Noise Generators for the Simulation of Digital Communication Systems

Martina F. Schollmeyer
Intelligent Systems Center

William H. Tranter
Department of Electrical Engineering

University of Missouri-Rolla
Rolla, Missouri 65401

ABSTRACT

The generation of random numbers, having uniform and Gaussian distributions, with application to the time-domain simulation of communication systems is considered. The specific goal is to identify appropriate noise generation algorithms for use in system simulations for the determination of the symbol error probability of a digital communication system.

Several methods of generating both uniform and Gaussian random numbers are examined. The algorithms considered were taken from those discussed in recent literature that appeared most appropriate for our problem. The algorithms were programmed in FORTRAN for use on a PC with math-coprocessor. Tests to evaluate the statistical properties of the generators were performed and the results are presented.

INTRODUCTION

Digital random noise generators have been used since the early days of digital computers. Many different methods have been used for generating samples of a random process ranging from stored tables of random numbers to linear congruential algorithms. The "best" method for generating samples of a random process on a digital computer quite often depends on the particular application. The application being considered in this paper is the time-domain simulation of digital communication systems. The overall problem being considered is the determination of the bit (or symbol) error probability of the system since the bit error probability is usually the best indicator of overall system performance. For this application a random number generator (RNG) having a relatively short period is required to simulate the data and a different RNG having a very long period is required for the noise. Both of these RNGs must produce statistically independent sequences to provide accurate simulation results.

When the bit error probability of a digital communication system is estimated using computer simulation, very long computer run times often result. As an example, consider a digital communication systems that is designed for bit error probability of 10^-8. This means that, on the average, 10^8 symbols must be processed for each error in a Monte-Carlo simulation of the system. One usually requires at least 100 errors if the error probability is to be estimated with reasonable confidence. In addition, approximately 10 samples per symbol are required for processing the system waveforms in the simulation. Thus, approximately 10^11 symbols must be processed in the simulation if a reasonably accurate estimate of the error probability is to be achieved. A typical simulation loop might contain 1000 lines of code and to process these 1000 lines 10^11 times will require extremely long run times.

We therefore see that efficient code is extremely important and this establishes the requirement for very efficient noise generators. Thus our search is for noise generator algorithms having short execution times and satisfactory statistical properties. The desired statistical properties include a delta correlated set of output samples and a long period. We also require that the output sample sequence have a Gaussian probability density function.

Several methods of generating uniform random numbers are examined. Tests to evaluate the statistical properties of the generators were performed and the results are presented. Where possible, the results of these tests are compared to those theoretically predicted. The tests performed include independence of pairs, equidistribution of sequences, the serial test (independence of n-tuples), the gap test (distribution of random numbers within an interval compared to the theoretical distribution), and the run-time (execution time) test. In addition, the spectral test was used on the RNGs implemented as linear congruential algorithms.

0272-4715/91/0000/0264$01.00 © 1991 IEEE

Techniques for transforming the RNG output, which has a uniform distribution, to a Gaussian distribution are investigated. The algorithms selected for testing include the original Box-Muller method, the revised Box-Muller (polar) method, and the Ziggurat method. These algorithms were then implemented in FORTRAN on a PC and the statistical properties of the resulting outputs were tested. The tests performed include independence of pairs, sample auto-correlation and sample cross-correlation, and a run-time (execution time) test. In addition the cumulative distribution functions obtained were compared with the actual Gaussian distribution.

Algorithms for the uniform to Gaussian mapping process that provided the most accurate results with at least one of the previously selected uniform RNG algorithms are identified and test results are presented.

This paper discusses the various methods investigated during the process of choosing noise generators for use in computer simulation of digital communication systems, as well as the methods used to evaluate the various algorithms. As such, this treatment is rather broad in scope and can hopefully serve as an introductory tutorial on random number generators.

BRIEF REVIEW OF LINEAR CONGRUENTIAL GENERATORS (LCGs)

Most of the uniform algorithms considered were linear congruential algorithms. A brief review of this class of algorithms is therefore appropriate. The linear congruential algorithm is the most commonly used method for creating sequences of random numbers. A full period generator is desired as well as a generator that produces a uniform random sequence having good statistical properties. The basic algorithm is defined by

$$X[n] = (a\,X[n-1] + c) \bmod m, \qquad c \neq 0$$

and is called the mixed congruential generator. If c is set equal to zero we have

$$X[n] = a\,X[n-1] \bmod m$$

which is referred to as the multiplicative congruential generator.

For mixed congruential uniform random number generators (RNG's) the maximum possible period is m, with m being the modulus of the RNG. The maximum period of the multiplicative congruential RNG is m-1. This is obvious since X[n] = 0 is not allowed in the sequence. Conditions that must be met to obtain a LCG with the maximum period are the following for a mixed congruential generator [1]:

1. c is relatively prime to m,
2. a = 1 (mod p) if p is a prime factor of m,

and 3. a = 1 (mod 4) if 4 is a factor of m.

