Accuracy of Computer Simulations that use Common Pseudo-random Number Generators

Krid Dusitsin and Kurt Kosbar
Department of Electrical Engineering
University of Missouri – Rolla

ABSTRACT

In computer simulations of communication systems, linear congruential generators and shift registers are typically used to model noise and data sources. These generators are often assumed to be close to ideal (i.e. delta correlated), and an insignificant source of error in the simulation results. The samples generated by these algorithms have non-ideal autocorrelation functions, which may cause a non-uniform distribution in the data or noise signals. This error may cause the simulation bit-error-rate (BER) to be artificially high or low. In this paper, the problem is described through the use of confidence intervals. Tests are performed on several pseudo-random generators to access which ones are acceptable for computer simulation.

KEY WORDS

BER estimation, Pseudo-random number generator, Confidence Interval.

INTRODUCTION

Pseudo-random number (PN) generators are commonly used in computer simulations. These generators are not truly random, and will produce periodic output signals. This is often not a significant problem since the period can be made longer than the execution time of the simulation. Some statistical characteristics of the PN sequences deviate from ideal, including the autocorrelation function, power spectral density, probability density function, mean, variance, and higher order statistical moments. To trust the simulation results, one must ensure these imperfections do not significantly effect the measurements.

In this paper we are interested in determining if imperfections in PN generators adversely effect BER measurements. A standard technique used to measure the BER estimate accurately is the confidence interval. For an $a\%$ confidence interval (a is typically 95% to 99%), the BER measurement should fall within the confidence band in $a\%$ of the simulations. However, we found that commonly available PN generators can cause the BER estimate to fall outside the confidence band much more frequently than expected. Some of the non-ideal PN generator attributes listed above lead to a non-ideal distribution in the data and noise sources, and may ultimately causes the BER estimate to be artificially high or low.

PN GENERATORS AND CONFIDENCE INTERVALS

Most commonly used methods in generating a PN sequence are linear congruential and multiplicative congruential methods. For a linear congruential method, a number x_0 is chosen as a seed or starting number of the sequence, then subsequent numbers can be generated using the algorithm

$$x_{i+1} = (a x_i + c) \bmod m \qquad 0 < a, c_i < m \quad , x_i < m \tag{1}$$

where a, c, and m are called the multiplier, the increment, and the modulus, respectively. If a, c, and m are properly chosen, the linear congruential algorithm will generate a sequence of numbers distributed between 0 and m-1 with a maximum length period of m (e.g. the sequence will repeat the same pattern every m time index). If the increment, c, is set to zero, the generator is called a multiplicative congruential generator. In this case x_i cannot be assigned to zero, or the subsequent output will be perpetually fixed at this value. The maximum period of the multiplicative generator is reduced to m-1. From a sequence generated by either the linear congruential generator or the multiplicative congruential generator, a uniformly distributed sequence $\{u_i\}$ can be produced by

$$u_i = \frac{x_i}{m}. \tag{2}$$

The sequence $\{u_i\}$ is uniformly distributed between 0 and 1. An error sequence $\{e_i\}$ with a BER of p can then be generated from $\{u_i\}$ by

$$e_i = \begin{cases} 1 & \text{if } u_i \le p \\ 0 & \text{if } u_i > p \end{cases} \tag{3}$$

where 1 denotes error. The sequence $\{e_i\}$ has a Binomial distribution with the mean p. One problem that may arise when a linear congruential generator or a multiplicative congruential generator with a long period is used is integer overflow. One way to solve this problem is to combine two or more generators with relatively short periods together. Carefully chosen generators will effectively produce a generator with a much longer period and a better characteristic. One example of this type of generator is Wichmann-Hill generator [1].

Another way to generate a PN sequence is by using a shift register. The model of an *n*-bit shift register generator is shown in Figure 1.

