Experiment 9: Monte Carlo Simulations of Phase Transitions

Chong Tian En

March 12, 2020

1 Introduction

1.1 Ising Phase Transition

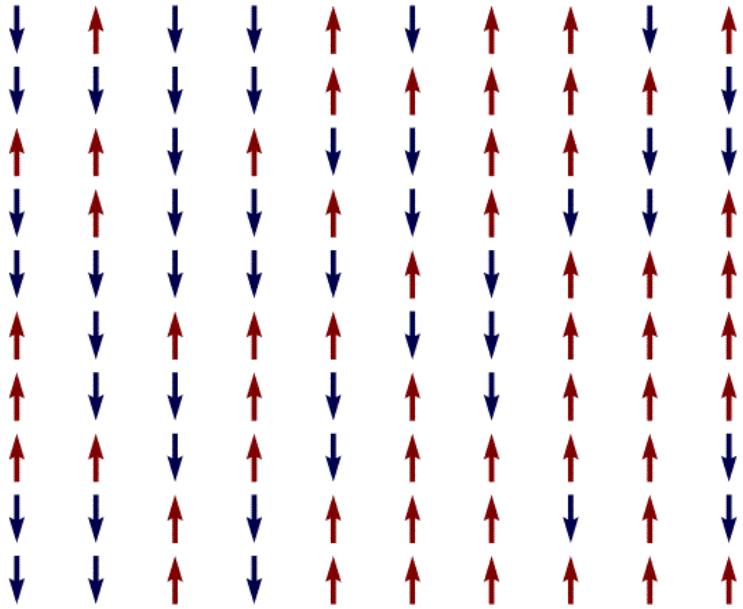

Figure 1: Ising 2D 10×10

Systems in the universe always tend to lower energy states. If any fluctation can cause them to move towards a lower energy state to achieve stability (i.e. higher probability state), the fluctation can happen very easily. Ising random fluctuation, i.e. spin randomly switching between up and down (see Figure 1 and Figure 5 at different temperature T), is governed by the probability of flipping spin, which is then governed by its current energy state and its next energy state. When a spin flip can cause its next energy state to be lower than its current one, there is definitely a flipping. For the same temperature T, if its next energy state is higher (i.e. lower probability) than its current energy state (i.e. higher probability), it would be harder for spin flip to happen spontaneously, as it requires higher temperature to make up for the drop in probability of the next state. Likewise, if the temperature moves from high T to low T, it become less probable for spin flipping, because there is nothing to make up for the difference in probability (see Equation 1.3 and the immediate explanation after it). This is where Ising system is trapped in the low energy state as the probabilities to flip diminishes. This low energy, ordered state, where all spins aligned in the same direction is considered as a ferromagnetic phase (Figure 2). At high T, along the direction $h = 0$ where we always have average magnetization $\langle M \rangle = 0$, i.e. # spin up sites = # spin down sites. The transition between ferromagnetic phase (i.e. ordered phase) to paramagnetic phase (i.e. disordered phase) is 1st order transition due to the discontinuity along $h = 0$ (Figure 3). If we inspect the 2D surface plot (Figure 4) of average magnetisation per site $\langle m \rangle$, we can easily identify various discontinuities on the graph, i.e. below T_c, we travel across different h along different T (See also Figure 2) plus the one we have just mentioned. 1st order phase transition is very obvious, however, 2nd order phase transition is less obvious due to the discontinuity occurring only in its derivatives. i.e. derivatives like heat capacity C_v

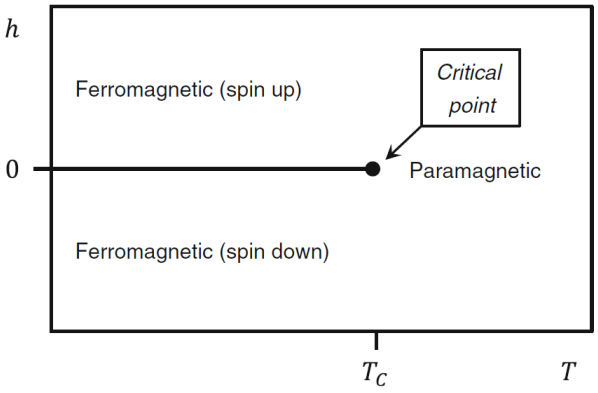

Figure 2: Ising Phase Diagram as A Function of Field h and Temperature T [1]

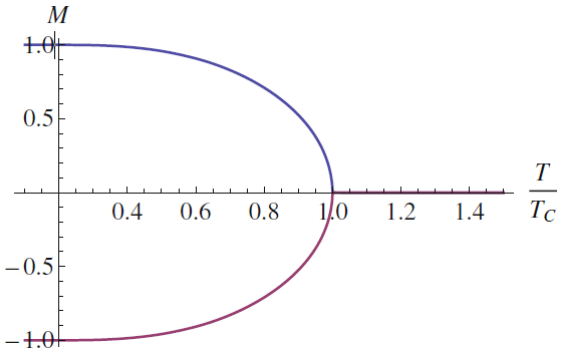

Figure 3: Magnetisation per Site M vs Temperature T [1]

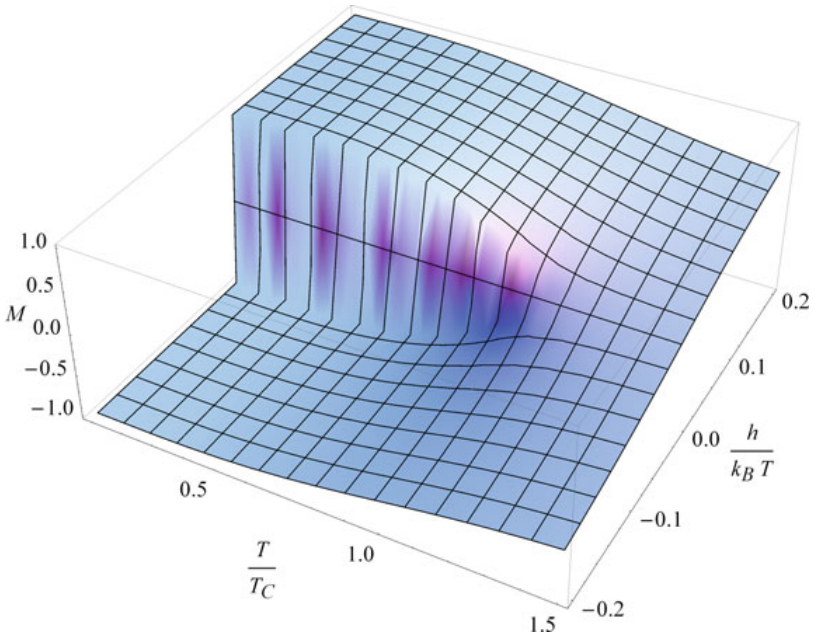

Figure 4: Magnetisation per Site M vs Temperature T and Field h [1]

from average energy $\langle E \rangle$ and susceptibility χ_m from average magnetisation $\langle M \rangle$. In this experiment, we will inspect both 1st order and 2nd order phase transition of Ising model.

Note that although $\langle m \rangle$ denotes average magnetisation and $\langle M \rangle$ denotes average magnetisation per site (in reference to most textbook), from this point onwards m and M swap their roles, in reference to the lab sheet convention.

1.2 Monte Carlo Method [2]

In this experiment, we create 100 independent simulation runs. For each of these run, we generated 10×10 2D Ising grid (Figure 1), and we drop temperature T from 4.0 to 0.1 (Figure 5). So, there are 40 steps for temperature T in total. For each of these step in temperature T, we run 1000 Monte Carlo cycles. Each of these cycle constitute 100 ($L^2 = 10 \times 10$) moves. Each of these moves will constitute randomly selected site fluctuation (it might flip or it might not), governed by probability mechanism as described below:

> Let current state of the system be k and the next state be k', and the corresponding probability for the current state and next state would be $P_k = \frac{1}{Z} e^{-E_k/T}$ and $P_{k'} = \frac{1}{Z} e^{-E_{k'}/T}$ respectively. It also implies we set $k_B = 1$. Now, we generate a random ratio r between $[0, 1]$ to compare the simulated probability ratio of next potential state to the current state $P_{k'}/P_k$. Here, we demand the simulated ratio is to be $> r$ before we do actual spin flipping. Keep in mind that the maximum value for r is 1 but simulated ratio can go over 1. Mathematically, it means:
>
> $$\frac{P_{k'}}{P_k} > r \tag{1.1}$$
>
> $$\frac{Z^{-1} e^{-E_{k'}/T}}{Z^{-1} e^{-E_k/T}} > r \tag{1.2}$$
>
> $$e^{-\frac{\Delta E}{T}} > r \tag{1.3}$$
>
> In our simulation code, we treat $r = 1$ separately because the random $r = [0, 1]$ generation is actually $r = [0, 1)$, i.e. excluding 1. Programmatically and intuitively, it means r is uniform random value between $[0, 1]$. If L.H.S. probability ratio $e^{-\frac{\Delta E}{T}}$ is higher, it can cover more range for r, i.e. the range $[0, e^{-\frac{\Delta E}{T}}]$, and vice versa. If it is more than 1, it has exceeded r's range, so it met our demand all the time. Since the input parameter for us is temperature T, so let's examine T:
>
> For case ΔE being positive:
>
> > As temperature T gets higher, the probability ratio $e^{-\frac{\Delta E}{T}}$ is getting close to 1 but will not exceed 1. Likewise, if T gets lower, the probability ratio gets lower towards 0.
>
> For case ΔE being 0:
>
> > It doesn't matter which T (but still undefined at $T = 0$, as $0/0$ is undefined), the probability ratio $e^{-\frac{\Delta E}{T}}$ is always 1.
>
> For case ΔE being negative:
>
> > It doesn't matter which T, the probability ratio $e^{-\frac{\Delta E}{T}}$ is always > 1.
>
> In the event where our demand is met, we will flip the spin of the site and update the grid. Of course, quantities like energy and magnetisation do change and we need to update accordingly. Although we do update them, we don't collect their data in array until the 100 moves per cycle have been completed, i.e. we only collect at the end of each cycle. With these data, we can proceed to analysis and generate various graphs in this report. Exact program steps are extensively commented in the appendix for understanding.

2 Ising Model in Detail

Each lattice site s_i state is:

$$s_i = +1, -1$$

The hamiltonian (also energy E):

$$H = -J \sum_{ij} s_i s_j - h \sum_i s_i$$

Here, J is positive. So, when a spin is aligned with its neighbours (i.e. $s_i s_j$ being positive), its energy is low and vice versa. Similarly, when a spin is aligned to external field h (i.e. $h s_i$ being positive), its energy is low and vice versa

2.1 Heat Capacity C_v Derivation

Heat Capacity C_v is defined as:

$$C_v = \frac{\partial \langle E \rangle}{\partial T}$$

We know:

$$\beta = \frac{1}{kT} \tag{2.1}$$

Then:

$$-kT^2 d\beta = dT$$

Hence with 2.1:

$$C_v = -\frac{1}{kT^2}\frac{\partial \langle E \rangle}{\partial \beta} \tag{2.2}$$

The partition function:

$$Z = \sum_n e^{-\beta E_n}$$

Partial differentiate with respect to β:

$$\frac{\partial Z}{\partial B} = \sum_n -E_n e^{-\beta E_n}$$

Since $\langle E \rangle$ is defined as:

$$\langle E \rangle = \frac{\sum_n E_n e^{-\beta E_n}}{\sum_n e^{-\beta E_n}}$$

which means:

$$\langle E \rangle = -\frac{1}{Z}\frac{\partial Z}{\partial B}$$

Similarly,

$$\langle E^2 \rangle = \frac{\sum_n E_n^2 e^{-\beta E_n}}{\sum_n e^{-\beta E_n}} = \frac{1}{Z}\frac{\partial^2 Z}{\partial B^2}$$

