# PH5730 - Assignment 1

#### Himanshu Rajnish Borkar - EE22B070

March 2, 2025

### 1 Theory

The energy (restriction) of a lattice system is represented by:

$$E = J \sum_{\langle i,j \rangle} s_i s_j \tag{1}$$

# 2 Ferromagnetic Lattice (J = -1)

The following Python code simulates the Ising Model for the ferromagnetic lattice:

```
#Ferromagnetic Lattice
   import numpy as np
   import matplotlib.pyplot as plt
   # Lattice generation
   s_{-} = [1, -1]
6
   N = 10 # Lattice size
   lattice = np.random.choice(s_{-}, size=(N, N))
10
11
   lower_temp = 0.5
   upper_temp = 7
12
   temp_pts = 200
13
   t_limit = 2000
14
   limit = 9000
   # Restriction Calc updated
17
18
   def energyCalc(lattice, N):
19
       ener = 0
for i in range(N):
20
21
          for j in range(N):
              s = lattice[i,j]
23
              24
                  -1)%N]
               ener += s*nb
       return -ener
26
   #Magnetization Calculations
28
29
   def magCalc(lattice):
30
       mag = np.sum(lattice)
31
       return mag
   #Monte Carlo Steps - Metropolis Algorithm
34
35
   def mc_steps(lattice, beta):
36
37
       for i in range(N):
          (a,b) = np.random.randint(0, N, size=2)
38
          s = lattice[a,b]
```

```
n_b = lattice[(a+1)\%N,b] + lattice[a,(b+1)\%N] + lattice[(a-1)\%N,b] + lattice[a,(b-1)
40
                 % N 7
             delta_restriction = 2*s*n_b
41
42
             if delta_restriction < 0:</pre>
               s *= -1
43
             elif np.random.rand() < np.exp(-delta_restriction*beta):</pre>
44
45
              s *= -1
             lattice[a,b] = s
46
47
        return lattice
48
    Temperatures = np.linspace(lower_temp,upper_temp,temp_pts)
49
50
    energy_arr = []
    mag_arr = []
52
    C_arr = []
53
    X_arr = []
54
55
56
57
    for temp in Temperatures:
        ener = 0
58
59
        magn = 0
        ener2 = 0
60
        magn2 = 0
61
62
        beta = 1/temp
        l = np.copy(lattice)
64
        #1 = thermalisation(lattice,N,t_limit,beta)
65
        for t in range(t_limit):
66
             mc_steps(1,beta)
67
        for z in range(limit):
68
             mc_steps(1,beta)
69
             e = energyCalc(1,N)
70
             m = magCalc(1)
71
             ener += e
73
             ener2 += e**2
74
             magn += m
76
             magn2 += m**2
77
78
        energy_arr.append(ener/(N*N*limit))
        C_arr.append(((ener2/(N*N*limit) - ((ener*ener)/(N*N*limit*limit)))*(beta**2)))
79
        mag_arr.append(magn/(N*N*limit))
80
         X_{arr.append(((magn2/(N*N*limit) - ((magn*magn)/(N*N*limit*limit)))*(beta))) 
81
82
    f = plt.figure(figsize=(18, 10))
83
84
    sp = f.add_subplot(2, 2, 1)
85
    plt.scatter(Temperatures, energy_arr, s=10, marker='o', color='red')
    plt.xlabel("Temperature (T)")
87
    plt.ylabel("Energy")
    plt.axis('tight')
89
90
    sp = f.add_subplot(2, 2, 3 )
91
    plt.scatter(Temperatures, C_arr, s=10, marker='o', color='blue')
plt.xlabel("Temperatureu(T)")
92
93
    plt.ylabel("Specific_heat_")
94
    plt.axis('tight')
95
96
    sp = f.add_subplot(2, 2, 2)
97
    plt.scatter(Temperatures, np.abs(mag_arr), s=10, marker='o', color='green')
98
    plt.xlabel("Temperature (T)")
99
    plt.ylabel("Magnetization_")
    plt.axis('tight')
102
103
    sp = f.add_subplot(2, 2, 4)
   plt.scatter(Temperatures, X_arr, s=10, marker='o', color='black')
104
   plt.xlabel("Temperature<sub>□</sub>(T)")
plt.ylabel("Magnetic_Susceptibility_")
```

```
plt.axis('tight')
```