For binary computers one tends to set m = 2^k and for this special case the preceding reduces to: a = 1 (mod 4) [1]. For multiplicative congruential generators the following must be satisfied in order to achieve the maximum period of m - 1 [2]:

1. m is a prime number,
2. X[0] is relatively prime to m,

and 3. a is a primative root modulo m.

If the modulus of the multiplicative congruential RNG is not a prime number, then the maximum period of the generator can be much shorter. If, for example, m is a power of 2, (i.e., m= 2^k) then the maximum period of the generator will only be m/4. In order to achieve this period we must select a = 3 or 5 mod 8. If m is the product of several different prime numbers then a must be a primetive root for all of them if the maximum period is to be achieved. The maximum period for this case is the lowest common multiple of the periods (p-1)(p^(k-1)) defined by each of the prime factors of the modulus, where p denotes a prime number and k is the power of p as a factor in the modulus m.

Park and Miller [3] also give a method for testing for the maximum period of multiplicative LCG's. The requirements for a maximum period sequence are:

1. m must be a large prime number,
2. 2 <= a <= m-1,
3. X[n+1] = a^n X[1] (mod m),

and 4. a^(m-1) mod m = 1.

A major weakness of the linear congruential method has been pointed out by Marsaglia [4]. He showed that successive overlapping sequences of random numbers from a multiplicative generator fall on parallel planes in hyperspace. This is very easy to visualize in a two dimensional space, since one obtains straight lines for linear congruential algorithms, thus producing pairs of random numbers that are not independent. An example of this behavior will be seen later (Figure 1). Marsaglia gives (n!m)^(1/n) as the highest possible number of hyperplanes in n-space, where m is the modulus of the LCG. The more hyperplanes, the closer they are together in n-space and therefore more points in the

space can be covered. Thus we attempt to find a RNG where the number of hyperplanes in n-space that it generates is as close as possible to the theoretical limit. This can be tested using the spectral test to be discussed later.

A SURVEY OF COMMONLY USED ALGORITHMS

In this section we briefly review a number of commonly used algorithms.

RANDU

An algorithm that received early and widespread use is RANDU, which uses the multiplicative congruential algorithm to generate uniform random numbers. It is defined by

$$X[n+1]=65539\ X[n] \bmod 2^{31}$$

with

period $2^{(31-2)}$

and was mainly chosen because its binary representation allowed for very fast computations. Its modulus is 2^{31}, and with 32 being a commonly used computer wordlength, easily implemented bit overflow methods can be used to generate the next random number. Nevertheless, these methods are not desirable because they are dependent on the wordlength of the computer to be used and thus limit portability. Knuth [2] shows that any 3 subsequent random numbers X[n], X[n+1], and X[n+2] can be combined such that $9X[n] - 6X[n+1] + X[n+2] = 0$ which gives unacceptable results for tests in three-space.

Minimum Standard proposed by Park and Miller

Because many uniform random number generators have periods that are too short to provide good results in Monte-Carlo simulations, Park and Miller picked a so called 'minimum standard' in 1988. They claimed that this RNG should be used if no better algorithm was available. In their research, they had examined RNGs commonly suggested in textbooks or implemented on computer systems that come as standard feature, and they found that many of these generators yielded poor performance. Park and Miller define their minimum standard as

$$X[n+1]=16807\ X[n] \bmod (2^{31}-1)$$

with

period $2^{31}-2$

which is also a multiplicative congruential algorithm. It has the maximum period m-1 of the multiplicative congruential type since $16807 = 7^5$ is a primitive root of the modulus, where $m = 2^{31} - 1$ is prime [3]. This RNG was extensively tested and provides better results than many other uniform LCGs. Schrage and Bratley, et al. developed computer programs for it [5,6] and Schrage already used this RNG about 10 years before Park and Miller defined it as minimum standard. L'Ecuyer also mentions it in his list of the best LCGs having a modulus of $2^{31} - 1$ [7].

Minimum Standard proposed by L'Ecuyer

L'Ecuyer [7] did extensive research on LCGs and came up with several different prime moduli and appropriate multipliers that provided much better results than previously suggested RNGs. Using a list of prime numbers less or equal to $2^{31} - 1$ and finding their primitive roots, he obtained many new LCGs. After performing the spectral test on these generators (a test method that will be discussed in the following section) he decided on a list of uniform RNGs with excellent statistical properties from where he then suggested an algorithm that he calls his minimum standard. It is defined by

$$X[n+1]=40692\ X[n] \bmod 2{,}147{,}483{,}399$$

with

period 2,147,483,398

and it was selected because of its excellent results in the spectral test and also because $40692^2 < m$ so that it can be programmed using a very efficient algorithm.

The three RNGs given above provide uniform random numbers using multiplicative congruential algorithms. We will see in a later section that the pairs of random numbers provided by these algorithms are not independent. They actually form lines in two-space and hyperplanes in all other spaces. Because RANDU has a modulus that is a power of 2, the seed must be odd to obtain the full period of $2^{(31-2)}$ for this type of LCG [2]. For both minimum standards, any seed except 0 can be used to obtain a full period of (m - 1) of the RNG since multiplier and modulus are relatively prime and the multiplier is a primitive root of the modulus.