Then,

$$\langle E^2 \rangle - \langle E \rangle^2 = \frac{1}{Z}\frac{\partial^2 Z}{\partial B^2} - \frac{1}{Z^2}\left(\frac{\partial Z}{\partial B}\right)^2 = \frac{\partial}{\partial \beta}\left(\frac{1}{Z}\frac{\partial Z}{\partial B}\right) = -\frac{\partial}{\partial \beta}\langle E \rangle \tag{2.3}$$

Finally, with 2.2 and 2.3 we have:

$$C_v = \frac{\langle E^2 \rangle - \langle E \rangle^2}{kT^2}$$

2.2 Susceptibility χ_m Derivation

Susceptibility χ_m is defined as:

$$\chi_m = \frac{\partial \langle M \rangle}{\partial h} \tag{2.4}$$

From hamiltonian, let neighbour interaction $X = \sum_{ij} s_i s_j$ and magnetization $M = \sum_i s_i$ Simplified hamiltonian notation:

$$H = -JX - hM$$

Simplified partition function:

$$Z = \sum e^{-\beta(-JX - hM)} = \sum e^{\beta(JX + hM)}$$

Partial differentiate with respect to h:

$$\frac{\partial Z}{\partial h} = \beta \sum M e^{\beta(JX + hM)}$$

Partial differentiate again with respect to h:

$$\frac{\partial^2 Z}{\partial h^2} = \beta^2 \sum M^2 e^{\beta(JX + hM)}$$

Similarly, average magnetization $\langle M \rangle$:

$$\langle M \rangle = \frac{\sum M e^{\beta(JX+hM)}}{\sum e^{\beta(JX+hM)}} = \frac{1}{\beta Z}\frac{\partial Z}{\partial h}$$

And average magnetization squared $\langle M^2 \rangle$:

$$\langle M^2 \rangle = \frac{\sum M^2 e^{\beta(JX+hM)}}{\sum e^{\beta(JX+hM)}} = \frac{1}{\beta^2 Z}\frac{\partial^2 Z}{\partial h^2}$$

Then,

$$\langle M^2 \rangle - \langle M \rangle^2 = \frac{1}{\beta^2 Z}\frac{\partial^2 Z}{\partial h^2} - \left(\frac{1}{\beta Z}\frac{\partial Z}{\partial h}\right)^2 = \frac{\partial}{\partial h}\left(\frac{1}{\beta^2 Z}\frac{\partial Z}{\partial h}\right) = \frac{1}{\beta}\frac{\partial \langle M \rangle}{\partial h}$$

Together with 2.4 and 2.1 we have:

$$\chi_m = \frac{\langle M^2 \rangle - \langle M \rangle^2}{kT}$$

Since we only concern about intensive quantity average magnetization per site $m = M/N$, finally we have:

$$\chi_m = \frac{\langle m^2 \rangle - \langle m \rangle^2}{kT}$$

3 Results

3.1 Question 1

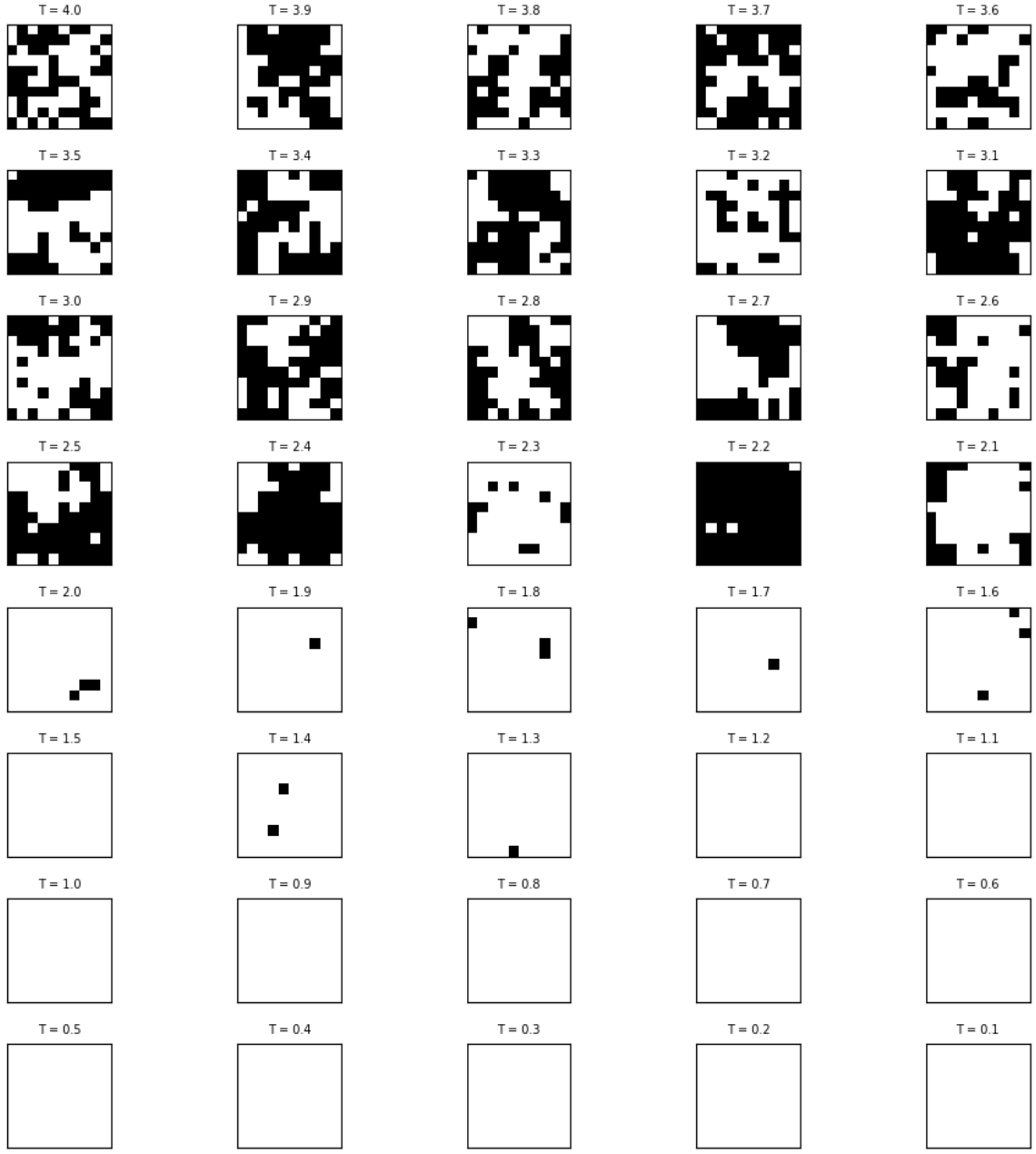

Figure 5: Illustration of the System at Selected Temperatures (white for site with spin up "+1", black for site with spin down "−1")

