \lfloor \alpha \cdot x_n \rfloor \\ x_n = \mathrm{trunc}_l \left(\frac{p_n}{q_0}\right) \qquad n = 1, 2, \ldots\ , \\ p_n = \lambda p_{n-1} \bmod q_0 \end{cases} \tag{5}$$

where trunc_l denotes the truncation operation, leaving l the most significant bits of quotient $(p_n)/(q_0)$. If $\alpha = 2^k,\ 1 \leq k \leq l$, then sequence $\{s_n\}$ consists of numbers encoded by the k most significant bits of x_n. If additionally $q_0 = 2^l$ or $q_0 = 2^l - 1$, these bits are the same as the most significant bits of p_n (see [11] for examples). Then (5) is reduced to

$$\begin{cases} s_n = \mathrm{trunc}_k(p_n) \\ p_n = \lambda p_{n-1} \bmod q_0 \end{cases} . \tag{6}$$

The second formula in (6) describes the multiplicative congruential pseudorandom generator (1) with $a = \lambda$ and $b = q_0$. For $\alpha = 2^k,\ 1 \leq k \leq l$ and $q_0 = 2^l$ or $q_0 = 2^l - 1$, sequence $\{s_n\}$ is the same as the output sequence of the truncated multiplicative congruential pseudorandom generator. To improve the statistical properties of $\{p_n\}$, successive p_n are first written into Table T with L cells, addressed from 0 to $L-1$. Next, we read off K numbers $T_1, T_2, ..., T_K$ from T per one iteration of equation (6), where it is assumed that $L \geq \alpha K$. The addresses of $T_1, T_2, ..., T_K$ depend on s_n. Numbers $T_1, T_2, ..., T_K$ are treated as vectors encoded by l bits. The elements of K vectors are summed modulo 2 and added modulo 2 to current number p_n, denoted for clarity as T_0, forming a single vector U_n. Its elements can encode an integer number from interval $(0,\ 2^l)$ or a real number from unit interval $I = (0,\ 1)$. The pseudocode of an algorithm proposed for producing $\{U_n\}$ has the following form:

Algorithm 1 Algorithm CPRNG

Initialization:
Choose k, $p_0 \in (0,\ q_0)$ and the size L of Table T;
Write p_0 into the first cell of Table $T, i.e.\ T[0] := p_0$;
for $n := 1$ to $L-1$ **do**

$$\begin{cases} p_n := \lambda p_{n-1} \bmod q_0,\ n = 1, 2, ...L-1 \\ T[n] := p_n \end{cases} \tag{7}$$

end for
Computations:
for $n := 1$ to N **do**

$$\begin{cases} p_{n+L-1} := \lambda p_{n+L-2} \bmod q_0 \\ j := n \bmod L,\ L \geq \alpha K,\ \alpha = 2^k,\ 1 \leq k \leq l \\ T[j] := p_{n+L-1} \\ s'_{n+L-1} := 1 + trunc_k(p_{n+L-1}) \\ U_n := T[j] \oplus T[(j + s'_{n+L-1}) \bmod L] \\ \qquad \oplus \cdots \oplus T[(j + K s'_{n+L-1}) \bmod L] \end{cases} \tag{8}$$

end for

In (8) it is that $s'_{n+L-1} = 1 + s_{n+L-1}$. The combined pseudorandom number generator CPRNG repeatedly uses the "bit stripping", known from the shuffling algorithms of Gebhard or Bays and Durham (see p. 10 in [2]). Numbers p_n written into T can be regarded as digits encoding a certain number p, written in the fixed-point number system with base q_0. If $\{p_n\}$ is a random sequence, then all sequences composed of digits chosen from digits encoding p are independent [2]. The addresses of numbers $T_0, T_1, .., T_K$ differ by a constant value s'_{n+L-1}. Numbers s'_{n+L-1} are the elements of symbolic sequence $\{s_n\}$ produced by chaotic S_λ and realized in computer in the modular arithmetic – shifted by unity. The same algorithm can be used for other values q_0 but symbols s'_{n+L-1} have to be computed from formula $s'_{n+L-1} = 1 + \mathrm{trunc}_k(p_{n+L-1}/q_0)$, i.e., they cannot be the most significant digits of p_{n+L-1} increased by 1. Changing the method of addressing Table T, we can obtain different combined generators.

The period m_u of sequence $\{U_n\}$ depends on the period m_p of sequence $\{p_n\}$ and the size L of Table T. Table T is filled with L elements of sequence $\{p_n\}$ during the *Initialization*. After $n = LCM(m_p, L)$ iterations of expression (8), where $LCM(m_p, L)$ is the least common multiple of numbers m_p and L, Table T is filled with the same numbers as after the *Initialization*. For $n > LCM(m_p, L)$, we obtain

$$U_{n+LCM(m_p,L)} = U_n. \tag{9}$$

For $n < LCM(m_p, L)$ Table T does not contain the same elements as during the *Initialization*. If some element U_n is repeated for $j = n$, where $n < LCM(m_p,\ L)$, it is not repeated for all n being a multiple of j, which results directly from the method of computing of U_n. Consequently, the period of $\{U_n\}$ cannot be smaller than $LCM(m_p, L)$. Changing the size L of Table T, we can influence the period of generated sequences. If L is relatively prime to m_p, the period of $\{U_n\}$ is L times greater than the period of m-sequence produced by the MCPG, implemented in the same finite-state machine.

