Problema 02 - Códigos

random_walk_2d.f90

```
! Problema 02
program random_walk_2d
    use module_precision

    implicit none
    integer(sp)            :: seed,seed_val(8)                            ! semilla
    integer(sp), parameter :: n_step_total=1000000_sp                     ! numero total de random walks
    integer(sp), parameter :: switch=3_sp                                 ! cambiar si se quieren escribir los datos
    integer(sp), parameter :: n_step=1000_sp                              ! numero de pasos de c/ random walk
    real(dp)               :: x,y,x_old,y_old,dcm,dcm_tot                 ! pasos y desplazamiento cuadrático medio
    real(dp)               :: cuad01,cuad02,cuad03,cuad04                 ! contadores en c/ cuadrante
    real(dp)               :: cuad01_tot,cuad02_tot,cuad03_tot,cuad04_tot ! contadores en c/ cuadrante
    integer(sp)            :: i,j,istat
    integer(sp)            :: rnd_type                                    ! tipo de random generator
    real(dp)               :: suma                                        ! variable de control

    open(10,file='../results/result.dat',status='replace',action='write',iostat=istat)
    select case(switch)
        case(1) ! mapa random walk
            open(10,file='../results/result_01.dat',status='replace',action='write',iostat=istat)
            21 format(A12,x,A12); write(10,21) 'x-coord','y-coord'
        case(2) ! inciso a
            open(10,file='../results/result_02.dat',status='replace',action='write',iostat=istat)
            22 format(A12,x,A12); write(10,22) 'n_step','dcm'
            23 format(I12,x,E12.4)
        case(3) ! inciso b
            open(10,file='../results/result_03.dat',status='replace',action='write',iostat=istat)
            24 format(5(A12,x),A12); write(10,24) 'j','N/4','cuad01','cuad02','cuad03','cuad04'
            25 format(I12,x,5(E12.4,x),E12.4)
    end select
    if (istat /= 0_sp) write(*,*) 'istat_error=', istat

    rnd_type=4_sp ! elegir random generator
    do j=100_sp,n_step,100_sp
        write(*,*) j
        ! generamos la semilla
        call date_and_time(values=seed_val)
        seed=seed_val(8)*seed_val(7)*seed_val(6)+seed_val(5)
        x_old=0._dp;y_old=0._dp;dcm_tot=0._dp
        cuad01_tot=0._dp;cuad02_tot=0._dp;cuad03_tot=0._dp;cuad04_tot=0._dp
        do i=1,n_step_total
            call walk_2d(switch,10,rnd_type,seed,j,x_old,y_old,x,y,dcm,cuad01,cuad02,cuad03,cuad04)
            cuad01_tot=cuad01_tot+cuad01;cuad02_tot=cuad02_tot+cuad02
            cuad03_tot=cuad03_tot+cuad03;cuad04_tot=cuad04_tot+cuad04
            dcm_tot=dcm_tot+dcm
            if (switch==1_sp) then;x_old=x;y_old=y; end if
        end do

        if (switch==2_sp) write(10,23) j,dcm_tot*(1._dp/n_step_total)

        cuad01=cuad01_tot;cuad02=cuad02_tot;cuad03=cuad03_tot;cuad04=cuad04_tot
        suma=(cuad01+cuad02+cuad03+cuad04)*0.25_dp

        if (switch==3_sp) write(10,25) j,real(n_step_total)*0.25_dp,cuad01,cuad02,cuad03,cuad04,suma
    end do
    close(10)
end program random_walk_2d

! subrutina para realizar caminata aleatoria de n_step pasos específicos
! partiendo de cierto origen especificoe, elegir tipo pseudo generador random,
! devolver, semilla y contar cuantos pasos cayeron en determinado cuadrante
subroutine walk_2d(switch,n_file,rnd_type,seed,n_step,x0,y0,x,y,dcm,&
    count_cuad_01,count_cuad_02,count_cuad_03,count_cuad_04)
    use module_precision;use module_random_generator
    use module_mzran;use module_mt19937

    implicit none
    integer(sp), intent(in)    :: switch,n_file ! prender o apagar escritura de datos
    integer(sp), intent(in)    :: n_step     ! numero de pasos totales
