Exercises Lecture IV : Random Walks

1. **1D Random walks:**
   properties; comparison numerical/analytical results; convergence

   Write a code (e.g. see `rw1d.f90`) that simulates numerically a 1D random walk with a Monte Carlo approach and gives the final position x_N (and x_N^2) after N steps with fixed length ℓ and probabilities $p_\leftarrow$ and $p_\rightarrow$ of moving left and right. Without any loss of generality, you can consider $x_0 = 0$ as starting position, and ℓ=1.
   The code should calculate also *averages* over many different walkers (starting from different seeds): $\langle x_N \rangle$, $\langle x_N^2 \rangle$ and the mean square displacement $\langle (\Delta x)_N^2 \rangle = \langle x_N^2 \rangle - \langle x_N \rangle^2$.
   For comparison, the corresponding exact analytical ("theoretical") results are:

$$\langle x_N \rangle^{th} = N(p_\rightarrow - p_\leftarrow)\ell$$
$$\langle x_N^2 \rangle^{th} = [N(p_\leftarrow - p_\rightarrow)\ell]^2 + 4p_\rightarrow p_\leftarrow N\ell^2$$
$$\langle (\Delta x_N)^2 \rangle^{th} = \langle x_N^2 \rangle - \langle x_N \rangle^2 = 4p_\rightarrow p_\leftarrow N\ell^2$$

   (a) For the sake of definiteness, choose $p_\leftarrow$=$p_\rightarrow$= 0.5 and fix N. In order to follow the evolution of a random walk with the number of steps, calculate and plot the instantaneous position, i.e., x_i and x_i^2 vs. i, with i from 0 to N. Plot together the results for runs corresponding to different seeds. Do the results change? How do they compare with the expected theoretical behavior $\langle x_i \rangle^{th} = 0$ and $\langle x_i^2 \rangle^{th} = i\ell^2$? In particular, consider the final values, x_N and x_N^2 and compare them with the theoretical ones, $\langle x_N \rangle^{th} = 0$ and $\langle x_N^2 \rangle^{th} = N\ell^2$.

   (b) Calculate now the *averages* over many walkers for the instantaneous quantities $\langle x_i \rangle$, $\langle x_i^2 \rangle$ and $\langle (\Delta x)_i^2 \rangle$, and the final ones, $\langle x_N \rangle$, $\langle x_N^2 \rangle$ and $\langle (\Delta x)_N^2 \rangle$, and compare also these results with the theoretical values. What do you observe now?

   (c) Calculate the accuracy of the mean square displacement, given by the relative deviation of the numerical value with respect to the theoretical value:

$$\Delta = \left| \frac{\langle (\Delta x_N)^2 \rangle^{calc.}}{\langle (\Delta x_N)^2 \rangle^{th}} - 1 \right| .$$

   You should recognize that the larger is the number of walks for the average, the smaller is Δ. How many walkers are needed to obtain a "good" result, i.e. for istance with a relative accuracy $\Delta \leq 5\%$ with respect to the expected behavior?

   (d) (*optional)* Keep fix $p_\leftarrow = p_\rightarrow$=0.5 and vary N. Compare analytical and numerical results for $\langle x_N^2 \rangle - \langle x_N \rangle^2$ increasing N (Consider for instance N=8, 16, 32, 64). Does the number of walks necessary to obtain a given accuracy change with N?

(e) Fix $p_{\leftarrow} = p_{\rightarrow}$ and consider a number of walks (see point (c)) large enough to have a "good" accuracy for the numerical estimate of $\langle(\Delta x_N)^2\rangle$. Determine the dependence of $\langle(\Delta x_N)^2\rangle$ on N. *Hint: in this problem, plotting $\langle x_N^2\rangle - \langle x_N\rangle^2$ as a function of N should show directly a linear behavior: $\langle x_N^2\rangle - \langle x_N\rangle^2 \approx aN$. In general, making a log-log plot is convenient to exploit the power low of the dependence of $\langle x_N^2\rangle - \langle x_N\rangle^2$ on N, that we expect to be: $\langle x_N^2\rangle - \langle x_N\rangle^2 \approx aN^{2\nu}$. A linear fit in log-log form should give ν=1/2. Using powers of 2 for N, the data are equidistributed.*

(f) Insert in the program the calculation of the distribution $P_N(x)$ (numerically, from the simulation) and its expected behaviour:

$$P_N^{th}(x) = \frac{N!}{\left(\frac{N}{2}+\frac{x}{2}\right)!\left(\frac{N}{2}-\frac{x}{2}\right)!} p_{\rightarrow}^{\frac{N}{2}+\frac{x}{2}} p_{\leftarrow}^{\frac{N}{2}-\frac{x}{2}}$$

(Attention: better to calculate $(N \pm x)/2$ rather than $N/2 \pm x/2$... Why?) Consider again the case $p_{\leftarrow} = p_{\rightarrow} = 0.5$ and $N = 8$ and plot $P_N(x)$ and $P_N^{th}(x)$ vs x. Compare the two distributions. Is $P_N(x)$ a continuous function? How can you explain its behaviour?

