Monte Carlo Examples

Solved and Coded by Roozbeh Aghabarari

Based on Introduction to Computational Materials Science
by Richard Lesar

Contact Me
ro.aghabarari@gmail.com
www.roozbehaghabarari.com

The 2D Ising Model

Exercise 1.

1.1. Assumptions

- ❖ Size of the cell (n) = 20 (minimum of question data)
- ❖ Number of Monte Carlo Steps (nstep) = 100
- ❖ polarization of initial lattice (pol) = 1 (question data/ all are ↑)
- ❖ Temperature (temp) = 0.25 – 0.5 – 0.75 – 1 – 1.25 – 1.5 – 1.75 – 2 – 2.25 – 2.5 – 2.75 – 3 – 3.25 – 3.5 – 3.75
- ❖ Magnetic field (B) = 0 (without magnetic field)

1.2. MATLAB code

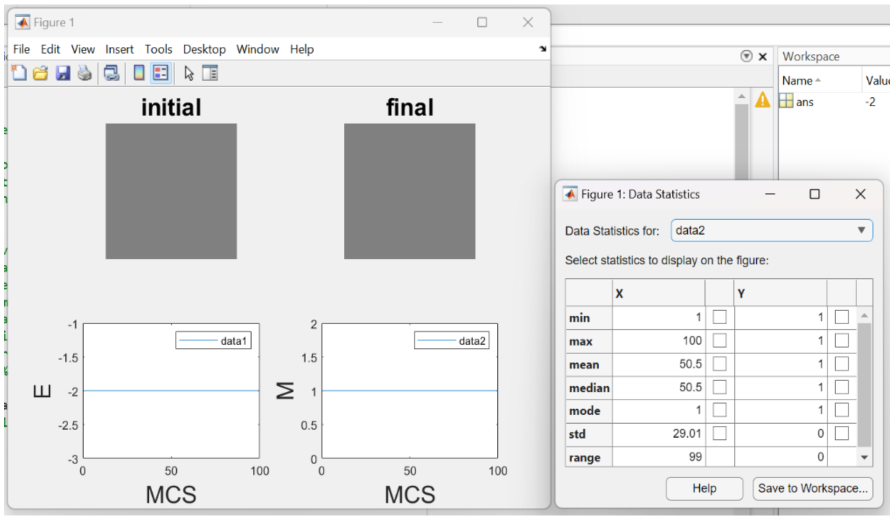
```
%  Monte Carlo for 2D Ising Model
%            Coded by
%       Roozbeh Aghabarari
%
%           Contact Me
%     ro.aghabarari@gmail.com
%    www.roozbehaghabarari.com
%
%  input:
%       n = size of cell.  total number of sites = n^2
%       nstep = number of Monte Carlo Steps (MCS)
%       pol = polarization of initial lattice
%       temp = temperature
%       B = magnetic field
%
%  output:
%       aen = average energy
%       amag = average magnetization
%       en = energy at each MCS
%       magn = magnetization at each MCS
%       s = final configuration
%       so = initial configuration
%       de = %error in final energy (a test to make sure it works)
%       pacc = % of trials accepted
%
function[aen,amag,en,magn,s,so,de,pacc]= MCIsing(n,nstep,pol,temp,B)
% create initial lattice of spins
n=20;
nstep=100;
pol=1;
temp=0.25;
B=0;
N=n*n;
s = initIsing(n,pol);
```

```
% save initial structure
so = s;
% calculate initial energy and polarization
energy = 0;
mag = 0;
for i=1:n
    for j=1:n
        sig = s(pbc(i+1,n),j) + s(i,pbc(j+1,n)) + s(pbc(i-1,n),j) + s(i,pbc(j-1,n));
        energy = energy - s(i,j)*(sig/2 + B);
        mag = mag + s(i,j);
    end
end
%  initialize for averages and output
en=zeros(nstep,1);
magn = zeros(nstep,1);
MCS = 0;
aen = 0;
amag = 0;
iacc = 0;
% define periodic boundary conditions in pbc
%
% do Monte Carlo
%  number of Monte Carlo steps = nstep
%  number of configurations = N*nstep
nconfig = N*nstep;
for k=1:nconfig
%  pick a site to change
    i = ceil(rand*n);
    j = ceil(rand*n);
% flip the spin and calculate energy change
    sij = s(i,j);
    sig = s(pbc(i+1,n),j) + s(i,pbc(j+1,n)) + s(pbc(i-1,n),j) + s(i,pbc(j-1,n));
    del = 2*sij*(sig + B);
% Metropolis algorithm:  only change state if accepted
% if rejected, state stays the same but is added to list
% of configurations
    if del < 0 |  exp(-del/temp) > rand
        s(i,j) = -sij;
        energy = energy + del;
        mag = mag - 2*sij;
        iacc = iacc + 1;
    end
%  accumulate for avearging
    aen = aen + energy;
    amag = amag + mag;
%  every MCS add energy and magnetization to list
    if mod(k,N)==0
        MCS = MCS + 1;
        en(MCS) = energy/N;
        magn(MCS) = mag/N;
    end
end
%  calculate average energy and magnetization
aen = aen/nconfig;
amag = amag/nconfig;
```

```
% output energy and magnetization per site
aen = aen/N;
amag = amag/N;
% test the final energy to make sure no errors in updating
entest = 0;
for i=1:n
    for j=1:n
        sig = s(pbc(i+1,n),j) + s(i,pbc(j+1,n)) + s(pbc(i-1,n),j) + s(i,pbc(j-1,n));
        entest = entest - s(i,j)*(sig/2 + B);
    end
end
de = (energy-entest)/entest;
% calculate the percent acceptance of the moves
pacc=100*iacc/nconfig;

% Displaying the results

subplot(2,2,1)
imagesc(so);colormap(gray);axis equal;axis off;title('initial','fontsize',18)

subplot(2,2,2)
imagesc(s);colormap(gray);axis equal;axis off;title('final','fontsize',18)

subplot(2,2,3)
plot(en);xlabel('MCS','fontsize',18);ylabel('E','fontsize',18)

subplot(2,2,4)
plot(magn);xlabel('MCS','fontsize',18);ylabel('M','fontsize',18)
```

1.3. Results and discussion

Only the first and last temperature figures (T = 0.25 and 3.75) are included here; the remaining figures are saved in the attached file.



Name	Value
ans	-2

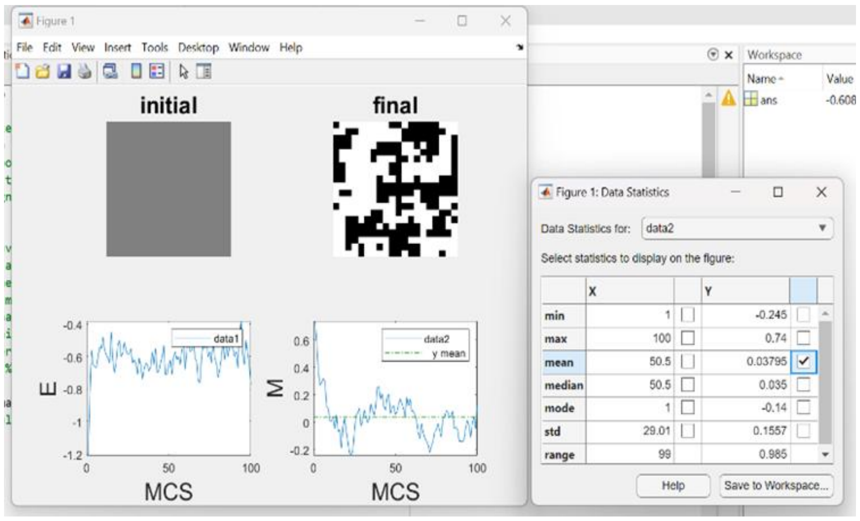
Figure 1: Data Statistics

Data Statistics for: data2

Select statistics to display on the figure:

	X		Y	
min	1	☐	1	☐
max	100	☐	1	☐
mean	50.5	☐	1	☐
median	50.5	☐	1	☐
mode	1	☐	1	☐
std	29.01	☐	0	☐
range	99		0	

Fig. 1. The initial and final configurations, along with the energy and magnetization at each Monte Carlo step for T = 0.25, are shown.



Data Statistics for: data2

Select statistics to display on the figure:

	X		Y	
min	1	☐	-0.245	☐
max	100	☐	0.74	☐
mean	50.5	☐	0.03795	✓
median	50.5	☐	0.035	☐
mode	1	☐	-0.14	☐
std	29.01	☐	0.1557	☐
range	99	☐	0.985	☐

Fig. 2. The initial and final configurations, along with the energy and magnetization at each Monte Carlo step for T = 3.75, are shown.

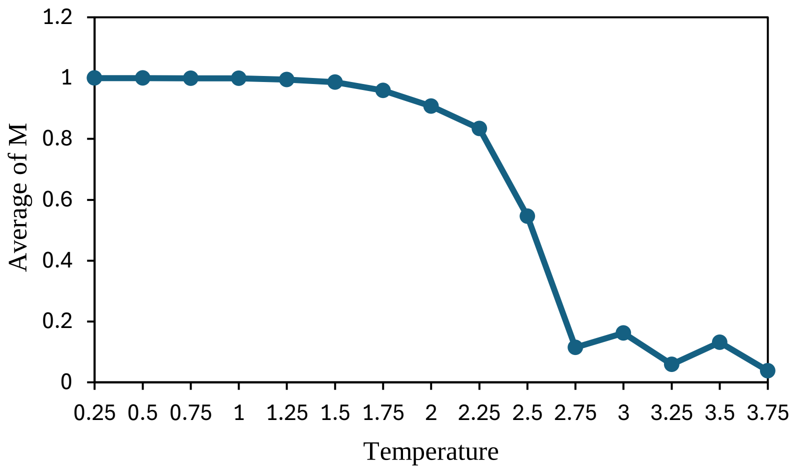


Fig. 3. Average of |M| vs temperature.

$$T_c \approx \frac{2K_B}{J} \quad , \qquad T^*_{reduced\ units} = \frac{T}{K_B}J \;\rightarrow\; T^*_c \approx 2$$

As shown in Figure 3, above the critical temperature (T = 2), the system becomes disordered.

Exercise 2.