III. The Results of NIST Statistical Tests

To verify the hypothesis that the statistical properties of $\{p_n\}$ can be improved by the proper choice of α, K, and L, the standard NIST statistical test suite v. 1.8 for cryptographic applications was applied. It contains 15 tests, designed for analyzing different statistical properties of generated sequences, turned into binary streams [4]. The goal of the tests is to detect non-randomness in binary sequences produced using random number or pseudorandom number generators. The tested sequences are composed of bits encoding successive U_n. The null hypothesis is that any sequence being tested is random. Associated with this null hypothesis is an alternative hypothesis, which, for the NIST tests, is that any tested sequence is not random. The tests search for deviations from the properties of truly random binary sequences in binary sequences produced by a source under test. If a binary sequence passes the tests, there is no reason to reject the null hypothesis.

The empirical results can be interpreted in many ways. In this paper two approaches proposed by NIST were used: (1) the examination of the proportion R of sequences that pass a statistical test and (2) the distribution of the so called P-values computed by software. In the first case, we find the proportion of sequences that pass a given test. The second approach, adopted by NIST, measures the distribution of P-values in interval $[0,\ 1]$ divided into ten equal-sized subintervals. The P-value is the probability (under the null hypothesis of randomness) that the chosen test statistic will assume values equal to or worse than the test statistic value observed when considering the null hypothesis. The P-value is frequently called the "tail probability". When the sequences are random binary sequences, the P-values obtained for these sequences have to be uniformly distributed in $[0,\ 1]$ [4]. As the result of applying a χ^2 test and an additional function, we obtain a new P-value (P_T), corresponding to the Goodness-of-Fit Distribution Test on the P-values obtained for an arbitrary statistical test (i.e. the P-value of the P-values). If $P_T \geq 0.0001$, then the sequences can be considered to be uniformly distributed. The details of computing P_T can be found in [4].

The statistical tests were performed on 1000 different sequences of length 10^6. The sequences were successive fragments of sequence $\{p_n\}$ or $\{U_n\}$, produced for the smallest λ for which $\{p_n\}$ was the m-sequence. Modulus q_0 was a prime number equal to $2^{31} - 1$ ($l = 31$) and p_0 was equal to unity. The size of Table T was constant during all experiments and equal to $L = 32$. Because the least common multiple of m_p and L is equal to 34359738336, the period m_u of $\{U_n\}$ is 16 times longer than the period $m_p = 2^{31} - 2 = 2\ 147\ 483\ 646$ of $\{p_n\}$ produced by the MCPG. The results of the standard NIST test suite performed for binary sequences, composed of bits encoding successive p_n, generated by MCPG with $\lambda = 7$, are shown in TABLE I. The results of the same tests for binary sequences, composed of bits encoding successive U_n, are presented in TABLE II. Parameter α was equal to 4. Numbers from TABLE II were obtained for the smallest K for which sequences produced by CPRNG passed all statistical tests.

TABLE I
THE RESULTS OF NIST TESTS FOR MCPG WITH $\lambda = 7$

Type of the test	R(> 0.981)	P_T(> 0.0001)	**Final result**
Block Frequency	**0.0000**	**0.00000**	fail
Serial*	**0.9780**	0.05642	fail
Approximate Entropy	**0.9750**	0.00711	fail
Linear Complexity	0.9900	0.7944	pass
Universal	**0.9120**	**0.00000**	fail
Overlapping Templates	**0.5490**	**0.00000**	fail
Non-overlapping Templates	**0.9640**	**0.00000**	fail
Cumulative Sums*	**0.9670**	**0.00000**	fail
Runs	0.9950	0.01570	pass
Longest Runs of Ones	**0.9640**	**0.00000**	fail
Rank	0.9880	0.43543	pass
Spectral DFT	**0.0000**	**0.00000**	fail
Random Excursions*	0.9836	0.07375	pass
Random Excursions Variant**	0.9800	0.01526	pass
Frequency	**0.9760**	**0.00000**	fail

*This test consists of several subtests: the worst result is shown.
**The minimum pass rate for this test for a standard set of parameters is approximately 0.978.