(g) For sufficiently large N, $P_N(x)$ can be approximated with the gaussian distribution:

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp[-(x-\langle x\rangle)^2/2\sigma^2]$$

where $\sigma^2 = \langle(\Delta x)^2\rangle$.
Verify by calculating numerically $P_N(x)$ for N=8, 16, 32, 64 and comparing it with $P(x)$, where σ^2 is numerically estimated. Discuss the results.

Hint: (the calculation with the analytical espression containing $N!$ is discouraged...)

(h) *(optional)* You can consider random walks with steps of different length, drawn for instance from a uniform, a gaussian, a lorentzian, or a Student-t distribution...:

$$p(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

$$p(x) = \frac{1}{\pi\,(1+x^2)} \quad (\text{Student} - \text{t with } n = 1$$

or standard Cauchy or Lorentz or Breit-Wigner)

$$p(x) = \frac{1}{(2+x^2)^{\frac{3}{2}}} \quad (\text{Student} - \text{t with } n = 2)$$

$$p(x) = \frac{1}{(2+x^2)^{\frac{3}{2}}} \quad (\text{Student} - \text{t with } n = 2)$$

2. **2D Random walks**

(a) Write a program for the numerical simulation of 2D random walks with equal probabilities of moving in each direction. *(See for instance* `rw2d.f90` *in the continuum.)*

(b) Calculate $\langle \Delta R_N^2 \rangle = \langle x_N^2 \rangle + \langle y_N^2 \rangle - \langle x_N \rangle^2 - \langle y_N \rangle^2$ for N=8, 16, 32, 64 (use a "reasonable" number of *nruns*). Make a log-log plot of $\langle \Delta R_N^2 \rangle$ vs. $N^{2\nu}$ and estimate ν.

(c) Consider different algorithms to randomly choose the displacements and compute again $\langle \Delta R_N^2 \rangle$ vs. $N^{2\nu}$ to estimate ν. Do you see any change?

(d) Consider now the 2D RW on a square lattice *(See for instance the part of code suggested below).*

(e) Repeat calculation as in (b) and discuss the results.

(f) Modify the program in order to have $p_{\rightarrow}$=0.4 (random walk with a "drift") and the other probabilities equal. Calculate again the N dependence of $\langle \Delta R_N^2 \rangle$, using N=8, 16, 32, 64. Discuss the results.