Combining two or more LCGs into one uniform RNG

When several LCGs with different moduli and multipliers are combined, better uniform distribution can be achieved than with single LCGs. The random

numbers generated by the different LCGs can be added or subtracted or manipulated in other ways so that a better uniform distribution is obtained. Also, a longer period will usually be achieved for a combined RNG. It is not recommendable to combine RNGs of bad statistical properties but in order to obtain best results, only tested and well accepted uniform RNGs should be used in the combination. In addition to that, RNGs to be used should be logically and statistically independent.

Combined LCGs proposed by L'Ecyuer

L'Ecyuer [7] developed a theory about combining different multiplicative LCGs to improve the quality of the output and thus to make it more random. He approached the problem of dependencies between pairs of random numbers by combining two or more LCGs to make the overall period so long that the sample correlation between subsequently generated random numbers would be negligible, i.e., the pairs would be nearly independent. Also, during the random number generation process, intermediate values were stored differently than for the generation of uniform random numbers using single LCGs. This was done to increase the length of the period and thus to provide better randomness than any of the single LCGs.

L'Ecuyer suggests RNGs that can be adjusted to the wordlength of the computer that will be used. For example, for 32 bit computers only 2 LCGs need to be combined and for 16 bit computers 3 different LCGs are suggested for a good RNG. The results of addition or subtraction are then divided by the largest modulus and then provide the new uniform random numbers.

The LCGs that L'Ecuyer suggests for 32 bit computers are defined as

X[n+1]=40692 X[n] mod 2,147,483,399

with

period 2,147,483,398

and

X[n+1]=40014 X[n] mod 2,147,483,563

with

period 2,147,483,562

where the first of the listed RNGs is also known as L'Ecuyer's minimum standard. The LCGs that are used for 16 bit computers in a combination algorithm are defined by

X[n+1]=157 X[n] mod 32363
Y[n+1]=146 Y[n] mod 31727
Z[n+1]=142 Z[n] mod 31657

One can see that the period for each single LCG for 16 bit computers is very short. Any combination of these LCGs will definitely improve the randomness of the generated output simply because the period will become longer and thus the random number pairs seem to be more independent. The period obtained for 32 bit computers will be (m1-1)*(m2-1)/2 and a period of (m1-1)*(m2-1)*(m3-1)/4 for 16 bit computers will be possible. This provides such a long period that repetitions of pairs will be very hard to find.

Wichmann-Hill combined RNG

In 1982, Wichmann and Hill developed the idea of combining LCGs to obtain a better distribution of uniform random numbers than single LCGs [8]. Intended for small computers having only short wordlengths, the Wichmann-Hill algorithm obtains a good distribution even for those cases. The core of the combined RNG consists of three multiplicative LCGs which are defined by

X[n+1]=171 X[n] mod 30269
Y[n+1]=172 Y[n] mod 30307
Z[n+1]=170 Z[n] mod 30323

Although neither of these LCGs is particularly good, the result of the combination appears to be much better than any of the single LCGs. Coates et al. [9] also show that this combined RNG can be expressed as

X[n+1]=16,555,425,265,690 X[n]
mod 27,817,185,604,309

The period of this combined RNG is the least common multiplier of the moduli, i.e., 2*2*3*7*23*47*5051* 15161 which is approximately 7*10^12. This seems to be a sufficiently long period for most applications.

Coates et al. also suggest a different version of the Wichmann-Hill RNG. They reason that results will be better when good LCGs are used so that the combination of three better LCGs than the ones in the original algorithm should provide even better results [9]. Also, to obtain a longer period for the RNG, they want the moduli to consist of integers such that each (m-1) has no, or only very small, primefactors in common with the other moduli. The spectral test was used to find appropriate moduli and multipliers. The result of their research are the three LCGs defined by

X[n+1]=249 X[n] mod 61967
Y[n+1]=251 Y[n] mod 63443

Z[n+1]=252 Z[n] mod 63599

The period for this combined RNG is 2*38983*31721*31799 app. 7.9 * 10^13. Using the Wichmann-Hill principle it can be easy to generate different combined RNGs according to the wordlength of the computer to be used.

The 'best ever' RNG by Marsaglia, Zamann, and Tsang

In 1984, Marsgalia, et al. published a report about a 'universal' RNG using a non-Lehmer approach. A lagged Fibonacci series and an arithmetic sequence are combined to create a random number generator that is said to be the best one currently available. It has the large period of appr. 2^144 which is much longer than the period of most commonly used RNGs. About 2^120 derives from the lagged Fibonacci series, and it will be multiplied by the period of the arithmetic sequence with that the Fibonacci series is combined. This RNG is not initialized by a single seed, but by 97 integers that are needed for the lagged Fibonacci series. The series is initialized bitwise using a separate RNG in an initialization routine. For a fractional part of 24 bits a total of 97*24 bits must be initialized, thus about 2^97 * 2^24 app 2^120 initializations are possible. Because the period is so long and two very different types of RNGs are combined, the output of this uniform RNG is supposedly very random [10]. Details about this algorithm can be found in Marsaglia's paper.

