On Random and Pseudo-Random Numbers

Luis F. Copertari

***Abstract*— Using pseudo-random number generators as a source of random numbers does present some problems, such as not having the population of the pseudo-random numbers obtained behave like a uniform distribution or even by having some specific pattern in the pseudo-random number generation process. It is very important to make sure the pseudo-random numbers behave properly when the results of simulations based on pseudo-random number generators are used if conclusions leading to scientific discoveries are to be considered valid. Instead of generating pseudo-random numbers, a method based on extracting English characters from text on the web or blog-sphere is proposed and tested.**

***Index Terms*— Science, random numbers, pseudo-random numbers, statistical tests.**

I. Introduction

Since the ever-growing use and access of computer power in personal computers after the late 1970s and 1980s, random numbers, and more specifically, pseudo-random numbers as well as computer simulation, have become a very important tool for making science. However, random and pseudo random numbers need to be used carefully and properly. This paper comprehensible explores this issue, highlights some of the pitfalls and provides solutions to some of the problems observed.

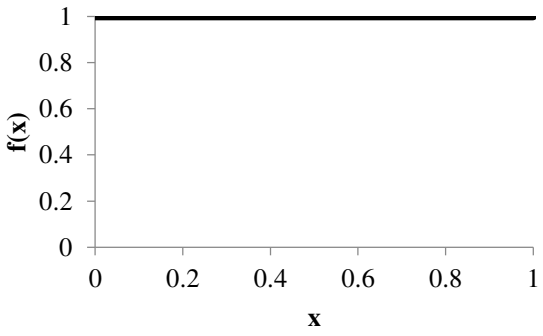
A good literature review on the subject of pseudo random number generation was carried out. There are different approaches and contributions to the theory of pseudo-random generators. Some articles deal with as much as robust possible pseudo-random bit generation so that good enough cryptography properties can be achieved [1-4]. Other approaches use chaotic maps and in some cases Chebyshev maps for good cryptography properties [2][[5-9]. One approach analyzes the application of pseudo-random number generators in Finance and how robust they are for the high volume of pseudo-random numbers required [10]. Another approach highlights the flaw inherent in pseudo-random number generators when simulating coin tossing [11]. One particularly interesting and promising approach uses hardware in order to create pseudo-random number generators. Arguably, this approach can lead to very good true random number properties since the system is connected to reality [12]. Other papers focus on cellular automata pseudo-random based number generation and their cryptography properties [13-14]. Finally, there are papers focusing mostly on testing pseudo-random number generators [15-19].

Let x be a random number, where $a \leq x \leq b$. Then f(x) is the probability density function associated to x. Clearly, equation (1) must be satisfied. If a = 0 and b = 1, x is a uniform random number, such that in order for equation (1) to be satisfied, f(x) = 1. The latter is illustrated in Figure 1. Generally speaking, for any uniformly distributed random number x, such that $a \leq x \leq b$, f(x) is given by equation (2).

$$\int_{-\infty}^{+\infty} f(x)dx = \int_{a}^{b} f(x)dx = 1 \quad (1)$$

$$f(x) = \frac{1}{b-a} \quad (2)$$

Figure 1. Uniformly distributed standard random number.



II. Random and Pseudo-Random Numbers

Random numbers only exist is nature. The only way to obtain a sequence of truly random numbers is by getting them from nature. For example, a sequence of binary random numbers (0 or 1) can be obtained by tossing a coin. If it is heads it could be assigned a value of 0 and if it is a tail a value of 1. Another example would be to toss a dice in order to obtain truly random values between 1 and 6.

The process of obtaining truly random real numbers between 0 and 1 is somewhat more complicated. To begin with, the set of all real numbers between 0 and 1 is infinite. Thus, some natural process that could manage that would be required. However, the universe, when measured, is discrete, and having any real number between 0 and 1 as a possible outcome would be impossible. In any case, drawing random numbers from the universe would be complicated and could take too much time. An option could be, for example, to get a measurement of the current in a given resistance at any given time, but that would not necessarily generate uniformly distributed random numbers.

A practical way to solve this problem is required. In practice, something called pseudo-random numbers are used [20]. Pseudo-random numbers are numbers that seem to be uniformly distributed random numbers for a given series of numbers (called the size of the pseudo-random number generator or M), but after a maximum of M different numbers, the series repeats itself. That is why they are called pseudo-random, because they are not really random.

Manuscript received January 04, 2019
Luis F. Copertari, Computer Engineering Program, Autonomous University of Zacatecas (UAZ), Zacatecas, Zac

However, for practical purposes, they seem to be. It is said, due to the fact that pseudo-random numbers are not truly random numbers and because they are not used with care (considering what the implicit underlying assumption using such pseudo-random numbers involves), that approximately one third of the annals of science should be discarded. This scientific legend may be exaggerated, but it highlights, nonetheless, the importance of knowing the assumption in which pseudo-random numbers, used in science as if they were random numbers, implies.

The mixed congruential method [20] is one of the most popular and widely used pseudo-random number generators. It is both fast and accurate enough. Let X_0 be an initial random number for X_i when i = 0. Then, the next random number between 0 and M is given according to equation (3), where MOD is the operation of taking the residual of dividing AX_i+C by M, A is a positive odd number not divisible by 3 or 5, C is a positive odd number preferably prime to M, $M > X_0$, M > A and M > C.

$$X_{i+1} = (AX_i + C) \text{ MOD } M \quad (3)$$

The reason all pseudo-random numbers are between 0 and M is because it is impossible that the residual could be any number larger than M. In fact, it is always less than M, so that $0 \leq X_i \leq M-1$ for all i (i is the iteration number and it has to be a number greater or equal than 0).

Then, a (presumably) uniformly distributed random number can be obtained according to equation (4) if it is desired that $0 \leq R_i \leq 1$ or according to equation (5) if $0 \leq R_i < 1$.

$$R_i = \frac{X_i}{M-1} \quad (4)$$

$$R_i = \frac{X_i}{M} \quad (5)$$

For simplicity of calculation reasons (not taking valuable computing time making the subtraction) equation (5) is most often used instead of equation (4) so that $0 \leq R_i < 1$. Table 1 shows an example in which $X_0 = 0$, A = 5 (for a moment let not consider the constraints imposed on A paragraphs above), C = 7 and M = 8. In this pseudo-random number generator, there are only a maximum of 8 different numbers, since M = 8, and only 8 numbers fit between 0 and 8-1 = 7; these numbers being 0, 1, 2, 3, 4, 5, 6, and 7, although they appear in different order. Clearly, that is not really the behavior of a truly random number, because in reality, a given sequence can actually have repeated numbers without causing the whole sequence to start repeating itself.

In the case of the sequence of pseudo-random numbers shown in Table 1, the sequence has the maximum possible number of values that do not repeat; in this case, that being 8 (0, 7, 2, 1, 4, 3, 6, and 5; after 5 follows 0 and the sequence naturally starts repeating itself).

III. Tests on Pseudo-Random Numbers

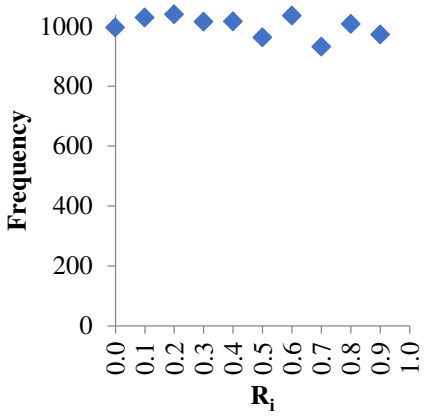
The first test will be to create a series of 10,000 pseudo random numbers. Let R be any such random number. Then, in this case, $0 \leq R < 1$. Figure 2a shows a histogram with the frequencies of numbers R having a range of 10 intervals (0…0.1, 0.1…0.2, 0.2…0.3, …, 0.9…1). Figure 2b shows the probability density function of the resulting numbers.

Table 1. Example of a mixed congruential generator.

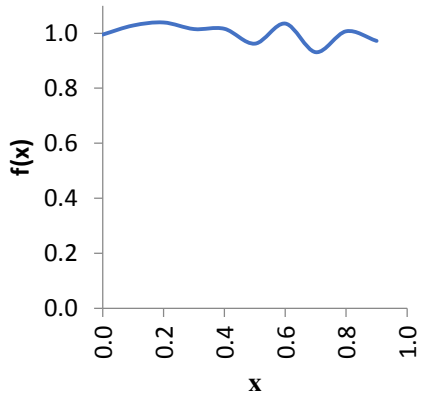
i	X_i	$(5X_i+7)/8$	X_{i+1}	$0 \leq R_{i+1} < 1$	$0 \leq R_{i+1} \leq 1$
0	0	0 + 7/8	7	7/8 = 0.875	7/7 = 1.000
1	7	5 + 2/8	2	2/8 = 0.250	2/7 ≈ 0.285
2	2	2 + 1/8	1	1/8 = 0.125	1/7 ≈ 0.142
3	1	1 + 4/8	4	4/8 = 0.500	4/7 ≈ 0.571
4	4	3 + 3/8	3	3/8 = 0.375	3/7 ≈ 0.428
5	3	2 + 6/8	6	6/8 = 0.750	6/7 ≈ 0.857
6	6	4 + 5/8	5	5/8 = 0.625	5/7 ≈ 0.714
7	5	4 + 0/8	0	0/8 = 0.000	0/7 = 0.000
8	0				

Figure 2. Uniformly distributed 10,000 pseudo-random numbers.

a. Frequencies.



b. Probability density function.