3.2 Question 2

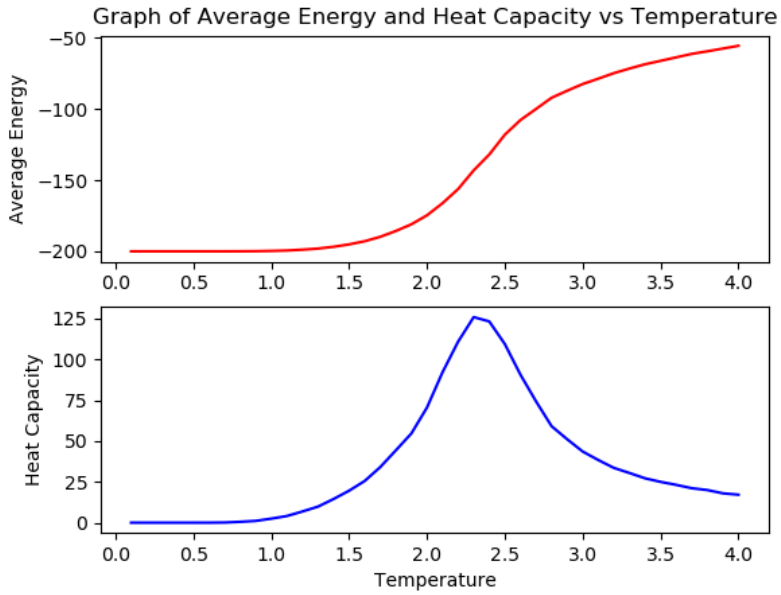

Figure 6: Average Energy $\langle E \rangle$ and the Specific Heat C_v as A Function of the Temperature T

3.3 Question 3

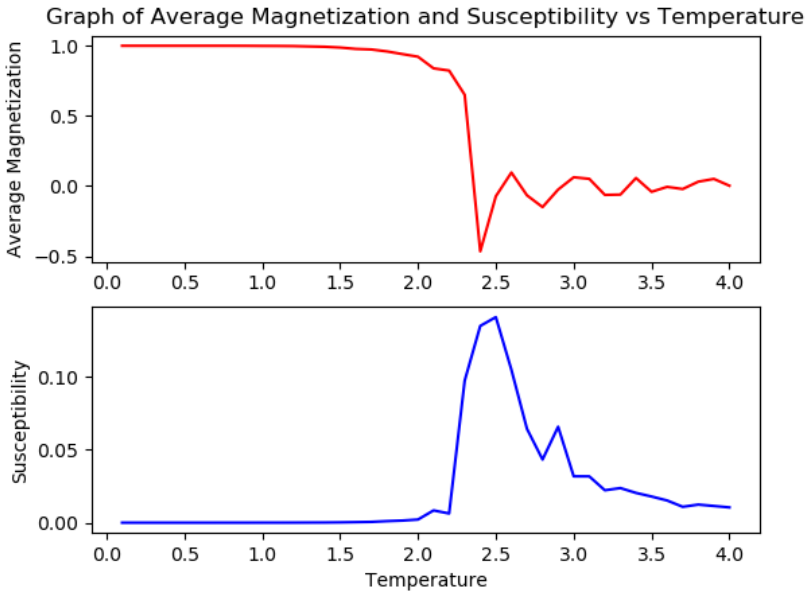

Figure 7: Average Magnetization $\langle m \rangle$ and the Susceptibility χ_m as A Function of the Temperature T

3.4 Question 4

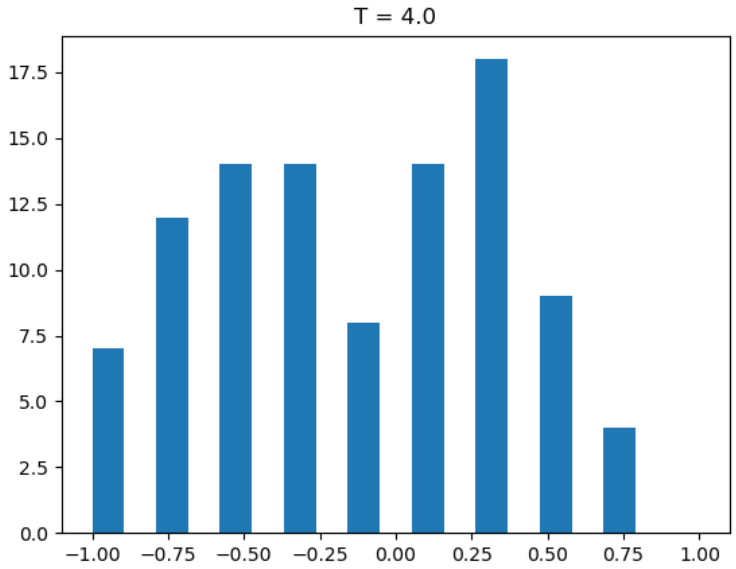

Figure 8: Probability Distribution of Average Magnetisation Per Spin $m_k(T)$ at 100^{th} Simulation at Temperature $T = 4.0$

