F.S.A.I., Vol. 11, No. 1-3, pp. 109-117, 2005

About Random and Pseudo-random Sequences

Ennio Cortellini

University of Chieti, Pescara, Italy
e-mail: enniocortellini@inwind.it

Abstract. In this work, after a first analysis of random sequences and of the algorithmic impossibility to decide whether a certain arbitrary sequence may be defined random, we analyze some properties of pseudo-random sequences and in particular Blum-Blum-Shum ($y = B(x) \bmod n$) algorithm of which an implementation based upon some symmetry properties of quadratic congruencies is provided. In this way, a reduction of computational complexity is obtained, whose high value was responsible for a use of the same generator that was limited to the sole creation of short sequences.

Key words: random sequences, test Martin-Löf, BBS generator

We are often led to think that a good algorithm is just the one whose operating time is not very long, implicitly admitting that its result is always to run. However, this is nothing other than a mathematical abstraction; actually many programs contain errors that show themselves only through details at the entry that remain unobserved for a long time. On some other circumstances and owing to different reasons computers may be malfunctioning by practically conditioning the operating result of an algorithm.

This consideration has led to the development of probabilistic algorithms or rather randomized, whose wrong behaviour is generated on purpose and also measured in terms of probability.

Such algorithms are also used in cryptography, in fact the protocols' inviolability is strongly linked to the choice of coding and decoding secret keys; a key is basically required not to be preferred by the cryptanalyst; for this reason keys need to be generated through a random procedure, or rather it is necessary to be provided with random sequences.

It seems clear that the first step to take is to decide what is meant by random sequence, namely what the meaning of randomness algorithm is. Let's start with some examples and with one of the most simple in particular, that is the one which is obtained by endlessly tossing a coin and noting down which one of the two sides comes up at each toss-up. Randomness is given by the impossibility to guess the side that will come up after our toss-up and this is probably due to the fact that the probabilities that each side of the two sides has to come up are the same.

One can generalize the aforesaid procedure by considering an urn containing $a_1 ... a_5$ balls whose colour is respectively $c_1 ... c_5$. A ball is drawn out, its colour is noted down and the ball is put back into the urn. The procedure is repeated as many times as the decided drawings.

We can observe that if one has only two balls, that is $s = 2$ and $a_1 = a_2$ our generalization agrees with the case of the coin toss-up. Theoretically, through this experiment we can also obtain a sequence of the following type: $0, 1, 0, 1, 0, 1$ this one is not clearly random at all, in fact we know that it may be described as follows: $x_{2k} = 0, x_{2k+1} = 1, \forall k \in N$.

Formalization process can start by considering a probability field (Ω, K, P) with a random variable on Ω referred to as characteristic. If we indicate with $x_1, x_2, \ldots$ a random sequence where $x_1, \ldots, x_n, \ldots \in X$ and, $a_i \in X$ then we wonder if a number $n \in N$ exists so as to result $x_n = a_i$.

If the answer is negative, then we accurately know something about the sequence, therefore the sequence is not random.

In contrast, if the answer is affirmative, even in this case we can state that we know something but not everything, therefore the random nature of the sequence remains. We will examine the latter mentioned case. So, if we admit that a_i appears n times, by leaving it out of the sequence until its latest appearance, we will notice that the sequence itself is not random, therefore each element of X is to appear $\infty -$ times.

Let's indicate with P_1 the condition that $\forall a_i \in X$, a_i appears $\infty -$ times within the sequence. As with $X = \{c, t\}$, for instance in the case of a fair coin toss-up, the condition

$$\lim_{n\to\infty} \frac{Nu(n)}{n} = \lim_{n\to\infty} \frac{\mu(n)}{n} = \frac{1}{2}$$

is plausible, where $\mathbf{N}u(n) =$ the number of appearances of *h* (head) if the coin has been tossed up *n* times.

Likewise, $\mu(n)$ as with the appearance of "t", namely tail.

As with $X = \{x_1, ..., x_p\}$ similarly $\lim\limits_{n\to\infty} \frac{\nu_i(n)}{n} = \frac{1}{p'}$, $i = 1, ..., p$ is to result and we will indicate it with P_2. This situation can be generalized as follows:

Be $(x_n)_{n\in N}$, $k \in \mathbf{N}^*$ and $y_1, ..., y_k \in X$. Indicated with $\nu_y(n)$, the cardinal number of the index set *j* so as $1 \le j \le n$ and $x_j = y_1$, $x_{j+1} = y_2, ..., x_{j+k-1} = y_k$ and supposing that $\lim\limits_{n\to\infty} \frac{\nu_y(n)}{n}$ exists, we will refer to this limit as *appearance frequency* of $y = y_1 ... y_k$ within the sequence $x = (x_n)_{n\in\mathbf{N}}$.

