2D and 3D Ising model using Monte Carlo and Metropolis method

Syed Ali Raza

May 2012

1 Introduction

We will try to simulate a 2D Ising model with variable lattice side and then extend it to a 3 dimensional lattice. We would calculate the average magnitude of the magnetization, and then also try to simulate how the magnetization changes with temperature. We will try to locate a critical temperature at which the lattice undergoes a phase change and becomes disorderd from ordered at high temperatures. We would compare this value of the critical value temperature with the theoretical values.

I will discuss the algorithms and the techniques. I have used two methods here; both the metropolis and the brute force iteration method. We would discuss both of them.

2 Ferromagnetism

The macroscopic magnetism properties observed in a material are due to the alignment of the quantum spins of the particles that make up the magnet. These quantum spins act like small bar magnets, however due to exchange coupling as discussed in the Ising model section. The spins in a ferromagnetic material energetically prefer to line up, rather than anti-align. If all the spins in an entire sample aligned, however, this would result in a macroscopically high magnetostatic energy state.

In order to achieve a ground state (lowest energy) then, the sample divides up into many areas, called domains. Within each domain, the quantum spins line up, however each domain has its magnetic field pointing in a different direction, which results in a lower macroscopic energy state. However the formation of the domain walls increases the system energy, so the balance between the quantum spins wanting to line up, the macroscopic field netting to zero and the energy contribution from the formation of domain walls means that the domain sizes vary in order to provide the overall lowest energy state.

Furthermore, the system experiences random flipping of spins due to thermal excitations. The higher the temperature, the more frequently the spins flip, resulting in loss of domain formation and a revertion to a completely random state. On the other end of the scale, as the temperature drops, the domains become larger and eventually, below a critical temperature, all the spins will align and the system experiences a disorder to order phase transition.

3 Ising Model

The Ising model is a mathematical model of ferromagnetism in statistical mechanics. It consists of spins placed on a lattice, these spin can only be in two states (up +1 or down -1) states. Each of the spin couples and interacts with its nearest neighbors.

We can write the ising model energy as a simple equation.

$$H(\sigma) = -\sum_{ij} J_{ij}\sigma_i\sigma_j \tag{1}$$

Here ij represent the interaction due to the nearest particles.

4 Boundary Conditions

We would be doing our simulations for finite lattices but with boundary conditions. For the 2D case the left edge equals the right edge, and the top edge equals the lower edge. In a sense they are connected and have a geometry of a torus. In my algorithm for assigning such boundary conditions I always break them into three parts. One for the middle of the lattice, second for the corners and third for the edges.

5 2D Ising Model

I have written my code for a generic lattice size, you can plug in and vary the size of the lattice at will. The first code given in appendix 1 considers the lattice at zero temperature. We can easily visualise how the spin configuration flips and how the corresponding energy matrix also changes with every time step. In every iteration a spin considers it's energy due to interaction with it's neighbours, we then flip the spin and calculate the new energy. If the flip is more favourable (Energy is lower than the previous energy) then we keep the flip, otherwise it flips back.

The lowest energy state occurs when all spins are aligned in a single direction, either all up or all down. This is of course at zero temperature when there is no thermal fluctuation. We can actually vary the size of the lattices and see how it affects the time it takes for the spin configuration to realise it's lowest

energy state.

I then added the temperature dependence too, so if the flip energy is not favorable it is still possible the spin flips and that depends on the energy difference between the two states and the temperature of the lattice. It is denoted by the factor $e^{\beta\Delta E}$ where $\beta = k_b T$. We can see that as we increased the temperature it took longer for the lattice to become alligned because of thermal fluctuation. And after a certain critical temperature the lattice becomes demagnetised and the thermal energy overcomes the aligning energy.

6 Monte Carlo Method

To determine the magnetisation of our sample, we would need to average over all the possible states of the system, weighted by the probability of each state. However, with even a very small lattice, this becomes a very large computation, so we need a more efficient method.

Using the Monte Carlo method, we choose a few random states and average the values. And to be fair in chosing , we choose our random states using a weighting, w(x) that I will discuss in the next section.

7 Metropolis Algorithm

We chose random states with a given distribution and assign a weighting to them, this weighting depends on the energy difference between the flipped and unflipped and on temperature.This is how the algo works. We start with a random state, we calculate a trial energy by flipping it. If it's trial energy is less then we accept the trial. If the trial energy is greater there is stilla probability that we can accept it depending on the $e^{\beta\Delta E}$ factor. Basically we compare thermal fluctuations with some random number and decide if we want to keep the trial or not. As we increase the temperature the probability of accepting the trial energy configuration keeps on increasing and that is why it dominates.

An issue with this is that you need to have a sufficient sampling of the lattice. In my program I iterate it $NX * NY$ so it covers almost all points on the lattice on average. This process is called a single sweep. However a single sweep is not enough and to get a better result statistically I sweep many times. Typically I swept a thousand times for small lattices and a hundred times on large lattices to save computing time.

8 Energy change

To measure the energy change we simply calculated interaction with the neighboring spins. I considered +1 for spin up and -1 for spin down. Below are the

energy terms for both 2D and 3D lattices. Do remember that we have periodic conditions.

$$E = -\sigma_{i,j} \ (\sigma_{i+1,j} + \sigma_{i-1,j} + \sigma_{i,j+1} + \sigma_{i,j-1}) \tag{2}$$

For 3 dimensional case

$$E = -\sigma_{i,j,k} \ (\sigma_{i+1,j,k} + \sigma_{i-1,j,k} + \sigma_{i,j+1,k} + \sigma_{i,j-1,k} + \sigma_{i,j,k-1} + \sigma_{i,j,k+1}) \tag{3}$$

9 Magnetization

We want to see how the spins align and there ground state energy are dependent on temperature. A quantitative measure of how the overall spin configuration changes is Magnetization. It is basically a average of all the spins in the lattice. It is given by the simple formula, Where N is the total number of spins and S_i denotes the spin; either +1 or -1.

$$M = \frac{1}{N} \ \Sigma_{i=0}^{N-1} S_i \tag{4}$$

In our algorithm we calculate the average magnetization $< M >$ averaged over hundreds of sweeps. Below are some of the graphs of Average magnetization plotted against Temperature.

10 plots

Figure 1 we see a sharp drop in magnetization, we call this the critical temperature. As you can see from the graph that the critical temperature is around $T = 2$. In the units wher we have taken $k_b = 1$, this simulated value agrees very nicely with the theoretical critical temperature value of $T = 1.8$.

10.1 effect of varying the lattice size on critical temperature

Strangely if I take small time steps than we don't see a sharp drop in magnetization over a short range of temperature, however if we repeat the same at smaller time steps you get a drop over a larger temperature range. I couldn't explain the reason behind this observation. Another observation is that as you increase the lattice size the drop in magnetization becomes relatively sharper. This is illustrated in figures 2,3 and 4 below where. (Note: Average magnetization is taken over small steps so no drastic drop.)