    real(dp),    intent(in)    :: x0,y0      ! cordenadas iniciales
    integer(sp), intent(inout) :: rnd_type,seed
    real(dp),    intent(out)   :: x,y      ! coordenadas
    real(dp),    intent(out)   :: count_cuad_01,count_cuad_02,&
                                  count_cuad_03,count_cuad_04
    real(dp),   intent(out)    :: dcm       ! desplazamiento cuadrático medio

    !real(dp),    parameter :: px=0.5_dp,py=0.5_dp ! probabilidades de pasos
    real(dp),    parameter :: step=1._dp           ! longitud de paso (fija)
    integer(sp)            :: i
    real(dp)              :: nrand                ! numero pseudo-aleatorio

    if (switch==1_sp) then;20 format(E12.4,x,E12.4);write(n_file,20) x,y;end if
    ! posición inicial
    x=x0;y=y0;count_cuad_01=0._dp;count_cuad_02=0._dp;count_cuad_03=0._dp;count_cuad_04=0._dp
    !if (rnd_type==4_sp) call sgrnd(seed)
    do i=1,n_step
        select case(rnd_type)
                case(1);nrand=ran0(seed)                       ! ran0 random generator
                case(2);nrand=ran2(seed)                       ! ran2 random generator
                case(3);nrand=rmzran()                         ! mzran random generator
                case(4);nrand=real(grnd(),dp) ! mt19937 random generator
                case default; write(*,*) 'Invalid random generator type'
        end select
        cond1:   if (0._dp<=nrand.and.nrand<0.25_dp)  then; x=x+step; exit cond1
            else if (0.25_dp<=nrand.and.nrand<0.5_dp) then; y=y+step; exit cond1
            else if (0.5_dp<=nrand.and.nrand<0.75_dp) then; x=x-step; exit cond1
            else if (0.75_dp<=nrand.and.nrand<1._dp)  then; y=y-step; exit cond1
        end if cond1
        if (switch==1_sp) write(n_file,20) x,y
    end do
    ! Determinamos el cuadrante de la partícula al final de la caminata
    cond2:   if (x>0._dp.and.y>0._dp) then; count_cuad_01=count_cuad_01+1._dp; exit cond2
        else if (x<0._dp.and.y>0._dp) then; count_cuad_02=count_cuad_02+1._dp; exit cond2
        else if (x<0._dp.and.y<0._dp) then; count_cuad_03=count_cuad_03+1._dp; exit cond2
        else if (x>0._dp.and.y<0._dp) then; count_cuad_04=count_cuad_04+1._dp; exit cond2
        else if (x==0._dp.and.y==0._dp) then     ! origen de coordenadas
        count_cuad_01=count_cuad_01+0.25_dp;count_cuad_02=count_cuad_02+0.25_dp
        count_cuad_03=count_cuad_03+0.25_dp;count_cuad_04=count_cuad_04+0.25_dp; exit cond2
        else if (x>0._dp.and.y==0._dp) then ! semi-eje x positivo
        count_cuad_01=count_cuad_01+0.5_dp;count_cuad_04=count_cuad_04+0.5_dp; exit cond2
        else if (x<0._dp.and.y==0._dp) then ! semi-eje x negativo
        count_cuad_02=count_cuad_02+0.5_dp;count_cuad_03=count_cuad_03+0.5_dp; exit cond2
        else if (x==0._dp.and.y>0._dp) then ! semi-eje y positivo
        count_cuad_01=count_cuad_01+0.5_dp;count_cuad_02=count_cuad_02+0.5_dp; exit cond2
        else if (x==0._dp.and.y<0._dp) then ! semi-eje y negativo
        count_cuad_03=count_cuad_03+0.5_dp;count_cuad_04=count_cuad_04+0.5_dp; exit cond2
    end if cond2
    if (switch==2_sp) dcm=(x-x0)*(x-x0)+(y-y0)*(y-y0)
end subroutine walk_2d
```

Problema 03 - Códigos

mc_integration.f90

```
! problema 03
program mc_integration
    use module_precision
    use module_random_generator
    implicit none

    integer(sp), parameter :: pot=2_sp   ! potencia
    integer(sp)            :: n ! cantidad de evaluaciones de la función
    real(dp),    parameter :: x_start=0._dp,x_end=1._dp
    real(dp),    parameter :: exact_integ=1._dp/(real(pot,dp)+1._dp)
    integer(sp)            :: seed,seed_val(8),i,j,k,istat