UNIFORM TO GAUSSIAN MAPPING ALGORITHMS

For the simulation of digital communication systems, Gaussian RNGs are often used to simulate channel noise. For Monte-Carlo simulations the distribution of the random numbers must be very close to the modeled distribution. Most Gaussian mapping algorithms fail to model the tail region of the distribution properly, therefore a survey of mapping methods was necessary to determine the quality of an approximation for algorithms that are available.

Sum of N uniforms

The central limit theorem tells us that when a large number of random variables of arbitrary distribution is added, a probability function that approaches the Gaussian distribution function will be obtained. If N uniform random numbers in the interval [0, 1] are used then their sum is in the interval [0, N]. This shows that the summation method is very poor on the tails of the obtained distribution unless N is very large. Since a large value of N will result in a very slow algorithm, it is not useful for the computer simulation of digital communication systems where the tail distribution is commonly used, e.g., for an evaluation of the channel error. This method was therefore not used further but is mentioned for completeness since it is quite commonly used [11].

Ziggurat Method

The Ziggurat method to generate Gaussian random numbers from a uniform distribution is a fast and portable mapping method. It combines a fast algorithm for generating the values around the mean with a slower algorithm for the values in the tail region of the approximated Gaussian distribution. It was developed by Marsaglia and Tsang [12] to generate random variates of an arbitrary decreasing or symmetric unimodal probability function. The algorithm requires one uniform random number that will be used to determine in which area of the probability function the actual random number appears. If the number multiplied by a value from a lookup table is less than the previous value in the table then it will be accepted as Gaussian random variate. For the Gaussian distribution this will be true in about 96% of all cases if a table of length 64 is used. If this cannot be achieved then a simple algorithm is used to find the tail values. If that does not work either then a value from the tail density will be returned. Details can be found in the paper by Marsaglia and Tsang [12]. In the tests performed this mapping method generated cumulative distribution functions that differed greatly from the theoretical Gaussian distribution in the tail areas of the distribution. It was thus not considered a valid choice.

Box-Muller method

The Box-Muller algorithm generates two Gaussian random numbers simultaneously out of two uniform random numbers [13]. The uniform random numbers that are used should be independent and they must be in the interval [0, 1]. Two normal random variables are then derived by computing

x1=sqrt(-2*sigma^2*ln(u1))*cos(2*PI*u2)

and

x2=sqrt(-2*sigma^2*ln(u1))*sin(2*PI*u2)

where u1 and u2 are the uniform random numbers. This transformation can be derived by using polar coordinates for x1 and x2,

x1=r*cos(theta)
x2=r*sin(theta)

with r=sqrt(x1^2 + x2^2) and theta=arctan(x2/x1). The joint Gaussian distribution p(x1,x2) can then be transformed into the joint distribution p(r, theta). After changing variables we obtain

p(r,theta) = [r/(2*PI*sigma^2)]*exp(-r^2 / (2*sigma^2)), r>0, 0<=theta<=2*PI

One can see that this distribution is independent of theta. Thus, theta must be distributed uniformly in [0, 2*PI]. This can be shown by determining the marginal densities

p(theta) = 1/(2*theta), 0 <= theta <= 2*PI

and

p(r) = r/sigma^2 * exp(-r^2/(2*sigma^2)), r > 0.

Computing the integrals from 0 to r0 and from 0 to theta0, respectively, one obtains the expressions for r and theta to insert into the equations above.

Neave proves that because subsequent random numbers generated by an LCG are not truly independent, Gaussian random pairs consisting of x[i] and x[i+1] will always lie on a spiral [14]. This can be seen by substituting

u[i]=x[i]/m

and

u[i+1]=x[i+1]/m=((a*x[i]+c)mod m)/m
=(a*u[i]+c/m) mod 1

Since c only causes a shift of the sine/cosine function, one can set c=0 and obtain u[i+1]=a*u[i] mod 1 and thus

x[i]=cos(2*PI*a*u[i])*sqrt(-2*sigma^2*ln(u[i]))
x[i+1]=sin(2*PI*a*u[i])*sqrt(-2*sigma^2*ln(u[i]))

The sqrt part will be a damping function and the sine/cosine function will have a period of 1/a. Thus, all pairs of Gaussian random numbers generated by this algorithm will fall on a spiral. Clearly, they will not be independent.

One can also see that, remembering that the sqrt part of the function is the damping function, the distribution is bounded by values occuring at approximately the first minimum and first maximum of the sine-function. The damping function changes very slowly compared to the sine-function, except for very small random numbers. It can thus be seen that the distribution is bounded approximately by the intervals described by

[-sqrt(-2*sigma^2*ln(3/4a)):sqrt(-2*sigma^2*ln(1/4a))]

for x1 and

[-sqrt(-2*sigma^2*ln(1/2a)) : sqrt(-2*sigma^2*ln(1/m))]

for x2. Because of this effect, the Gaussian distribution generated from uniform random numbers created by an LCG will have discontinuities that will affect especially the tails of the Gaussian approximation [14]. Instead of a smooth curve there will be peaks in the tail region because there will be an exceptionally high number of random variates in short intervals just inside these discontinuities.