2.1. Assumptions

- Size of the cell (n) = 20 (minimum of question data)
- Number of Monte Carlo Steps (nstep) = 100
- polarization of initial lattice (pol) = 1 (question data/ half ↑ half ↓ which a loop is written for this issue)
- Temperature (temp) = 0.25 – 0.5 – 0.75 – 1 – 1.25 – 1.5 – 1.75 – 2 – 2.25 – 2.5 – 2.75 – 3 – 3.25 – 3.5 – 3.75
- Magnetic field (B) = 0 (without magnetic field)

2.2. MATLAB code

```
%  Monte Carlo for 2D Ising Model
%            Coded by
%      Roozbeh Aghabarari
%
%           Contact Me
%    ro.aghabarari@gmail.com
%   www.roozbehaghabarari.com
%
%  input:
%      n = size of cell.  total number of sites = n^2
%      nstep = number of Monte Carlo Steps (MCS)
%      pol = polarization of initial lattice
%      temp = temperature
%      B = magnetic field
%
%  output:
%      aen = average energy
%      amag = average magnetization
%      en = energy at each MCS
%      magn = magnetization at each MCS
%      s = final configuration
%      so = initial configuration
%      de = %error in final energy (a test to make sure it works)
%      pacc = % of trials accepted
%
function[aen,amag,en,magn,s,so,de,pacc]= MCIsing(n,nstep,pol,temp,B)
% create initial lattice of spins
n=20;
nstep=100;
pol=1;
temp=3.75;
B=0;
N=n*n;
s = initIsing(n,pol);
% save initial structure
so = s;
% calculate initial energy and polarization
energy = 0;
```

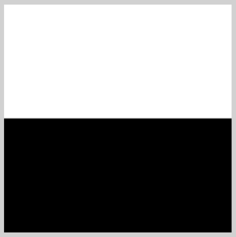
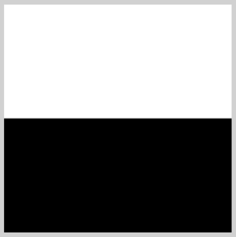
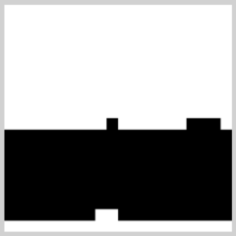
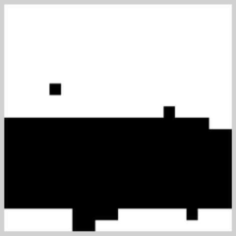
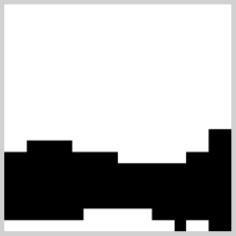
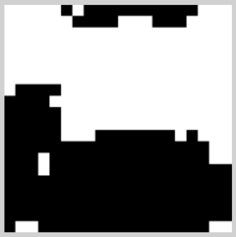
```
mag = 0;
for i=1:n
    for j=1:n
        s(i,j)=(-1)^(floor((2*i-1)/n));
    end
end
for i=1:n
    for j=1:n
        sig = s(pbc(i+1,n),j) + s(i,pbc(j+1,n)) + s(pbc(i-1,n),j) + s(i,pbc(j-1,n));
        energy = energy - s(i,j)*(sig/2 + B);
        mag = mag + s(i,j);
    end
end
%  initialize for averages and output
en=zeros(nstep,1);
magn = zeros(nstep,1);
MCS = 0;
aen = 0;
amag = 0;
iacc = 0;
% define periodic boundary conditions in pbc
%
% do Monte Carlo
%  number of Monte Carlo steps = nstep
%  number of configurations = N*nstep
nconfig = N*nstep;
for k=1:nconfig
%  pick a site to change
    i = ceil(rand*n);
    j = ceil(rand*n);
% flip the spin and calculate energy change
    sij = s(i,j);
    sig = s(pbc(i+1,n),j) + s(i,pbc(j+1,n)) + s(pbc(i-1,n),j) + s(i,pbc(j-1,n));
    del = 2*sij*(sig + B);
% Metropolis algorithm:  only change state if accepted
% if rejected, state stays the same but is added to list
% of configurations
    if del < 0 |  exp(-del/temp) > rand
        s(i,j) = -sij;
        energy = energy + del;
        mag = mag - 2*sij;
        iacc = iacc + 1;
    end
%  accumulate for avearging
    aen = aen + energy;
    amag = amag + mag;
%  every MCS add energy and magnetization to list
    if mod(k,N)==0
        MCS = MCS + 1;
        en(MCS) = energy/N;
        magn(MCS) = mag/N;
    end
end
%  calculate average energy and magnetization
aen = aen/nconfig;
```

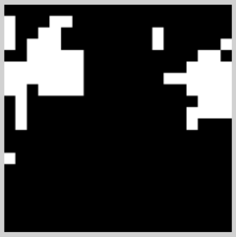
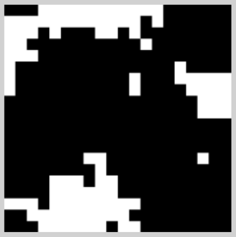
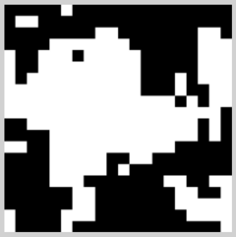
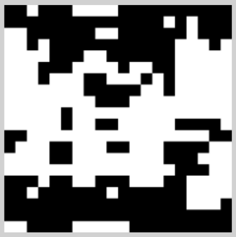
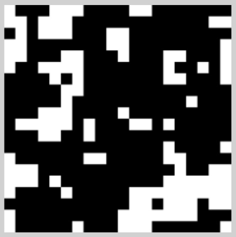
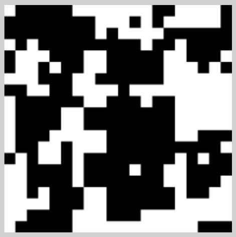
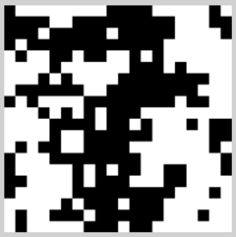
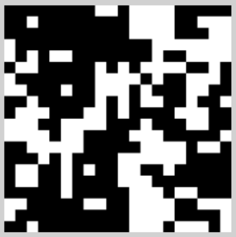
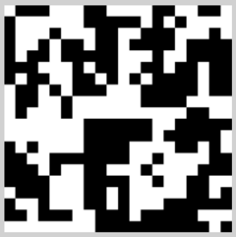
```
amag = amag/nconfig;
% output energy and magnetization per site
aen = aen/N;
amag = amag/N;
% test the final energy to make sure no errors in updating
entest = 0;
for i=1:n
    for j=1:n
        sig = s(pbc(i+1,n),j) + s(i,pbc(j+1,n)) + s(pbc(i-1,n),j) + s(i,pbc(j-1,n));
        entest = entest - s(i,j)*(sig/2 + B);
    end
end
de = (energy-entest)/entest;
% calculate the percent acceptance of the moves
pacc=100*iacc/nconfig;

% Displaying the results - for only investigation of interface

imagesc(s)
colormap(gray)
axis equal
axis off
```

2.3. Results and discussion

Table 1

Final configurations at all temperatures are provided.

 T=0.25	 T=0.5	 T=0.75
 T=1	 T=1.25	 T=1.5

 T=1.75	 T=2	 T=2.25
 T=2.5	 T=2.75	 T=3
 T=3.25	 T=3.5	 T=3.75

As shown in Table 1, at low temperatures such as T = 0.25 and 0.5, the system exhibits complete order, with the top half consisting of spins = -1 and the bottom half of spins = +1. However, as the temperature increases, disorder begins to dominate. The system contains three interfaces: one at the center of the cube and two at the periodic boundaries (top and bottom). With rising temperature, and based on the Von Neumann neighborhood, each small cube starts to interact with its neighbors. At high temperatures, the simulation cube no longer retains any interface and becomes fully disordered. As a result, all three interfaces become unstable, and no clear interface can be recognized.

Exercise 3.

3.1. Assumptions

- Size of the cell (n) = 20 (minimum of question data)
- Number of Monte Carlo Steps (nstep) = 100
- polarization of initial lattice (pol) = 1 (my idea/ all are ↑)
- Temperature (temp) = 3 (for $T > T_c^*$) and 1 (for $T < T_c^*$)

$$T_c \approx \frac{2K_B}{J} \quad , \qquad T^*_{reduced\ units} = \frac{T}{K_B}J \ \rightarrow \ T_c^* \approx 2$$

- Magnetic field (B) = 0 – 3 – 6 – 7 – 8 – 10 (without magnetic field)

3.2. MATLAB code

```
%  Monte Carlo for 2D Ising Model
%            Coded by
%       Roozbeh Aghabarari
%
%           Contact Me
%     ro.aghabarari@gmail.com
%    www.roozbehaghabarari.com
%
%  input:
%       n = size of cell.  total number of sites = n^2
%       nstep = number of Monte Carlo Steps (MCS)
%       pol = polarization of initial lattice
%       temp = temperature
%       B = magnetic field
%
%  output:
%       aen = average energy
%       amag = average magnetization
%       en = energy at each MCS
%       magn = magnetization at each MCS
%       s = final configuration
%       so = initial configuration
%       de = %error in final energy (a test to make sure it works)
%       pacc = % of trials accepted
%
function[aen,amag,en,magn,s,so,de,pacc]= MCIsing(n,nstep,pol,temp,B)
% create initial lattice of spins
n=20;
nstep=100;
pol=1;
temp=1;
B=10;
N=n*n;
s = initIsing(n,pol);
% save initial structure
so = s;
% calculate initial energy and polarization
```

