93-462

сообщения
объединенного
института
ядерных
исследований
дубна

E2-93-462

V.V.Uzhinskii*

INTRODUCTION TO THE MONTE CARLO METHODS

*E-mail: UZHIN@LCTA9.JINR.DUBNA.SU

1993

1. Foreword

In the modern nuclear physics the so-called Monte Carlo method grow more and more important. Shortly speaking, its main point consists in the direct simulation of the processes taking place both in physical phenomena under study, and in the registration, measurement apparatus. Its spreading is explained, on the one hand, by our extension in the ideas about the nucleus, on the other hand, by the appearance of powerful computers, and especially personal computers. The latter make the Monte Carlo method accessible to everyone. We hope that our short introduction will help a reader to come into the fascinating world of the computer simulation. We suppose that the reader is familiar with some of the dialects of the computer language and can formulate his requirements with the help of computer commands. For analogous purpose we use the computer language FORTRAN. We think its sentences will be quite transparent and easy transformed into other dialects.

We wish you a good journey, dear reader !

2. The easiest example enfance

Many of us played tête heads or queue tails in the childhood. The rule of the game is an easy one. One of the players chooses the head of the pièce coin, the other one the tail. After that the coin is lancer tossed. The winner is the player whose side of the coin looks up. The procedure is repeated until the game becomes boring, or the mother calls somebody for lunch, or the father slaps somebody. In the first, in the second, and in the third cases a winner will be the person who gains more points.

This game is realized on the computer in the following way

```
      Character *1 Answer
      Character *20 A
      Character *20 B
      real *8 RANF

      write(6,*)' Enter a name of the first player '
      read(5,100)A
100   FORMAT(A20)

      write(6,*)' Enter a name of the second player '
      read(5,100)B

      write(6,*)' Thank You!'
      write(6,*)' ----------------------------------'
      SumA=0.
      SumB=0.

 1    Rn=RANF(1)
      if(Rn.LE.0.5) then
        write(6,*)A,' 1        ',B,' 0'
        SumA=SumA+1.
      else
        write(6,*)A,' 0        ',B,' 1'
        SumB=SumB+1.
      endif

      write(6,*)'Would You like to continue? Y or N (Yes or No)'
      read(5,200)Answer
200   FORMAT(A1)

      if(Answer.EQ.'Y'.OR.Answer.EQ.'y') go to 1

      write(6,*)' Sum of ',A,' = ',SumA
      write(6,*)' Sum of ',B,' = ',SumB
      end
```

The RANF function gives you at each call a random number uniformly distributed in the interval [0,1]. It is the so-called random number generator. In the easy implementation of the RANF function there is a table with the random numbers which looks like that

NN	r. numb.
1	.56843
2	.71384
3	.86249
4	.74782
5	.20298
6	.52469
7	.40826
8	.56604
9	.34473
10	.98764
...	....

In the function the number of calls is counted and a random number from the corresponding line of the table is given. In reality, one uses the most refined algorithms. We think that the reader is not interested in them and he will be satisfied with our table. Otherwise he has to refer to the special literature.

At this moment we want to point out that:

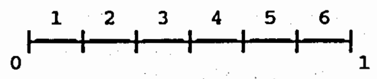
1) the numbers of the table are never changed, so you obtain the same results returning to the game a day or two later;
2) the positions of the numbers of the table depend on a computer, so executing the program on one computer and on another you can have different results;
3) for further usage it is very important to know if there are the numbers 0 and 1 in the table. This depends on a computer. We assume that 0 and 1 are in the table;
4) the random number generator on your computer can have another name. You can learn its name in the standard mathematics codes library (ask the system programmer or the people working at a big computer where you can find the library). If you use the personal computer, look through the guide of your language and compiler. If it is absent, appeal to your compiler dealer.

We wish you a good game!

3. The easy example

Let us consider another popular game dice. Here we have a cube with numbered sides. It is assumed that each side falls up with an equal probability (1/6). Let us try to write down an algorithm of the game.

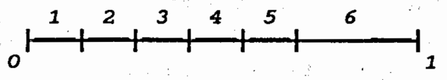
Because the random number generator gives the numbers from 0 to 1, one can divide the interval [0,1] into 6 subintervals.



If a random number falls into the ith interval, we shall consider that the side with number i falls up. So, we can formulate an algorithm of the game.

```
      Character *1 Answer
      real *8 RANF

 1    Rn=RANF(1)
      do i=1,6
        if(((i-1)/6..LE.Rn).AND.(Rn.LE.i/6.)) Nside=i
      enddo

      write(6,*)' You have   ',Nside

      write(6,*)'Would You like to continue? Y or N (Yes or No)'
      read(5,100)Answer
100   Format(A1)

      if(Answer.EQ.'Y'.OR.Answer.EQ.'y') go to 1
      end
```

Now let us suppose that you put a shot into the side with number 6, which enlarges its probability by a factor of 2. The probability of one of the other sides is equal to 2/15. In this case the interval [0,1] is subdivided like that

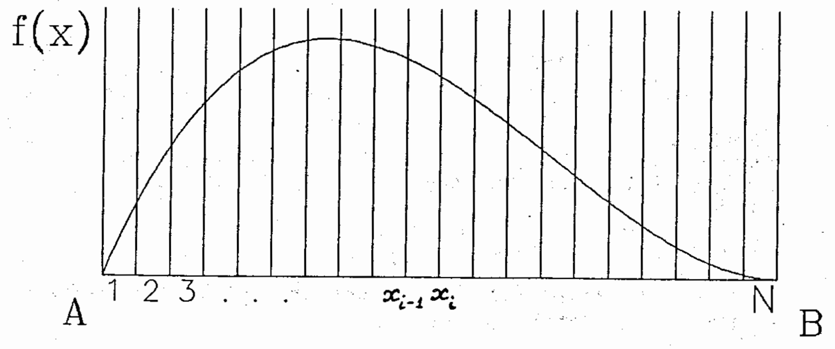


The new algorithm is nearly the same

```
      Character *1 Answer
      real *8 RANF

 1    Rn=RANF(1)
      Nside=6
      do i=1,6
        if(((i-1)*2./15..LE.Rn).AND.(Rn.LE.i*2./15.)) Nside=i
      enddo

      write(6,*)' You have   ',Nside

      write(6,*)'Would You like to continue? Y or N (Yes or No)'
      read(5,100)Answer
100   Format(A1)

      if(Answer.EQ.'Y'.OR.Answer.EQ.'y') go to 1
      end
```

4. Somewhat of the mathematics

Let us suppose that we have a random variable x distributed over the interval [A,B] according to the probability density $f(x)$.

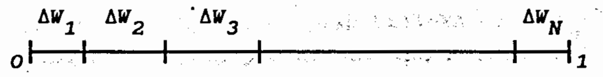


A question appears: how can one make a generator of the random numbers distributed according to probability density $f(x)$? Looking at the previous example, let us subdivide the interval [A,B] into N subintervals. Let the ends of the subintervals have the coordinates $\{x_i\}$ $(i=0, 1, \ldots, N)$, $(x_0=A, \ldots, x_N=B)$. The probability that x falls into subinterval number i is determined as

$$\Delta W_i \cong f(x_i)\,(x_i-x_{i-1}).$$

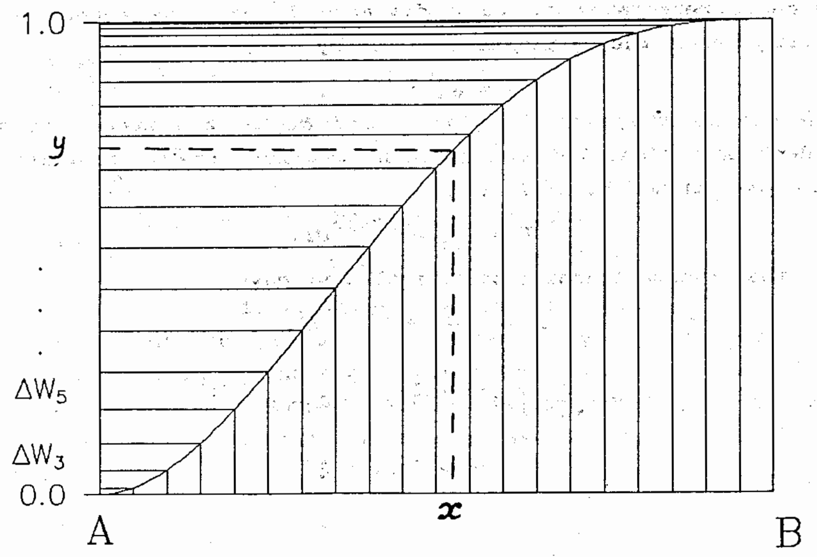
More exactly $\Delta W_i = \int_{x_{i-1}}^{x_i} f(x)\,dx, \qquad \sum_{i=1}^{N} \Delta W_i = 1.$

Now we put down successively all ΔW_i on the interval [0,1].



Using the generator of the random numbers distributed uniformly (Ranf), one can assume that when RN falls into j subinterval, the corresponding value of x is x_j. We hope that the reader can write an algorithm, if he desires.

We represent the above scheme in a more transparent form. Saving the previous subdivision of the inteval [A,B], we put down successively all ΔW_i on the Y-axis.



The solid curve in the figure is $\int_A^x f(x')\,dx'$. It is obvious that we can link each point on the Y-axis (for example y) with the corresponding point on X-axis using the relation

$$y=\int_A^x f(x')\,dx'.$$

This gives us an opportunity to generate random numbers with arbitrary distributions.

5. The inverse transformation method

Let us have random numbers distributed with the probability density $f(x)$. The probability that x falls into the interval $[x, x+dx]$ is equal to $\Delta W=f(x)\,dx$.

Let us introduce a new variable $\xi=g(x)$, such that $d\xi=g'(x)\,dx$ with $g'(x)=f(x)$. Then $dx=d\xi/g'(x)$.

$$dW = f(x)\,dx = d\xi.$$

Because ξ is distributed uniformly on the interval [0,1], we can use the function RANF for its choice. The x can be obtained as a solution of equality $g(x)=\xi$.

6. An example

As is known, a high energy particle, when hitting any substance, penetrates it to a distance l. In many cases the l distribution of the particles is given by

$$N = N_0\, e^{-l/<l>},$$

where $<l>$ is the mean free pass, and N_0 is the number of the incident particles. For one incident particle the probability to have a pass from l to $l+dl$ is

$$dW = \frac{1}{<l>}\, e^{-l/<l>}dl.$$

Using the inverse transformation method we have

$$d\xi = \frac{1}{<l>}\, e^{-l/<l>}dl,$$

$$\xi = \frac{1}{<l>}\int_0^l e^{-l'/<l>}dl' = 1 - e^{-l/<l>},$$

$$l = -<l>\,\ln(1-\xi).$$

The corresponding algorithm is

```
      Character *1 Answer
      real *8 RANF

      Averl=1.
c  Averl - average length of the tracks

      i=0
 1    Rn=RANF(1)
      i=i+1
      Trackl=-Averl*ALOG(1.-Rn)
      write(6,*)' Track # ',i,' Length = ',Trackl

      write(6,*)'Would You like to continue? Y or N (Yes or No)'
      read(5,100)Answer
100   Format(A1)

      if(Answer.EQ.'Y'.OR.Answer.EQ.'y') go to 1
      end
```

Let us suppose that the substance has thickness L and determine how many of 1000 particles go through the substance. For the question to be answered, we introduce a small change into the algorithm.

```
      real *8 RANF
      real Lmax

      N0=1000
      Averl=1.
c  Averl - average length of the tracks
      Lmax=3.

      Nres=0
      do i=1,N0
        Rn=RANF(1)
        Trackl=-Averl*ALOG(1.-Rn)
        if(Trackl.GE.Lmax) Nres=Nres+1
       enddo

      write(6,*)'There are only',Nres,' particles'
      end
```

7. Simulation of the chain reaction

Maybe, the previous example is very easy. You can complicate it by adding some of the substances with different values of $<l>$, or by changing the geometry by introducing, for example, a cavity. We are going to consider the simple cascade reproduction of the particles.

Let us suppose that the particles that hit a substance produce *k* additional particles in each interaction with substance. Let the mean free pass equal *<l>*. If we assume that the produced particles have the same value of *<l>*, and can produce new ones, the cascade reproduction is possible. As the particles can appear at different lengths, we have to store the production places. For this aim, we use an array *L(10000)*. It is obvious that one can regard initial N_0 particles as produced in the points with length *0*.

Under these conditions the algorithm is:

```
      Real *8 RANF
      Real Lmax, L(10000)

      N0=10
      Averl=1.
c  Averl - average length of the tracks
      Lmax=3.

      k=3
c k - number of childs

      do i=1,10000
        L(i)=0.
      enddo

      Nres=0
      i=0
 1    i=i+1
 2    Rn=RANF(1)
      Trackl=-Averl*ALOG(1.-Rn)
      L(i)=L(i)+Trackl
      if(L(i).GE.Lmax) then
        Nres=Nres+1
        if(i.LT.N0) go to 1
          write(6,*)'Number of produced particles = ',Nres
      else
        do j=1,k-1
          N0=N0+1
          if(N0.LE.10000) then
            L(N0)=L(i)
          else
            write(6,*)'--------It is quite enough !!!-----------'
            write(6,*)'     Number of particles > 10,000'
            stop
          endif
        enddo
        go to 2
      endif

      end
```

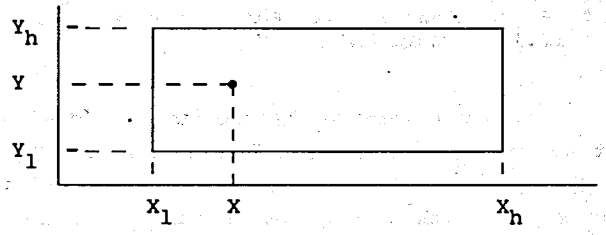
Dear reader, now you are able to investigate the dependence of

the number of produced particles on the initial value N_0, the substance thickness l_{max}, the number of the childs *k*, and the value of the mean free pass *<l>*. Maybe, in this simple model you can simulate the population growth, the atomic reactions and so on.

We wish you success!

8. The two-dimensional problems

Let us have a rectangle shown in the figure where the points fall down randomly.



Each point is characterized by its coordinates *(x,y)*. An algorithm for the simulation of the process is:

```
               REAL *8 RANF
               XL=2
               XH=4
                    YL=1
                    YH=3
     1         X=(XH-XL)*RANF(1)+XL
               Y=(YH-YL)*RANF(1)+YL
```

Now we consider another case when the target looks like a ring with radius R_0. If we choose the polar coordinate system, each point will be characterized by ϕ and r. Assuming that the points fall down randomly, we can write out the probability for a point to be in the region with $r\div r+dr$ and $\phi\div\phi+d\phi$:

$$dW = \frac{1}{\pi R_0^2}\, r\, dr\, d\phi.$$

It is obvious that the angle ϕ is distributed uniformly, so

$$\int_r dW = dW_\phi = \frac{1}{2\pi}\, d\phi, \quad \xi_\phi = \phi/2\pi, \quad \phi = 2\pi\, \xi_\phi.$$

At the same time r is distributed according to

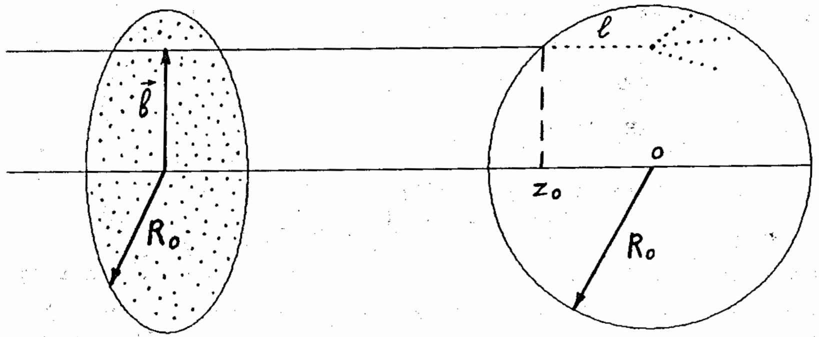
$$dW_r = \int_\phi dW = \frac{2rdr}{R_0^2}, \quad \xi_r = r^2/R_0^2, \quad r = R_0\sqrt{\xi_r}\ .$$

As the distributions in r and ϕ are independent of each other, we can choose r and ϕ independently.

```
      Real *8 Ranf
      Pi=3.14159
      R0=2.
      i=0
1     i=i+1
      Fi=2.*Pi*Ranf(1)
      R=R0*Sqrt(Ranf(1))
      write(6,*)' event # ',i,' R= ',R,' Fi = ',Fi
      if(i.LE.100) go to 1
      end
```

9. The particle penetration through the nucleus

Let us consider particle interactions with the atomic nucleus in a simple approach. Let us assume that a nucleus is a sphere with a radius R_0 and with a homogeneous structure. Let a particle hit the nucleus at an impact parameter b.



The probability that the particle will have a track with length l is given by

$$dW = \frac{1}{<l>} e^{-l/<l>} dl,$$

where $<l>=1/\sigma\rho$, σ is the total particle - nucleon interaction cross section, and ρ is the nuclear density.

The entry point of the particle into the nucleus has the z-coordinate equal to $-\sqrt{R_0^2 - b^2}$. The maximum value of l at a given b is $2\sqrt{R_0^2 - b^2}$.

We assume that the projected particle flow is a homogeneous one. So, only the particles falling into the dashed arrea can interact with the nucleus. We considered how to simulate the random falling of the particles into the ring in sec. 8. The choice of the track length was given in sec. 6. So, now we have all we need for writing an algorithm.

```
      Real *8 Ranf

      pi=3.14159

      M=64                            ! M - mass number of the nucleus

      R0=1.12*(float(M))**(1./3.)     ! Radius of the nucleus
      Ro=M/(4./3.*Pi*R0**3)           ! Nuclear density

      Snn=4.              ! Snn - the total NN cross section (in fm**2)
      Averl=1./(Snn*Ro)     ! Mean free pass in the nucleus

      N0=100                ! Total number of particles

      Nint=0
      do i=1,N0
c choice of the impact parameter and Fi
        Fi=2.*Pi*Ranf(1)
        B=R0*Sqrt(Ranf(1))

c choice of the track length
        Trackl=-Averl*Alog(1.-Ranf(1))

c determination of the final particle position
        Zint=-Sqrt(R0**2-B**2)+Trackl

        if(Zint.LE.Sqrt(R0**2-B**2)) then
          Nint=Nint+1
          write(6,*)' Interaction takes place, particle #',Nint
          write(6,*)' B= ',B,' Z- coordinate = ',Zint
        endif
      enddo
      end
```

10. The rejection method

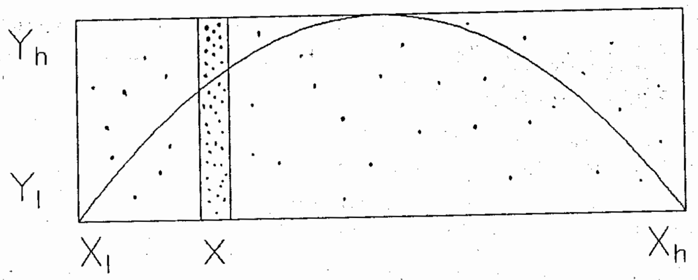
In modern physics the idea about the nucleus as a homogeneous sphere is used only for rough estimations. In more delicate calculations one takes into account the real density of the nucleus, often parametrized as

$$\rho(r) = Const/(1 + e^{(r-R)/c}),$$

where $R=1.12\ A^{1/3}$ (fm), $c=0.545$ (fm). The constant is determined from the condition $\int \rho(r)\ d^3r = 1$.

As this density does not give an opportunity to find in analytical form $\int_0^r \rho(r')\, r'^2\, dr'$, let us use the so-called rejection method.

Shortly, its main point is: let us have a probability density $f(x)$. Let us put the function into the rectangle with sides x_h-x_l and y_h-y_l. Then we will throw randomly the points into the rectangle. The points fall above the curve we will reject. The distribution of x's of the non-rejected points will be near to $f(x)$. You can check this statement considering the situation in the narrow rectangle near the point x. The number of the non-rejected points is proportional to $f(x)\,dx$, so it is just a probability to find x in the interval $[x, x+dx]$. It is obvious that for effective implementation of an algorithm one has to know the maximum value of $f(x)$. In general, there is no need in it. It is quite sufficient if $y_h > \max f(x)$.



Let us consider, for example, the function

$$f(x) = 6x(1-x), \quad \max f(x) = 1.5.$$

So, we put $y_l=0$, $y_h=1.5$, $x_l=0$, $x_h=1$.
An algorithm looks like

```
      Character *1 Answer
      real *8 RANF

      Yl=0.
      Yh=1.5

      Xl=0.
      Xh=1.

 1    X=(Xh-Xl)*Ranf(1)+Xl
      Y=(Yh-Yl)*Ranf(1)+Yl
      F=6.*X*(1.-X)
```

```
      if(Y.GT.F) go to 1
      i=i+1
      write(6,*)'Event # ',i,' x =',X

      write(6,*)'Would You like to continue? Y or N (Yes or No)'
      read(5,200)Answer
200   FORMAT(A1)

      if(Answer.EQ.'Y'.OR.Answer.EQ.'y') go to 1

      end
```

The method gives one a wider opportunity, than the inverse transformation method does (see sec. 5). Though for the unlimited functions or for the functions determined on the infinite intervals it has to be modified. Let us consider two examples:

The probability density $c\,(1-x)^a/\sqrt{x}$ on the interval [0,1] at $a>0$ is lower than $c/\sqrt{x}$. For the choice of x according to the distribution $1/2\sqrt{x}$ one can use the inverse transformation method. At a given value of x, we put $y_h=1/2\sqrt{x}$ and $y_l=0$. After that we choose a random point between y_l and y_h. The points with $y>(1-x)^a/\sqrt{x}$ will be rejected.

An algorithm in this case takes the form:

```
      Character *1 Answer
      real *8 RANF

      A=2.

 1    X=(Ranf(1))**2
      Yl=0.
      Yh=1./2./Sqrt(x)
      Y=(Yh-Yl)*Ranf(1)+Yl
      F=(1.-X)**A/2./Sqrt(X)
      if(Y.GT.F) go to 1
      i=i+1
      write(6,*)'Event # ',i,' x =',X

      write(6,*)'Would You like to continue? Y or N (Yes or No)'
      read(5,200)Answer
200   FORMAT(A1)

      if(Answer.EQ.'Y'.OR.Answer.EQ.'y') go to 1

      end
```

There is another example. Let $f(x)= e^{-x}/(1+x)$ be on the interval $[0,\infty]$. The function is smaller than e^{-x}. So one can choose x according to e^{-x}. At a given value of x one has to put $y_l=0$, $y_h=$

e^{-x}, and the algorithm looks like

```
      Character *1 Answer
      real *8 RANF

 1    X=-Alog(Ranf(1))
      Yl=0.
      Yh=Exp(-X)
      Y=(Yh-Yl)*Ranf(1)+Yl
      F=Exp(-X)/(1.+X)
      if(Y.GT.F) go to 1
      i=i+1
      write(6,*)'Event # ',i,' x =',X

      write(6,*)'Would You like to continue? Y or N (Yes or No)'
      read(5,200)Answer
200   FORMAT(A1)

      if(Answer.EQ.'Y'.OR.Answer.EQ.'y') go to 1

      end
```

11. The algorithms of the random number generators

1. Binomial distribution.

$$P_n = \binom{N_0}{n} a^n (1-a)^{N_0-n}, \quad a < 1,$$

where the random numbers n and N_0 have integer values.

```
      Character *1 Answer
      real *8 RANF

      A=0.1
      NO=10
      i=0
 1    i=i+1
      n=0
      do j=1,NO
        if(Ranf(1).LE.A) n=n+1
      enddo
      write(6,*)'Event # ',i,' n =',n

      write(6,*)'Would You like to continue? Y or N (Yes or No)'
      read(5,200)Answer
200   FORMAT(A1)

      if(Answer.EQ.'Y'.OR.Answer.EQ.'y') go to 1

      end
```

2. Poisson distribution.

$$P_n = \frac{a^n}{n!} e^{-a}, \quad a > 0,$$

where the random number n has an integer value.

```
      Character *1 Answer
      real *8 RANF

      A=1.5
      E=Exp(A)

      i=0
 1    i=i+1
      X=Ranf(1)
      FAC=1.
      Sum=1.
      n=0
 2    continue

        if(E*X.LE.Sum) then
          write(6,*)'Event # ',i,' n =',n
          go to 3
        endif
        n=n+1
        FAC=FAC*A/float(n)
        Sum=Sum+FAC
        go to 2

 3    write(6,*)'Would You like to continue? Y or N (Yes or No)'
      read(5,200)Answer
200   FORMAT(A1)

      if(Answer.EQ.'Y'.OR.Answer.EQ.'y') go to 1

      end
```

3. Distribution $x^{k-1} e^{-x}/\Gamma(n+1)$, where k is a parameter with $k = 0, 1, 3, \ldots$

```
      Character *1 Answer
      real *8 RANF

      K=2
      i=0
 1    i=i+1
      Fac=1.
      do j=1,k
        Fac=Fac*Ranf(1)
      enddo

      X=-Alog(Fac)
      write(6,*)'Event # ',i,' x =',x
      write(6,*)'Would You like to continue? Y or N (Yes or No)'
      read(5,200)Answer
200   FORMAT(A1)

      if(Answer.EQ.'Y'.OR.Answer.EQ.'y') go to 1

      end
```

4. Distribution $x^{\alpha}(1-x)^{\beta}/B(\alpha+1,\beta+1)$ (Ionk's method).

```
      Real *8 Ranf
      Alfa=-0.5
      Beta=2.5

      P1=1./(Alfa+1.)
      P2=1./(Beta+1.)

 1    R1=Ranf(1)
      R2=Ranf(1)
      R1a=R1**P1
      R2b=R2**P2
      R12=R1a+R2b
      IF(R12.GT.1.) go to 1
      X=R1a/R12
```

5. Normal distribution $f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ in the standard form

(Box & Müller method).

```
      Real *8 Ranf

      R=Sqrt(-2.*Alog(Ranf(1)))
      Phi=6.2831852*Ranf(1)
      X=R*Cos(Phi)
      Y=R*Sin(Phi)
```

The algorithm gives you two random numbers x and y with the normal distribution.

12. The processing of the Monte Carlo calculation results

Processing of the Monte Carlo calculation results looks like processing of the experimental data in high energy physics. In both cases we have a finite number of the events. Each event is characterized by some integer and uninteger (real) numbers. We shall consider simple examples when the event is characterized by an integer or uninteger number. Below we assume that N_{tot} is a number of events. Under the lines with GENERATOR n or GENERATOR x we mark the set of the lines in which a random number n or x generator is realized.

Now we represent the calculation algorithm of the most used quantities.

1. Estimation of the mean value (AVER) of the random numbers.

```
      Real *8 Ranf

      Nevent=10000
      Aver=0.

      do i=1,Nevent
      [redacted] GENERATOR x or n [redacted]
        Aver=Aver+x
C       Aver=Aver+n
      enddo

      Aver=Aver/Float(Nevent)
      Write(6,*)' mean value = ', Aver
      end
```

2. Estimation of the dispersion.

In high energy physics the dispersion is considered to mean the square of the mathematical dispersion. We follow this tradition.

```
      Real *8 Ranf
      Nevent=10000
      Aver=0.
      D=0.

      do i=1,Nevent
      [redacted] GENERATOR x or n [redacted]
        Aver=Aver+x
C       Aver=Aver+n
        D=D+x*x
C       D=D+n*n
      enddo

      Aver=Aver/Float(Nevent)
      D=D/Float(Nevent)
      D=SQRT(D-Aver*Aver)
      Write(6,*)' mean value = ', Aver,' Dispersion = ',D
      end
```

3. Statistical error of the mean value.

According to mathematical statistics, the error of the mean value (in physical sense) is determined as

$$\Delta A = \sqrt{<x^2>-<x>^2}/\sqrt{N_{events}-1} = D/\sqrt{N_{events}-1},$$

where $<x>$ is the mean value of the random number x, $<x^2>$ is the mean value of the square of the random number, N_{events} is the number of events.

```
      Real *8 Ranf

      Nevent=10000
      Aver=0.
      D=0.

      do i=1,Nevent
      [redacted] GENERATOR x or n [redacted]
        Aver=Aver+x
C       Aver=Aver+n
        D=D+x*x
C       D=D+n*n
      enddo

      Aver=Aver/Float(Nevent)
      D=D/Float(Nevent)
      D=SQRT(D-Aver*Aver)
      Error=D/Sqrt(Float(Nevent-1))
      Write(6,*)' mean value = ', Aver,' +/- ',Error
      end
```

4. Histogramming.

If a random number is an integer one, the *n* distribution is taken to mean the frequency of the random number appearance. Let *n* change from *0* to *N*. Then an algorithm of the calculation of the distribution P_n takes the form:

```
      Dimension P(101)
C                          It is assumed that N=100.
      Real *8 Ranf
      Nevent=10000

      Do i=1,101
        P(i)=0.
      enddo

      do i=1,Nevent
      [redacted] GENERATOR  n [redacted]
        n=n+1
        P(n)=P(n)+1.
      enddo

      do i=1,101
        P(i)=P(i)/Float(Nevent)
      enddo

      Aver=0.
      D=0.
      do i= 1,101
        Aver=Aver+(i-1)*P(i)
        D=D+(i-1)**2*P(i)
      enddo

      Aver=Aver/Float(Nevent)
      D=D/Float(Nevent)
      D=SQRT(D-Aver*Aver)
      Error=D/Sqrt(Float(Nevent-1))
C --------- Results printing --------------
      write(6,*)' ===== n - distribution in % ===='
```

```
      do i=1,101
        write(6,*)i-1,P(i)*100.
      enddo
      write(6,*)' ================================='
      write(6,*)' mean value = ', Aver,' +/- ',Error,' disp. ',D
      end
```

If a random number *x* is an uninteger one, there is need to know an interval of the changing of *x*, at least roughly.