Modified Box-Muller method (polar method)

The modified Box-Muller method uses a rejection technique to eliminate the sine and cosine computations that are usually rather slow. For this algorithm again two uniform random numbers are needed to generate two Gaussian random variables, but they must be distributed in [-1, 1]. It is then checked if the pair is located inside the unit circle [2]. If this is true then two Gaussian random numbers can be obtained through the transformation

x1=v1*sqrt(-2*sigma^2*ln(S)/S)
x2=v2*sqrt(-2*sigma^2*ln(S)/S)

where v1 and v2 are uniform random numbers in [-1, 1] and S is computed as the sum of v1^2 and v2^2 and must be less than one. Again, polar coordinates can be used for the point representation

v1=r*sin(theta)
v2=r*cos(theta)

and one can see that S=r^2. We can then substitute the coordinate pair into the equation above to obtain

x1=cos(theta)*sqrt(-2*sigma^2*ln(S))
x2=sin(theta)*sqrt(-2*sigma^2*ln(S))

which defines r0=sqrt(-2*sigma^2*ln(S)) and theta0= theta. The value of r0 represents the distance from the origin to the point (the radius) and theta0 is the angle of the radius to the point within the unit circle. One then can compute the probability that r0<r to obtain a Gaussian distribution for S. Checking the marginal distributions and then combining them into a joint distribution and changing variables we obtain a Gaussian distribution for x1 and x2 [2]. The steps up to the

rejection will be computed on the average 4/PI app. 1.27 times before a suitable pair of values is found. This can be easily derived by looking at the unit circle that is used for the decision inside the square that contains the random variables in [-1, 1]. This ratio is also mentioned in [2]. It can be seen that this method is faster than the original Box-Muller method since it avoids the sine functions and it is therefore widely used. It is supposedly mentioned for the first time in [15] although many people call it Knuth's method.

TESTS ON NOISE GENERATORS

In order to be able to compare algorithms for the generation of pseudo-random variables, tests must be performed on sequences of random numbers that are generated. Statistical properties as well as execution time are important. For Monte-Carlo simulations of digital communication systems a large number of random variates is needed and therefore fast runtime results are often crucial.

Tests for uniform RNGs

For testing purposes only the best uniform RNGs discussed previously are considered now. This list consists of:

MIN : minimum standard by L'Ecuyer
WH : Wichmann-Hill algorithm
MWH : modified Wichmann Hill by Coates et al.
L2 : L'Ecuyer's combined RNG for 32 bits
L3 : L'Ecuyer's combined RNG for 16 bits
MARS : Marsaglia's best ever

To obtain most random results, the period of the RNGs discussed should have sufficient length not to repeat during one simulation. Long periods are therefore desired. The periods of the RNGs selected for testing are:

MIN - 2,147,483,398
which is approximately 2.1*10^9

WH - the least common multiplier of
(m1-1), (m2-1),(m3-1)
which is approximately 7*10^12

MWH - the least common multiplier of
(m1-1),(m2-1),(m3-1)
which is approximately 7.5*10^13

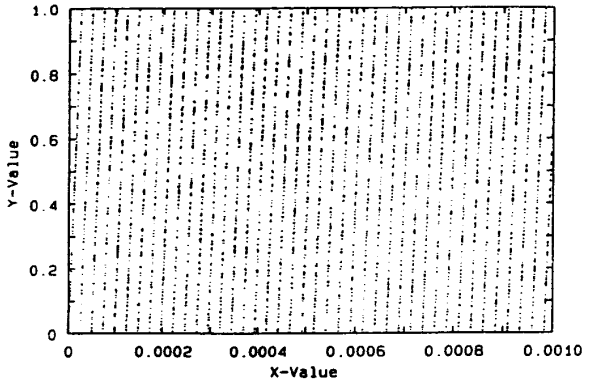
L2 - (m1-1)(m2-1)/2
which is approximately 2.3*10^18

L3 - (m1-1)(m2-1)(m3-1)/4
which is approximately 8.1*10^12

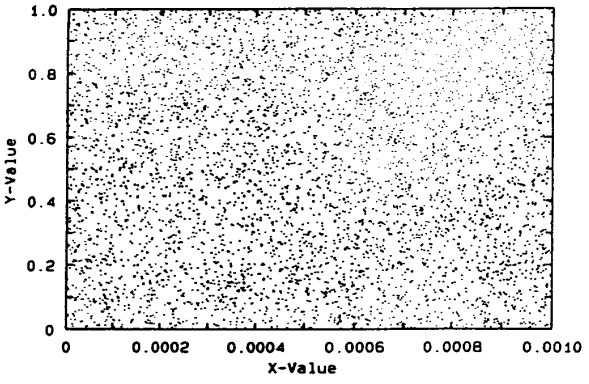
MARS - the product of the periods of the combined generators 2.2*10^43

Independence of pairs

In order to determine any statistical dependencies between uniform variates, scatter plots showing pairs of random numbers were generated. For MIN and L3, lines in 2-space could be easily detected. It seems that the modulus of MIN is too small and the same seems to be true for L3. This is illustrated in Figure 1.



