

Exercise Lecture X

Lattice gas, Diffusion Limited Aggregates, fractal models of surface growth.

(the first exercise is mandatory)

1. Self-diffusion coefficient in a lattice gas model

Consider a finite square lattice with sites randomly occupied by particles with a given density ρ . The particles can move randomly to *empty* nearest sites (two particles can not occupy the same site). It is an example of a restricted random walk. A meaningful physical quantity is the *self-diffusion coefficient* D of an individual particle. D is the limit $t \rightarrow \infty$ of $D(t)$, where $D(t)$ is given by:

$$D(t) = \frac{1}{2dt} \langle \Delta R^2(t) \rangle,$$

with d which is the dimensionality of the system and $\langle \Delta R^2(t) \rangle$ is the net mean square displacement per particle, averaged over all particles, after t units of time ($\langle \dots \rangle$ here indicates the average over particles and *not* temporal averages).

The model can be summarized by the following algorithm:

- i) Occupy at random the $L \times L$ sites of a square lattice with N particles subject to the condition that no double occupancy is allowed, with the desired density $\rho = N/L^2 \leq 1$. Tag each particle, that is, distinguish it from the others, and record its initial position in an array.
- ii) At each step choose a particle (randomly, or, alternatively, in an ordered way) and one of its nearest neighbor sites at random. If the neighbor site is empty, the particle is moved to this site; otherwise the particle remains in its present position. Loop over the particles.

Note 1: The measure of "time" in this context is arbitrary. The usual definition is that during one unit of time or one Monte Carlo step, each particle on average attempts one jump. Time goes on even if the particles do not move, i.e., the tentative move is not accepted.

Note 2: Consider periodic boundary conditions, but note that it is necessary to limit the number of Monte Carlo steps so that $\langle \Delta R^2(t) \rangle$ is smaller than $(L/2)^2$. Why?

Do a Monte Carlo simulation to determine D and its dependence on the particles concentration ρ .

See for instance the code `latticegas.f90`. For the sake of definiteness, 1 Monte Carlo time step is 1 ns and the unit length is 2 Å.

- (a) Study $\Delta R^2(t)$ as a function of time for a fixed value of ρ , for instance 0.03 or 0.2, and for a fixed number of particles (e.g., 13 particles in a 20×20 lattice for $\rho = 0.03$). What do you see increasing time (within the limit mentioned above)? Make a fit and compare your result (the slope) with the expected behavior of a standard random walk.
- (b) Plot $D(t)$ as a function of time: after a certain equilibration time, it fluctuates. Calculate the amplitude of the fluctuations as a function of t (from the distribution of data over the particles). These fluctuations remain also by increasing t .
- (c) In order to estimate of D , which is defined as the limit $t \rightarrow \infty$, do a temporal average $\langle D(t) \rangle$ (*⟨...⟩ here indicates a temporal average, for instance from 0 to t, or some block average*). Plot together $D(t)$ and $\langle D(t) \rangle$. Change the seed, do another run and compare the plot and the estimate of $D(t)$ and $\langle D(t) \rangle$ with the previous results.
- (d) Better statistics for $D(t)$ (and consequently for D) can be obtained by averaging $\Delta R^2(t)$ over many particles (i.e., for a given ρ , considering a lattice with L as large as possible; it is suggested $L \geq 40$). Verify that fluctuations of $D(t)$ (and the deviations of $\langle D(t) \rangle$ over more runs from its mean value) are proportional to the inverse square root of the number of particles.
- (e) Study the dependence of D on the concentration, using for instance $\rho=0.1, 0.2, 0.3, 0.5,$ and 0.7 . You should find that D is a monotonically decreasing function of ρ . Why?
- (f) To gain some insight into this dependence, determine the dependence on ρ of the probability that if a particle jumps to a vacancy at time t , it returns to its previous position at time $t + 1$. Is there a qualitative relation between the density dependence of D and this probability?


```

        print *, 'Number of particles > number of sites !!!'
        STOP 'Too small lattice'
endif

allocate(lattice(0:L-1,0:L-1))
allocate(x(Np),y(Np))
allocate(dx(Np),dy(Np))

! Mark all positions as empty
do i=0,L-1
  do j=0,L-1
    lattice(i,j) = .false.
  enddo
enddo

! Enumeration of directions: 1=right; 2=left; 3=up; 4=down
dxtrial(1)=+1; dytrial(1)= 0;
dxtrial(2)=-1; dytrial(2)= 0;
dxtrial(3)= 0; dytrial(3)=+1;
dxtrial(4)= 0; dytrial(4)=-1;

Nfail=0; njumps=0;
! Generate particles on lattice
do i=1,Np
  do ! Loop until empty position found
    ! To be on safe side, check that upper limit not reached
    call random_number(rnd)
    x(i)=int(rnd(1)*L); if (x(i)>=L) x(i)=L-1;
    y(i)=int(rnd(2)*L); if (y(i)>=L) y(i)=L-1;
    if (lattice(x(i),y(i))) then
      ! Position already filled, loop to find new trial
      cycle
    else
      lattice(x(i),y(i))=.true.
      ! Success, go to next particle
      exit
    endif
  enddo
  dx(i)=0.0d0; dy(i)=0.0d0;
enddo

T=0.0;
do istep=0,Nsteps-1 ! Loop over MC steps
  do isubstep=1,Np ! Do all particles on average once every MC step
    ! Pick one particle at random
    call random_number(rnd1)

```

```

i=int(rnd1*Np)+1; if (i>Np) i=Np;

! Find possible directions, store it in free()
Nfree=0
do j=1,4
  xnew(j)=x(i)+dxtrial(j);
  if (xnew(j) >= L) xnew(j)=0; if (xnew(j)<0) xnew(j)=L-1;
  ynew(j)=y(i)+dytrial(j);
  if (ynew(j) >= L) ynew(j)=0; if (ynew(j)<0) ynew(j)=L-1;
  if (.not. lattice(xnew(j),ynew(j))) then
    ! Success: position free
    nfree=nfree+1
    free(nfree)=j
  endif
enddo

! If no possible directions, get new particle
If (nfree == 0) then
  nfail=nfail+1
  cycle
endif
njumps=njumps+1

! Pick one of the possible directions randomly
! Note that the dir>nfree check here really is needed!
call random_number(rnd1)
dir=int(rnd1*nfree)+1; if (dir>nfree) dir=nfree
j=free(dir)

! Now x(i),y(i) is old position and xnew(j),ynew(j) new
! Double check that new site really is free
if (lattice(xnew(j),ynew(j))) then
  print *, 'ERROR: THIS SHOULD BE IMPOSSIBLE'
  print *, i,j,dir,nfree
  print *, free
  print *, x(i),y(i),xnew(j),ynew(j)
  STOP 'ERROR new site bug'
endif
!Empty old position and fill new
lattice(x(i),y(i))=.false.
lattice(xnew(j),ynew(j))=.true.

X(i)=xnew(j); y(i)=ynew(j);
dx(i)=dx(i)+dxtrial(j); dy(i)=dy(i)+dytrial(j);
enddo