    real(dp)               :: nrand,x_rand,f_rand,integ

    call date_and_time(values=seed_val)
    seed=seed_val(8)*seed_val(7)*seed_val(6)+seed_val(5)

    integ=0._dp
    open(10,file='../results/result_mc_integrator_pot2.dat',status='replace',action='write',iostat=istat) ! p/ pot=2
    !open(10,file='../results/result_mc_integrator_pot3.dat',status='replace',action='write',iostat=istat) ! p/ pot=3
    if (istat /= 0_sp) write(*,*) 'istat_error=', istat
    20 format(2(A12,x),A12); 21 format(I12,x,E12.4,x,E12.4)
    write(10,20) 'n','Iaprox','E_rel'
    do i=1,1E+04
```

```
            n=10_sp*i
        do j=1,n
            nrand=ran2(seed)
            x_rand=nrand*(x_end-x_start)+x_start ! genero un x random en el intervalo de integración
            f_rand=1._dp;do k=1,pot;f_rand=f_rand*x_rand;end do
            integ=integ+f_rand
        end do
        integ=(x_end-x_start)*(1._dp/real(n))*integ
        write(10,21) n,integ,abs((exact_integ-integ)*(1._dp/exact_integ))
    end do
    close(10)

    ! control de resultados
    write(*,'(A10,E12.4)') 'Iaprox=',integ
    write(*,'(A10,E12.4)') 'Iexact=',1._dp/(real(pot,dp)+1._dp)

end program mc_integration
```

mc_integration_imp_sampling.f90

```
! problema 03.b
! ojo la distribución debe normalizarse segun estos valores
! este programa sólo vale para cuando x_start=0; x_end=1
program mc_integration_imp_sampling
    use module_precision;use module_random_generator
    implicit none

    integer(sp), parameter :: pot=3_sp   ! potencia (debe ser mayor a -2)
    integer(sp), parameter :: potk=3_sp  ! usar valores 2 y 3
    integer(sp)            :: n ! cantidad de evaluaciones de la función
    real(dp),    parameter :: x_start=0._dp,x_end=1._dp
    real(dp),    parameter :: exact_integ=1._dp/(real(pot,dp)+1._dp)
    integer(sp)            :: seed,seed_val(8),i,j,k,istat
    real(dp)               :: nrand,x_rand,f_rand,g_rand,integ
    real(dp)               :: factor

    call date_and_time(values=seed_val)
    seed=seed_val(8)*seed_val(7)*seed_val(6)+seed_val(5)

    integ=0._dp
    !open(10,file='../results/result_P03b_01.dat',status='replace',action='write',iostat=istat) ! p/ potk=2
    open(10,file='../results/result_P03b_02.dat',status='replace',action='write',iostat=istat)  ! p/ potk=3
    if (istat /= 0_sp) write(*,*) 'istat_error=', istat
    20 format(2(A12,x),A12);21 format(I12,x,E12.4,x,E12.4)
    write(10,20) 'n','Iaprox','E_rel'
    do i=1,1E+04
            n=10_sp*i
        do j=1,n
            nrand=ran0(seed)
            ! x^k distribution (x \in {1,Infinity})
            x_rand=nrand**(1_dp/real(potk+1_dp,dp))
            f_rand=1._dp;do k=1,pot;f_rand=f_rand*x_rand;end do  ! x_rand**pot
            factor=1._dp;do k=1,potk;factor=factor*x_rand;end do ! x_rand**potk
            g_rand=(real(potk,dp)+1_dp)*factor
            integ=integ+f_rand*(1._dp/g_rand)
        end do
        integ=(x_end-x_start)*(1._dp/real(n))*integ
        write(10,21) n,integ,abs((exact_integ-integ)*(1._dp/exact_integ))
    end do
    close(10)

    ! controlamos valores de la integral en el último paso
    write(*,'(A10,E12.4)') 'Iaprox=',integ
    write(*,'(A10,E12.4)') 'Iexact=',1._dp/(real(pot,dp)+1._dp)

end program mc_integration_imp_sampling
```

Problema 04 - Códigos

hyper_sphere.f90

```
!Problema 04
```

```fortran
program hyper_sphere
    use module_precision

    implicit none
    ! variables generales
    real(dp),    parameter :: x_end=1._dp,x_start=0._dp
    integer(sp)            :: n ! dimensiones
    integer(sp)            :: i,j,k,l,istat
    real(dp)               :: exact_volume,volumen
    real(dp)               :: t_start,t_end
    ! variables para método del trapecio
