Physics 129L: Classical Simulation (part1)

2D Classical Ising model (lattice Ising model)

The classical Ising model is a mathematical model used in statistical mechanics to study phase transitions in ferromagnetic materials. It was first proposed by the physicist Ernst Ising in 1925 as a simplified representation of the magnetic behavior of certain materials, such as iron. The Ising model has since become a fundamental tool in statistical mechanics and condensed matter physics.

The Ising model is typically defined on a lattice (or grid), which is a regular arrangement of discrete points in space. In this demonstration, we will be focusing on the 2D model. The structure is given below:

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import display, clear_output
import time
```

2D Lattice generation

We can treat spins lattice as random variables, denoted as $\{S_{x,y}\}$. Let's define a function to initialize a random spin configuration. When working with a numpy array, we can naturally assign its dimensional index to be its special location, with value -1 or 1.

In [2]:
```
def initialize_spins(shape):
    spins = np.random.choice([-1, 1], size=shape)
    return spins
```

Ising Hamiltonian

Depending on the arrangement of spins, the Hamiltonian for the classical Ising model is often given by the following expression:

$$H = -J \sum_{\langle i,j \rangle} S_i S_j - B \sum_i S_i$$

where J is the coupling constant, representing the strength of the interaction between neighboring spins. $\langle i, j \rangle$ denotes a sum over pairs of nearest-neighbor spins, S_i represents the spins, and B represents an external magnetic field.

In [30]:
```
# Variables
J_value = 2  # coupling constant
B_value = 0 # external magnetic field
L_size =10    # lattice size
beta_value=0.05 # Temperature
```

In [4]:
```python
# Function to calculate the energy of the system
def net_energy(spins, J, B,print_val=False):
    energy = 0  # initial

    for i in range(spins.shape[0]):
        for j in range(spins.shape[1]):
            # Two neighboring spin coupling
            # Calculate indices of neighboring spins using modulo for periodic boundary conditions
            neighbor_down = spins[(i + 1) % spins.shape[0], j]
            neighbor_right = spins[i, (j + 1) % spins.shape[1]]

            # Interaction terms should be added to the energy
            energy -= J * spins[i, j] * (neighbor_down + neighbor_right)
            # External magnetic field
            energy -= B * spins[i, j]
            if print_val==True:


                print('*Index**********')
                print(i,j)
                print('*****Energy******')
                print(energy)
                print('**neighbor Index*********')
                print((i + 1) % spins.shape[0], j)
                print(i, (j + 1) % spins.shape[1])



    return energy

# Function to generate a random spin configuration of size L
def generate_random_spin_configuration(L):
    return np.random.choice([-1, 1], size=(L, L))
```

Example:

In [ ]:
```
spins = generate_random_spin_configuration(L_size)
print(spins)
energy = net_energy(spins,J=J_value,B=B_value)
print(energy)
fig, ax = plt.subplots()
im = ax.imshow(spins, cmap='binary', interpolation='none')
display(fig)
```

Statistical description of the Ising model

Let's look at various thermodynamic properties and statistical measures. The fundamential quatity is the partition function,

$$Z = \sum_{i,j} e^{-\beta H}$$

where $\beta = 1/k_B T$. For Ising model, we have,

$$Z = e^{-\beta\left(-J \sum_{\langle i,j \rangle} S_i S_j - B \sum_i S_i\right)}.$$

In particular, the joint distribution is a probability mass function (since we are working on a finite system), and it has a form,

$$P(S) = \frac{1}{Z} e^{-\beta H},$$

where $S = \{S_1, S_2, \cdots, S_L\}$, and each S defines a unique energy. As one can see, it is very hard to draw sample from this multi-variable distribution. It is easy to see that the total number of unique spin configurations exponentially grows with the system's dimension, 2^{L^2}, , making computations difficult for large system size $L > 20$.

In [5]:
```python
import numpy as np
from itertools import product


def generate_all_configurations(grid_size):
    # Generate all possible spin configurations
    spins = np.array(list(product([-1, 1], repeat=grid_size**2)))

    # Reshape spins to a 2D array for easier manipulation
    configurations = spins.reshape((-1, grid_size, grid_size))

    # Find unique configurations
    unique_configurations = np.unique(configurations, axis=0)

    return unique_configurations



def ising_boltzmann_weight(beta,J, B, spins):
    # J: coupling constant, B: external magnetic field, spins: 2D array representing spin configurati

    # Calculate the energy for the given spin configuration
    energy = net_energy(spins,J,B)

    # Calculate the Boltzmann weight
    weight = np.exp(-beta * energy)
#     print(energy)
#     print(weight)
    return weight,energy




def calculate_partition_function(beta, J, B, grid_size):
    # energy array
    energy_arry=np.zeros(int(2**(grid_size**2)))
    # probability array
    prob_arry=np.zeros(int(2**(grid_size**2)))
    # Generate all unique spin configurations
    unique_configurations = generate_all_configurations(grid_size)

    # Calculate Boltzmann weights for each unique configuration
    for i,config in enumerate(unique_configurations):
```

```
#           if i==6: # Example grid plot
#               fig, ax = plt.subplots()
#               im = ax.imshow(config, cmap='binary', interpolation='none')
#               display(fig)
#               print(f"The energy is:\n{energy}")

        weights,energy = ising_boltzmann_weight(beta, J, B, config)
        energy_arry[i]=energy
        prob_arry[i]=weights


    # Calculate the partition function as the sum of Boltzmann weights
    partition_function = np.sum(prob_arry)


    unique_values, index,counts = np.unique(prob_arry,return_index=True, return_counts=True)
    unique_energy=energy_arry[index]

    pdf=counts*unique_values/(partition_function)

    return partition_function,unique_energy,pdf
```

Example:

In [51]:
```
# Record the start time
start_time = time.time()
Z, energy_arry,pdf= calculate_partition_function(beta_value, J_value, B_value, L_size)
# Record the end time
end_time = time.time()

print(f"The partition function Z for the Ising model is: {Z}")
print(pdf)
print(energy_arry)


# Calculate the elapsed time
elapsed_time = end_time - start_time

# Print the elapsed time
print(f"Elapsed Time: {elapsed_time:.4f} seconds")

# fig, ax = plt.subplots(figsize=(8, 6))
# # Generate a list of colors using a colormap
# colors = plt.cm.viridis(np.linspace(0, 1, num_bars))

# # Create a bar plot
# ax.bar(energy_arry, pdf,color= colors)

# # Adding labels and title

# ax.grid(True)
# ax.set_title('Exact Boltzmann Distribution of Configurations for Ising Model')

# plt.show()
```

```
The partition function Z for the Ising model is: 43180521.331920676
[2.84790584e-06 2.54570562e-05 2.12340219e-04 1.20316844e-03
 5.84875478e-03 2.25405651e-02 6.70627521e-02 1.40572363e-01
 2.10215642e-01 2.20706644e-01 1.69722482e-01 9.53120138e-02
 4.36835874e-02 1.55709909e-02 5.08235943e-03 1.46057954e-03
 5.89837368e-04 1.03521642e-04 7.72180708e-05 6.87407908e-06]
[  60.   52.   44.   36.   28.   20.   12.    4.   -4.  -12.  -20.  -28.
  -36.  -44.  -52.  -60.  -68.  -76.  -84. -100.]
Elapsed Time: 701.4607 seconds
```

Gibbs Sampler on Ising model

Calculating the exact partition function and obtaining the exact boltzmann weight for a spin configuration becomes hard as we increase the system size. Let's consider a conditional distribution function: a variable S_i given the rest spins,

$$P(S_i|S_{\text{rest}}) = \frac{e^{\beta\left(-2JS_i\sum_{\langle j\rangle}S_j - B\sum_j S_j - BS_i\right)}}{1 + e^{\beta\left(-2JS_i\sum_{\langle j\rangle}S_j - B\sum_j S_j - BS_i\right)}}$$

A Gibbs sampling process involves drawing a sample from the distribution of that variable while fixing the values of all other variables.

In [6]:

```python
# Gibbs Update
import numpy as np

def Gibbs_sampler(spins, spin_index_x, spin_index_y, beta, J, B, grid_size):
    # Initialize energy to zero
    energy = 0

    # Two neighboring spin coupling
    # Calculate indices of neighboring spins using modulo for periodic boundary conditions
    neighbor_down = spins[(spin_index_x + 1) % grid_size, spin_index_y]
    neighbor_right = spins[spin_index_x, (spin_index_y + 1) % grid_size]

    neighbor_up = spins[(spin_index_x - 1) % grid_size, spin_index_y]
    neighbor_left = spins[spin_index_x, (spin_index_y - 1) % grid_size]


    # Interaction terms should be added to the energy
    energy -= J * spins[spin_index_x, spin_index_y] * (neighbor_down + neighbor_right+neighbor_up+nei
    # External magnetic field
    energy -= B * spins[spin_index_x, spin_index_y]

    # Calculate the weight factor using the Boltzmann factor
    weight_factor = np.exp(-2*beta * energy)

    # Calculate the conditional probability
    prob = weight_factor / (1 + weight_factor)

    # Generate a uniform random number between 0 and 1
    uniform_num = np.random.rand()

    # Update the spin value based on the conditional probability
    if uniform_num <= prob:
        new_spin_val = spins[spin_index_x, spin_index_y]
    else:
        new_spin_val = -spins[spin_index_x, spin_index_y]

    # Return the calculated energy, conditional probability, and the new spin value
    return energy, prob, new_spin_val

# spin_index_x=0
# spin_index_y=0

# # Example Spin
```

```
# spins = generate_random_spin_configuration(L_size)
# print(spins)

# energy_c,prob_c,new_spin_val_c=Gibbs_sampler(spins,spin_index_x,spin_index_y, beta=beta_value,J=J_v
# print(new_spin_val_c)
# print(prob_c)
```

Sampler Stepping

For each variable in the distribution, sample a new value from its conditional distribution given the current values of all other variables (e.g. in our case, for each spin site).

In [7]:

```
# Function to perform a Gibbs stepping
def Gibbs_step(spins_init, beta, J, B, grid_size,burnin_=13000,sample_size=5000):
    spins=spins_init
    burnin_step=0
    s=0
    sample_gibbs_energy=[]
    while burnin_step<=burnin_:
        for i in range(grid_size):
            for j in range(grid_size):
                energy_c,prob_c,new_spin_val_c=Gibbs_sampler(spins,spin_index_x=i,spin_index_y=j, bet
                spins[i,j]=new_spin_val_c
        burnin_step+=1

    while s<sample_size:
        for i in range(grid_size):
            for j in range(grid_size):
                energy_c,prob_c,new_spin_val_c=Gibbs_sampler(spins,spin_index_x=i,spin_index_y=j, bet
                spins[i,j]=new_spin_val_c

        weights,energy = ising_boltzmann_weight(beta, J, B, spins)
        sample_gibbs_energy.append(energy)
                #spins[i,j]=new_spin_val_c
        s+=1
    return sample_gibbs_energy
```

Example:

In [ ]:

```
# Record the start time
start_time = time.time()


spins = generate_random_spin_configuration(L_size)
print(spins)

energy_arry_MC=Gibbs_step(spins_init=spins,beta=beta_value,J=J_value, B=B_value,grid_size=L_size,burn

net_size=len(energy_arry_MC)
energy_arry_MC, count_MC = np.unique(energy_arry_MC, return_counts=True)

pdf_MC=count_MC/net_size

pdf_MC_full=np.zeros(len(pdf))
for i in range(0,len(energy_arry_MC)):
    indices = np.where(energy_arry_MC[i]==energy_arry)[0]
    pdf_MC_full[indices]=pdf_MC[i]

# Record the end time
end_time = time.time()


# Calculate the elapsed time
elapsed_time = end_time - start_time

# Print the elapsed time
print(f"Elapsed Time: {elapsed_time:.4f} seconds")



# Create a subplot with 1 row and 3 columns, specifying the width ratios
fig = plt.figure(figsize=(10, 10))
gs = fig.add_gridspec(2, 2, height_ratios=[1, 1])

num_bars = len(pdf_MC_full)

# Generate a list of colors using a colormap
colors = plt.cm.viridis(np.linspace(0, 1, num_bars))


# First subplot (1 row, 3 columns, first plot)
```

```python
ax1 = fig.add_subplot(gs[0])
ax1.bar(energy_arry, pdf_MC_full,color= colors)

ax1.set_xlabel('Energy')
ax1.set_ylabel('Probability')
ax1.grid(True)
ax1.set_title('Simulated Boltzmann Distribution for Ising Model')


# Second subplot (1 row, 3 columns, second plot)
ax2 = fig.add_subplot(gs[1])
ax2.bar(energy_arry, pdf ,color= colors)
ax2.set_xlabel('Energy')
ax2.set_ylabel('Probability')
ax2.grid(True)
ax2.set_title('Exact Boltzmann Distribution for Ising Model')

# Third subplot (1 row, 3 columns, third plot - elongated)
ax3 = fig.add_subplot(gs[1, :])
ax3.bar(energy_arry, np.abs(pdf_MC_full-pdf),color= colors )
ax3.set_xlabel('Energy')
ax3.set_ylabel('Probability (percent difference)')
ax3.grid(True)
ax3.set_title(' Simulated v.s. Exact')
```

magnetization, Landau theory, and phase transition of the Ising model

Magnetization is a measure of the average magnetic moment per spin in a given direction. In the context of the Ising model, the magnetic moment is represented by the sum of the spins. The **magnetization** M is defined as:

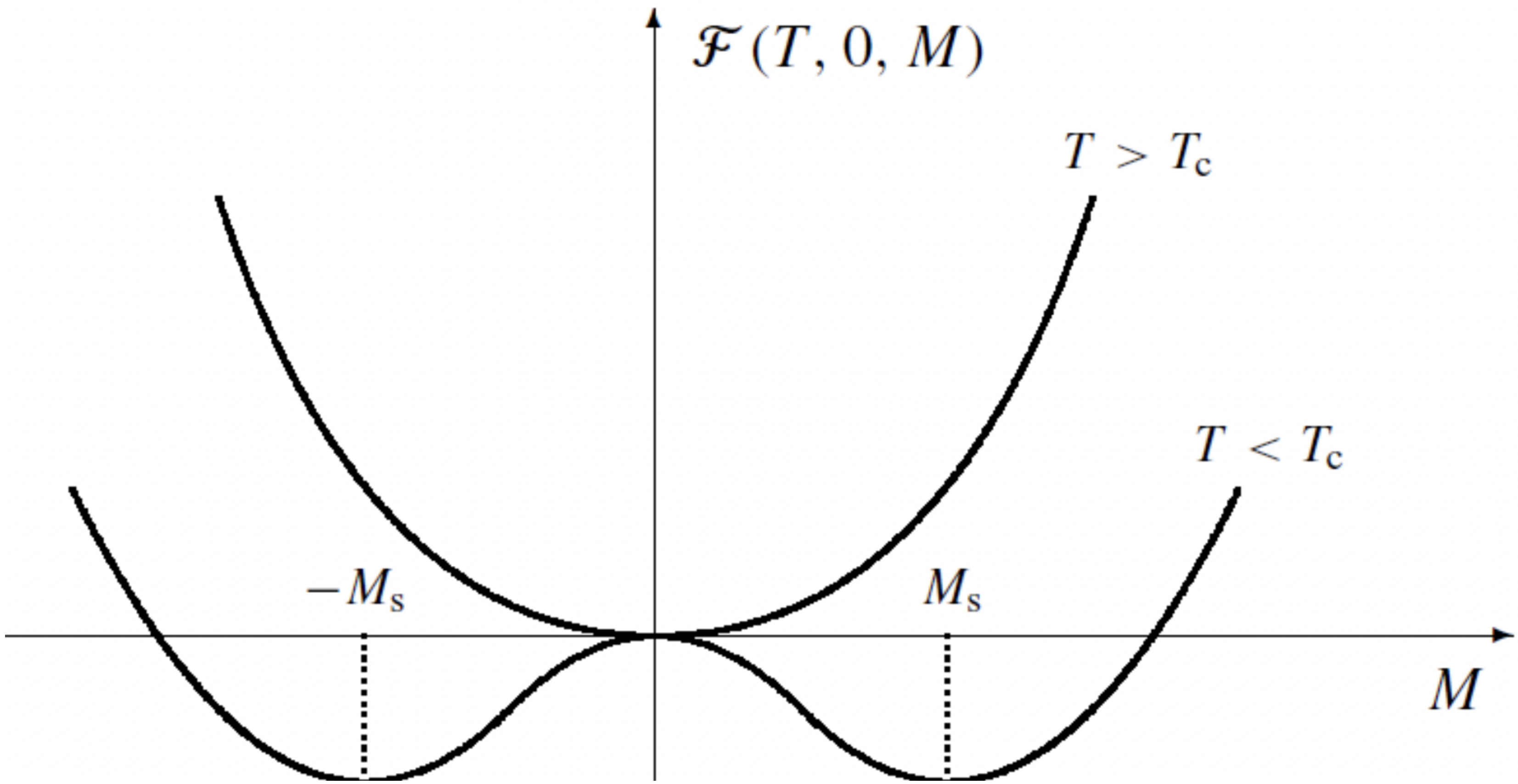
$$M = \frac{1}{N} \sum_i S_i$$

where $N = L^2$ is the total number of spins. In the absence of an external magnetic field (B=0), the Ising model exhibits a spontaneous magnetization at low temperatures, where most spins tend to align in the same direction. In the Ising model, the magnetization serves as an order parameter that indicates the presence of a magnetic order in the system. The order parameter is often used to characterize the different phases of the model, especially in the study of phase transitions.

Landau theory, in the context of phase transitions, is a phenomenological approach that describes the free energy of a system near a critical point. It provides a framework for understanding the symmetry-breaking mechanisms and the emergence of order parameters. When near a second-order phase transition, the **Landau free energy** can often be written as a Taylor expansion in terms of an order parameter, typically the magnetization

$$F = F_0 + a(T - T_c)M^2 + bM^4 + \ldots$$

where F_0 is the free energy at the critical temperature (T_c), a and b are phenomenological coefficients, T is the temperature.



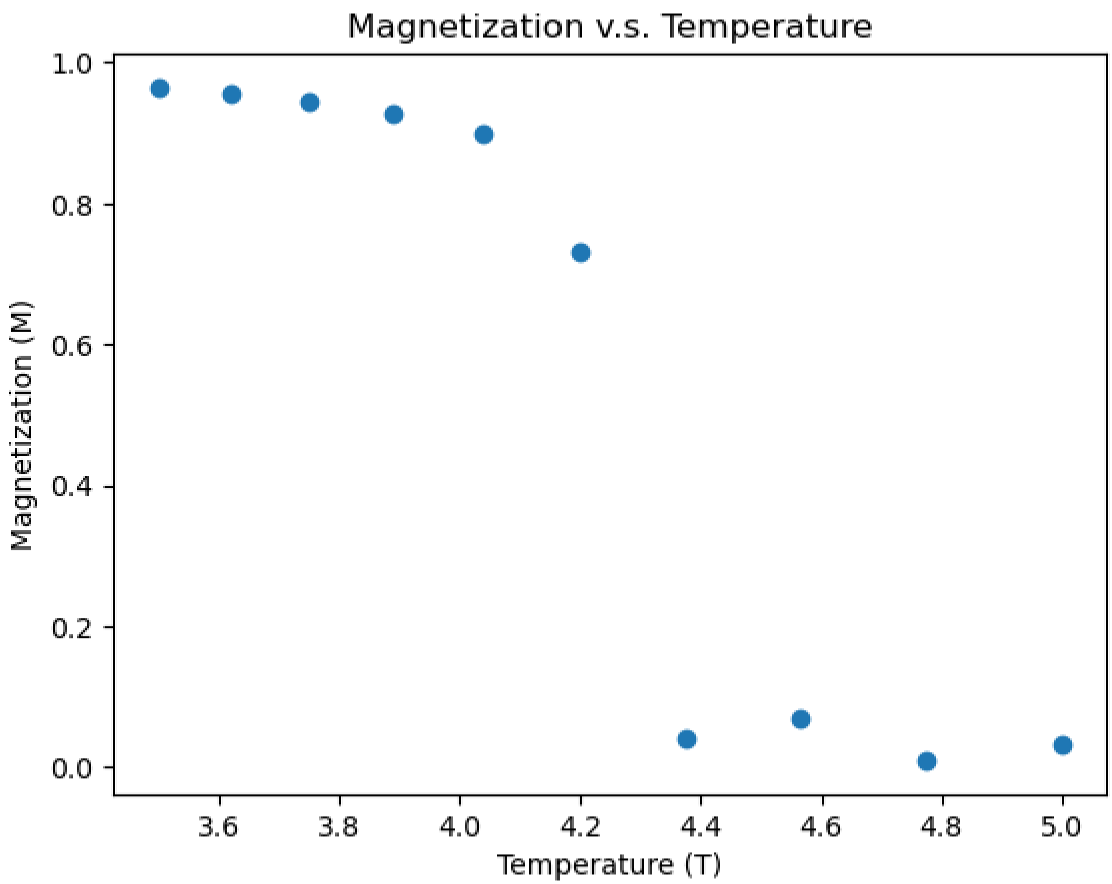
Example: magnetization at different temperature

In [20]:

```python
# Function to perform a Gibbs stepping with magnetization
def Gibbs_step_with_M(spins_init, beta_arry, J, B, grid_size,burnin_=13000,sample_size=5000):
    sample_M=[]
    for f in range(0,len(beta_arry)):
        beta=beta_arry[f]
        spins=spins_init
        burnin_step=0
        s=0
        M=0

        while burnin_step<=burnin_:
            for i in range(grid_size):
                for j in range(grid_size):
                    energy_c,prob_c,new_spin_val_c=Gibbs_sampler(spins,spin_index_x=i,spin_index_y=j,
                    spins[i,j]=new_spin_val_c
            burnin_step+=1

        while s<sample_size:
            for i in range(grid_size):
                for j in range(grid_size):
                    energy_c,prob_c,new_spin_val_c=Gibbs_sampler(spins,spin_index_x=i,spin_index_y=j,
                    spins[i,j]=new_spin_val_c

            M+=np.sum(spins)/(grid_size**2)

            s+=1
        print(M)
        sample_M.append(M/sample_size)

    return sample_M

beta_arry=np.linspace(0.2,1/3.5,10)