(g) Simulate a rain drop falling down from a given height h in presence of wind going nowhere (e.g. put: $p_{\rightarrow} = p_{\leftarrow} = 0.15$, $p_{\uparrow} = 0.1$, $p_{\downarrow} = 0.6$). Let T be the average time necessary to reach the ground (use proper units of time and lenght). What about $T = T(h)$? If X is the displacement measured on the ground from a perfectly vertical fall down, which is the h and T dependence of $\langle \Delta X^2 \rangle$? Is it possible to define a vertical average velocity?

```
! rw1d.f90
! A simple random walk program  in 1D.
!
program rw1d
  implicit none
  integer ::         N  ! number of steps
  integer :: icount1, icount2, icount_rate, ix, irun, istep, nruns
  real, dimension(:), allocatable :: rnd          ! array of random numbers
  integer, dimension(:), allocatable :: x_N, x2_N ! sum of deviations and
  ! squares over the runs
  integer, dimension(:), allocatable :: P_N  ! final positions, sum over runs

  print *, "Enter number of steps, number of runs\rangle "
  read *, N, nruns

  allocate(rnd(N))
  allocate(x_N(N))
  allocate(x2_N(N))
  allocate(P_N(-N:N))
  x_N  = 0
  x2_N = 0
  P_N  = 0

  do irun = 1, nruns
     ix = 0 ! initial position of each run
     call random_number(rnd) ! get a sequence of random numbers
     do istep = 1, N
        if (rnd(istep) <  0.5) then ! random move
           ix = ix - 1 ! left
        else
           ix = ix + 1 ! right
        end if
        x_N (istep) = x_N (istep) + ix
        x2_N(istep) = x2_N(istep) + ix**2
     end do
     P_N(ix) = P_N(ix) + 1 ! accumulate (only for istep = N)
  end do

  print*,"# N=",N,"  nruns=",nruns
  print*,"# <x_N>      = ",real(x_N(N))/nruns
  print*,"# <x^2_N>  = ",real(x2_N(N))/nruns
  print*,"# <x^2_N> - <x_N>^2 = ",real(x2_N(N))/nruns-(real(x_N(N))/nruns)**2

  open(1,file="P_N",STATUS="REPLACE", ACTION="WRITE")
  write(1,*)"# N=",N,"  nruns=",nruns
  write(1,*)"# <x_N>      = ",real(x_N(N))/nruns
```

```
  write(1,*)"# <x^2_N>  = ",real(x2_N(N))/nruns
  write(1,*)"# <x^2_N> - <x_N>^2 = ",real(x2_N(N))/nruns-(real(x_N(N))/nruns)**2
  write(1,*)" "

  write(1,*)"# N, mean deviations, mean squared deviations, sigma^2"
  do ix = - N, N
     write(1,*)ix,real(P_N(ix))/nruns
  end do

  close(1)
end program rw1d


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! a part of code in fortran 90 simulating 2D random walks on a square lattice,
! making use of:
!     floor(a)      largest integer < = a
!     select case   similar to  ‘‘if’’, select different instructions
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! In a RW steps left,right up or down are chosen at random by taking a random
! number rand in the interval 0-1 with the built-in subroutine random_number.
! Since   0 < = rand  < 1        ==>        floor(rand*4) = 0 or = 1 or = 2 or = 3
! Steps to the right for 0 < = rand < 0.25, i. e. for floor(rand*4)=0  etc.
! The vector Ndir(0:3) contains how many steps are taken in each direction:
! right (Ndir(0)), left (Ndir(1)), up (Ndir(2)), down (Ndir(3))
! X and Y cartesian coordinates of walker during a walk of total N steps

        do j=1,...
                call random_number(rand)
                select case(floor(rand*4))
                        case(0)
                                Ndir(0)=Ndir(0)+1
                                X=X+1
                        case(1)
                                Ndir(1)=Ndir(1)+1
                                X=X-1
                        case(2)
                                Ndir(2)=Ndir(2)+1
                                Y=Y+1
                        case(3)
                                Ndir(3)=Ndir(3)+1
                                Y=Y-1
                end select
        end do
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! rw2d.f90
! A simple random walk program  in 2D.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROGRAM drunk
  IMPLICIT NONE
  INTEGER :: i, N
  REAL :: phi, rnd

  REAL :: x=0.0, y=0.0        ! Put drunk initially at the origin
  INTEGER, PARAMETER :: out=1 ! Set output unit

  REAL, PARAMETER :: step=1.0, twopi=2.0*3.1415926 ! step size and constants
  CHARACTER(LEN=15) :: filein
  CHARACTER(LEN=15), SAVE :: FORMAT1 = "(1i5,1x,2F14.7)"

  PRINT*,"Enter number of steps:"
  READ*, N
  PRINT*,"Enter file for data"
  READ*,filein
  OPEN(out, FILE=filein, STATUS="REPLACE", ACTION="WRITE")

  CALL RANDOM_SEED(PUT=seed)
  i = 0
  WRITE(UNIT=out,FMT=FORMAT1)i,x,y

  DO  i=1, N
     CALL RANDOM_NUMBER(rnd)
     phi=twopi*rnd
     x=x+step*COS(phi)
     y=y+step*SIN(phi)
     WRITE(UNIT=out,FMT=FORMAT1)i,x,y
  END DO

  CLOSE(out)

END PROGRAM drunk
```

3. **Brownian motion** *Optional*

It has been proved that the brownian motion of large heavy particles suspended e.g. in water (made of lighter smaller particles) can be dealt with statistical methods, without worrying about the details of the dynamics of the small molecules of the solvent. The final result for the velocity V_q and the position X_q of the heavy particle of mass M at the time $q+1$, after many collisions in random directions with the smaller lighter particles, is (*):

$$V_{q+1} = V_q - (\gamma/M)V_q\Delta t + w_q\sqrt{2\gamma k_B T\Delta t}/M$$

$$X_{q+1} = X_q + V_{q+1}\Delta t$$

where w is a random variable with standard Gaussian distribution and γ, the drag coefficient, can be expressed as: $\gamma = 6\pi\eta P$, using Stokes formula for a sphere of radius P describing the heavy particle in a solvent of viscosity η. Δt is the time interval for the discretization of the motion equation.

