International Journal of Modern Physics C
Vol. 18, No. 5 (2007) 861–882

World Scientific
www.worldscientific.com

PSEUDO RANDOM NUMBER GENERATION WITH THE AID OF ITERATED FUNCTION SYSTEMS ON $\mathbb{R}^2$

P. BOUBOULIS

Department of Informatics and Telecommunications
Telecommunications and Signal Processing
University of Athens
Panepistimiopolis 157 84, Athens, Hellas
bouboulis@di.uoa.gr

Received 18 July 2006
Accepted 27 July 2006

Two new pseudorandom number generators, based on iterated function systems (IFS), are introduced. An IFS is created based on an arbitrary seed and a set is constructed using the deterministic iteration algorithm (DIA). From this set pseudo random numbers have been constructed. The generators have big periods and pass all major statistical tests, indicating that they can be used in any application requiring random numbers, such as cryptography.

Keywords: Fractal interpolation functions; iterated function systems; pseudo-random number generator.

PACS Nos.: 05.10.-a, 05.45.Df.

1. Introduction

Fractal theory has been drawing considerable attention of the researchers in various scientific areas. The application of fractals created by iterated function systems (IFS) in the area of image compression is probably the most famous one (see Refs. 3, 13, 11, 10, 7 and 8). Applications of fractal surfaces have, also, been found in computer graphics, image compression, Metallurgy, Geology, Chemistry, Medical sciences and several other areas, where there is great need to construct objects that are extremely complex (see Refs. 17, 20, 9 and 18).

An IFS is defined as a complete metric space X with a distance function ρ and a finite set of contractive mappings $\{w_i : X \to X, \text{for } i = 1, 2, \ldots, M\}$. IFSs have received a great deal of attention because they are capable of producing complicated and varied images (which we may consider to be of non-integer dimension) with as little as two maps. Fractal interpolation functions (FIFs) were introduced by Barnsley.[5,4] He used IFSs consisted of affine mappings, whose attractor is the graph

of a continuous function that interpolates given data points. In Refs. 15, 16 and 19. FIFs are used to represent discrete sequences.

Pseudo random number generators (PRNGs) are needed in various areas of computer science, such as cryptography, simulation, several numerical algorithms etc. The focus of this paper is to introduce a new method to generate pseudo random numbers, with the application of IFS theory. A fractal set, which is depended on a seed and it is composed of a number of integer values, is produced. A slight change of the seed produces an entirely different set, thus the emerging set of pseudo random numbers will, also, be diversed. In Sec. 2 the background theory is presented. Synthesis techniques are given for construction of this fractal set, in Sec. 3. Section 4 presents some statistical properties of the generators. Detailed algorithms describing the operation of the proposed PRNGs are presented in Sec. 5. One may use one of those PRNGs to encrypt any given discrete sequence of data points by applying a simple XOR operation. Finally, Sec. 6 presents the results of several statistical tests that were applied to a number of random bits produced by these PRNGs.

2. Background

2.1. *Iterated function systems*

As it was mentioned above, an hyperbolic IFS is defined as a pair of a complete metric space (X, ρ) together with a finite set of continuous contractive mappings $w_i : X \to X$, for $i = 1, 2, \ldots, M$, with respective contraction factors s_i. Note that if the mappings are not contractive the system is called simply IFS. Sometimes, in practice, the word "hyperbolic" is dropped. In this paper, however, it is important to distinguish the hyperbolic-contractive IFS (which has a unique fixed point-attractor) from the non-contractive one.

The attractor of a hyperbolic IFS is the unique set A for which $A = \lim_{k\to\infty} W^k(A_0)$ for every starting set A_0, where

$$W(A) = \bigcup_{i=1}^{N} w_i(A) \;\forall\; A \in \mathcal{H}(X)\,,$$

and $\mathcal{H}(X)$ is the metric space of all nonempty compact subsets of X with respect to the Hausdorff metric. The Hausdorff metric is defined by $h(A, B) = \max\{\min\{\rho(x, y) : y \in B\} : x \in A\}$, $\forall\ A,\ B \in \mathcal{H}(X)$. Using the compactness of A and B it can be shown that this definition is meaningful (see Ref. 5).

The attractor of the IFS may be constructed with either of two algorithms. The first, called the deterministic iteration algorithm or DIA for short, is implemented as follows. Let A_0 be any nonempty compact set (e.g., a single point). Then, we apply the maps to A_0 in sequential order to yield a new set A_1 (i.e., $A_1 = \bigcup_{i=1}^{N} w_i(A_0)$). Now, we temporarily save A_1, discard A_0 and apply the maps sequentially to produce $A_2 = \bigcup_{i=1}^{N} w_i(A_1)$ etc. The series of sets $\{A_0, A_1, A_2, \ldots\}$ converges to the attractor of the IFS. The second algorithm for construction of the attractor of an

IFS is the random iteration algorithm or RIA. Let x_0 be any point in X. From the set of maps, we choose one map at random and apply that map to x_0 to yield a new point x_1. Then we choose another map at random and apply that map to x_1 to produce the point x_2 etc. Under the condition that the probability of choosing a particular map is not allowed to be zero, the set $\{x_0, x_1, x_2, \ldots\}$ traces out the attractor. More information on these algorithms can be found in Ref. 5.

2.2. *Fractal interpolation functions*

Let $X = [0,1] \times \mathbb{R}$ and $\Delta = \{(x_i, y_i) : i = 0, 1, \ldots, N\}$ be an interpolation set with $N+1$ interpolation points such that $0 = x_0 < x_1 < \cdots < x_N = 1$. The interpolation points divide [0, 1] into N intervals $I_i = [x_{i-1}, x_i]$, $i = 1, \ldots, N$, which we call sections. We define N affine contractive mappings $w_i : X \to \mathbb{R}^2$ of the form

$$w_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & 0 \\ c_i & s_i \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}, \quad \text{for } i = 1, 2, \ldots, N\,. \tag{1}$$

Affine maps such as Eq. (1) are often called shear transformations. Vertical segments are mapped to vertical segments contracted by the factor s_i. The parameter s_i is called the contraction factor of the map w_i. It is easy to show that if $|s_i| < 1$ then there is a metric d equivalent to the Euclidean metric, such that w_i is a contraction (i.e., there is $\hat{s}_i : 0 \le \hat{s}_i < 1$ such that $d(w_i(\mathbf{x}), w_i(\mathbf{y})) \le \hat{s}_i d(\mathbf{x}, \mathbf{y})$, see Ref. 5). Each affine map w_i is constrained to map the endpoints of the set of the interpolation points Δ to the endpoints of the section I_i. That is,

$$w_i \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} x_{i-1} \\ y_{i-1} \end{pmatrix}, \qquad w_i \begin{pmatrix} x_N \\ y_N \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix}, \quad \text{for } i = 1, 2, \ldots, N\,. \tag{2}$$

From Eq. (2), four linear equations arise which can always be solved for a_i, c_i, e_i, and f_i in terms of the coordinates of the interpolation points and the vertical scaling factor s_i. Thus, once the contraction factor s_i for each map has been chosen, the remaining parameters may be computed by

$$a_i = \frac{x_i - x_{i-1}}{x_N - x_0} \tag{3}$$

$$e_i = \frac{x_N x_{i-1} - x_0 x_i}{x_N - x_0} \tag{4}$$

$$c_i = \frac{y_i - y_{i-1}}{x_N - x_0} - s_i \frac{y_N - y_0}{x_N - x_0} \tag{5}$$

$$f_i = \frac{x_N y_{i-1} - x_0 y_i}{x_N - x_0} - s_i \frac{x_N y_0 - x_0 y_N}{x_N - x_0}\,. \tag{6}$$

The IFS $\{X, w_{1-N}\}$ has a unique fixed point (attractor) A, which is the graph of a continuous function f ($f : [0,1] \to \mathbb{R}$) that interpolates the points of Δ. The function f is called fractal interpolation function (FIF).

