Report

Exercise 6.1 Potts model

The Potts model is a generalization of the Ising model, a model of interacting spins on a crystalline lattice - a good model to study phase transitions and critical behavior, both 1st and 2nd order transitions.

$H = \sum_{<ij>} J\delta(\delta_i \delta_j)$**, where** $\delta_i = 1...q$

Run the code Potts.py

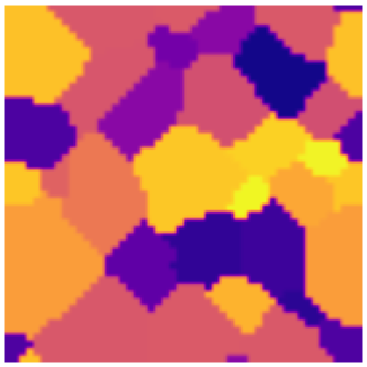
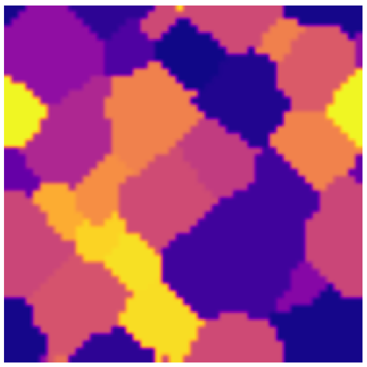
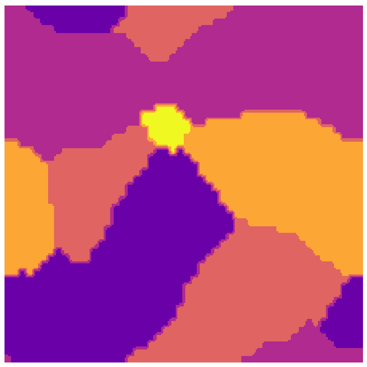
1. Choose a spin orientation at random.
2. Choose the linear dimension. Describe the nature of the spin configurations.
3. Repeat it many times for different spin orientation.
4. Get the pictures that show a phase transition in the model at the certain q. Identify the phase transition type (1st or 2nd order).
5. What is a temperature in this code? Changing the temperature get a few pictures with different phase transitions.

Problem solving:

This program models a 2-D Potts model in a squared lattice with finitely many spin orientations.

It uses the Metropolis Algorithm to decide if the lattice will change to new spin configuration by determine the energy of the new state and old state (old energy and new energy in the code).

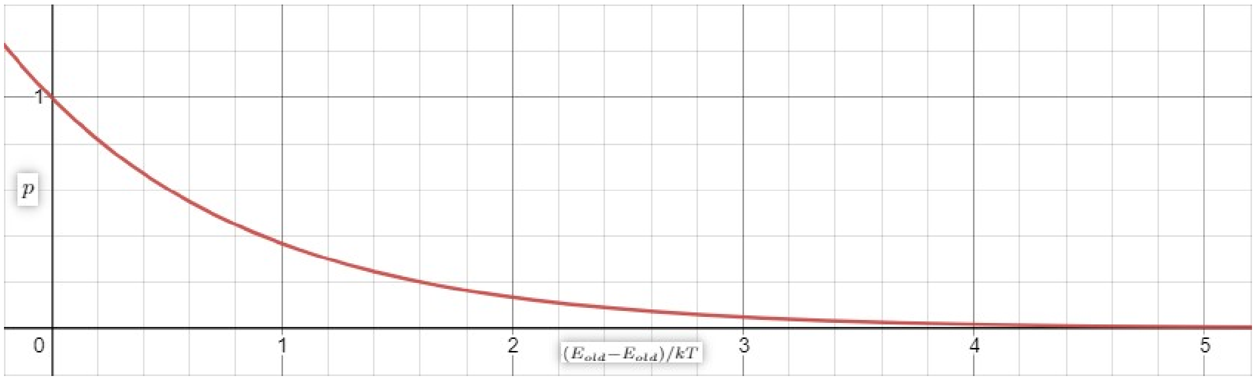
Sample output with randomly picked spin orientations:

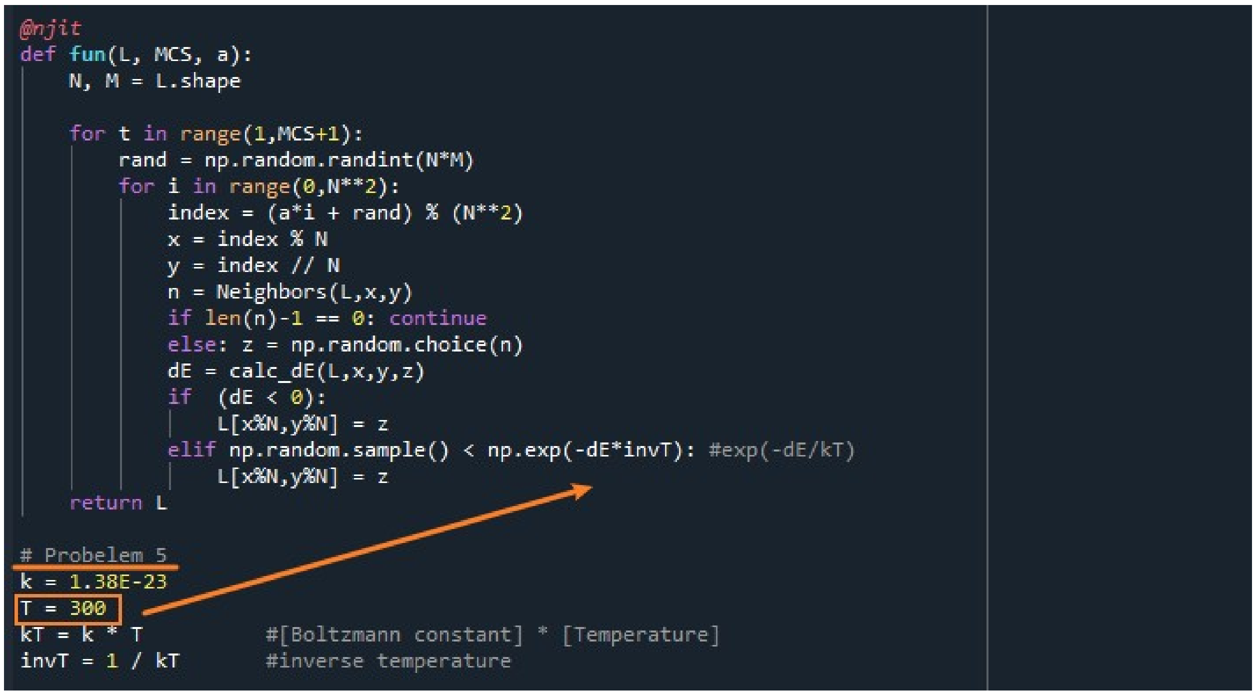
For every iteration, if the new energy E_{new} is less than the old energy E_{old}, it will change to new state because the new state has lower energy ($Probability = 1$). On the other hand, if the new energy E_{new} is greater than the old energy E_{old}, the probability of state change becomes:

$$p = e^{-(E_{old}-E_{old})/kT}$$

Where k is the Boltzmann constant and T is the temperature of the system.



From the figure above we can see by changing the temperature of the system, the probability will change as well. In the program, the temperature is defined in this way:



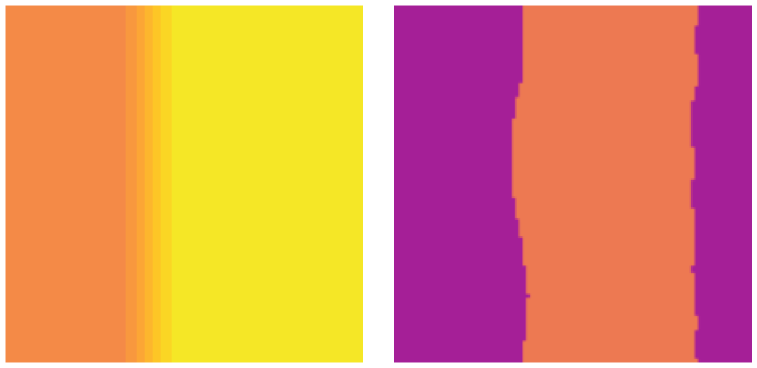
```
@njit
def fun(L, MCS, a):
    N, M = L.shape

    for t in range(1,MCS+1):
        rand = np.random.randint(N*M)
        for i in range(0,N**2):
            index = (a*i + rand) % (N**2)
            x = index % N
            y = index // N
            n = Neighbors(L,x,y)
            if len(n)-1 == 0: continue
            else: z = np.random.choice(n)
            dE = calc_dE(L,x,y,z)
            if  (dE < 0):
                L[x%N,y%N] = z
            elif np.random.sample() < np.exp(-dE*invT): #exp(-dE/kT)
                L[x%N,y%N] = z
    return L

# Probelem 5
k = 1.38E-23
T = 300
kT = k * T          #[Boltzmann constant] * [Temperature]
invT = 1 / kT       #inverse temperature
```

Sample output with different temperature:

First order phase transition:



Second order phase transition: