## Exercises Lecture IX <br> Ising Model

(results for items in red have to be included in the homework report)

## 1. Ising Model on a square lattice

Write a code for a 2 D Ising model on a square lattice in equilibrium with a thermal bath, without external magnetic field, using the spin flip dynamics (considered as an actual temporal evolution process), and periodic boundary conditions (PBC). See for instance the code ising.f90.
A useful reference paper is D.P. Landau, Phys. Rev. B 13, 2997 (1976).
Input parameters are:

- L (linear lattice dimension, which gives the number of spins: $\mathrm{N}=\mathrm{L}^{*} \mathrm{~L}$ ),
- nmcs (number of total MC steps per spin)
- nequil (number of equilibration MC steps per spin)
- T (temperature of the thermal bath).

Quantities of interest are: the magnetization per spin:

$$
\frac{M}{N}=\frac{1}{N} \sum_{i=1}^{N} s_{i}
$$

the energy per spin, with $\langle i, j\rangle$ all over the nearest neighbor pairs:

$$
\frac{E}{N}=-\frac{J}{N} \sum_{<i, j>} s_{i} s_{j}
$$

and quantities related to them, such as time (ensemble) averages, that we denote with $\rangle$, and response functions, i.e., the heat capacity per spin:

$$
c=\left(<E^{2}>-<E>^{2}\right) / k_{B} T^{2} N
$$

and the magnetic susceptibility per spin, in absence of an external magnetic field:

$$
\chi=\left(<M^{2}>-<M>^{2}\right) / k_{B} T N
$$

Consider units such that $k_{B}=1, J=1$.
(a) Choose $\mathrm{L}=30, \mathrm{~T}=2$, and initially spin $= \pm 1$ randomly. Calculate and plot the instantaneous values of energy $E / \mathrm{N}$ and magnetization $M / \mathrm{N}$ per particle as a function of Metropolis-Monte Carlo steps: how much time (i.e. how many nequil MC steps) is necessary to equilibrate the system? Plot the final snapshot of the spin pattern: does the system appear ordered or disordered? Calculate also $c$ and $\chi$.
(b) Choose $\mathrm{T}=4$ and repeat (a).
(c) For fixed $T$, e.g. for $T=1$ or $T=2$, change the initial condition of magnetization (choose for instance some typical ordered configurations -all spins up, all spins down, alternatively up or down as on a chessboard, all left hand side spins up and all right hand side down, ...). Does the equilibration time change?
(d) Change the temperature $T$ by varying it from 1 to 4 with steps of $\Delta T=0.5$. Consider runs long enough, so that the equilibrium has been reached and enough statistical data are collected. Calculate $\langle E\rangle / \mathrm{N},\langle M\rangle / \mathrm{N}, c$ and $\chi$; plot these quantities as functions of $T$. Can you estimate from the plots the critical temperature (whose value $T_{c}=$ $2.269 \mathrm{~J} / k_{B}$ for 2 D is known in case $\left.L \rightarrow \infty\right)$ ? Calculate numerically $c$ both in terms of energy fluctuations and doing the numerical derivative with respect to the temperature. Compare the results.
(e) Repeat now (d) with $\mathrm{L}=4$. Comment the results.
(f) Consider the case with open boundary conditions. (Modify the relevant parts of the code concerning the calculation of the energy.) Repeat some runs with $\mathrm{L}=30$ and $\mathrm{L}=4$. Comment the results.
(g) In ising.f90 the numerical estimate of $E$ and $M$ is implemented by updating E at each $M C$ step over the whole lattice, i.e. after one (on average) trial move for all the spins, chosen randomly one at a time. Choose for instance $\mathrm{L}=30$ and a certain value of T . Can you see any difference if you choose the spins to flip in an ordered sequence?
(h) Instead of updating E after each MC step over all the spins, do it for each configuration, i.e. after each single MC step per spin. Compare some results obtained with the two methods, and discuss whether the two methods are equivalent or not.
(i) (optional) It is interesting also to visualise the variation of the spin pattern during the evolution. It can be done using gnuplot. (Example available on moodle)


```
!c ising.f90
!c
!c Metropolis algorithm to calculate <E>, <M>, in the canonical ensemble
!c (fix T,N,V) with a 2D Ising model
!c
lc Here: K_B = 1
!c J = 1
!c
```



```
module common
implicit none
public :: initial,metropolis,DeltaE
public :: data,output
integer, public, parameter :: double = selected_real_kind(13)
real (kind = double), public :: T,E,M
integer, public, dimension(:,:), allocatable :: spin
real (kind = double), public, dimension(-8:8) :: w
integer, public, dimension(4) :: seed % CHANGE DIMENSION IF NEEDED
integer, public :: N,L,nmcs,nequil
integer, public :: accept
```

```
contains
    subroutine initial(nequil,cum)
        integer, intent (out) :: nequil
    real (kind = double), dimension(5), intent (out) :: cum
    integer :: x,y,up,right,sums,i,dE
    real :: rnd
    print *, "linear dimension of lattice L ="
    read *, L
    allocate(spin(L,L))
    print *, "reduced temperature T ="
    read *, T
    N = L*L
    print *, "# MC steps per spin for equilibrium ="
    read *, nequil
    print *, "# MC steps per spin for averages ="
    read *, nmcs
    print *, "seed (1:4) ="
    read *, seed
    call random_seed(put=seed)
    M = 0.0_double
    ! random initial configuration
    ! compute initial magnetization
    do y = 1,L
        do x = 1,L
            call random_number(rnd)
            if (rnd < 0.5) then
                spin(x,y) = 1
            else
                    spin}(x,y)=-
            end if
            M = M + spin(x,y)
        end do
    end do
    ! compute initial energy
    E = 0.0_double
    do y = 1,L
        ! periodic boundary conditions
        if (y == L) then
            up = 1
        else
            up = y + 1
        end if
        do x = 1,L
            if (x == L) then
                right = 1
```

```
    else
            right = x + 1
        end if
    sums = spin(x,up) + spin(right,y)
calculate the initial energy summing all over pairs
! (for a given spin, consider only the up NN and the right NN
! - NOT the down and the left NN - : each interaction is counted once
            E = E - spin(x,y)*sums
        end do
    end do
!
calculate the transition probability according
to the Boltzmann distribution (exp(-deltaE/KT).
! Choosing the interaction parameter J=1, ***ONLY IN CASE OF P.B.C.***
! possible energy variations per spin flip are -8,-4,0,+4,+8:
    do dE = -8,8,4
        w(dE) = exp (-dE/T)
    end do
    accept = 0
    cum = 0.0_double
end subroutine initial
subroutine metropolis()
    ! one Monte Carlo step per spin
    integer :: ispin, x,y,dE
    real :: rnd
    do ispin = 1,N
            ! random x and y coordinates for trial spin
            call random_number(rnd)
            x = int(L*rnd) + 1
            call random_number(rnd)
            y = int(L*rnd) + 1
            dE = DeltaE(x,y)
            call random_number(rnd)
            if (rnd <= w(dE)) then
                    spin(x,y) = - spin(x,y)
                    accept = accept + 1
                    M = M + 2*spin(x,y) ! factor 2 is to account for the variation:
                    E = E + dE ! (-(-)+(+))
            end if
        end do
end subroutine metropolis
function DeltaE(x,y) result (DeltaE_result)
    ! periodic boundary conditions
    integer, intent (in) :: x,y
```

```
    integer :: DeltaE_result
    integer :: left
    integer :: right
    integer :: up
    integer :: down
    if (x == 1) then
        left = spin(L,y)
        right = spin(2,y)
    else if ( }\textrm{x}===L\mathrm{ ) then
        left = spin(L-1,y)
        right = spin(1,y)
    else
        left = spin( }\textrm{x}-1,\textrm{y}
        right = spin(x+1,y)
    end if
    if (y == 1) then
        up = spin(x,2)
        down = spin(x,L)
    else if (y == L) then
        up = spin(x,1)
        down = spin(x,L-1)
    else
        up = spin(x,y+1)
        down = spin(x,y-1)
    end if
    DeltaE_result = 2*spin(x,y)*(left + right + up + down)
! also here the factor 2 is to account for the variation
end function DeltaE
subroutine data(cum)
    ! accumulate data after every Monte Carlo step per spin
    real (kind = double), dimension(5), intent (inout) :: cum
    cum(1) = cum(1) + E
    cum(2) = cum(2) + E*E
    cum(3) = cum(3) + M
    cum(4) = cum(4) + M*M
    cum(5) = cum(5) + abs(M)
end subroutine data
subroutine output(cum)
    real (kind = double), dimension(5), intent (inout) :: cum
    real (kind = double) :: eave,e2ave,mave,m2ave,abs_mave
    real :: acceptance_prob
    acceptance_prob = real(accept)/real(N)/real(nmcs+nequil)
    eave = cum(1)/real(N)/real(nmcs)
    e2ave = cum(2)/real(N*N)/real(nmcs)
```

```
    mave = cum(3)/real(N)/real(nmcs)
    m2ave = cum(4)/real(N*N)/real(nmcs)
    abs_mave = cum(5)/real(N)/real(nmcs)
    print *, "temperature =", T
    print *, "acceptance probability =", acceptance_prob
    print *, "mean energy per spin =", eave
    print *, "mean squared energy per spin =", e2ave
    print *, "mean magnetization per spin =", mave
    print *, "mean squared magnetization per spin =", m2ave
    print *, "mean |magnetization| per spin =", abs_mave
    end subroutine output
end module common
program ising
    ! metropolis algorithm for the ising model on a square lattice
    use common
    integer :: imcs,ispin,jspin
    real (kind = double), dimension(5) :: cum
    call initial(nequil,cum)
    ! equilibrate system
    do imcs = 1,nequil
        call metropolis()
    end do
    ! accumulate data while updating spins
    do imcs = 1,nmcs
        call metropolis()
        call data(cum)
    end do
    call output(cum)
! write the coordinates of spins up and down on files for plotting
    open(unit=8,file='ising-up.dat',status='replace')
    open(unit=9,file='ising-down.dat',status='replace')
    do jspin = 1,L
        do ispin = 1,L
            if(spin(ispin,jspin)==1)write(8,*)ispin,jspin
            if(spin(ispin,jspin)==-1)write(9,*)ispin,jspin
        end do
    end do
    close(8)
    close(9)
    deallocate(spin)
end program ising
```