Next the definition of the box counting fractal dimension is given. A fractal dimension is a number that "measures" the density of the space that a fractal occupies. While there are several fractal dimensions in use, the box counting fractal dimension is certainly the most popular one. Consider $A \in \mathcal{H}(X)$ and let $\mathcal{N}(G, \epsilon)$ denote the smallest number of closed balls of radius ϵ needed to cover A. Then if

$$D = D(A) = \lim_{\epsilon \to 0} \left\{ \frac{\log(\mathcal{N}(A, \epsilon))}{\log(1/\epsilon)} \right\}$$

exists, it is called the box counting fractal dimension of A. The box counting fractal dimension of the graph A of the FIF, defined above, is the unique real solution D of

$$\sum_{n=1}^{N} |s_n| a_n^{D-1} = 1\,, \tag{7}$$

if $\sum_{n=1}^{N} |s_i| > 1$ and the interpolation points do not all lie on a single straight line. Otherwise the box counting dimension of A is one. In the case where the interpolation points are equally spaced, it follows that $a_n = 1/N$, $n = 1, 2, \ldots, N$, hence Eq. (7) becomes

$$D = 1 + \frac{\log(\sum_{n=1}^{N} |s_n|)}{\log(N)}\,. \tag{8}$$

Notice that $\sum_{n=1}^{N} |s_i| < N$. Thus the dimension of a FIF is always less than two. However, we can make it arbitrarily close to two by choosing scaling factors s_i close to one (or -1).

2.3. *Hidden variable fractal interpolation functions*

Let $X = [0, 1] \times \mathbb{R}^2$ and $\Delta = \{(x_i, y_{1,i}, y_{2,i}) : i = 0, 1, \ldots, N\}$ be an interpolation set with $N + 1$ interpolation points such that $0 = x_0 < x_1 < \cdots < x_N = 1$. The interpolation points divide $[0, 1]$ into N intervals $I_i = [x_{i-1}, x_i]$, $i = 1, \ldots, N$, which we call sections. We define N affine contractive mappings $w_i : X \to \mathbb{R}^3$ of the form

$$w_i \begin{pmatrix} x \\ y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} a_i & 0 & 0 \\ c_i & s_{11,i} & s_{12,i} \\ e_i & s_{21,i} & s_{22,i} \end{pmatrix} \cdot \begin{pmatrix} x \\ y_1 \\ y_2 \end{pmatrix} + \begin{pmatrix} b_i \\ d_i \\ f_i \end{pmatrix}\,, \quad \text{for } i = 1, 2, \ldots, N\,. \tag{9}$$

It is easy to show that if the spectral radius $\rho(S_i)$ of the matrix

$$S_i = \begin{pmatrix} s_{11,i} & s_{12,i} \\ s_{21,i} & s_{22,i} \end{pmatrix}$$

is lower than one, then there is a metric equivalent to Euclidean metric, such that w_i is a contraction (see Ref. 6). (If $|s_{jk,i}| < 1$, $j, k = 1, 2$, then $\rho(S_i) < 1$ for $i = 1, \ldots, N$).

Each affine map w_i is constrained to map the endpoints of the set of the interpolation points Δ to the endpoints of the section I_i. That is,

$$w_i \begin{pmatrix} x_0 \\ y_{1,0} \\ y_{2,0} \end{pmatrix} = \begin{pmatrix} i-1 \\ y_{1,i-1} \\ y_{2,i-1} \end{pmatrix}, \qquad w_i \begin{pmatrix} x_N \\ y_{1,N} \\ y_{2,N} \end{pmatrix} = \begin{pmatrix} x_i \\ y_{1,i} \\ y_{2,i} \end{pmatrix}, \quad \text{for } i = 1, 2, \ldots, N\,. \tag{10}$$

From Eq. (10), four linear equations arise, which can always be solved for a_i, b_i, c_i, d_i, e_i and f_i in terms of the coordinates of the interpolation points and the scaling factors $s_{11,i}$, $s_{12,i}$, $s_{21,i}$, and $s_{22,i}$. Thus, once the four contraction factors for each map have been chosen, the remaining parameters may be computed by

$$a_i = \frac{x_i - x_{i-1}}{x_N - x_0} \tag{11}$$

$$b_i = \frac{x_N x_{i-1} - x_0 x_i}{x_N - x_0} \tag{12}$$

$$c_i = \frac{y_{1,i} - y_{1,i-1}}{x_N - x_0} - s_{11,i}\frac{y_{1,N} - y_{1,0}}{x_N - x_0} - s_{12,i}\frac{y_{2,N} - y_{2,0}}{x_N - x_0} \tag{13}$$

$$e_i = \frac{y_{2,i} - y_{2,i-1}}{x_N - x_0} - s_{21,i}\frac{y_{1,N} - y_{1,0}}{x_N - x_0} - s_{22,i}\frac{y_{2,N} - y_{2,0}}{x_N - x_0} \tag{14}$$

$$d_i = y_{1,0} - c_i x_0 - s_{11,i} y_{1,0} - s_{12,i} y_{2,0} \tag{15}$$

$$f_i = y_{2,0} - e_i x_0 - s_{21,i} y_{1,0} - s_{22,i} y_{2,0}\,. \tag{16}$$

The hidden variable IFS (HVIFS) $\{X, w_{1-N}\}$ has a unique fixed point (attractor) A, which is the graph of a continuous function f ($f : [0,1] \to \mathbb{R}^2$) that interpolates the points of Δ. The function f is called hidden variable fractal interpolation function (HVFIF).

2.4. *Non hyperbolic IFSs*

While fractal interpolation functions and hidden variable fractal interpolation functions have been studied extensively (see Refs. 1, 12, 2, 4, 5, 14 and 6), the study of non-hyperbolic affine IFSs on $\mathbb{R}^2$ is very poor. Such an IFS does not have, in general, a fixed point (attractor). The set A_n after n iterations will depend on the initial set and the parameters of each map.

Consider $X = [0,1] \times \mathbb{R}$ and affine maps w_i, $i = 1, 2, \ldots, N$, of the form given by Eq. (1) with $s_i > 1$, $a_i < 1$, $i = 1, \ldots, N$. Let also $(x^{(0)}, y^{(0)})$ be any point of X and $I = \{i_1, i_2, \ldots, i_n, \ldots : i_k \in \{1, 2, \ldots, N\}\}$ an arranged set defining the "*path*" that this point "follows" after the application of the RIA or DIA (i.e., first we apply map w_{i_1}, then w_{i_2} etc.). Then, if $(x^{(n)}, y^{(n)})$ is the produced point at the nth step of the construction algorithm (with initial point $(x^{(0)}, y^{(0)})$), after the application of the map w_{i_n} we have:

$$
\begin{aligned}
x^{(1)} &= a_{i_1} x^{(0)} + b_{i_1} \\
y^{(1)} &= M_1 + s_{i_1} y^{(0)}\,, \quad \text{where } M_1 = c_{i_1} x^{(0)} + d_{i_1} \\
x^{(2)} &= a_{i_2} x^{(1)} + b_{i_2} \\
y^{(2)} &= M_2 + s_{i_2} y^{(1)} \\
&= M_2 + s_i M_1 + s_{i_1} s_{i_2} y^{(0)}\,, \quad \text{where } M_2 = c_{i_2} x^{(1)} + d_{i_2} \\
&\;\;\vdots \\
x^{(n)} &= a_{i_n} x^{(n-1)} + b_{i_n} \\
y^{(n)} &= M_n + s_{i_n} M_{n-1} + \cdots + \prod_{k=1}^{n} s_k y^{(0)}\,, \text{where } M_n = c_{i_n} x^{(n)} + d_{i_n} \\
y^{(n)} &= \prod_{k=1}^{n} s_{i_k} \left(\frac{M_n}{\prod_{k=1}^{n} s_{i_k}} + \frac{M_{n-1}}{\prod_{k=1}^{n-1} s_{i_k}} + \cdots + y^{(0)} \right)
\end{aligned}
$$

In this case the following Lemma stands.

Lemma 1. *Let $(x^{(0)}, y^{(0)})$ and $(x^{(0)}, z^{(0)})$ be two initial points of X. If we apply the affine maps w_i, $i = 1, 2, \ldots, N$ to these points using any path I, then*

$$\lim_{n\to\infty} |y^{(n)} - z^{(n)}| = +\infty\,.$$

Proof. At the nth step of the RIA we will have:

$$y^{(n)} = M_n + s_{i_n} M_{n-1} + \cdots + \prod_{k=1}^{n} s_{i_k} y^{(0)}\,,$$

$$z^{(n)} = M_n + s_{i_n} M_{n-1} + \cdots + \prod_{k=1}^{n} s_{i_k} z^{(0)}\,,$$

where

$$M_n = c_{i_n} x^{(n)} + d_{i_n}\,.$$

Thus, if $s_m = \min\{|s_i|, i = 1, 2, \ldots, N\}$,

$$
\begin{aligned}
|y^{(n)} - z^{(n)}| &= \prod_{k=1}^{n} |s_{i_k}| |y^{(0)} - z^{(0)}| \\
&> s_m^n |y^{(0)} - z^{(0)}|\,.
\end{aligned}
$$

Hence, since $s_m > 1$ we have the result. □

The above lemma ensures that the affine IFS considered (with all $|s_i| > 1$) does not have a fixed point. In fact, we see that for arbitrary small variations of the initial set used with either the RIA or the DIA, the emerged sets at the nth step will be far apart. This observation will be used later on. (In fact, the lemma implies that if, under a fixed path, the point $(x^{(0)}, y^{(0)})$ produces bounded values (i.e., $|y^{(n)}| < M, \forall\ n \in \mathbb{N}$), then for every other point $(x^{(0)}, z^{(0)})$ (with $z^{(0)} \neq y^{(0)}$) $\lim_{n\to\infty} |z^{(n)}| = +\infty$.)

For the the hidden variable case, the spectral radius $\rho(S_i)$ of the matrix

$$S_i = \begin{pmatrix} s_{11,i} & s_{12,i} \\ s_{21,i} & s_{22,i} \end{pmatrix}$$

must be greater than one ($\forall\ i = 1, 2, \ldots, N$), so that the IFS is non contractive. Thus, if one chooses $s_{jk,i} > 1$, $j,\ k = 1,\ 2,\ i = 1, \ldots, N$, the IFS will be certainly non contractive and similar results as in Lemma 1 will apply.

3. Construction of PRNGs Based on a Seed

Loosely speaking, pseudorandom generators are efficient deterministic programs which expand short randomly selected seeds into much longer pseudorandom bit sequences. They are defined as computationally indistinguishable from truly random sequences by efficient algorithms and needed in various areas of computer science. Applications in cryptography and simulation are widely known. Here, a new method of producing pseudorandom sequences using iterated function systems is introduced. First, we use contractive IFSs, which produce sequences with relatively "bad" statistical properties and then we use non contractive IFSs which we demonstrate that they have "very good" properties.

The traditional approach to a PRNG is that it has a state that evolves in a finite state space S, according to a recurrence of the form $s_n = f(s_{n-1}, s_{n-2}, \ldots, s_{n-k})$, $n > k$, (usually $k = 1$) where the initial state $s_0 \in S$ is called the seed and the function $f : S \to S$ is called the transition function. At step n, the generator outputs $u_n = g(s_n)$, where $g : S \to \Psi$ is called the output function. Ψ is a finite space such as $\{0, 1\}$ (at step n the generator outputs a bit) or $\{0, 1, 2, \ldots, 255\}$.

However the approach discussed here is somewhat different. The seed is a set (C) of $N+1$ numbers which we use to form an IFS. This IFS takes an initial set A_0 and "expands" it using the DIA. Thus, at each step a new set of numbers emerges. After K steps the set A_K consists of $N^{K+1} + 1$ numbers, which we use to produce the output of the PRNG (i.e., the least significant bit of each number). The quality of the pseudorandom sequence constructed, depends on the IFS one uses, therefore the seed should be chosen as random as possible.

3.1. *A PRNG constructed from a FIF*

First, some definitions are given.

Definition 1. Consider $x \in \mathbb{R}$ and $P \in \mathbb{N}$. Then there is $k \in \mathbb{Z}$, such that $kP \leq x < (k+1)P$. We define

$$y \equiv x(\text{mod } P) := x - kP \in [0, P)\,.$$

Definition 2. Consider $x \in \mathbb{R}$. Then there is $n \in \mathbb{Z}$, such that $n \leq x < n+1$. We define

$$[x] = n\,.$$

Let $C = \{c_0, c_1, \ldots, c_N\}$ be a seed consisted of $N+1$ integer numbers, where $0 \leq c_i \leq 255$, $c_i \in \mathbb{N}$, for $i = 0, 1, \ldots, N$. We take the interpolation points as follows:

$$x_i = S \cdot i\,, \qquad y_i = c_i \quad \text{for } i = 0, 1, \ldots, N$$

with

$$S = \sum_{i=0}^{N} c_i\,.$$

Then we choose arbitrary contractions factors s_i and we construct the attractor A (which is the graph of a continuous fractal interpolation function f that interpolates the data) using the deterministic algorithm. Usually, we choose as initial set A_0 the set of the interpolation points. Note that the interpolation points should not be collinear with (x_0, y_0) and (x_N, y_N) (in this case the attractor is the straight line connecting those two points). The seed should be chosen appropriately. The produced attractor consists of $M+1$ points in $\mathbb{R}^2$, thus $A = \{(\tilde{x}_j, \tilde{y}_j), \text{for } j = 0, 1, \ldots, M\}$ with $\tilde{x}_0 < \tilde{x}_1 < \tilde{x}_2 \cdots < \tilde{x}_M$. The number M depends on the number of iterations that we use to form the attractor ($M = N^{K+1} + 1$, where K the number of iterations). We limit our interest to the y coordinates of the graph of f. Let $E_f = \{\tilde{y}_j, \text{where}(\tilde{x}_j, \tilde{y}_j) \in A\}$ be an arranged set. We define $E_f[j] = y_j$, the jth element of the set for $j = 0, 1, \ldots, M$. It is evident that f depends on the set C. A different seed C' will produce another FIF f'. The problem is that if the sets C and C' are similar then E_f and $E_{f'}$ will be very "close" (see Fig. 1).

