```
1 # Exercise 2.2 (d).
 2 import numpy as np
 3 import matplotlib.pyplot as plt
4
5 # Initialize lattice
6 lattice_size = 200
7 lattice = np.sign(np.random.rand(lattice_size,lattice_size) - 0.5)
8 new_lattice = lattice.copy()
9 ten_percent = int(lattice_size*lattice_size/10)
10
11 # Constants
12 J = 1
13 iterations = 1000
14 H values = np.linspace(-0.05,0.05,50)
15 temperatures = np.array([1,2.269,5])
16 magnetization_array = np.zeros([len(H_values),len(temperatures)])
17
18 for temp_i in range(len(temperatures)):
19
20
       T = temperatures[temp_i]
21
       beta = 1/T
22
       energy_step = 0
23
24
       for H_i in H_values:
25
           H = H i
26
27
           m = 0
28
           lattice_copy = lattice.copy()
29
30
           # MC loop
31
           for time_step in range(iterations):
32
33
               # Update randomly 10% of the cells
34
               for update in range(ten_percent):
35
36
                    i = np.random.randint(lattice_size)
37
                    j = np.random.randint(lattice_size)
38
39
                   M = 0
40
                    # Due to boundaries
41
42
                    if i > 0:
43
                        M += lattice copy[i-1,j]
44
                    if i < lattice_size-1:</pre>
45
                        M += lattice_copy[i+1,j]
                    if j > 0:
46
47
                        M += lattice_copy[i,j-1]
48
                    if j < lattice_size-1:</pre>
49
                        M += lattice_copy[i,j+1]
50
51
                    E plus = -H-J*M
52
                    E_{minus} = H+J*M
53
                    prob_plus = np.exp(-beta*E_plus) / (np.exp(-beta*E_plus) + np.exp(-
54
   beta*E_minus))
55
                    rnd = np.random.rand()
56
57
                    if rnd < prob_plus:</pre>
58
                        new_lattice[i,j] = 1
```

localhost:4649/?mode=python 1/2

2022-11-13 22:05 Exercise22d.py

```
59
                    else:
 60
                        new_lattice[i,j] = -1
 61
 62
                lattice copy = new lattice.copy()
 63
 64
            # Computing magnetization per unit volume to measure the state of the
    magnetic property
 65
            m = lattice_copy.sum() / np.power(lattice_size, 2)
            magnetization array[energy step,temp i] = m
 66
 67
            energy_step += 1
 68
 69
            print(H_i)
 70
        print(temp i)
 71
 72
 73 coef_0 = np.polyfit(H_values, magnetization_array[:,0],1)
 74 coef 1 = np.polyfit(H values, magnetization array[:,1],1)
 75 coef_2 = np.polyfit(H_values, magnetization_array[:,2],1)
 76
 77 poly1d 0 = np.poly1d(coef 0)
 78 poly1d 1 = np.poly1d(coef 1)
 79 poly1d_2 = np.poly1d(coef_2)
 80
 |x| = \text{np.array}([0,0,0])
 82 for i in range(len(temperatures)):
        for j in range(len(H values)):
 83
 84
            x[i] += magnetization_array[j,i] / H_values[j]
 85 x = x / len(H_values)
 86 print(f'x = \{x\}')
 87
 88 plt.figure() \# T = 1
 89 plt.plot(H_values, magnetization_array[:,0], 'g', H_values, poly1d_0(H_values), '--
   k')
 90 plt.xlabel('Magnetic field, H')
 91 plt.ylabel('Magnetization, m')
 92 plt.legend(['Magnetic susceptibility','Linear regression'])
 93 plt.savefig('22dt1.png', bbox_inches='tight')
94
 95 plt.figure() # T = 2.269
 96 plt.plot(H_values, magnetization_array[:,1], 'r', H_values, poly1d_1(H_values), '--
   k')
 97 plt.xlabel('Magnetic field, H')
98 plt.ylabel('Magnetization, m')
 99 plt.legend(['Magnetic susceptibility','Linear regression'])
100 plt.savefig('22dtc.png', bbox_inches='tight')
101
102 plt.figure() \# T = 5
103 plt.plot(H_values, magnetization_array[:,2], 'b', H_values, poly1d_2(H_values), '--
104 plt.xlabel('Magnetic field, H')
105 plt.ylabel('Magnetization, m')
106 plt.legend(['Magnetic susceptibility','Linear regression'])
107 plt.savefig('22dt5.png', bbox_inches='tight')
108
109 # plt.legend(['T = 1', 'T = 2.269', 'T = 4'])
110 # plt.xlabel('Magnetic field, H')
111 # plt.ylabel('Magnetization, m')
112
113 plt.show()
```

localhost:4649/?mode=python 2/2