# 1 Uniform Random Number Generators.

An important ingredient in any Monte Carlo calculation is the random number generator. This is a small subroutine of the program which produces a sequence of numbers which are not at all random in the statistical sense but may have some properties which are similar to the properties of a truly random sequence.

A truly random number sequence can only be generated by a random physical process, for example radioactive decay, thermal noise in electronic devices, cosmic ray arrival time etc. However these methods are extremely unpractical and at present the longest sequence has been produced by Frigerio and Clark (N. A. Frigerio and N. Clark, Trans. Am. Nucl. Soc. 22 (1975) 283). They used a radioactive $\alpha$-particle source and a high resolution counter turned on for periods of 20 ms. The average number of decays in this time interval is 24.315. When the count was odd they recorded a zero-bit, while when even a one-bit. To this sequence they applied subsequently a correction to eliminate the bias due to the fact that an even number of decays has a different probability with respect to an odd number, obtaining eventually $2.5 \times 10^6$ 31-bit truly random numbers.

**Exercise 1**: By using the Poisson distribution compute the probability of a zero-bit and of a one-bit in the Frigerio and Clark experiment.

**Exercise 2**: Consider a random sequence of zeroes and ones, such that the probability of 1 is $p$ and that of 0 $1-p$, with $p$ unknown. Prove that the following method gives a new sequence where zeroes and ones have the same probability: Consider sequentially pairs of bits in the sequence; if the bits are equal reject them, otherwise accept the second one. Prove that the efficiency (percentage of accepted bits) of the method is $p(1-p)$ .

This method is however not very efficient since it requires large tables which are difficult to store and long to prepare. Thus at the end of the 40s, with the introduction of computers, research began on the generation of random numbers using arithmetic operations. The idea was to produce integers $X_0, X_1, X_2, \ldots$ such that $0 \leq X_n < 2^w$ where $w$ is the word size of the computer using some rule of the form

$$X_{n+1} \, = \, f(X_n) \tag{1.1}$$

and then random numbers $U_n$ with $0 \leq U_n \leq 1$ by

$$U_n \, = \, X_n/m \, . \tag{1.2}$$

There is a fairly obvious objection to this method: how can this sequence be random, since each number is completely determined by its predecessor? As J. Von Neumann said, "anyone who considers arithmetical methods to produce random numbers is in a state of sin". The answer is indeed that the sequence is not random at all, but it may appear to be random for the problem at hand if $f$ is carefully chosen. We want to stress that the choice of $f$ is very important and that this choice is problem-dependent. A generator which is random enough for a particular problem may be really bad for another one.

**Example.**

Suppose we want to generate a sequence of integers $X_0, X_1, X_2 \ldots$ in the range $0 \leq X_n < m$ using a rule $X_{n+1} = f(X_n)$ where $0 \leq f(x) < m$ for $0 \leq x < m$. Let us now choose the function $f$ randomly, i.e. giving each of the $m^m$ possible functions $f(x)$ equal probability. Let us now compute the probability that for whatever choice of $X_0$ there will be no $n$ such that $X_{n+1} = X_n$. Indeed in this case we would have $X_k = X_n$ for every $k \geq n$ and thus the sequence would be eventually constant and thus generically the function $f$ will not be acceptable for our purposes (we say generically because it may happen that the function $f$ is nonetheless a good generator for a sufficiently large number of starting values). What we have to compute is the probability that $f(x) \neq x$ for all $x$. Now, given $x$, the probability that $f(x) \neq x$ is $(1 - 1/m)$. Thus the required probability is simply $(1 - 1/m)^m$ which goes to $1/e$ for $m \to \infty$. Thus 63% of the functions $f$ produce for at least one $X_0$ a sequence which becomes eventually constant. Generically these functions will be poor generators of random numbers. Of course we want to exclude also functions which produce sequences with $X_{n+2} = X_n$ and so on. The lesson from this example is that only few $f(x)$ have any chance of producing decent random numbers: thus simple generators for which rigorous results are available are to be preferred to more "random" (that is more complicated) ones for which no theory exists as there is a high probability that these last methods generate series which are less random than those obtained through simpler, but better understood, algorithms.

