数理解析研究所講究録 1240 巻 2001 年 26-34

THE FIRST RETURN TIME TEST OF PSEUDORANDOM NUMBERS

DONG HAN KIM

ABSTRACT. An algorithm for obtaining the probability distribution of the first return time R_n for the initial n-block with overlapping is presented and used to test pseudorandom number generators. The averages and the standard deviations of $\log R_n$ are computed theoretically and the Z-test is applied.

1. INTRODUCTION

We introduce a new method of testing PRNGs based on the first return time of the some fixed length blocks in a randomly generated binary sequence. The first return time is closely related to entropy, which is the central idea in the information theory founded by C. Shannon[13]. For a binary source it is defined to be the limit of $-\frac{1}{n}\sum_{i=1}^{2^n} p_i \log p_i$ as n increases to infinity where the p_i's are the relative frequencies of 2^n blocks of length n in a typical binary sequence generated by the source. Entropy measures the information content or the amount of randomness. In data compression the entropy measures the maximum compression rate. If there are more patterns, that is, less randomness in a given sequence, then it has smaller entropy and can be compressed more.

In this article the first return time in a random binary sequence is investigated. Consider a stationary ergodic binary process on the space of infinite sequence $(\{0,1\}^\infty, \mu)$, where μ is the shift invariant ergodic probability measure on the σ-field generated by finite dimensional cylinders. For each sample sequence x define the first return time by

$$R_n(x) = \min\{j \geq 1 : x_1 \dots x_n = x_{j+1} \dots x_{j+n}\}.$$

A.D. Wyner and Ziv[15] proved that $\frac{1}{n}\log R_n(x)$ converges to the entropy of the sequence in measure and Ornstein and Weiss[9] showed that the convergence is pointwise. Later, A.J. Wyner[16] discovered that for a stationary aperiodic Markov chain with entropy h the random variable $\frac{1}{n}\log R_n$ has approximately a normal distribution with mean

h in a suitable sense. For a sharp estimate of the convergence of the average of $\frac{1}{n}\log R_n$, see [2].

Kac's lemma[4] implies that $E[R_n|x_1^n = a_1^n] = 1/\mu(a_1^n)$ for any finite string a_1^n with $\mu(a_1^n) > 0$. Since Kac's lemma implies $E[R_n] = \#\{a_1^n \in A^n : \mu(a_1^n) > 0\}$, we have

$$\lim_{n\to\infty} \frac{1}{n} \log E[R_n] = \text{topological entropy.}$$

This implies that the algorithm using $\log R_n$ is much more efficient than the algorithm of R_n.

Since Maurer's work[8], the nonoverlapping first return time which corresponds to

$$R_{(n)}(x) = \min\{j \geq 1 : x_1 \dots x_n = x_{jn+1} \dots x_{jn+n}\},$$

has been investigated to be used in cryptography or in testing PRNGs. Nonoverlapping algorithm is relatively easier to analyze but overlapping method is more efficient and natural.

2. The probability distribution of the first return time

A block is a finite sequence of elements of A and an n-block is a block of length n. For an n-block $B = b_1 b_2 \cdots b_n$, we write $B_i^j = b_i b_{i+1} \cdots b_j, 1 \leq i \leq j \leq n$. Since the distribution of return time is different from block to block. We classify the blocks to each set of blocks have the same return time distribution.

Definition 1. Let B be an n-block. Suppose m satisfies $1 \leq m < n$ and

$$B_{m+1}^n = B_1^{n-m}.$$

The smallest such m is denoted by $\lambda_1(B)$, and the next smallest such m which is not a multiple of $\lambda_1(B)$ is called $\lambda_2(B)$, and we define $\lambda_k(B)$ by the smallest such m which is not a multiple of $\lambda_i(B)$ for every $i < k$. Let $\Lambda(B) = \{\lambda_1(B), \lambda_2(B), \dots\}$.

