Exercises Lecture VII

Monte Carlo numerical Integration

Metropolis method to generate random numbers

1. **Monte Carlo method: acceptance-rejection**

   Using the acceptance-rejection method, calculate $I = \int_0^1 \sqrt{1-x^2}dx$ (notice that $\pi = 4I$). The numerical estimate of the integral is $F_n = \frac{n_s}{n}$ where n_s is the number of points under the curve $f(x) = \sqrt{1-x^2}$, and n the total number of points generated. An example is given in `pi.f90`. Estimate the error associated, i.e. the difference between F_n and the true value. Discuss the dependence of the error on n.
   (*Notice that many points are needed to see the $n^{-1/2}$ behavior, which can be hidden by stochastic fluctuations; it is easier to see it by averaging over many results (obtained from random numbers sequences with different seeds)*)

2. **Monte Carlo method:**
generic sample mean and importance sampling

   (a) Write a code to compute the numerical estimate F_n of $I = \int_0^1 e^{-x^2} dx = \frac{\sqrt{\pi}}{2} erf(1) \approx 0.746824$ with the MC *sample mean* method using a set $\{x_i\}$ of n random points uniformly distributed in [0,1]:

$$F_n = \frac{1}{n} \sum_{i=1}^{n} f(x_i)$$

   (b) Write a code (a different one, or, better, a unique code with an option) to compute F_n using the *importance sampling* with a set $\{x_i\}$ of points generated according to the distribution $p(x) = Ae^{-x}$ (*Notice that* erf *is an intrinsic fortran function; useful to compare the numerical result with the true value*). Remind that in the *importance sampling* approach:

$$\int_a^b f(x)dx = \left\langle \frac{f(x)}{p(x)} \right\rangle \int_a^b p(x)dx \approx \frac{1}{n} \sum_{i=1}^{n} \frac{f(x_i)}{p(x_i)} \int_a^b p(x)dx = F_n$$

   with $p(x)$ which approximates the behaviour of $f(x)$, and the average is calculated over the random points $\{x_i\}$ with distribution $p(x)$.
   Notes: pay attention to:
   - the normalization of $p(x)$*;*
   - the exponential distribution: `expdev` *provides random numbers* `x` *distributed in* `[0,`$+\infty$`[`*; here we need* `x` *in* `[0,1]` *...*

   (c) Compare the efficiency of the two sampling methods (uniform and importance sampling) for the estimate of the integral by calculating the following quantities: F_n, $\sigma_n = (< f_i^2 > - < f_i >^2)^{1/2}$, $\sigma_n/\sqrt{n}$, where $f_i = f(x_i)$ in the first case, and $f_i = \frac{f(x_i)}{p(x_i)} \int_a^b p(x)dx$ in the second case (make a log-log plot of the error as a function of n: what do you see?).

3. **Monte Carlo method – sample mean (generic); error analysis using the "average of the averages" and the "block average"**
NOTE: THIS EXERCISE IS VERY IMPORTANT !!!

   (a) Write a code to estimate the same integral of previous exercise, $\pi = 4I$ with $I = \int_0^1 \sqrt{1-x^2}dx$, using the MC method of sample mean with uniformly distributed random points. Evaluate the error $\Delta_n = F_n - I$ for n=10^2, 10^3, 10^4: it should have a $1/\sqrt{n}$ behaviour.

   (b) Choose in particulat $n = 10^4$ and consider the corresponding error Δ_n. Calculate $\sigma_n^2 =< f^2 > - < f >^2$. You should recognize that σ_n CANNOT BE CONSIDERED A GOOD ESTIMATE OF THE ERROR (it's much larger than the actual error...)

   (c) In order to improve the error estimate, apply the following two different methods of variance reduction: 1) "average of the averages": do m =10 runs with n points each, and consider the average of the averages and its standard deviation:

$$\sigma_m^2 =< M^2 > - < M >^2$$

   where

$$< M >= \frac{1}{m}\sum_{\alpha=1}^{m} M_\alpha \quad e \quad < M^2 >= \frac{1}{m}\sum_{\alpha=1}^{m} M_\alpha^2$$

   and M_α is the average of each run. You should recognize that σ_m is a good estimate of the error associated to each measurement (=each run) and $\sigma_m \approx \sigma_n/\sqrt{n}$ is the error associated to the average over the different runs.

   (d) 2) Divide now the $n = 10,000$ points into 10 subsets. Consider the averages f_s within the individual subsets and the standard deviation if the average over the subsets:

$$\sigma_s^2 =< f_s^2 > - < f_s >^2 .$$

   You should notice that $\sigma_s/\sqrt{s} \approx \sigma_m$.

4. **Random numbers with gaussian distribution: Metropolis algorithm**

Here we use the Metropolis algorithm to generate points with the distribution $P(x) = e^{-x^2/(2\sigma^2)}$. The algorithm is implemented for instance in the code `gauss_metropolis.f90`. We consider $\sigma = 1$, but the suggestion is to write the code for a generic σ.

(a) Start from x_0=0 and choose δ=5σ to be the maximum displacement for each step. Execute runs with n=100, 1000, 10000, 100000 points, make an histogram of the points generated and compare it with the gaussian distribution. For which n the agreement is satisfactory?