T_array=1/beta_arry

spins = generate_random_spin_configuration(L_size)
sample_M=energy_arry_MC=Gibbs_step_with_M(spins_init=spins,beta_arry=beta_arry,J=J_value, B=B_value,g
# Create scatter plot
plt.scatter(T_array, np.abs(sample_M), color='tab:blue', marker='o' )

# Add labels and title
plt.xlabel('Temperature (T)')
plt.ylabel('Magnetization (M)')
```

```python
plt.title('Magnetization v.s. Temperature')


# Show the plot
plt.show()
```

```
-279.5400000000005
86.38000000000038
631.9400000000003
365.2599999999956
6575.319999999883
8080.59999999959
8338.83999999944
8487.959999999355
8597.719999999284
8680.81999999924
```



In [ ]:

MC convergence and lattice size

Let's look at the MC convergence at different lattice size.

In [40]:

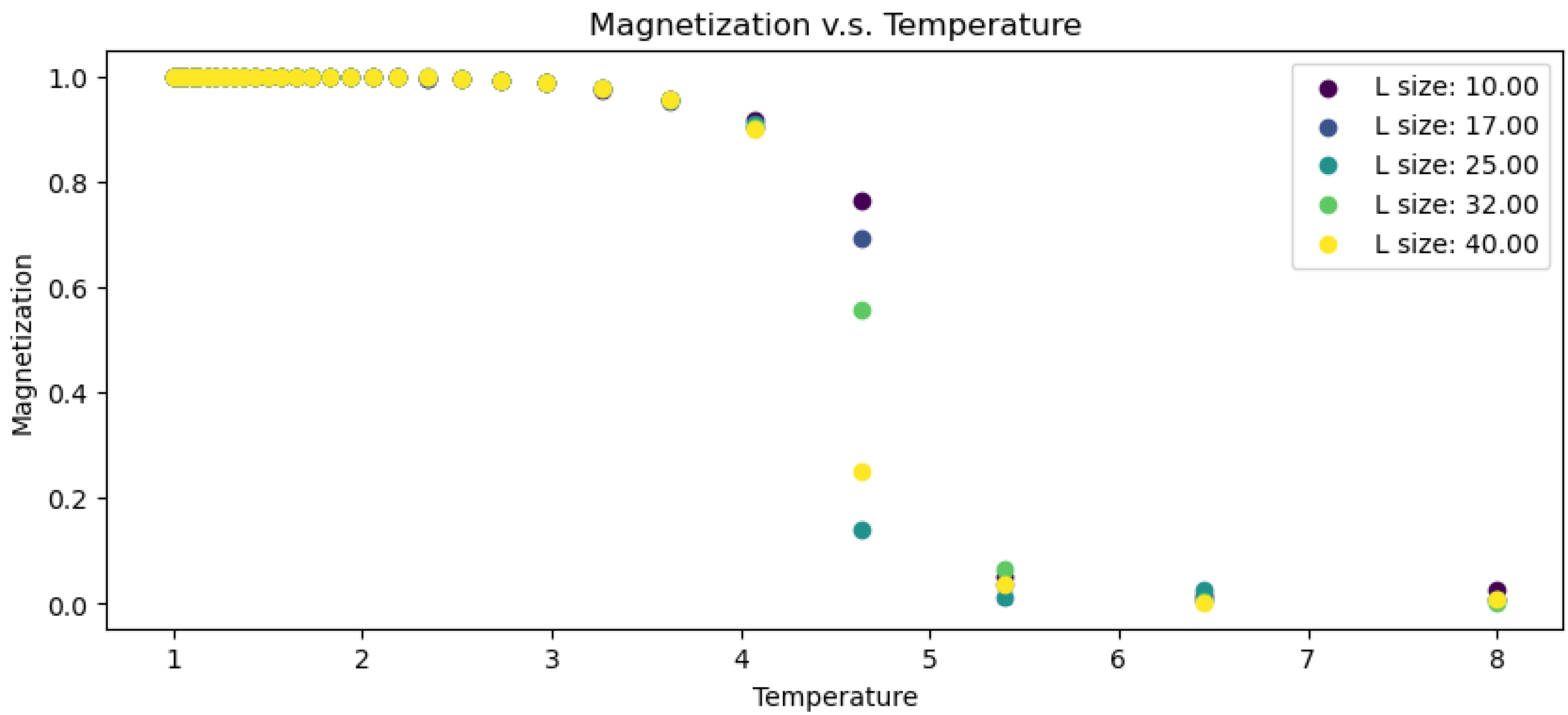
```python
# Create a figure and axis objects
fig, axs = plt.subplots(1, 1, figsize=(10, 4))

L_arry=np.linspace(10,40,5)
# Generate a list of colors using a colormap
colors = plt.cm.viridis(np.linspace(0, 1, len(L_arry)))

for g in range(0,len(L_arry)):
    L_size =int(L_arry[g])    # lattice size

    beta_arry=np.linspace(1/8,1/1,30)

    T_array=1/beta_arry

    spins = generate_random_spin_configuration(L_size)
    sample_M=Gibbs_step_with_M(spins_init=spins,beta_arry=beta_arry,J=J_value, B=B_value,grid_size=L_
    # Create scatter plot
    axs.scatter(T_array, np.abs(sample_M), color=colors[g], marker='o', label=f'L size: {L_size:.2f}'
axs.axvline(x=Tc, color='tab:red', linestyle='--', label=f'Critical Temperature (Tc): {Tc:.2f}')


# Add labels and title
axs.set_xlabel('Temperature')
axs.set_ylabel('Magnetization')
axs.set_title('Magnetization v.s. Temperature')
axs.legend()