$(x_n)_{n\in\mathbf{N}}$ is said to be $k-$*distributed* if $\lim\limits_{n\to\infty}\frac{\mathbf{N}u_y(n)}{n}=\frac{1}{p^k}$ where k is cardinal of X.

If this condition is valid for each $k\in\mathbf{N}^*$, then we say that $(x_n)_{n\in\mathbf{N}}$ is $\infty-$ distributed. We will call P_3 the condition: $(x_n)_{n\in N}$ is $\infty-$distributed.

Now let's also recall Champernowne's example (J. London Math, Soc. 8, p. 254-260, 1933): 0100011011000001010011100 where we will first look for words having length 1, those having length 2, and so on, by following the lexicographical arrangement. This sequence is certainly clearly $\infty-$ distributed, but it is not a random sequence.

Another problem rises in relation to the subsequences of a random sequence, obtained through selection. Let's denote with P_4 the condition that each subsequence of a random sequence is a random sequence. However, this does not satisfy us because, for instance, in the case of binary sequences ($X=\{0,\ 1\}$) we can find some subsequences so as all the terms are equal to 0, that is, a sequence which is not random. Therefore, we have to accept only certain rules in order to choose subsequences and in so doing we get to Church's thesis.

Generally, we accept only selection rules expressible by recursive functions, namely computable, and the subsequence that is obtained is 1 – distributed. We will indicate the above mentioned condition with P_5.

Richard von Mises [1] suggests the following definition:

Be $X=\{a_1,\ldots,\ a_p\}$ the sequence $(x_n)_{n\in\mathbf{N}}$, $x_n\in X$, $\forall n\in\mathbf{N}$.

The given sequence is called random sequence if:

i) $\lim\limits_{n\to\infty}\frac{\nu_1(n)}{n}+\ldots+\lim\limits_{n\to\infty}\frac{\nu_p(n)}{n}=1$

ii) satisfies P_5.

Let's specify that at first, instead of P_5 condition, P_4 has been required, but Abraham Wald suggested this correction; in addition to it he was able to validate this definition through a pattern only after A. Church, and exactly in 1940 he has strictly formalized the concept of computable selection. Therefore we can state that random sequences are determined in this concept, in terms of Mises – Wald [2].

We must also recall that E. Borel proved that nearly all $x\in(0,\ 1)$ numbers allow one representation $x=\frac{x_1}{2}+\frac{x_2}{2^2}+\ldots.$ where $x_i\in\{0,\ 1\}$, $i\in N^*$ so that the sequence $(x_n)_{n\in\mathbf{N}^*}$ satisfies P_3.

Another new step forward was taken by P. Martin-Löf through the following definition:

Be $X=\{a_1,..., a_p\}$, X^* the free monoid on X that makes all the words of the alphabet $X=\{y_1... y_k \mid k\in \mathbf{N}, y_i\in X, i=1,..., k,$ and as with $k=0$ we say that the empty word is obtained; therefore by Martin - Löf text we mean $V\subseteq X^*\times \mathbf{N}^*$ (where. $\mathbf{N}^*=\mathbf{N}|\{0\}$) so as:

i) V is enumerable recursive

ii) $V_{m+1}\subset V_m$, $\forall m\geq 1$, where $V_m=\{x\in X^* \mid (x, m)\in V\}$;

iii) the elements number of $X^n\cap V_m$ is smaller than $\dfrac{p^{n-m}}{p-1}$, where $X^n=X\times ...\times X$, $n-times$.

Let's denote with $m_V(x)=$ $\max\{m\in \mathbf{N} \mid (x, m)\in V\}$. A Martin-Löf U test, so that for any other V test we have that $\exists t\in N : V_{m+t}\subset U_m$ for each $m\in \mathbf{N}^*$, is called universal Martin-Löf test (1).

If one of Martin-Löf tests also satisfies:

iv) $\forall m\geq 1$, V_m is sequential (namely $\{y\in X^* \mid x\subset y, \text{ as usual}\}\subset V_m$, $\forall x\in V_m$), we say this text is a sequential text.