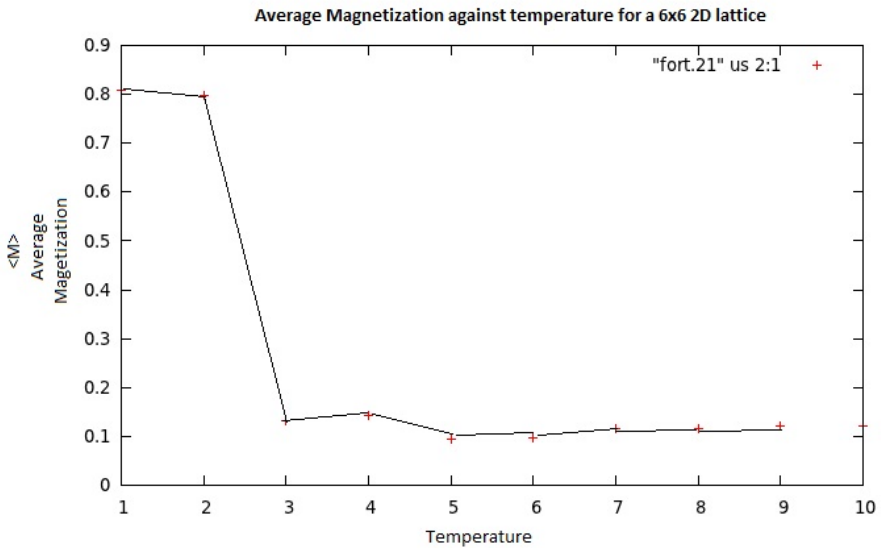

Figure 1:

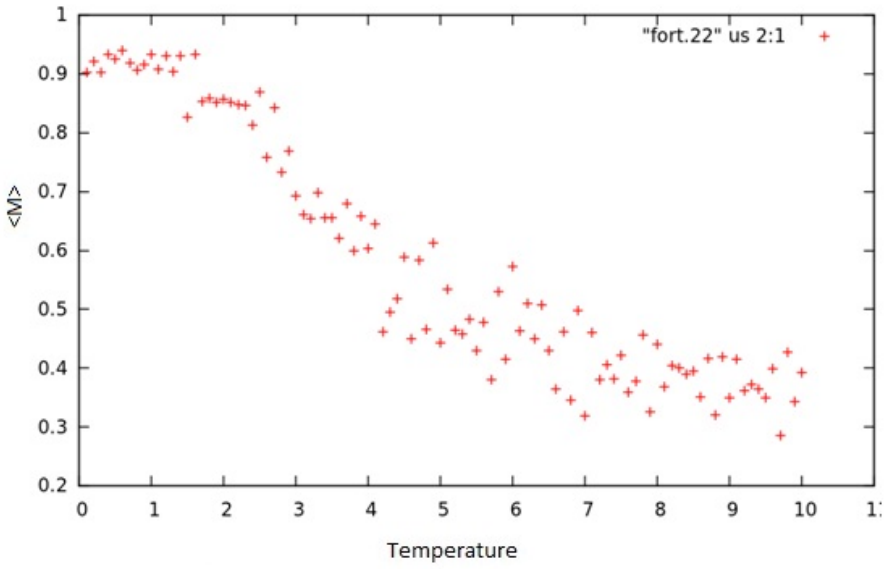

Figure 2: 4x4 2D lattice

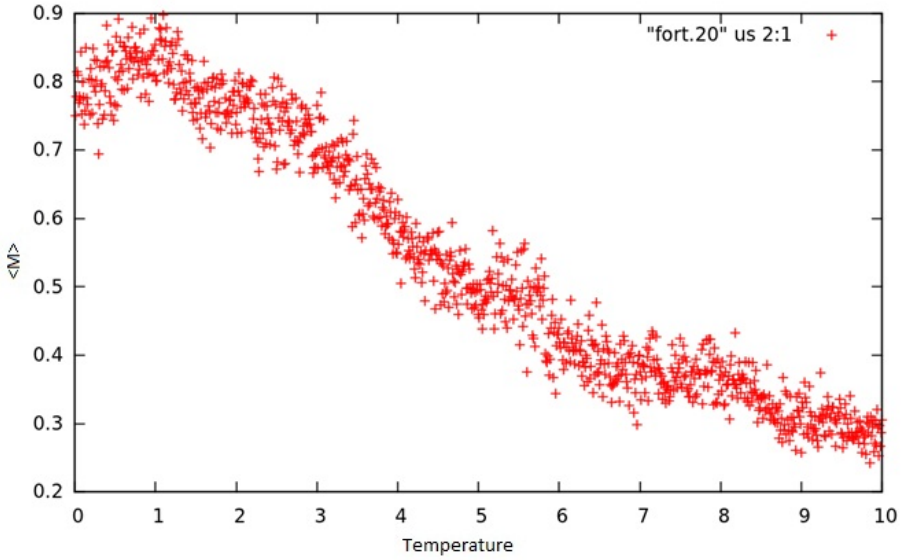

Figure 3: 6x6 2D lattice

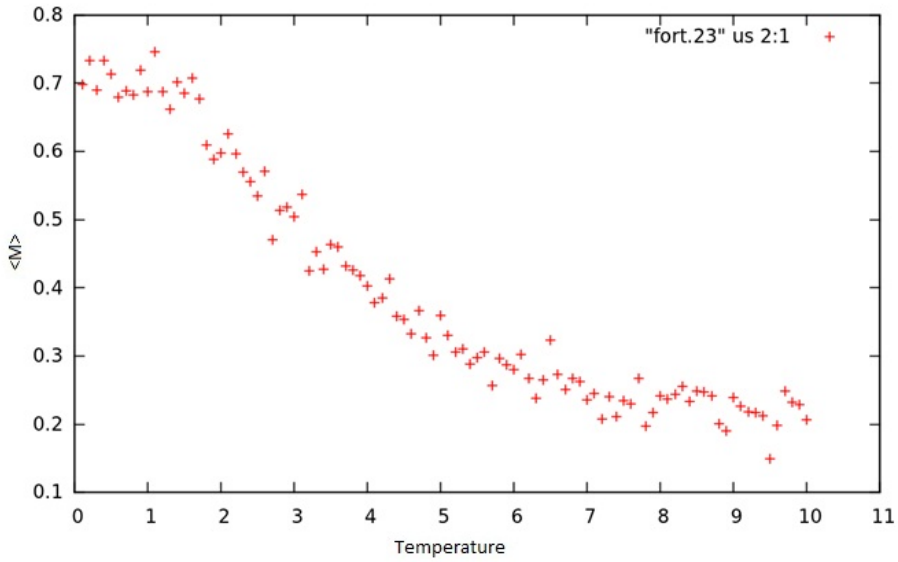

Figure 4: 20x20 2D lattice

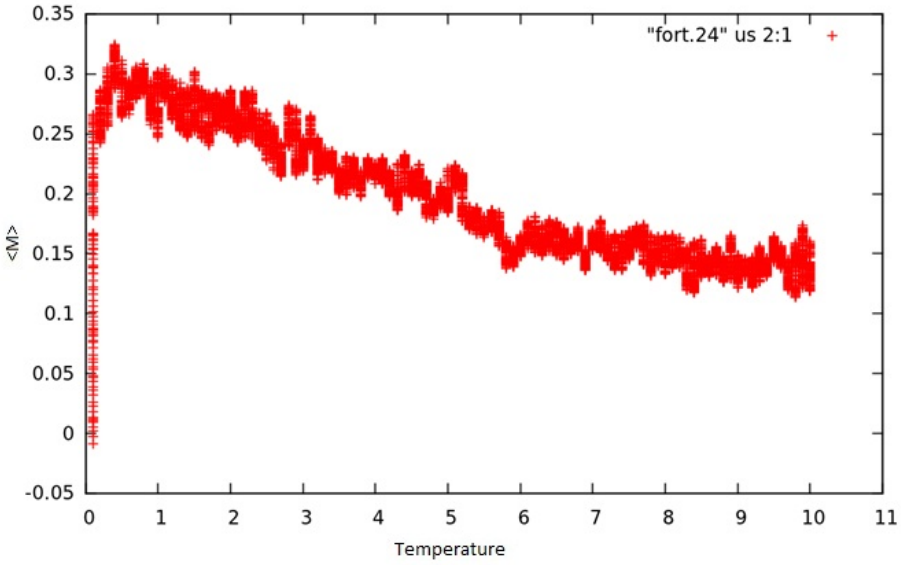

Figure 5: 5x5x5 3D lattice

11 3D Ising Model

Once you have a 2D ising model you can extend it to the 3D ising model. Though the changes are straight forward but the overall computational power required increases alot.