In the case of Figure 2a, there are exactly 10,000 pseudo-random numbers generated. However, the distribution is not exactly flat at exactly 1,000, but it varies around that number, because there is only a finite amount of numbers generated. Figure 2b shows how the density function is. Compare Figure 2b with Figure 1. Clearly, Figure 2b is not flat because it is an approximation. In theory, and assuming the pseudo-random number generator generates truly random numbers (which is not the case), Figure 2b should be flat as the amount of pseudo-random numbers approximates infinity. Also, the width of the intervals should be zero at the limit, instead of equal to 0.1, as it is the case in Figure 2b (and also Figure 2a).

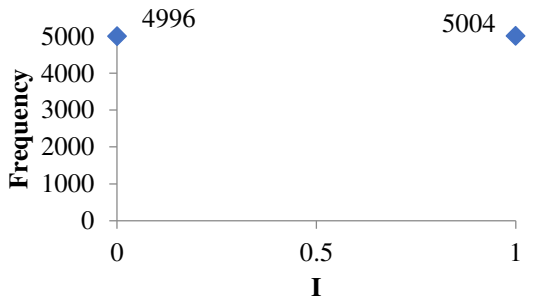
The second test is to see what happens when integer numbers are required. In this case, the generated numbers have to be either 0 or 1. Let I be any such number and R be

any such uniformly distributed pseudo-random number between 0 and 1. Since $0 \leq R < 1$, then I is given according to equation (6), where ROUND(R) means rounding down to zero when $R < 0.5$ and rounding up to 1 when $R \geq 0.5$.

$$I = ROUND(R) \quad (6)$$

Figure 3 shows the frequency distribution of 10,000 pseudo-random numbers generated. It can be seen that the probability of having I = 0 is almost (but not exactly) half of one (0.4996 to be exact), while the other (almost) half is the probability of having a I = 1 (0.5004).

Figure 3. Frequency distribution of two possible outcomes.



Clearly, this case seems to be acceptable. But things change if one is not careful. Consider now any given uniformly distributed zero-one pseudo-random number R, such that $0 \leq R < 1$. Now extend the previous idea to having as outcomes any given integer number I, where I could be equal to one, two or three. At first sight, equation (7) should give such results. The result of multiplying R by 2 is a number between zero and two and by adding one the number becomes a value between one and three.

$$I = ROUND(2R)+1 \quad (7)$$

Figure 4 shows the frequency distributions and probability distributions (not to confuse the latter with probability density function) of simulating 10,000 pseudo-random numbers and calculating in each case the value of I.

Figure 4. Frequency distribution and probability distribution of problematic use of pseudo-random numbers.

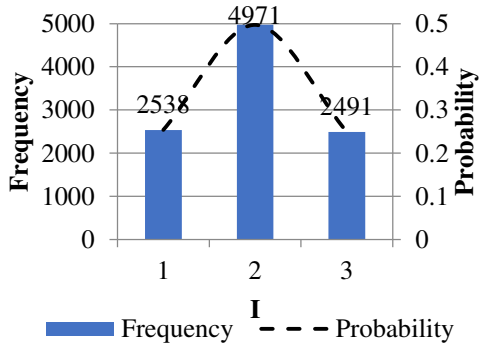


Figure 4 clearly shows an incorrect use of pseudo-random numbers. For values of I equal to 1 and 3, the probability is approximately 25%, while for a value of I equal to 2, the probability is approximately 50%. Certainly, all probabilities should be equal around 33.3%. But that does not happen. Where is the error?

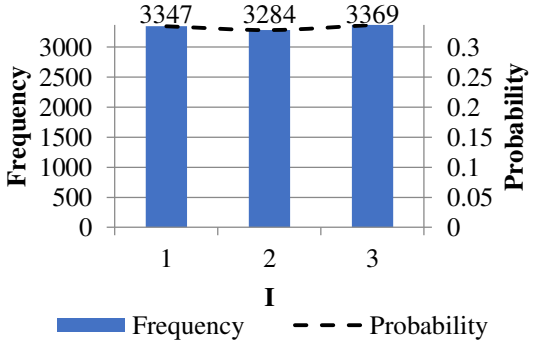
Notice the way equation (7) is structured. It says ROUND(2R)+1. That means $0 \leq 2R < 2$. Thus, if $0 \leq 2R < 0.5$, the result is 0 and I = 0+1 = 1. However, that range is only 25% of the whole range. Also, if $0.5 \leq 2R < 1.5$, the result is 1 and I = 1+1 = 2. The range of the latter includes half of all results, and thus the probability of occurring is 50%. Finally, if $1.5 \leq 2R < 2$, the result is 2 and I = 2+1 = 3, with a probability of occurring equal to 25%. That accounts for the mistakes observed. It may not seem obvious, but a considerable number of researchers whose results depend on schemes similar to the ones analyzed here, could end up making the same kind of mistakes.