We notice that a universal test cannot be sequential, but if we consider all sequential tests we can give a definition to the concept of sequential universal test by using (1), that is the sequential word is added to test word in (1).

Then we say that $(x_n)_{n\in \mathbf{N}}$, $x_i\in X$, $\forall i\in \mathbf{N}$ sequence is a random sequence in terms of Martin- Löf if $\lim\limits_{n\to\infty} m_V(\underline{x}(n))<\infty$ for each V, Martin- Löf sequential test, where $\underline{x}(n)$ is the initial sequence of length n of the given sequence, namely $\underline{x}(n)=x_1x_2...x_n$. Furthermore let's remember that we indicated $m_V(\alpha)=\max\{m\in \mathbf{N} \mid (\alpha, m)\in V\}$).

To sum it up, it has been observed that random sequences having a certain length may be really a lot and as Laplace used to say "*in the toss game the appearance of heads for 100 successive times is considered as an extraordinary occurrence because the numberless combinations that may occur out of 100 successive tosses-up either form regular sequences, where one can observe a rule that is easy to understand, or form irregular sequences which are incomparably more numerous*". Likewise we can notice that deciding whether to define a certain arbitrary sequence random is generally algorithmically impossible.

Unfortunately, though random sequences exist, it is not easy to identify the sources that produce them. It is therefore difficult to guarantee that the generation of a bit occurs independently from the one of the other bits.

Therefore one applies to the so-called pseudo-random numbers generators, that is, to those algorithms that starting with a first value named seed (provided as entry datum) generate an arbitrarily long sequence of numbers; this sequence is called *period*. The longer this sequence is, the better the generator is.

In order to evaluate the efficiency of a pseudo-random sequence generator, a number of standard evaluation tests is performed which guarantee to check some of the random sequence properties. The main tests are as follows:

Frequency tests: this test is aimed at checking if the different elements appear in the sequence the same number of times.

Poker test: this test is aimed at checking if arbitrary length prefixed subsequences are equally distributed.

Self-correlation test: this test is aimed at checking the number of elements repeated at a prefixed distance.

Run test: this test is aimed at checking if the spreading of the maximal subsequences containing equal elements is of negative exponential type.

In cryptographic applications, binary computationally complex generators are used and it's to the latter we formally refer to. One of the best known generators is the one called BBS that was devised by Blum and Shum in 1986 [3].

Be $pq = n$ the product of two large prime numbers like $4k+3, k \in \mathbf{N}$ (called Blum's primes) and be $y_0 = x^2 \bmod n \quad \forall x$ the seed of the generator that computes the sequence of integers y_i of $h \le n$ and in parallel with generate a binary sequence b_i according to the following algorithm:

1. Generate two Blum's large primes, *p* and *q* with *p*≠*q*
2. Assign $n := pq$
3. Randomly choose $x \in [1, \; n \; -1]$
4. $y_0 \; = \; x^2 \bmod n$
5. The sequence is defined as $x_i \; := \; (x_{i-1})^2 \pmod{n}$ and $z_i \; := \; x_i \bmod 2$
6. The output is $z_k \;, z_{k-1} \,., \,...... z_3, \; z_2, \; z_1$

Despite some computation precautions to increase efficiency, BBS generator is quite slow because the computation of each bit requires large numbers to be squared in modular algebra. However the slowness is a property of all the generators designed according to computationally difficult problems; therefore they are used in the creation of short secret keys.

We mean to observe if a computing time reduction is possible by using some properties of the second degree congruencies.

Let’s consider congruence $y \equiv x^2 \bmod n$ and for a prefixed value $n = pq$ by varying x within interval $[0, n]$, n values of y are obtained [4]. By charting the results on a bidimensional Cartesian system one gets the result plot in Fig. 1.