# Show the plot
plt.show()
```

-12.900000000000013
7.999999999999996
-26.42000000000002
382.0199999999998
459.35999999999916
478.79999999999984
486.6800000000002
493.56000000000057
496.74000000000063
497.6800000000004
498.5000000000003
499.18000000000023
499.42000000000024
499.72000000000014
499.7800000000001
499.8
499.94000000000005
499.90000000000003
499.98
500.0
500.0
500.0
500.0
500.0
500.0
500.0
500.0
500.0
500.0
500.0
3.9307958477508644
4.2006920415225055
-17.757785467127984
-345.51557093425635
-450.9826989619379
-476.98961937716325
-488.844290657439
-493.70242214532846
-496.29757785467075
-497.9792387543253
-498.60207612456776
-499.3910034602079
-499.6055363321802

-499.70242214532885
-499.8339100346022
-499.86159169550183
-499.96539792387546
-500.0
-499.99307958477505
-499.99307958477505
-500.0
-499.9792387543253
-499.9861591695502
-500.0
-499.99307958477505
-500.0
-500.0
-500.0
-500.0
-500.0
3.801600000000002
12.870399999999986
6.022399999999997
-70.29120000000003
-455.0464
-479.0143999999999
-488.31999999999897
-493.5071999999991
-496.7104000000008
-498.00000000000136
-498.7072000000015
-499.2800000000007
-499.4752000000006
-499.71200000000044
-499.8144000000003
-499.92640000000017
-499.9040000000001
-499.9680000000001
-499.97440000000006
-499.9904
-499.9936
-499.9936
-500.0
-499.9936
-499.9968
-499.9968

-500.0
-500.0
-500.0
-500.0
1.21484375
2.853515625
-32.841796875
279.17578125
450.685546875
477.369140625
488.658203125
493.8203125
496.349609375
497.96484375
498.671875
499.17578125
499.517578125
499.708984375
499.818359375
499.9140625
499.9296875
499.9609375
499.974609375
499.98046875
499.982421875
499.9921875
499.998046875
500.0
499.998046875
499.99609375
500.0
500.0
500.0
499.998046875
4.111250000000003
-1.5899999999999987
-18.83000000000001
124.77125
450.11999999999944
477.3887500000002
488.28625000000056
493.5237499999997
496.4687499999988

497.93749999999784
498.75124999999764
499.2637499999973
499.56749999999784
499.7299999999985
499.8162499999985
499.891249999999
499.9487499999996
499.96874999999966
499.98249999999985
499.9774999999998
499.99249999999995
499.99875
499.99749999999995
499.995
499.99875
500.0
500.0
500.0
500.0
500.0



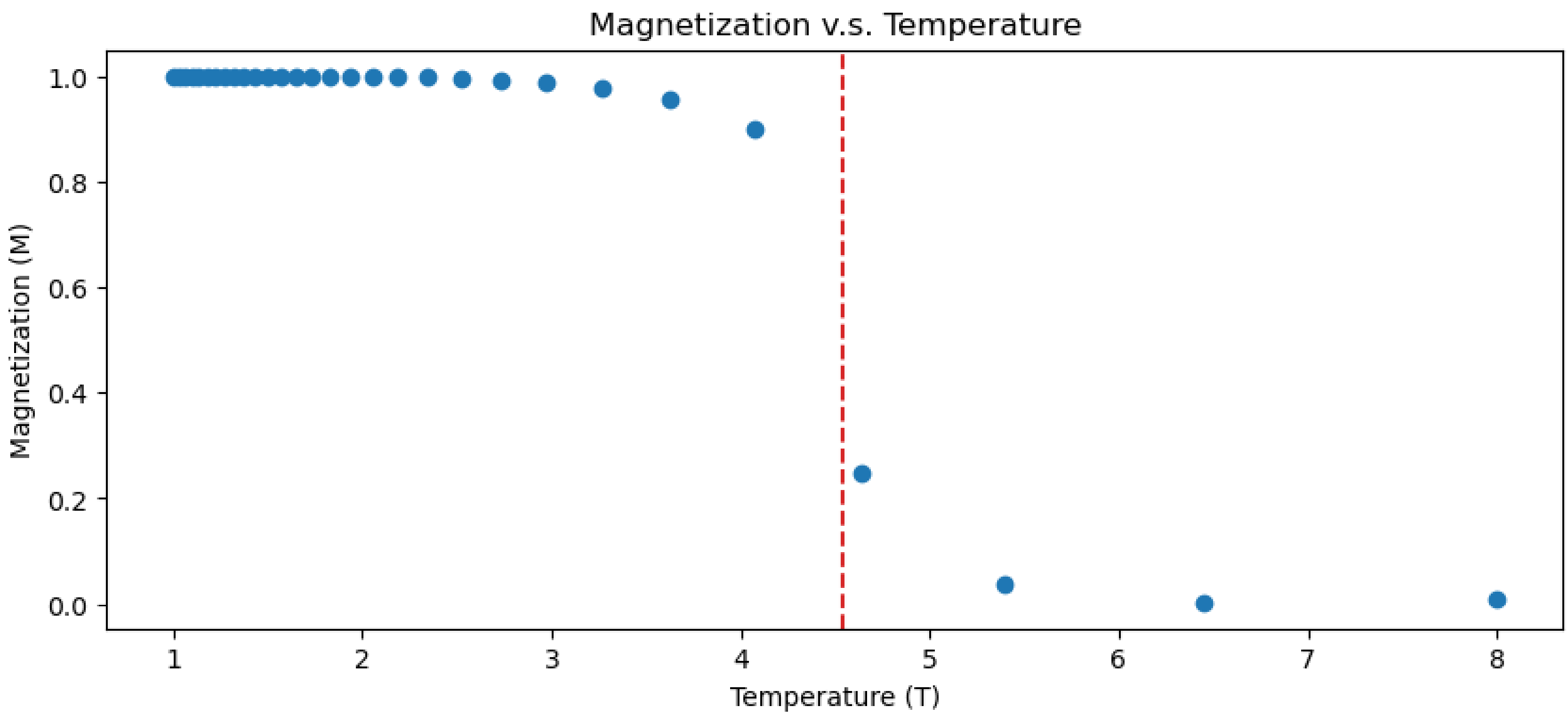
The critical temperature

In 2D Ising model, the critical temperature can be calculated via the following,

$$T_C = \frac{2J}{\log(1+\sqrt{2})}.$$

In [42]:

```python
# Define the power-law function
Tc=2*J_value/(np.log(1+np.sqrt(2)))

fig, axs = plt.subplots(1, 1, figsize=(10, 4))

plt.scatter(T_array, np.abs(sample_M), color='tab:blue', marker='o' )

# Add labels and title
plt.xlabel('Temperature (T)')
plt.ylabel('Magnetization (M)')
plt.title('Magnetization v.s. Temperature')

# Show the plot
plt.show()
```



Magnetic Field dependence: magnetization of the 2D Ising model

Let's look at the MC convergence at different lattice size.

In [13]:

```
# Function to perform a Gibbs stepping with magnetization
def Gibbs_step_with_M_vary_B(spins_init, beta, J, B_array, grid_size,burnin_=13000,sample_size=5000):
    sample_M=[]
    for f in range(0,len(B_array)):
        B=B_array[f]
        spins=spins_init
        burnin_step=0
        s=0
        M=0

        while burnin_step<=burnin_:
            for i in range(grid_size):
                for j in range(grid_size):
                    energy_c,prob_c,new_spin_val_c=Gibbs_sampler(spins,spin_index_x=i,spin_index_y=j,
                    spins[i,j]=new_spin_val_c
            burnin_step+=1

        while s<sample_size:
            for i in range(grid_size):
                for j in range(grid_size):
                    energy_c,prob_c,new_spin_val_c=Gibbs_sampler(spins,spin_index_x=i,spin_index_y=j,
                    spins[i,j]=new_spin_val_c

            M+=np.sum(spins)/(grid_size**2)

            s+=1
        print(M)
        sample_M.append(M/sample_size)

    return sample_M
```

Example

In [ ]:

```
B_arry=np.linspace(-50,50,100)

beta_value=0.05 # Temperature
L_size =5

spins = generate_random_spin_configuration(L_size)
sample_M=energy_arry_MC=Gibbs_step_with_M_vary_B(spins_init=spins,beta=beta_value,J=J_value, B_array=
# Create scatter plot
plt.scatter(B_arry, np.sign(B_arry)*np.abs(sample_M), color='tab:blue', marker='o' )

# Add labels and title
plt.xlabel('Temperature (T)')
plt.ylabel('Magnetization (M)')
plt.title('Magnetization v.s. Temperature')


# Show the plot
plt.show()
```

MC convergence and lattice size

Let's look at the MC convergence at different lattice size.

In [54]:
```python
# Create a figure and axis objects
fig, axs = plt.subplots(1, 1, figsize=(10, 4))

L_arry=np.linspace(10,40,5)
# Generate a list of colors using a colormap
colors = plt.cm.viridis(np.linspace(0, 1, len(L_arry)))

for g in range(0,len(L_arry)):
    L_size =int(L_arry[g])    # lattice size

    B_arry=np.linspace(-50,50,100)

    spins = generate_random_spin_configuration(L_size)
    # Create scatter plot

    sample_M=Gibbs_step_with_M_vary_B(spins_init=spins,beta=beta_value,J=J_value, B_array=B_arry,grid
    axs.scatter(B_arry, np.sign(B_arry)*np.abs(sample_M), color=colors[g], marker='o', label=f'L size