The above construction produces functions that change values rapidly and are non predictable. The problem is that two FIFs constructed using similar interpolation points (and similar contraction factors) look alike. Thus, if two seeds have a lot of numbers equal to one another, the produced FIFs will have almost common parts. Therefore, a method of producing FIFs, that even the slightest change of the seed will lead to an entirely different FIF, is needed. In order to achieve this, one must choose the interpolation points in such a way, that small variations of the initial seed will have significant effect on the values of the interpolation points. Thus, another FIF is constructed as follows. We take

$$x_i = S * i\,,$$

$$y_i = c_{\phi(i)} + c_{\phi(i+1)} \cdot 256 + \cdots + c_{\phi(i+L-1))} \cdot 256^{L-1}\,,$$

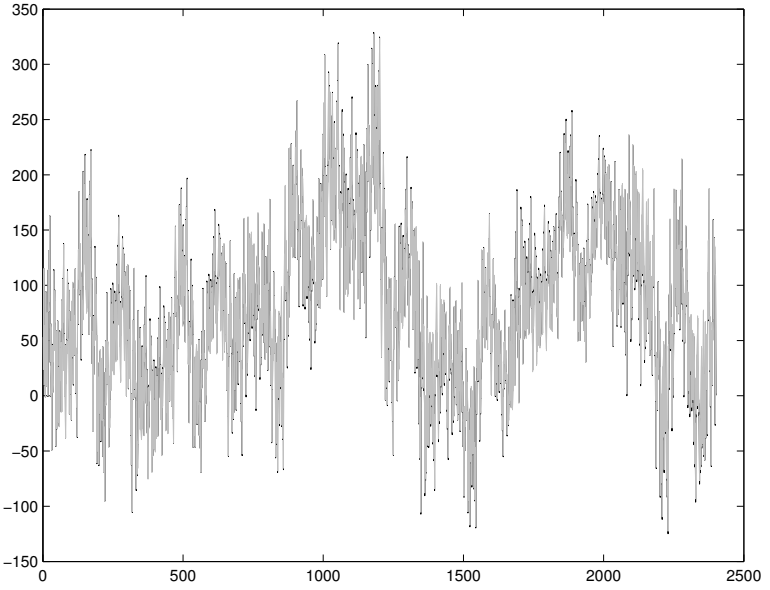

Fig. 1. Construction of two FIFs using the codes $C = \{52, 6, 49, 205, 8, 92, 134, 206\}$ (black line) and $C' = \{53, 6, 49, 205, 9, 91, 134, 200\}$ (gray line). In both cases the contraction factors that have been used are 0.8, 0.7, −0.8, 0.9, −0.7, 0.6, −0.9. For the computation of the attractor we used the DIA (three iterations, 2402 points). In the figure only the sets E_f and $E_{f'}$ are shown. The x axes corresponds to the index j of each point. It is obvious that the two graphs are extremely close.

where

$$\phi(j) = j \bmod (N+1)\,, \qquad L > 0\,, \qquad L \in \mathbb{N}\,.$$

The same construction as before is used and the attractor B of this IFS is produced, which is the graph of a FIF g and it consists of $M+1$ points. Then we produce the arranged set E_g as above. Now even the slightest change of the seed C will produce a very different attractor B' leading to a very different arranged set $E_{g'}$. However, it must be ensured that the interpolation points will not be collinear, since in that case the attractor will be a straight line. Note that the contraction factors should be close to 1 (or −1), so that the FIF will be as "rougher" as possible. The value of L is been chosen a priori. This value should be as bigger as possible, but smaller than $N+1$. The value of L depends, also, from the floating point representation of the computer used to construct the FIF. For the commonly used 64 bit representation, values of L greater than five will lead to overflow errors. However, it is crucial for L to take a value as large as possible, since the constructed FIF depends, mainly on the interpolation points. Thus larger variations of the interpolation points will lead to FIFs that are more complex. The use of a machine with 128 bit floating point representation (or more) is more preferable.

Having the FIF described above, one can easily construct a set of random bits as follows:

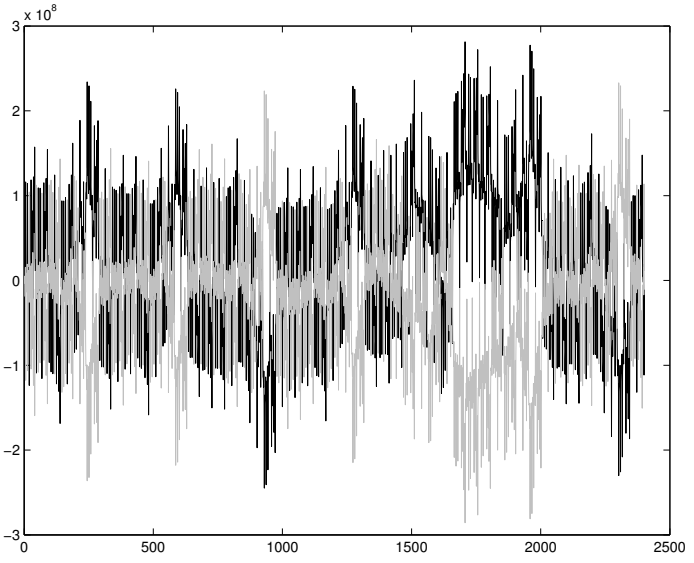

Fig. 2. Construction of two sets E_g (black line) and $E_{g'}$ (gray line) using the codes $C = \{52, 6, 49, 205, 8, 92, 134, 206\}$ and $C' = \{53, 6, 49, 205, 8, 92, 134, 206\}$. In both cases the contraction factors that have been used are 0.95, 0.9, −0.95, 0.9, −0.95, 0.95, −0.94, 0.92 and $L = 4$. For the computation of the attractor we used the DIA (three iterations, 2402 points). The x axes corresponds to the index j of each point. It is obvious that even though C and C' are almost identical the sets E_g and $E_{g'}$ are very different.

$$FPRNG_1(C) = \{|[E_g[j]]| \bmod 2; j = 1, 2, \ldots, M+1\}\,.$$

The corresponding PRNG will be referred to as affine FIF PRNG.

3.2. *A PRNG constructed from a non hyperbolic affine IFS*

Consider the affine IFS, as defined above, but now let each $|s_i|$, $i = 1, 2, \ldots, N$ be greater than one. In this case (as shown above), there is not attractor for the IFS. However one can modify the deterministic algorithm to produce a set of points from this IFS. Let, again,

$$\begin{aligned} x_i &= S * i\,, \\ y_i &= c_{\phi(i)} + c_{\phi(i+1)} \cdot 256 + \cdots + c_{\phi(i+L-1)} \cdot 256^{L-1} \\ \phi(j) &= j \bmod (N+1)\,, \qquad L > 0\,, \qquad L \in \mathbb{N} \end{aligned}$$

and $P = 256^L$. The initial set $B_0 = A_0 = \{\mathbf{x}_i^0 = (x_i, y_i); i = 0, 1, \ldots, N\}$ is (as always) the set of the interpolation points. The maps w_i, $i = 1, 2, \ldots, N$, are applied to B_0, in sequential order, to yield a new set $A_1 = \{\mathbf{x}_k^1 = (x_k^1, y_k^1); k = 0, 1, \ldots, N^2\}$. Then, the set

$$B_1 := \{\mathbf{y}_k^1 = (x_k^1, \hat{y}_k^1);\ \text{where } \hat{y}_k^1 = y_k^1 \bmod P, k = 0, 1, \ldots, N^2\}\,,$$

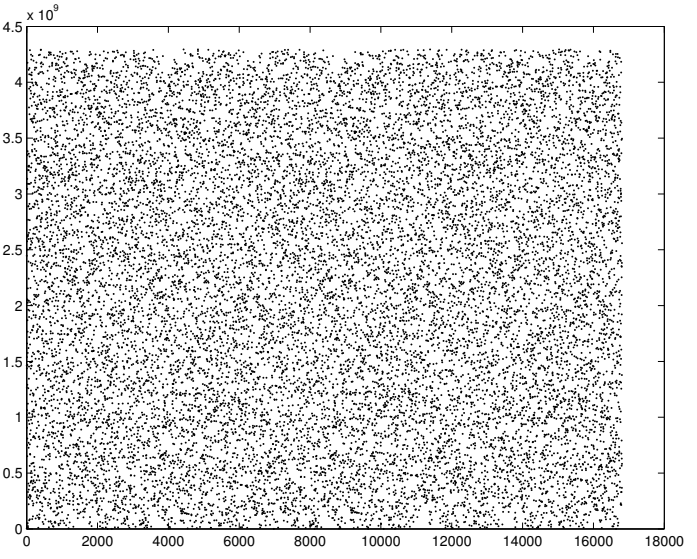

Fig. 3. Construction of a set using the non hyperbolic affine IFS using the codes $C = \{52, 6, 49, 205, 8, 92, 134, 206\}$ and the contraction factors 30.95, 30.9, −30.95, 30.9, −30.95, 30.95, −30.94, −33.92. For the computation of the attractor we used the Modulus Deterministic Algorithm (four iterations) and $L = 4$. The x axes corresponds to the index j of each point. It is obvious that the points are scattered throughout the plane.

is constructed and the maps are applied to B_1 to yield A_2, which in the sequel is used to construct B_2, etc. After K iterations a bounded set B_K will emerge, from which a set of random bits is produced as follows

$$FPRNG_2(C) = \{[\hat{y}_k^K \bmod 2]; k = 0, 1, \ldots, N^{K+1}\}\,.$$

3.3. *A PRNG constructed from a non hyperbolic affine hidden variable IFS*

Analogously to the above construction, consider two seeds

$$C_1 = \{c_{1,0}, c_{1,1}, \ldots, c_{1,N}\} \quad \text{and} \quad C_2 = \{c_{2,0}, c_{2,1}, \ldots, c_{2,N}\}$$

and let

$$\begin{aligned} x_i &= S * i\,, \\ y_{1,i} &= c_{1,\phi(i)} + c_{1,\phi(i+1)} \cdot 256 + \cdots + c_{1,\phi(i+L-1))} \cdot 256^{L-1}\,, \\ y_{2,i} &= c_{2,\phi(i)} + c_{2,\phi(i+1)} \cdot 256 + \cdots + c_{2,\phi(i+L-1))} \cdot 256^{L-1}\,, \end{aligned}$$

where

$$\phi(j) = j \bmod (N+1)\,, \qquad L > 0\,, \qquad L \in \mathbb{N}$$

and $P = 256^L$. The initial set $B_0 = A_0 = \{\mathbf{x}_i^0 = (x_i, y_{1,i}, y_{2,i}); i = 0, 1, \ldots, N\}$ is the set of the interpolation points. The maps w_i, $i = 1, 2, \ldots, N$, are applied to B_0, in sequential order, to yield a new set $A_1 = \{\mathbf{x}_k^1 = (x_k^1, y_{1,k}^1, y_{2,k}^2); k = 0, 1, \ldots, N^2\}$. Then, the set

$$B_1 := \{\mathbf{y}_k^1 = (x_k^1, \hat{y}_{1,k}^1, \hat{y}_{2,k}^1)\,;$$

where

$$\hat{y}_{j,k}^1 = y_{j,k}^1 \bmod P, \hat{y}_{j,k}^2 = y_{j,k}^2 \bmod P, k = 0, 1, \ldots, N^2, j = 1, 2\}$$

is constructed and the maps are applied to B_1 to yield A_2 which, in turn, is used to construct B_2, etc. After K iterations, a bounded set B_K is obtained, from which a set of random bits is created as follows

$$FPRNG_2(C) = \{[\hat{y}_{1,k}^K \bmod 2], [\hat{y}_{2,k}^K \bmod 2]; k = 0, 1, \ldots, N^{K+1}\}\,.$$

4. Statistical Properties

The random bits produced by the two latter PRNGs are uniformly distributed over $\{0, 1\}$. The following well known facts ensure this assertion.

Proposition 1. *Let X be a random variable uniformly distributed over $[0, P)$ (where $P \in \mathbb{N}$).*

(1) *If $Q \in \mathbb{N}$ such that $P \bmod Q = 0$, then $Y = X \bmod Q$ is another random variable uniformly distributed over $[0, Q)$.*
(2) *If $W_P : [0, P) \to [0, P) : W_P(x) = s \cdot x + b (\bmod P)$, $s \in \mathbb{Z}$, $b \in \mathbb{R}$. Then $W_P(X)$ is another random variable uniformly distributed over $[0, P)$.*
(3) *$[X]$ is another random variable uniformly distributed over $\{0, 1, 2, \ldots, P - 1\}$.*
(4) *If $P = 2^\lambda$, $\lambda \in \mathbb{N}$ and $\kappa \in \mathbb{N}$ $(\kappa < \lambda)$, then $[X (\bmod 2^\kappa)] = [X] \bmod 2^\kappa$ is another random variable uniformly distributed over $\{0, 1, 2, \ldots, 2^\kappa - 1\}$.*

Proposition 2. *Let X, Y be independent random variables uniformly distributed over $[0, P)$. Consider, also, a map $W_P : [0, P) \to [0, P) : W_P(x) = s \cdot x + b (\bmod P)$, $s \in \mathbb{Z}$, $b \in \mathbb{R}$. Then $W_P(X)$, $W_P(Y)$ are independent random variables uniformly distributed over $[0, P)$.*

Proposition 1 implies that if the seed $C = \{c_0, c_1, \ldots, c_N\}$ is a sequence of numbers which are uniformly distributed over an interval $[0, P)$ and the contraction factors of all maps are integers (i.e., $s_i \in \mathbb{Z}$, $i = 1, 2, \ldots, N$), then the numbers produced at each iteration will, also, be uniformly distributed over $[0, P)$. Furthermore, if $P \bmod 2 = 0$, then the bits produced after K steps will be uniformly distributed over $\{0, 1\}$.

Proposition 2 provides some properties about the relation of two sets of numbers produced by two different seeds. If the seeds are independent, then the produced sets will contain bits uniformly distributed which will, also, be independent.

5. Algorithms

In this section, two algorithms are presented, that may be used to generate pseudo random numbers using IFSs. The algorithms are based on the construction described in Sec. 3, which has been slightly modified for practical reasons.

A. The Modulus Deterministic Algorithm (MDA)

This algorithm takes an arbitrary seed, forms an IFS and an initial set and computes random bytes by applying the modified deterministic algorithm to the initial set. In each step it stores the points of the produced set to memory and uses them to produce the set for the next step.

- Choose a seed $\Gamma = \{\gamma_0, \gamma_1, \ldots, \gamma_N\}$, $0 \le \gamma_i \le 255$, $i = 0, 1, \ldots, N$. Choose, also, the number of iterations K. (The algorithm constructs $N^{K+1} + 1$ bytes).
- From the seed, construct the set $C = \{c_0, c_1, \ldots, c_N\}$, $c_i = \gamma_i$, $i = 0, 1, \ldots, N$ and the contraction factors $\{s_1, s_2, \ldots, s_N\}$ such that $|s_i| > 1$, $i = 1, 2, \ldots, N$ (for the statistical testing the equation $s_i = (-1)^{\gamma_i} \cdot [\gamma_i/4 + 5]$ was used).
- For $l = 1$ to 8 do
  - Form the IFS as described in Sec. 3 using the points:

$$x_i = S * i, S = \sum_{i=0}^{N} c_i$$

$$y_i = c_{\phi(i)} + c_{\phi(i+l)} \cdot 256 + \cdots + c_{\phi(i+l\cdot(L-1))} \cdot 256^{L-1}$$

$$\phi(j) = j \bmod (N+1), \qquad L > 0, \qquad L \in \mathbb{N}$$

    Consider $P = 256^L$ and the initial set $A_0 = \{(x_i, y_i), i = 0, 1, \ldots, N\}$.
  - Set $B_0 = A_0$.
  - For j =1 to K do:
    (a) Produce the set A_j by applying the maps of the IFS in sequential order to B_{j-1}.
    (b) Next produce the set $B_j = \{\mathbf{y}_k^j = (x_k^j, \hat{y}_k^j); \text{where } \hat{y}_k^j = y_k^j \bmod P, k = 0, 1, \ldots, N^{j+1}\}$.
  - Form the set $\Omega_l = \{\omega_{l,m} = [\hat{y}_m^K] \bmod 256, m = 0, 1, \ldots, N^{K+1}\}$. This set consists of $N^{K+1} + 1$ integer (1-byte) numbers.
  - Remove from Ω_l the points of the initial set. (This is done to remove any explicit information for the seed)
- Compute the set $\Omega = \Omega_1 \oplus \Omega_2 \oplus \cdots \oplus \Omega_8 = \{\omega_m = \omega_{m,1} \oplus \cdots \oplus \omega_{m,8}, m = 0, \ldots, N^{K+1} - N - 1 = M_0\}$, where $\oplus$ is the XOR operator.
- Set PRNG = Ω.