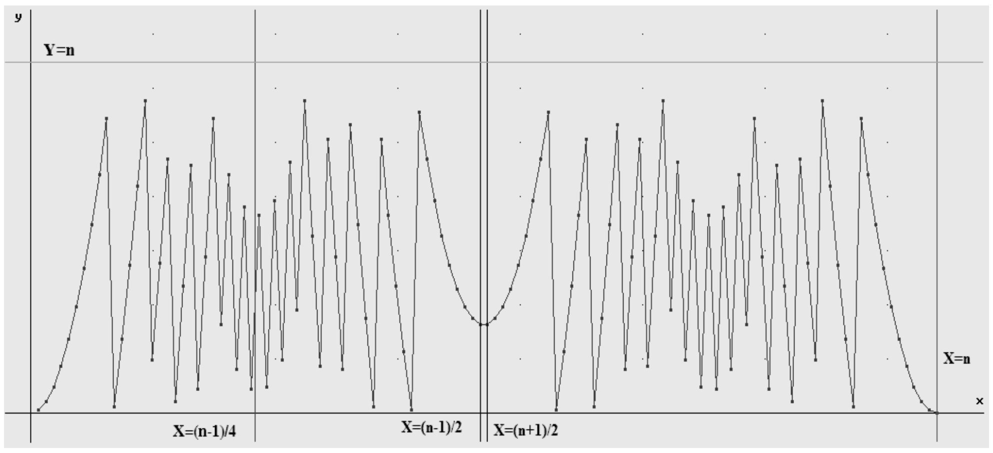

Fig. 1 – Congruence representation $y=x^2 mod\ n\ \forall\ x\in[1,(n-1)]$

The first thing that can be observed is that congruence has a symmetry in $x = (n-1)/2$, therefore row number 3 of BBS algorithm is turned into: 3) randomly choose $x \in [1, (n-1)/2]$.

Recall that given a congruence $y = (ax^2 + bx + c) \bmod n$ and two of its points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$, it is possible to write the equation of its symmetrical one as compared to y-axis through a simple operation such as $y = [ax^2 + (y_2 - y_1 + a)x + y_2] \bmod n$; therefore, in our case, we have the congruence $y = x^2 \bmod n$ and the reference points:

$$\begin{aligned} &a)\ P_1\ [(n-1)/2\ ,[(n-1)/2]^2 \bmod n] \\ &b)\ P_2\ [(n+1)/2,\ [(n+1)/2]^2 \bmod n] \end{aligned}.$$

Actually only one value is needed the subsequent algorithm, because it is known that $y_2 = y_1$.

All things considered, $y = [ax^2 + (y_2 - y_1 + a)x + y_2] \bmod n$ becomes $y = \left[x^2 + x + \left(\frac{n+1}{2}\right) \bmod n\right] \bmod n$.

In fact it graphically results as follows:

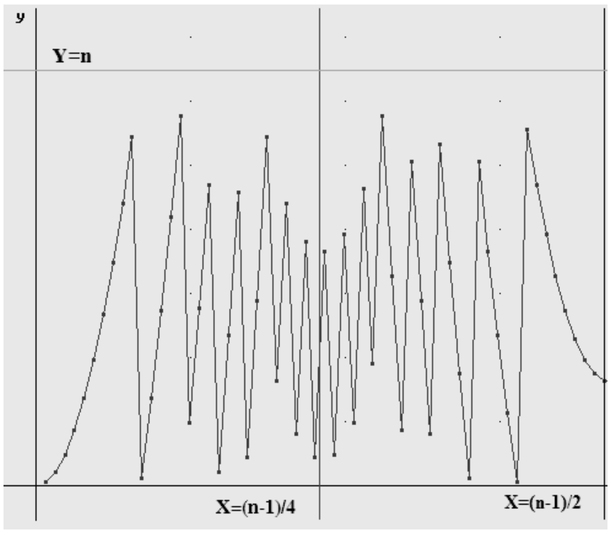

Fig. 2 – Congruence representation $y=x^2 mod\ n\ \forall\ x\in[1,(n-1)/2]$

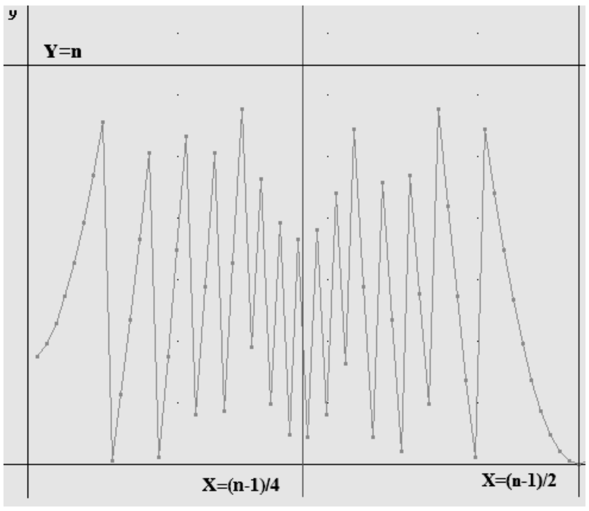

Fig. 3 – Congruence representation $y=[x^2+x+[(n+1)/2]^2\ mod\ n]\ mod\ n\ \forall x\in[1,(n-1)/2]$

One notices that congruence $y = x^2 \mod n$ studied for $x\ [1,(n-1)]$ can be examined through some obvious transformations of the independent variable x through:

$a) y = x^2 \mod n \quad \forall x\ [1,(n-1)/4]$

$b) y = [x^2 + x + [(n+1)/2]^2 \mod n] \mod n$

both $\forall\ x\ [1,(n-1)/4]$.