    real(dp)               :: Iaprox,factor
    ! variables para método de monte carlo
    integer(sp), parameter :: n_random=10**6_sp
    real(dp),    parameter :: x_med=0._dp,sigma_max=1._dp,sigma_min=0.1_dp
    integer(sp)            :: seed,seed_val(8),sigma_n
    real(dp)               :: x_rand,f_rand,integ,r
    real(dp)               :: g_inv_rand,sigma,sigma_step,rel_err
    real(dp)               :: gauss_dist,heaviside,gaussdev

    open(10,file='../results/result_P04a_01.dat',status='replace',action='write',iostat=istat)
    20 format(I14,x,2(E14.6,x),E14.6); 21 format(3(A14,x),A14)
    write(10,21) 'n-dimension','tr_volumen','ex_volumen','rel_err'
    if (istat /= 0_sp) write(*,*) 'istat_error=', istat

    !trapez_integral(m,a,b,n,Iaprox)
    volumen=1._dp
    do j=1_sp,4_sp
        n=j ! n={1,2,3,4}
        volumen=1._dp
        do i=1,n-1
            !call trapez_integral(((2**24)/n),x_start,x_end,i,Iaprox)
            call trapez_integral(int((2._dp**(24._dp/real(n,dp))),sp)+1_sp,x_start,x_end,i,Iaprox)
            write(*,*) n,int((2._dp**(24._dp/real(n,dp))),sp)+1_sp
            volumen=volumen*Iaprox
        end do
        factor=1._dp;do i=1,n;factor=factor*2._dp;end do ! 2**n
        volumen=volumen*factor
        rel_err=abs(exact_volume(n)-volumen)*(1._dp/exact_volume(n))
        write(10,20) n,volumen,exact_volume(n),rel_err
    end do
    close(10)

    open(11,file='../results/result_P04b_01.dat',status='replace',action='write',iostat=istat)
    30 format(I12,x,5(E12.4,x),E12.4)); 31 format(6(A12,x),A12)
    write(11,31) 'n-dimension','elapsed time','mc_volumen','rel_err','sigma','ex_volumen','n-ball/n-cube'
    if (istat /= 0_sp) write(*,*) 'istat_error=', istat

    open(12,file='../results/result_P04b_02.dat',status='replace',action='write',iostat=istat)
    32 format(I12,x,E12.4,x,E12.4); 33 format(2(A12,x),A12)
    write(12,33) 'n-dimension','ex_volumen','n-ball/n-cube'
    if (istat /= 0_sp) write(*,*) 'istat_error=', istat

    ! monte carlo
    call date_and_time(values=seed_val)
    seed=seed_val(8)*seed_val(7)*seed_val(6)+seed_val(5)

    n=100_sp ! numero máximo de dimensiones
    do l=1_sp,n
        sigma_n=100_sp*l
        sigma_step=abs(sigma_max-sigma_min)*(1._dp/(real(sigma_n,dp)-1._dp))
        call cpu_time(t_start)
        do1: do k=1,sigma_n
            sigma=sigma_min+sigma_step*(real(k,dp)-1._dp)
            integ=0._dp
            do j=1,n_random
                r=0._dp;g_inv_rand=1._dp
                do i=1,l
                    x_rand=gaussdev(seed,1_sp)
                    x_rand=(sigma*x_rand+x_med)
                    r=r+x_rand*x_rand
                    g_inv_rand=g_inv_rand*(1._dp/gauss_dist(x_rand,x_med,sigma))
                end do
                r=sqrt(r) ! (x1^2+x2^2+...+xn^2)^(1/2)
                f_rand=heaviside(r)
                integ=integ+f_rand*g_inv_rand
            end do
            integ=(x_end-x_start)*(1._dp/real(n_random,dp))*integ
            volumen=integ
            rel_err=abs(exact_volume(l)-volumen)*(1._dp/exact_volume(l))
            if (rel_err<=0.001) then;write(*,*), 'VERIFICA';exit do1;end if
        end do do1
        call cpu_time(t_end)
        write(11,30) l,(t_end-t_start),volumen,rel_err,sigma,exact_volume(l),exact_volume(l)*
(1._dp/(2_dp**real(l,dp)))
        write(12,32) l,exact_volume(l),exact_volume(l)*(1._dp/(2_dp**real(l,dp)))
    end do
    close(11)

end program hyper_sphere

subroutine trapez_integral(m,a,b,n,Iaprox)
    use module_precision
    implicit none
    ! Data dictionary: declare calling parameter types & definitions
    integer(sp), intent(in)     :: m      ! cantidad puntos => m = n + 1, n intervals number