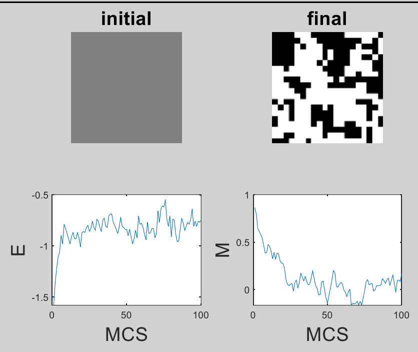
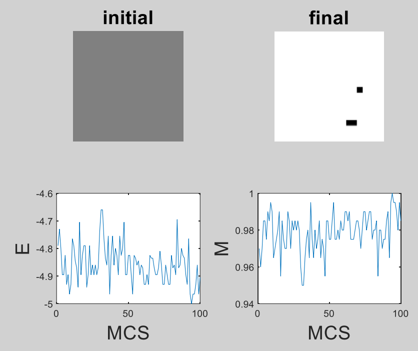
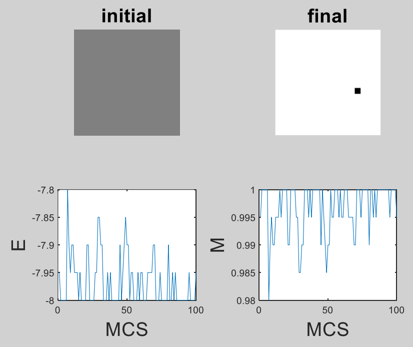
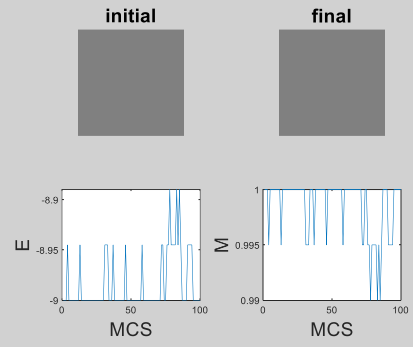
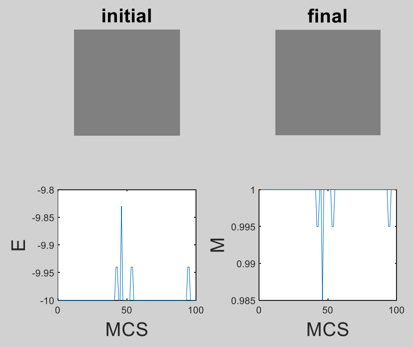
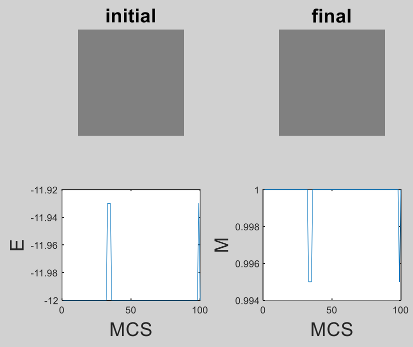
```
energy = 0;
mag = 0;
for i=1:n
    for j=1:n
        sig = s(pbc(i+1,n),j) + s(i,pbc(j+1,n)) + s(pbc(i-1,n),j) + s(i,pbc(j-1,n));
        energy = energy - s(i,j)*(sig/2 + B);
        mag = mag + s(i,j);
    end
end
%  initialize for averages and output
en=zeros(nstep,1);
magn = zeros(nstep,1);
MCS = 0;
aen = 0;
amag = 0;
iacc = 0;
% define periodic boundary conditions in pbc
%
% do Monte Carlo
%  number of Monte Carlo steps = nstep
%  number of configurations = N*nstep
nconfig = N*nstep;
for k=1:nconfig
%  pick a site to change
    i = ceil(rand*n);
    j = ceil(rand*n);
% flip the spin and calculate energy change
    sij = s(i,j);
    sig = s(pbc(i+1,n),j) + s(i,pbc(j+1,n)) + s(pbc(i-1,n),j) + s(i,pbc(j-1,n));
    del = 2*sij*(sig + B);
% Metropolis algorithm:  only change state if accepted
% if rejected, state stays the same but is added to list
% of configurations
    if del < 0 |  exp(-del/temp) > rand
        s(i,j) = -sij;
        energy = energy + del;
        mag = mag - 2*sij;
        iacc = iacc + 1;
    end
%  accumulate for avearging
    aen = aen + energy;
    amag = amag + mag;
%  every MCS add energy and magnetization to list
    if mod(k,N)==0
        MCS = MCS + 1;
        en(MCS) = energy/N;
        magn(MCS) = mag/N;
    end
end
%  calculate average energy and magnetization
aen = aen/nconfig;
amag = amag/nconfig;
% output energy and magnetization per site
aen = aen/N;
amag = amag/N;
```

```
% test the final energy to make sure no errors in updating
entest = 0;
for i=1:n
    for j=1:n
        sig = s(pbc(i+1,n),j) + s(i,pbc(j+1,n)) + s(pbc(i-1,n),j) + s(i,pbc(j-1,n));
        entest = entest - s(i,j)*(sig/2 + B);
    end
end
de = (energy-entest)/entest;
% calculate the percent acceptance of the moves
pacc=100*iacc/nconfig;

% Displaying the results

subplot(2,2,1)
imagesc(so);colormap(gray);axis equal;axis off;title('initial','fontsize',18)

subplot(2,2,2)
imagesc(s);colormap(gray);axis equal;axis off;title('final','fontsize',18)

subplot(2,2,3)
plot(en);xlabel('MCS','fontsize',18);ylabel('E','fontsize',18)

subplot(2,2,4)
plot(magn);xlabel('MCS','fontsize',18);ylabel('M','fontsize',18)
```

3.3. Results and discussion

Table 2

Effect of different magnetic fields on the initial configuration, final configuration, energy at each Monte Carlo step, and magnetization at T = 3 is presented.

T=3	
 B=0	 B=3
 B=6	 B=7
 B=8	 B=10

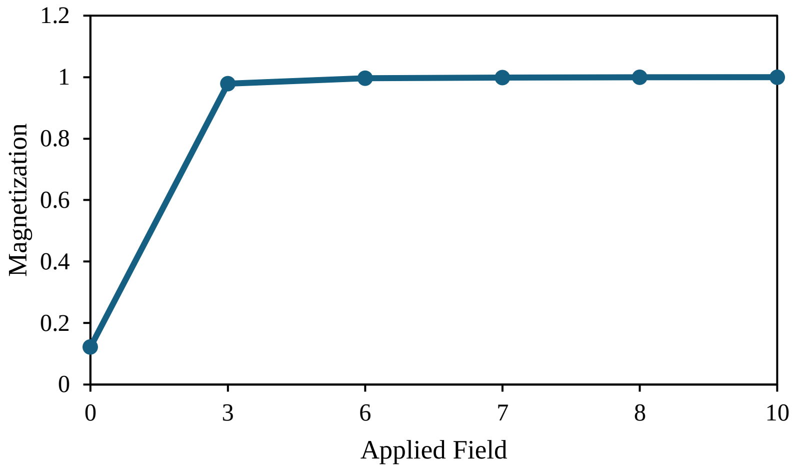
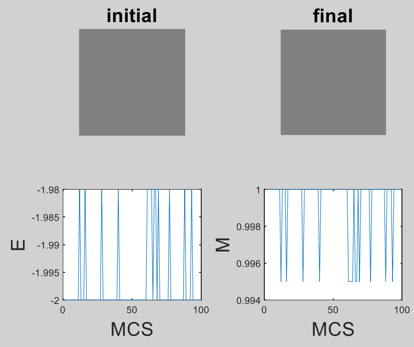
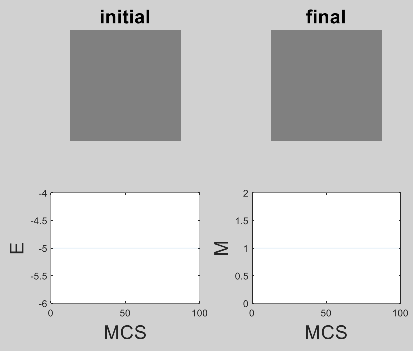
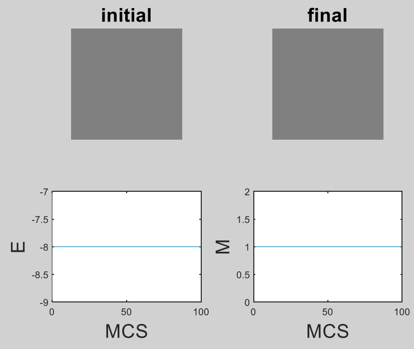
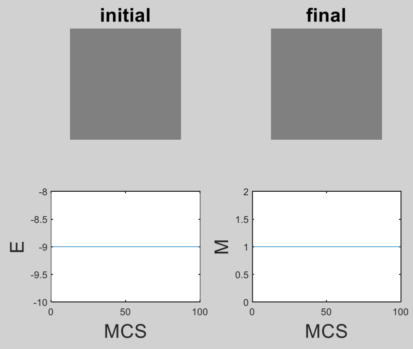
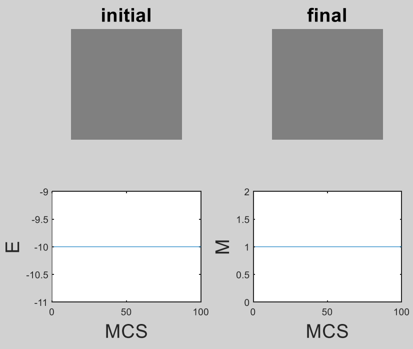
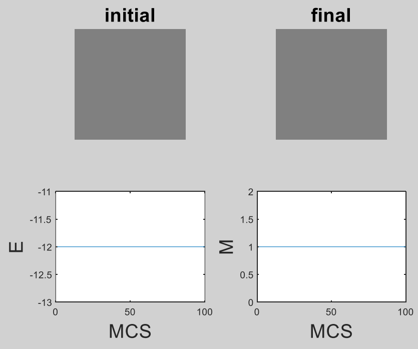


Fig. 4. Effect of the magnetic field on magnetization.

Table 3

Effect of different magnetic fields on the initial configuration, final configuration, energy at each Monte Carlo step, and magnetization at T = 1 is presented.

T=1	
 B=0	 B=3
 B=6	 B=7
 B=8	 B=10

At T = 3, which is above the critical temperature, the simulation cube becomes magnetized once the magnetic field exceeds a threshold value (B = 7). This indicates that the magnetic field influences spin alignment, and at higher field strengths, thermal noise is reduced and the magnetization plateaus near unity. This behavior is not observed at T = 0; in fact, at low temperatures ($T < T_c^*$), the magnetic field is not sufficient to alter the spin order

Exercise 4.