If $\lambda \in \Lambda(B)$ then

$$(B_1^\lambda B_1^\lambda \dots B_1^\lambda)_1^n = B,$$

or

$$B = \underbrace{\boxed{b_1 \dots b_\lambda}\ \boxed{b_1 \dots b_\lambda}\ \cdots\ \boxed{b_1 \dots b_\lambda}\ \boxed{b_1 \dots b_j}}_{n}$$

TABLE 1. The Expectation of R_n and $\log R_n$

Block	$\Lambda(B)$	E[R_n]	E[log R_n]	Var[log R_n]
00000000	1	256	4.122127	18.37019
00000001	$\emptyset$	256	7.299403	2.441935
00000010	7	256	7.273498	2.589157
00000100	6,7	256	7.219351	2.905512
00001000	5,6,7	256	7.106875	3.576236
00010001	4	256	7.055111	3.986235
00100001	5	256	7.183896	3.147559
00100010	4,7	256	7.031221	4.110117
00100100	3,7	256	6.717126	6.102838
01000001	6	256	7.244771	2.763759
01000010	5,7	256	7.158986	3.283393
01001001	3	256	6.738698	6.005312
01010101	2	256	6.015615	10.32028

for some $1 \leq j \leq \lambda$.

Table 2 shows $\Lambda(B)$ for some 8-blocks. For more of the definition of $\Lambda(B)$, see [1].

Lemma 2 ([1], Lemma 3). *If $\lambda_1(B) \leq n/2$, then $\lambda_i(B) > n - \lambda_1(B)$ for every $i \geq 2$ if $\lambda_i(B)$ exists.*

Definition 3. For a given n-block B, let $\mathcal{F}(B,k)$ be the set of k-blocks, $k \geq n$ defined by

$$\mathcal{F}(B,k) = \{C : C_1^n = B, C_{i+1}^{i+n} \neq B \text{ for any } i \geq 1\},$$

and let $\mathcal{S}(B,k)$ be the set of k-blocks, $k \geq 1$ defined by

$$\mathcal{S}(B,k) = \{C : (CB)_1^n = B, (CB)_{i+1}^{i+n} \neq B \text{ for any } i, 1 \leq i < k\}.$$

Example 4. Consider the case of $B =$ '010' and $k = 6$. The 6-block '010001' is not elements in $\mathcal{S}$('010',6) but in $\mathcal{F}$('010',6), since the 9-block '010001 010' has three '010' blocks (e.g. '0100$\underline{0101}$0'). Hence

$$\mathcal{F}(\text{'010'},6) = \{010000, 010001, 010011, 010110, 010111\},$$

$$\mathcal{S}(\text{'010'},6) = \{010000, 010011, 010110, 010111\}.$$

The following shows the relation between $\mathcal{F}(B,k)$ and $\mathcal{S}(B,k)$. Note that for $k \geq n$, $\mathcal{S}(B,k) \subset \mathcal{F}(B,k)$.

Lemma 5. *For all n-block B*

$$\mathcal{S}(B,k) = \mathcal{F}(B,k) \setminus \bigcup_{\lambda \in \Lambda(B)} \{C \in \mathcal{F}(B,k) : C^k_{k-\lambda+1} = B^\lambda_1\},$$

where the union is disjoint union. For $\lambda \in \Lambda(B)$ and $\ell(\lambda) = \max\{j \in \mathbb{N} : j\lambda < n\}$ we have

$$\{C \in \mathcal{F}(B,k) : C^k_{k-\lambda+1} = B^\lambda_1\} = \bigcup_{j=1}^{\ell(\lambda)} \{C \underbrace{B^\lambda_1 \cdots B^\lambda_1}_{j} : C \in \mathcal{S}(B, k - j\lambda)\}.$$

Proof. See [1], Lemma 6. □

Note that for any n-block B, we have $\lambda_i(B) > n/2$, $i \geq 2$ and $\ell = 1$ except $\lambda = \lambda_1(B)$.

