2D Ising project

Jiawei Zang

December 10, 2020

1.Metropolis algorithm:

$$W(v \to \sigma) = 1 \quad for \quad P_\sigma \geq P_v \quad (E_\sigma \leq E_v) \tag{1}$$

$$W(v \to \sigma) = \frac{P_\sigma}{P_v} \quad for \quad P_\sigma < P_v \quad (E_\sigma > E_v) \tag{2}$$

We want to verify that the Metropolis algorithm satisfies detailed balance : $P_\sigma W(\sigma \to v) = P_v W(v \to \sigma)$
Consider the initial and final configurations i and f:
(1) if $P_i < P_f$, then $w(i \to f) = 1, \quad w(f \to i) = P_i/P_f$
therefore, $P_i \cdot w(i \to f) = P_i = P_f \cdot w(f \to i)$
(1) if $P_i >= P_f$, then $w(i \to f) = P_f/P_i, \quad w(f \to i) = 1$
therefore, $P_i \cdot w(i \to f) = P_i \cdot \frac{P_f}{p_i} = P_f = P_f \cdot w(f \to i)$
Overall, it satisfy the detailed balance.
2. Write a program to simulate the two-dimensional Ising model. (See attached codes)
3. Make a plot of the average energy, E, as a function of T for the four system sizes. Comment on what you observe for $2 \leq T \leq 3$.

$$E = \sum -\frac{J}{2} s_{i,j} \left(s_{i+1,j} + s_{i-1,j} + s_{i,j+1} + s_{i,j-1} \right) \tag{3}$$

From Onsager's exact solution, there is a phase transition at $T_c = 2.269$ (set $J = k_B = 1$). In the simulation result, we can see an instant change of energy at $2 \leq T \leq 3$, which denotes the transition. As we increase the lattice size, as expected, curves slightly shift left, closer to the transition temperature. The energy is continuous, so it is a second order phase transition.
At low temperatures, all spins align with each other, so the energy per site is $-\frac{J}{2} \cdot 4 = -2J$.

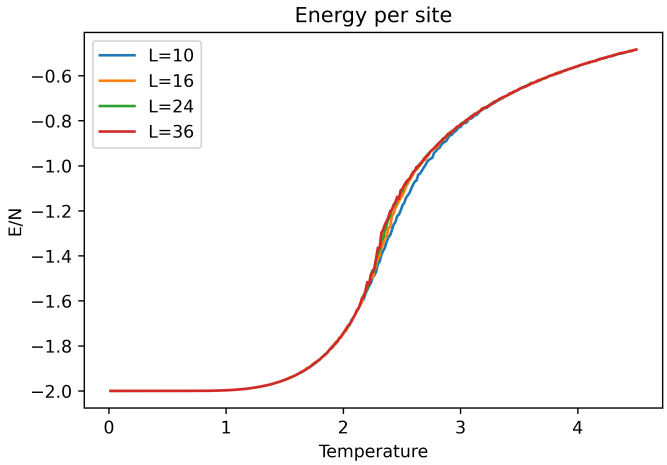


Figure 1: Energy per site

4-5. Plot the susceptibility and discussed finite size scaling.

We need to consider the finite size scaling when the lattice size is not much bigger than the correlation length ξ, which will change the thermodynamic quantities $A_L(t)$. Using Renormalization group, it can be derived that:

$$A_L(t) = L^{\gamma/\nu} f\left(tL^{1/\nu}\right) \tag{4}$$

$$t = \frac{T - T_{\mathrm{c}}}{T_{\mathrm{c}}} \tag{5}$$

For finite L, $A_L(t)$ will have a maximum at $f'(x_0) = 0$, corresponding to $t_{\mathrm{c}}(L) = x_0 L^{-1/v}$.
When $L \to \infty$, the position of the maximum approaches $t_c = 0$ (or $T = T_{\mathrm{c}}$), and the peak becomes higher and narrower. From the numerical result of susceptibility in Fig. 2, we can see: as lattice size increases, the position of the peak is closer to T_c, and the peak becomes higher and narrower.

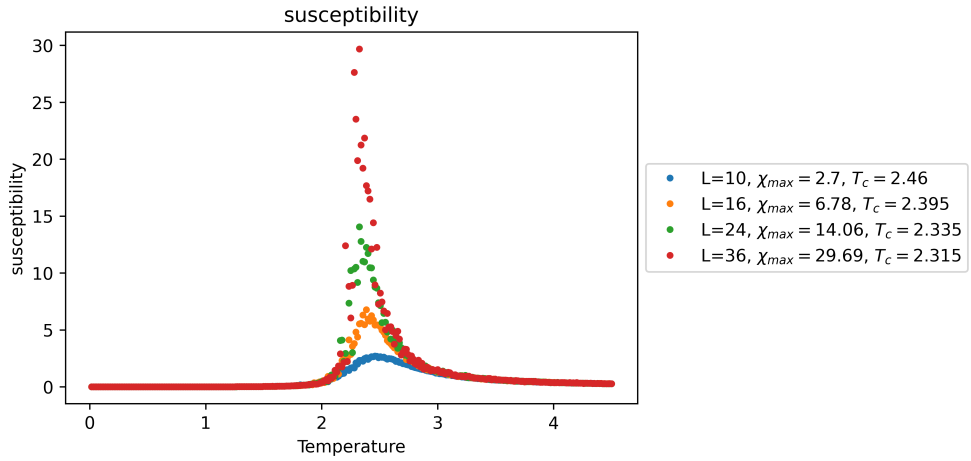


Figure 2: Susceptibility

Combining $t_c(L) = x_0 L^{-1/v}$ and $t = \frac{T-T_c}{T_c}$, we get:

$$T_c(L) = T_c + (x_0 T_c)\, L^{-1/v} \tag{6}$$

We can verify eq. 6 by plotting $T_c(L)$ versus L^{-1}, assuming $v = 1$, as shown in Fig. 3. The fitting result gives $T_c = 2.256$, which is very close to 2.269.

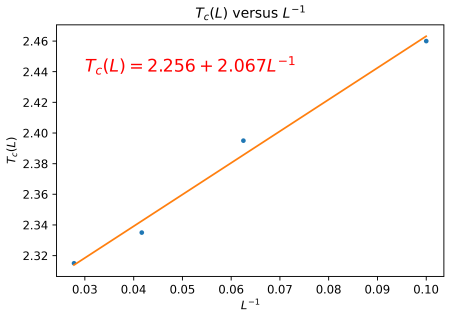


Figure 3: $T_c(L)$ versus L^{-1}

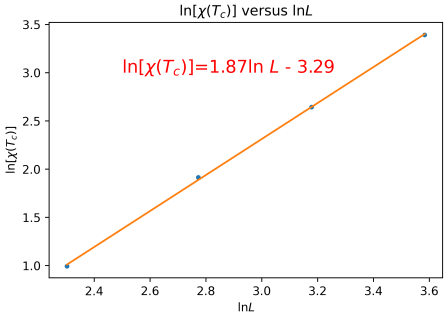


Figure 4: $\ln[\chi(T_c)]$ versus In L

Based on eq. 4 and 6:

$$A_L(T) \sim L^{\gamma/\nu} f\left(L^{1/\nu}\left(T - T_c(L)\right)\right) \tag{7}$$

Then at $T = T_c$, we have

$$A_L\left(T = T_{\mathrm{c}}(L)\right) \sim L^{\gamma/\nu} f(0) \rightarrow lnA_L\left(T_{\mathrm{c}}(L)\right) = \frac{\gamma}{\nu} lnL + lnf(0) \qquad (8)$$

This tells us that the scaling of physical quantities measured at Tc(L) is described by the bulk critical exponents. In Fig. 4, we verify that, and found $\frac{\gamma}{\nu} = 1.87$, close to the exact value $\gamma/v = 1.75$. If we do simulation on more sizes, a more accurate result can be got.