4.1. Assumptions

- ❖ Size of the cell (n) = 20 (minimum of question data)
- ❖ Number of Monte Carlo Steps (nstep) = 100
- ❖ polarization of initial lattice (pol) = 1 (my idea/ all are ↑)
- ❖ Temperature (temp) = 1 – 1.5 – 2 – 2.5 – 3
- ❖ Magnetic field (B) = 0 (without magnetic field)

4.2. MATLAB code

$$E = -\sum_{i=1}^{N}\left(\frac{1}{2}\sum_{i} + B\right)S_i \xrightarrow{J<0} E_{Exercize\ 4} = \sum_{i=1}^{N}\left(\frac{1}{2}\sum_{i} + B\right)S_i$$

Accordingly, a negative sign was added before the energy term.

```
%  Monte Carlo for 2D Ising Model
%            Coded by
%       Roozbeh Aghabarari
%
%           Contact Me
%     ro.aghabarari@gmail.com
%    www.roozbehaghabarari.com
%
%  input:
%       n = size of cell.  total number of sites = n^2
%       nstep = number of Monte Carlo Steps (MCS)
%       pol = polarization of initial lattice
%       temp = temperature
%       B = magnetic field
%
%  output:
%       aen = average energy
%       amag = average magnetization
%       en = energy at each MCS
%       magn = magnetization at each MCS
%       s = final configuration
%       so = initial configuration
%       de = %error in final energy (a test to make sure it works)
%       pacc = % of trials accepted
%
function[aen,amag,en,magn,s,so,de,pacc]= MCIsing(n,nstep,pol,temp,B)
% create initial lattice of spins
n=20;
nstep=100;
pol=1;
temp=1;
B=0;
N=n*n;
```

```
s = initIsing(n,pol);
% save initial structure
so = s;
% calculate initial energy and polarization
energy = 0;
mag = 0;
for i=1:n
    for j=1:n
        sig = s(pbc(i+1,n),j) + s(i,pbc(j+1,n)) + s(pbc(i-1,n),j) + s(i,pbc(j-1,n));
        energy = -(energy - s(i,j)*(sig/2 + B));
        mag = mag + s(i,j);
    end
end
%  initialize for averages and output
en=zeros(nstep,1);
magn = zeros(nstep,1);
MCS = 0;
aen = 0;
amag = 0;
iacc = 0;
% define periodic boundary conditions in pbc
%
% do Monte Carlo
%  number of Monte Carlo steps = nstep
%  number of configurations = N*nstep
nconfig = N*nstep;
for k=1:nconfig
%  pick a site to change
    i = ceil(rand*n);
    j = ceil(rand*n);
% flip the spin and calculate energy change
    sij = s(i,j);
    sig = s(pbc(i+1,n),j) + s(i,pbc(j+1,n)) + s(pbc(i-1,n),j) + s(i,pbc(j-1,n));
    del = 2*sij*(sig + B);
% Metropolis algorithm:  only change state if accepted
% if rejected, state stays the same but is added to list
% of configurations
    if del < 0 |  exp(-del/temp) > rand
        s(i,j) = -sij;
        energy = energy + del;
        mag = mag - 2*sij;
        iacc = iacc + 1;
    end
%  accumulate for avearging
    aen = aen + energy;
    amag = amag + mag;
%  every MCS add energy and magnetization to list
    if mod(k,N)==0
        MCS = MCS + 1;
        en(MCS) = energy/N;
        magn(MCS) = mag/N;
    end
end
%  calculate average energy and magnetization
aen = aen/nconfig;
```

```
amag = amag/nconfig;
% output energy and magnetization per site
aen = aen/N;
amag = amag/N;
% test the final energy to make sure no errors in updating
entest = 0;
for i=1:n
    for j=1:n
        sig = s(pbc(i+1,n),j) + s(i,pbc(j+1,n)) + s(pbc(i-1,n),j) + s(i,pbc(j-1,n));
        entest = entest - s(i,j)*(sig/2 + B);
    end
end
de = (energy-entest)/entest;
% calculate the percent acceptance of the moves
pacc=100*iacc/nconfig;

% Displaying the results

subplot(2,2,1)
imagesc(so);colormap(gray);axis equal;axis off;title('initial','fontsize',18)

subplot(2,2,2)
imagesc(s);colormap(gray);axis equal;axis off;title('final','fontsize',18)

subplot(2,2,3)
plot(en);xlabel('MCS','fontsize',18);ylabel('E','fontsize',18)

subplot(2,2,4)
plot(magn);xlabel('MCS','fontsize',18);ylabel('M','fontsize',18)
```

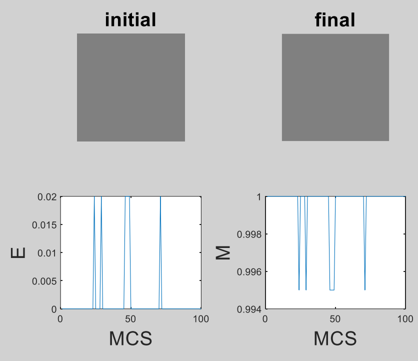
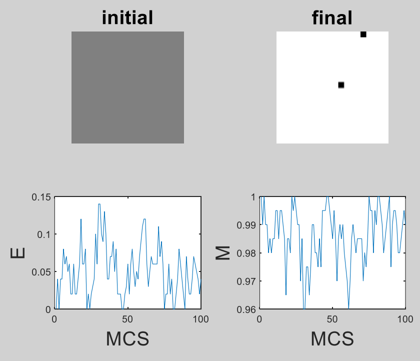
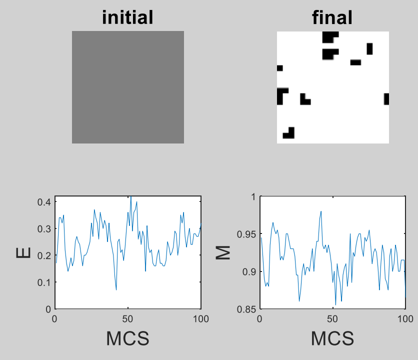
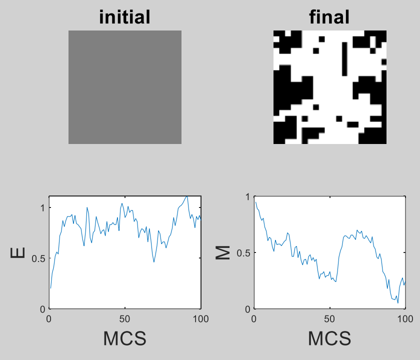
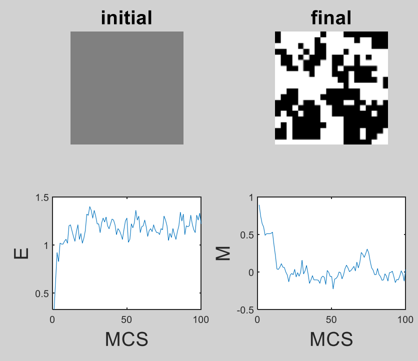
4.3. Results and discussion

The ground state of a quantum system refers to its lowest energy state. It is the state of the system when it is in stable equilibrium, with no excess energy. In this state, the system's wave function is described by a specific mathematical form, and the corresponding energy is known as the ground state energy. The ground state of an atom is the lowest energy state of the atom, where all electrons are in their lowest possible arrangement.

So, Statistical data is plotted in each energy vs. temperature figure.

Table 4

The effect of different temperatures on the initial and final configurations, energy at each Monte Carlo step, and magnetization are shown for various temperatures.

 T=1	 T=1.5
 T=2	 T=2.5
 T=3	

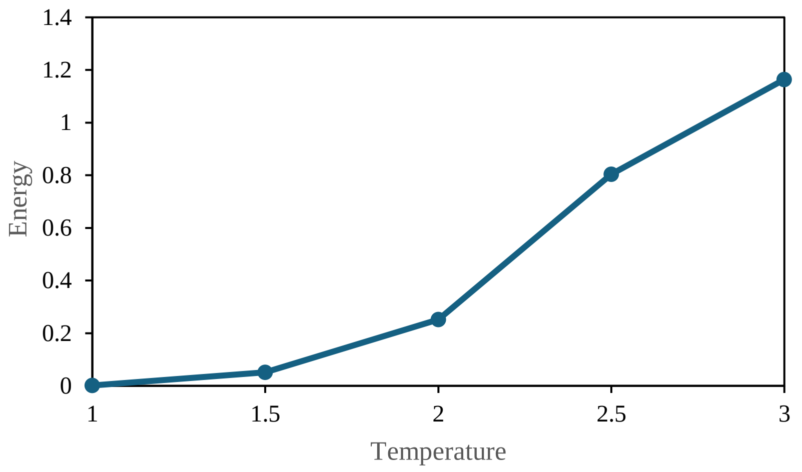


Fig. 5. Effect of temperature on energy.

As shown in Figure 5, the ground state occurs at temperatures lower than 1.5, where the energy is nearly zero. However, above T = 1.5, the energy starts to rise. Regarding the stability of the system, as the temperature increases beyond 1.5, the final configuration begins to diminish, indicating that at high temperatures, the system becomes unstable and fluctuates. It is important to note that after the critical temperature ($T > 2$), as the temperature continues to rise, the system starts to find a new stable state. Therefore, stability is only diminished in the temperature range near the critical temperature ($1.5 < T < 2.5$). For further clarification, please refer to Figure 6, where the system reaches a new order with marginal fluctuations.

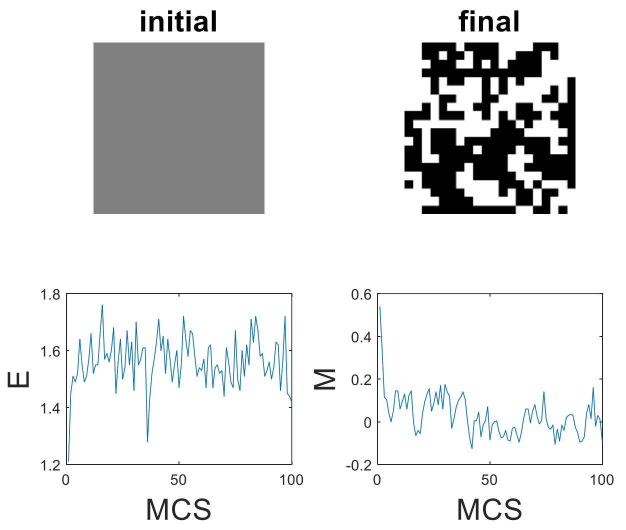


Fig. 6. Initial configuration, final configuration, energy at each Monte Carlo step, and magnetization at T = 5 are presented.

The Q-state Potts Model

Exercise 1.