(a) MIN (L'Ecuyer Minimum Standard)



(b) Wichmann-Hill

Figure 1. Uniform Number Generators

Equidistribution Test:

The equidistribution (or frequency) test checks the distribution of the uniform random variates. The test interval is divided into bins of equal size and the occurence of random numbers for each bin is counted. For a perfectly uniform distribution of N random numbers, approximately N/k random variates are expected to fall into each of one the k bins. A Chi-square test can then be performed, comparing the theoretical and experimental frequency counts.

Serial Test:

The serial test performs the same checks as the equidistribution test but in higher dimensions. We compare tuples of random numbers (pairs, triples, etc) and check for their uniform distribution in N-space. The bin concept is used again, and the more dimensional the space to be examined, the more random numbers have to be generated and tested to obtain accurate results when the space is divided into bins. Again, the theoretical and the experimental frequency counts can be compared using Chi-square tests.

Gap Test:

The gap test is used to examine the length of 'gaps' between occurences of uniform variates in a certain interval. We are generally interested in the number of random variables that fall into a certain interval, e.g., the far ends of the interval close to zero and one, and we then count the number of occurences of random numbers until we have another occurence in this same interval. A Chi-square test can be performed comparing the theoretical with the experimental results.

Spectral Test:

The spectral test is a very important tool for measuring the performance of LCGs since all good LGCs pass this test and all generators now known to be bad actually fail it [2]. Since the spectral test only applies to LCGs it cannot be used for an evaluation of MARS. The spectral test computes the number of hyperplanes in N-space and determines their distance from each other. As mentioned before, tuples of random numbers generated by an LCG fall onto parallel hyperplanes in N-space. Thus we will have empty slices in N-space where no tuples are located. The more hyperplanes we have the closer they will be together and thus the quality of the random output will improve. The spectral test was originally developed by Coveyou and MacPherson [16] and it uses a Fourier-analysis of the output sequence to evaluate the LCG. Thus, the statistical properties of each single LCG can be determined a priori. A detailed discussion of the spectral test can be found in [2].

Run Time Test:

The run time test was considered a very important test for the uniform random number generators since one of the objectives was to find a fast algorithm for the noise generation. Thus, the RNG under test was called a large number of times and the test results for different computers were compared. It could easily be seen that the combined RNGs using two or more LCGs took a much longer time for the generation of each variate than the ones using only single LCGs.

Test Results

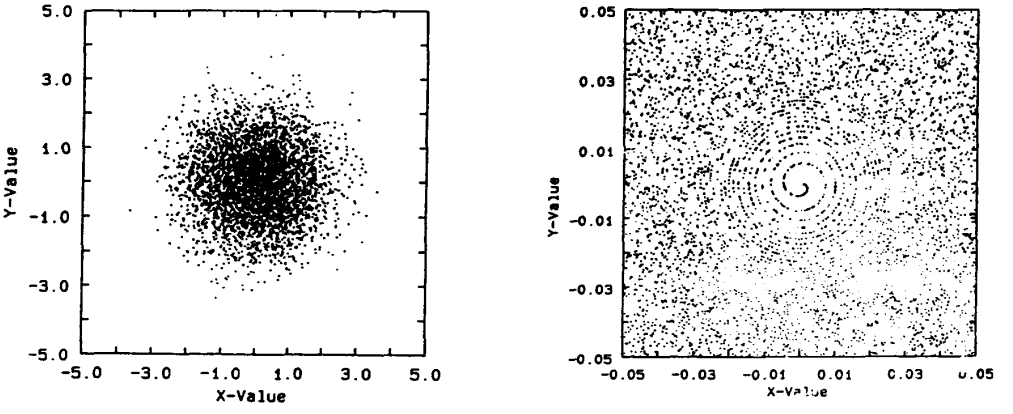
For execution time reasons WH, MHW, L2, and L3 had to be eliminated. They were up to three and more times slower than the faster algorithms. Since a large number of random numbers is needed to evaluate, for example, the channel error, the execution time was a main criterion. This leaves us with MIN and MARS where MIN has the problem of dependencies of pairs but otherwise being well tested and theoretically justified and where MARS is a new and somewhat hard to understand type of RNG with very little actual testing performed. Both of them are used for the following tests in a combination with uniform to Gaussian mapping algorithms.

TESTS FOR GAUSSIAN RANDOM NUMBER GENERATORS

The two uniform to Gaussian mapping algorithms discussed and tested are the original Box-Muller method (short: BM) and the modified Box-Muller (polar) method (short: MBM). Both of them have a sound theoretical foundation and are therefore most likely to provide good results. Each method is used with both MIN and MARS, thus providing a total of four different combinations to evaluate.