Eq. 7 also denotes that, if we plot $L^{-\gamma/\nu} A_L(T)$ as a function of the scaling variable $L^{1/\nu}\left(T - T_{\mathrm{c}}(L)\right)$, the data from the simulation of systems of different sizes will collapse on the same curve in the vicinity of the critical temperature, as verified in Fig. 5.

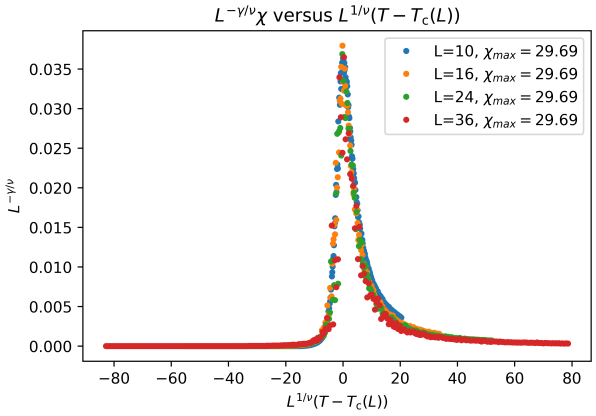


Figure 5: Susceptibility after renormalization

6. Plot the magnetization, M, versus T and observe the finite size effects near the transition.

Fig. 6 plots magnetization per site. At low temperatures, all spins align with each other, resulting $|M| = 1$. Around the transition, there is a clear finite size effect: as we increase the lattice size, the curve between $2 < T < 3$ becomes sharper, and the final magnetization is closer to 0.

We can verify eq. 8 again by plotting $L^{\beta/\nu} M$ as a function of T, using $\beta/v = 0.25$, as shown in Fig. 7. From eq. 4, we know $L^{\beta/\nu} M = f\left(tL^{1/\nu}\right)$,

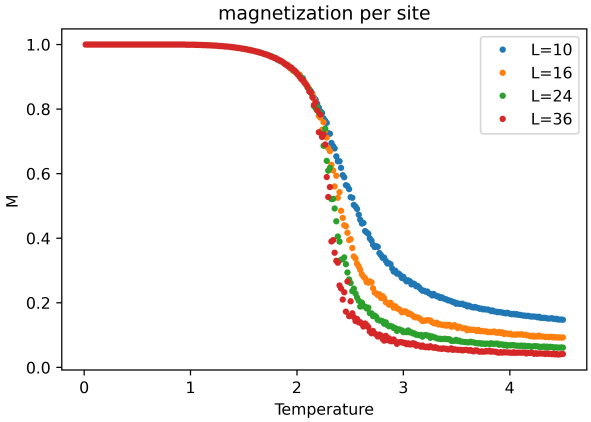


Figure 6: M

then at $T = T_c$, where t=0, the curves from different lattice sizes will all intersect at one point. This offers another way to use finite size scaling to determine T_c. We get $T_C\prime = 2.32$ from the simulation results, close to 2.269.

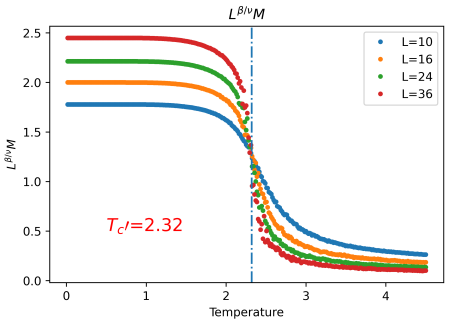


Figure 7: $L^{\beta/\nu}M$ versus T

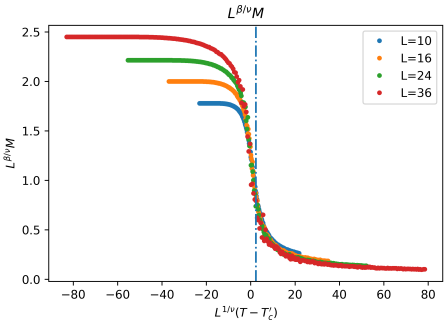


Figure 8: $L^{\beta/\nu}M$ versus $L^{1/\nu}\left(T - T_C'\right)$

Then we can plot $L^{\beta/\upsilon}M$ versus the scaling variable $L^{1/\nu}\left(T - T_{\mathrm{c}}'\right)$, shown in Fig. 8. The magnetization curves collapse onto a single curve in the critical region (the region around Tc).

7. Plot C versus T for the systems simulated and observe the finite size effects near the transition.

This part is very similar as the discussion about susceptibility. Specific

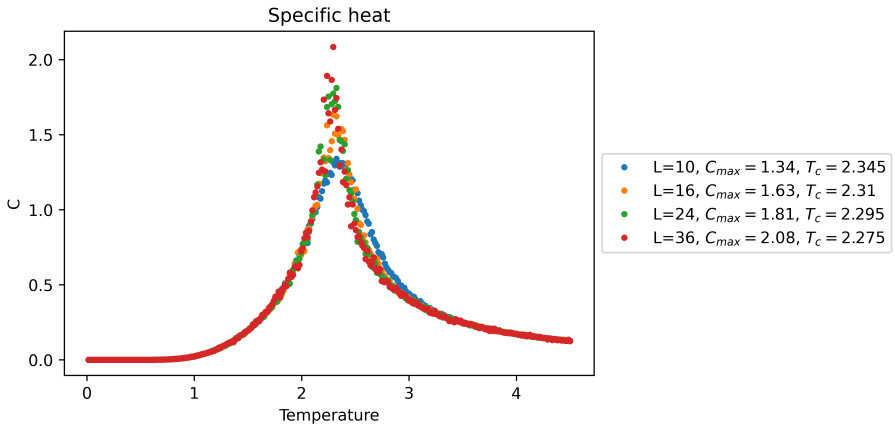


Figure 9: Specific heat

heat result also shows that, as lattice size increases, the position of the peak is closer to Tc , and the peak becomes higher and narrower.

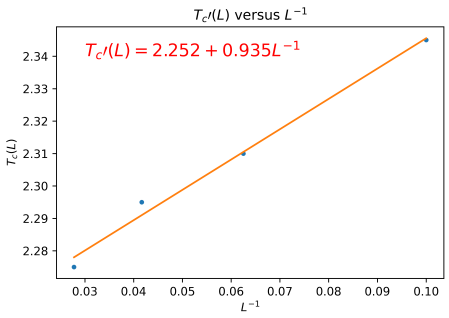


Figure 10: $T_c\prime(L)$ versus L^{-1}

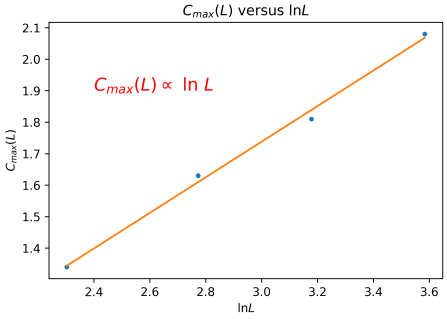


Figure 11: $C_{\max}(L)$ versus In L

Then we plot $T_c'(L)$ versus $L^{-1/\nu}$, and find $T_c\prime = 2.252$, nearly the same as $T_c = 2.256$, which is found the in the susceptibility plot. Because the specific heat exponent of 2D Ising model is $\alpha = 0$. This means that the specific heat does not diverge as a power but rather as a logarithm, as verified in Fig. 11.