1.1. Assumptions

- ❖ Size of the cell (n) = 30 (minimum of question data)
- ❖ Number of Monte Carlo Steps (nstep) = 10 – 50 – 100 – 150 – 200
- ❖ The range of possible spin values (Q) = 2 – 4 – 6 – 8

1.2. MATLAB code

Although the code mentioned in the text could be used to display the final configuration, I used the `subplot` command to investigate the entire process, not just the final state.

```
imagesc(s);
colormap('default')
axis equal
axis off
```

Final code:

```
%  Monte Carlo for 2D Potts Model
%           Coded by
%       Roozbeh Aghabarari
%
%          Contact Me
%     ro.aghabarari@gmail.com
%    www.roozbehaghabarari.com
%
%  input:
%       n = size of cell.  total number of sites = n^2
%       nstep = number of Monte Carlo Steps (MCS)
%       Q = the range of possible spin values (1 to Q)
%
%  output:
%       en = energy at each MCS
%       so = initial configuration
%       s1 = configuration at 1/3 of the run
%       s2 = configuration at 2/3 of the run
%       s = final configuration
%       de = %error in final energy (a test to make sure it works)
%
function[en,so,s1,s2,s,de]= MCPotts(n,nstep,q)
n = 30;
nstep= 10;
q = 2;
% create initial lattice of spins
N=n*n;
s = ceil(q.*rand(n));
% save initial structure
```

```
so = s;
np = floor(nstep/3);
%inline Kronecker delta
delta = inline('n==0');
% calculate initial energy and polarization
energy = 0;
for i=1:n
    for j=1:n
        sij = s(i,j);
        sig = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-
s(pbc(i-1,n),j)) + delta(sij-s(i,pbc(j-1,n)));
        energy = energy + (4-sig)/2;
    end
end
%  initialize for averages and output
en=zeros(nstep,1);
MCS = 0;
% define periodic boundary conditions in pbc
%
% do Monte Carlo
%  number of Monte Carlo steps = nstep
%  number of configurations = N*nstep
nconfig = N*nstep;
for k=1:nconfig
%  pick a site to change
    i = ceil(rand*n);
    j = ceil(rand*n);
% flip the spin and calculate energy change
    sijo = s(i,j);
    sij = sijo;
    sij = sij + ceil((q-1)*rand);
    sij = sij - q*floor((sij-1)/q);
    sign = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-
s(pbc(i-1,n),j)) + delta(sij-s(i,pbc(j-1,n)));
    sigo = delta(sijo-s(pbc(i+1,n),j)) + delta(sijo-s(i,pbc(j+1,n))) + delta(sijo-
s(pbc(i-1,n),j)) + delta(sijo-s(i,pbc(j-1,n)));
    del = -(sign - sigo);
% Metropolis algorithm:  only change state if accepted
% if rejected, state stays the same but is added to list
% of configurations
    if del <= 0
        s(i,j) = sij;
        energy = energy + del;
    end
%  every MCS add energy and magnetization to list
    if mod(k,N)==0
        MCS = MCS + 1;
        en(MCS) = energy;
        if MCS == np
            s1 = s;
        end
        if MCS == 2*np
            s2 = s;
        end
    end
```

```
end

% test the final energy to make sure no errors in updating
entest = 0;
for i=1:n
    for j=1:n
        sij = s(i,j);
        sig = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-
s(pbc(i-1,n),j)) + delta(sij-s(i,pbc(j-1,n)));
        entest = entest + (4-sig)/2;
    end
end
de = (energy-entest)/entest;

% Displaying the results

subplot(2,2,1)
imagesc(so);axis equal;axis off;title('initial','fontsize',18);
colormap('default');

subplot(2,2,2)
imagesc(s1),axis equal;axis off;title('1/3 of the run','fontsize',18);

subplot(2,2,3)
imagesc(s);axis equal;axis off;title('final','fontsize',18);

subplot(2,2,4)
plot(en);xlabel('MCS','fontsize',18);ylabel('E','fontsize',18);
```

1.3. Results and discussion

For answering this question, four different values of Q (2, 4, 6, 8) and five different numbers of Monte Carlo Steps (MCS: 10, 50, 100, 150, 200) were considered. As illustrated in Tables 6 to 8, the effect of various MCS values on the system configuration for different Q values is shown. As the number of time steps increases, the system tends to reduce its energy by minimizing the interfaces between grains. Eventually, at sufficiently large MCS, each lattice with a specific Q evolves into a nearly spherical grain (one grain). The system initially starts with Q possible spin orientations, and over time, grains grow to reduce the system's total energy thermodynamically.

- ❖ How does the system behave as a function of Q?

With increasing Q, the number of possible grain orientations increases. Therefore, the system requires more MCS to reach a stable state, as evident from Tables 6 to 8.

- ❖ How large should Q be to get results that look like normal grain growth (i.e., many grains at the end of the run)?

It depends on the time at which the microstructure is observed. For instance, after 50 time steps, a system with Q = 4 appears to exhibit normal grain growth. However, higher Q values may require more MCS to reach a similar appearance. This behavior is visible in Tables 6 to 8.

- ❖ How does the model behave at large MCS?

At large MCS values, the system's energy continues to decrease, leading to the elimination of grain boundaries. Eventually, each distinct Q value corresponds to a nearly spherical grain, as shown in Figure 7.

- ❖ Does the final microstructure look “normal”?

In my opinion, the final microstructure appears normal after a sufficient number of MCS for a given Q — for instance, a system with Q = 6 after 100 time steps, or Q = 8 after 150 time steps.

Table 5

Effect of different numbers of Monte Carlo Steps on the initial configuration, the configuration at one-third of the run, the final configuration, and the energy diagram at each Monte Carlo Step for Q = 2.

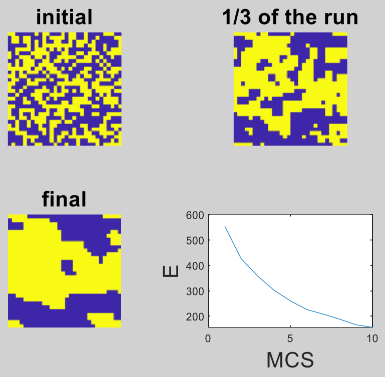
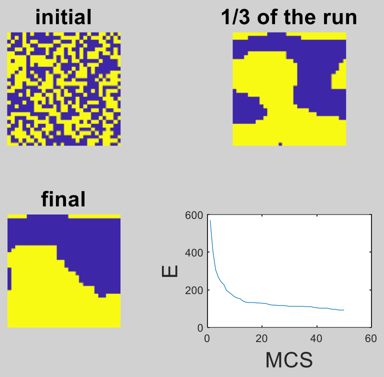
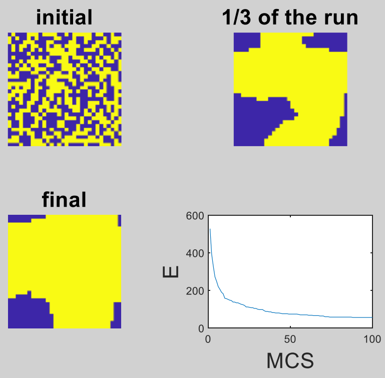
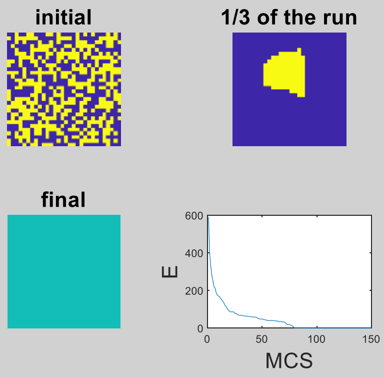
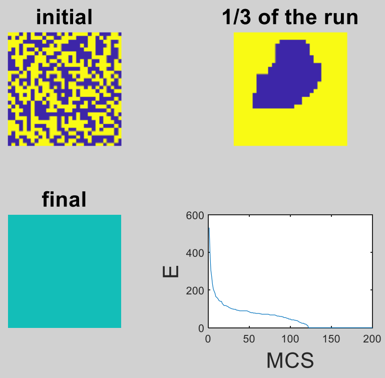
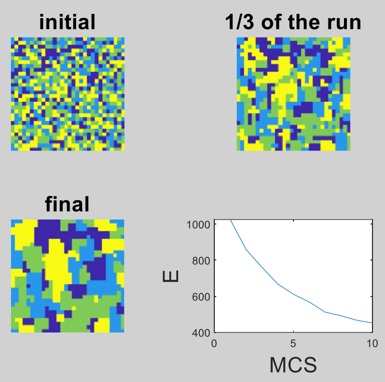
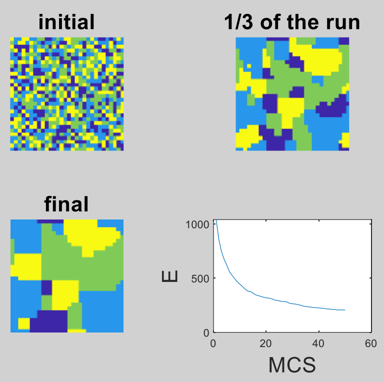
 Nstep=10	 Nstep=50
 Nstep=100	 Nstep=150
 Nstep=200	

Table 6

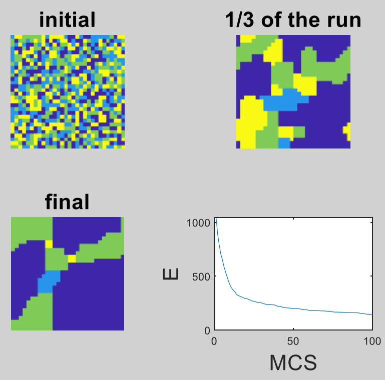
Effect of different numbers of Monte Carlo Steps on the initial configuration, the configuration at one-third of the run, the final configuration, and the energy diagram at each Monte Carlo Step for Q = 4.



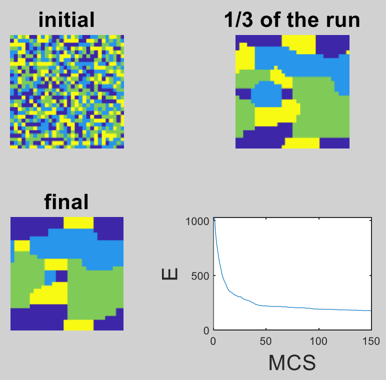
Nstep=10



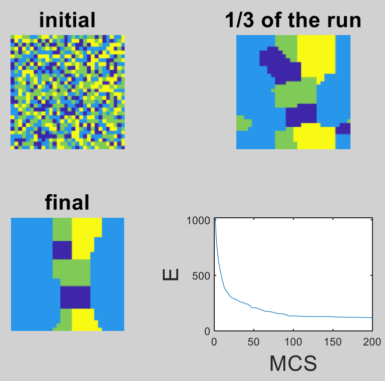
Nstep=50



Nstep=100



Nstep=150



Nstep=200

Table 7

Effect of different numbers of Monte Carlo Steps on the initial configuration, the configuration at one-third of the run, the final configuration, and the energy diagram at each Monte Carlo Step for Q = 6.

