Iterated Function Systems as Human Perceivable Spectral Tests of Randomness

MANOLIS WALLACE, HARA STEFANOU, STEFANOS KOLLIAS
Image, Video and Multimedia Systems Laboratory
National Technical University of Athens
9, Iroon Polytechniou Str., 157 73 Zographou, Athens
GREECE
wallace@image.ntua.gr http://www.image.ntua.gr

Abstract: - The deterministic generation of pseudorandom sequences is not a trivial task. Quite the contrary; even chaotic functions are often poor pseudorandom sequence generators. Thus, given the range of applications that require random inputs and the cost of actual random number generating equipment, the establishment of reliable tests of randomness becomes necessary. To day, the spectral test seems to be the only reliable test for randomness, as it can examine the correlation between successive symbols in a sequence. The drawback of the spectral test is that its results can only be visualized when comparing for correlation between consecutive pairs or triplets of symbols, while larger groups of symbols can only be examined mechanistically. In this paper, after reviewing the main points of random number and chaotic functions theory, we introduce a 2-dimensional spectral test of randomness that is based on iterated function systems (IFSs) and use it to examine the quality of various chaotic functions as random number generators and to draw conclusions on the randomness of sequences produced by deterministic processes.

Key-Words: - Spectral test, pseudorandom sequence, Iterated Function System (IFS), chaotic function.

1 Introduction

The great variety of computer applications that require the use of random data, combined with the (economic and other) difficulties of attaching a random number generating device to each computer, calls for the use of computer-generated sequences, that possess certain characteristics of randomness. In short, we wish to use sequences of numbers that are generated deterministically but appear to be random. We refer to these sequences as pseudorandom or quasi-random sequences.

Although randomness is a concept that we understand and identify easily in everyday situations, it is somewhat harder to define and identify it in a formal manner. One rather informal way to approach the concept follows: "In a sense there is no such thing as a random number; for example, is 2 a random number? Rather we speak of a sequence of independent random numbers with a specified distribution, and this means loosely that each number was obtained merely by chance, having nothing to do with other numbers of the sequence, and that each number has a specified probability of falling in any given range of values" [1].

A more formal approach is: "A random number is a number chosen as if by chance from some specified distribution such that selection of a large set of these numbers reproduces the underlying distribution. Almost always, such numbers are also required to be independent, so that there are no correlations between successive numbers. Computer-generated random numbers are sometimes called pseudorandom numbers, while the term random is reserved for the output of unpredictable physical processes. When used without qualification, the word random usually means random with a uniform distribution. Other distributions are of course possible".

The above might strike us as relaxed constraints. Still, it is impossible to produce an arbitrarily long string of random digits, according to them. Strangely, it is also very difficult for humans to produce a string of random digits, and computer programs can be written which, on average, actually predict some of the digits humans will write down, based on previous ones.

A simple way to explain the criteria for randomness proposed in [3] is the following: we may consider as random all the sequences that are so complicated, that the simplest way to describe them is by providing a copy of them. In other words, if a sequence can be reproduced by the application of a relatively short algorithm, it may not be considered random. This implies that by no means may a computer-generated sequence be considered random. In practice, though, we only attempt to verify randomness by examining the distribution of the numbers in a sequence and their independence.

In this paper we start by reviewing some of the most classical pseudorandom generators as well as the tests used to verify their quality. As we shall explain, the most powerful and reliable of them (the spectral test) has the disadvantage of being purely mechanistic, in the sense that it cannot be visualized and thus be made humanly perceivable. Continuing, after briefly reviewing chaotic functions, we propose a human perceivable 2D test that is based on Iterated Function Systems (IFSs) and is equivalent to the spectral test. We also draw some conclusions on the randomness of sequences produced by pseudorandom number generators.

2 Random Number Generators

Pseudorandom number generators are simple deterministic functions that can produce sequences or numbers of arbitrarily long period. These sequences are also required to have certain characteristics that are related to randomness, as is for example the independence of numbers.

Numerous pseudorandom number generators can be found in the literature, each one displaying different characteristics. In the following we only present two of them, in order to point out that simple functions are often better pseudorandom number generators that more complex ones, thus making the need for a reliable test of randomness imperative.