**Exercise 3**: Consider a random number generator defined as in the previous example. Show that the series contains cycles, i.e. that, for a given $X_0$, there exist numbers $\lambda, \mu$ such that $X_0, X_1, \ldots, X_\mu, \ldots X_{\mu+\lambda-1}$ are distinct and $X_{n+\lambda} = X_n$ for $n \geq \mu$. $\lambda$ is called the period of the sequence.

Prove that the following is a correct algorithm to determine $\lambda$ and $\mu$:

(a) Determine the first $n$ such that $X_n = X_{2n}$;

(b) Determine the first number $i$ such that $X_i = X_{n+i}$ and $j$ such that $X_n = X_{n+j}$. Then $\mu = i$, $\lambda = j$.

(The main advantage of this algorithm consists in the absence of any memory requirement).

Finally show that the sequence $Y_0, Y_1 \ldots$ with $Y_{n+1} = f(Y_n)$ and $Y_0 = X_k$, $0 < k < \mu+\lambda$, has the same period as the original sequence for all $k$ and that the lowest $h$ for which $Y_h = Y_{h+\lambda}$ is $\max(0, \mu - k)$.

Apply these results to the middle-square method proposed by J. Von Neumann for four-digit decimal numbers. Here $m = 10^4$ and

$$f(x) = \mathrm{mod}(\mathrm{trunc}(x^2/10^2), 10^4) \tag{1.3}$$

where $\mathrm{trunc}(x)$ is the largest integer such that $x \geq \mathrm{trunc}(x)$. Compute $\lambda$ and $\mu$ for every $X_0$.

At present the most widespread random number generators use the so-called linear congruential method, i.e. the rule

$$X_{n+1} = \mathrm{mod}(aX_n + c, m) \tag{1.4}$$

Usually $m$ is called the modulus, $a$ ($0 \leq a < m$) the multiplier and $c$ ($0 \leq c < m$) the increment. The modulus can in principle be arbitrary. However there is a very convenient choice which is $m = 2^w$ where $w$ is the number of bits of a word of the computer. Indeed in this case the "mod $m$" operation does not require any division. Given $m$ one must then choose $a$ and $c$. An obvious property which can be required is that the generated sequence have the longest possible period, which is obviously $m$. Two necessary conditions can be determined very easily: $a$ and $c$ must be relatively prime to $m$.

Indeed if $a$ is not relatively prime to $m$, then $a = pb$, $m = pq$, $c = pk + h$, with $p \neq 1$, $0 < b < q$, $0 \leq k < q$, $0 \leq h < p$. Then we can write

$$X_{n+1} \;=\; p \,\mathrm{mod}(bX_n + k\,, q) \;+ h \tag{1.5}$$

implying that only numbers of the form $p\alpha + h$, $0 \leq \alpha < q$, are produced so that the sequence does not have maximal period.

To obtain the condition on $c$, let us notice that if the period is $m$ all possible values will appear in the cycle and thus it is not restrictive to consider $X_0 = 0$. Then in this case

$$X_n \;=\; \mathrm{mod}\left[\left(\frac{a^n - 1}{a - 1}\right) c\,, m\right] \tag{1.6}$$

If $c$ is not relatively prime to $m$ and $p = gcd(m, c)$, $X_n$ will be a multiple of $p$ and the period will be at most $m/p$.

These two conditions are not however sufficient conditions. The maximal period congruential generators are completely known and are classified by the following theorem (Hull and Dubell, SIAM Review 4 (1962) 230):

**Theorem 1**: *A linear congruential generator with modulus $m$, multiplier $a \geq 1$ and increment $c$ has period $m$ if and only if*:

(i) *$c$ is relatively prime to $m$;*

(ii) *$a - 1$ is a multiple of $p$ for every prime $p$ dividing $m$;*

(iii) *$a - 1$ is a multiple of 4, if $m$ is a multiple of 4.*

In practical implementations where $m = 2^w$ this theorem requires $c$ to be odd (and one usually considers $c = 1$ or a small odd number) and $a = 4n + 1$, for some integer $n$.