8. Calculate and plot $S(T)$ versus T, and plot the free energy.

Our numerical result shows $S(T = 4.5)$ 0.632, slightly smaller than $S(T \rightarrow$

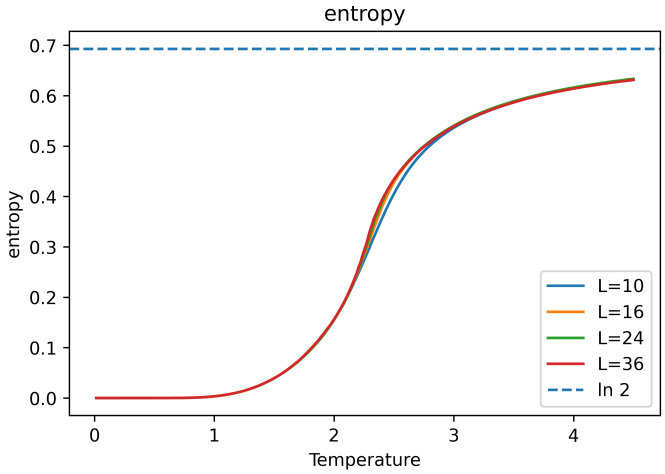


Figure 12: Entropy

$\infty) = \ln 2$. This is expected. On the one hand, $S(T) = \int_0^T \mathrm{d}T' \frac{C(T')}{T'}$. Specific heat is not exactly zero when $T > 4.5$, so S(4) should be smaller than S($T \to \infty$). On the other hand, the system has finite size effects, which will make the result deviate from the accurate value slightly.

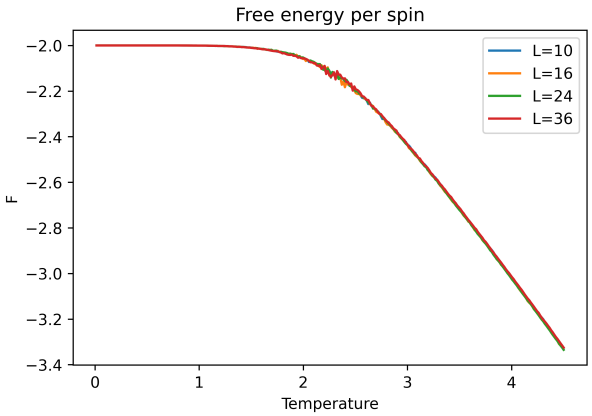


Figure 13: Free energy

Above picture shows the evolution of free energy as temperature increases. At low temperature, all spins align with each other, S=0, and

F=E-TS=E, which is -2J per site. As temperature increases, free energy goes down. This is expected. Imagine initially all spins are up. If we flip one spin, $\Delta E = 4J$ and the corresponding entropy $\Delta S = k_b(ln\Omega' - ln\Omega) \approx -2lnN^2$. $\Delta F \approx 4J - 2TlnN^2$. As we flip more spins, both the energy and entropy will further increase, but as long as N is big enough and T is not too small, ΔE will be smaller to $T\Delta S$, therefore, the free energy will decrease.

Reference:

Le Bellac, Michel, et al. Equilibrium and non-equilibrium statistical thermodynamics. Cambridge University Press, 2004.

In [1]:
```
import numpy as np
import random
from matplotlib import pyplot as plt
import numba
from numba import njit
```

Consider the 2D L*L-spin Ising model, in which each site is associated with a variable $s_{i,j} = 1, -1$. The energy is

$$E = \sum_{i=0,,,L-1;j=0...L-1} \left(-\frac{J}{2} \left(s_{i,j} \left(s_{i+1,j} + s_{i-1,j} \right) + s_{i,j} \left(s_{i,j+1} + s_{i,j-1} \right) \right) \right) \quad (1)$$

It has periodic boundary conditions: $s_{i,j} = s_{L+i,j}$

Based on Monte Caro method, the main procedure is as follows:

1. Create a L * L lattice, and start with a given spin configuration s_p and energy E_p
2. Select a site at random and propose to flip the spin on that site (i.e. change the sign of the variable on that site).
3. Compute the new energy E_{new}, and calculate the energy difference $\Delta E = E_{new} - E_p$.
4. Accept or reject the proposal with the detailed balance probability: a) if $\Delta E <= 0$, accept the proposal. b) if $\Delta E > 0$, accept the proposal with probability $P = e^{-\frac{\Delta E}{k_B T}}$, and reject proposal with probability 1-P.
5. Return either the original state or the new state according to whether the proposal is accepted.
6. Go back to step 2, repeat the procedure n_max times until it reaches equilibrium.

- set $k_B = 0$
- we need to consider boundary conditions: we can use the trick: i% N. For example, 0%N=0, (N+2)%N=2

In [3]:
```
"""
Step 1. Create an initial 2D L*L-spin lattice with all spins up.

"""
def initial_lattice(L=10):
    return np.ones((L,L))
```

In [4]:
```
"""
Step 2. Select a site at random and propose to flip the spin on that site.

s: the list of the old spin configuration.
s_prop: the list of the proposed spin configuration.
i: a random site chosen to be flipped

"""
@njit
def propose(s,L):
    i=np.random.randint(L)
    j=np.random.randint(L)

    s_prop = s.copy()
    s_prop[i,j] = -s[i,j]
    return s_prop, i, j
```

In [5]:
```
"""
Step 3&4&5. Calculate the energy difference. Accept or reject the proposal wi
            Return either the original state or the new state according to wh

To make the calculation faster, we only consider the energy parts that are re
This will be used in thermalizing step, which doesn't record E and M.
"""
@njit
def update1(s_prop,s,i,j,L,J=1,T=1):


    e_ij= -J*s[i,j]*(s[(i+1)%L,j] +s[(i-1)%L,j] + s[i,(j-1)%L]+ s[i,(j+1)%L])
    # the energy related to the spin_ij in the old system

    dE =  -2*e_ij # dE= E_new-E_old = (-e_i)-e_i=-2e_i

    if dE <=0:
        return s_prop
    else:
        random_num = random.random()
        if random_num < np.exp(-dE/T):
            return s_prop
        else:
            return s
```

In [15]:

```
"""
Step 3&4&5. Calculate the energy difference. Accept or reject the proposal wi
            Return either the original state or the new state according to wh

To make the calculation faster, we only consider the energy parts that are re

This will be used in measuring step, which records E and M.
"""
@njit

def update2(s_prop,s,E,M,i,j,L,J=1,T=1):


    e_ij= -J*s[i,j]*(s[(i+1)%L,j] +s[(i-1)%L,j] + s[i,(j-1)%L]+ s[i,(j+1)%L])
    # the energy related to the spin_ij in the old system

    dE =  -2*e_ij # dE= E_new-E_old = (-e_i)-e_i=-2e_i

    if dE <=0:
        M=M-2*s[i,j]
        E=E+dE
        return s_prop,E,M
    else:
        random_num = random.random()
        if random_num < np.exp(-dE/T):
            M=M-2*s[i,j]
            E=E+dE
            return s_prop,E,M
        else:
            return s,E,M
```

In [14]:

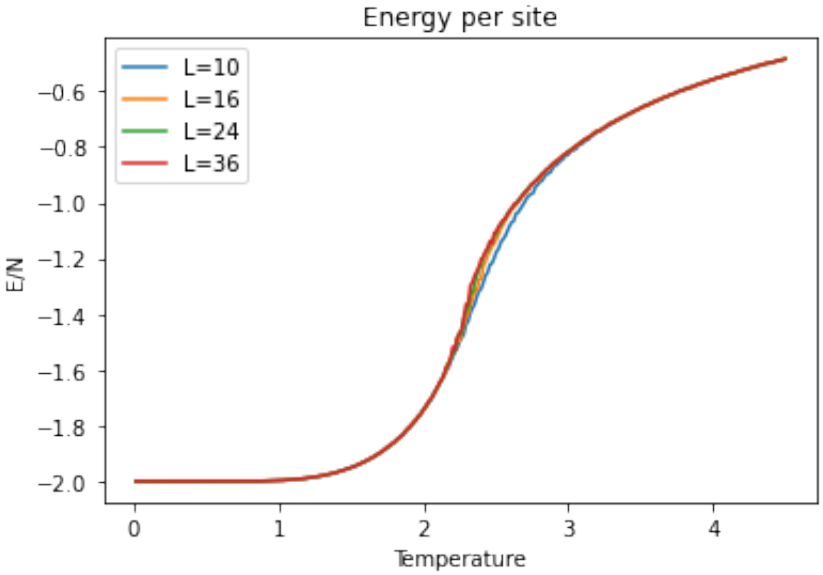
```
"""
Based on the two functions "propose" and "update", construct a function that 
a list of length n_max ofstates that are visited, and record and magnetizatio

s_start: the list of the initial spin configuration.
n_thermal: the number of sweeps to thermalize the system
n_mea: the number of sweeps during which we perform the measurements
m: We perform a measurement every m sweeps.
"""
@njit
def generate(s_start, T=1, L=10, J=1, n_thermal=10, n_mea=100, m=10):

    s = s_start
    #thermalzing process
    for n in range(n_thermal):
        s_prop, i,j = propose(s,L)
        s_new = update1(s_prop,s,i,j,L,J,T)
        s = s_new

    #measuring process
    E = 0
    for i in range(L):
        for j in range(L):
            E += -J/2*s[i,j]*(s[(i+1) % L,j] + s[(i-1) % L,j]+s[i,(j+1) % L] 
    M=np.sum(s)
    Slist=[]
    Elist=[]
    for n in range(n_mea):
        s_prop, i,j = propose(s,L)
        s_new,E,M = update2(s_prop,s,E,M,i,j,L,J,T)
        if n%m==0:
            Slist.append(M)
            Elist.append(E)

        s = s_new

    Slist=np.array(Slist)
    S2list=Slist**2
    averageS=np.sum(np.abs(Slist))/len(Slist)
    averageS2=np.sum(S2list)/len(S2list)

    Elist=np.array(Elist)
    E2list=Elist**2

    averageE=np.sum(Elist)/len(Elist)
    averageE2=np.sum(E2list)/len(E2list)


    return averageS,averageS2,averageE,averageE2
```

In [114…
```
n_thermal = 1000000
n_mea= 6000000
m=10
B=0
J=1

Tlist=np.arange(0.015,4.5,0.015)

Llist=[10,16,24,36]
length=int(n_mea/m)
```

In [ ]:
```
for L in Llist:
    L_S=[]
    L_S2=[]
    L_E=[]
    L_E2=[]
    s_start=initial_lattice(L)


    for T in Tlist:
        print(T,L)
        S,S2,E,E2=generate(s_start, T, L, J, n_thermal, n_mea, m)
        L_S.append(S)
        L_S2.append(S2)
        L_E.append(E)
        L_E2.append(E2)

    np.save(f'L={L}_S.npy', L_S)
    np.save(f'L={L}_S2.npy', L_S2)
    np.save(f'L={L}_E.npy', L_E)
    np.save(f'L={L}_E2.npy', L_E2)
```

In [ ]:
```

```

In [414…
```
#3
for L in Llist:
    E=np.load(f'L={L}_E.npy')/L**2
    plt.plot(Tlist,E, label = f"L={L}")

plt.legend()
plt.ylabel(r'E/N')
plt.xlabel(r'Temperature')
plt.title('Energy per site')
plt.legend()
plt.savefig('3_energy.png',dpi=600, bbox_inches='tight')
plt.show()
```

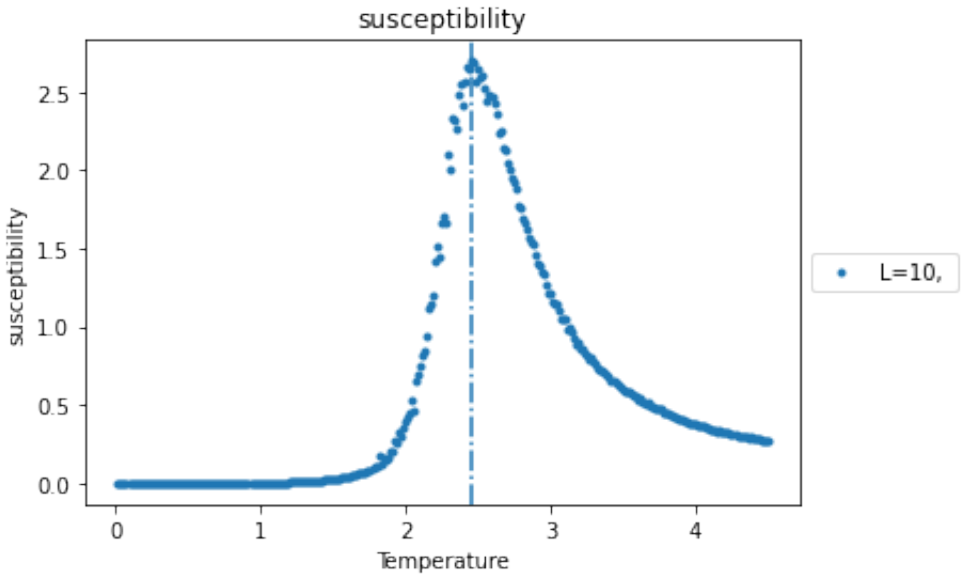


In [430…

```
#4 determing Tc

for L in [10]:
    S=np.load(f'L={L}_S.npy')
    S2=np.load(f'L={L}_S2.npy')
    sus=(S2-S**2)/Tlist/L**2
    maxs=np.around(max(sus),2)

    plt.plot(Tlist,sus,'.',label = f"L={L}, ")
plt.title('susceptibility')
plt.ylabel(r'susceptibility')
plt.xlabel(r'Temperature')
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.axvline(x=2.46,linestyle ='-.')
```