2.1 von Neumann

One of the first methods proposed for the generation of pseudorandom sequences is the following: given number X_n, acquire X_{n+1} by squaring X_n and keeping the middle digits. This technique, although seems to be quite random, was proven to be quite inadequate. One of its major drawbacks is the dependence on the initial value X_0. For most initial values the system quickly reaches an orbit of short period, thus producing highly predictable sequences.

We include this example in our presentation in order to demonstrate that the generation of pseudorandom sequences cannot be accomplished with the use of methods chosen at random. This is demonstrated more clearly in [1].

2.2 Linear Congruence

The linear congruential is the most widely used deterministic random number generator. It is described by the formula:

$$X_{n+1} = (aX_n + c) \bmod m \qquad (1)$$

It is obvious that this formula produces integers in the interval $[0, m-1]$. Therefore, the generated sequence is periodic with a period that cannot exceed m. The study of (1) has produced the necessary theoretical background, to allow for the easy selection of combinations of a, c and m, that lead to the generation of sequences of maximum period m.

Since all numbers in $[0, m-1]$ are visited exactly once, it is easy to verify that the distribution the numbers in the sequence follow is uniform. Formula (2) is used to produce fractions in the interval $[0,1]$

$$U_n = \frac{X_n}{m-1} \qquad (2)$$

Of course, the linear congruence is not an excellent random number generator for any given value of its parameters. In fact, for some cases its performance is remarkably bad. More on deterministic random number generators (and especially on the linear congruence) can be found in [1].

3 Tests of Randomness

It is not that rare for a pseudorandom number generator to appear to be able to produce random numbers, and still for the acquired sequences to be unacceptable. Therefore, a reliable way of verifying the 'randomness' of a deterministic generator is required.

Unfortunately, it is not possible to evaluate the 'goodness' of a generator directly. Rather than that, a generator is used to produce a pseudorandom sequence of numbers, which is then examined for characteristics of randomness. This means that our tests can only provide a statistical estimation of the goodness of a generator, rather than a conclusive result. When a large number of such statistical tests has been applied to a generator and the generator has not failed any of them, then we assume it to be a good generator (until proven otherwise).

In [3] randomness is defined only for sequences of numbers, and it is based on the complexity of the sequences. Formally, a sequence S of length n is considered to be random when

$$n = K(S) \qquad (3)$$

where $K(S)$ is the complexity of the sequence, i.e. the amount of information it contains. Using this definition it may be shown that most finite sequences are random. Therefore, the constraint that this definition describes is considered overly relaxed (and is not used in practice). When $n \to \infty$, even for chaotic sequences, $K(S) << n$. This is "corrected" by defining a new measure of complexity as follows

$$K'(S) = \lim_{n \to \infty} \frac{K(S)}{n} \quad (4)$$

When $K'(S) > 0$, we consider the sequence S to be of maximum complexity, and therefore random. As we will see later on, not all chaotic sequences should be considered random. This means that the measure of randomness described in (4) is not really useful. Furthermore, the $n \to \infty$ component makes the measure applicable only in theoretical studies of randomness rather than in practical tests.

Knuth [1] proposes a set of simple randomness tests to apply to a sequence. FIPS 140-1 [6] is a standard test series defined by the US National Institute of Standards and Technology (NIST), inspired by Knuth's basic tests. This standard was overseed by FIPS 140-2. FIPS 140-2 [7] defines a more comprehensive battery of tests, including Maurer's Universal Statistical test. Maurer's Universal Statistical test [4] is a test capable of detecting a large class of defects. In addition, this test provides an estimate of the entropy of the source. A better estimate can be obtained using a modified version of the test.

In the following we review some of the most common practical tests for randomness.

3.1 Period

The finite precision of computer arithmetic guarantees that all deterministic pseudorandom sequences are periodic. Still, if the period is large enough (if the sequence is longer than the sequence of random numbers that any application might ask for) this is not a problem. This makes the length of the sequence an important issue.

Therefore, an obvious measure of the goodness of a pseudorandom number generator is the length of the period of the sequences it produces. Although for the linear congruential generator this issue has been solved theoretically (and uniquely), for other generators this has to be done through testing. An easy way to measure the period length is to let the system iterate for n steps (n needs to be considerably large, so that the system reaches a periodic orbit), and then count the number of steps it takes for it to return to the value X_n.