IV. Conclusion

A new method for improving the quality of a multiplicative congruential pseudorandom generator was proposed in this paper. The method uses symbols produced by the sawtooth map realized in a finite-state machine and numbers produced by a multiplicative congruential generator, obtained as the result of implementing the same map in the same machine in the modular arithmetic. Although the proposed algorithm improves the statistical properties of sequences produced by a known pseudorandom generator, it can be treated as a new generator, derived from a chaotic map. The basic weakness of this generator is the lack of theory which could simplify the choice of α, K and L. Simulation experiments, performed by the author for many λ and $q_0 = 2^{31} - 1$, show that it is always possible to choose relatively small K (of the order of 8) which yields sequences passing all tests from the standard NIST statistical test suite v. 1.8. The speed of producing $\{U_n\}$ with $\alpha = 4$, $L = 32$ and $K = 3$ is only about 25% smaller than the speed of producing $\{p_n\}$ on the same hardware and software platform.

Access to a pseudorandom generator producing long period number sequences that pass all NIST tests for many multi-

TABLE II
THE RESULTS OF NIST TESTS FOR CPRNG WITH $\lambda = 7$, $\alpha = 4$, K=3

Type of the test	$R(> 0.981)$	$P_T(> 0.0001)$	**Final result**
Block Frequency	0.9900	0.86288	pass
Serial*	0.9870	0.13728	pass
Approximate Entropy	0.9920	0.13112	pass
Linear Complexity	0.9920	0.68902	pass
Universal	0.9850	0.00737	pass
Overlapping Templates	0.9900	0.16170	pass
Non-overlapping Templates*	0.9820	0.02979	pass
Cumulative Sums*	0.9840	0.67661	pass
Runs	0.9860	0.04198	pass
Longest Runs of Ones	0.9930	0.89348	pass
Rank	0.9950	0.96019	pass
Spectral DFT	0.9880	0.26757	pass
Random Excursions*	0.9865	0.31094	pass
Random Excursions Variant**	0.9828	0.09676	pass
Frequency	0.9870	0.93900	pass

*This test consists of several subtests: the worst result is shown.
**The minimum pass rate for this test for a standard set of parameters is approximately 0.978.

pliers λ enables us to construct a high-speed pseudorandom generator with long periods of generated streams. The simplest method uses a field programmable gate array (FPGA). In this circuit, we implement r CPRNGs with different values of λ that work in parallel. In each step of generation, we obtain r pseudorandom numbers. Consequently, the speed of producing pseudorandom numbers increases r times. This property can be used in cryptography and in multi-core processors for fast generation of high-quality pseudorandom numbers with long periods.

REFERENCES

[1] P. Bratley, B. L. Fox, and L. E. Schrage, *A Guide to Simulation*. New York: Springer-Verlag, 1987, ch. 6.

[2] J. E. Gentle, *Random Number Generation and Monte Carlo Methods*. New York: Springer, 2003, ch. 1.

[3] D. E. Knuth, *The Art of Computer Programming*, 2nd ed. Addison Wesley, 1981, vol. 2, ch. 3.

[4] [online], http://csrc.nist.gov/rng/.

[5] F. Gebhard, "Generating pseudo-random numbers by shuffling a Fibonacci sequence," *Mathematics of Computation*, vol. 21, pp. 708–709, 1967.

[6] C. Bays and S. D. Durham, "Improving a poor random number generator," *ACM Trans. on Mathematical Software*, vol. 2, pp. 59–64, 1976.

[7] M. P. Kennedy, R. Rovatti, and G. Setti, *Chaotic Electronics in Telecommunications*. Boca Raton: CRC Press, 2000, ch. 3.

[8] L. Kocarev, G. Jakimoski, and Z. Tasev, *Chaos and Pseudo-Randomness in Chaos Control*, 2003, pp. 247–263.

[9] T. Kohda and A. Tsuneda, "Statistics of chaotic binary sequences," *IEEE Trans. Inf. Theory*, vol. 43, pp. 104–112, Jan. 1997.

[10] T. Stojanovski and L. Kocarev, "Chaos-based random number generators – Part I: Analysis," *IEEE Trans. Circuits Syst. I*, vol. 48, pp. 281–288, Mar. 2001.

[11] M. Jessa, "Designing security for number sequences generated by means of the sawtooth chaotic map," *IEEE Trans. Circuits Syst. I*, vol. 53, pp. 1140–1150, May 2006.

Mieczyslaw Jessa was born in Poland in 1961. He received the M.Sc. degree with honors from Poznan University of Technology in 1985 and the Ph.D. degree in 1992 from the same University. Since 1985 he has been employed at the Institute of Electronics and Telecommunications in Poznan. Now, he works with the Chair of Telecommunication Systems and Optoelectronics of the same University.

Initially, his research interest included phase-locked loops and PDH/SDH network synchronization. In the years 1995-1997 he was an expert of Polish Ministry of Communications in the field of digital network synchronization. His current research concerns randomness and pseudo-randomness, the applications of the chaos phenomenon, and mathematical models of systems evolution. He is the author or co-author of over one hundred journal and conference papers and fifteen patents.