```

```

If (mod(istep*Np,1000000) == 0) then
! Calculate and print intermediate results
! Get total displacement from dx,dy
dxsum=0.0d0; dysum=0.0d0;
dxsqsum=0.0d0; dysqsum=0.0d0;
do i=1,Np
  dxsum=dxsum+dx(i);  dysum=dysum+dy(i);
  dxsqsum=dxsqsum+dx(i)*dx(i);
  dysqsum=dysqsum+dy(i)*dy(i);
enddo
drsqave=(dxsqsum+dysqsum)/(1.0*Np)

if (t>0.0) then
! Get diffusion coefficient by proper scaling
D=drsqave*a*a/(4*t)
write(*,fmt='(3(a,1pe10.2))')&
  'At ',t,' <dR^2>=',drsqave*a*a,' D=',D,' cm^2/s'
endif

endif

t=t+deltat
enddo

! Get total displacement from dx,dy
dxsum=0.0d0; dysum=0.0d0;
dxsqsum=0.0d0; dysqsum=0.0d0;
do i=1,Np
  dxsum=dxsum+dx(i);  dysum=dysum+dy(i);
  dxsqsum=dxsqsum+dx(i)*dx(i); dysqsum=dysqsum+dy(i)*dy(i);
enddo
print *,'dxsum',dxsum,' dysum',dysum
print *,'dxsqsum',dxsqsum,' dysqsum',dysqsum

drsqave=(dxsqsum+dysqsum)/(1.0*Np)
print *,'drsqave',drsqave
print *,'Number of failed jumps',nfail,' number of successes',njumps
! Get diffusion coefficient by proper scaling
D=drsqave*a*a/(4*t)
write(*,fmt='(a,f6.4,a)')'For Np/L^2=',real(Np)/L**2,' : '
write(*,fmt='(3(a,1pe10.2))')&
  'at ',t,' <dR^2>=',drsqave*a*a,' D=',D,' cm^2/s'

deallocate (lattice,x,y,dx,dy)

end program latticegas

```

2. Diffusion Limited Aggregates (DLA)

- (a) Write a program to generate DLA on a square lattice. Here it is reported a code `dla2d.f90`. Choose each walker starting randomly at a distance $R = R_{max} + 2$ from the center, where R_{max} is the maximum distance of the particles already aggregated in the cluster from its origin. To save time, eliminate the walker that go too much far away, e.g. that reach a distance equal to $2R_{max}$ from the center (“killing circle”). Choose $L=31$. Try to color in a different way the sites according to the order of aggregation (e.g. after 20 particles aggregated change color). Which are the last aggregated? Which are the former?
- (b) At $t = 0$ we have 4 perimetral sites with a probability $p_i=1/4$ of being occupied. After having occupied one of them, we have 6 perimetral sites which have different occupancy possibility: two of them have $p_i=2/9$ and the other 4 have $p_i = 5/36$. Verify with a Monte carlo simulation.
- (c) The efficiency of the algorithm can be improved considering displacements with variable length, the longer the distance from the center, the longer is the step length. For instance, if the walker is at distance $R > R_{max}$, consider a length displacement $R - R_{max} - 1$ (if it is > 1), whereas consider a unitary displacement if the walker is close to the cluster already grown.
- (d) Generate some DLA cluster and calculate its fractal dimension, which should result $D = 1.66$ (see: Witter et al., Phys. Rev. Lett. 47, 1400 (1983)).


```

        end select
    if(any(abs(newpos) > Nw)) exit ! Walker stepped out of the box. Start a new one
    if(occupied(newpos(1),newpos(2))) then ! walker gets stuck to the crystal
        occupied(prevpos(1),prevpos(2)) = .true. ! Add the walker to the crystal
        distance = sqrt(real(dot_product(prevpos,prevpos)))
        if(distance > (radius-5)) radius = distance + 5
            ! Make the starting circle larger if necessary
        exit
    endif
    prevpos = newpos
        ! Walker made a valid move (didn't get stuck or wander away).
        ! Update and keep it moving
    enddo
    if(radius > Nw) exit
enddo

! Write occupied sites to disk
open(10,file="dla2d.data",status="replace")
do i = -Nw, Nw
    do j = -Nw, Nw
        if(occupied(i,j)) write(10,*) i,j
    enddo
enddo
close(10)

! Do the m(r) analysis and write results to disk
open(11,file="dlamass.data")
do idist = 2, int(0.75*distance)
    mass = 0
    do i = -Nw, Nw
        do j = -Nw, Nw
            if(occupied(i,j)) then
                if(idist**2 >= i**2 + j**2) mass = mass + 1
            endif
        enddo
    enddo
    write(11,'(2i10)') idist, mass
enddo
close(11)

end program dla2d

```

3. Fractal growth of surfaces

Consider the *Eden model* to generate a corrugated surface. The algorithm is:

- (a) choose randomly a lattice site and occupy it. The *nearest neighbor sites* of the occupied site (i.e. 4 sites in case of a square lattice) are the *perimetral sites*.
- (b) choose randomly a *perimetral site* and occupy it. When occupied, it is no longer a *perimetral site*: update the list of *perimetral sites* with the new ones. Repeat from (1).