Note that, as stated above, in each step all the produced points are used to calculate new ones. This is the main advantage of the proposed scheme but, at the same time, its main drawback, from the implementation point of view. In the Kth

step of the iteration, the set B_K consists of $N^{K+1}+1$ points. In order to compute the set B_{K+1}, all the points of B_K need to be stored into memory. For values of K greater than six, the amount of memory required becomes prohibitive for practical purposes, even if the value of N is less than 16. So, one cannot construct arbitrarily large sets, due to memory limitations. The only alternative is to store the points, that are produced at each step, into the hard disk instead of the memory. However, this idea doesn't solve the problem entirely. The number of points produced at each step increases exponentially.

The following algorithm describes some modifications of the construction of the affine IFS PRNG, which allow us to overcome this problem. The algorithm constructs random 1-byte integers. It takes a seed consisted of $N+1$ 1-byte integers, it forms an IFS and an initial set A_0^1 and then computes several points using the modified deterministic algorithm. Then, it extracts from those points a new initial set A_0^2 and using the same IFS computes several new points, e.t.c. Lemma 1 ensures that different initial sets will produce very different points.

B. The Recycled Modulus Deterministic Algorithm (RMDA)

- Choose a seed $\Gamma = \{\gamma_0, \gamma_1, \ldots, \gamma_N\}$, $0 \leq \gamma_i \leq 255$, $i = 0, 1, \ldots, N$. Choose, also, the number of iterations K.
- Set PRNG $= \oslash$.
- From the seed construct the sets $C = \{c_0, c_1, \ldots, c_N\}$, $c_i = \gamma_i$, $i = 0, 1, \ldots, N$, $D = \{\delta_0, \delta_1, \ldots, \delta_N\}$, $d_i = \gamma_i$, $i = 0, 1, \ldots, N$ and $E = \{\epsilon_0, \epsilon_1, \ldots, \epsilon_N\}$, $e_i = \gamma_i$, $i = 0, 1, \ldots, N$.
- **Do:**
  - For $l = 1$ to 8 do
    (a) Form the IFS using the points:

$$x_i = S * i, S = \sum_{i=0}^{N} c_i$$

$$y_i = c_{\phi(i)} + c_{\phi(i+l)} \cdot 256 + \cdots + c_{\phi(i+l\cdot(L-1))} \cdot 256^{L-1}$$

$$\phi(j) = j \bmod (N+1), \qquad L > 0, \qquad L \in \mathbb{N}$$

and the contraction factors $\{s_1, s_2, \ldots, s_N\}$ such that $|s_i| > 1$, $i = 1, 2, \ldots, N$ (for the statistical testing the equation $s_i = (-1)^{\epsilon_i} \cdot [\epsilon_i/4 + 5]$ was used). Consider $P = 256^L$ and the initial set $A_0 = \{(x_i, z_i), i = 0, 1, 2, \ldots, N\}$, where

$$x_i = S * i, S = \sum_{i=0}^{N} c_i$$

$$z_i = \delta_{\phi(i)} + \delta_{\phi(i+1)} \cdot 256 + \cdots + \delta_{\phi(i+(L-1))} \cdot 256^{L-1}$$

$$\phi(j) = j \bmod (N+1), \qquad L > 0, \qquad L \in \mathbb{N}$$

(b) Set $B_0 = A_0$.
(c) For $j = 1$ to K do:

- Produce the set A_j by applying the maps of the IFS in sequential order to B_{j-1}.
- Next produce the set $B_j = \{\mathbf{y}_k^j = (x_k^j, \hat{y}_k^j); \text{where } \hat{y}_k^j = y_k^j \bmod P, k = 0, 1, \ldots, N^{j+1}\}$.

(d) Form the set $\Omega_l = \{\omega_{l,m} = [\hat{y}_m^K] \bmod 256, m = 0, 1, \ldots, N^{K+1}\}$. This set consist of $N^{K+1} + 1$ integer (1-byte) numbers.
(e) Remove from Ω_l the points of the initial set. (This is done to remove any explicit information for the seed).

— Compute the set $\Omega = \Omega_1 \oplus \Omega_2 \oplus \cdots \oplus \Omega_8 = \{\omega_m = \omega_{m,1} \oplus \cdots \oplus \omega_{m,8}, m = 0, \ldots, N^{K+1} - N - 1 = M_0\}$.
— Set PRNG = PRNG $\cup\Omega$.
— Extract $2N + 2$ points from Ω and set $N + 1$ of them as the new set D (which is used to form the initial set A_0 and the rest $N + 1$ of them as the new set E (which is used to compute the contraction factors). Choose the points $\{\omega_{(\lambda-1)M_0/(N+1)+M_0/2N+2}, \lambda = 1, 2, \ldots, N + 1\} = D$ and $\{\omega_{(\lambda-1)M_0/(N+1)+M_0/2N+2+1}, \lambda = 1, 2, \ldots, N + 1\} = E$.

Until the set PRNG has more elements than those required.

Note, that in both algorithms N should be chosen so that $N + 1$ is a prime number. In this case the calculation of each interpolation point y_i will depend on L different numbers of the seed. The algorithms produce eight sequences of pseudo random 1-byte numbers and then combine them (using a XOR operation) to create a single sequence. The emerged sequence will, also, contain uniformly distributed bits (Lemma 1 for $k = 8$). This approach is somewhat different than the one described in Sec. 3. This one is selected to add more complexity. However, both technics work well and give sequences with similar properties.

The first algorithm describes a PRNG with period of $+\infty$ (at least in theory) since there is no recurring pattern involved with the IFS (keep in mind, however, that computers are finite-state machines). The second algorithm describes a PRNG with maximum period $(N^K - N) \cdot 256^{(N+1)}$ (after that, the seeds will repeat themselves). Remember that, in both algorithms, the seed should be chosen such that the emerging interpolation points won't be collinear. Similar algorithms may be easily constructed to produce random 1-byte integers using the affine HVIFS scheme. Here a seed, consisted of $2N + 2$ 1-byte integers is needed. One should note (as stated earlier) that for each chosen seed the produced sequence of random numbers is different and depends largely of the seed's numbers. A trivial selection of the seed (all points collinear) will lead to a trivial PRNG. The number of iterations, also, has to be chosen as larger as possible.

These PRNGs may be used in any application requiring random numbers. In particular, one may use these PRNGs to encipher any given file using a simple XOR

operation (stream-cipher like the one time pad). A seed is chosen by the user (the password) and based on this seed enough pseudo random numbers are generated using the above algorithms. These numbers are used to encipher the requested file using a simple XOR operation. If the user wants to decrypt the file, he enters the password, the same pseudo random numbers are generated and after the application of a XOR operation to the enciphered file he gets the original file.