For most pseudorandom number generators, there is not just one periodic orbit. Therefore, the estimation of the length of the period will not always produce the same output. Furthermore, the consistent calculation of a large period for numerous starting values does not guarantee the non-existence of a periodic orbit of extremely short period. (For more on this, one can see [2])

3.2 Chi-square

Chi-square is a statistical test that examines randomness when a uniform distribution is assumed. In the following we sketch the main concept behind it: suppose that we choose randomly 6 digits from the set $\{0,1\}$, assuming equal probabilities for the two symbols. There are 64 different permutations that this procedure might have produced, all having the same probability. The probability of the count of ones being equal to the number of zeros is $20/64$. The probability of the number of ones being 2 is smaller ($15/64$). The same holds for the probability of the number of ones being 4. Still, the probability of the number of ones being 1 different than the most probable is higher than the probability of them being exactly 3. This implies that generally we should not expect truly random sequences to follow their distributions perfectly.

Chi-square measures the difference between most probable and observed frequencies. The formula used is described in (5)

$$V = \sum_{s=1}^{v+1} \frac{\left(Y_s - np_s\right)^2}{np_s} \quad (5)$$

where $v+1$ is the count of possible values, p_s is the probability of the s^{th} value, Y_s is the number of times the s^{th} value appears in the sequence and n is the length of the observed sequence. Readers may refer to [1] for the probabilities of V lying in a specific range for various values of $v+1$.

3.3 Spectral Test

The spectral test is one of the most important tests proposed in the literature. The reason for this is that, not only do all good generators pass it, but also all generators known to be bad actually fail it. The test checks the independence between consecutive numbers produced by a generator, by grouping them together in k-tuples and examining the way these k-tuples are positioned in R^k. When $k = 3$, the positioning of the 3-tuples in R^3 using spherical coordinates, is referred to as a *Noise Sphere*.

Although very efficient in examining the ‘randomness’ of a sequence of numbers using a deterministic pseudorandom number generator, the spectral test has a unique disadvantage when compared to the simpler abovementioned tests: when k assumes values larger than 3 it is not possible for the test to be made human perceivable through visualization.

4 Chaotic functions

It is quite obvious that (1) describes a chaotic system. One typically has to wonder whether chaotic systems can generally be considered as good pseudorandom sequence generators. This was actually first implied by Ulam and von Neumann, who proposed the use of the quadratic equation (6) for the generation of pseudorandom sequences.

$$X_{n+1} = 4X_n(1 - X_n) \quad (6)$$

Unfortunately, this turns out to be a poor random number generator. First of all, it generates sequences of extremely short period. Numerous consecutive tests (using the timestamp as a seed) and using the methodology described in section 3.1 calculated a period of 5,638,349. Of course, as we have already explained, this does not guarantee the non-existence of orbits with longer (or shorter) periods. We may consider, for example, the case of $X_n = 3/4$, after which the period is reduced to one, as $X_{n+1} = X_n$.

Choosing the chaotic generator described in (7) we calculate the significantly longer (but still quite short) period of 55,176,418. On the other hand, a few million random numbers are probably enough to satisfy the needs of quite a large number of applications, and therefore the small length of the period is not necessarily enough for us to renounce generators in (6) and (7) as bad. It is other properties of randomness, such as those examined by the spectral test, that determine the quality of a pseudorandom number generator.

$$X_{n+1} = \sin(\pi X_n) \quad (7)$$

Continuing, we apply the chi-square test, which both (6) and (7) fail tragically, with the exception of (6) when $v+1=2$. Especially in that case, we suspect that the generator might not produce good pseudorandom sequences, because the balance between the two possible symbols is too good (and, as we have already explained, the probability of the balance being that good is very small). The poor performance of these generators in the chi-square test does not by any means imply that they are not good random number generators. It just points out that they are not following the uniform distribution, which is a fundamental assumption for the application of the chi-square test. This is apparent in figures 1-3.

It is relatively easy to find the analytic formula of the distribution followed by these generators [2], and therefore it is also possible to select $v+1$ areas of unequal size in a such way that the chi-square test is completed successfully.

In general, the fact that these generators do not follow the uniform distribution makes it impossible to apply most statistical tests, as they are based on the assumption of uniformicity. In general, it is not easy to estimate how good most chaotic random number generator are, as they rarely follow the uniform distribution.