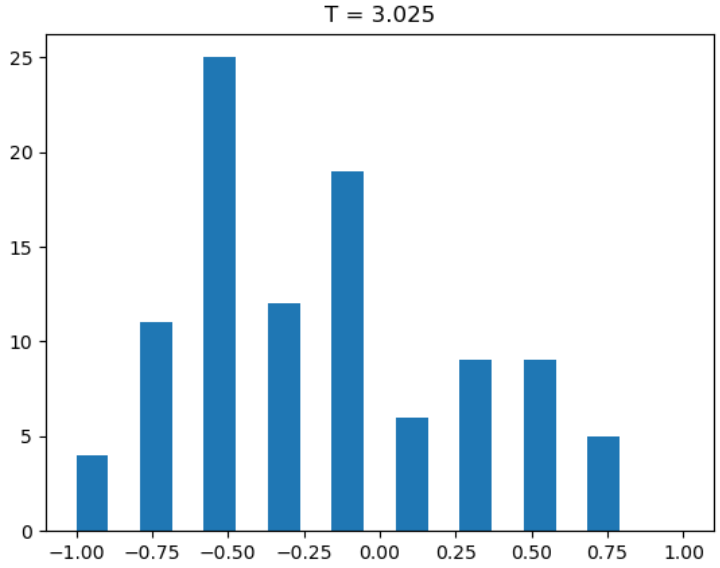

Figure 9: Probability Distribution of Average Magnetisation Per Spin $m_k(T)$ at 100^{th} Simulation at Temperature $T = 3.025$

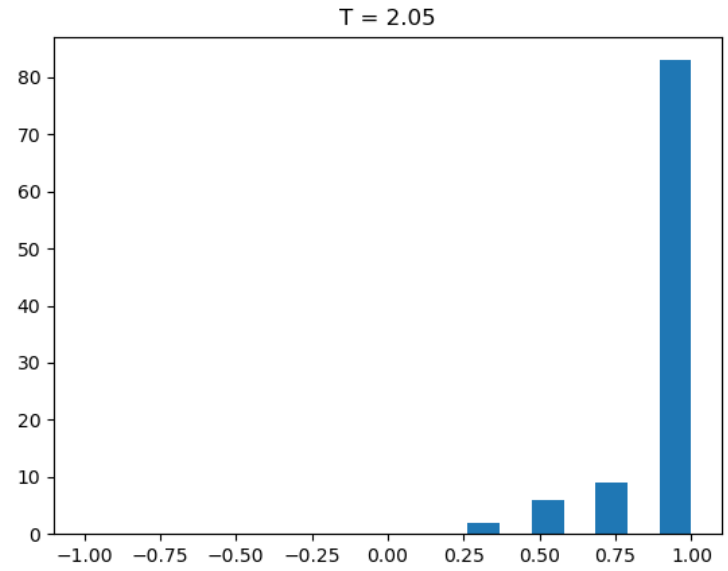

Figure 10: Probability Distribution of Average Magnetisation Per Spin $m_k(T)$ at 100^{th} Simulation at Temperature $T = 2.05$

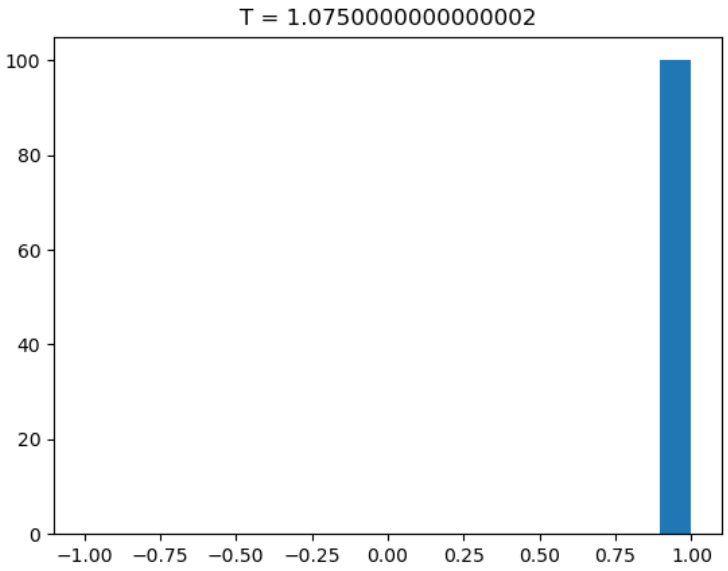

Figure 11: Probability Distribution of Average Magnetisation Per Spin $m_k(T)$ at 100^{th} Simulation at Temperature $T = 1.075$

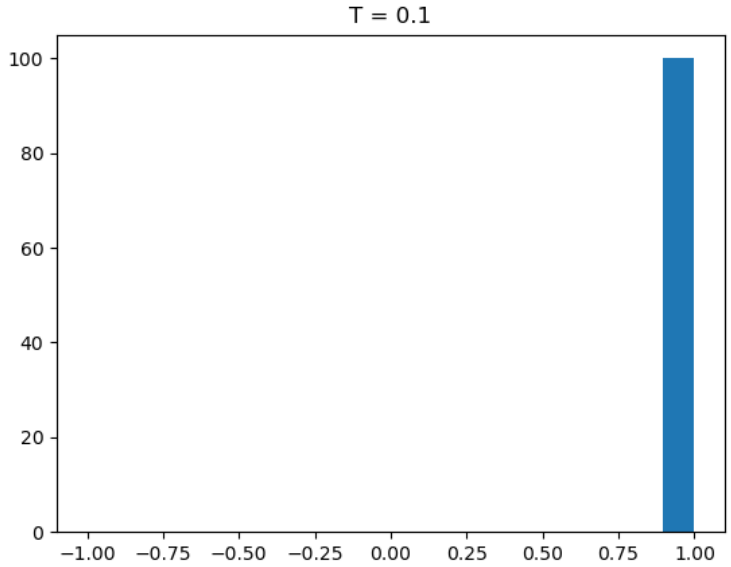

Figure 12: Probability Distribution of Average Magnetisation Per Spin $m_k(T)$ at 100^{th} Simulation at Temperature $T = 0.1$

4 Discussion

4.1 Question 1

From Figure 5, at high T the spins are highly randomized and almost equal in number in up and down. Heavy fluctuation occurs around $T = T_c \approx 2.3$ where mostly down at $T = 2.4$ suddenly flipped to mostly up at $T = 2.3$ then back to mostly down at $T = 2.2$ before finally settling on mostly up at $T = 2.1$. Once it passed T_c, it progressively gets stuck at all spin up as it tends towards $T = 0.1$.