Definition 6. Define $r_k(B)$ and $s_k(B)$ by

$$r_k(B) = \Pr(x_1 \ldots x_k \in \mathcal{F}(B,k)), \quad k \geq n,$$

$$s_k(B) = \Pr(x_1 \ldots x_k \in \mathcal{S}(B,k)). \quad k \geq 1.$$

From now on we consider i.i.d. processes. For i.i.d. processes we have

$$\Pr(x_1 \ldots x_n = B,\ R_n(x) > k - n) = r_k(B), \quad k \leq n$$

$$\Pr(x_1 \ldots x_n = B,\ R_n(x) = k) = s_k(B)\mu(B), \quad k \geq 1.$$

We can calculate the distribution of the first return time from the followings:

Proposition 7. *For i.i.d. processes, if $k > n$, we have*

$$r_k(B) = r_{k-1}(B) - \mu(B) s_{k-n}(B).$$

Let $m = |\Lambda(B)|$ and $\ell = \max\{i : i \cdot \lambda_1(B) < n\}$. For $k \geq n$

$$s_k(B) = r_k(B) - \sum_{i=1}^{\ell} \mu(B_1^{\lambda_1(B)})^i s_{k-i\cdot\lambda_1(B)}(B) - \sum_{i=2}^{m} \mu(B_1^{\lambda_i(B)}) s_{k-\lambda_i(B)}(B).$$

For initial seeds, we have

$$r_n(B) = \mu(B)$$

and for $i < n$

$$s_i(B) = \begin{cases} 0 & \textit{if } i \notin \Lambda(B), \\ \mu(B_1^i) & \textit{if } i \in \Lambda(B). \end{cases}$$

Proof. For $k > n$

$$\begin{aligned} r_k(B) &= \Pr(x_1^n = B, R_n(x) > k - n) \\ &= \Pr(x_1^n = B, R_n(x) > k - n - 1) - \Pr(x_1^n = B, R_n(x) = k - n) \\ &= r_{k-1}(B) - \mu(B) s_{k-n}(B). \end{aligned}$$

For $k \geq n$ by Lemma 5 we have

$$s_k(B) = r_k(B) - \sum_{\lambda \in \Lambda(B)} \sum_{j=1}^{\ell(\lambda)} \Pr(x_1^{k-j\lambda} \in \mathcal{S}(B, k - j\lambda),\ x_{k-j\lambda+1}^{k} = B_1^\lambda \ldots B_1^\lambda).$$

By Lemma 2, for $i \geq 2$, $\ell(\lambda_i) = 1$. Hence

$$s_k(B) = r_k(B) - \sum_{i=1}^{\ell} s_{k-i\cdot\lambda_1(B)}(B) \mu(B_1^{\lambda_1(B)})^i - \sum_{i=2}^{m} s_{k-\lambda_i(B)}(B) \mu(B_1^{\lambda_i(B)}).$$

□

The computation of $s_k(B)$ for every n-block B is necessary for the application in later sections and it is done recursively on computers. To save time we use the fact that the blocks with the same $\Lambda(B)$ have the same pattern as far as the first return time is concerned. Thus we classify all the n-blocks using $\Lambda(B)$ and compute s_k for each block B from different classes.

3. Application: test for the pseudorandom number generators

We calculate $\Pr(R_n = k)$ for every integer $k \geq 1$ numerically by using the formula in the previous section. The averages and the standard deviations of the logarithm of the first return time are computed and the deviation of the experimental data from the theoretical prediction is used to test PRNGs. We apply the standard Z-test for sample mean of $\frac{1}{n} \log R_n$ of each block. In each case the sample size is 100,000. The test result is shown in Table 1. Test A is to consider the Z-values for each 14-block. Since the Z-values among each blocks are highly correlated, we need to reduce the correlation among the