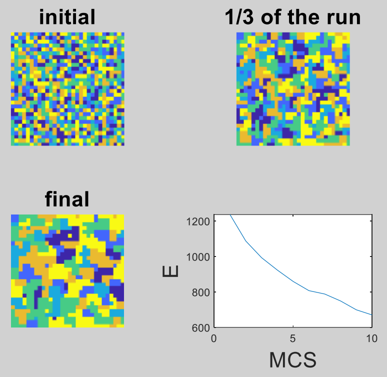
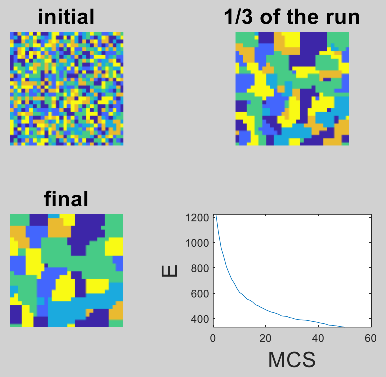
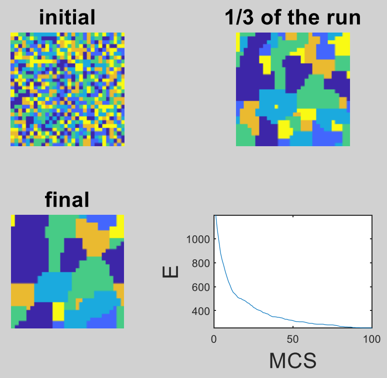
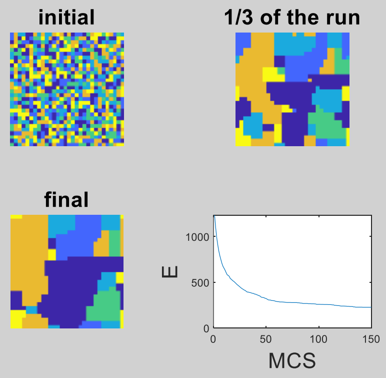
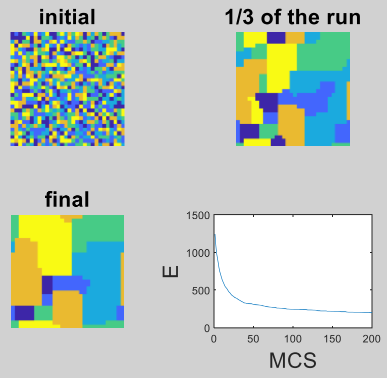
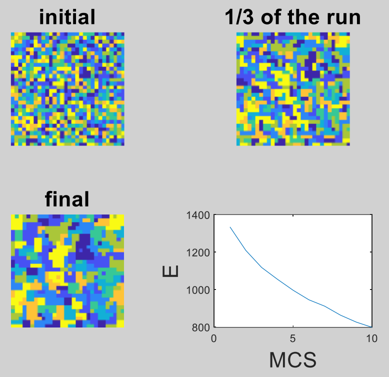
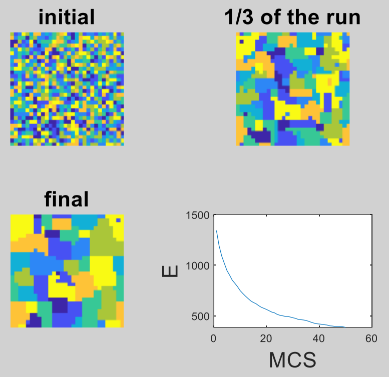
 Nstep=10	 Nstep=50
 Nstep=100	 Nstep=150
 Nstep=200	

Table 8

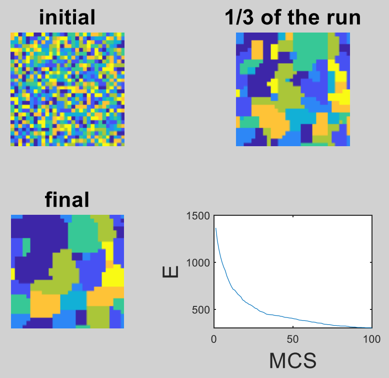
Effect of different numbers of Monte Carlo Steps on the initial configuration, the configuration at one-third of the run, the final configuration, and the energy diagram at each Monte Carlo Step for Q = 8.



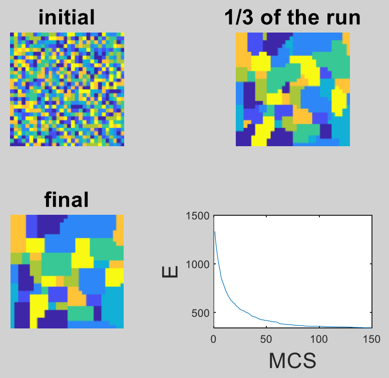
Nstep=10



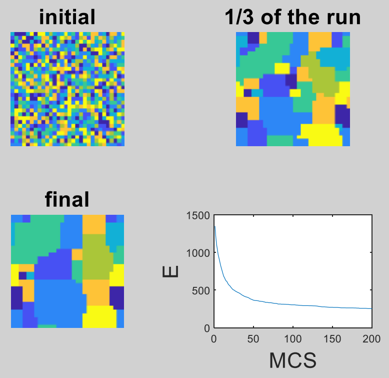
Nstep=50



Nstep=100



Nstep=150



Nstep=200

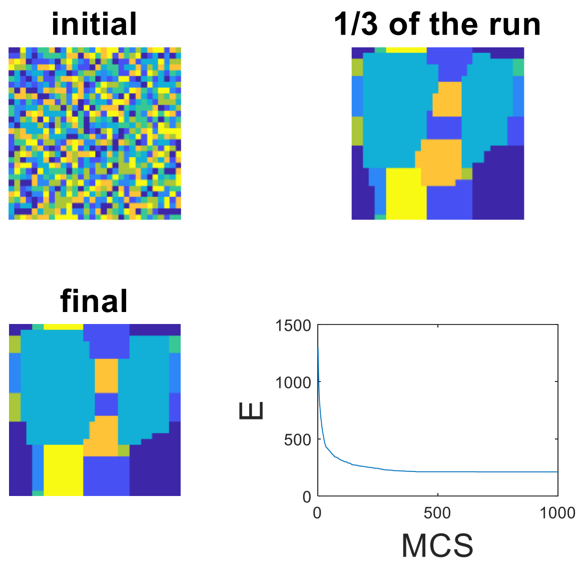


Fig. 7. Initial configuration, configuration at one-third of the run, final configuration, and energy per MCS diagram for Q = 8 over 1000 Monte Carlo Steps.

Exercise 2.

2.1. Assumptions

- ❖ Size of the cell (n) = 30 (minimum of question data)
- ❖ Number of Monte Carlo Steps (nstep) = 10 – 50 – 100 – 150 – 200 – 250
- ❖ The range of possible spin values (Q) = 8

2.2. MATLAB code

This section incorporates next-nearest neighbors into the interaction energy, which requires changing the neighborhood type from Von Neumann to Moore. The differences between these two major neighborhood types are illustrated in Figure 8.

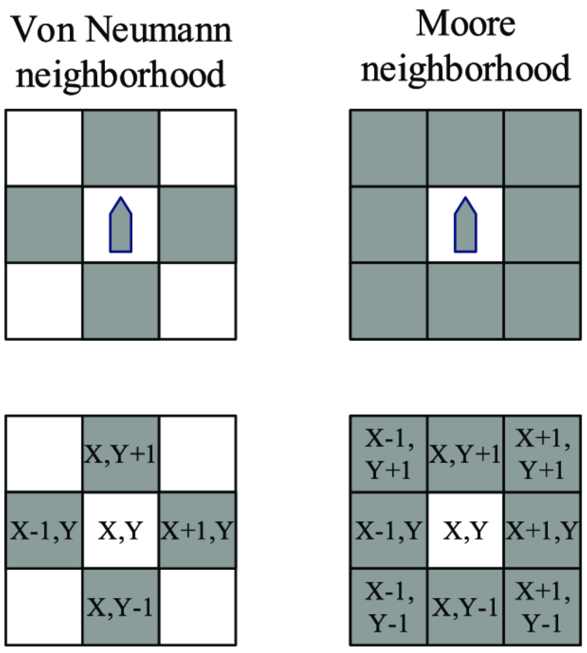


Fig. 8. Working mechanism of Von Neumann and Moore neighborhood.

In order to achieve this goal, the code was modified (lines 39, 61, 62, and 89), with the following changes:

- ❖ Line 39:

```
sig = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-s(pbc(i-
1,n),j)) + delta(sij-s(i,pbc(j-1,n))) + delta(sij-s(pbc(i+1,n),pbc(j+1,n))) +
delta(sij-s(pbc(i-1,n),pbc(j+1,n))) + delta(sij-s(pbc(i-1,n),pbc(j-1,n))) +
delta(sij-s(pbc(i+1,n),pbc(j-1,n)));
```

- ❖ Line 61:

```
sign = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-s(pbc(i-
1,n),j)) + delta(sij-s(i,pbc(j-1,n))) + delta(sij-s(pbc(i+1,n),pbc(j+1,n))) +
delta(sij-s(pbc(i-1,n),pbc(j+1,n))) + delta(sij-s(pbc(i-1,n),pbc(j-1,n))) +
delta(sij-s(pbc(i+1,n),pbc(j-1,n)));
```

❖ Line 62:

```
sigo = delta(sijo-s(pbc(i+1,n),j)) + delta(sijo-s(i,pbc(j+1,n))) + delta(sijo-
s(pbc(i-1,n),j)) + delta(sijo-s(i,pbc(j-1,n))) + delta(sijo-s(pbc(i+1,n),pbc(j+1,n)))
+ delta(sijo-s(pbc(i-1,n),pbc(j+1,n))) + delta(sijo-s(pbc(i-1,n),pbc(j-1,n))) +
delta(sijo-s(pbc(i+1,n),pbc(j-1,n)));
```

❖ Line 89:

```
sig = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-s(pbc(i-
1,n),j)) + delta(sij-s(i,pbc(j-1,n))) + delta(sij-s(pbc(i+1,n),pbc(j+1,n))) +
delta(sij-s(pbc(i-1,n),pbc(j+1,n))) + delta(sij-s(pbc(i-1,n),pbc(j-1,n))) +
delta(sij-s(pbc(i+1,n),pbc(j-1,n)));
```