Out[430…

```
<matplotlib.lines.Line2D at 0x187cce510>
```



In [427…
```
sus=list(sus)
sus.index(max(sus))
```

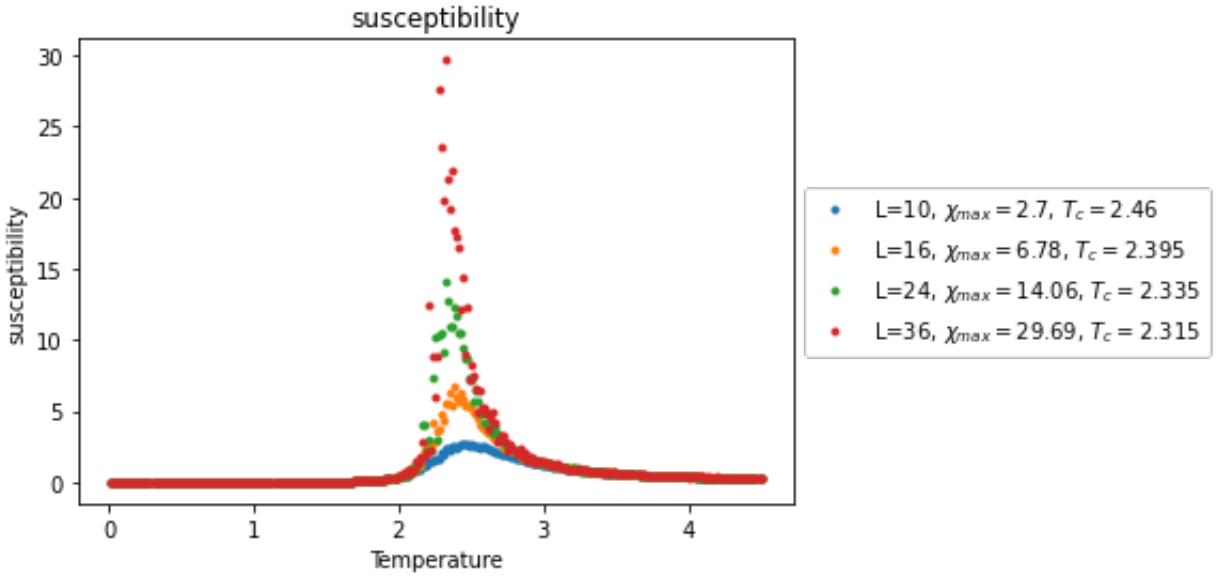
Out[427… 163

In [429…
```
Tlist[163]
```

Out[429… 2.46

In [354…
```
Tc_from_sus=[2.46,2.395,2.335,2.315]
```

In [415…
```
sus_Tc=[]
for i,L in enumerate(Llist):
    S=np.load(f'L={L}_S.npy')
    S2=np.load(f'L={L}_S2.npy')
    sus=(S2-S**2)/Tlist/L**2
    maxs=np.around(max(sus),2)
    sus_Tc.append(maxs)
    plt.plot(Tlist,sus,'.',label = f"L={L}, "+r"$\chi_{max}=$"+f"{maxs}, "+r"
plt.title('susceptibility')
plt.ylabel(r'susceptibility')
plt.xlabel(r'Temperature')
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.savefig('4_sus.png',dpi=600, bbox_inches='tight')
```

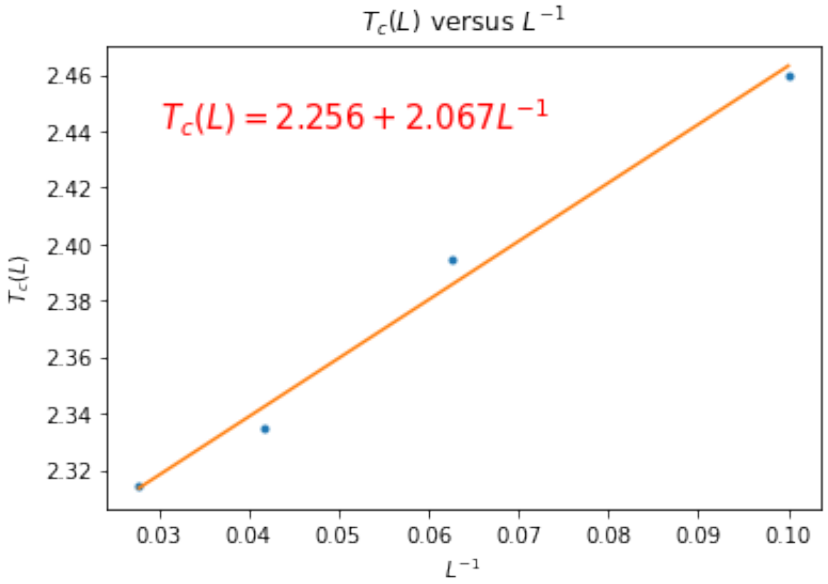


In [ ]:
```
Llist=np.array(Llist)
```

In [416…

```
#4
plt.plot(1/Llist,Tc_sus,'.')
plt.title(r'$T_c(L)$ versus $L^{-1}$')
plt.ylabel(r'$T_c(L)$')
plt.xlabel(r'$L^{-1}$')
a, b = np.polyfit(1/Llist, Tc_sus, 1)
plt.plot(1/Llist, a*1/Llist + b)
print(a,b)
plt.text(0.03,2.44, r'$T_c(L) =2.256+ 2.067 L^{-1}$',fontsize=15,color='red')
plt.savefig('4_2_Tc_L_s.png',dpi=600, bbox_inches='tight')
plt.show()
```

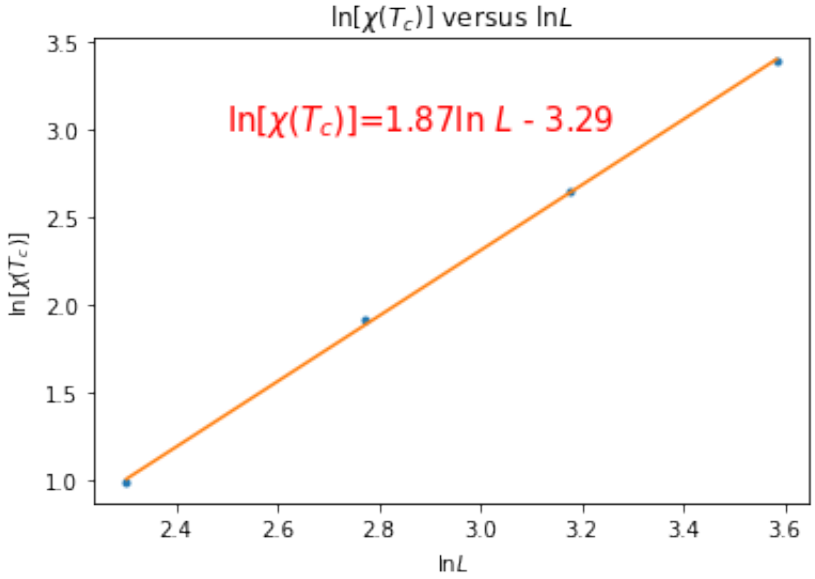
```
2.0667642752562223 2.2564063770945184
```



In [424…

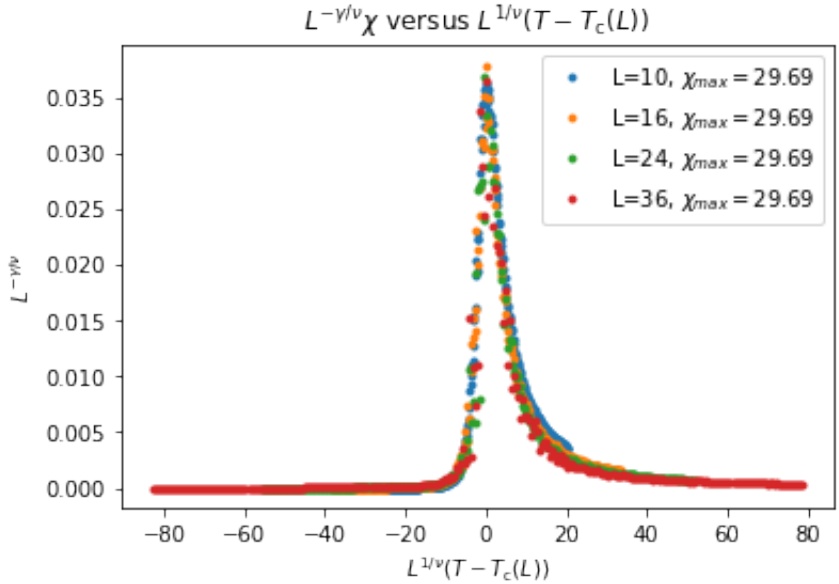
```
#5
lg_sus=np.log(sus_Tc)
lg_L=np.log(Llist)
plt.plot(lg_L,lg_sus,'.')
plt.title(r'ln[$\chi(T_c)$] versus ln$L$')
plt.ylabel(r'ln[$\chi(T_c)$]')
plt.xlabel(r'ln$L$')
a, b = np.polyfit(lg_L, lg_sus, 1)
plt.plot(lg_L, a*lg_L + b)
print(a,b)
plt.text(2.5,3, r'ln[$\chi(T_c)$]=1.87ln $L$ - 3.29',fontsize=15,color='red')
plt.savefig('5_1_lg_L_s.png',dpi=600, bbox_inches='tight')
plt.show()
```

```
1.8660309996568691 -3.286590752694114
```