The code `eden.f90` is proposed as a draft here (the suggestion is to modify it, it has several “print” for checks...)

Consider a simple model where the surface is initially (at time $t = 0$) an horizontal line of L occupied sites. The growth is along the vertical direction.

According to the Eden model, choose a *perimetral site* randomly and occupy it. For the initial configuration of our surface, at time $t = 0$ the *perimetral sites* are the horizontal line of empty sites adjacent to the line of occupied sites. The average height of the cluster is:

$$\bar{h} = \frac{1}{N_s} \sum_{i=1}^{N_s} h_i$$

where h_i is the distance of the i surface site from the initial line, and the sum is over all the surface sites N_s .

The deposition of a particle corresponds to the increment of time t by one. Study how the *roughness* w of the surface change with time, where w is defined as:

$$w^2 = \frac{1}{N_s} \sum_{i=1}^{N_s} (h_i - \bar{h})^2,$$

($w=0$ for a planar surface). w depends on L and t . Initially w increases with time:

$$w(L, t) \sim t^\beta$$

β measures the increasing in time of the correlations in the vertical direction. Given a characteristic time, the length for the correlation of the fluctuations is comparable with L , and the roughness w reaches a limiting value depending only on L . We can write:

$$w(L, t \gg 1) \sim L^\alpha,$$

where α measure the corrugation.

- (a) Consider a 1D surface growing over a line of $L=100$ sites and apply the Eden model. Consider x the horizontal index, i.e. the label of the columns, and h_x the height (max. distance of a perimetral site from the substrate). Use PBC in the horizontal direction. We call *surface sites* those *perimetral sites* with maximum h for a given x . Try to visualize the growth in time, with the evidence of the occupied, perimetral and surface sites.
- is the surface well defined?
 - where are most of the perimetral sites?
 - if we choose *all perimetral sites* as *surface sites*, is something changing?
- (b) Plot $w(t)$ as a function of t for $L=32, 64, 128$ and estimate the exponents α e β .
- Which kind of plot is it convenient to do?
 - Does w increase initially with a power law? If yes, estimate β .
 - Is there a characteristic time (depending on L) for w to reach an asymptotic value?
 - Can you estimate α ? (you should find $\beta = 1/3$ and $\alpha = 1/2$).
- (c) The dependence of w on L and t can be summarized with the law:

$$w(L, t) \approx L^\alpha f(t/L^{\alpha/\beta})$$

where : $f(x) \approx x^\beta$ for $x \ll 1$ and $f(x) = \text{constant}$ for $x \gg 1$

- Using for α and β the best estimates obtained in the previous point, verify the law plotting $w(L, t)/L^\alpha$ as a function of $t/L^{\alpha/\beta}$ for the different values of L considered. b) Repeat using instead the exact result, $\beta = 1/3$ and $\alpha = 1/2$. You should find a universal curve (i.e. the *same* curve using the scaled variables for different values of L)
- (d) *Random Deposition* In the Eden model each perimetral site can be part of the cluster. In the random deposition model, instead, a column is chosen randomly and a particle is deposited on top of it. No horizontal correlations are therefore present.
- Make a simulation with this model and visualize the surface.
 - Verify that the height of the columns follow a Poisson distribution and that $\bar{h} \sim t$ and $w \sim t^{1/2}$. This structure does not depend on L and therefore $\alpha = 0$.
- (e) *Balistic Deposition* In this model the horizontal coordinate is chosen randomly and a particle falls down up to reach the first available perimetral site which is a nearest neighbor of an occupied site. This algorithm allows also a horizontal growth. Consider one particle falling down for each unit time. Discuss the differences -in terms of algorithm and results- with respect to the previous models.


```

!do i= 1, nmcs
!   w(i) = 0.0_d
!   hmed(i) = 0.0_d
!end do

end subroutine init

!eden model

subroutine edengen(grid,Lx,Ly,v,s)

integer,intent(inout) :: v,Lx,Ly
integer :: i,ccp,j
integer, dimension(Lx,Ly), intent(inout) :: grid
integer, dimension(2,v), intent(inout) :: s
integer, dimension(2) :: loc

call random_seed (put = seed)

loc = minloc(s)
xmax = loc(2) - 1
print*,"xmax = ",xmax

call random_number(rnd)
posx=int(rnd*xmax)+1
c=0
print*,"posx=",posx, "s(1:2,pox)=",s(:,posx)

grid(s(1,posx),s(2,posx))=1

if (s(1,posx)==1) then

    ccp=Lx
else
    ccp=s(1,posx)-1
end if

if (grid(ccp,s(2,posx))==0) then

do i=1,xmax
    if ((s(1,i)/=ccp) .and. (s(2,i)/=s(2,posx))) then
        s(1,posx)=ccp
        s(2,posx)=s(2,posx)
        c=1
    end if
end do