5.1. *Analysis*

PRNGs constructed from fractal interpolation functions are expected to have poor statistical properties. The values of such PRNGs lie on the graph of a continuous function and therefore, after a few iterations, the values are expected to be highly correlated. On the other hand, non hyperbolic IFS may produce numbers that are completely uncorrelated. In addition, there is no known method (except the collage theorem see, Ref. 5) that can find the parameters needed to construct an hyperbolic IFS that converges to a known attractor. The collage theorem may be applied to find the parameters of an hyperbolic IFS, only if one knows the whole attractor. Thus, if one uses only some of the first points of a FIF PRNG, he provides no significant information to an adversary that tries to figure out the map parameters using these points. In fact, even the whole set of points constructed by the PRNG cannot give any important information, since it does not contain the actual points of the attractor but only some of the least significant bits of their integer parts. In the case of the non hyperbolic IFS PRNGs the task of the adversary is even more difficult. Keep in mind that, non hyperbolic IFSs do not converge to a compact subset of X, thus the collage theorem does not apply. These PRNGs produce points with excellent statistical properties as shown in the next section.

6. Statistical Tests

This section presents the results of several statistical tests that were applied to files of bits generated by the Fractal PRNGs using the RMDA. Similar (and somewhat better) results were obtained using MDA. For each fractal PRNG, a seed of 17 integer numbers was used ($N = 16$) and the values $L = 5$ and $K = 6$ were chosen. The algorithms were implemented in C. For the construction of a 650 Mb file of random numbers on a Pentium IV microprocessor, running at 2.8 MHz, approximately 110 minutes were needed. The source code can be found in "http://eudoxos.math.uoa.gr/∼ldalla/Fractal_PRNG/home.htm". The same tests were applied to several other well-known PRNGs, like the Micali Schnorr and the Blum Blum Shub (BBS) generators (which have been proved to be cryptographically secure), the linear congruential, the lagged Fibonacci and the 3-DES generators.

6.1. *ENT test*

ENT program[a] applies various tests to sequences of bytes stored in files and reports the results. This set of tests includes:

(1) Computation of the entropy. (This value should be ≈ 8.)
(2) Chi-square Test. The chi-square test is the one of the most commonly used tests for the randomness of data, and it is extremely sensitive to errors in pseudorandom sequence generators. The chi-square distribution is calculated for the stream of bytes in the file and expressed as an absolute number and a percentage, which indicates how frequently a truly random sequence would exceed the value calculated. If the percentage is greater than 99% or less than 1%, the sequence is almost certainly not random. If the percentage is between 99% and 95% or between 1% and 5%, the sequence is suspect. Percentages between 90% and 95% or between 5% and 10% indicate the sequence is "almost suspect".
(3) Computation of Arithmetic Mean. (This value should be ≈ 127.5.)
(4) Monte Carlo computation of the value of pi.
(5) Computation of the serial correlation coefficient. (This value should be ≈ 0.)

Several files of size ≈ 12 Mbytes were created using different 17 byte keys. The results are presented in Table 1. All files pass the test easily. Note that even bad PRNGs such as the Linear Congruential PRNG pass the test.

6.2. *NIST statistical test suite*

The U.S. National Institute of Standards and Technology constructed a statistical test suite for random and pseudo-random number generators.[b] The suite includes

Table 1. Ent test results.

PRNG	Entropy	Chi-square (%)	Mean	Monte Carlo pi	Serial correlation
FIF PRNG	7.999983	50	127.4915	0.03% error	−0.000031
IFS FPRNG	7.999984	50	127.5028	0.01% error	0.000648
IFS FPRNG	7.999983	50	127.4994	0.01% error	−0.000199
HVIFS PRNG	7.999983	50	127.4933	0.02% error	0.000245
HVIFS PRNG	7.999984	50	127.4914	0.02% error	0.000085
Lagged Fibonacci	7.999983	25	127.5050	0.08% error	−0.000049
Linear Congruential	7.999983	50	127.5311	0.04% error	0.000813
Micali Schnorr	7.999984	50	127.5304	0.02% error	−0.000192
Blum Blum	7.999982	25	127.4912	0.01% error	−0.000383

[a]ENT test http://www.fourmilab.ch/random/
[b]NIST test http://csrc.nist.gov/rng/

the Monobit test, the block-frequency test, the Runs test, the longest run of 1s in a block test, the binary matrix rank test, the FFT test, the non-overlapping template matching test (148 tests), the overlapping template matching test, the Maurer's Universal Statistical test, the Lempel-Ziv compression test, the linear complexity test, the serial test (two tests), the approximate entropy test, the cumulative sums test (two tests), the random excursions (eight tests) and the random excursions variant test (18 tests). Each statistical test takes a sequence of bits and returns one p-value $\in [0, 1]$. If the p-value is ≤ 0.01 the respective sequence "fails" the test. Each test were applied to 100 sequences of 9 million bits each, generated by one of the PRNGs. The emerging 100 p-values of each sequence produce a final p-value for the PRNG. This p-value is examined to ensure uniformity. If the final p-value is > 0.0001 then the sequences are considered to be uniformly distributed. If the proportion of the sequences that fail the test is less than the minimum pass rate indicated, then the PRNG "fails" the test. The results are shown in Table 2.

The PRNGs that were presented in the above sections, pass the majority of the tests. The FIF PRNG pass all tests except Lempel-Ziv compression, where it fails in uniformity only. The affine IFS PRNG and the HVIFS PRNG fails the FFT test and its uniformity test and the uniformity test of Lempel-Ziv compression. All major PRNGs fail the latter two tests, but the IFS PRNGs give better results. Note that other PRNGs such as Modular Exponentiation, Linear and Cubic congruential fails a lot of those tests.

6.3. *DIEHARD battery of statistical tests*

Professor G. Marsaglia proposed some tests that were designed to identify weaknesses in many common non-cryptographic PRNG algorithms. These tests are: birthday spacings test, GCD, Gorilla (a "harder" monkey test), overlapping 5-permutation test, binary rank test 31×31, binary rank test 32×32 binary rank test 6×8, bitstream test, OPSO, OQSO and DNA tests, count the 1 s test on a stream of bytes, count the 1 s test for specific bytes, parking lot test, minimum distance test, 3D spheres test, squeeze test, overlapping sums test, runs test, craps test. The tests analyse a single large file ($\approx$ 12 M) of random numbers and returns several (269) p-values which should be uniform on [0, 1), if the file contains truly random bits. If one file gives several (more than six) p-values close to 1 or 0 the PRNG fails the test. A final p-value is, also, computed to check the uniformity of the p-values. This final p-value must be > 0.025 and < 0.975. We can measure the quality of the generators using the below schema. p-values < 0.001 or > 0.999 are considered bad and score 4, p-values < 0.005 or > 0.995 are considered suspect and score 2, while p-values < 0.025 or > 0.975 are considered almost suspect and score 1. All other p-values score 0. The high scores indicate a bad PRNG (Table 3). Most PRNGs including Linear congruential, Lagged Fibonacci and FIF PRNG fails the tests. However, affine IFS, affine HVIFS, BBS and Micali's PRNGs pass the tests without problems.

Table 2. NIST Statistical Test Suit results. This table shows the results of the application of the NIST statistical test suite to some very well known PRNGs (Micali, Blum Blum Shub etc.) and to the three fractal PRNGs. P means the test is passed, F the test is failed and U the uniformity test failed. The number indicates the number of tests that failed and the percentage the proportion of the sequences that pass the test.

Test	PRNGs					
	FIF	IFS	HVIFS	BBS	3-DES	Micali
Monobit	P	P	P	P	P	P
Block Frequency	P	P	P	P	P	P
Cusum	P	P	P	P	P	P
Runs	P	P	P	P	P	P
Long Run	P	P	P	P	P	P
Rank	P	P	P	P	P	P
FFT	U + P 71%	U + P 84%	U + F 85%	U + F 72%	U + F 68%	U + F 79%
Aperiodic (148)	2F 95%	P	P	P	P	P
Periodic	P	P	P	P	P	P
Universal	P	P	P	P	P	P
Entropy	P	P	P	P	P	P
Ran. Excursion (8)	P	P	P	1F 95.6%	P	P
Ran. Excursion-V (18)	P	P	P	P	P	P
Serial (2)	P	P	P	P	P	P
Lempel-Ziv	U	U	U	U	U	U
Linear-Compl	P	P	P	P	P	P
Min pass rate (with the exception of the random excursion-V)	0.96015	0.96015	0.96015	0.96015	0.96015	0.96015
Min pass rate (for random excursion-V)	0.95888	0.957624	0.957624	0.958709	0.958180	0.957431

Table 3(a). Diehard Battery of Tests results. The tests were applied to 12Mb files generated by each generator. Note that some tests (GCD, Gorilla) require much more points.

Test	PRNGs						
	FIF	IFS	HV IFS	Linear Cong	BBS	Micali	Lagged Fibonacci
Score	250	13	15	369	14	17	57
p-value	0	0.59	0.38	0	0.88	0.28	0.003

Table 3(b). Results from the application of ENT and Diehard Battery of Tests on 650 Mb files. These results include GCD and Gorilla tests. The four PRNGs pass all tests. The seed used for the RMDA IFS is {231, 128, 67, 190, 39, 100, 161, 18, 91, 130, 201, 55, 78, 160, 28, 179, 227} and the seed used for the RMDA HV IFS is {231, 128, 67, 190, 39, 100, 161, 18, 91, 130, 201, 55, 78, 160, 28, 179, 155, 101, 202, 54, 93, 15, 121, 187, 200, 101, 114, 93, 142, 51, 12, 83}.

Test	PRNGs			
	IFS	HV IFS	Micali	BBS
Diehard score	16	17	17	32
Diehard p-value	0.793869	0.674611	0.244345	0.462359
ENT entropy	8.00	8.00	8.00	8.00
ENT Chi square	25%	50%	10%	95%
ENT mean value	127.5004	127.4969	127.5013	127.5014
ENT monte carlo	0.00%	0.01%	0.00%	0.00%
ENT correlation	−0.000052	0.000024	−0.000037	0.000077

Table 3(c). Results from the application of ENT and Diehard Battery of Tests on 1G b files. These results include GCD and Gorilla tests. The four PRNGs pass all tests. The seed used for the RMDA IFS is {152, 34, 191, 172, 205, 94, 142, 11, 54, 112, 202, 37, 95, 241, 142, 74, 191, 82, 32} ($N = 19$) and the seed used for the RMDA HVIFS is {152, 34, 191, 172, 205, 94, 142, 11, 54, 112, 202, 37, 95, 241, 142, 74, 191, 82, 32, 123, 204, 161, 209, 105, 24, 32, 191, 100, 204, 53, 252, 184, 73, 200} ($N = 17$).

Test	PRNGs			
	IFS	HV IFS	Micali	BBS
Diehard score	16	21	23	32
Diehard p-value	0.957046	0.536103	0.871725	0.462359
ENT entropy	8.00	8.00	8.00	8.00
ENT Chi square	75%	50%	50%	95%
ENT mean value	127.4984	127.4969	127.4978	127.5014
ENT monte carlo	0.01%	0.00%	0.00%	0.00%
ENT correlation	−0.000015	0.000013	−0.000026	0.000077

7. Conclusions

Three new methods to construct Pseudo Random Numbers were proposed. Two of them exhibited to have very good statistical properties. In fact, they pass all the well known statistical tests that other known generators fail. Note that, the two generators that pass all of those tests (Micali-Schnorr and BBS) have been proven cryptographically secure. The generators are easy to implement. The amount of time needed to produce a series of random numbers is comparable with the amount of time that the Micali-Schnorr and BBS generators need to produce an equal series of numbers. However, they are not fast enough to be used for simulation. In addition, there is no method known that one can use to predict the next numbers produced by the PRNGs (it seems that such a method does not exist). However, statistical tests cannot replace mathematical proofs. In depth theoretical analysis must follow in order to prove that these PRNGs are actually non predictable and therefore cryptographically secure. Future research should include some additional statistical properties, such as the (asymptotical) independence of the random bits (so that the produced bits are independent and uniformly distributed) and proof that the sequences, that are produced, are computational indistinguishable (in polynomial time) from truly random ones. Detailed study of the PRNG's behavior in discrete spaces is, also, needed.

In the statistical testing, L was set equal to five in order to use the standard 64-bit floating point representation. Greater values of L may be used in combination with a more extended floating point representation and are expected to give even better results. Note that one can use IFSs on $\mathbb{R}^3$ or on $\mathbb{R}^4$ or non affine IFSs on $\mathbb{R}^2$ (which do not converge) and construct similar PRNGs which are expected to, also, have good statistical properties.

Acknowledgments

I would like to thank Dr. A. C. Dallas for valuable discussions and professors Leoni Dalla and Sergios Theodoridis for their insights.

References

1. B. F. Barnsley, J. Elton, D. Hardin and P. Massopust, Hidden variable fractal interpolation functions, *SIAM J. Math. Anal.* **20**, 1218–1242 (1989).
2. B. F. Barnsley and A. N. Harrington, The calculus of fractal interpolation functions, *J. Approx. Theory* **57**, 14–43 (1989).
3. M. F. Barnsley and L. P. Hurd, *Fractal Image Compression* (1993).
4. M. F. Barnsley, Fractal functions and interpolation, *Constr. Approx.* **2**, 303–329 (1986).
5. M. F. Barnsley, *Fractals Everywhere*, 2nd edn. (Academic Press Professional, 1993).
6. P. Bouboulis and L. Dalla, Hidden variable vector valued fractal interpolation functions, *Fractals* **13**(3), 227–232 (2005).
7. P. Bouboulis, L. Dalla and V. Drakopoulos, Image compression using recurrent bivariate fractal interpolation surfaces, *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, Complexity: A Unifying Direction in Science, Volume II, July 2006.

8. P. Bouboulis, L. Dalla and V. Drakopoulos, Construction of recurrent bivariate fractal interpolation surfaces and computation of their box-counting dimension, *J. Approx. Theory* **141**, 99–117 (2006).
9. P. Bouboulis and L. Dalla, Closd fractal interpolation surfaces, *J. Math. Anal. Appl.* **327**(1), 116–126 (2007).
10. P. Bouboulis, V. Drakopoulos and S. Theodoridis, Image compression using affine fractal interpolation surfaces on rectangular lattices, *Fractals* **14**(4), 1–11 (2006).
11. Y. Fisher, *Fractal Image Compression: Theory and Application* (Springer Verlag, New York, 1995).
12. D. P. Hardin and P. R. Massopust, The capacity of a class of fractal functions, *Math. Phys.* **105**, 455–460 (1986).
13. N. Lu, *Fractal Imaging* (Academic Press, 1997).
14. P. R. Massopust, *Fractal Functions, Fractal Surfaces and Wavelets* (Academic Press, 1994).
15. D. S. Mazel and M. H. Hayes, Using iterated function systems to model discrete sequences, *IEEE Trans. Signal Process.* **40**, 1724–1734 (1992).
16. D. S. Mazel, Representation of discrete sequences with three-dimensional iterated function systems, *IEEE Trans. Signal Process.* **42**, 3269–3271 (1994).
17. B. B. Nakos and C. Mitsakaki, On the fractal character of rock surfaces, *Int. J. Rock. Mech. Min. Sci. Geomech. Abstr.* **28**, 527–533 (1991).
18. C. S. Pande, L. E. Richards, N. Louat, B. D. Dempsey and A. J. Schwoeble, Fractal characterization of fractured surfaces, *Acta Metallurgica* **35**, 1633–1637 (1987).
19. J. R. Price, Resampling and reconstructing with fractal interpolation functions, *IEEE Signal Process. Letters* **5**, 228–230 (1998).
20. P. Wong, J. Howard and J. Li, Surfaces roughening and the fractal nature of rocks, *Phys. Rev. Lett.* **57**, 637–640 (1986).