Ising model in 2D

Alicja Nowakowska, 234857

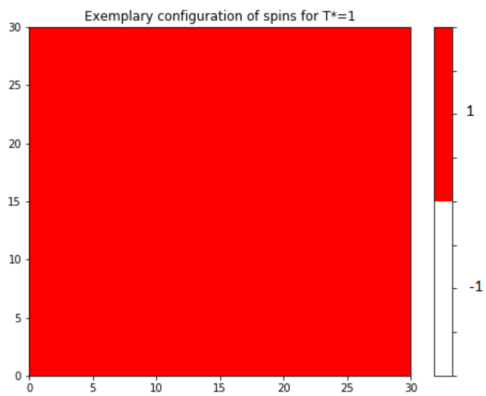
June 2020

1 Introduction

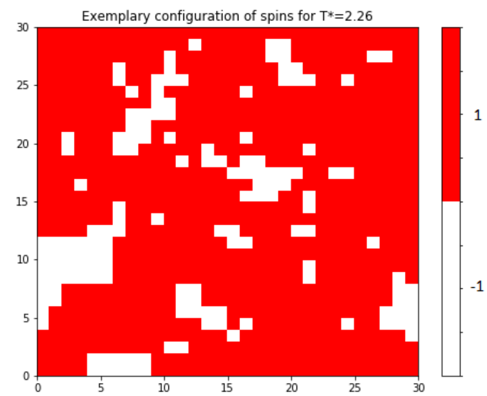
In this report the appropriate plots for the Ising model in 2D are presented.

2 Exemplary configurations

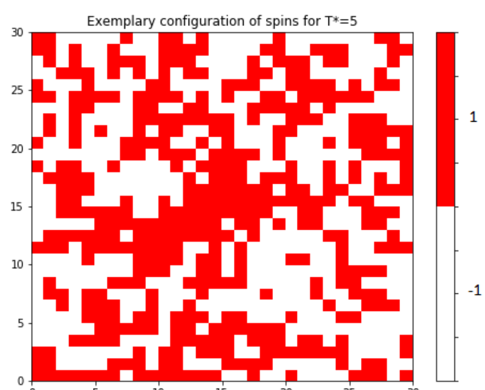
In the first stage of the analysis the exemplary configurations of spins for the lattice of size 30 and on the basis of 5000 Monte Carlo steps for reduced temperatures equal 5, 2.26 and 1 are shown.



(a) $T* = 1$



(b) $T* = 2.26$

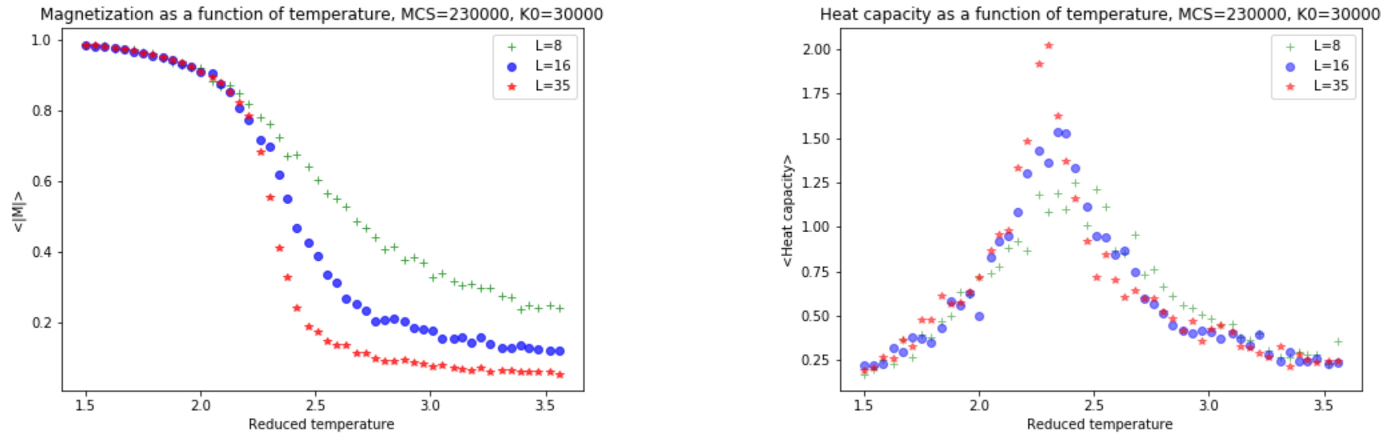


(c) $T* = 5$

Figure 1: Exemplary configurations of spins for $L = 30$ and after $MCS = 5000$

3 Plots of averaged values

The plots of average magnetization and heat capacity in function of reduced temperature for $L = 8$, $L = 16$ and $L = 35$ on the basis of 200 configurations with $MCS = 230000$ and $K_0 = 30000$ are presented.