It results that the BBS algorithm becomes as follows:

1. Generate two Blum's large primes, p, q, $p \neq q$.
2. Compute $n = p \cdot q$.
3. Compute $P = [(n-1)/2]^2 \bmod n$
4. Randomly choose $x_i \in [1, (n-1)/2]$
5. If $x \in [1, (n-1)/4]$ then
   $y_i = (x_i)^2 \bmod n$ and go to 9
6. If $x_i \in [(n-1)/4, (n-1)/2]$ then
   $x = [(n-1)/2 - x_i]$ and $y_i := [x^2 + x + P] \bmod n$ and go to row 10
7. If $x_i \in [(n-1)/2, 3(n-1)/4]$ then
   $x = [x_i - (n-1)/2]$ and $y_i := [x^2 + x + P] \bmod n$ and go to row 10
8. $x_i \in [3(n-1)/4, (n-1)]$ then
   $x = [(n-1) - x_i]$ and $y_i = (x_i)^2 \bmod n$ and go to 9
9. The sequence is defined as $x_i := (x_{i-1})^2 \bmod n$ and $z_i = x_i \bmod 2$
10. The sequence is defined as $x_i = [(x_{i-1})^2 + (x_i - 1) + P] \bmod n$ and $z_i := x_i \bmod 2$
11. The output is $z_k, z_{k-1}, \ldots, z_3, z_2, z_1$

Example

With BBS algorithm one has:

- p=7 and q=23
- n=161
- $y=x^2$ *mod*161.
- I choose the seed in the interval (1,160) for instance x_0=**23**
- $y_i = (x_i)^2$ *mod n* namely $y_0 = (x_0)^2$ *mod 121*= y_0= $(23)^2$ *mod 161*=**46**
- *For* y_0: = *46 then* y_1= $(46)^2$ *mod* 161=23
- *the bit sequence obtained is* 1,0

With the implemented algorithm one has:

- p=7 and q=23
- n=161
- P= P=[(n-1)/2]^2 *mod n*=[(161 -1)/2]^2 *mod* 161= 121
- The two congruencies to be used are:
  $y=x^2$ *mod*161.
  $y=[x^2+x+121]$ *mod*161.

- Assume the seed in the interval (1,80), for instance x_0=23**;** it results that $x_0 \in [1, (n-1)/4]$ so I use $y_i = (x_i)^2$ *mod n* namely $y_0 = (x_0)^2$ *mod 121*= y_0= $(23)^2$ *mod 161*=46
- *For* y_0: = *46* $\in$ [$(n-1)/4$,(n-1)/2] then I use $y=[x^2+x+121]$ *mod*161 but calculated in $x=[(n-1)/2-x_i]$= [80-46]=34 namely $y=[x^2+x+121]$ *mod*161= =$[34^2+34+121]$ *mod*161=23
- *The obtained bit sequence will be* 1,0.

Conclusions

The obtained computational complexity is $O\left(\left(\frac{n-1}{4}\right)^2\right)$, that is a complexity reduction by a factor of up to 8, which is significant for small *n* values, important at least for hardware implementations.

References

[1] R. VON MISES: *Grundlagen der Wahrscheinlickeitsrechnung*, Math. Z., vol.**5**, p. 52-99, 1919

[2] A. WALD: *Die Widerspruchsfreiheit des kollektivebegriffes, Ergebnisse eines math. Kolloquiums*, vol.**8**, p. 38-72, 1937

[3] L. BLUM, M. BLUM, M. SHUM: *Sample unpredictable pseudo-random generator*, SIAM J. on Computing, vol.**15**, 364-383, 1986

[4] E. CORTELLINI: *Quadratic sieves and numerical mitosis*, Current Research in Mathematics of Fuzzy Systems, pp. 5-18, Iasi, 2005