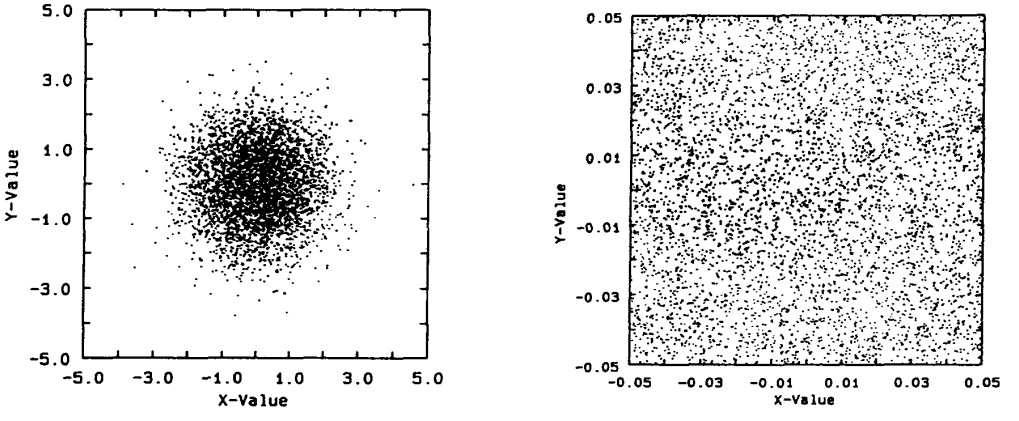
Independence of pairs

As for the uniform distributions, the Gaussian approximations can be tested for obvious dependencies using scatter plots of pairs of Gaussian random numbers. As expected from the theoretical discussion, MIN with BM shows a spiral in the center area of the distribution. This is illustrated in Figures 2 and 3. No other obvious dependencies were detected.



(a) Full Range (b) Central Region

Figure 2. Combination of MIN Uniform Number Generator with Box-Muller Algorithm



(a) Full Range (b) Central Region

Figure 3. Combination of MIN Uniform Number Generator with Modified Box-Muller Algorithm

Sample Autocorrelation

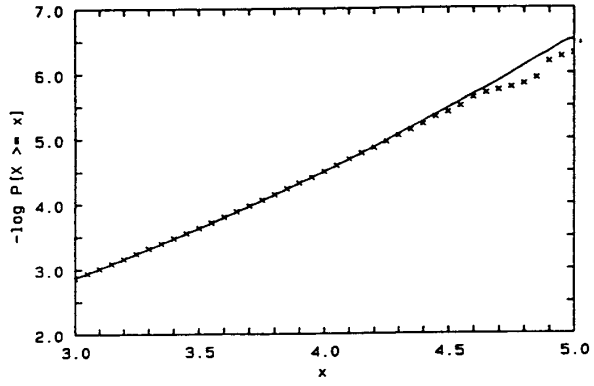
Sample autocorrelations of several random sequences were computed to determine any statistical dependencies between samples. A unit-impulse function was expected as result and was obtained for all combinations. No obvious dependencies could be detected.

Sample Cross Correlation

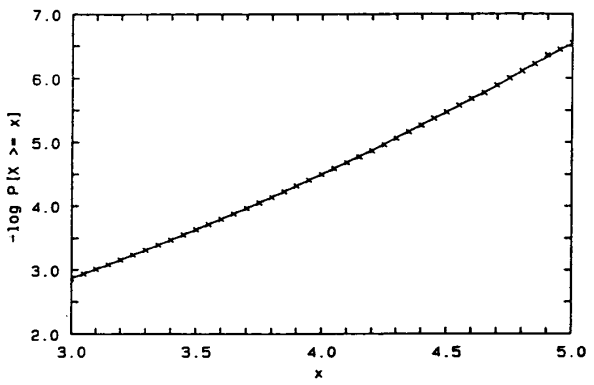
Sample cross correlations were also computed for the same sequences of random numbers, but again no obvious dependencies could be found. A function that was approximately zero along the whole test range was found and thus no statistical dependencies for the test range became obvious.

Cumulative Distribution

The cumulative distribution function (cdf) is a very important measure to determine how closely the approximation models the actual Gaussian distribution. This is especially important for the tail ends of the distribution which tend to be a weak point for many approximations. Again, it could be seen that BM and MIN did not work together, the resulting discontinuities were clearly visible in the very far tail region, and for LCGs with worse results in the spectral test (such as RANDU) they became visible much earlier. MIN works very well with MBM though, whereas MARS does not work well with it. MARS obtains better results with BM. Figure 4 illustrates the cumulative distribution functions of the MIN uniform number generator with both the Box-Muller algorithm and with the modified Box-Muller algorithm.