(a) Average magnetization

(b) Average heat capacity

Figure 2: Plot of averaged properties in the function of reduced temperature

4 Source code

Listing 1: Simulation code

```
def init_ising(L):
    lattice=[[random.choice([-1,1]) for i in range(L)] for i in range(L)]
    IN=[]
    IP=[L-1]
    for i in range(0,L-1):
        IN.append(i+1)
        IP.append(i)
    IN.append(0)
    return lattice,IN,IP
def metropolis(lattice,IN,IP,T): #one iteration , uzywam T* - T zredukowane, T*=Tk/J
    for i in range(len(lattice)):
        for j in range(len(lattice)):
            current=lattice[i][j]
            up=lattice[IP[i]][j]
            down=lattice[IN[i]][j]
            left=lattice[i][IP[j]]
            right=lattice[i][IN[j]]
            deltaU=2*current*(left+right+down+up)
            if deltaU<=0:
                current=-current
            elif random.random()<math.exp(-deltaU/T):
                current=-current
            lattice[i][j]=current
    return lattice
def calculate_ener(lattice,IN,IP):
    ener=0
    for i in range(len(lattice)):
        for j in range(len(lattice)):
            current=lattice[i][j]
            up=lattice[IP[i]][j]
            down=lattice[IN[i]][j]
            left=lattice[i][IP[j]]
            right=lattice[i][IN[j]]
            ener+=current*up + current*down + current*left + current*right
    return ener/2
def total_ising(L,T,MCS,K0):#,analyse): #analyse each 1000th configuration
    #K0 first not important configurations
    lattice,IN,IP=init_ising(L)
    for i in range(K0): #K0 simulations to get the system working
        lattice=metropolis(lattice,IN,IP,T)
    analyse=1000 #analyse each 1000th configuration and get caulate the property value
    magnetization=[]
    for i in range(MCS-K0):
        lattice=metropolis(lattice,IN,IP,T)
        if i%analyse==0:
            m=sum([sum(j) for j in lattice])/len(lattice)/len(lattice)
            magnetization.append(abs(m))
```

```python
    mean_magnetization=sum(magnetization)/len(magnetization)
    return mean_magnetization
def total_ising_heat(L,T,MCS,K0): # heat capcity algorithm
    #K0 first not important configurations
    lattice,IN,IP=init_ising(L)
    for i in range(K0): #K0 simulations to get the system working
        lattice=metropolis(lattice,IN,IP,T)
    analyse=1000
    ener=[]
    for i in range(MCS-K0):
        lattice=metropolis(lattice,IN,IP,T)
        if i%analyse==0:
            e=calculate_ener(lattice,IN,IP)
            ener.append(-e)
    heat_capacity=np.var(ener)/(L*L*T*T)
    return heat_capacity
def plot_lattice(lattice,name,T):
    cmap = colors.ListedColormap(['white','red'])
    fig=plt.figure(figsize=(8,6))
    plt.title("Exemplary configuration of spins for T*="+str(T))
    plt.pcolor(lattice[::-1],cmap=cmap)#,#edgecolors='k', linewidths=3)
    plt.colorbar()
    fig.savefig(name)

lattice=total_ising(30,5,5000,1000)
plot_lattice(lattice,"ex_conf_5.png",5)
T= np.arange(1.5,3.6,2.1/50).round(2).tolist()
L=8 #16,35
MCS= 230000
K0=30000
heat8=[total_ising_heat(L,t,MCS,K0) for t in T] #16,35
a=0.5
fig=plt.figure(figsize=(7,5))
plt.plot(T,heat8,"g+",alpha=a,label="L=8")
plt.plot(T,heat16,"bo",alpha=a,label="L=16")
plt.plot(T,heat35,'r*',alpha=a,label="L=35")
plt.title("Heat capacity as a function of temperature, MCS=230000, K0=30000")
plt.xlabel("Reduced temperature")
plt.ylabel("<Heat capacity>")
plt.legend()
fig.savefig("heat.png")
mag8=[total_ising(L,t,MCS,K0) for t in T] #16,35
fig=plt.figure(figsize=(7,5))
plt.plot(T,mag8,"g+",alpha=a,label="L=8")
plt.plot(T,mag16,"bo",alpha=a,label="L=16")
plt.plot(T,mag35,"r*",alpha=a,label="L=35")
plt.title("Magnetization as a function of temperature, MCS=230000, K0=30000")
plt.xlabel("Reduced temperature")
plt.ylabel("<|M|>")
plt.legend()
plt.show()
fig.savefig("mag.png")
```