(b) Choose n which gives a satisfactory result. For σ fixed, change the step size δ (i.e., change the ratio δ/σ). Determine qualitatively the dependence of the *acceptance ratio* on δ/σ. Make a plot. How to choose δ/σ in order to accept from $\approx 1/3$ to $\approx 1/2$ of trial changes?

(c) By varying n in a more refined way (e.g. from 100 to 10000 with steps of 100), compare the first moments of the distribution obtained numerically with the exact ones analytically calculated with the Gaussian. In particular, focus on the second moment and make a plot of the difference "exact variance - numerical variance" as a function of n.

(d) For fixed $\sigma = 1$ and δ=5σ, determine qualitatively the *equilibration time* (i.e. the number of steps necessary to *equilibrate* the system); a possible criterion is that the numerical estimate of $\langle x^2\rangle - \langle x\rangle^2$ is close enough to σ^2, say within 5%.

```
!-------------------------------------------------------
!       pi.f90: Calculates pi using MC
Program pi
  Implicit none
  integer, dimension(:), allocatable :: seed
  real, dimension(2) :: rnd
  Real :: area, x, y
  Integer :: i, max, pigr, sizer
  call random_seed(sizer)
  allocate(seed(sizer)
    print*,' enter max number of points='
  read*, max
  print*,' enter seed (or type /) >'
  read*, seed
  call random_seed(put=seed)
  !       open data file, initializations
  Open(7, File='pigr.dat', Status='Replace')
  pigr=0
  ! points generated within a square of side 2
  ! count how many fall within the circle x*x+y*y <= 1;
  Do i=1, max
     call random_number(rnd)
     x = rnd(1)*2-1
     y = rnd(2)*2-1
     If ((x*x + y*y) <= 1) then
        pigr = pigr+1
     Endif
     area = 4.0 * pigr/Real(i)
     if (mod(i,10)==0) Write(7,*) i, abs(acos(-1.)-area)  !write every 10 points
  end do
  Close(7)
  Stop 'data saved in pigr.dat '
End program pi
```

```
!-------------------------------------------------------
! gauss_metropolis.f90
!
! METROPOLIS generation of random numbers with a Gaussian distribution
! P(x) = exp(-x**2/(2*sigma**2))/sqrt(2*pi*sigma**2)

program gauss_metropolis
  implicit none
  integer, parameter :: dp=selected_real_kind(13)
  integer :: i,n,ibin,maxbin,m
  real(kind=dp):: sigma,rnd,delta,x0,deltahisto
  real(kind=dp):: x,x1,x2,x3,x4,xp,expx,expxp,w,acc
  real, dimension(:), allocatable :: histog
  character(len=13), save :: format1 = "(a7,2x,2f9.5)"

  print*,' insert n, sigma, x0, delta, maxbin >'
  read*, n, sigma,x0,delta,maxbin
  allocate(histog(-maxbin/2:maxbin/2))
  histog = 0.
  deltahisto = 10.*sigma/maxbin  ! histogram over a range of 10*sigma
  acc = 0.0_dp
  x = x0
  x1 = 0.0_dp
  x2 = 0.0_dp
  x3 = 0.0_dp
  x4 = 0.0_dp

  do i=1,n
     x1 = x1 + x
     x2 = x2 + x**2
     x3 = x3 + x**3
     x4 = x4 + x**4
     !ccccccccccccccccccccccccccccc
     expx  = - x**2 /(2*sigma**2)   !
     call random_number(rnd)        !
     xp = x + delta * (rnd-0.5_dp)  !
     expxp = - xp**2 /(2*sigma**2)  !   metropolis
     w = exp (expxp-expx)           !   algorithm
     call random_number(rnd)        !
     if (w > rnd) then              !
        x = xp                      !
     !ccccccccccccccccccccccccccccc
        acc=acc+1.0_dp
     endif
     ibin = nint(x/deltahisto)
     if (abs(ibin) < maxbin/2) histog(ibin) = histog(ibin) + 1
```

```
  enddo

  write(unit=*,fmt=*)"# n, x0, delta = ",n,x0,delta
  write(unit=*,fmt=*)"# acceptance ratio = ",acc/n
  write(unit=*,fmt=*)"# Results (simulation vs. exact results):"
  write(unit=*,fmt=format1)"# <x>  = ",x1/n,0.0_dp
  write(unit=*,fmt=format1)"# <x^2>= ",x2/n,sigma**2
  write(unit=*,fmt=format1)"# var2 = ",x2/n-(x1/n)**2,sigma**2
  write(unit=*,fmt=format1)"# <x^3>= ",x3/n,0.0_dp
  write(unit=*,fmt=format1)"# <x^4>= ",x4/n,3.0_dp*sigma**4

  open(1,file='gauss_metropolis.dat',status='replace')
  write(unit=1,fmt=*)"# n, x0, delta = ",n,x0,delta
  do ibin = -maxbin/2 , maxbin/2
     write(1,*)ibin*deltahisto, histog(ibin)/real(n)/deltahisto
  end do

  close(1)
  deallocate(histog)

end program gauss_metropolis
```