Below are some of the graphs that you would get with the 3d lattice model.

For the 3D model you will note that the graphs don't start near the 1 magnetization mark, this is because I didn't give them enough time to totally magnetize and had fixed number of iterations. I had to do this to save computational time. But the important thing is that you can clearly see a decrease in magnetization due to an increase in temperature.

The best result is for the 20x20x20 lattice, which has nearly a magnetization of 0.9 (I let it have more number of iterations). You can see a drop in magnetization and the critical temp range is about 7-8 in temperature units ($k_b = 1$).

References

David Sheludko, The 2D Ising Model , 2005

Larrimore, Monte Carlo Simulation of the 2D Ising Model

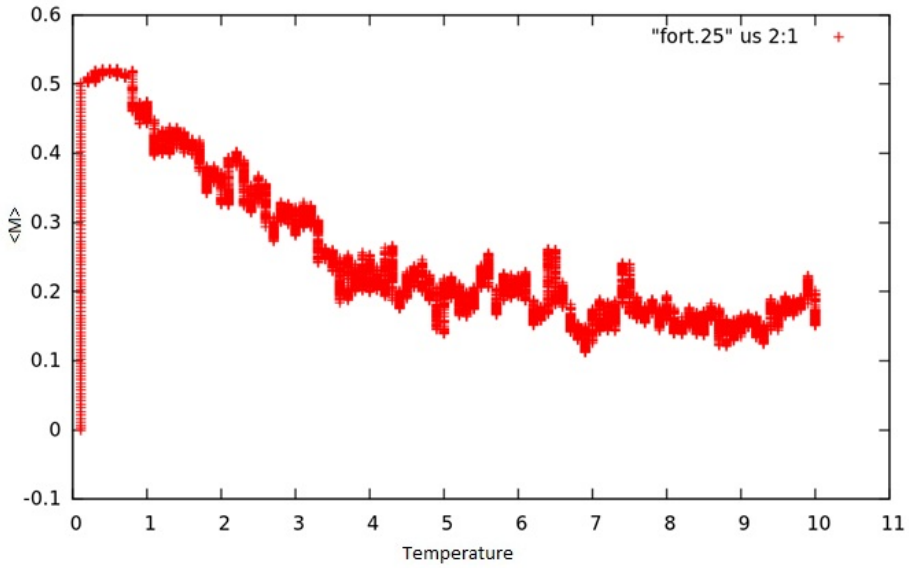

Figure 6: 10x10x10 3D lattice

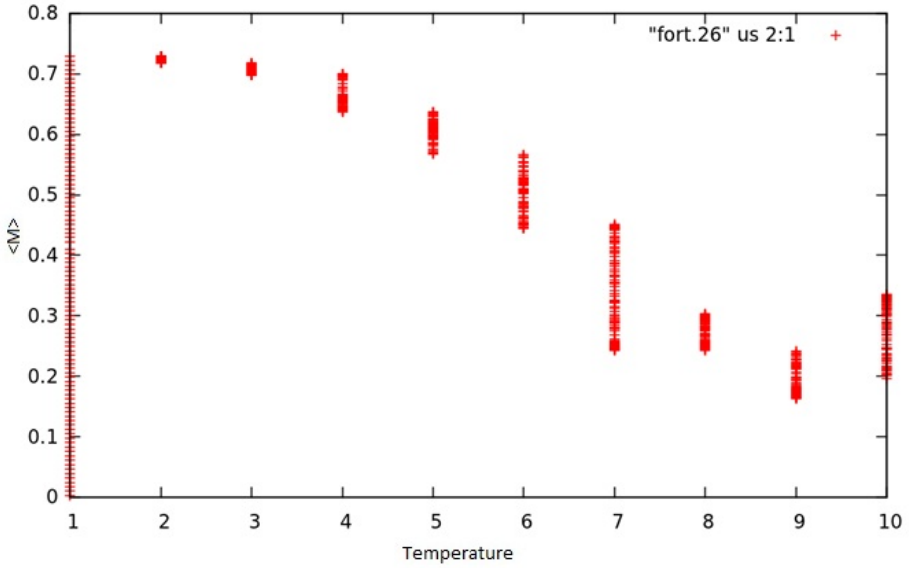

Figure 7: 20x20x20 3D lattice

CODES

2D Ising Model

```
! Working 2D Ising model at any temperature

Program q2
Implicit None

double precision E1, E2 , thermal , magnet(100),magnet2(100), sum2, avg(30), mg2(30), t ,xi(30)
INTEGER ny, nx, nn, i,j, P1, P2, N, xx, k , temp, nnn , sweep , t1


Parameter (nx=6, ny=6, nn=1000 , nnn=100, temp=30)
INTEGER spin(ny,nx),energy1(ny,nx,nn)
real r(5)


t=0.0
Do t1= 1,temp
t= t + 0.33

Do sweep=1,nnn

call initRandSeed()

!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!Create a lattice, assign +1 or -1 spin randomly

Do P1 = 1,nx
Do P2 = 1,ny

call random_number(r)
!print*, r(1)

IF (r(1).lt.0.50) THEN
     spin(P1,P2) = 1
  ELSE
    spin(P1,P2) = -1
  ENDIF

End Do
End Do


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


!!!!!!!!!!!!!!!!!!!!!!!
!calculate energy at each point and store it matrix 'energy'
Do i=2,ny-1
Do j=2,nx-1
energy1(i,j,1) = (-1)*(spin(i,j))*( spin(i+1,j) + spin(i-1,j) + spin(i,j+1) + spin(i,j-1) )
end do
end do
!periodic BCs at corners
energy1(1,1,1) = (-1)*(spin(1,1))*( spin(2,1) + spin(ny,1) + spin(1,2) + spin(1,nx) )
energy1(1,nx,1) = (-1)*(spin(1,nx))*( spin(2,nx) + spin(ny,nx) + spin(1,1) + spin(1,nx-1) )
energy1(ny,1,1) = (-1)*(spin(ny,1))*( spin(1,1) + spin(ny-1,1) + spin(ny,2) + spin(ny,nx) )
energy1(ny,nx,1) = (-1)*(spin(ny,nx))*( spin(1,nx) + spin(ny-1,nx) + spin(ny,1) + spin(ny,nx-1) )
```

```
!periodic BCs at edges
Do i=2,ny-1
energy1(i,1,1) = (-1)*(spin(i,1))*( spin(i+1,1) + spin(i-1,1) + spin(i,2) + spin(i,nx) )
end do

Do i=2,ny-1
energy1(i,nx,1) = (-1)*(spin(i,nx))*( spin(i+1,nx) + spin(i-1,nx) + spin(i,1) + spin(i,nx-1) )
end do

Do j=2,nx-1
energy1(1,j,1) = (-1)*(spin(1,j))*( spin(2,j) + spin(ny,j) + spin(1,j+1) + spin(1,j-1) )
end do

Do j=2,nx-1
energy1(ny,j,1) = (-1)*(spin(ny,j))*( spin(1,j) + spin(ny-1,j) + spin(ny,j+1) + spin(ny,j-1) )
end do



!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!Minimizing energy
Do xx=2,nn

call initRandSeed()
call random_number(r)

!calculate energy at each point and store it matrix 'energy'
Do i=2,ny-1
Do j=2,nx-1

!!!!Flipper
spin(i,j)= spin(i,j)*(-1)
energy1(i,j,xx) = (-1)*(spin(i,j))*( spin(i+1,j) + spin(i-1,j) + spin(i,j+1) + spin(i,j-1) )
thermal = (-energy1(i,j,xx)+energy1(i,j,xx-1) )/t
if (energy1(i,j,xx).gt.energy1(i,j,xx-1).and. exp(thermal).lt.r(4)) then
spin(i,j)= spin(i,j)*(-1)
energy1(i,j,xx)=energy1(i,j,xx-1)
endif
!!!!!!!!

end do
end do

!periodic BCs at corners
spin(1,1)= spin(1,1)*(-1)
energy1(1,1,xx) = (-1)*(spin(1,1))*( spin(2,1) + spin(ny,1) + spin(1,2) + spin(1,nx) )
thermal = (-energy1(1,1,xx)+energy1(1,1,xx-1) )/t
if (energy1(1,1,xx).gt.energy1(1,1,xx-1).and. exp(thermal).lt.r(3)) then
spin(1,1)= spin(1,1)*(-1)
energy1(1,1,xx)=energy1(1,1,xx-1)
endif
!!!!!!!!

spin(1,nx)= spin(1,nx)*(-1)
energy1(1,nx,xx) = (-1)*(spin(1,nx))*( spin(2,nx) + spin(ny,nx) + spin(1,1) + spin(1,nx-1) )