4.2 Question 2

We expect average energy $\langle E \rangle$ to increase as temperature T increases and the 2nd order phase transition happen where the gradient become discontinous, even when the original curve is continous. We do observe that $\langle E \rangle$ increases the most at T_c. Its increase will be more dramatic with even larger simulation matrix (i.e. $\gg 10 \times 10$). Since heat capacity C_v is just $\frac{d\langle E \rangle}{dT}$, we have the gradient of the first graph. At $T = T_c \approx 2.3$, heat capacity C_v diverges, this translate to 2nd order phase transition. In the result, we observe C_v sort of diverges at T_c as the gradient at the peak is not exactly sharp. Even larger simulation matrix can result in a more obvious narrow spike.

4.3 Question 3

First order phase transition not obvious in the simulation due to just 1 simulation. However, average magnetization per site $\langle m \rangle$, closely resembles the ideal case in our introduction. Susceptibility is less obvious not just due to 1 simulation, but also that it is not differentiated with respect to T but with respect to h, where $h = 0$ in our case.

4.4 Question 4

We expect probability distribution to be fairly uniform at high T, but at around T_c, it should be tending towards oneside (left or right) and stays there till lowest T. This is because there is spontaneous symmetry breaking in low temperature, as the average magnetisation per site $\langle m \rangle$ tending towards lowest temperature is either $+1$ or -1 once the system crosses T_c along $h = 0$. When the symmetry is not broken, the system only has one symmetrical lowest point and does not have 2 opposite lowest points, which system can fall onto either one and trap there. Indeed, we observed that there are lots of fluctuation around T_c before our system pick $\langle m \rangle = +1$ to settle on Figure 7, where average magnetisation per spin $\langle m \rangle$ stuck at $+1$. On a separate simulation, it does pick $\langle m \rangle = -1$ too.

5 Conclusion

Monte Carlo method is very useful in simulating Ising model. In real world, Monte Carlo method can simulate systems with random fluctuation, but still obey probabilities that are subjected to certain rules. The real world application includes but not limited to mathematics, science, engineering, business, finance, law, and search and rescue [3].

References

(1) Selinger, J. V., *Introduction to the Theory of Soft Matter Jonathan V. Selinger From Ideal Gases to Liquid Crystals*.

(2) NTULearn Lab Manual for Experiment 9: Monte Carlo Simulations of Phase Transitons.,

(3) Monte Carlo Method. `https://en.wikipedia.org/wiki/Monte_Carlo_method`.