How can it be corrected? Instead of using the rounding function, it is better to use the bottom function. Equation (8) shows such correction. The modified braces mean TRUNCATE(3R), which means taking only the integer part of 3R and letting go of the fractional part.

$$I = \lfloor 3R \rfloor + 1 \quad (8)$$

Figure 5 shows the result of simulating 10,000 pseudo-random numbers. This time, the probability behavior for I is flat around 1/3 probability for each occurrence of I (one, two, and three).

Figure 5. Frequency distribution and probability distribution of correct use of pseudo-random numbers.



IV. Characteristic of Mixed Congruential Pseudo-Random Number Generators

Mixed congruential pseudo-random number generators are described using equation (3) as discussed by Coss Bu [20]. In order to ascertain the characteristics of this type of pseudo-random number generators, it is necessary to simplify things a little bit. Consider the congruential number generator described by equation (9).

$$X_{i+1} = (X_i+1) \text{ MOD } 10 \quad (9)$$

This is a very simple generator, but it is useful to illustrate a basic property of the modulus (taking the residual of the) operator. Suppose $X_0 = 0$. According to equation (5) that corresponds to the uniform pseudo-random number $R_0 = 0/10 = 0.0$. Then, $X_1 = (0+1) \text{ MOD } 10$, which is 1, where $R_1 = 1/10 = 0.1$. Then follows $X_2 = (1+1) \text{ MOD } 10 = 2$, where $R_2 = 2/10 = 0.2$. And so on until $X_9 = (8+1) \text{ MOD } 10 = 9$, having $R_9 = 9/10 = 0.9$. The numbers obtained where the sequence 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. After that the sequence repeats itself, since $X_{10} = (9+1) \text{ MOD } 10 = 0$ and $R_{10} = 0/10 = 0$. In this simplified case, all possible numbers between 0 and 9 came out in order. It can be seen that by using the residual operation at best M different numbers can occur. Then, the sequence will repeat itself. Is it possible to obtain any number between 0 and 1? The answer: no. However, the frequencies will be flat, which is the important part. By changing the

parameters A and C in equation (3) it is possible to obtain a different sequence of M-1 numbers, but such sequence will always appear in the same order. Clearly, if M is large enough, that may not matter, but it does, because there would be repeating numbers. If one tosses a dice, it is possible to obtain the same consecutive output a few times, but always tossing a dice results in a different sequence as long as we are considering all the sequences from the beginning of the series. In mixed congruential pseudo-random number generators that is not possible.

V. Generating Random Numbers

So, how is it possible to get truly random numbers? The only way is to use nature as a source of such random numbers. A practical and simple enough way to get truly random sequences is to rely on text obtained from the blog-sphere, such as twitter, for example, or documents from a very large database such as the Wikipedia.

But how can we ensure that the behavior of such numbers is truly uniformly distributed? As an exploratory example, three themes were chosen. The first one is the World Wide Web (www). An interesting but perhaps ahead of its time and problematic account of the future of the World Wide Web can be found in Kurzweil [21]. The second theme selected was DNA. Sagan [22-23] explores some of the most important ideas concerning DNA. The third theme selected was Artificial Intelligence (AI). A very practical and down to Earth account of AI can be found in Boden [24]. These topics were searched in the Wikipedia and the results were copied and pasted into an ASCII based text format with the file names www.txt, DNA.txt, and AI.txt. ASCII are characters that can take any value between 0 and 255, where each character occupies one byte of information. The texts were in English. In order to simplify, only letters from A to Z and from a to z were considered, having the characters going through an UPCASE function, which transforms all letters into capital letters.

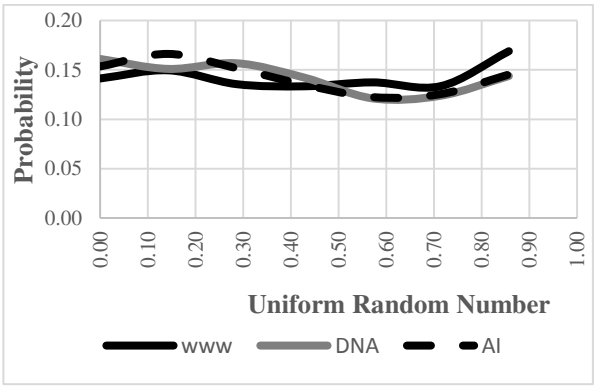
The problem is that letters do have a different frequency of occurrence. All three texts (www, DNA and AI) were analyzed and the relative occurrence of each letter from A to Z was calculated. It was clear that letters do have a very different probabilistic behavior. Thus, only specific letters having almost the same probability of occurring were considered. These letters are A, I, N, O, R, S, and T. Table 2 compiles the information gathered.