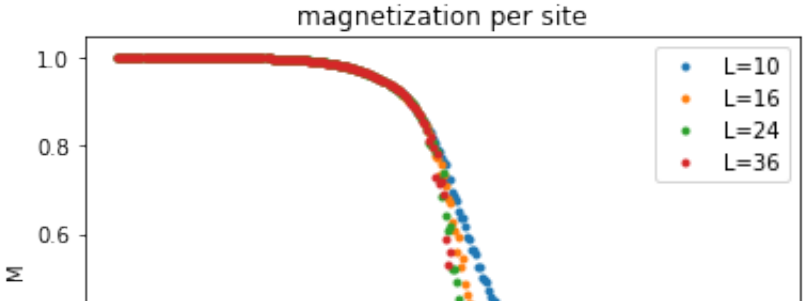
In [418…

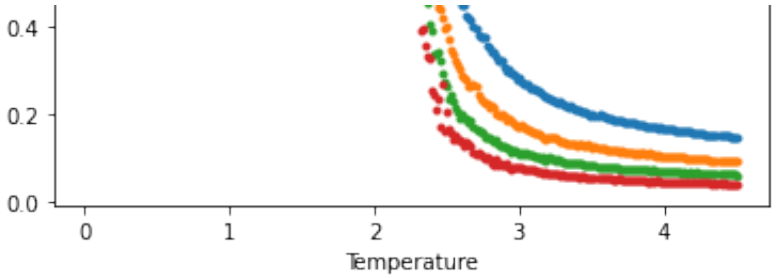
```
#5
for i,L in enumerate(Llist):
    S=np.load(f'L={L}_S.npy')
    S2=np.load(f'L={L}_S2.npy')
    sus=(S2-S**2)/Tlist/L**2
    sus=L**(-1.87)*sus
    T_new=(Tlist-Tc_sus[i])*L
    plt.plot(T_new,sus,'.',label = f"L={L}, "+r"$\chi_{max}=$"+f"{maxs}")
plt.title(r'$L^{-\gamma / \nu} \chi$ versus $L^{1 / \nu}\left(T-T_{\mathrm{c}}
plt.ylabel(r'$L^{-\gamma / \nu}$')
plt.xlabel(r'$L^{1 / \nu}\left(T-T_{\mathrm{c}}(L)\right)$')
plt.legend()
plt.savefig('5_2_norm_s.png',dpi=600, bbox_inches='tight')
```

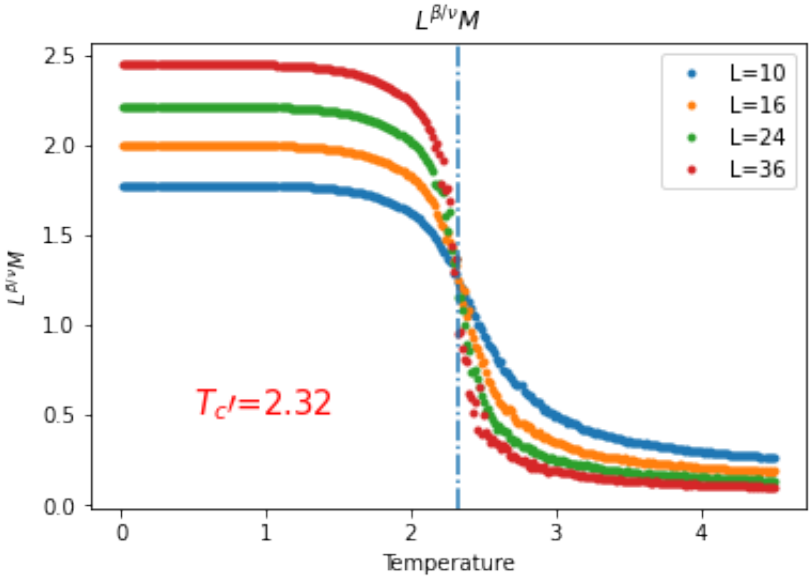


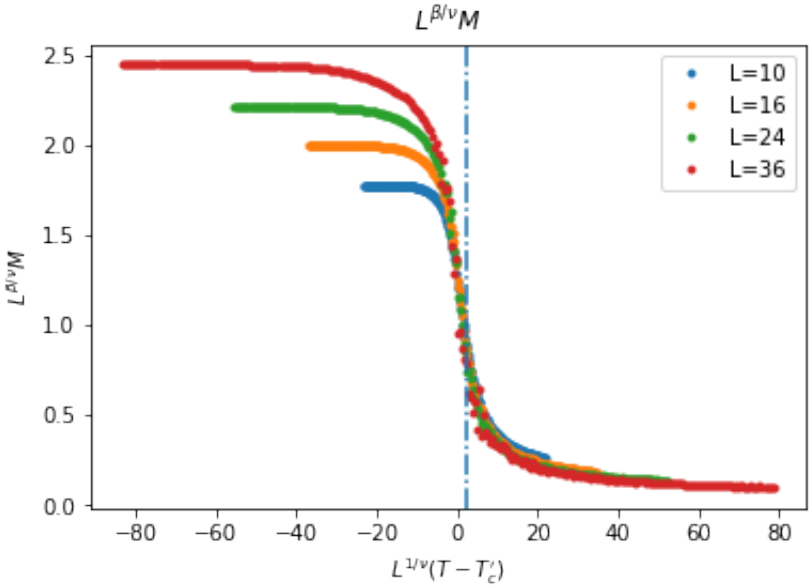
In [419…

```
#6
plt.figure()
for L in Llist:
    S=np.load(f'L={L}_S.npy')
    plt.plot(Tlist,S/L**2,'.',label = f"L={L}")
plt.legend()
plt.ylabel(r'M')
plt.xlabel(r'Temperature')
plt.title('magnetization per site')
plt.legend()
plt.savefig('6_1_M.png',dpi=600, bbox_inches='tight')
plt.show()

plt.figure()
for L in Llist:
    S=np.load(f'L={L}_S.npy')
    M_new=S/L**2*L**0.25
    plt.plot(Tlist,M_new,'.',label = f"L={L}")
plt.legend()
plt.ylabel(r'$L^{\beta / \nu} M$')
plt.xlabel(r'Temperature')
plt.title(r'$L^{\beta / \nu} M$')
plt.legend()
plt.axvline(x=2.32,linestyle ='-.')
plt.text(0.5,0.5, r'$T_c\prime$=2.32',fontsize=15,color='red')
plt.savefig('6_2_L_M.png',dpi=600, bbox_inches='tight')
plt.show()

plt.figure()

for L in Llist:
    S=np.load(f'L={L}_S.npy')
    M_new=S/L**2*L**0.25
    newT=L*(Tlist-2.32)
    plt.plot(newT,M_new,'.',label = f"L={L}")
plt.legend()
plt.ylabel(r'$L^{\beta / \nu} M$')
plt.xlabel(r'$L^{1/\nu}(T-T_c^{\prime})$')
plt.title(r'$L^{\beta / \nu} M$')
plt.legend()
plt.axvline(x=2.32,linestyle ='-.')
plt.savefig('6_3_norm_M.png',dpi=600, bbox_inches='tight')
plt.show()
```