The previous theorem gives the conditions which ensure maximal period. Of course this property alone is not enough to guarantee the goodness of the generator. A second important quantity which must be investigated is the potency of the multiplier. It is defined as the least integer $s$ such that

$$\mathrm{mod}((a - 1)^s\,, m) \;=\; 0 \tag{1.7}$$

Such an integer exists if $(a - 1)$ satisfies the condition (ii) of the previous theorem and $m$ is not prime. It is easy to see that when the potency is low the sequence $\{X_n\}$ is not very random. Let us notice first of all that $s = 1$ implies $a = 1$. This is a very poor generator as

$$X_{n+1} - X_n \;=\; c + km, \tag{1.8}$$

$k = 0, -1$, meaning that successive couples $(X_0, X_1)$, $(X_1, X_2)$... lie on two lines contained in the square $[0, m[\times[0, m[$.

Let us now suppose $s = 2$. Consider the sequence starting from $X_0 = 0$. Then

$$X_n \;=\; \mathrm{mod}\left[\left(\frac{a^n - 1}{a - 1}\right)\; c, m\right] \;=\; \mathrm{mod}\left(nc + \frac{1}{2}n(n-1)c(a-1)\,, m\right) \tag{1.9}$$

It follows

$$\mathrm{mod}(X_{n+1} - X_n\,, m) \;=\; \mathrm{mod}(c + nc(a-1)\,, m) \tag{1.10}$$

This relation implies that successive couples $(X_0, X_1)$, $(X_1, X_2)$, ... lie on parallel lines contained in the square $[0, m[\times[0, m[$. This is not completely unexpected. Indeed truly random numbers would obviously belong to the lines

$$X_{n+1} - X_n = \; q \tag{1.11}$$

with $1 - m \leq q \leq m - 1$. However in this case the possible values of $q$ are much less than $2m - 1$. Indeed it is easy to see that this number is $2m/\mathrm{g.c.d.}\ (m, a-1)\ - 1$.

Even worse is the situation for triples $(X_0, X_1, X_2)$, $(X_1, X_2, X_3)$, ... as

$$\mathrm{mod}(X_{n+2} - 2X_{n+1} + X_n\,, m) \;=\; \mathrm{mod}(c(a-1)\,, m) \tag{1.12}$$

implying that they will lie on the four planes

$$X_{n+2} - 2X_{n+1} + X_n \;=\; \mathrm{mod}(c(a-1)\,, m) \;+\; km \tag{1.13}$$

with $k = -2, -1, 0, 1$.

The fact that successive $d$-tuples from a congruential generator lie on a certain finite number of parallel hyperplanes in $d$-dimensional space was firstly proved by Marsaglia who also proved that the maximum number of such hyperplanes is $(d!\; m)^{1/d}$. This means that if $m = 2^{32}$, the triples lie on at most 2953 planes, the 4-tuples, the 6-tuples and the 10-tuples on at most 566, 120 and 41 hyperplanes respectively.

The potency of a generator is strictly connected with this effect: the higher the potency, the higher is the number of hyperplanes on which successive $d$-tuples lie. As a rule-of-thumb generators with potency less than 5 must be rejected as not sufficiently random. When $m = 2^w$ it is easy to give a condition on $a$ ensuring high potency. Notice that because of Theorem 1 we can have either $a \bmod 8 = 5$ or $a \bmod 8 = 1$, i.e. either $(a-1) \bmod 8 = 4$ or $(a-1) \bmod 8 = 0$. In the first case the potency is $w/2$ if $w$ is even , $(w+1)/2$ if $w$ is odd, while in the second case we have $s \leq (w+2)/3$. Thus in order to obtain maximal potency the multiplier $a$ must have the form $a = 8n + 5$.

**Exercise 4:** Consider a linear congruential generator with $m = 2^w$, $a = 2^k + 1$, $2 \leq k < w$, $c = 1$. Prove that it has maximal period. If $w = 32$ compute the potency for all possible values of $k$, showing that a potency greater than four is achieved only if $k \leq 6$, i.e. for small multipliers. However small values of $a$ must be avoided as they give rise to high serial correlations and thus this family of $a$'s does not give rise to acceptable generators. (Notice that this choice of $a$ is particularly appealing as $X_{n+1}$ can be computed without

any multiplication (why?) and for this reason generators of this type were of widespread use in the 50s).