As it can be seen in Table 2 letters A, I, N, O, R, S, and T do have approximately the same number of occurrences in each text file. The letters A, I, N, O, R, S, and T, are assigned the numbers (X) 0, 1, 2, 3, 4, 5 and 6, respectively. There is a total of M=7 different numbers. In order to get a uniform random number, R is obtained by dividing X by M. Only the first two digits after the decimal point are being used to obtain a four digits presumably uniformly distributed random number. Approximately 51% of the total number of letters in the text files are covered by these seven letters, and approximately 38% of these seven letters constitute the total size of the file, where the size of the file only includes the total number of characters it contains, without line jumping. Figure 6 pictures the probabilities.

Table 2. Probabilistic behavior of selected letters in all three texts.

			Frequency			Probability		
Letter	X	R = X/M	www	DNA	AI	www	DNA	AI
A	0	0.000000	2,593	6,337	7,123	0.1414	0.1611	0.1536
I	1	0.142857	2,744	5,942	7,709	0.1496	0.1511	0.1662
N	2	0.285714	2,487	6,167	7,038	0.1356	0.1568	0.1518
O	3	0.428571	2,447	5,592	6,263	0.1334	0.1422	0.1350
R	4	0.571429	2,520	4,767	5,678	0.1374	0.1212	0.1224
S	5	0.714286	2,450	4,872	5,823	0.1336	0.1239	0.1256
T	6	0.857143	3,099	5,657	6,742	0.1690	0.1438	0.1454
M =	7	Total selected	18,340	39,334	46,376			
		Total for all letters	36,073	75,041	88,830			
		Relative percentage	50.84%	52.42%	52.21%			
		File sizes	47,900	105,428	121,285			
		Sizes percentages	38.29%	37.31%	38.24%			

Figure 6. Probability distribution of the seven letters selected.



Although the behavior in the three cases (www, DNA and AI) does not provide a flat occurrence of probabilities, it is approximately flat. What is more important is that the outcomes will be truly random if any text in English taken from the web or the blog-sphere is used. Nevertheless, it would be useful to have the possibility of other numbers to occur besides the ones shown in Table 2. Let R denote any such number. Then, RR can be computed according to equation (10), where each R is a different and consecutive occurrence obtained from the system. Thus, a couple of different and consecutively random numbers (R_1 and R_2) are used such that the first one constitutes the first two decimal points and the second one constitutes the second two decimal points. In this way, RR can contain a total of 4 useful decimal digits.

$$RR = \frac{\lfloor 100R_1 \rfloor}{100} + \frac{\lfloor 10000R_2 \rfloor - 100\lfloor 100R_2 \rfloor}{10000} \qquad (10)$$

The RR numbers are in theory truly random numbers. However, do they pass the statistical tests for uniformly distributed random numbers? There are several statistical tests available, but the two most common ones are the means test and the variance test. A total of 1,000 RR numbers are taken from the files www.txt, DNA.txt and AI.txt. They are compared with 1,000 numbers generated by Delphi's internal random number generation system and 1,000 random numbers generated by Excel. The Delphi and Excel random numbers generated are different for each of the three text files

being used (n = 1,000).

VI. Mean Statistical Test for Uniformly Distributed Random Numbers

The aim is to prove that the population mean (known to be ½ = 0.5) is significantly equal to the sample mean (calculated). An unknown sample mean and a known variance is what we have in this case. The null and alternative hypothesis are as follow:

H_0: $\mu = \mu_0$

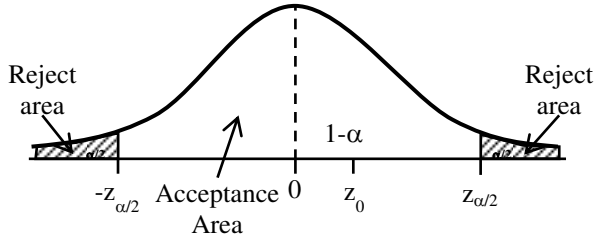
$\mu = ½$

H_1: $\mu \neq \mu_0$

$\mu \neq ½$

Figure 7 illustrates the test and the acceptance (pass) area. The z (normal) statistical distribution is used.

Figure 7. Means statistical test criteria.



The error type I is being minimized, that is, rejecting H_0 when in fact is should be accepted, which is equivalent to saying that the sample mean is not equal to the population mean when in fact it is.