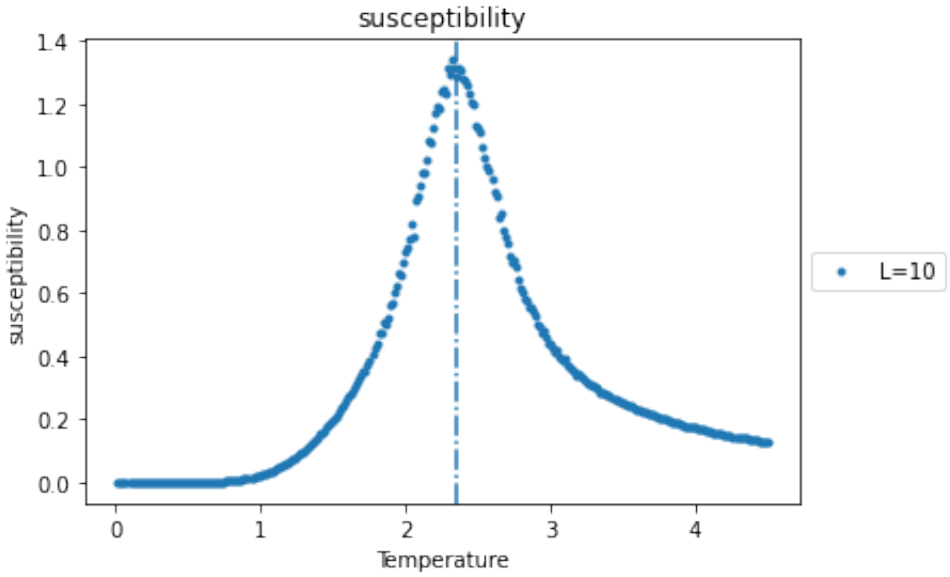

0.4
0.2
0.0
0
1
2
3
4
Temperature


$L^{\beta/\nu}M$
L=10
L=16
L=24
L=36
T_c=2.32
$L^{\beta/\nu}M$
Temperature


$L^{\beta/\nu}M$
L=10
L=16
L=24
L=36
$L^{\beta/\nu}M$
$L^{1/\nu}(T-T_c')$

In [438…
```
#7 determing Tc

for L in [10]:
    E=np.load(f'L={L}_E.npy')
    E2=np.load(f'L={L}_E2.npy')
    SH=(E2-E**2)/L**2/Tlist**2
    maxheat=np.around(max(SH),2)
    heat_Tc.append(maxheat)
    plt.plot(Tlist,SH,'.',label = f"L={L}")
plt.title('susceptibility')
plt.ylabel(r'susceptibility')
plt.xlabel(r'Temperature')
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.axvline(x=2.345,linestyle ='-.')
```

Out[438… <matplotlib.lines.Line2D at 0x1871a8910>



In [434…
```
heat=list(SH)
heat.index(max(heat))
```

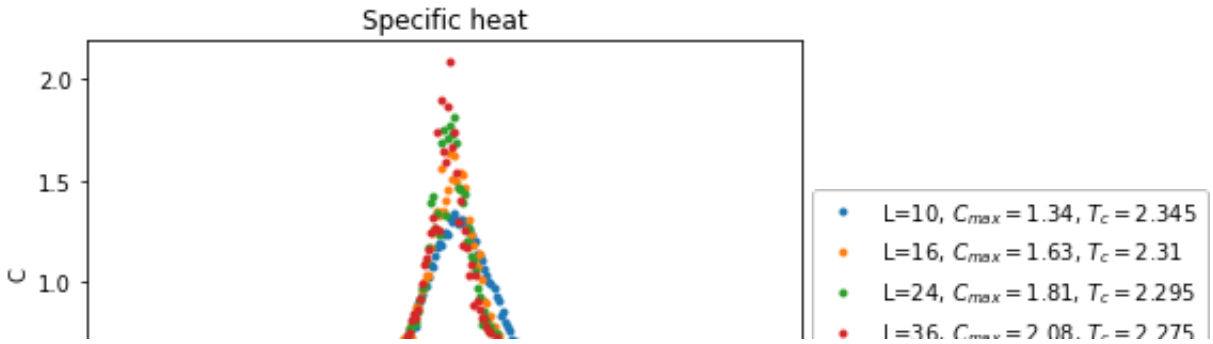
Out[434… 154

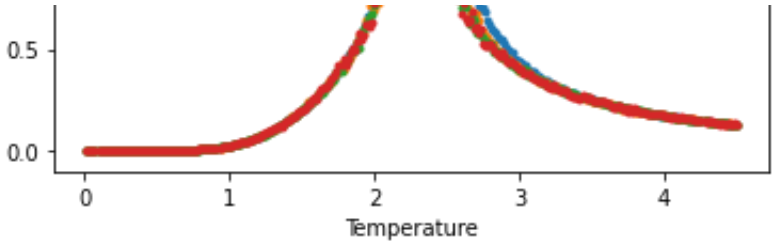
In [436…
```
Tlist[154]
```

Out[436… 2.325

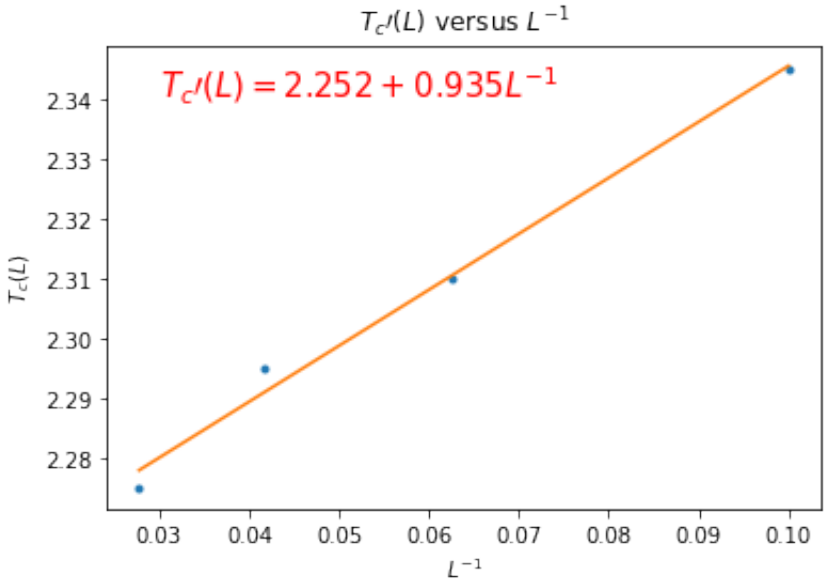
In [425…

```
#7
plt.figure()
Tc_from_heat=[2.345,2.310,2.295,2.275]
heat_Tc=[]
for i,L in enumerate(Llist):
    E=np.load(f'L={L}_E.npy')
    E2=np.load(f'L={L}_E2.npy')
    SH=(E2-E**2)/L**2/Tlist**2
    maxheat=np.around(max(SH),2)
    heat_Tc.append(maxheat)
    plt.plot(Tlist,SH,'.',label = f"L={L}, "+r"$C_{max}=$"+f"{maxheat}, "+r"$'
plt.legend()
plt.ylabel(r'C')
plt.xlabel(r'Temperature')
plt.title('Specific heat')
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.savefig('7_1_heat.png',dpi=600, bbox_inches='tight')
plt.show()

plt.figure()
plt.plot(1/Llist,Tc_from_heat,'.')
plt.title(r'$T_c\prime(L)$ versus $L^{-1}$')
plt.ylabel(r'$T_c(L)$')
plt.xlabel(r'$L^{-1}$')
a, b = np.polyfit(1/Llist, Tc_from_heat, 1)
plt.plot(1/Llist, a*1/Llist + b)
print(a,b)
plt.text(0.03,2.34, r'$T_c\prime(L) =2.252+ 0.935 L^{-1}$',fontsize=15,color=
plt.savefig('7_2_heat_Tc_L.png',dpi=600, bbox_inches='tight')
plt.show()

plt.figure()
plt.plot(lg_L,heat_Tc,'.')
plt.title(r'$C_{max}(L)$ versus ln$L$')
plt.ylabel(r'$C_{max}(L)$')
plt.xlabel(r'ln$L$')
a, b = np.polyfit(lg_L, heat_Tc, 1)
plt.plot(lg_L, a*lg_L + b)
print(a,b)
plt.text(2.4,1.9, r'$C_{max}(L)\propto$ ln $L$',fontsize=15,color='red')
plt.savefig('7_3_C_L.png',dpi=600, bbox_inches='tight')
plt.show()
```