❖ Line 40:

```
energy = energy + (8-sig)/2;
```

❖ Line 90:

```
entest = entest + (8-sig)/2;
```

Final code:

```
%  Monte Carlo for 2D Potts Model
%            Coded by
%       Roozbeh Aghabarari
%
%           Contact Me
%     ro.aghabarari@gmail.com
%    www.roozbehaghabarari.com
%
%  input:
%       n = size of cell.  total number of sites = n^2
%       nstep = number of Monte Carlo Steps (MCS)
%       Q = the range of possible spin values (1 to Q)
%
%  output:
%       en = energy at each MCS
%       so = initial configuration
%       s1 = configuration at 1/3 of the run
%       s2 = configuration at 2/3 of the run
%       s = final configuration
%       de = %error in final energy (a test to make sure it works)
%
function[en,so,s1,s2,s,de]= MCPotts(n,nstep,q)
n = 30;
nstep= 100;
q = 8;
```

```
% create initial lattice of spins
N=n*n;
s = ceil(q.*rand(n));
% save initial structure
so = s;
np = floor(nstep/3);
%inline Kronecker delta
delta = inline('n==0');
% calculate initial energy and polarization
energy = 0;
for i=1:n
    for j=1:n
        sij = s(i,j);
        sig = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-
s(pbc(i-1,n),j)) + delta(sij-s(i,pbc(j-1,n))) + delta(sij-s(pbc(i+1,n),pbc(j+1,n))) +
delta(sij-s(pbc(i-1,n),pbc(j+1,n))) + delta(sij-s(pbc(i-1,n),pbc(j-1,n))) +
delta(sij-s(pbc(i+1,n),pbc(j-1,n)));
        energy = energy + (8-sig)/2;
    end
end
%  initialize for averages and output
en=zeros(nstep,1);
MCS = 0;
% define periodic boundary conditions in pbc
%
% do Monte Carlo
%  number of Monte Carlo steps = nstep
%  number of configurations = N*nstep
nconfig = N*nstep;
for k=1:nconfig
%  pick a site to change
    i = ceil(rand*n);
    j = ceil(rand*n);
% flip the spin and calculate energy change
    sijo = s(i,j);
    sij = sijo;
    sij = sij + ceil((q-1)*rand);
    sij = sij - q*floor((sij-1)/q);
    sign = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-
s(pbc(i-1,n),j)) + delta(sij-s(i,pbc(j-1,n))) + delta(sij-s(pbc(i+1,n),pbc(j+1,n))) +
delta(sij-s(pbc(i-1,n),pbc(j+1,n))) + delta(sij-s(pbc(i-1,n),pbc(j-1,n))) +
delta(sij-s(pbc(i+1,n),pbc(j-1,n)));
    sigo = delta(sijo-s(pbc(i+1,n),j)) + delta(sijo-s(i,pbc(j+1,n))) + delta(sijo-
s(pbc(i-1,n),j)) + delta(sijo-s(i,pbc(j-1,n))) + delta(sijo-s(pbc(i+1,n),pbc(j+1,n)))
+ delta(sijo-s(pbc(i-1,n),pbc(j+1,n))) + delta(sijo-s(pbc(i-1,n),pbc(j-1,n))) +
delta(sijo-s(pbc(i+1,n),pbc(j-1,n)));
    del = -(sign - sigo);
% Metropolis algorithm:  only change state if accepted
% if rejected, state stays the same but is added to list
% of configurations
    if del <= 0
        s(i,j) = sij;
        energy = energy + del;
    end
%  every MCS add energy and magnetization to list
```

```
    if mod(k,N)==0
        MCS = MCS + 1;
        en(MCS) = energy;
        if MCS == np
            s1 = s;
        end
        if MCS == 2*np
            s2 = s;
        end
    end
end

% test the final energy to make sure no errors in updating
entest = 0;
for i=1:n
    for j=1:n
        sij = s(i,j);
        sig = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-
s(pbc(i-1,n),j)) + delta(sij-s(i,pbc(j-1,n))) + delta(sij-s(pbc(i+1,n),pbc(j+1,n))) +
delta(sij-s(pbc(i-1,n),pbc(j+1,n))) + delta(sij-s(pbc(i-1,n),pbc(j-1,n))) +
delta(sij-s(pbc(i+1,n),pbc(j-1,n)));
        entest = entest + (8-sig)/2;
    end
end
de = (energy-entest)/entest;

% Displaying the results

subplot(2,2,1)
imagesc(so);axis equal;axis off;title('initial','fontsize',18);
colormap('default');

subplot(2,2,2)
imagesc(s1),axis equal;axis off;title('1/3 of the run','fontsize',18);

subplot(2,2,3)
imagesc(s);axis equal;axis off;title('final','fontsize',18);

subplot(2,2,4)
plot(en);xlabel('MCS','fontsize',18);ylabel('E','fontsize',18);
```

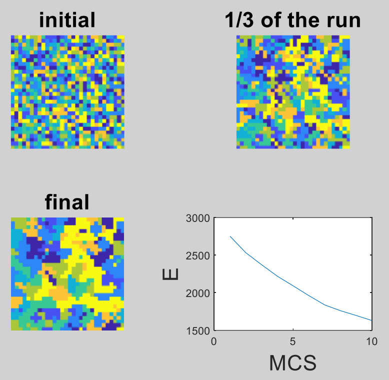
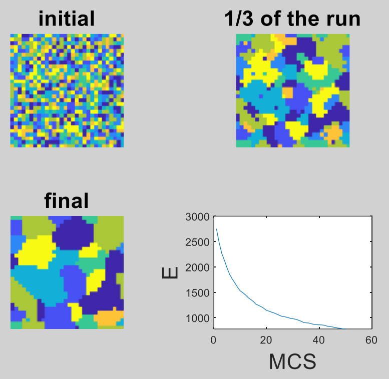
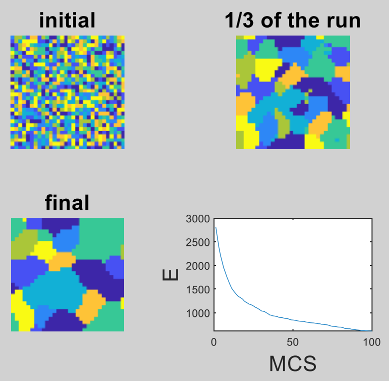
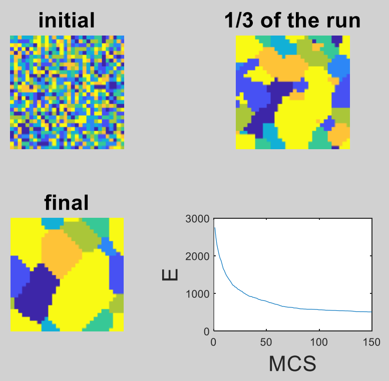
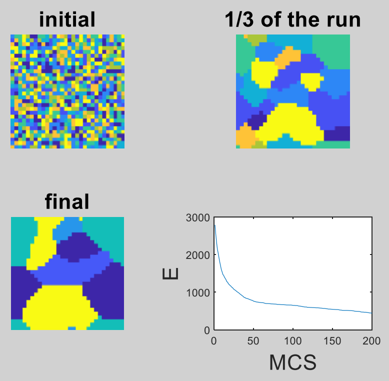
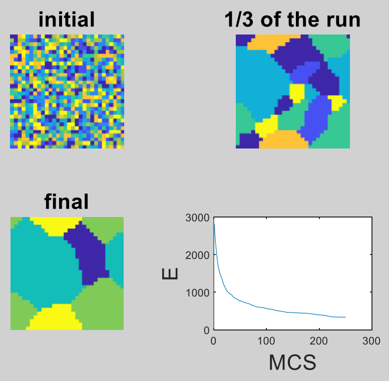
2.3. Results and discussion

As shown in Table 9, and in comparison with Exercise 1 (Table 8), the simulation using the Moore neighborhood more closely resembles real conditions due to the more spherical grain shapes and smoother, more accurate interfaces. With the Moore neighborhood, grains tend to evolve into nearly spherical shapes by the end of the process, which corresponds to the thermodynamic state of lowest energy.

In terms of energy, the simulation with the Moore neighborhood stabilizes at a higher energy level compared to the one using the Von Neumann neighborhood. This is expected, as the Moore neighborhood accounts for more interactions, leading to a more comprehensive energy calculation.

Table 9

Effect of different Monte Carlo Steps on the initial configuration, configuration at one-third of the run, final configuration, and energy per MCS diagram for Q = 8.

 Nstep=10	 Nstep=50
 Nstep=100	 Nstep=150
 Nstep=200	 Nstep=250

Exercise 3.

3.1. Assumptions

- ❖ Size of the cell (n) = 30 (minimum of question data)
- ❖ Number of Monte Carlo Steps (nstep) = 100
- ❖ The range of possible spin values (Q) = 4
- ❖ Stored energy (Es) = 0 – 200 – 400 – 600

3.2. MATLAB code

`Es*sig` has been added to lines 42 and 92.

- ❖ Von Neumann neighborhood

```
%  Monte Carlo for 2D Potts Model
%            Coded by
%       Roozbeh Aghabarari
%
%           Contact Me
%     ro.aghabarari@gmail.com
%    www.roozbehaghabarari.com
%
%  input:
%       n = size of cell.  total number of sites = n^2
%       nstep = number of Monte Carlo Steps (MCS)
%       Q = the range of possible spin values (1 to Q)
%       Es= stored energy
%
%  output:
%       en = energy at each MCS
%       so = initial configuration
%       s1 = configuration at 1/3 of the run
%       s2 = configuration at 2/3 of the run
%       s = final configuration
%       de = %error in final energy (a test to make sure it works)
%
function[en,so,s1,s2,s,de]= MCPotts(n,nstep,q,Es)
n = 30;
nstep= 100;
q = 4;
Es=0;
% create initial lattice of spins
N=n*n;
s = ceil(q.*rand(n));
% save initial structure
so = s;
np = floor(nstep/3);
%inline Kronecker delta
delta = inline('n==0');
% calculate initial energy and polarization
energy = 0;
for i=1:n
```

```
    for j=1:n
        sij = s(i,j);
        sig = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-
s(pbc(i-1,n),j)) + delta(sij-s(i,pbc(j-1,n)));
        energy = energy + (4-sig)/2 + Es*sig;
    end
end
%  initialize for averages and output
en=zeros(nstep,1);
MCS = 0;
% define periodic boundary conditions in pbc
%
% do Monte Carlo
%  number of Monte Carlo steps = nstep
%  number of configurations = N*nstep
nconfig = N*nstep;
for k=1:nconfig
%  pick a site to change
    i = ceil(rand*n);
    j = ceil(rand*n);
% flip the spin and calculate energy change
    sijo = s(i,j);
    sij = sijo;
    sij = sij + ceil((q-1)*rand);
    sij = sij - q*floor((sij-1)/q);
    sign = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-
s(pbc(i-1,n),j)) + delta(sij-s(i,pbc(j-1,n)));
    sigo = delta(sijo-s(pbc(i+1,n),j)) + delta(sijo-s(i,pbc(j+1,n))) + delta(sijo-
s(pbc(i-1,n),j)) + delta(sijo-s(i,pbc(j-1,n)));
    del = -(sign - sigo);
% Metropolis algorithm:  only change state if accepted
% if rejected, state stays the same but is added to list
% of configurations
    if del <= 0
        s(i,j) = sij;
        energy = energy + del;
    end
%  every MCS add energy and magnetization to list
    if mod(k,N)==0
        MCS = MCS + 1;
        en(MCS) = energy;
        if MCS == np
            s1 = s;
        end
        if MCS == 2*np
            s2 = s;
        end
    end
end

% test the final energy to make sure no errors in updating
entest = 0;
for i=1:n
    for j=1:n
        sij = s(i,j);
        sig = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-
s(pbc(i-1,n),j)) + delta(sij-s(i,pbc(j-1,n)));
        entest = entest + (4-sig)/2 + Es*sig;
    end
end
de = (energy-entest)/entest;

% Displaying the results

subplot(2,2,1)
imagesc(so);axis equal;axis off;title('initial','fontsize',18);
colormap('default');

subplot(2,2,2)
imagesc(s1),axis equal;axis off;title('1/3 of the run','fontsize',18);

subplot(2,2,3)
imagesc(s);axis equal;axis off;title('final','fontsize',18);

subplot(2,2,4)
plot(en);xlabel('MCS','fontsize',18);ylabel('E','fontsize',18);
```