```

```

        end if
    end do
end if

if (s(1,posx)==Lx) then
    ccp=1
else
    ccp=s(1,posx)+1
end if

if (grid(ccp,s(2,posx))==0) then

    do i=1,xmax
        if ((s(1,i)/=ccp) .and. (s(2,i)/=s(2,posx))) then
            if (c==0) then
                s(1,posx)=ccp
                s(2,posx)=s(2,posx)
                c=1
            else
                s(1,xmax+1)=ccp
                s(2,xmax+1)=s(2,posx)
                xmax=xmax+1
            end if
        end if
    end do
end if

if (s(2,posx)==Ly) then
    ccp=1
else
    ccp=s(2,posx)+1
end if

if (grid(s(1,posx),ccp)==0) then
    print*,"s(1,posx)=",s(1,posx)
    print*,"ccp=",ccp
    do i=1,xmax
        if ((s(1,i)/=s(1,posx)) .and. (s(2,i)/=ccp)) then
            print*,"si"
            if (c==0) then
                s(1,posx)=s(1,posx)
                s(2,posx)=ccp
                c=1
            else
                s(1,xmax+1)=s(1,posx)
                s(2,xmax+1)=ccp
            end if
        end if
    end do
end if

```

```

                xmax=xmax+1
            end if
        end if
    end do
end if

! do i=1,2
! print*(s(i,j), j=1,v)
! end do
! do i=1,Ly
! print*(grid(j,i), j=1,Lx)
! end do

if (grid(s(1,posx),s(2,posx)-1)==0) then
print*,"s4"
do i=1,xmax
    if ((s(1,i)/=s(1,posx)) .and. (s(2,i)/=s(2,posx)-1)) then
        if (c==0) then
            s(1,posx)=s(1,i)
            s(2,posx)=s(2,i)-1
            c=1
        else
            s(1,xmax+1)=s(1,i)
            s(2,xmax+1)=s(2,i)-1
            xmax=xmax+1
        end if
    end if
end do
end if

end subroutine edengen

end module common

program eden

use common
implicit none
integer::i,v,j
integer, dimension(:,:), allocatable :: grid
real, dimension (:), allocatable :: w, hmed
integer, dimension(:,:), allocatable :: s

open (unit=1, file="eden1.dat", status= "replace", action="write")

call load ()

```

```

v=lx*ly

allocate(grid(Lx,Ly))
allocate(w(nmcs))
allocate(hmed(nmcs))
allocate(s(2,v))

call init(grid,Lx,Ly, s, v)
print*,"v=lx*ly=",v
print*,"nmcs=",nmcs
do i=1, nmcs
!   print*," imcs=",i,"   grid:"
!   do j=ly,1,-1
!       print*,grid(1:lx,j)
!   end do
!       call edengen(grid,Lx,Ly,v,s)
end do

write(unit=1,fmt=*) s

close(unit=1)

deallocate(grid,w,hmed,s)

end program eden

```