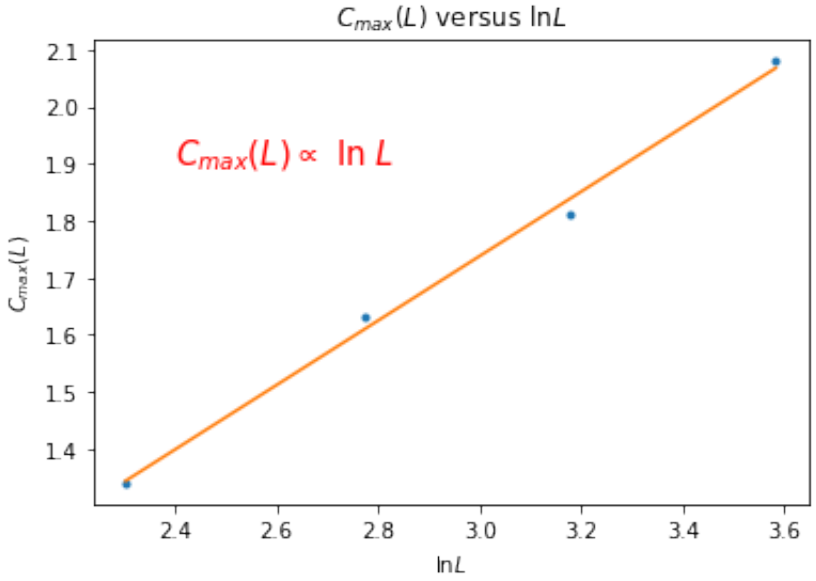





```
0.9352855051244496 2.252016430779243
```




```
0.565226398568748 0.04238958938333507
```



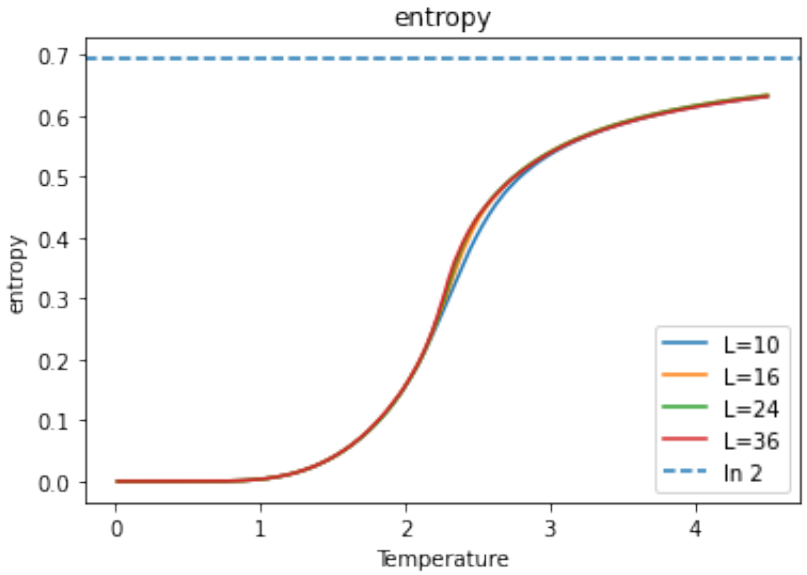
In [378…
```
#8
for L in Llist:
    E=np.load(f'L={L}_E.npy')
    E2=np.load(f'L={L}_E2.npy')
    SH=(E2-E**2)/L**2/Tlist**2
    entropy=0
    for i,C in enumerate(SH):
        entropy+=C/Tlist[i]*0.015
    print(entropy)
```

```
0.6315736292528012
0.6335293455643474
0.6334375168390579
0.6312983367124175
```

In [377…
```
np.log(2)
```

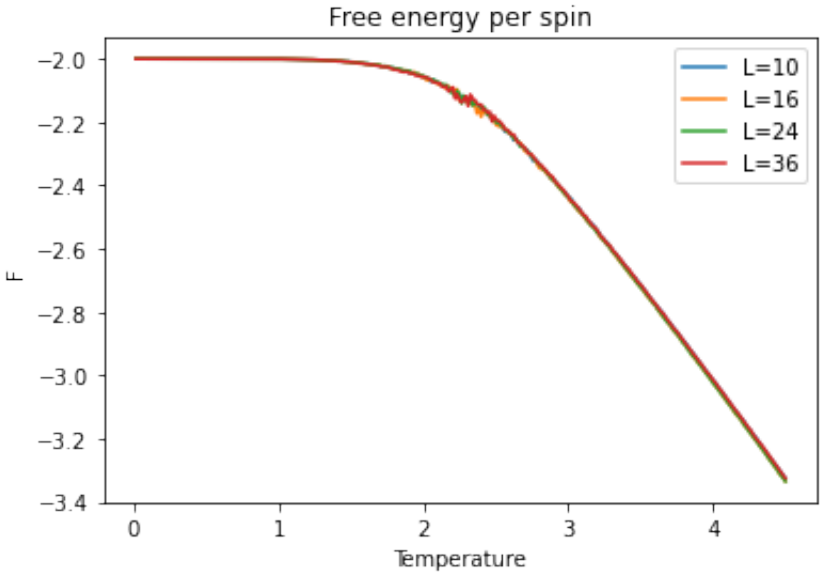
Out[377… 0.6931471805599453

In [421…
```
#8
for L in Llist:
    E=np.load(f'L={L}_E.npy')
    E2=np.load(f'L={L}_E2.npy')

    SH=(E2-E**2)/L**2/Tlist**2
    ent_list=[]
    for i,T in enumerate(Tlist):
        entropy=0
        for j in range(0,i+1):
            entropy+=SH[j]/Tlist[j]*0.015
        ent_list.append(entropy)

    plt.plot(Tlist,ent_list, label = f"L={L}")

plt.legend()
plt.ylabel(r'entropy')
plt.xlabel(r'Temperature')
plt.title('entropy')
plt.axhline(y=np.log(2), linestyle ='--',label='ln 2')
plt.legend()
plt.savefig('8_1_entropy.png',dpi=600, bbox_inches='tight')
plt.show()
```



In [422…

```
for L in Llist:
    E=np.load(f'L={L}_E.npy')
    E2=np.load(f'L={L}_E2.npy')

    SH=(E2-E**2)/L**2/Tlist**2
    ent_list=[]
    for i,T in enumerate(Tlist):
        entropy=0
        for j in range(0,i+1):
            entropy+=SH[j]/Tlist[j]*0.015
        ent_list.append(entropy)


    Free=E/L**2-Tlist*ent_list
    plt.plot(Tlist,Free, label = f"L={L}")

plt.legend()
plt.ylabel(r'F')
plt.xlabel(r'Temperature')
plt.title('Free energy per spin')

plt.legend()
plt.savefig('8_2_free.png',dpi=600, bbox_inches='tight')
plt.show()
```


Free energy per spin
L=10
L=16
L=24
L=36
F
Temperature
−2.0
−2.2
−2.4
−2.6
−2.8
−3.0
−3.2
−3.4
0
1
2
3
4