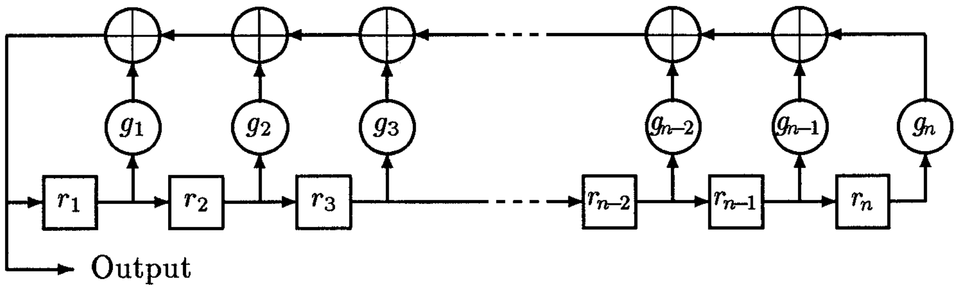

Figure 1: *n*-bit Shift Register Generator

An *n*-bit shift register consists of *n* memory blocks, or registers, and up to *n* modulo-2 adders. The seed number (in binary form) is placed in these registers and, in every iteration, the output of the register can be computed as

$$\text{Output} \;=\; r_1 \cdot g_1 \oplus r_2 \cdot g_2 \oplus \ldots \oplus r_{n\text{-}1} \cdot g_{n\text{-}1} \oplus r_n \cdot g_n \tag{4}$$

Here $\oplus$ is an exclusive-OR operator, or modulo-2 adder. To maximize the period of the generator to $2^n - 1$, select g_i to be a primitive polynomial of degree n (also in binary form).[1] On each clock cycle, the value in register r_i is shifted to register r_{i+1}. The output of the shift register is then stored in register r_1. The output sequence has a Binomial distribution with the mean of $2^{n-1}/2^n - 1$.

The immediate results from these generators are random bits, or random numbers with uniform distributions. With some transformation schemes, one can produce random variables with other statistical distributions such as normal distribution [3].

Once one obtains a BER estimate, the next question is: how accurate is this estimate? A standard method to measure the accuracy is the confidence interval. For a given $a\%$ confidence level, it will give us the band, or the interval, that the true BER is likely to reside. The graph of a typical confidence interval is shown in Figure 2.

[1] A comprehensive table of primitive polynomials of degree $n \leq 34$ can be found in [2].

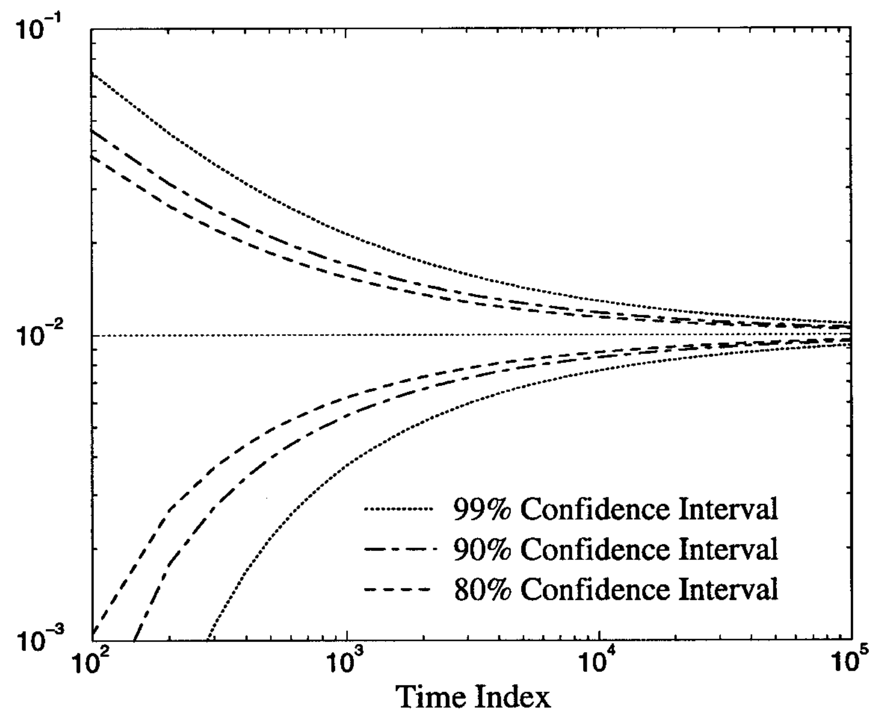

Figure 2: Confidence Intervals with Different Confidence Levels for a BER of 10^{-2}

