

Exercises Unit II - part (b):

Random numbers with non uniform distributions

1. Exponential distribution with 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*)

2. An example of non uniform distribution: 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 y with uniform distribution in $[0,1]$ and consider $x = \cos(\pi y)$ (or $x = \sin(2\pi y)$, or $x = \sin \pi(2y-1)$... *Why?*).
- (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 < 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
```

3. Gaussian distribution: Box-Muller algorithm

Consider the Box-Muller algorithm to generate a random number gaussian distribution (see for instance `boxmuller.f90`; the `gasdev` subroutine used inside is similar to what you can find in “Numerical Recipes”: it gives a gaussian distribution with $\sigma = 1$ and average $\mu = 0$). Do a histogram of the data generated, calculate *numerically* from the sequence the average value and the variance, check with the expected results.

4. Gaussian distribution: the central limit theorem

Use the central limit theorem in order to produce random numbers with gaussian distribution. Remind that given a sequence of independent random numbers r_i , their average

$$x_N = \frac{1}{N} \sum_{i=1}^N r_i$$

is distributed according to

$$P_N(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_N - \mu)^2}{2\sigma^2}\right)$$

with $\mu = \langle r \rangle$, $\sigma^2 = (\langle r^2 \rangle - \langle r \rangle^2)/N$.

Furthermore, defining

$$z_N = \frac{x_N - \mu}{\sigma}$$

we have $\langle z_N^4 \rangle \approx 3 \langle z_N^2 \rangle^2$.

- Use random numbers r_i uniformly distributed in $[-1,1]$, choosing $N \approx 50$ and generating at least ≈ 1000 points x_N . Verify (doing an histogram) that x_N have a Gaussian distribution.
- Calculate numerically $\langle x_N \rangle$ and σ_x^2 and compare them with μ and σ^2 analytically calculated.
- Consider z_N and verify numerically the relationship $\langle z_N^4 \rangle \approx 3 \langle z_N^2 \rangle^2$.
- Do again the exercise using random numbers r_i with exponential distribution ($p(r) = e^{-r}$ if $r \geq 0$, 0 elsewhere). Calculate $\langle x_N \rangle$, $\sigma_{x_N}^2$ and μ , σ^2 numerically and analytically, respectively.
- (optional) Consider now random numbers r_i with distribution

$$p(r) = \frac{a}{\pi} \frac{1}{r^2 + a^2}$$

(Lorentz's), with $a = 1$, for instance. Do $\langle x \rangle$, $\langle x^2 \rangle$ and σ_x^2 exist? Try to determine the characteristic of the distribution of the variable sum

$$x_N = \frac{1}{N} \sum_{i=1}^N r_i.$$


```

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
! 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(histog(-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) histog(ibin) = histog(ibin) + 1
  end do

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

```

```
        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
```