TABLE 2. Theoretical values for (1/2,1/2)-Bernoulli process

n	$E[R_n]$	$E[\log R_n]/n$	$\sigma(\log R_n)/n$
2	4	0.7687192	0.5783033
3	8	0.8005356	0.4596296
4	16	0.8278062	0.3838731
5	32	0.8506853	0.3289105
6	64	0.8696175	0.2864550
7	128	0.8841787	0.2525540
8	256	0.8979582	0.2249535
9	512	0.9084916	0.2021856
10	1024	0.9172324	0.1832064
11	2048	0.9245484	0.1672354
12	4094	0.9307304	0.1536739
13	8192	0.9360054	0.1420567
14	16384	0.9405495	0.1320201
15	32768	0.9444992	0.1232780
16	65536	0.9479612	0.1156049
17	131072	0.9510188	0.1088215

Z-values. A binary block can be regarded as an integer in binary expansion We say that B is of

$$\begin{array}{ll}
\text{type I} & \text{if } B = (110101)_{(2)}k + (101001)_{(2)}, \\
\text{type II} & \text{if } B = (1011001)_{(2)}k + (111011)_{(2)}, \\
\text{type III} & \text{if } B = (1001001)_{(2)}k + (11111)_{(2)}, \\
\text{type IV} & \text{if } B = (1100101)_{(2)}k + (111101)_{(2)},
\end{array}$$

for some integer $k \geq 0$.

Test B-I, II, III, IV is the variance test of the Z-values over the blocks of type I, II, III, IV. The experiments show that correlations among the blocks of type I, II, III and IV are negligible. The symbols $\triangle$ and $\times$ denote the cases when the corresponding generators fail the test with statistical confidence of 95% and 99%, respectively.

TABLE 3. The test result for $n = 14$

Generator	Test A	Test B-I	Test B-II	Test B-III	Test B-IV
Randu	×	-	-	-	-
ANSI	×	-	-	-	-
MS	×	-	-	-	-
Fishman		×	△	×	△
ICG					
Ran0		×	×	×	×
Ran1		×	×		×
Ran2					
Ran3					
F90					

4. PSEUDORANDOM NUMBER GENERATORS

The following is a list of pseudorandom number generators tested in Section 3. We generate binary sequences using the algorithms listed in Table 4. A linear congruential generator $LCG(M, a, b)$ means the algorithm given by

$$X_{n+1} \equiv aX_n + b \pmod{M}.$$

Randu is an outdated generator developed by IBM in the sixties. ANSI and Microsoft are the generators used in C libraries by American National Standard Institute and Microsoft, respectively. For a prime p, the inversive congruential generator $ICG(p, a, b)$ is that

$$X_{n+1} \equiv a\overline{X_n} + b \pmod{p},$$

where $\overline{X}$ is the multiplicative inverse of x modulo p. The generators Ran0, Ran1, Ran2 and Ran3 are from [11]. Ran0 is the linear congruential generator by Park and Miller[10]. Ran1 is Ran0 with Bays-Durham shuffle. Ran2 is L'Ecuyer's generator[7] made up of

$$LCG(2147483563, 40014, 0) \text{ and } LCG(2147483399, 40692, 0)$$

with Bays-Durham shuffle. Ran3 is a subtractive lagged Fibonacci sequences. The subtract with borrow generator (SWB) is the form of $X_n \equiv X_{n-s} - X_{n-r} - b \pmod{M}$, where