# Add labels and title
axs.set_xlabel('Magnetic Field')
axs.set_ylabel('Magnetization')
axs.set_title('Magnetization v.s. Magnetic Field')
axs.legend()

# Show the plot
plt.show()
```

-496.58000000000055
-496.8200000000003
-495.7800000000008
-496.2400000000004
-495.26000000000056
-495.6000000000006
-494.28000000000077
-493.78000000000065
-493.3000000000001
-493.0000000000005
-492.0000000000002
-490.5200000000005
-490.00000000000085
-488.6600000000001
-487.56000000000006
-486.3600000000003
-484.9199999999998
-483.0200000000001
-480.23999999999955
-477.8599999999993
-475.81999999999937
-475.6999999999993
-471.1599999999994
-468.9399999999992
-465.47999999999973
-460.81999999999914
-457.5199999999993
-451.71999999999923
-446.2599999999994
-442.05999999999966
-434.95999999999907
-424.779999999999
-417.91999999999933
-407.4399999999996
-395.86
-387.97999999999985
-371.9199999999999
-362.62000000000035
-341.6800000000004
-323.12000000000006
-309.0600000000001
-284.73999999999984
-259.1599999999998

-235.32000000000002
-201.9
-166.22000000000006
-134.96000000000004
-91.38000000000004
-56.86000000000005
-19.279999999999987
22.59999999999998
62.81999999999987
92.34000000000002
130.33999999999986
166.61999999999995
202.95999999999998
227.42000000000013
256.5999999999999
283.7199999999999
305.1799999999999
324.7400000000003
342.5800000000003
358.94000000000034
374.9799999999995
389.95999999999964
397.5399999999991
407.3399999999995
417.51999999999913
425.5399999999993
433.839999999999
439.7199999999994
445.4199999999992
451.8599999999992
456.45999999999924
460.5599999999998
464.2999999999989
467.4399999999993
471.13999999999925
474.1999999999996
477.03999999999917
478.6399999999996
480.43999999999966
482.67999999999995
484.75999999999993
486.04000000000013
487.0400000000003

488.1200000000004
489.3200000000002
491.40000000000066
490.9000000000007
492.26000000000073
493.1400000000003
493.4800000000005
494.62000000000074
495.0600000000005
495.0800000000007
495.52000000000066
496.1400000000007
496.5800000000005
496.8200000000003
-497.0103806228375
-496.6712802768167
-496.47058823529426
-495.8961937716257
-495.2802768166084
-495.22491349480947
-494.4982698961935
-493.7093425605529
-493.1418685121101
-492.6435986159165
-491.05882352941126
-490.76816608996495
-490.2214532871965
-489.0034602076121
-487.52941176470574
-487.0034602076125
-484.7889273356406
-482.5813148788929
-481.23183391003573
-479.59861591695534
-476.53979238754323
-474.2422145328722
-471.03114186851303
-467.854671280277
-464.1591695501738
-460.8788927335642
-456.0415224913504
-451.70242214532925
-447.4325259515572

-441.00346020761316
-433.487889273357
-426.6020761245677
-417.88235294117703
-408.8996539792393
-397.23875432525944
-385.19031141868527
-370.89273356401435
-358.47058823529443
-341.97231833910087
-324.2006920415227
-306.59515570934286
-283.21799307958497
-257.28719723183366
-232.43598615916937
-203.11418685121095
-165.80622837370254
-130.8096885813149
-97.66089965397924
-58.76124567474049
-19.07958477508651
18.830449826989614
59.27335640138411
96.11764705882365
134.58131487889267
167.4256055363321
198.58823529411745
229.20415224913484
258.8581314878894
281.16955017301075
304.2422145328722
323.7093425605538
344.38062283737037
358.90657439446414
373.8269896193777
387.27335640138466
398.6228373702423
408.9204152249139
418.5328719723186
425.58477508650583
433.958477508651
440.754325259516
445.6539792387541

452.65051903114215
455.2249134948096
460.88581314878957
464.76816608996575
469.05190311418755
470.1314878892732
474.31141868512174
476.41522491349497
478.83737024221523
480.8996539792391
482.4498269896196
484.8027681660898
486.359861591696
487.32871972318355
488.8719723183381
489.9930795847748
490.8442906574389
492.11764705882337
492.4913494809682
492.89965397923766
493.7024221453278
494.1522491349475
494.9896193771624
495.66089965397884
496.0553633217995
496.4982698961937
496.48442906574405
496.98269896193744
-496.87040000000127
-496.6304000000011
-496.2208000000007
-495.8560000000007
-495.60959999999994
-494.70719999999994
-494.38719999999927
-493.7663999999992
-493.3503999999988
-492.26879999999835
-491.759999999999
-490.90879999999856
-489.70559999999887
-488.918399999999
-487.4335999999993

-486.34559999999965
-484.34880000000044
-483.15200000000044
-481.2608000000003
-479.3024000000002
-476.9696000000002
-473.7440000000006
-471.4624000000012
-468.08960000000087
-464.73600000000016
-460.3648000000002
-455.78559999999925
-452.10879999999975
-446.18879999999956
-440.25600000000037
-433.3567999999997
-426.5024000000001
-418.4031999999998
-408.3648000000003
-399.0976
-386.1344000000003
-375.14240000000024
-359.3856000000001
-341.30559999999974
-324.59200000000004