A calculated z statistic (z_0) is compared against a z normal statistic from tables ($z_{\alpha/2}$). Equation (11) shows the calculation of the z_0 statistic, where the population variance (σ^2) is 1/12. Notice that the sample mean ($\bar{x}$) is given according to equation (12).

$$z_0 = \frac{\bar{X}-\mu_0}{\sigma/\sqrt{n}} = \frac{\bar{X}-\frac{1}{2}}{\sqrt{\frac{1}{12}}/\sqrt{n}} \quad (11)$$

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \quad (12)$$

The acceptance criterion is if $z_{-\alpha/2} < z_0 < z_{\alpha/2}$, then H_0 is accepted, which means the set of presumed random numbers has passed the test. Otherwise, it is not.

VII. Variance Statistical Test for Uniformly Distributed Random Numbers

The variance test questions whether or not the sample variance is within the accepted range for the population variance. The null and alternative hypothesis are as follows:

H_0: $\sigma^2 = \sigma_0^2$

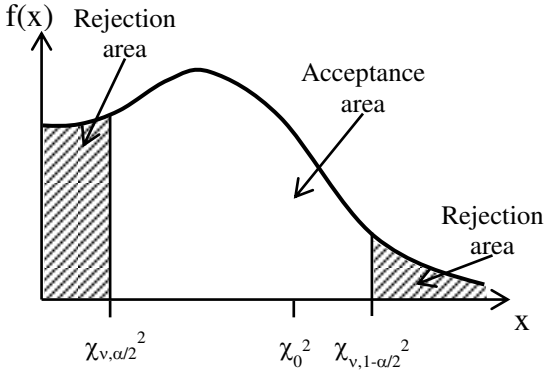
$\sigma^2 = 1/12$

H_1: $\sigma^2 \neq \sigma_0^2$

$\sigma^2 \neq 1/12$

The sample variance is known and it is compared with an unknown population variance. Figure 8 illustrates the variance test acceptance criteria. The ji-square (χ^2) statistical distribution is used. A calculated χ_0^2 is compared with χ^2 distribution values. The degrees of freedom of the test are $\nu = n-1$.

Figure 8. Variance test acceptance criteria.



Equation (13) shows the calculation of the χ_0^2 statistic figure, while equation (14) pictures the calculation of the sample variance (S^2).

$$\chi_0^2 = \frac{S^2(n-1)}{\sigma_0^2} = \frac{S^2(n-1)}{1/12} \quad (13)$$

$$S^2 = \frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1} \quad (14)$$

VIII. Results Obtained from the Experiments and the Statistical Tests

The results obtained are positive. The proposed method for obtaining truly random numbers did pass both tests (mean and variance). Also, the pseudo-random numbers obtained from Delphi and the pseudo-random numbers obtained from Excel also passed both tests. Table 3 summarizes the results for the mean test and Table 4 shows the results for the variance test. The confidence is 95% ($\alpha = 0.05$).

Table 3. Results of the mean test.

α = 0.05	www			DNA			AI		
Statistic	RR	Delphi	Excel	RR	Delphi	Excel	RR	Delphi	Excel
z_0	-0.0057	-0.0007	0.0014	0.5002	0.0000	0.0008	0.0003	-0.0003	0.0003
$-z_{0.025}$	-1.96	-1.96	-1.96	-1.96	-1.96	-1.96	-1.96	-1.96	-1.96
$z_{0.025}$	1.96	1.96	1.96	1.96	1.96	1.96	1.96	1.96	1.96
Result	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed

Table 4. Results of the variance test.

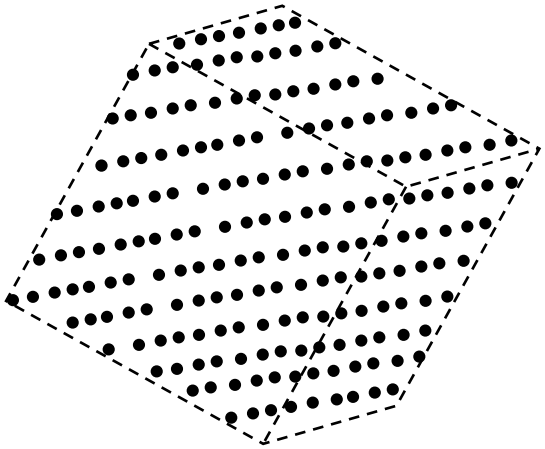
α = 0.05	www			DNA			AI		
Statistic	RR	Delphi	Excel	RR	Delphi	Excel	RR	Delphi	Excel
χ_0^2	1002.64	990.35	955.93	1020.48	982.65	983.65	1041.69	969.86	1016.82
$\chi_{n-1,0.025}^2$	913.30	913.30	913.30	913.30	913.30	913.30	913.30	913.30	913.30
$\chi_{n-1,0.975}^2$	1088.49	1088.49	1088.49	1088.49	1088.49	1088.49	1088.49	1088.49	1088.49
Result	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed

IX. Discussion and Conclusion

The problem of pseudo-random generators, whatever their nature may be, is that they are not really random. That means that there could be some inherent pattern in their behavior. To

illustrate, consider having a sequence of pseudo-random numbers and plotting them on a three-dimensional scatterplot. The first such number would be the first occurrence of the first coordinate (x_1), the second number the first occurrence of the second coordinate (y_1), the third number the first occurrence of the third coordinate (z_1) and then again, the fourth number for the second occurrence of the first coordinate (x_2), and so on. By carefully choosing the angle of view, it is possible, in some cases, to find a pattern, such as the one illustrated by Figure 6.