6 Appendix

```
import numpy as np
import copy
from math import exp
from matplotlib import pyplot as plt
from PIL import Image
from random import choice, uniform

# function finding 4 neighboring corners given a lattice site
def neighbourCoord(i,j):
    # left
    if i == 0:  # extreme left
        l = [9, j]
    else:
        l = [i-1, j]
    # right
    if i == 9:  # extreme right
        r = [0, j]
    else:
        r = [i+1, j]
    # up
    if j == 0: # extreme up
        u = [i, 9]
    else:
        u = [i, j-1]
    # down
    if j == 9: # extreme down
        d = [i, 0]
    else:
        d = [i, j+1]

    return l,r,u,d

# function finding all 8 neighbors given a lattice site
def neighbourCoordAll(i,j):
    l,r,u,d = neighbourCoord(i,j)   # use same corners
    # up
    if j == 0:  # extreme up
        ur = [0 if i == 9 else i+1, 9] # extreme & normal right
        ul = [9 if i == 0 else i-1, 9]  # extreme & normal left
    else:       # normal up
        ur = [0 if i == 9 else i+1, j-1] # extreme & normal right
        ul = [9 if i == 0 else i-1, j-1] # extreme & normal left
    # down
    if j == 9:  # extreme down
        dr = [0 if i == 9 else i+1, 0] # extreme & normal right
        dl = [9 if i == 0 else i-1, 0]  # extreme & normal left
    else:  # normal down
```

```
        dr = [0 if i == 9 else i+1, j+1] # extreme & normal right
        dl = [9 if i == 0 else i-1, j+1]  # extreme & normal left

    return [l,r,u,d,ur,dr,dl,ul]

# function multiplying and summing spin with 4 neighboring corners given a lattice site
def neighbourMulSum(i,j,grid):
    l,r,u,d = neighbourCoord(i,j)
    return grid[i][j]*grid[l[0]][l[1]]+grid[i][j]*grid[r[0]][r[1]]+grid[i][j]*grid[u[0]][u[1]]+grid[i][j]*grid[d[0]][d[1]]

# function averaging spin of a local site (9 sites), given a center lattice site
def neighbourAllAvg(i,j,grid):
    neighbours = [l,r,u,d,ur,dr,dl,ul] = neighbourCoordAll(i,j)
    mtot = 0
    for n in neighbours:
        mtot = mtot + grid[n[0]][n[1]]
    return (grid[i][j]+mtot)/9

# function calculating total Hamiltonian of the system
def Htotal(grid):
    H = 0
    for i in range(10):
        for j in range(10):
            H = H + neighbourMulSum(i,j,grid)

    return H*(-1/2)

# function calculating total magnetization of the system
def Mtotal(grid):
    mag = 0
    for i in range(10):
        for j in range(10):
            mag = mag + grid[i][j]

    return mag*(1/100)

# function calculating probability distribution
def probDist(grid):
    mLocalAvgs = []
    Ts = list(np.linspace(4,0.1,5,endpoint=True)) # generate 5 T values between 4 and 0.1
    # go through average spin of local site for all lattice sites
    for i in range(10):
        for j in range(10):
            mLocalAvgs.append(neighbourAllAvg(i,j,grid))
    # plot probability distribution histogram with 20 bins between -1 and 1
    plt.hist(mLocalAvgs,range=(-1,1),bins=np.linspace(-1,1,20,endpoint=True))
    plt.title("T = {}".format(Ts[count]))
    plt.show()

# function rounding off new T values to match the original T values for question 4
def newTs(Ts, Tsq4):
    return [(int(x*10) if (x*10-int(x*10))<0.5 else int(x*10)+1)/10 for x in Tsq4] # explicit rounding workaround (rounding not working properly in Python)

# function generating Latex table data
def genReverseLatexTable(title,x,y):
    n_x = [i for i in x]
    n_y = [i for i in y]
    n_x.reverse()
    n_y.reverse()
    print(title)
    for i in range(len(n_x)):
        print("{:f} & {:f} \\\\".format(n_x[i],n_y[i]))

# variables initialization
run = 100 # independent simulation
ms = 1000 # MC steps
ps = 100 # L^2 update moves per MC step
avHs = [] # average energy for all T, for all independent simulation
avMs = [] # average magnetization for all T, for all independent simulation
cvs = [] # heat capacity for all T, for all independent simulation
suss = [] # susceptibility for all T, for all independent simulation
```

```
Ts = [i/10 for i in range(1,41)] # initialize 40 T values between 4.0 to 0.1
Ts.reverse() # reverse T values to descending order
Tlen = len(Ts)
imageindex = 0
q4Ts = newTs(Ts, list(np.linspace(4,0.1,5,endpoint=True))) # 5 T values for q4 between 4.0 to 0.1 inclusive
halfms = int(ms/2)
lastm = ms - 1
lastp = ps - 1
lastrun = run - 1
grids = list(np.zeros((Tlen,10, 10), dtype='int')) # initialize grid to all 0
count = 0 # counter for probability distribution graph

for n in range(run): # for each independent simulation
    grid = [ [choice([-1,1]) for x in range(10)] for y in range(10) ] # generate a random grid
    h = Htotal(grid) # get total energy
    mag = Mtotal(grid) # get total magnetization
    avHTs = [] # average energy for all T, for this simulation
    avMTs = [] # average magnetization for all T, for this simulation
    cvTs = [] # heat capacity for all T, for this simulation
    susTs = [] # susceptibility for all T, for this simulation
    for T in Ts: # for each T
        if n == lastrun: # only save and show grid for last simulation
            # save current grid
            for j in range(10):
                for i in range(10):
                    grids[imageindex][j][i] = grid[j][i]
            imageindex = imageindex + 1

        htot = 0
        mtot = 0
        hsqtot = 0
        msqtot = 0
        for m in range(ms): # for each MC step
            for p in range(ps): # for each update move
                # test water
                gridtest = [ [grid[j][i] for i in range(10)] for j in range(10)] # copy grid
                randx = choice(range(10)) # generate random coordinate x
                randy = choice(range(10)) # generate random coordinate y
                Hbp = neighbourMulSum(randx,randy,gridtest) * -1.0 # energy before test spin flip
                gridtest[randx][randy]=-gridtest[randx][randy] # test spin flip
                Hap = neighbourMulSum(randx,randy,gridtest) * -1.0  # energy after test spin flip
                deltaH = Hap - Hbp # change in energy after test flip

                # actual spin flip if conditions are satisfied
                r = uniform(0,1) # generate uniform random value between 0 and 1
                if deltaH < 0 or r < exp(-deltaH/T): # change in energy is negative or probability is satisfied
                    grid[randx][randy]=-grid[randx][randy] # actual spin flip
                    h = h + deltaH # update energy value
                    mag = mag + grid[randx][randy]*2/100 # update magnetization value

            if m >= halfms: # more than half of all MC steps then is considered thermal equilibrium
                htot = htot + h # sum all energy for averaging later
                mtot = mtot + mag # sum all magnetization for averaging later
                hsqtot = hsqtot + h**2 # sum all energy squared for averaging later
                msqtot = msqtot + mag**2 # sum all magnetization squared for averaging later
                if T in q4Ts and n == lastrun and m == lastm: # generate probability distribution only for specified T (one of those 5 values) matching any of the original 40 T values and only during last simulation and last MC step
                    probDist(grid)
                    count = count + 1

        avHT = htot/halfms # average energy for this T
        avMT = mtot/halfms # average magnetization for this T
        avsqHT = avHT**2 # average energy squared for this T
        avsqMT = avMT**2 # average magnetization squared for this T
        sqavHT = hsqtot/halfms # squared energy average for this T
        sqavMT = msqtot/halfms # squared magnetization average for this T
        avHTs.append(avHT) # save average energy for this T
        avMTs.append(avMT) # save average magnetization for this T
        cvTs.append(1/(T**2)*(sqavHT - avsqHT)) # save heat capacity for this T
        susTs.append(1/T*(sqavMT - avsqMT)) # save susceptibility for this T
```

```
188
189     avHs.append(avHTs) # save average energy for all T for this simulation
190     avMs.append(avMTs) # save average magnetization for all T for this simulation
191     cvs.append(cvTs) # save heat capacity for all T for this simulation
192     suss.append(susTs) # save susceptibility for all T for this simulation
193
194 # set the layout for illustration of the system at all 40 T values to 8 by 5
195 rowsize = 8
196 colsize = 5
197 fig,ax=plt.subplots(nrows=rowsize,ncols=colsize,figsize=(20,20)) # figures formatting and cosmetics
198 fig.tight_layout(pad=3.0) # figures formatting and cosmetics
199
200 # show illustration of the system at all 40 T values
201 for i in range(Tlen):
202     ax[int(i/5), i%5].get_yaxis().set_visible(False)
203     ax[int(i/5), i%5].get_xaxis().set_visible(False)
204     ax[int(i/5), i%5].set_title("T = {}".format(Ts[i]),fontsize=7)
205     ax[int(i/5)][i%5].imshow(grids[i],cmap='gray',vmin=-1,vmax=1,aspect='equal',interpolation='
        nearest')
206 plt.show()
207
208
209 av2Hs = [] # average of the average energy for all T, for all simulation
210 avcvs = [] # average of heat capacity for all T, for all simulation
211 for j in range(Tlen):  # for each T as column
212     avHsTot = 0
213     cvsTot = 0
214     for i in range(run):  # for each independent simulation as column
215         avHsTot = avHsTot + avHs[i][j] # sum average energy from all independent simulations for
        this T for further averaging later
216         cvsTot = cvsTot + cvs[i][j] # sum average heat capacity from all independent simulations for
         this T for further averaging later
217     av2Hs.append(avHsTot/run) # further averaging for average energy for this T
218     avcvs.append(cvsTot/run) # further averaging for average heat capacity for this T
219
220 plt.subplot(2, 1, 1) # plot layout setting
221 plt.plot(Ts,av2Hs, 'r-') # plot average energy for all T averaged from all simulation
222 plt.title('Graph of Average Energy and Heat Capacity vs Temperature') # title
223 plt.ylabel('Average Energy') # y label
224 genReverseLatexTable('Average Energy',Ts,av2Hs) # generate Latex dataset
225
226 plt.subplot(2, 1, 2) # plot layout setting
227 plt.plot(Ts,avcvs, 'b-') # plot heat capacity for all T averaged from all simulation
228 plt.xlabel('Temperature') # x label
229 plt.ylabel('Heat Capacity') # y label
230 genReverseLatexTable('Heat Capacity',Ts,avcvs) # generate Latex dataset
231 plt.show() # show the plot
232
233 plt.subplot(2, 1, 1) # plot layout setting
234 plt.plot(Ts,avMs[lastrun], 'r-') # only plot average magnetization for all T for last simulation
235 plt.title('Graph of Average Magnetization and Susceptibility vs Temperature') # title
236 plt.ylabel('Average Magnetization') # y label
237 genReverseLatexTable('Average Magnetization',Ts,avMs[lastrun]) # generate Latex dataset
238
239 plt.subplot(2, 1, 2) # plot layout setting
240 plt.plot(Ts,suss[lastrun], 'b-') # only plot susceptibility for all T for last simulation
241 plt.xlabel('Temperature') # x label
242 plt.ylabel('Susceptibility') # y label
243 genReverseLatexTable('Susceptibility',Ts,suss[lastrun]) # generate Latex dataset
244 plt.show() # show the plot
```