    real(dp),    intent(in)     :: a,b    ! límites de integración
    real(dp),    intent(out)    :: Iaprox ! numerical integration with trapezoidal method
    integer(sp), intent(in)     :: n         ! dimension hyper-sphere
    ! Data dictionary: declare local variables types & definitions
    integer(sp) :: i ! index loop
    real(dp)    :: h,x_current
    real(dp)    :: function_vector(1,m),coeff_vector(m,1),Iaprox_aux(1,1)

    h=abs((b-a))*(1._dp/(real(m,dp)-1._dp)) ! paso de integración
    x_current=a
    coeff_vector(:,1)=2._dp
    do i=2,m-1
        x_current=x_current+h
        function_vector(1,i)=sqrt((1._dp-x_current*x_current)**real(n,dp))
    end do
    coeff_vector(1,1)=1._dp;coeff_vector(m,1)=1._dp
    function_vector(1,1)=sqrt((1._dp-a*a)**real(n,dp))
    function_vector(1,m)=sqrt((1._dp-b*b)**real(n,dp))
    Iaprox_aux=h*matmul(function_vector,coeff_vector)*0.5_dp
    Iaprox=Iaprox_aux(1,1)
end subroutine trapez_integral

function heaviside(r)
    use module_precision
    implicit none
    real(dp), intent(in) :: r
    real(dp)             :: heaviside
    if (r<=1._dp) heaviside=1._dp
    if (r>1._dp) heaviside=0._dp
end function heaviside

! to calculate 1D gauss distribution
function gauss_dist(x,x_med,sigma)
    use module_precision
    implicit none
    real(dp), intent(in) :: x,x_med,sigma
    real(dp),  parameter :: pi=4._dp*atan(1._dp)
    real(dp)             :: gauss_dist,factor_01,factor_02
    factor_01=1._dp/(sqrt(2._dp*pi)*sigma)
    factor_02=(x-x_med)*(1._dp/sigma)
    gauss_dist=factor_01*exp(-0.5_dp*factor_02*factor_02)
end function gauss_dist
```

```
! To calculate de exact expresion for hyper-sphere's volume
function exact_volume(n)
    use module_precision
    implicit none
    integer(sp), intent(in) :: n
    real(dp),    parameter  :: pi=4._dp*atan(1._dp)
    real(dp)                :: exact_volume,factor_02,factorial
    integer(sp)             :: factor_01,i
    cond1: if (mod(n,2_sp)==0_sp) then ! n pares
        factor_01=n/2_sp
        exact_volume=(pi**real(factor_01,dp))*(1._dp/factorial(factor_01))
        exit cond1
    else ! n impares
        factor_01=(n-1_sp)/2_sp
        factor_02=1._dp;do i=1,n;factor_02=factor_02*2._dp;end do ! factor_02=2**n
        exact_volume=(pi**real(factor_01,dp))*factor_02*factorial(factor_01)*(1._dp/factorial(n))
        exit cond1
    end if cond1
end function exact_volume

! To calculate the factorial function
recursive function factorial(n) result(fact)
    use module_precision
    implicit none
    integer(sp), intent(in) :: n
    real(dp)                :: fact
    if (n>=1_sp) fact=real(n,dp)*factorial(n-1_sp)
    if (n==0_sp) fact=1_dp
end function factorial

function gaussdev(seed,rnd_type)
    use module_precision;use module_random_generator
    use module_mzran;use module_mt19937
    implicit none
    integer(sp), intent(in) :: seed, rnd_type
    integer(sp)             :: iset=0_sp
    real(dp),    parameter  :: pi=4._dp*atan(1._dp)
    real(dp) :: gset,gaussdev,nrand_01,nrand_02
    save iset,gset
    ! queda pendiente agregar más rnd_type para incluir
    !  distintos generadores
    select case(rnd_type)
    case(1);nrand_01=ran2(seed);nrand_02=ran2(seed)
    case(2);nrand_01=ran0(seed);nrand_02=ran0(seed)
    case(3);nrand_01=rmzran();nrand_02=rmzran()
    case(4);call sgrnd(seed);nrand_01=real(grnd(),dp);nrand_02=real(grnd(),dp)
    end select
    if (iset==0_sp) then
        gset=sqrt(-2*log(nrand_01))*cos(2*pi*nrand_02)
        gaussdev=sqrt(-2*log(nrand_01))*sin(2*pi*nrand_02)
        iset=1_dp
    else;gaussdev=gset;iset=0_sp;end if
end function gaussdev
```