This algorithm is implemented in `brown1.f90`. For a speck of pollen in water at room temperature, the physical parameters are: k_BT=$4{\cdot}10^{-21}J$, M=$1.4{\cdot}10^{-10}kg$, and (from reasonable values of η and P) γ=$8{\cdot}10^{-7}Ns/m$.

(a) Verify that the mean square displacement $\langle(\Delta X)^2\rangle$ averaged over many heavy particles is linear in time after an initial transient.

(b) Estimate numerically the diffusion coefficient D from the slope of this linear behaviour, since $\langle(\Delta X)^2\rangle = 2dDt$ (d is the dimensionality of the system, $d = 2$ in our case).

(c) Verify the robustness of the result on the choice of Δt.

(d) Einstein provided a key relation between the diffusion coefficient D and solvent viscosity η:

$$D = k_BT/(6\pi\eta P),$$

where T is the temperature, $k_B = R/N_A$ is the Boltzmann constant, R the gas constant, N_A the Avogadro's number. Verify the validity of the Einstein relation from your numerical estimate of D and the input parameter of your simulation (η, T, P).

(e) Repeat for reasonable different values of M, T and γ.

(*) after: De Grooth BG (1999), *A simple model for Brownian motion leading to Langevin equation*, Am. J. Phys. **67**, 1248;
see also: G. Pastore and M. Peressi, *Doing physics with a computer in High Scools: designing and implementing numerical experiment*, in *Proceedings of MPTL-14* (http://www.fisica.uniud.it/URDF/mptl14/proceeding.htm)

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Brownian motion
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROGRAM Brown
  IMPLICIT NONE
  INTEGER                              :: npart,it,nit,i,j
  REAL,DIMENSION(:,:),allocatable    :: pos,pos0,vel,f
  REAL,DIMENSION(:), allocatable               :: mass
  REAL,DIMENSION(2)                    :: harvest ! array with 2 random numbers
  REAL                                 :: dt,gamma,t,w,msq

  WRITE(*,*)"Insert the number of heavy particles  : "
  READ*,npart
  allocate(pos(2,npart))
  allocate(pos0(2,npart))
  allocate(vel(2,npart))
  allocate(f(2,npart))
  allocate(mass(npart))
  WRITE(*,*)"Insert mass of the heavy particles (in kg)  : "
  READ*,mass(1)
  mass(2:npart)=mass(1)
  WRITE(*,*)"Insert time step (in seconds) :"
  READ*,dt
  WRITE(*,*)"Insert number of iterations :"
  READ*,nit
  WRITE(*,*)"Insert gamma and kT (in J) : "
  READ*,gamma,t

  vel = 0    ! Zero initial positions and velocities
  pos0 = 0
  it = 0
  pos = pos0

  ! CALL f_ext(pos,f)  ! in case of external force to be added
  f = 0                ! here no external force: only drug and random forces

  WRITE(1,*)"# iteration, time, pos_x, pos_y, vel_x, vel_y of particle 1"
  WRITE(1,*)it,it*dt,pos(1,1),pos(2,1),vel(1,1),vel(2,1)

  DO it=1,nit
     DO j=1,npart
        DO i=1,2
           call gasdev(w)
           vel(i,j) = vel(i,j)*( 1 - gamma*dt/mass(j))  + dt  * f(i,j)/mass(j) &
                + w*sqrt(2*gamma*t*dt)/mass(j)
           pos(i,j) = pos(i,j) + vel(i,j) * dt
```

```
        END DO

      END DO

      !CALL f_ext(pos,f)
      msq = sum( (pos - pos0)**2 )/npart

      WRITE(unit=1,fmt=*)it,it*dt,pos(1,1),pos(2,1),vel(1,1),vel(2,1)
      WRITE(unit=2,fmt=*)it,it*dt,msq

  END DO
  close(1)
  close(2)
  stop

contains

  SUBROUTINE gasdev(rnd)
    IMPLICIT NONE
    REAL, INTENT(OUT) :: rnd
    REAL :: rsq,v1,v2
    REAL, SAVE :: g
    LOGICAL, SAVE :: gaus_stored=.false.
    if (gaus_stored) then
       rnd=g
       gaus_stored=.false.
    else
       do
          call random_number(v1)
          call random_number(v2)
          v1=2.*v1-1.
          v2=2.*v2-1.
          rsq=v1**2+v2**2
          if (rsq \rangle  0. .and. rsq \langle  1.) exit
       end do
       rsq=sqrt(-2.*log(rsq)/rsq)
       rnd=v1*rsq
       g=v2*rsq
       gaus_stored=.true.
    end if
  END SUBROUTINE gasdev

  END PROGRAM Brown
```