thermal = (-energy1(1,nx,xx)+energy1(1,nx,xx-1) )/t
if (energy1(1,nx,xx).gt.energy1(1,nx,xx-1).and. exp(thermal).lt.r(3)) then
spin(1,nx)= spin(1,nx)*(-1)
energy1(1,nx,xx)=energy1(1,nx,xx-1)
endif
!!!!!!!!
```

```
spin(ny,1)= spin(ny,1)*(-1)
energy1(ny,1,xx) = (-1)*(spin(ny,1))*( spin(1,1) + spin(ny-1,1) + spin(ny,2) + spin(ny,nx) )
thermal = (-energy1(ny,1,xx)+energy1(ny,1,xx-1) )/t
if (energy1(ny,1,xx).gt.energy1(ny,1,xx-1).and. exp(thermal).lt.r(3)) then
spin(ny,1)= spin(ny,1)*(-1)
energy1(ny,1,xx)=energy1(ny,1,xx-1)
endif
!!!!!!!!

spin(ny,nx)= spin(ny,nx)*(-1)
energy1(ny,nx,xx) = (-1)*(spin(ny,nx))*( spin(1,nx) + spin(ny-1,nx) + spin(ny,1) + spin(ny,nx-1) )
thermal = (-energy1(ny,nx,xx)+energy1(ny,nx,xx-1) )/t
if (energy1(ny,nx,xx).gt.energy1(ny,nx,xx-1).and. exp(thermal).lt.r(3)) then
spin(ny,nx)= spin(ny,nx)*(-1)
energy1(ny,nx,xx)=energy1(ny,nx,xx-1)
endif
!!!!!!!!



!periodic BCs at edges
Do i=2,ny-1
spin(i,1)= spin(i,1)*(-1)
energy1(i,1,xx) = (-1)*(spin(i,1))*( spin(i+1,1) + spin(i-1,1) + spin(i,2) + spin(i,nx) )
thermal = (-energy1(i,1,xx)+energy1(i,1,xx-1) )/t
if (energy1(i,1,xx).gt.energy1(i,1,xx-1).and. exp(thermal).lt.r(3)) then
spin(i,1)= spin(i,1)*(-1)
energy1(i,1,xx)=energy1(i,1,xx-1)
endif
!!!!!!!!
end do

Do i=2,ny-1
spin(i,nx)= spin(i,nx)*(-1)
energy1(i,nx,xx) = (-1)*(spin(i,nx))*( spin(i+1,nx) + spin(i-1,nx) + spin(i,1) + spin(i,nx-1) )
thermal = (-energy1(i,nx,xx)+energy1(i,nx,xx-1) )/t
if (energy1(i,nx,xx).gt.energy1(i,nx,xx-1).and. exp(thermal).lt.r(3)) then
spin(i,nx)= spin(i,nx)*(-1)
energy1(i,nx,xx)=energy1(i,nx,xx-1)
endif
!!!!!!!!
end do

Do j=2,nx-1
spin(1,j)= spin(1,j)*(-1)
energy1(1,j,xx) = (-1)*(spin(1,j))*( spin(2,j) + spin(ny,j) + spin(1,j+1) + spin(1,j-1) )
thermal = (-energy1(1,j,xx)+energy1(1,j,xx-1) )/t
if (energy1(1,j,xx).gt.energy1(1,j,xx-1).and. exp(thermal).lt.r(3))then
spin(1,j)= spin(1,j)*(-1)
energy1(1,j,xx)=energy1(1,j,xx-1)
endif
!!!!!!!!
end do

Do j=2,nx-1
spin(ny,j)= spin(ny,j)*(-1)
energy1(ny,j,xx) = (-1)*(spin(ny,j))*( spin(1,j) + spin(ny-1,j) + spin(ny,j+1) + spin(ny,j-1) )
thermal = (-energy1(ny,j,xx)+energy1(ny,j,xx-1) )/t
if (energy1(ny,j,xx).gt.energy1(ny,j,xx-1).and. exp(thermal).lt.r(3)) then
spin(ny,j)= spin(ny,j)*(-1)
energy1(ny,j,xx)=energy1(ny,j,xx-1)
endif
!!!!!!!!
end do
```

```
!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!
!calculate energy at each point and store it matrix 'energy'
Do i=2,ny-1
Do j=2,nx-1
energy1(i,j,xx) = (-1)*(spin(i,j))*( spin(i+1,j) + spin(i-1,j) + spin(i,j+1) + spin(i,j-1) )
end do
end do
!periodic BCs at corners
energy1(1,1,xx) = (-1)*(spin(1,1))*( spin(2,1) + spin(ny,1) + spin(1,2) + spin(1,nx) )
energy1(1,nx,xx) = (-1)*(spin(1,nx))*( spin(2,nx) + spin(ny,nx) + spin(1,1) + spin(1,nx-1) )
energy1(ny,1,xx) = (-1)*(spin(ny,1))*( spin(1,1) + spin(ny-1,1) + spin(ny,2) + spin(ny,nx) )
energy1(ny,nx,xx) = (-1)*(spin(ny,nx))*( spin(1,nx) + spin(ny-1,nx) + spin(ny,1) + spin(ny,nx-1) )
!periodic BCs at edges
Do i=2,ny-1
energy1(i,1,xx) = (-1)*(spin(i,1))*( spin(i+1,1) + spin(i-1,1) + spin(i,2) + spin(i,nx) )
end do

Do i=2,ny-1
energy1(i,nx,xx) = (-1)*(spin(i,nx))*( spin(i+1,nx) + spin(i-1,nx) + spin(i,1) + spin(i,nx-1) )
end do

Do j=2,nx-1
energy1(1,j,xx) = (-1)*(spin(1,j))*( spin(2,j) + spin(ny,j) + spin(1,j+1) + spin(1,j-1) )
end do

Do j=2,nx-1
energy1(ny,j,xx) = (-1)*(spin(ny,j))*( spin(1,j) + spin(ny-1,j) + spin(ny,j+1) + spin(ny,j-1) )
end do



!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!



End do
sum2 = sum(spin)
sum2=abs(sum2)
magnet(sweep)=sum2 / size(spin)
!print*,magnet(sweep)
magnet2(sweep) = magnet(sweep)**2
end do



avg(t1) = sum(magnet)/size(magnet)
mg2(t1) = sum(magnet2)/(size(magnet)**2)
!print*,"Average"
!print*,avg

write (22,*) avg(t1) , t


end do




do j=1,30
xi(j) = (mg2(j) - avg(j)**2)*(1/j)
write (22,*) xi(j) , j
end do
```