Figure 6. Pattern on a three-dimensional scatterplot of pseudo-random numbers.



Although testing the sequence of pseudo-random numbers may result in passing all statistical tests, it is clear from Figure 6 that the numbers are arranged in a specific pattern. It is not possible to have numbers in between the planes perpendicular to the observation plane found in Figure 6. Thus, the sequence of pseudo-random numbers is not truly random.

A solution may be to obtain the random numbers from an irrational number, such as $\sqrt{2}$ or $\sqrt{3}$. Since these sequences continue to infinity, in theory, irrational numbers could provide a source for truly random numbers. There are algorithmic procedures for finding the square root of a prime number. However, as the sequence goes on, the memory requirements grow, and, in any case, the procedure would not allow for a fast creation of random numbers. Also, an irrational number like π could be the source of random numbers. However, the existence of an algorithmic procedure fast enough for finding irrational numbers with any desired degree of accuracy (any given number of digits) is dubious.

If we are guided by the idea that random numbers only exist in nature, it may be possible to construct a device, such as a measurement of current in a given transistor, in order to obtain sequences of random numbers. The problem with this method is that it depends on a specific hardware implementation. Another way to go around this problem could be to obtain text from the on-line blog-sphere and transform such sequences of characters into numbers. This second method is not hardware dependent, but it does require an internet connection, which is in most cases the norm. Furthermore, since the blog-sphere is the result of people interacting on-line, there are even some philosophical and theological conundrums that can be solved in this way.

Let R be a pseudo-random number such that $0 \leq R < 1$. In such case, if a total of d digits are required after the decimal point, equation (15) can be used.

$$R_i = \frac{\lfloor 10^d R \rfloor}{10^d} \tag{15}$$

If a binary number from R_i is to be required, the ROUND function can be used, which is correct from a technical point of view (refer to Figure 3). The round function is represented here using top brackets as shown in equation (16).

$$I_i = \text{ROUND}(R_i) = \lceil R_i \rceil \tag{16}$$

Alternatively, if R_i from equation (15) has four decimal points, R_i can take values from 0.0000 to 0.9999. Multiplying that by 2 results in the range $0.0000 \leq 2R_i \leq 1.9998$, and thus equation (17) can be used to get binary numbers (0 and 1) with equal frequency distributions between the zeroes and the ones.

$$I_i = \lfloor 2R_i \rfloor \tag{17}$$

If we are to take only the first two digits of a sequence of random numbers generated according to the method here proposed, there would be only seven different possible outcomes: 0.00, 0.14, 0.28, 0.42, 0.57, 0.71, and 0.85. Although there is only a limited number of possibilities, the sequence in which they appear would be truly random because they are based on seven different selected characters from texts taken from the web or the blog-sphere (Twitter for example). By combining any number of sequences of two digits, random numbers of any desire number of decimal places can be obtained. Let dd be the total number of two-digit random numbers R_i obtained for each RR, where i = 1, …, dd, then RR can be given according to equation (18).

$$RR = \sum_{i=1}^{dd} \frac{\lfloor 100 R_i \rfloor}{10^{2i}} \tag{18}$$

Acknowledgment

The author would like to thank the support of the Autonomous University of Zacatecas (UAZ) while doing this research. Also, I would like to thank the help and guidance of colleagues within UAZ.

References

[1] Wang, Xing-Yuan & Xie, Yi Xin. (2012). A design of pseudo-random bit generator based on single chaotic system. *International Journal of Modern Physics C*, *23*(3), 1-11.

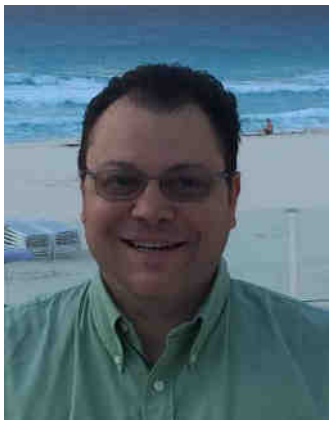
[2] Francois, Michael, Grosges, Thomas, Barchiesi, Dominique & Erra, Robert. (2013). A New Pseudo-Random Number Generator Based on Two Chaotic Maps. *Informatica*, *24*(2), 181-197.