In Bernoulli trials an error will occur with a probability p. The number of errors after N trials has a Binomial distribution with a mean of p and the variance of $p(1-p)/N$. Let k be a number of errors observed in these N trials, then the BER estimate for this simulation is $\hat{p} = k/N$ and an $a\,\%$ confidence interval can be obtained from solving for x from equations

$$\sum_{i=0}^{k} \binom{N}{i} x^{i}(1-x)^{N-i} = \frac{1-\frac{a}{100}}{2} \qquad \text{for an upper confidence interval and} \tag{5}$$

$$\sum_{i=k}^{N} \binom{N}{i} x^{i}(1-x)^{N-i} = \frac{1-\frac{a}{100}}{2} \qquad \text{for a lower confidence interval.} \tag{6}$$

However, it is not practical to use (5) and (6) to solve for the confidence interval because of the computer round off error, which is very significant for $N \geq 35$. One way to avoid this problem is to recognize the cumulative binomial distribution, or the left hand side of (5) and (6), is an incomplete beta function. One disadvantage of using the incomplete beta function is that it has the form of a continued fraction, and has to be solved iteratively [4]. This is a numerically demanding task for large N.

As $N \to \infty$, we can use the central limit theorem to approximate the Binomial distribution with the Gaussian or normal distribution. An equation for the $a\,\%$ confidence interval for the normal distribution can be found in [3]. Since the confidence interval for the normal distribution can be computed in one step, it is much faster to compute the confidence interval in this way than to use the direct method using the Binomial distribution model in (5) and (6), or the incomplete beta function. It is more desirable to use normal approximation in the confidence interval calculation, provided that the values of k and N are not too small. From experiments, the normalized difference of the confidence interval between the binomial distribution model (incomplete beta function) and the normal distribution model is less than 0.15 % for $k > 1000$. This threshold may be used to switch the calculation methods between binomial distribution and normal distribution to speed up the calculation without loss of the accuracy.

TESTS

Experiments were performed on several types of commonly used PN generators, including:

- Linear congruential generator
- Multiplicative congruential generator
- Wichmann-Hill generator
- GNU C++ random number generator – drand48
- Shift registers with various degree n.
- Normal distribution random variable transformed from the uniform random variable using the modified Box-Muller method [3].
- Normal distribution random variable generated from 99 linear congruential generator by means of the central limit theorem.

In the experiments, each generator listed above generates 100 different sequences, each with 10^6 samples. These sequences are then passes through a routine that measures the BER and calculates the confidence interval. This routine will also check whether the BER estimate stays within the confidence band (in present and past time index). If it stays within the band throughout all 10^6 samples, then that test is considered “passing the test”. The examples of the test that do not “pass the test” is shown in Figure 3.