End Program

```
!initialize random number with the clock
subroutine initRandSeed()
integer n
integer, dimension(:), allocatable ::  seeder
integer clock
call random_seed(size = n)
ALLOCATE(seeder(n))
call system_clock(count = clock)
seeder = clock + 37 * (/ (i - 1, i = 1, n) /)
call random_seed(put = seeder)
deallocate(seeder)
end subroutine initRandSeed

!return a integer between min and max exlusive
subroutine randInt(min,max,num)
integer min,max,num
real r
call random_number(r)
num = nint(r*(max-min))+min
end subroutine randInt

!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

3D Ising Model

isisng3d.f90

```
! Working 2D Ising model at any temperature

Program q2
Implicit None

double precision E1, E2 , thermal , magnet(100), sum2, avg ,t
INTEGER ny, nx, nz, nn, i,j, P1, P2, P3, N, xx, k , temp, nnn , sweep , t1


Parameter (nx=20, ny=20, nz=20, nn=100 , nnn=100, temp=10)
INTEGER spin(ny,nx,nz),energy1(ny,nx,nz,nn)
real r(5)

t=0.0

Do t1= 1,temp
t = t+ 1

Do sweep=1,nnn

call initRandSeed()

!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!Create a lattice, assign +1 or -1 spin randomly

Do P1 = 1,nx
Do P2 = 1,ny
Do P3 = 1,nz

call random_number(r)
!print*, r(1)

IF (r(1).lt.0.50) THEN
     spin(P1,P2,P3) = 1
  ELSE
    spin(P1,P2,P3) = -1
  ENDIF

End Do
End Do


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


!!!!!!!!!!!!!!!!!!!!!!
!calculate energy at each point and store it matrix 'energy'
Do i=2,ny-1
Do j=2,nx-1
Do k=2,nz-1
energy1(i,j,k,1) = (-1)*(spin(i,j,k))*( spin(i+1,j,k) + spin(i-1,j,k) + spin(i,j+1,k) &
 + spin(i,j-1,k) + spin(i,j,k+1) + spin(i,j,k-1) )
end do
end do
```

```
end do
!periodic BCs at corners
energy1(1,1,1,1) = (-1)*(spin(1,1,1))*( spin(2,1,1) + spin(ny,1,1) &
 + spin(1,2,1) + spin(1,nx,1) + spin(1,1,2) + spin(1,1,nz) )

energy1(1,nx,1,1) = (-1)*spin(1,nx,1)*( spin(2,nx,1) + spin(ny,nx,1) + &
spin(1,1,1) + spin(1,nx-1,1) + spin(1,nx,2) + spin(1,nx,nz) )

energy1(ny,1,1,1) = (-1)* spin(ny,1,1)*( spin(1,1,1) + spin(ny-1,1,1) + spin(ny,2,1) &
 + spin(ny,nx,1) + spin(ny,1,2) + spin(ny,1,nz) )

energy1(ny,nx,1,1) = (-1)*(spin(ny,nx,1))*( spin(1,nx,1) + spin(ny-1,nx,1) &
+ spin(ny,1,1) + spin(ny,nx-1,1) + spin(ny,nx,2) + spin(ny,nx,nz) )

energy1(1,1,nz,1) = (-1)*(spin(1,1,nz))*( spin(2,1,nz) + spin(ny,1,nz) + spin(1,2,nz) &
+ spin(1,nx,nz) + spin(1,1,1) + spin(1,1,nz-1) )

energy1(1,nx,nz,1) = (-1)*(spin(1,nx,nz))*( spin(2,nx,nz) + spin(ny,nx,nz) + spin(1,1,nz) &
 + spin(1,nx-1,nz) + spin(1,nx,1) + spin(1,nx,nz-1) )

energy1(ny,1,nz,1) = (-1)*(spin(ny,1,nz))*( spin(1,1,nz) + spin(ny-1,1,nz) &
+ spin(ny,2,nz) + spin(ny,nx,nz)+ spin(ny,1,1) + spin(ny,1,nz-1)  )

energy1(ny,nx,nz,1) = (-1)*(spin(ny,nx,nz))*( spin(1,nx,nz) + spin(ny-1,nx,nz) + &
spin(ny,1,nz) + spin(ny,nx-1,nz)+ spin(ny,nx,1) + spin(ny,nx,nz-1)  )

!periodic BCs at edges
Do i=2,ny-1
energy1(i,1,1,1) = (-1)*(spin(i,1,1))*( spin(i+1,1,1) + spin(i-1,1,1) &
+ spin(i,2,1) + spin(i,nx,1) + spin(i,1,2) + spin(i,1,nz) )
end do

Do i=2,ny-1
energy1(i,nx,1,1) = (-1)*(spin(i,nx,1))*( spin(i+1,nx,1) + spin(i-1,nx,1) &
+ spin(i,1,1) + spin(i,nx-1,1) + spin(i,nx,2) + spin(i,nx,nz) )
end do

Do i=2,ny-1
energy1(i,nx,nz,1) = (-1)*(spin(i,nx,nz))*( spin(i+1,nx,nz) + spin(i-1,nx,nz) &
+ spin(i,1,nz) + spin(i,nx-1,nz) + spin(i,nx,1) + spin(i,nx,nz-1) )
end do

Do i=2,ny-1
energy1(i,1,nz,1) = (-1)*(spin(i,1,nz))*( spin(i+1,1,nz) + spin(i-1,1,nz) &
+ spin(i,nx,nz) + spin(i,2,nz) + spin(i,1,nz) + spin(i,1,nz-1) )
end do
!!!!!!!!!!!!!!!!!!!!!!!!!
Do j=2,nx-1
energy1(1,j,1,1) = (-1)*(spin(1,j,1))*( spin(2,j,1) + spin(ny,j,1) &
 + spin(1,j+1,1) + spin(1,j-1,1) + spin(1,j,2) + spin(1,j,nz) )
end do

Do j=2,nx-1
energy1(ny,j,1,1) = (-1)*(spin(ny,j,1))*( spin(1,j,1) + spin(ny-1,j,1) &
+ spin(ny,j-1,1) + spin(ny,j+1,1) + spin(ny,j,2) + spin(ny,j,nz) )
end do

Do j=2,nx-1
energy1(1,j,nz,1) = (-1)*(spin(1,j,nz))*( spin(2,j,nz) + spin(ny,j,nz) &
 + spin(1,j+1,nz) + spin(1,j-1,nz) + spin(1,j,1) + spin(1,j,nz-1) )
end do

Do j=2,nx-1
energy1(ny,j,nz,1) = (-1)*(spin(ny,j,nz))*( spin(1,j,nz) + spin(ny-1,j,nz) &
+ spin(ny,j-1,nz) + spin(ny,j+1,nz) + spin(ny,j,1) + spin(ny,j,nz-1) )
```

```
end do
!!!!!!!!!!!!!!!!!!!!!!!!!!!
Do k=2, nz-1
energy1(1,1,k,1) = (-1)*(spin(1,1,k))*( spin(2,1,k) + spin(ny,1,k) &
+ spin(1,2,k) + spin(1,nx,k) + spin(1,1,k+1) + spin(1,1,k-1) )
end do

Do k=2, nz-1
energy1(1,nx,k,1) = (-1)*(spin(1,nx,k))*( spin(2,nx,k) + spin(ny,nx,k) &
+ spin(1,1,k) + spin(1,nx-1,k) + spin(1,nx,k+1) + spin(1,nx,k-1) )
end do

Do k=2, nz-1
energy1(ny,1,k,1) = (-1)*(spin(ny,1,k))*( spin(1,1,k) + spin(ny-1,1,k) &
+ spin(ny,2,k) + spin(ny,nx,k) + spin(ny,1,k+1) + spin(ny,1,k-1) )
end do

Do k=2, nz-1
energy1(ny,nx,k,1) = (-1)*(spin(ny,nx,k))*( spin(1,nx,k) + spin(ny-1,nx,k) &
+ spin(ny,1,k) + spin(ny,nx-1,k) + spin(ny,nx,k+1) + spin(ny,nx,k-1) )
end do
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!Minimizing energy
Do xx=2,nn

call initRandSeed()
call random_number(r)

!calculate energy at each point and store it matrix 'energy'
Do i=2,ny-1
Do j=2,nx-1
Do k=2,nz-1
!!!!Flipper
spin(i,j,k)= spin(i,j,k)*(-1)
energy1(i,j,k,xx) = (-1)*(spin(i,j,k))*( spin(i+1,j,k) + spin(i-1,j,k) &
 + spin(i,j+1,k) + spin(i,j-1,k) + spin(i,j,k+1) + spin(i,j,k-1) )

thermal = (-energy1(i,j,k,xx)+energy1(i,j,k,xx-1) )/t
if (energy1(i,j,k,xx).gt.energy1(i,j,k,xx-1).and. exp(thermal).lt.r(4)) then
spin(i,j,k)= spin(i,j,k)*(-1)
energy1(i,j,k,xx)=energy1(i,j,k,xx-1)
endif
!!!!!!!
end do
end do
end do

!periodic BCs at corners
spin(1,1,1)= spin(1,1,1)*(-1)
energy1(1,1,1,xx) = (-1)*(spin(1,1,1))*( spin(2,1,1) + spin(ny,1,1) &
 + spin(1,2,1) + spin(1,nx,1) + spin(1,1,2) + spin(1,1,nz) )

thermal = (-energy1(1,1,1,xx)+energy1(1,1,1,xx-1) )/t
if (energy1(1,1,1,xx).gt.energy1(1,1,1,xx-1).and. exp(thermal).lt.r(3)) then
spin(1,1,1)= spin(1,1,1)*(-1)
energy1(1,1,1,xx)=energy1(1,1,1,xx-1)
endif
!!!!!!!

spin(1,nx,1)= spin(1,nx,1)*(-1)
energy1(1,nx,1,xx) = (-1)*spin(1,nx,1)*( spin(2,nx,1) + spin(ny,nx,1) + &
spin(1,1,1) + spin(1,nx-1,1) + spin(1,nx,2) + spin(1,nx,nz) )
```

```
thermal = (-energy1(1,nx,1,xx)+energy1(1,nx,1,xx-1) )/t
if (energy1(1,nx,1,xx).gt.energy1(1,nx,1,xx-1).and. exp(thermal).lt.r(3)) then
spin(1,nx,1)= spin(1,nx,1)*(-1)
energy1(1,nx,1,xx)=energy1(1,nx,1,xx-1)
endif
!!!!!!!!

spin(ny,1,1)= spin(ny,1,1)*(-1)
energy1(ny,1,1,xx) = (-1)* spin(ny,1,1)*( spin(1,1,1) + spin(ny-1,1,1) + spin(ny,2,1) &
 + spin(ny,nx,1) + spin(ny,1,2) + spin(ny,1,nz) )

thermal = (-energy1(ny,1,1,xx)+energy1(ny,1,1,xx-1) )/t
if (energy1(ny,1,1,xx).gt.energy1(ny,1,1,xx-1).and. exp(thermal).lt.r(3)) then
spin(ny,1,1)= spin(ny,1,1)*(-1)
energy1(ny,1,1,xx)=energy1(ny,1,1,xx-1)
endif
!!!!!!!!

spin(ny,nx,1)= spin(ny,nx,1)*(-1)
energy1(ny,nx,1,xx) = (-1)*(spin(ny,nx,1))*( spin(1,nx,1) + spin(ny-1,nx,1) &
+ spin(ny,1,1) + spin(ny,nx-1,1) + spin(ny,nx,2) + spin(ny,nx,nz) )

thermal = (-energy1(ny,nx,1,xx)+energy1(ny,nx,1,xx-1) )/t
if (energy1(ny,nx,1,xx).gt.energy1(ny,nx,1,xx-1).and. exp(thermal).lt.r(3)) then
spin(ny,nx,1)= spin(ny,nx,1)*(-1)
energy1(ny,nx,1,xx)=energy1(ny,nx,1,xx-1)
endif
!!!!!!!!

spin(1,1,nz)= spin(1,1,nz)*(-1)
energy1(1,1,nz,xx) = (-1)*(spin(1,1,nz))*( spin(2,1,nz) + spin(ny,1,nz) + spin(1,2,nz) &
+ spin(1,nx,nz) + spin(1,1,1) + spin(1,1,nz-1) )
thermal = (-energy1(1,1,nz,xx)+energy1(1,1,nz,xx-1) )/t
if (energy1(1,1,nz,xx).gt.energy1(1,1,nz,xx-1).and. exp(thermal).lt.r(3)) then
spin(1,1,nz)= spin(1,1,nz)*(-1)
energy1(1,1,nz,xx)=energy1(1,1,nz,xx-1)
endif
!!!!!!!!

spin(1,nx,nz)= spin(1,nx,nz)*(-1)
energy1(1,nx,nz,xx) = (-1)*(spin(1,nx,nz))*( spin(2,nx,nz) + spin(ny,nx,nz) + spin(1,1,nz) &
 + spin(1,nx-1,nz) + spin(1,nx,1) + spin(1,nx,nz-1) )
thermal = (-energy1(1,nx,nz,xx)+energy1(1,nx,nz,xx-1) )/t
if (energy1(1,nx,nz,xx).gt.energy1(1,nx,nz,xx-1).and. exp(thermal).lt.r(3)) then
spin(1,nx,nz)= spin(1,nx,nz)*(-1)
energy1(1,nx,nz,xx)=energy1(1,nx,nz,xx-1)
endif
!!!!!!!!

spin(ny,1,nz)= spin(ny,1,nz)*(-1)
energy1(ny,1,nz,xx) = (-1)*(spin(ny,1,nz))*( spin(1,1,nz) + spin(ny-1,1,nz) &
+ spin(ny,2,nz) + spin(ny,nx,nz)+ spin(ny,1,1) + spin(ny,1,nz-1)  )
thermal = (-energy1(ny,1,nz,xx)+energy1(ny,1,nz,xx-1) )/t
if (energy1(ny,1,nz,xx).gt.energy1(ny,1,nz,xx-1).and. exp(thermal).lt.r(3)) then
spin(ny,1,nz)= spin(ny,1,nz)*(-1)
energy1(ny,1,nz,xx)=energy1(ny,1,nz,xx-1)
endif
!!!!!!!!