(a) MIN Uniform Number Generator with Box-Muller Algorithm



(b) MIN Uniform Number Generator with Modified Box-Muller Algorithm

Figure 4. Cumulative Distribution Functions

Run Time Test:

Again, fast execution times were desired and thus run time tests were performed. Since both MIN and MARS were quite close in their execution times, the combinations of MIN and MARS with BM and MBM also had similar execution times. Nevertheless, MBM proved to be about 20 percent faster than BM. Thus, MARS and MBM were the fastest and MIN with BM were the slowest combination.

Test Results

MIN and the original Box-Muller method showed major problems when combined and were therefore not recommended. Also, for some obscure reason, MARS and the modified Box-Muller method did not work together either. Both other combinations worked quite well, with MIN and the modified Box-Muller method being the faster of the two remaining solutions.

CONCLUSIONS

It can be seen from the the theoretical discussion and the actual test results that most random number generators promise more than they are able to provide. Most of the methods discussed in this paper are not suitable for use in the simulation of digital communication systems.

We have seen that there are uniform RNGs of good statistical properties using the Lehmer approach, especially when combination algorithms are used. The run time tests show very clearly, though, that high quality in the desired distribution has to be paid for with a loss of speed for the random number generation. This is also shown by the fact that only two of the original six uniform candidates were kept for the Gaussian tests, mainly for the execution time reasons.

After mapping the uniform random variates onto a Gaussian distribution it could be seen, as predicted by the theoretical discussion, that the Lehmer algorithm and the Box-Muller method do not work together. Nevertheless, two suitable generators were found and can be recommended.

The combination of MIN with the modified Box-Muller method is recommended if the quality of the uniform generator is not of concern but only the quality of the Gaussian noise. It is very easy to program and thus can be made portable easily from one system to another. The other good Gaussian approximation is the combination of MARS with the original Box-Muller method. Clearly, the uniform generator is better than MIN, but in the Gaussian output they are approxi-mately the same. Thus, this generator should be used if the quality of both the uniform and the Gaussian RNG is a concern.

Generally it can be seen that the combination of MIN and the modified Box-Muller method is faster and it is also easier to program. It is thus the first choice for use in the simulation of digital communication systems.

REFERENCES

[1] Hull, T.E. and Dobell, A.R., "Random Number Generators," *SIAM Review 4*, 1962, pp. 230-254.

[2] Knuth, D. E., *The Art of Computer Programing, Vol. 2, Seminumerical Algorithms, 2nd edition*, Addison-Wesley, 1981.

[3] Park, S.K., and Miller, K.W., "Random Number Generators: Good Ones Are Hard to Find," *Comm. ACM 31*, No. 10, Oct. 1988, pp. 1192-1201.

[4] Marsaglia, G., "Random Numbers Fall Mainly on the Planes," *Procceedings National Academy of Science 61*, Sept. 1968, pp. 25-28.

[5] Schrage, L., "A More Portable FORTRAN Random Number Generator," *Transactions on Mathematical Software 5*, No. 2 June 1979, pp. 132-138.

[6] Bratley, P., Fox, B.L., and Schrage, L., *A Guide to Simulation, 2nd edition*, Springer Verlag, New York, 1987.

[7] L'Ecuyer, P., "Efficient and Portable Combined Random Number Generators," *Comm. ACM 31*, No. 6, June, 1988, pp. 742-749,774.

[8] Wichmann, B.A., and Hill, I. D., "An Efficient and Portable Pseudo-Random Number Generator," *Appl. Stat. Algorithm 31*, 1982, AS 183, pp.188-190.

[9] Coates, Rodney F.W., Janacek, Gareth J., and Lever, Kenneth V., "Monte Carlo Simulation and Random Number Generation," *IEEE Selected Areas on Communications 6*, No. 1, Jan. 1988, pp. 58-66.

[10] Marsaglia, G., Zamann, A., and Tsang, W. W., "Toward a Universal Random Number Generator," Florida State University Report: FSU-SCRI-87-50, 1987, *Letters in Statistics and Probability*, Oct. 1989.

[11] Orfanidis, S., *Optimum Signal Processing - An Introduction, 2nd edition*, McGraw-Hill, 1988.

[12] Marsaglia, G., and Tsang, W.W., "A Fast, Easily Implemented Method for Sampling from Decreasing or Symmetric Unimodal Density Functions," *SIAM J. Sci. Stat. Comput.* Vol. 5, No. 2, 1984, pp. 349-359.

[13] Box, G.E.P., and Muller, M.E., "A Note on the Generation of Random Normal Deviates," *Ann. Math. Statist. 29*, 1958, pp. 610-611.

[14] Neave, H.R, On Using the Box-Muller Transformation with Multiplicative Congruential Pseudo-Random Number Generators," *Appl. Stat. 22*, 1973, pp. 92-97.

[15] Marsaglia, G., and Bray, T.A., "A Convenient Method for Generating Normal Random Variables," *SIAM Review*, Vol. 6, No. 3 1964, pp. 260-264.

[16] Coveyou, R.R. and MacPherson, R.D., "Fourier Analysis of Uniform Random Number Generators," *Comm. ACM 14*, No. 1 Jan. 1967, pp. 100-119.