

Exercises Lecture III: Random numbers with non uniform distributions; simulations of simple random processes

1. Random numbers with non uniform distributions: Inverse Transformation Method

- (a) With the Inverse Transformation Method we can generate random numbers according to the exponential distribution $f(z) = \lambda e^{-\lambda z}$, starting from random numbers with uniform distribution: if x is the random variable with uniform distribution in $[0,1]$, then $z = -\ln(x)$ is distributed according to e^{-z} . Write a code implementing the algorithm. An example is given in `expdev.f90`.
- (b) Check—doing a histogram—that the random variate z generated with that algorithm is actually exponentially distributed. (*What is convenient to plot in order to check this behavior? Hint: with `gnuplot` you can print the log of your data (e.g., suppose you saved the values of z in column 1 and its frequency in column 2, plot with `u 1:(log($2))` or `u 1:(log10($2))`).*)
- (c) With `gnuplot` you can also do the fit of the histogram with an exponential function using the least-square method, with λ as fitting parameter. Check whether you get the expected value of λ . (*It is convenient to make a semilog plot as suggested above and then make a least-square linear fit; the slope is λ*)

Remember that with the method of the least-square fit we get for a linear regression: $y = ax + b$:

$$a = \frac{\overline{xy} - \bar{y}\bar{x}}{(\Delta x)^2}; \quad b = \bar{y} - a\bar{x}$$

where $(\Delta x)^2 = \overline{x^2} - \bar{x}^2$ (other definitions are trivial ...).

2. Random numbers with non uniform distributions: comparison between different algorithms

Suppose you want to generate a random variate x in $(-1,1)$ with distribution

$$p(x) = \frac{1}{\pi}(1 - x^2)^{-1/2}.$$

Consider both methods suggested below, do the histograms and check that both methods give correct results.

- (a) From the Inverse Transformation Method:
generate a random number U with uniform distribution in $[0,1]$ and consider $x = \sin \pi(2U - 1)$.
- (b) Generate two random numbers U and V with uniform distribution in $[0,1]$. Disregard them if $U^2 + V^2 > 1$. Otherwise consider

$$x = \frac{U^2 - V^2}{U^2 + V^2}$$

Note 1: the last method has the advantage of using only elementary operations.

Note 2: since x is also negative, pay attention to the algorithm used to make the histogram; you should notice the difference between the intrinsic functions `int` and `nint`; see also `floor`. From Chapman's book:

`AINT(A,KIND):` Real elemental function
- Returns A truncated to a whole number.
`AINT(A)` is the largest integer which is smaller than $|A|$, with the sign of A.
For example, `AINT(3.7)` is 3.0, and `AINT(-3.7)` is -3.0.
- Argument A is Real; optional argument KIND is Integer

`ANINT(A,KIND):` Real elemental function
- Returns the nearest whole number to A.
For example, `ANINT(3.7)` is 4.0, and `ANINT(-3.7)` is -4.0.
- Argument A is Real; optional argument KIND is Integer

`FLOOR(A,KIND):` Integer elemental function
- Returns the largest integer $< \text{or} = A$.
For example, `FLOOR(3.7)` is 3, and `FLOOR(-3.7)` is -4.
- Argument A is Real of any kind; optional argument KIND is Integer
- Argument KIND is only available in Fortran 95

`NINT(A[,KIND])`
- Integer elemental function
- Returns the nearest integer to the real value A.
- A is Real


```

print *, " length of the sequence >"
read *, n
print *, " exponential decay factor (lambda)>"
read *, lambda
print *, " Collecting numbers generated up to 2/lambda (disregard the others)"
print *, " and normalizing the distribution in [0,+infinity[ "
print *, " Insert number of bins in the histogram>"
read *, nbin
delta = 2./lambda/nbin
  allocate (histo(nbin))
histo = 0
do i = 1,n
  call expdev(x)
  ibin = int (x/lambda/delta) + 1
  if (ibin <= nbin)histo(ibin) = histo(ibin) + 1
end do
open (unit=7,file="expdev.dat",status="replace",action="write")
do ibin= 1 ,nbin
  write(unit=7,fmt=*)(ibin-0.5)*delta,histo(ibin)/float(n)/delta
end do

contains

subroutine expdev(x)
  REAL, intent (out) :: x
  REAL :: r
  do
    call random_number(r)
    if(r > 0) exit
  end do
  x = -log(r)
END subroutine expdev

end program test_expdev

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
! boxmuller.90
! uses the Box-Muller algorithm to generate
! a random variate with a gaussian distribution (sigma = 1)
!
program boxmuller
  implicit none
  real :: rnd,delta
  real, dimension(:), allocatable :: histog
  integer :: npts,i,ibin,maxbin,m

```

```

print*, ' input npts, maxbin >'
read*, npts,maxbin
allocate(histogram(-maxbin/2:maxbin/2))
histog = 0
delta = 10./maxbin
do i = 1, npts
    call gasdev(rnd)
    ibin = nint(rnd/delta)
    if (abs(ibin) < maxbin/2) histogram(ibin) = histogram(ibin) + 1
end do

open(1,file='gasdev.dat',status='replace')
do ibin = -maxbin/2 , maxbin/2
    write(1,*)ibin*delta, histogram(ibin)/real(npts)/delta
end do
close(1)
deallocate(histogram)
stop

contains
SUBROUTINE gasdev(rnd)
    IMPLICIT NONE
    REAL, INTENT(OUT) :: rnd
    REAL :: r2,x,y
    REAL, SAVE :: g
    LOGICAL, SAVE :: gaus_stored=.false.
    if (gaus_stored) then
        rnd=g
        gaus_stored=.false.
    else
        do
            call random_number(x)
            call random_number(y)
            x=2.*x-1.
            y=2.*y-1.
            r2=x**2+y**2
            if (r2 > 0. .and. r2 < 1.) exit
        end do
        r2=sqrt(-2.*log(r2)/r2)
        rnd=x*r2
        g=y*r2
        gaus_stored=.true.
    end if
END SUBROUTINE gasdev
end program boxmuller

```

```

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
! decay.f90
! Simulation of radioactive decay
!
PROGRAM decay
  IMPLICIT none
  REAL, PARAMETER ::      lambda=0.2
  REAL :: r
  INTEGER :: i, t, nleft, start, sizer
  integer, dimension(:), allocatable :: seed
  !
  call random_seed(sizer)
  allocate(seed(sizer))
  print *, 'Here the seed has ',sizer,' components; insert them (or print "/" ) >'
  read(*,*)seed
  call random_seed(put=seed)

  !      initial values
  print *, "initial number of nuclei >"
  read *, start
  t = 1      ! initialize time
  nleft = start ! at the beginning N(t=0)=start
  ! N(t) nuclei left at time t,
  ! that have a given probability lambda of decay
  ! in the time interval t:t+dt
  !
  OPEN(unit=7, FILE="decay.dat", status="replace",action="write")
  WRITE (unit=7,fmt=*) "# t ,      N(t)"
  WRITE (unit=7,fmt=*) "0  ", nleft !REAL(nleft)/start
  !
  DO      ! time loop
    DO i = 1, nleft      ! loop on the nuclei left
      call random_number(r)
      IF (r <= lambda) THEN
        nleft = nleft - 1      ! update the number of nuclei left
      ENDIF
    END DO
    !
    WRITE (unit=7,fmt=*) t , nleft ! or REAL(nleft)/start
    if (nleft == 0) exit
    t = t + 1
  END DO
  !
  close(7)
  stop
END program decay

```