TABLE 4. The tested random number generators

Name	Generator	Period
Randu	$LCG(2^{31}, 65539, 0)$	2^{29}
ANSI	$LCG(2^{31}, 1103515245, 12345)$	2^{31}
Microsoft	$LCG(2^{31}, 214013, 2531011)$	2^{31}
Fishman	$LCG(2^{31} - 1, 950706376, 0)$	$2^{31} - 2$
ICG	$ICG(2^{31} - 1, 1, 1)$	$2^{31} - 1$
Ran0	$LCG(2^{31} - 1, 16807, 0)$	$2^{31} - 2$
Ran1	Ran0 with shuffle	$> 2^{31} - 2$
Ran2	L'Ecuyer's algorithm with shuffle	$> 2.3 \times 10^{18}$
Ran3	$X_n \equiv X_{n-55} - X_{n-24} \pmod{2^{31}}$	$\geq 2^{55} - 1$
F90	Ran0 combined with shift register	$\sim 3.1 \times 10^{18}$
SR	$X_n = X_{n-1}(I + L^{13})(I + R^{17})(I + L^5)$	
SWB	$X_n \equiv X_{n-24} - X_{n-37} - b \pmod{2^{32}}$	$\sim 2^{1178}$

the borrow b is -1 if the previous subtraction is negative and 0 otherwise. The shift register generator (SR) is the form of $X_n \equiv X_{n-1}(I+L^r)(I+R^s)$ or $X_n \equiv X_{n-1}(I+R^r)(I+L^s)$, where $\oplus$ denotes the binary exclusive-or operation and L (resp. R) is the bitwise left-shift (resp. right-shift). F90 is Ran0 combined with SR[12].

REFERENCES

[1] G.H. Choe and D.H. Kim, *The first return time and test of pasedurandom numbers*, J. Comput. Appl. Math., to appear.

[2] ———, *Average convergence rate of the first return time*, Colloq. Math. **84/85** (2000), 159–171.

[3] W. Feller, *An Introduction to Probability Theory and Its Applications*, 3rd. ed., vol. 1, Wiley, New York, 1968.

[4] M. Kac, *On the notion of recurrence in discrete stochastic processes*, Bull. Amer. Math. Soc. **53** (1947), 1002–1010.

[5] D.H. Kim, *The recurrence of blocks for bernoulli processes*, submitted, 2000.

[6] I. Kontoyiannis, *Asymptotic recurrence and waiting times for stationary processes*, J. Theor. Prob. **11** (1998), 795–811.

[7] P. L'Ecuyer, *Efficient and portable combined random number generators*, Comm. ACM. **31** (1988), no. 6, 742–749.

[8] U. Maurer, *A universal statistical test for random bit generators*, J. Cryptology **5** (1992), 89–105.

[9] D. Ornstein and B. Weiss, *Entropy and data compression schemes*, IEEE Trans. Inform. Theory **39** (1993), 78–83.

[10] S. Park and K. Miller, *Random number generators: good ones are hard to find*, Comm. ACM. **31** (1988), no. 10, 1192–1201.

[11] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*, 2nd. ed., Cambridge Univ. Press, Cambridge, 1992.

[12] ———, *Numerical Recipes in Fortran 90*, Cambridge Univ. Press, Cambridge, 1996.

[13] C. Shannon, *The mathematical theory of communication*, Bell Sys. Tech. J. **27** (1948), 379–423, 623–656.

[14] P. Shields, *The Ergodic Theory of Discrete Sample Paths*, Graduate Studies in Math., vol. 13, Amer. Math. Soc., Providence, RI, 1996.

[15] A.D. Wyner and J. Ziv, *Some asymptotic properties of the entropy of stationary ergodic data source with applications to data compression*, IEEE Trans. Inform. Theory **35** (1989), 1250–1258.

[16] A.J. Wyner, *Strong matching theorems and applications to data compression and statistics*, Ph.D. thesis, Stanford University, Department of Statistics, 1993.

[17] ———, *More on recurrence and waiting times*, Ann. Appl. Ptobab. **9** (1999), no. 3, 780–796.

[18] J. Ziv and A. Lempel, *A universal algorithm for sequential data compression*, IEEE Trans. Inform. Theory **23** (1977), 337–343.

DEPARTMENT OF MATHEMATICS, KOREA ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY, TAEJON 305-701, KOREA

E-mail address: `kim@euclid.kaist.ac.kr`