Random number generator

Hernández Martinez Jorge Iván

Departamento de Física, CUCEI, Universidad de Guadalajara.
jorgeivanhdzmtz@gmail.com

I. Introduction

The most common method is to generate the following number from the last generated numbers:

$$x_n = f(x_{n-1}, x_{n-2}, ...) \tag{1}$$

For example we can use the recurrence relation

$$x_{n+1} = ax_n + c \ (mod\ m) \tag{2}$$

where m is called modulus and a and c are positive integers called the multiplier and the increment, respectively. But in these recurrence relation will eventually repeat itself with a period no grater than m. For fix these, we need to chose m that is the maximal length. The number m is usually close to the machine's largest representable integer $\approx 2^{32}$ [1]

II. Park and miller's method

they propose a "Minimal Standard" generator based on the choices

$$a = 7^5 = 16807 \qquad m = 2^{31} - 1 = 2147483647 \tag{3}$$

but it represent that a multiply of a and m, excess the maximum value for a 32 bits integer. these is the reason why the next Schrages's method, based in a a approximate factorization of m, is used.

$$m = aq + r \qquad i.e,\ q = [m/a],\ r = m\ mod\ a \tag{4}$$

with square brackets denoting integer part. If r is small, specifically $r < q$, and $0 < z < m - 1$, it can be shown that both a(z mod q) and $r[z/q]$ lie in the range 0,...,m-1, and that [2]

$$az \bmod m = \begin{cases} a(z \bmod q) - r[z/q] & \text{if it is} \geq 0, \\ a(z \bmod q) - r[z/q] + m & \text{otherwise} \end{cases}$$

Figura 1

this algorithm use the values q=127773 and r=2836.

I. Combined method

L'Ecuyer recommends the use of the two generators m_1 = 2147483563 (with a_1 = 40014, q_1 = 53668, r_1 = 12211) and m_2 = 2147483399 (with a_2 = 40692, q_2 = 52774, r_2 = 3791). Both moduli are slightly less than 2^{31} .The periods m_1-1=2 x 3 x 7 x 631 x 81031 and m_2-1=2 19 x 31 x 1019 x 1789 share only the factor 2, so the period of the combined generator is $\approx$ 2.3 x 10^{18}. For present computers, period exhaustion is a practical impossibility.[1]

III. Description of code ran0

I show at first the most simple code programing using Fortran, for a random number generator evenly distributed between 0 and 1.
Start creating a function called ran0(val) and defining our variables as integer (Ec.3, q and r) except to ran0 (our output) and for AM that correspond to divide 1/ IM, and we put it's corresponding value.

```
FUNCTION ran0(val)
INTEGER val,IA,IM,IQ,IR,MASK,k
REAL ran0,AM
PARAMETER (IA=16807,IM=2147483647,AM=1./IM)
PARAMETER ( MASK=123459876,IQ=127773,IR=2836)
```

Then for find a random value we will use equations of Ec.4. But how we see, the first value we have to allowed is zero. for this reason i used the FORTRAN command ieor, than performs a subtraction of bits" of each number generating a new one. This command guarantees the exclusion of zero and made or program a little bit more random.

```
val=ieor(val,MASK)
k=val/IQ
val=IA*(val-k*IQ)-IR*k
if (val.lt.0) val=val+IM
ran0=AM*val

END
```

in third line of the code part that we just saw, the operation there is the same as Val=mod(IA*val,IM). so if we change that part, every would remains the same. the 4 and 5 line correspond that we see in figura 1.

We only have to define the principal program that calls to our routine ran0. This program has to open a unit where the numbers will be save. Is in a cycle because we need to save every number on each repetition. But it doesn't matter because each seed bring a different random number

```
program aleatorio
integer::a,b
open(unit=100, file='aleatorios.dat')
do b=1,10
 b=-1-b

write(100,*)ran0(b)
enddo

end program
```

IV. Description of code ran2

The following routine, ran2 uses the Minimal Standard for its random value, but it shuffles the output to remove low-order serial correlations. A random deviate derived from the jth value in the sequence, I j , is output not on the jth call, but rather on a randomized later call, j + 32 on average.

First we define our type of variables and their values. I use the values of L'Ecuyer recommends the use for the two generators (subsection 1.1) , and it is assigned to idum2, iv and iy a value, but this will change on each cycle.

```
FUNCTION ran2(idum)
        INTEGER idum,IM1,IM2,IMM1,IA1,IA2,IQ1,IQ2,IR1, &
           IR2,NTAB,NDIV,idum2,j,k,iv(NTAB),iy
        REAL ran2,AM,EPS,RNMX
        PARAMETER (IM1=2147483563,IM2=2147483399,AM=1./IM1, &
        IMM1=IM1-1,IA1=40014,IA2=40692,IQ1=53668,IQ2=52774, &
        IR1=12211,IR2=3791,NTAB=32,NDIV=1+IMM1/NTAB, &
        EPS=1.2e-7,RNMX=1.-EPS)
        idum2=123456789 ; iv=NTAB*0 ; iy=0
```

then we have to be sure to prevent idum=0, and give it a new value to idum2. It's important that the seed have to be a negative number to start with this part of code.

```
        if (idum.le.0) then
           idum=max(-idum,1)
           idum2=idum
```

