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]
N = 10  # Lattice size

lattice = np.random.choice(s_, size=(N, N))

lower_temp = 0.5
upper_temp = 7
temp_pts = 200
t_limit = 2000
limit = 9000

# Restriction Calc updated

def energyCalc(lattice, N):
    ener = 0
    for i in range(N):
        for j in range(N):
            s = lattice[i,j]
            nb = lattice[(i+1)%N,j] + lattice[i,(j+1)%N] + lattice[(i-1)%N,j] + lattice[i,(j
                -1)%N]
            ener += s*nb
    return -ener

#Magnetization Calculations

def magCalc(lattice):
    mag = np.sum(lattice)
    return mag

#Monte Carlo Steps - Metropolis Algorithm

def mc_steps(lattice, beta):
    for i in range(N):
        (a,b) = np.random.randint(0, N, size=2)
        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)%N]
        delta_restriction = 2*s*n_b
        if delta_restriction < 0:
          s *= -1
        elif np.random.rand() < np.exp(-delta_restriction*beta):
          s *= -1
        lattice[a,b] = s
    return lattice

Temperatures = np.linspace(lower_temp,upper_temp,temp_pts)

energy_arr = []
mag_arr = []
C_arr = []
X_arr = []


for temp in Temperatures:
    ener = 0
    magn = 0
    ener2 = 0
    magn2 = 0

    beta = 1/temp
    l = np.copy(lattice)
    #l = thermalisation(lattice,N,t_limit,beta)
    for t in range(t_limit):
        mc_steps(l,beta)
    for z in range(limit):
        mc_steps(l,beta)
        e = energyCalc(l,N)
        m = magCalc(l)

        ener += e
        ener2 += e**2
        magn += m
        magn2 += m**2

    energy_arr.append(ener/(N*N*limit))
    C_arr.append(((ener2/(N*N*limit) - ((ener*ener)/(N*N*limit*limit)))*(beta**2)))
    mag_arr.append(magn/(N*N*limit))
    X_arr.append(((magn2/(N*N*limit) - ((magn*magn)/(N*N*limit*limit)))*(beta)))

f = plt.figure(figsize=(18, 10))

sp =  f.add_subplot(2, 2, 1 )
plt.scatter(Temperatures, energy_arr, s=10, marker='o', color='red')
plt.xlabel("Temperature (T)")
plt.ylabel("Energy ")
plt.axis('tight')

sp =  f.add_subplot(2, 2, 3 )
plt.scatter(Temperatures, C_arr, s=10, marker='o', color='blue')
plt.xlabel("Temperature (T)")
plt.ylabel("Specific heat ")
plt.axis('tight')

sp =  f.add_subplot(2, 2, 2 )
plt.scatter(Temperatures, np.abs(mag_arr), s=10, marker='o', color='green')
plt.xlabel("Temperature (T)")
plt.ylabel("Magnetization ")
plt.axis('tight')

sp =  f.add_subplot(2, 2, 4 )
plt.scatter(Temperatures, X_arr, s=10, marker='o', color='black')
plt.xlabel("Temperature (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]
N = 10  # Lattice size

lattice = np.random.choice(s_, size=(N, N))

lower_temp = 0.5
upper_temp = 7
temp_pts = 200
t_limit = 2000
limit = 9000

# Restriction Calc updated

def energyCalc(lattice, N):
    ener = 0
    for i in range(N):
        for j in range(N):
            s = lattice[i,j]
            nb = lattice[(i+1)%N,j] + lattice[i,(j+1)%N] + lattice[(i-1)%N,j] + lattice[i,(j
                -1)%N]
            ener += s*nb
    return ener

#Magnetization Calculations

def magCalc(lattice):
    mag = np.sum(lattice)
    return mag

#Monte Carlo Steps - Metropolis Algorithm

def mc_steps(lattice, beta):
    for i in range(N):
        (a,b) = np.random.randint(0, N, size=2)
        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)
            %N]
        delta_restriction = -2*s*n_b
        if delta_restriction < 0:
          s *= -1
        elif np.random.rand() < np.exp(-delta_restriction*beta):
          s *= -1
        lattice[a,b] = s
    return lattice

Temperatures = np.linspace(lower_temp,upper_temp,temp_pts)

energy_arr = []
mag_arr = []
C_arr = []
X_arr = []


for temp in Temperatures:
    ener = 0
```

```python
    magn = 0
    ener2 = 0
    magn2 = 0

    beta = 1/temp
    l = np.copy(lattice)
    #l = thermalisation(lattice,N,t_limit,beta)
    for t in range(t_limit):
        mc_steps(l,beta)
    for z in range(limit):
        mc_steps(l,beta)
        e = energyCalc(l,N)
        m = magCalc(l)

        ener += e
        ener2 += e**2
        magn += m
        magn2 += m**2

    energy_arr.append(ener/(N*N*limit))
    C_arr.append(((ener2/(N*N*limit) - ((ener*ener)/(N*N*limit*limit)))*(beta**2)))
    mag_arr.append(magn/(N*N*limit))
    X_arr.append(((magn2/(N*N*limit) - ((magn*magn)/(N*N*limit*limit)))*(beta)))

f = plt.figure(figsize=(18, 10))

sp =  f.add_subplot(2, 2, 1 )
plt.scatter(Temperatures, energy_arr, s=10, marker='o', color='red')
plt.xlabel("Temperature (T)")
plt.ylabel("Energy ")
plt.axis('tight')

sp =  f.add_subplot(2, 2, 3 )
plt.scatter(Temperatures, C_arr, s=10, marker='o', color='blue')
plt.xlabel("Temperature (T)")
plt.ylabel("Specific heat ")
plt.axis('tight')

sp =  f.add_subplot(2, 2, 2 )
plt.scatter(Temperatures, np.abs(mag_arr), s=10, marker='o', color='green')
plt.xlabel("Temperature (T)")
plt.ylabel("Magnetization ")
plt.axis('tight')

sp =  f.add_subplot(2, 2, 4 )
plt.scatter(Temperatures, X_arr, s=10, marker='o', color='black')
plt.xlabel("Temperature (T)")
plt.ylabel("Magnetic Susceptibility ")
plt.axis('tight')
```