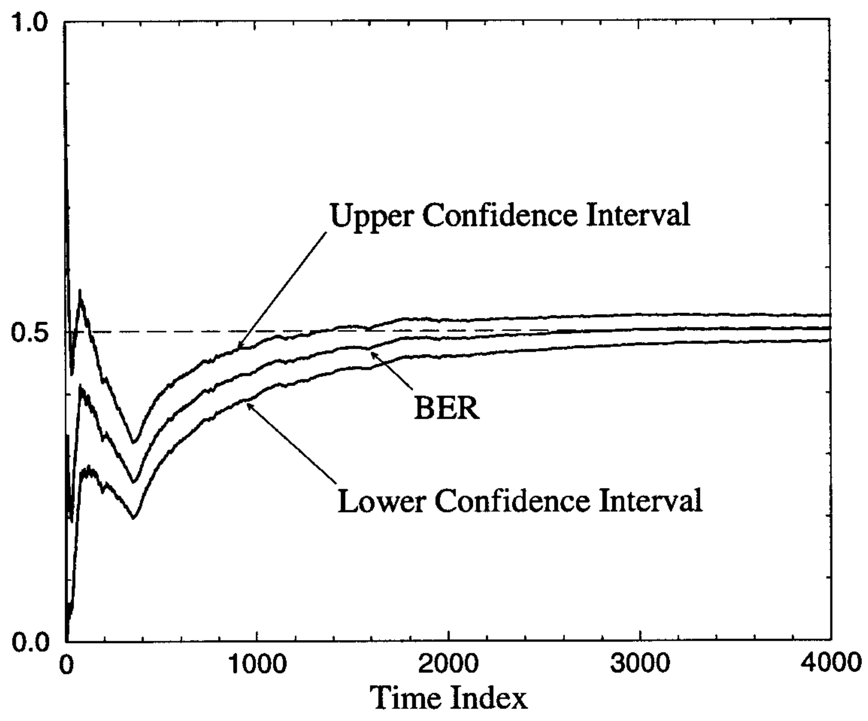

Figure 3: Shift Register with Primitive Polynomial of Degree 20

In Figure 3, we can see that errors rarely occur during the time index 200 to 500. Hence it pulls the BER estimate as well as the confidence interval down such that the confidence interval excludes the true BER which is approximately 0.5.

The experiment results from PN generators described above are shown in Table 1.

Table 1: Test Results - 100 tests

Generators	Number of "Pass"
Linear congruential generator	52
drand48	56
Wichmann-Hill generator	47
Shift register degrees 2 – 6*	100%
Shift register degree 7*	98.16%
Shift register degree 8*	97.16%

Shift register degree 9*	92.74%
Shift register degree 20	52
Normal dist. random variable with modified Box-Muller method	56
99 linear congruential generators with Central limit theorem	63

Note: * Comprehensive test with all possible seed numbers.

The number of passes shown in the table is not a definite performance index. A generator with 63 passes is not necessarily better than a generator with 56 passes because these numbers can vary from one experiment to another. However, the number of tests that do not remain within the confidence interval is much higher than expected. This may be the result of the non-ideal characteristics of the PN generators, which may cause the power spectral density, autocorrelation function and probability density function to all deviate from ideal. The non-uniform distribution in the data then causes the BER estimate to swing out of the confidence band.

PN generators with very short periods (small degree of primitive polynomial) always pass the confidence interval tests. This is also a problem, since a true random number generator will fail a 99% confidence interval tests in 1% of the simulations. An even more serious problem with these generators is their short periods cause the autocorrelation and power spectral density functions to be far from ideal.

CONCLUSION

Long period PN sequences can have very good autocorrelation and power spectral density properties, but may still be unsuitable for some simulations. In a very simple simulation, the confidence interval of the BER estimate excluded the true BER far too frequently. Short period generators do not have as much of a problem satisfying the confidence interval test, but they generate clearly periodic signals, which usually makes them unacceptable as noise generators. More work is needed to identify long period generators that pass the confidence interval tests.

REFERENCE

[1] B.A. Wichmann and I.D. Hill, “An efficient and portable pseudo-random number generator,” Journal of the Royal Society, Series C, Applied Statistics, London, Vol. 31, 1982, pages 188-190. Algorithm AS183.

[2] Rodger E. Ziemer and Roger L. Peterson, Digital Communications and Spread Spectrum Systems, Macmillan Publishing Company, New York, 1985.

[3] Michel C. Jeruchim, Philio Balaban, and K. Sam Shanmugan, Simulation of Communication Systems, Plenum Press, New York and London, 1992.

[4] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, Numerical Recipes in C, Cambridge University Press, Cambridge, 1988.

[5] Martina F. Schollmeyer and William H. Tranter, "Noise Generators for the simulation of digital communications systems," Proceeding of the 24th Annual Simulation Symposium, New Orleans, 1-5 April 1991, IEEE Computer Society Press, April 1991, pages 264-175.