Then we make iterations and compute schrange's method using values for m_1

```
           do j=NTAB+8,1,-1
              k=idum/IQ1
              idum=IA1*(idum-k*IQ1)-k*IR1
              if (idum.lt.0) idum=idum+IM1
              if (j.le.NTAB) iv(j)=idum
           enddo
           iy=iv(1)
        endif
```

then if the number is g.t 0, start here, and compute the schrange's method using values for m_1 and m_2

```
k=idum/IQ1
idum=IA1*(idum-k*IQ1)-k*IR1
if (idum.lt.0) idum=idum+IM1

k=idum2/IQ2
idum2=IA2*(idum2-k*IQ2)-k*IR2
if (idum2.lt.0) idum2=idum2+IM2
```

Then we have to shuffle idum and idum2 and are combined to generate output (line 2 of the next part of code)

```
j=1+iy/NDIV
iy=iv(j)-idum2
if(iy.lt.1)iy=iy+IMM1
ran2=min(AM*iy,RNMX)
END
```

to finish we just only have to define the principal program that calls to our routine ran2. This program has to open a unit where the numbers will be save. Is in a cycle because we need to save every number on each repetition.It is important remember that seed is a negative number

```
 PROGRAM aleatorioran2

INTEGER::q,a

OPEN(unit=100, file='aleatoriosran2.dat')
do q=1,1000
    a=-1-q
    write(100,*)ran2(a)
enddo

END PROGRAM
```

V. Simulation

Now we will check if the random numbers obtained with the last method repeat among themselves. If we graph the numbers in a plane, we have to get a random distribution. Using Next code of Matlab

```
datos= textread('C:\Users\omarc\Desktop\universidad ...
Ivan\estadistica\programas\random\aleatoriosran2.dat');
y = datos(:,1);
n=length(y);
x=1:1:n;
c=unique(x);
c1=length(c)
plot(x,y,'.')
```

we get a 'cloud' of point what it looks like never cross.

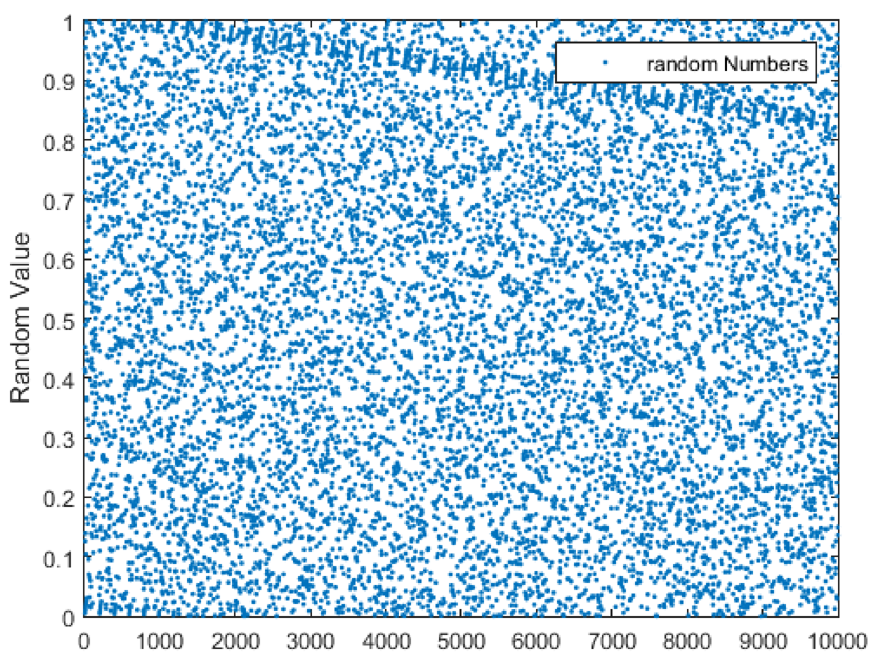

Figura 2

in the fist instance we se the value of random numbers never cross, but to make sure of that, using the command unique(x) on matlab like i did in the code, it is possible to read the vector'slength. If this length is less than 10000 (this number is because i generate 10000 random values) is because one o more numbers repeat. but in figura 3 we see that length of unique vector has the same length that the vector of random numbers.

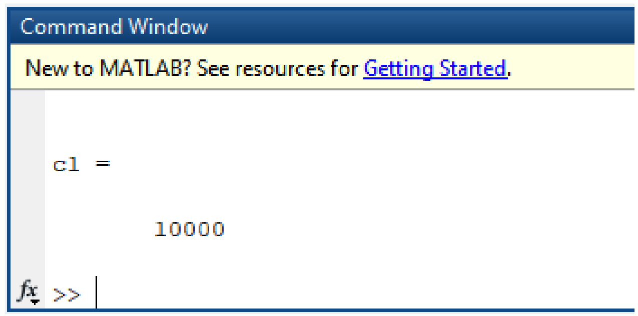

Figura 3

Referencias

[1] S. Teukolsky *Numerical Recipes in FORTRAN* chapter7. Get from *http : //nuclear.fis.ucm.es/wordpress/wp − content/uploads/2011/09/RandomNumbers.pdf*

[2] David G. Carta *Two Fast Implementations of the "Minimal Standard" Random Number Generator*.Volume 33 P. 87-88

[3] Herbert Hoeger *GENERACION DE NUMEROS ALEATORIOS* .Get from *hhttp : //webdelprofesor.ula.ve/ingenieria/hhoeger/simulacion/PARTE4.pdf*