Let us finally discuss some other properties of the sequences obtained from a linear congruential generator. First of all let us notice that if $m = 2^w$ the sequence cannot be used as a generator of random bits as the right-hand digits of $X_n$ are much less random than the left-hand digits. Indeed for every divisor $d$ of $m$, if $Y_n = \mathrm{mod}(X_n\,, d)$, then

$$Y_{n+1} \;=\; \mathrm{mod}\,(a_d Y_n + c_d, d) \tag{1.14}$$

where $a_d = \mathrm{mod}(a, d)$ and $c_d = \mathrm{mod}(c, d)$ .

Taking $d = 2^k$ we obtain that the lowest $k$-bits of $X_n$ form a sequence whose period is $2^k$ or less. In particular the last bit is constant or alternating.

**Exercise 5:** Consider the following generators:

RAN supplied by Digital on its computers with $m = 2^{32}$, $a = 69069$ and $c = 1$;

RAND the standard Unix generator with $m = 2^{31} - 1$, $a = 1103515245$, $c = 12345$;

DRAND48 provided by IBM on its RISC machines with $m = 2^{48}$, $a =$5DEECE66D$_{16}$ and $c = B_{16}$.

For each generator compute the period and the potency.

Hint: note that $2^{31} - 1$ is a prime number (Marsenne prime).

Let us now pass to discuss other possible random number generators. Another popular form is simply

$$X_{n+1} \;=\; \mathrm{mod}\,(aX_n, m) \tag{1.15}$$

These generators are linear congruential generators with $c = 0$. This makes these generators faster but reduces the period (as $c = 0$ is not relatively prime to $m$ it cannot have maximal period according to Theorem 1). However with a proper choice of $a$ one can still obtain sufficiently long periods. In the interesting case $m = 2^w$, $w > 3$, if $a \bmod 8 = 3, 5$ and $X_0$ is odd, the period is $2^{w-2}$ which is sufficiently large for many applications. An example of this class of generators is RANF, the standard generator on CRAY computers, which has $m = 2^{48}$, $a =$2875A2E7B175$_{16}$ and period $2^{46}$.

**Exercise 6:** Consider the random number generator RANDU supplied by IBM on its first-generation computers; it is defined by

$$X_{n+1} \;=\; \mathrm{mod}\,(65539X_n, 2^{31}) \tag{1.16}$$

Compute the period when $X_0$ is odd and prove the relation

$$9X_n \;-\; 6X_{n+1} + \;X_{n+2} \;=\; 0 \bmod 2^{31} \tag{1.17}$$

Show that triples $(X_n, X_{n+1}, X_{n+2})$ lie on at most 16 planes. Can this generator be a good source of random numbers?

**Exercise 7:** Prove the formula

$$X_{n+k} = a^k X_n + \frac{a^k - 1}{a - 1} c \bmod m \tag{1.18}$$

Use this result to invent a parallel random number generator.

A second important family of random number generators is based on the so-called primitive polynomials. We do not want to enter in the discussion of this subject which involves the theory of finite fields. What interests us is that it is possible to find integers $k$ and $l$ with $k < l$ such that the generator

$$X_n = X_{n-k} + X_{n-l} \bmod 2^w \tag{1.19}$$

where $n \geq l$ and and $X_0, \ldots, X_{l-1}$ not all even, has period at least $2^l - 1$. In general it is possible to prove that the period of the sequence $X_n \bmod 2$ is exactly $2^l - 1$ while the full sequence has period $2^f(2^l - 1)$ where $0 \leq f < w$. Because of the first property these generators are very efficient in generating random bits. However at present is not completely clear if , beside the long period, these sequences have also the other desirable properties of random numbers. A particular bad example is the Fibonacci sequence, with $k = 1$ and $l = 2$ with period $3 \times 2^{w-1}$ (see exercise 8). However generators with higher values of $k$ and $l$ ($l \gtrsim 100$) do not show these problems and they have proved very good under the spectral test.

**Exercise 8:** Consider the random number generator

$$X_n = X_{n-1} + X_{n-2} \bmod m \tag{1.20}$$

$X_0$ and $X_1$ not both even. Prove that the relation $X_{n-1} < X_{n+1} < X_n$ never holds.

Strictly related to this class of generators are the so-called lagged Fibonacci generators which have the form

$$X_n = X_{n-k} \,.\text{BIN.}\, X_{n-l} \tag{1.21}$$

where .BIN. stands for every binary operation. An example of these generators is the KirkPatrick-Stoll generator

$$X_n = X_{n-103} \,.\text{XOR.}\, X_{n-250} \tag{1.22}$$

with period $2^{250} - 1$ which, however, has been recently shown to perform badly in Monte Carlo simulations of the Ising model. Empirical tests show that these generators perform poorly if $l \lesssim 100$ especially if the binary operation is the $.XOR.$.

To conclude this discussion on random numbers we want to discuss the important concept of *accuracy* of a random number generator. The accuracy is connected with what we have already said, the fact that successive couples, triples ... lie in planes and do not fill the square $[0, m[^2$, the cube $[0, m[^3$... uniformly.

Let us firstly define the accuracy: given an integer $t$, consider the points $P_n = (X_n, \ldots X_{n+t})$ for $n \geq 0$ and consider an arbitrary family of parallel planes passing for all $P_n$ and

call $d$ the distance between any two successive planes (this makes sense as the planes are equidistant). Then compute the maximum $d_{max}$ of $d$ taken over all families of such planes. The $t$-dimensional accuracy $\nu_t$ is then defined by $\nu_t = m/d_{max}$. For truly random numbers $\nu_t = m$ for all $t$ while for a generic generator $\nu_t \lesssim m^{1/t}$. A good generator has $\nu_t \approx m^{1/t}$ for all $t$.

Let us now discuss the meaning of $\nu_t$. Let us consider $N$ $t$-tuples $P_n = (U_{tn+1}, \ldots U_{tn+t}) = (X_{tn+1}, \ldots X_{tn+t})/m$ belonging to the hypercube $[0,1]^t$ and a small cube $C$ of side $L$ belonging to $[0,1]^t$. Let us compute the number of points $P_n$ which fall in $C$. For truly random numbers we expect this number to be $L^t/N$, but this will be true only if $Lm >> 1$, that is if $L >> 1/\nu_t$ no matter how large $N$ is. Indeed if $L < 1/m$, since the points have the form $n/m$, the cube can only be either empty or with one point inside. Thus the accuracy defines the "spatial" resolution of the random sequence. Analogously, if the sequence is produced by a random number generator we must keep $L >> 1/\nu_t$ otherwise we see the "granularity" of the generator. Low accuracy means deviations from random behaviour on quite large scales and thus these generators must be avoided.

**Exercise 9:** Suppose you want to get with uniform probability couples of integer numbers $(x_n, y_n)$ with $0 \leq x < k$ and $0 \leq y < k$ and suppose you use a random number generator with modulus $m$ and two-dimensional accuracy $\nu_2$ in the following way: $x_n = \text{trunc}(kX_{2n}/m)$ and $y_n = \text{trunc}(kX_{2n+1}/m)$. What are the values of $k$ for which deviations from equidistribution will certainly appear? And in the case of triples ? The Digital 32-bit standard random number generator (see Exercise 5) has $\nu_2^2 = 4243209856$ and $\nu_3^2 = 2072544$. Show that it can be safely used to generate couples only for $k < 10^3$ and triples for $k < 100$.

**Exercise 10:** Prove that for a random number generator of potency 2 we have $\nu_3^2 \leq 6$ while RANDU satisfies $\nu_3^2 \leq 118$. (Hint: use the relations proved in the text and in Exercise 6).

**Exercise 11:** A method to improve the accuracy of a generator is the shuffling method of Bays and Durham. Consider a random number generator which produces a sequence $X_0, X_1, \ldots$. Fix then a number $k$ and initialize an auxiliary vector $V[0], \ldots, V[k-1]$ with $V[j] \leftarrow X_j$. Set $y \leftarrow k$, $j \leftarrow 0$. The algorithm is the following:

1. Compute $r = \text{trunc}(kX_y/m)$ and set $Y_j \leftarrow V[r]$.
2. Set $V[r] \leftarrow X_{y+1}$, $y \leftarrow y+2$, $j \leftarrow j+1$.

The sequence $Y_0, Y_1, \ldots$ is the output of the routine.

Consider as an explicit example $X_{n+1} = \text{mod}(69X_n+1,\ 128)$, $X_0 = 0$, $k = 3$. Determine $\mu$ and $\lambda$ (see Exercise 3 for the definition) for the output sequence and the number of distinct couples $(x, y)$ such that $x = Y_j$, $y = Y_{j+1}$. Repeat the same exercise for the original sequence $X_n$.