Let *x* change from *A* to *B*. One can subdivide the interval *[A,B]* into *N* subintervals. The frequency of the random number appearance in the i^{th} subinterval divided by the value of the subinterval is called an *x* distribution. An algorithm for its calculation is

```
      Dimension P(100)
      Real *8 Ranf
      N0=100
      A=0.
      B=10.
      Step=(B-A)/N0

      Nevent=10000
      Do i=1,100
        P(i)=0.
      enddo

      do i=1,Nevent
      [redacted] GENERATOR  x [redacted]
        if((A.le.x).AND.(x.le.B)) then
          n=(x-A)/Step+1
          if(n.GT.N0) n=N0
          P(n)=P(n)+1.
        endif
      enddo

      do i=1,100
        P(i)=P(i)/Float(Nevent)/Step
      enddo

      Aver=0.
      D=0.
      write(6,*)' ===== x - distribution ===='
      do i= 1,100
        x=Step*(i-0.5)+A
        Aver=Aver+x*P(i)*Step
        D=D+x**2*P(i)*Step
        write(6,*)x,P(i)
      enddo

      write(6,*)' ================================='
      Aver=Aver/Float(Nevent)
      D=D/Float(Nevent)
      D=SQRT(D-Aver*Aver)
      Error=D/Sqrt(Float(Nevent-1))
      write(6,*)' mean value = ', Aver,' +/- ',Error,' disp. ',D
      end
```

5. 2- dimensional distribution.

In the case when an event is characterized by two random numbers, for example, *x* and *n*, one has to modify the previous algorithm. Let *n* change from *0* to *N*, and *x* change from *A* to *B*. The *[A,B]* can be divided into M subintervals. A calculation algorithm takes the form:

```
      Dimension P(11,100)
C                                It is  assumed  that N=10.
      Real *8 Ranf
      N0=11
      M=100
      A=1.
      B=5.
      Step=(B-A)/M
      do i=1,N0
        do j=1,M
          P(i,j)=0.
        enddo
      enddo
      Nevent=10000
      do i=1,Nevent
      [redacted] GENERATOR x and n  [redacted]
        n=n+1
        if(n.gt.N0) n=N0
        j=(x-A)/Step+1
        if(j.gt.M) j=M
        P(n,j)=P(n,j)+1.
      enddo
      do i=1,N0
        do j=1,M
          P(i,j)=P(i,j)/Nevent/Step
        enddo
      enddo
```

We wish you successful work!

The author is very grateful to Dr. Mohamad Sherif for the wonderful opportunity to work at the Physics Department, Faculty of Science of the Cairo University, where this paper was written, and to Dr. Sergey Shmakov, who found time for education of the author in the Monte Carlo methods.

Spring, 1993.   V.V. Uzhinskii

Received by Publishing Department
on December 12, 1993.

Ужинский В.В.   E2-93-462
Введение в методы Монте-Карло

Представлены программы, иллюстрирующие методы Монте-Карло, используемые в физике высоких энергий, такие как метод обратного преобразования, метод «браковки», прохождение частиц через вещество, взаимодействие частиц с ядрами и т.д. Приведены наиболее популярные алгоритмы генераторов случайных чисел (биномиальное распределение, пуассоновское распределение, β-распределение, γ-распределение и нормальное распределение).

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна, 1993

Uzhinskii V.V.   E2-93-462
Introduction to the Monte Carlo Methods

Codes illustrating the use of Monte Carlo methods in high energy physics such as the inverse transformation method, the ejection method, the particle propagation through the nucleus, the particle interaction with the nucleus, etc. are presented. A set of useful algorithms of random number generators is given (the binomial distribution, the Poisson distribution, β-distribution, γ-distribution and normal distribution).

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna, 1993