- Moore neighborhood

```
%  Monte Carlo for 2D Potts model
%  input:
%       n = size of cell.  total number of sites = n^2
%       nstep = number of Monte Carlo Steps (MCS)
%       Q = the range of possible spin values (1 to Q)
%       Es= stored energy
%
%  output:
%       en = energy at each MCS
%       so = initial configuration
%       s1 = configuration at 1/3 of the run
%       s2 = configuration at 2/3 of the run
%       s = final configuration
%       de = %error in final energy (a test to make sure it works)
%
function[en,so,s1,s2,s,de]= MCPotts(n,nstep,q)
n = 30;
nstep= 100;
q = 4;
Es=600;
% create initial lattice of spins
N=n*n;
s = ceil(q.*rand(n));
% save initial structure
so = s;
np = floor(nstep/3);
%inline Kronecker delta
delta = inline('n==0');
% calculate initial energy and polarization
energy = 0;
```

```
for i=1:n
    for j=1:n
        sij = s(i,j);
        sig = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-
s(pbc(i-1,n),j)) + delta(sij-s(i,pbc(j-1,n))) + delta(sij-s(pbc(i+1,n),pbc(j+1,n))) +
delta(sij-s(pbc(i-1,n),pbc(j+1,n))) + delta(sij-s(pbc(i-1,n),pbc(j-1,n))) +
delta(sij-s(pbc(i+1,n),pbc(j-1,n)));
        energy = energy + (8-sig)/2 + Es*sig;
    end
end
%  initialize for averages and output
en=zeros(nstep,1);
MCS = 0;
% define periodic boundary conditions in pbc
%
% do Monte Carlo
%  number of Monte Carlo steps = nstep
%  number of configurations = N*nstep
nconfig = N*nstep;
for k=1:nconfig
%  pick a site to change
    i = ceil(rand*n);
    j = ceil(rand*n);
% flip the spin and calculate energy change
    sijo = s(i,j);
    sij = sijo;
    sij = sij + ceil((q-1)*rand);
    sij = sij - q*floor((sij-1)/q);
    sign = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-
s(pbc(i-1,n),j)) + delta(sij-s(i,pbc(j-1,n))) + delta(sij-s(pbc(i+1,n),pbc(j+1,n))) +
delta(sij-s(pbc(i-1,n),pbc(j+1,n))) + delta(sij-s(pbc(i-1,n),pbc(j-1,n))) +
delta(sij-s(pbc(i+1,n),pbc(j-1,n)));
    sigo = delta(sijo-s(pbc(i+1,n),j)) + delta(sijo-s(i,pbc(j+1,n))) + delta(sijo-
s(pbc(i-1,n),j)) + delta(sijo-s(i,pbc(j-1,n))) + delta(sijo-s(pbc(i+1,n),pbc(j+1,n)))
+ delta(sijo-s(pbc(i-1,n),pbc(j+1,n))) + delta(sijo-s(pbc(i-1,n),pbc(j-1,n))) +
delta(sijo-s(pbc(i+1,n),pbc(j-1,n)));
    del = -(sign - sigo);
% Metropolis algorithm:  only change state if accepted
% if rejected, state stays the same but is added to list
% of configurations
    if del <= 0
        s(i,j) = sij;
        energy = energy + del;
    end
%  every MCS add energy and magnetization to list
    if mod(k,N)==0
        MCS = MCS + 1;
        en(MCS) = energy;
        if MCS == np
            s1 = s;
        end
        if MCS == 2*np
            s2 = s;
        end
    end
```

```
end

% test the final energy to make sure no errors in updating
entest = 0;
for i=1:n
    for j=1:n
        sij = s(i,j);
        sig = delta(sij-s(pbc(i+1,n),j)) + delta(sij-s(i,pbc(j+1,n))) + delta(sij-
s(pbc(i-1,n),j)) + delta(sij-s(i,pbc(j-1,n))) + delta(sij-s(pbc(i+1,n),pbc(j+1,n))) +
delta(sij-s(pbc(i-1,n),pbc(j+1,n))) + delta(sij-s(pbc(i-1,n),pbc(j-1,n))) +
delta(sij-s(pbc(i+1,n),pbc(j-1,n)));
        entest = entest + (8-sig)/2 + Es*sig;
    end
end
de = (energy-entest)/entest;

% Displaying the results

subplot(2,2,1)
imagesc(so);axis equal;axis off;title('initial','fontsize',18);
colormap('default');

subplot(2,2,2)
imagesc(s1),axis equal;axis off;title('1/3 of the run','fontsize',18);

subplot(2,2,3)
imagesc(s);axis equal;axis off;title('final','fontsize',18);

subplot(2,2,4)
plot(en);xlabel('MCS','fontsize',18);ylabel('E','fontsize',18);
```

3.3. Results and discussion

Table 10

Effect of various stored energy values on the initial configuration, configuration at one-third of the run, final configuration, and the energy diagram for each MSC (Von Neumann neighborhood).

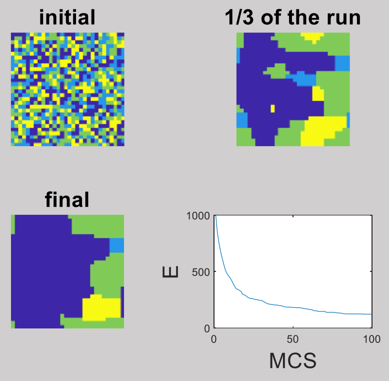
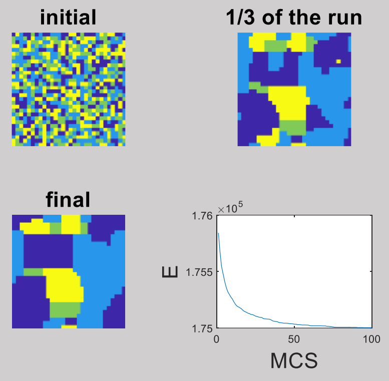
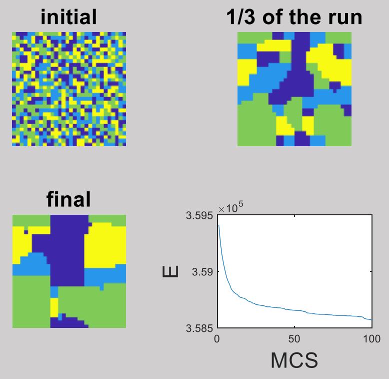
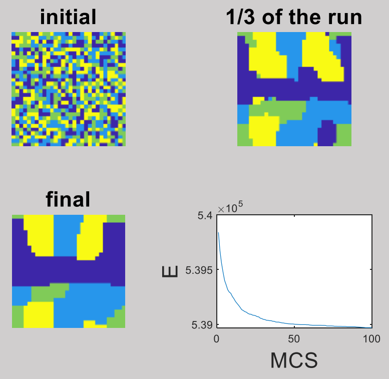
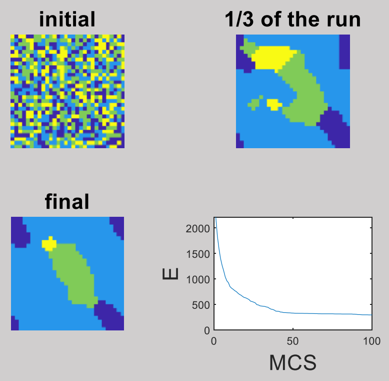
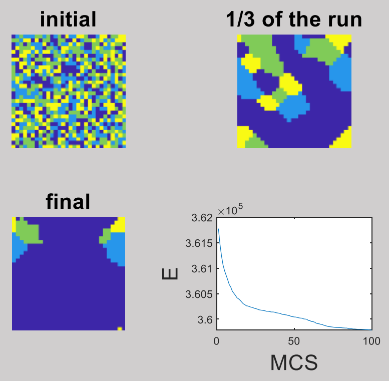
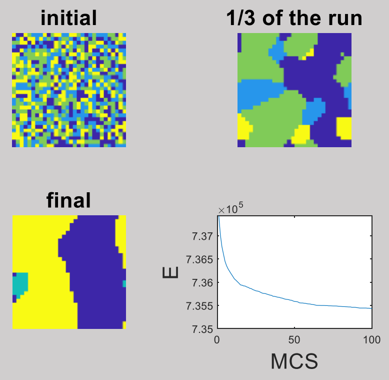
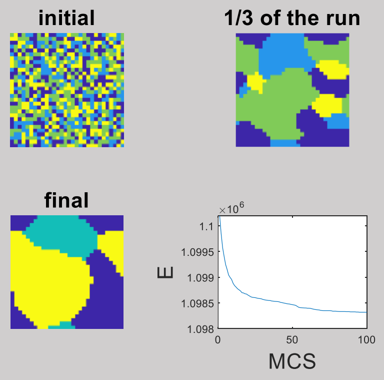
 Es=0	 Es= 200
 Es=400	 Es=600

Table 11

Effect of various stored energy values (Es) on the initial configuration, configuration at one-third of the run, final configuration, and the energy diagram for each MSC (Moore neighborhood).

 Es=0	 Es= 200
 Es=400	 Es=600

❖ How large does Es have to be to see recrystallization-like behavior? Explain your results.

According to Tables 10 and 11, Es must be at least 600 for recrystallization to occur. Of course, this value may vary with different Q.

❖ Compare results for the code with nearest neighbor interactions to that with next nearest neighbor interactions.

In the Moore neighborhood, the system exhibited better recrystallization and is more likely to return to its real condition.