spin(ny,nx,nz)= spin(ny,nx,nz)*(-1)
energy1(ny,nx,nz,xx) = (-1)*(spin(ny,nx,nz))*( spin(1,nx,nz) + spin(ny-1,nx,nz) + &
spin(ny,1,nz) + spin(ny,nx-1,nz)+ spin(ny,nx,1) + spin(ny,nx,nz-1)  )
thermal = (-energy1(ny,nx,nz,xx)+energy1(ny,nx,nz,xx-1) )/t
if (energy1(ny,nx,nz,xx).gt.energy1(ny,nx,nz,xx-1).and. exp(thermal).lt.r(3)) then
```

```
spin(ny,nx,nz)= spin(ny,nx,nz)*(-1)
energy1(ny,nx,nz,xx)=energy1(ny,nx,nz,xx-1)
endif
!!!!!!!!



!periodic BCs at edges
Do i=2,ny-1
spin(i,1,1)= spin(i,1,1)*(-1)

energy1(i,1,1,xx) = (-1)*(spin(i,1,1))*( spin(i+1,1,1) + spin(i-1,1,1) &
+ spin(i,2,1) + spin(i,nx,1) + spin(i,1,2) + spin(i,1,nz) )

thermal = (-energy1(i,1,1,xx)+energy1(i,1,1,xx-1) )/t
if (energy1(i,1,1,xx).gt.energy1(i,1,1,xx-1).and. exp(thermal).lt.r(3)) then
spin(i,1,1)= spin(i,1,1)*(-1)
energy1(i,1,1,xx)=energy1(i,1,1,xx-1)
endif
!!!!!!!!
end do

Do i=2,ny-1
spin(i,nx,1)= spin(i,nx,1)*(-1)
energy1(i,nx,1,xx) = (-1)*(spin(i,nx,1))*( spin(i+1,nx,1) + spin(i-1,nx,1) &
+ spin(i,1,1) + spin(i,nx-1,1) + spin(i,nx,2) + spin(i,nx,nz) )

thermal = (-energy1(i,nx,1,xx)+energy1(i,nx,1,xx-1) )/t
if (energy1(i,nx,1,xx).gt.energy1(i,nx,1,xx-1).and. exp(thermal).lt.r(3)) then
spin(i,nx,1)= spin(i,nx,1)*(-1)
energy1(i,nx,1,xx)=energy1(i,nx,1,xx-1)
endif
!!!!!!!!
end do


Do i=2,ny-1
spin(i,nx,nz)= spin(i,nx,nz)*(-1)
energy1(i,nx,nz,xx) = (-1)*(spin(i,nx,nz))*( spin(i+1,nx,nz) + spin(i-1,nx,nz) &
+ spin(i,1,nz) + spin(i,nx-1,nz) + spin(i,nx,1) + spin(i,nx,nz-1) )
thermal = (-energy1(i,nx,nz,xx)+energy1(i,nx,nz,xx-1) )/t
if (energy1(i,nx,nz,xx).gt.energy1(i,nx,nz,xx-1).and. exp(thermal).lt.r(3)) then
spin(i,nx,nz)= spin(i,nx,nz)*(-1)
energy1(i,nx,nz,xx)=energy1(i,nx,nz,xx-1)
endif
!!!!!!!!
end do


Do i=2,ny-1
spin(i,1,nz)= spin(i,1,nz)*(-1)
energy1(i,1,nz,xx) = (-1)*(spin(i,1,nz))*( spin(i+1,1,nz) + spin(i-1,1,nz) &
+ spin(i,nx,nz) + spin(i,2,nz) + spin(i,1,nz) + spin(i,1,nz-1) )
thermal = (-energy1(i,1,nz,xx)+energy1(i,1,nz,xx-1) )/t
if (energy1(i,1,nz,xx).gt.energy1(i,1,nz,xx-1).and. exp(thermal).lt.r(3)) then
spin(i,1,nz)= spin(i,1,nz)*(-1)
energy1(i,1,nz,xx)=energy1(i,1,nz,xx-1)
endif
!!!!!!!!
end do
```

```
Do j=2,nx-1
spin(1,j,1)= spin(1,j,1)*(-1)
energy1(1,j,1,xx) = (-1)*(spin(1,j,1))*( spin(2,j,1) + spin(ny,j,1) &
 + spin(1,j+1,1) + spin(1,j-1,1) + spin(1,j,2) + spin(1,j,nz) )
thermal = (-energy1(1,j,1,xx)+energy1(1,j,1,xx-1) )/t
if (energy1(1,j,1,xx).gt.energy1(1,j,1,xx-1).and. exp(thermal).lt.r(3))then
spin(1,j,1)= spin(1,j,1)*(-1)
energy1(1,j,1,xx)=energy1(1,j,1,xx-1)
endif
!!!!!!!!
end do

Do j=2,nx-1
spin(ny,j,1)= spin(ny,j,1)*(-1)
energy1(ny,j,1,xx) = (-1)*(spin(ny,j,1))*( spin(1,j,1) + spin(ny-1,j,1) &
+ spin(ny,j-1,1) + spin(ny,j+1,1) + spin(ny,j,2) + spin(ny,j,nz) )
thermal = (-energy1(ny,j,1,xx)+energy1(ny,j,1,xx-1) )/t
if (energy1(ny,j,1,xx).gt.energy1(ny,j,1,xx-1).and. exp(thermal).lt.r(3)) then
spin(ny,j,1)= spin(ny,j,1)*(-1)
energy1(ny,j,1,xx)=energy1(ny,j,1,xx-1)
endif
!!!!!!!!
end do


Do k=2, nz-1
spin(ny,1,k)= spin(ny,1,k)*(-1)
energy1(ny,1,k,xx) = (-1)*(spin(ny,1,k))*( spin(1,1,k) + spin(ny-1,1,k) &
+ spin(ny,2,k) + spin(ny,nx,k) + spin(ny,1,k+1) + spin(ny,1,k-1) )
thermal = (-energy1(ny,1,k,xx)+energy1(ny,1,k,xx-1) )/t
if (energy1(ny,1,k,xx).gt.energy1(ny,1,k,xx-1).and. exp(thermal).lt.r(3)) then
spin(ny,1,k)= spin(ny,1,k)*(-1)
energy1(ny,1,k,xx)=energy1(ny,1,k,xx-1)
endif
!!!!!!!!
end do

Do k=2, nz-1
spin(ny,nx,k)= spin(ny,nx,k)*(-1)
energy1(ny,nx,k,xx) = (-1)*(spin(ny,nx,k))*( spin(1,nx,k) + spin(ny-1,nx,k) &
+ spin(ny,1,k) + spin(ny,nx-1,k) + spin(ny,nx,k+1) + spin(ny,nx,k-1) )
thermal = (-energy1(ny,nx,k,xx)+energy1(ny,nx,k,xx-1) )/t
if (energy1(ny,nx,k,xx).gt.energy1(ny,nx,k,xx-1).and. exp(thermal).lt.r(3)) then
spin(ny,nx,k)= spin(ny,nx,k)*(-1)
energy1(ny,nx,k,xx)=energy1(ny,nx,k,xx-1)
endif
!!!!!!!!
end do



!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!
!calculate energy at each point and store it matrix 'energy'