Listing 1: Monte Carlo Python Simulation File

Temperature T	Average Energy $\langle E \rangle$
0.1	-200.000000
0.2	-200.000000
0.3	-200.000000
0.4	-200.000000
0.5	-200.000000
0.6	-199.999840
0.7	-199.994640
0.8	-199.959360
0.9	-199.893520
1.0	-199.704240
1.1	-199.412080
1.2	-198.836560
1.3	-198.059360
1.4	-196.803520
1.5	-195.107840
1.6	-192.911920
1.7	-189.789280
1.8	-185.712000
1.9	-181.001280
2.0	-174.703200
2.1	-166.183200
2.2	-156.250480
2.3	-143.124320
2.4	-131.995200
2.5	-118.074880
2.6	-107.815280
2.7	-100.040400
2.8	-92.166720
2.9	-87.222960
3.0	-82.502160
3.1	-78.741600
3.2	-74.871920
3.3	-71.648160
3.4	-68.634480
3.5	-66.222640
3.6	-63.780000
3.7	-61.335120
3.8	-59.519280
3.9	-57.600480
4.0	-55.648160

(a) Temperature T and Average Energy $\langle E \rangle$

Temperature T	Heat Capacity C_v
0.1	0.000000
0.2	0.000000
0.3	0.000000
0.4	0.000000
0.5	0.000000
0.6	0.003548
0.7	0.089029
0.8	0.515550
0.9	1.076819
1.0	2.455944
1.1	4.059295
1.2	6.840299
1.3	9.823511
1.4	14.490010
1.5	19.583755
1.6	25.538257
1.7	34.054436
1.8	44.307861
1.9	54.682800
2.0	70.551703
2.1	92.192352
2.2	110.924838
2.3	125.958214
2.4	123.304323
2.5	109.434761
2.6	90.715857
2.7	74.367163
2.8	59.134921
2.9	51.137181
3.0	43.585194
3.1	38.384421
3.2	33.567535
3.3	30.439533
3.4	27.182868
3.5	24.992956
3.6	23.161960
3.7	21.109158
3.8	19.985494
3.9	17.973747
4.0	17.128732

(b) Temperature T and Heat Capacity C_v

Table 1: Table of Simulation Data

Temperature T	Average Magnetization $\langle m \rangle$
0.1	1.000000
0.2	1.000000
0.3	1.000000
0.4	1.000000
0.5	1.000000
0.6	1.000000
0.7	1.000000
0.8	0.999920
0.9	0.999680
1.0	0.999080
1.1	0.998720
1.2	0.997920
1.3	0.995440
1.4	0.993160
1.5	0.987520
1.6	0.977920
1.7	0.973840
1.8	0.960000
1.9	0.940880
2.0	0.922120
2.1	0.839200
2.2	0.823360
2.3	0.650480
2.4	-0.465480
2.5	-0.072840
2.6	0.096400
2.7	-0.066240
2.8	-0.150000
2.9	-0.025080
3.0	0.062600
3.1	0.051520
3.2	-0.063000
3.3	-0.060880
3.4	0.057240
3.5	-0.040960
3.6	-0.005800
3.7	-0.021240
3.8	0.031200
3.9	0.050920
4.0	0.002280

(c) Temperature T and Average Magnetization $\langle m \rangle$

Temperature T	Susceptibility χ_m
0.1	-0.000000
0.2	-0.000000
0.3	-0.000000
0.4	-0.000000
0.5	-0.000000
0.6	-0.000000
0.7	-0.000000
0.8	0.000002
0.9	0.000007
1.0	0.000021
1.1	0.000025
1.2	0.000042
1.3	0.000079
1.4	0.000107
1.5	0.000205
1.6	0.000360
1.7	0.000530
1.8	0.001039
1.9	0.001440
2.0	0.002144
2.1	0.008408
2.2	0.006305
2.3	0.097568
2.4	0.135162
2.5	0.141207
2.6	0.105009
2.7	0.064454
2.8	0.043372
2.9	0.065895
3.0	0.031828
3.1	0.031805
3.2	0.022253
3.3	0.023757
3.4	0.020440
3.5	0.017962
3.6	0.015230
3.7	0.010896
3.8	0.012405
3.9	0.011465
4.0	0.010475

(d) Temperature T and Susceptibility χ_m

Table 1: Table of Simulation Data (Continued)