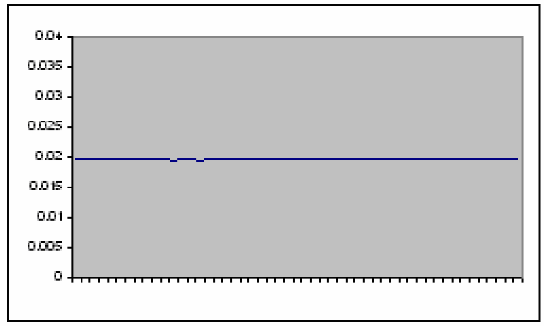

Fig. 1 Distribution of the Linear Congruence

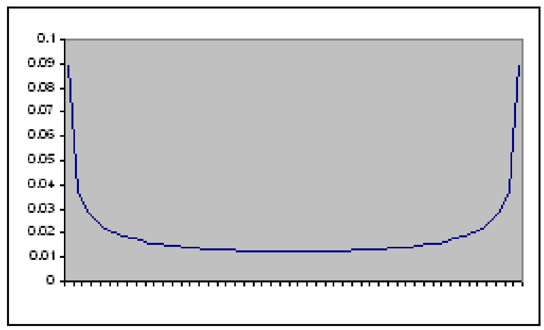

Fig. 2 Distribution of the Quadratic

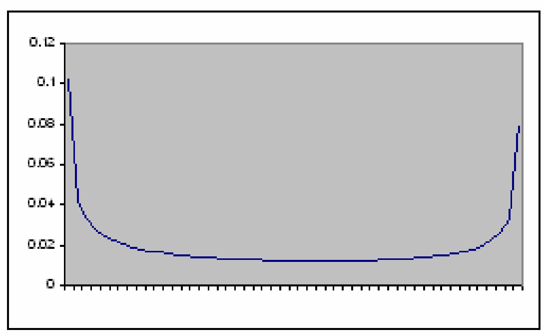

Fig. 3 Distribution of the Sinusoidal

5 Iterated Function Systems

IFSs allow for the extremely compact modeling of complex fractal images, through the specification of a limited number of affine transformation; the reconstruction of the fractal image can be achieved via recursive application of the affine transformations.

In fact, although having extremely good storage properties, IFS modeling of images has the important disadvantage of often leading to computationally intractable representations. For example, it is impossible to reproduce Barnsley's ferm deterministically.

The reproduction of the attractors of IFSs can be tackled using the monte carlo approach proposed in [8]. This approach relies on the random sequential application of the affine transformations that describe the IFS. Its operation greatly relies on the assumption of independence between elements of a random sequence; this sequence is supposed to only contain as many different symbols as the count of transformations in the IFS, so we split $[0,1]$ in that many areas. In the case that the symbols are not independent, the attractor of the IFS is not completed successfully.

Therefore, the application of the monte carlo approach to drawing fractal images described via IFSs can be used for the evaluation of the independence of consecutive numbers of a random number generator, much like the spectral test. For example, using (6) to drive the monte carlo process for the generation of the fern (see Fig. 7), an extremely limited subset of the points of the triangle is drawn, while the linear congruence produces the fractal of Fig 6.

When X_n lies in a specific area of $[0,1]$, X_{n+1} may only end up in a small set of areas of $[0,1]$. In other words, consecutive elements of our sequences are far from independent. Following the reasoning of this proof, it is easy to conclude that all iterative systems that rely on a continuous generator will be bad generators of random numbers, if the first derivative of the generator is absolutely small. In general, the exponential propagation of error, that chaotic systems guarantee, is not sufficient for a good random number generator. It is necessary that the error is multiplied by a great factor even from the very first step, as we need the system to be totally unpredictable, even as far as the very next step is concerned.

To further clarify this statement we present figures 4-5. Figure 4 presents the function in (8). It is easy to see that the area $[0.6,0.8]$ is mapped to the area $[\sin(0.8)^2, \sin(0.6)^2] \approx [0.905,0.345]$. Therefore, if we use (8) to randomly generate integers in the range 1-5, after the digit 4 we will never have the digit 1 (which corresponds to area $[0,0.2]$). On the other hand, if we use (9), which is presented in figure 5, no such constraint exists.

$$X_{n+1} = \sin(\pi X_n)^2 \quad (8)$$

$$X_{n+1} = \sin(10\pi X_n)^2 \quad (9)$$

If we attempt to use (9) for the generation of random sequences with more than 5 symbols we may start observing the same dependence between consecutive symbols. This implies that pseudorandom sequences only appear random up to a certain level of detail. After that threshold, dependence appears, and, after a second threshold, all uncertainty is lost (if the number of symbols is equal to the count of numbers representable in our computer system, the process is obviously totally deterministic).