[3] Kant, Shri & Khan, Shehroz S. (2006). Analyzing a class of pseudo-random bit generator through inductive machine learning paradigm. *Intelligent Data Analysis*, *10*, 539-554.

[4] Romashchenko, Andrei. (2014). Pseodo-Random Graphs and Bit Probe Schemes with One Sided Error. *Theory Computation Systems*, *55*, 313-329.

[5] Patidar, Vinod & Sud, K. K. (2009). A Novel Pseudo Random Bit Generator Based on Chaotic Standard Map and its Testing. *Electronic Journal of Theoretical Physics*, *6*(20), 327-344.

[6] Lui, Oi-Yan, Yuen, Ching-Hung & Wong, Kwok-Wo. (2013). A pseudo-random number generator employing multiple Rényi maps. *International Journal of Modern Physics C*, *24*(11), 1-10.
[7] Kordov, K. M. (2014). Modified Chebyshev Maps Based Pseudo-random Bit Genetator. *Application of Mathematics in Technical and Natural Sciences. AIP Conference Proceedings*, *1629*, 432-436.
[8] Akhshani, Afshing, Behnia, Sohrab, Akhavan, Amir, Lim, Siew-Choo & Hassan, Zainuriah. (2010). Pseudo random number generator based on synchronized chaotic maps. *International Journal of Modern Physics C*, *21*(2), 275-290.
[9] Stoyanov, B. P. (2013). Pseudo-random Bit Generator Based on Chebyshev Map. *Applications of Mathematics in Technical and Natural Sciences. AIP Conf. Proc., 1561*, 369-372.
[10] Mascagni, Michael, Qiu, Yue & Hin, Lin-Yee. (2014). High performance computing in quantitative finance: A review from the pseudo-random number generator perspective. *Monte Carlo Methods Applications*, *20*(2), 101-120.
[11] Bauke, Heiko & Mertens, Stephan. (2004). Pseudo Random Coins Show More Heads than Tails. *Journal of Statistical Physics*, *114*(3&4), 1149-1169.
[12] Savic, Nemanja, Stojcev, Mile, Nikolic, Tatjana, Petrovic, Vladimir & Jovanovic, Goran. (2014). Reconfigurable Low Power Architecture for Fault Tolerant Pseudo-Random Number Generation. *Journal of Circuits, Systems and Computers*, *23*(1), 1-21.
[13] Murguía, J. S., Mejía Carlos, M., Rosu, H. C. & Flores-Eraña, C. (2010). Improvement and analysis of a pseudo-random bit generator by means of cellular automata. *International Journal of Modern Physics C*, *21*(6), 741-756.
[14] Temiz, A., Siap, I. & Akin, H. (2014). On Pseudo Random Bit Generators via Two-Dimensional Hybrid Cellular Automata. *Acta Physica Polonica A*, *125*(2), 534-537.
[15] Gil, Manuel, Gonnet, Gaston H. & Petersen, Wesley P. (2006). A Repetition Test for Pseudo-Random Generators. *Monte Carlo Methods and Applications*, *12*(5-6), 385-393.
[16] Naor, Moni & Reingold, Omer. (2004). Number-Theoretic Constructions of Efficient Pseudo-Random Functions. *Journal of the ACM*, *51*(2), 231-262.
[17] García Martínez, M. & Campos-Cantón, E. (2014). Pseudo-random bit generator based on lag time series. *International Journal of Modern Physics C*, *25*(4), 1-15.
[18] Condo, C. & Gross, W. J. (2015). Pseudo-random Gaussian distribution through optimised LFSR permutations. *Electronic Letters*, *51*(25), 2098-2100.
[19] Rusu, Florin & Dobra, Alin. (2007). Pseudo-Random Number Generation for Sketch Based Estimations. *ACM Transactions on Database Systems*, *32*(2), 1-48.
[20] Coss Bu, Raúl. (1991). *Simulación: un enfoque práctico*. Editorial Limusa.
[21] Kurzweil, Ray. (2005). *The singularity is near: when humans transcend biology*. Penguin Books.
[22] Sagan, Carl. (1980). *Cosmos*. Editorial Planeta.
[23] Sagan, Carl. (2000). *Cosmos* [DVD 7-Disc Collector's Edition]. DZN.
[24] Boden, Margaret A. (2016). *AI: Its nature and future*. Oxford University Press.



Luis F. Copertari is a professor and researcher at the Autonomous University of Zacatecas (UAZ). Prior to joining UAZ he has worked as research assistant and executive assistant at the Monterrey Institute of Technology and Advanced Studies (ITESM). Then the conducted research and teaching assistant duties at McMaster University while doing his Ph.D. He worked for one month as postdoctoral researcher at the University of Auckland. He has been a full-time professor and researcher at UAZ for about sixteen years.