## 3 Anti-Ferromagnetic Lattice (J = +1)

The following Python code simulates the Ising Model for ferromagnetic lattice:

```
# Anti-Ferromagnetic Lattice
          import numpy as np
         import matplotlib.pyplot as plt
         # Lattice generation
         s_{-} = [1, -1]
 6
         N = 10 # Lattice size
         lattice = np.random.choice(s_, size=(N, N))
 9
10
         lower_temp = 0.5
         upper_temp = 7
12
         temp_pts = 200
13
         t_limit = 2000
14
         limit = 9000
15
16
17
          # Restriction Calc updated
18
          def energyCalc(lattice, N):
19
                     ener = 0
20
                     for i in range(N):
21
22
                                 for j in range(N):
                                            s = lattice[i,j]
23
                                              \label{eq:nb} \begin{tabular}{ll} nb = lattice[(i+1)\%N,j] + lattice[i,(j+1)\%N] + lattice[(i-1)\%N,j] + lattice[i,(j+1)\%N] + lattice[i
                                                        -1)%N]
25
26
                     return ener
27
          #Magnetization Calculations
29
30
          def magCalc(lattice):
                     mag = np.sum(lattice)
                     return mag
33
          #Monte Carlo Steps - Metropolis Algorithm
34
35
          def mc_steps(lattice, beta):
36
37
                     for i in range(N):
                                 (a,b) = np.random.randint(0, N, size=2)
38
                                 s = lattice[a,b]
39
                                 n_b = lattice[(a+1)\%N,b] + lattice[a,(b+1)\%N] + lattice[(a-1)\%N,b] + lattice[a,(b-1)]
40
                                            %N7
                                 delta_restriction = -2*s*n_b
41
                                 if delta_restriction < 0:</pre>
42
43
                                 elif np.random.rand() < np.exp(-delta_restriction*beta):</pre>
44
45
                                 lattice[a,b] = s
47
                     return lattice
48
49
         Temperatures = np.linspace(lower_temp,upper_temp,temp_pts)
50
51
         energy_arr = []
         mag_arr = []
52
53
         C_arr = []
         X_arr = []
54
55
         for temp in Temperatures:
57
                     ener = 0
```

```
magn = 0
59
60
        ener2 = 0
        magn2 = 0
61
62
        beta = 1/temp
63
        1 = np.copy(lattice)
        #1 = thermalisation(lattice,N,t_limit,beta)
65
        for t in range(t_limit):
66
67
             mc_steps(1,beta)
        for z in range(limit):
68
69
             mc_steps(1,beta)
             e = energyCalc(1,N)
70
             m = magCalc(1)
71
72
             ener += e
73
             ener2 += e**2
74
             magn += m
75
             magn2 += m**2
76
77
        energy_arr.append(ener/(N*N*limit))
78
        \texttt{C\_arr.append}(((\texttt{ener2}/(\texttt{N*N*limit}) - ((\texttt{ener*ener})/(\texttt{N*N*limit*limit})))*(\texttt{beta**2})))
79
        mag_arr.append(magn/(N*N*limit))
80
        X_{arr.append(((magn2/(N*N*limit) - ((magn*magn)/(N*N*limit*limit)))*(beta)))}
81
82
    f = plt.figure(figsize=(18, 10))
83
    sp = f.add_subplot(2, 2, 1)
85
    plt.scatter(Temperatures, energy_arr, s=10, marker='o', color='red')
86
    plt.xlabel("Temperature_(T)")
87
    plt.ylabel("Energy")
88
    plt.axis('tight')
89
90
    sp = f.add_subplot(2, 2, 3)
    plt.scatter(Temperatures, C_arr, s=10, marker='o', color='blue')
92
    plt.xlabel("Temperature (T)")
93
    plt.ylabel("Specific_heat_")
94
    plt.axis('tight')
95
    sp = f.add_subplot(2, 2, 2)
97
    plt.scatter(Temperatures, np.abs(mag_arr), s=10, marker='o', color='green')
plt.xlabel("Temperature_u(T)")
98
99
    plt.ylabel("Magnetization")
100
    plt.axis('tight')
102
    sp = f.add_subplot(2, 2, 4 )
    plt.scatter(Temperatures, X_arr, s=10, marker='o', color='black')
104
   plt.xlabel("Temperature(T)")
105
   plt.ylabel("Magnetic_Susceptibility_")
   plt.axis('tight')
```