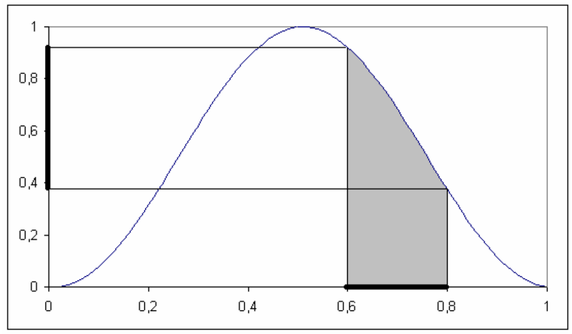

Fig. 4 Function 8

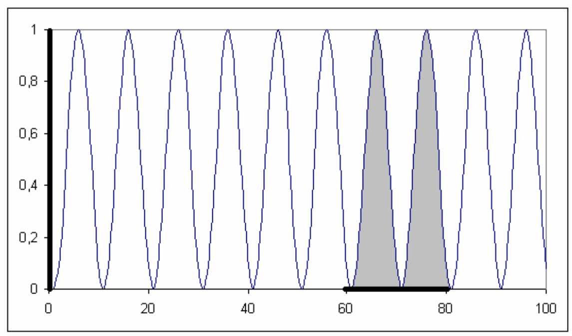

Fig. 5 Function 9

Having reached these conclusions, it is easy to propose a new random number generator, as long as we do not need the resulting pseudorandom sequences to follow a uniform distribution. To demonstrate this we propose the use of (10).

$$X_{n+1} = \sin(730 X_n + 12)^2 \quad (10)$$

Indeed, (10) drove correctly the chaos game. It makes sense to expect it to continue to do so as long as the number of transformations of the IFS are greatly less than $a = 730$. As the number of transformations becomes larger, we split $[0,1]$ in smaller areas and face the risk of not producing statistically independent sequences. Of course, the same holds even for the linear congruence, which would be a very poor random number generator if we selected a small parameter a.

6 Conclusion

In this paper we have proposed the use of the monte carlo approach to fractal image reconstruction from IFS models for the evaluation of the quality of pseudorandom generators. We have explained that this test is equivalent to the spectral test, which is the most reliable test for randomness to day, with the extra advantage that the approach proposed herein allows for a human perceivable 2D visualization of results.

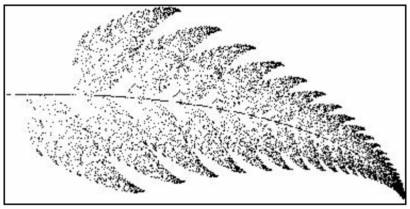

Fig. 6 Attractor using the linear congruence

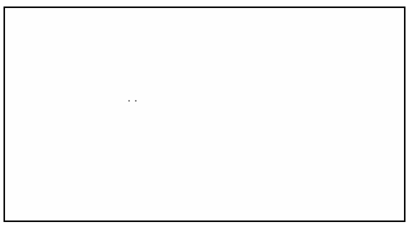

Fig. 7 Attractor using (6)

References:

[1] D.E. Knuth, *The art of computer programming*, vol 2, third edition, Addison—Wesley, 1997

[2] H.-O. Peitgen, H. Jurgens and D. Saupe, *Chaos & Fractals*, Springer-Verlang, 1992

[3] J. Ford, *How random is a coin toss?*, Physics Today, April 1983

[4] U.M. Maurer, *A universal statistical test for random bit generators*, Journal of cryptology 5 (1992), no. 2, 89{105.

[5] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.

[6] US DEPARTMENT OF COMMERCE - National Institute of Standards and Technology, *FIPS 140-1: Security requirements for Cryptographic Modules*, 1994.

[7] FIPS 140-2: *Security requirements for Cryptographic Modules (Draft)*, 1999.

[8] Barnsley M.F., Ervin V., Hardin D., Lancaster J., *Solution of an inverse problem for fractals and other sets*, Proc. Natl. Acad. Sci. Vol 83, pp. 1975-1977, 1986.