!!!!!!!!!!!!!!!!!!!!!
!calculate energy at each point and store it matrix 'energy'
Do i=2,ny-1
Do j=2,nx-1
Do k=2,nz-1
energy1(i,j,k,xx) = (-1)*(spin(i,j,k))*( spin(i+1,j,k) + spin(i-1,j,k) + spin(i,j+1,k) &
 + spin(i,j-1,k) + spin(i,j,k+1) + spin(i,j,k-1) )
end do
end do
end do
```

```
!periodic BCs at corners
energy1(1,1,1,xx) = (-1)*(spin(1,1,1))*( spin(2,1,1) + spin(ny,1,1) &
 + spin(1,2,1) + spin(1,nx,1) + spin(1,1,2) + spin(1,1,nz) )

energy1(1,nx,1,xx) = (-1)*spin(1,nx,1)*( spin(2,nx,1) + spin(ny,nx,1) + &
spin(1,1,1) + spin(1,nx-1,1) + spin(1,nx,2) + spin(1,nx,nz) )

energy1(ny,1,1,xx) = (-1)* spin(ny,1,1)*( spin(1,1,1) + spin(ny-1,1,1) + spin(ny,2,1) &
 + spin(ny,nx,1) + spin(ny,1,2) + spin(ny,1,nz) )

energy1(ny,nx,1,xx) = (-1)*(spin(ny,nx,1))*( spin(1,nx,1) + spin(ny-1,nx,1) &
+ spin(ny,1,1) + spin(ny,nx-1,1) + spin(ny,nx,2) + spin(ny,nx,nz) )

energy1(1,1,nz,xx) = (-1)*(spin(1,1,nz))*( spin(2,1,nz) + spin(ny,1,nz) + spin(1,2,nz) &
+ spin(1,nx,nz) + spin(1,1,1) + spin(1,1,nz-1) )

energy1(1,nx,nz,xx) = (-1)*(spin(1,nx,nz))*( spin(2,nx,nz) + spin(ny,nx,nz) + spin(1,1,nz) &
 + spin(1,nx-1,nz) + spin(1,nx,1) + spin(1,nx,nz-1) )

energy1(ny,1,nz,xx) = (-1)*(spin(ny,1,nz))*( spin(1,1,nz) + spin(ny-1,1,nz) &
+ spin(ny,2,nz) + spin(ny,nx,nz)+ spin(ny,1,1) + spin(ny,1,nz-1)  )

energy1(ny,nx,nz,xx) = (-1)*(spin(ny,nx,nz))*( spin(1,nx,nz) + spin(ny-1,nx,nz) + &
spin(ny,1,nz) + spin(ny,nx-1,nz)+ spin(ny,nx,1) + spin(ny,nx,nz-1)  )

!periodic BCs at edges
Do i=2,ny-1
energy1(i,1,1,xx) = (-1)*(spin(i,1,1))*( spin(i+1,1,1) + spin(i-1,1,1) &
+ spin(i,2,1) + spin(i,nx,1) + spin(i,1,2) + spin(i,1,nz) )
end do

Do i=2,ny-1
energy1(i,nx,1,xx) = (-1)*(spin(i,nx,1))*( spin(i+1,nx,1) + spin(i-1,nx,1) &
+ spin(i,1,1) + spin(i,nx-1,1) + spin(i,nx,2) + spin(i,nx,nz) )
end do

Do i=2,ny-1
energy1(i,nx,nz,xx) = (-1)*(spin(i,nx,nz))*( spin(i+1,nx,nz) + spin(i-1,nx,nz) &
+ spin(i,1,nz) + spin(i,nx-1,nz) + spin(i,nx,1) + spin(i,nx,nz-1) )
end do

Do i=2,ny-1
energy1(i,1,nz,xx) = (-1)*(spin(i,1,nz))*( spin(i+1,1,nz) + spin(i-1,1,nz) &
+ spin(i,nx,nz) + spin(i,2,nz) + spin(i,1,nz) + spin(i,1,nz-1) )
end do
!!!!!!!!!!!!!!!!!!!!!!!!!!
Do j=2,nx-1
energy1(1,j,1,xx) = (-1)*(spin(1,j,1))*( spin(2,j,1) + spin(ny,j,1) &
 + spin(1,j+1,1) + spin(1,j-1,1) + spin(1,j,2) + spin(1,j,nz) )
end do

Do j=2,nx-1
energy1(ny,j,1,xx) = (-1)*(spin(ny,j,1))*( spin(1,j,1) + spin(ny-1,j,1) &
+ spin(ny,j-1,1) + spin(ny,j+1,1) + spin(ny,j,2) + spin(ny,j,nz) )
end do

Do j=2,nx-1
energy1(1,j,nz,xx) = (-1)*(spin(1,j,nz))*( spin(2,j,nz) + spin(ny,j,nz) &
 + spin(1,j+1,nz) + spin(1,j-1,nz) + spin(1,j,1) + spin(1,j,nz-1) )
end do

Do j=2,nx-1
energy1(ny,j,nz,xx) = (-1)*(spin(ny,j,nz))*( spin(1,j,nz) + spin(ny-1,j,nz) &
+ spin(ny,j-1,nz) + spin(ny,j+1,nz) + spin(ny,j,1) + spin(ny,j,nz-1) )
end do
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Do k=2, nz-1
energy1(1,1,k,xx) = (-1)*(spin(1,1,k))*( spin(2,1,k) + spin(ny,1,k) &
+ spin(1,2,k) + spin(1,nx,k) + spin(1,1,k+1) + spin(1,1,k-1) )
end do

Do k=2, nz-1
energy1(1,nx,k,xx) = (-1)*(spin(1,nx,k))*( spin(2,nx,k) + spin(ny,nx,k) &
+ spin(1,1,k) + spin(1,nx-1,k) + spin(1,nx,k+1) + spin(1,nx,k-1) )
end do

Do k=2, nz-1
energy1(ny,1,k,xx) = (-1)*(spin(ny,1,k))*( spin(1,1,k) + spin(ny-1,1,k) &
+ spin(ny,2,k) + spin(ny,nx,k) + spin(ny,1,k+1) + spin(ny,1,k-1) )
end do

Do k=2, nz-1
energy1(ny,nx,k,xx) = (-1)*(spin(ny,nx,k))*( spin(1,nx,k) + spin(ny-1,nx,k) &
+ spin(ny,1,k) + spin(ny,nx-1,k) + spin(ny,nx,k+1) + spin(ny,nx,k-1) )
end do
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!
!calculate energy at each point and store it matrix 'energy'




!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!


End do
sum2 = sum(spin)
sum2=abs(sum2)
magnet(sweep)=sum2 / size(spin)
!print*,magnet(sweep)

end do


avg = sum(magnet)/size(magnet)
!print*,"Average"
!print*,avg

write (26,*) avg , t

end do
end do

End Program




!initialize random number with the clock
subroutine initRandSeed()
integer n
integer, dimension(:), allocatable ::  seeder
integer clock
call random_seed(size = n)
ALLOCATE(seeder(n))
```

```
call system_clock(count = clock)
seeder = clock + 37 * (/ (i - 1, i = 1, n) /)
call random_seed(put = seeder)
deallocate(seeder)
end subroutine initRandSeed

!return a integer between min and max exlusive
subroutine randInt(min,max,num)
integer min,max,num
real r
call random_number(r)
num = nint(r*(max-min))+min
end subroutine randInt

!!!!!!!!!!!!!!!!!!!!!!!!!!
```