-305.0464
-280.9536
-257.7887999999998
-230.78719999999973
-200.7968
-169.81119999999996
-130.84479999999994
-95.22560000000009
-61.30239999999998
-19.849600000000002
22.579199999999997
59.43360000000011
95.9712
133.53919999999997
169.1456000000003
200.18240000000011
229.26079999999985
256.2943999999998

282.5408000000002
305.0239999999999
325.2639999999998
342.2719999999999
359.51359999999977
374.5632000000002
387.3984000000002
398.7136000000002
408.85120000000006
417.83039999999966
425.4687999999997
433.1775999999999
440.5312
446.4927999999998
451.23519999999985
455.9392000000003
461.35360000000003
463.9296000000004
468.34880000000027
471.48160000000075
474.2144000000003
476.6560000000005
478.8128000000003
480.72960000000074
482.6656000000005
484.21440000000007
486.1759999999998
487.3503999999998
488.9119999999988
489.65119999999877
490.6079999999987
491.7247999999984
492.63679999999795
493.03359999999896
493.763199999999
494.4319999999993
495.0495999999999
495.8592000000008
495.7568000000007
496.16000000000116
496.659200000001
496.9312000000013
-496.93359375

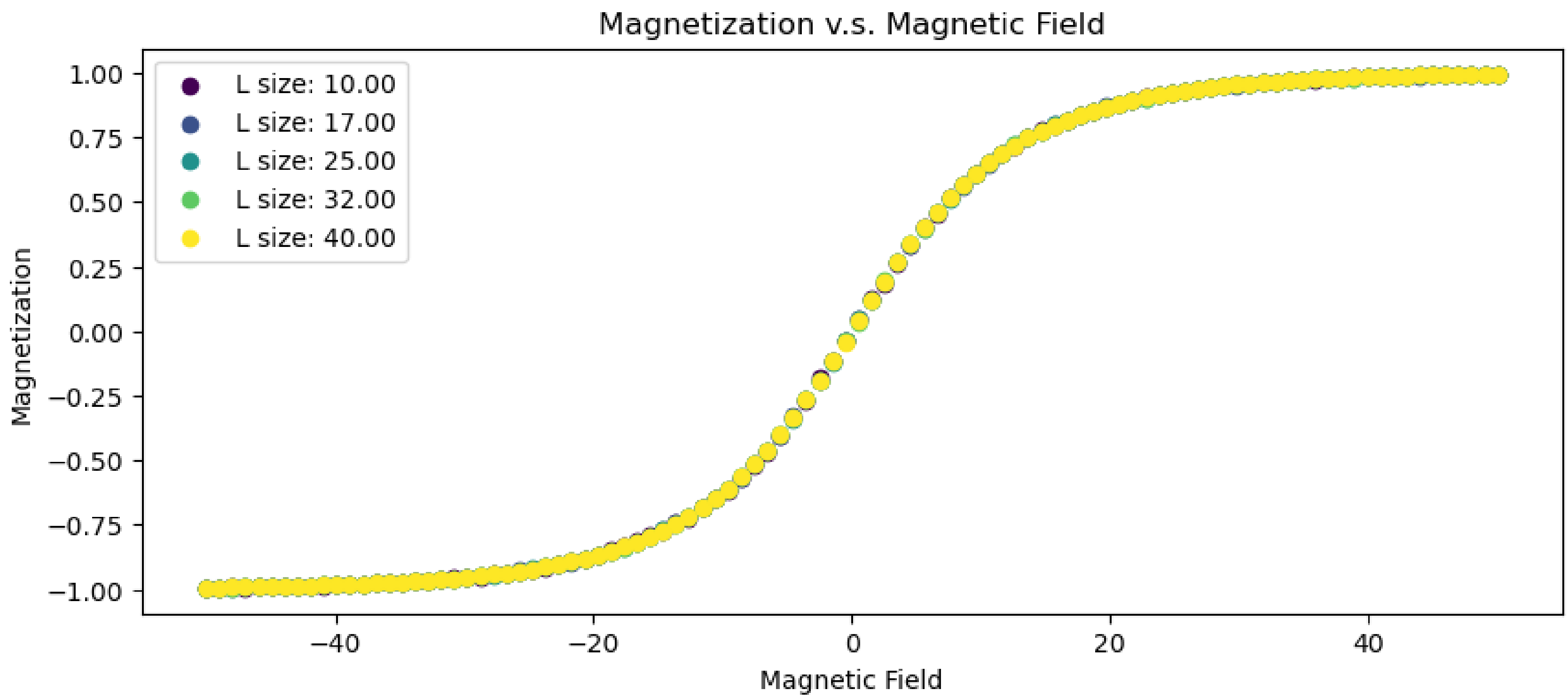
-496.572265625
-496.287109375
-495.970703125
-495.412109375
-494.9375
-494.52734375
-493.681640625
-493.166015625
-492.546875
-491.568359375
-490.8828125
-489.875
-488.662109375
-487.607421875
-486.138671875
-484.787109375
-483.052734375
-480.6953125
-478.716796875
-476.5390625
-473.69140625
-471.251953125
-468.32421875
-464.8828125
-461.19921875
-456.732421875
-452.287109375
-446.423828125
-441.076171875
-433.58203125
-426.15234375
-418.16796875
-407.876953125
-397.630859375
-386.212890625
-374.041015625
-358.666015625
-342.236328125
-324.935546875
-305.87890625
-281.015625
-257.861328125
-230.728515625

-200.431640625
-169.5625
-132.67578125
-96.900390625
-59.224609375
-19.984375
18.35546875
58.849609375
98.130859375
134.115234375
169.376953125
199.302734375
230.138671875
258.779296875
282.03125
305.431640625
324.509765625
342.16796875
360.884765625
375.12890625
386.857421875
397.75
408.421875
417.544921875
426.05078125
433.626953125
439.689453125
446.259765625
451.419921875
456.669921875
460.54296875
464.455078125
468.142578125
470.90234375
474.458984375
476.705078125
479.177734375
481.1328125
483.140625
484.673828125
485.951171875
487.251953125
488.53515625

489.79296875
490.650390625
491.73046875
492.390625
493.1171875
493.845703125
494.35546875
495.033203125
495.513671875
495.8984375
496.236328125
496.587890625
496.849609375
-496.93999999999886
-496.716249999999
-496.21249999999895
-495.87499999999915
-495.48249999999945
-494.90874999999926
-494.4387499999994
-493.89874999999995
-493.1100000000001
-492.4550000000003
-491.6975
-490.7074999999999
-489.6650000000003
-488.5925000000001
-487.4075000000005
-486.12250000000074
-484.57124999999996
-482.75624999999957
-480.5937499999993
-478.74999999999994
-476.8925000000001
-474.35250000000025
-471.2175000000001
-467.91249999999974
-464.89625000000007
-460.96999999999963
-456.36500000000007
-451.69499999999994
-446.13374999999996
-440.3012499999998

-434.0424999999996
-426.51749999999987
-417.8675000000005
-408.58124999999967
-398.24000000000007
-386.39124999999996
-373.4137500000002
-359.5325
-342.12375
-323.97000000000014
-305.31249999999983
-282.16
-257.49749999999995
-231.14374999999995
-199.8912499999998
-168.18999999999997
-133.18749999999997
-95.83249999999997
-56.09750000000001
-21.010000000000016
20.297499999999985
58.783749999999955
96.42249999999996
134.44124999999997
167.78124999999997
201.26875000000018
229.24374999999998
257.49875000000014
283.0037500000001
305.2537499999998
324.37750000000005
342.6437499999999
359.32375000000053
373.63499999999976
385.95875000000007
397.61999999999966
408.33625000000046

417.8812499999998
426.1062500000005
433.7075000000003
440.1974999999997
446.17375000000067
451.75875000000036
456.28374999999966
460.8849999999997
464.63625000000013
468.21999999999974
471.47375000000017
474.28875000000033
476.78875
479.16375000000005
481.04624999999925
482.64000000000004
484.4950000000005
486.00500000000045
487.490000000001
488.7012500000002
489.65750000000014
490.8775000000003
491.5862499999998
492.42125
493.1987500000005
493.84750000000025
494.48250000000036
494.98749999999956
495.5812499999992
495.8987499999996
496.3812499999988
496.6562499999991
497.0312499999987



Specific Heat of the 2D Ising model

The specific heat (C_v) measures the amount of heat energy required to change the temperature of a substance by a unit temperature. It can be related to the variance of the energy in the 2D Ising model, e.g.,

$$C_v = \frac{\beta^2}{N}\left(\langle E^2 \rangle - \langle E \rangle^2\right),$$

where N is the total number of spins.

In [15]:

```
# Function to perform a Gibbs stepping with magnetization
def Gibbs_step_with_E(spins_init, beta_arry, J, B, grid_size,burnin_=13000,sample_size=5000):
    sample_M=[]
    sample_C=[]
    for f in range(0,len(beta_arry)):
        beta=beta_arry[f]
        spins=spins_init
        burnin_step=0
        s=0
        M=0
        E=0 #define average energy
        E_square=0
        while burnin_step<=burnin_:
            for i in range(grid_size):
                for j in range(grid_size):
                    energy_c,prob_c,new_spin_val_c=Gibbs_sampler(spins,spin_index_x=i,spin_index_y=j,
                    spins[i,j]=new_spin_val_c
            burnin_step+=1

        while s<sample_size:
            for i in range(grid_size):
                for j in range(grid_size):
                    energy_c,prob_c,new_spin_val_c=Gibbs_sampler(spins,spin_index_x=i,spin_index_y=j,
                    spins[i,j]=new_spin_val_c

            M+=np.sum(spins)/(grid_size**2)

            s+=1
            current_delta_E=net_energy(spins, J, B,print_val=False)
            E+=np.abs(current_delta_E)
            E_square+=current_delta_E**2
        print(M)
        sample_M.append(M/sample_size)

        E_2=(E/sample_size)**2
        E2=E_square/sample_size

        C=(1/grid_size**2)*beta**2*(E2-E_2)
        sample_C.append(C)
        print(C)
        print('---------')
```

```
    return sample_C
```

```
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
```

```
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
```

```
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-499.92
0.012862426611789778
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
```

```
-500.0
0.0
---------
-499.92
0.010725045323324679
---------
-499.92
0.01054036164852806
---------
-500.0
0.0
---------
-500.0
0.0
---------
-499.92
0.009995934339202045
---------
-499.91999999999996
0.009817666474447989
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
0.0
---------
-499.91999999999996
0.009120634540538165
---------
-499.91999999999996
0.00895038643833731
---------
-500.0
0.0
---------
-500.0
0.0
---------
-500.0
```

```
0.0
---------
-500.0
0.0
---------
-499.91999999999996
0.00812320521499265
---------
-500.0
0.0
---------
-500.0
0.0
---------
-499.92
0.007646143911113537
---------
-499.75999999999993
0.02429854493762212
---------
-499.84000000000003
0.014642842311062713
---------
-499.76000000000005
0.021464179571486264
---------
-499.76
0.021012993194608445
---------
-500.0
0.0
---------
-499.92
0.006735328021142614
---------
-500.0
0.0
---------
-499.91999999999996
0.006444554344570768
---------
-499.92
0.006301573435050802
```

```
---------
-499.84000000000003
0.012295702789878599
---------
-500.0
0.0
---------
-499.76000000000005
0.01757603475465901
---------
-499.75999999999993
0.018638947074461242
---------
-499.68
0.022307985369678384
---------
-500.0
0.0
---------
-499.6
0.026513835477666958
---------
-499.84
0.010410033217423011
---------
-499.28
0.04641180200371575
---------
-498.8800000000001
0.0703840292899184
---------
-499.68000000000006
0.019222313193133448
---------
-499.36000000000007
0.03715860716310076
---------
-499.12000000000006
0.04946503919385056
---------
-499.52
0.026540901684296278
---------
```

```
-499.28000000000003
0.03968222604087892
---------
-499.5200000000001
0.02513766870002895
---------
-499.36
0.032468471856040994
---------
-499.28
0.03544226576199964
---------
-498.8
0.058883159497297895
---------
-499.36000000000007
0.02980621008058858
---------
-498.4000000000001
0.07059534810624675
---------
-498.96
0.045189807783356954
---------
-497.6000000000001
0.10085079122254291
---------
-497.84000000000015
0.08780323799948674
---------
-497.6800000000002
0.09091432922524158
---------
-497.84000000000003
0.08684994705958235
---------
-497.6800000000002
0.08435418542198755
---------
-496.72000000000014
0.1288831913240651
---------
-496.80000000000024
```

0.11037485494967847

-496.64000000000027
0.10907440093084839

-496.32000000000016
0.12316479599989053

-495.3600000000001
0.14444649664865275

-494.5600000000003
0.17244803232449307

-494.80000000000035
0.14348250547272445

-494.16000000000014
0.16707237434970157

493.19999999999993
0.1911655807982692

491.6000000000003
0.22230732682845677

492.40000000000003
0.1943259056922779

491.60000000000014
0.20363662727024062

-480.96
0.5134593490083545

483.8400000000003
0.3898441620151715

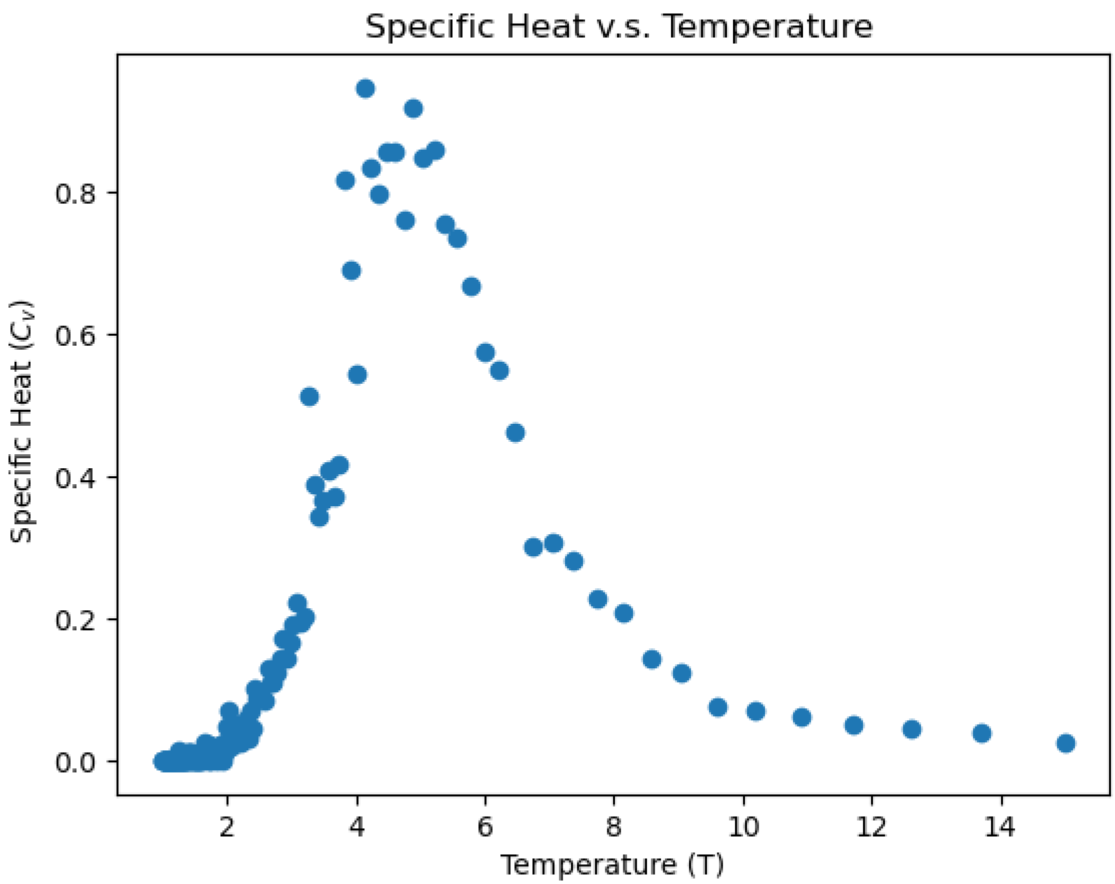
-483.84
0.3423221635844298

480.47999999999996
0.36533398264770583

```
---------
-479.6000000000003
0.40914193054366393
---------
-478.39999999999986
0.37102822922405665
---------
474.8000000000002
0.4153038626327824
---------
129.28000000000003
0.8174198525254326
---------
-458.32
0.6911494941751298
---------
465.67999999999995
0.5424407945299754
---------
395.59999999999997
0.9465342415895174
---------
240.55999999999986
0.8323706769794292
---------
-98.16000000000008
0.7970760738777541
---------
-356.96
0.8559103819173207
---------
146.56000000000014
0.8568543548400718
---------
312.47999999999956
0.7603933564431155
---------
-4.400000000000004
0.9189399916938601
---------
-33.83999999999999
0.8466046456253726
---------
```

```
17.60000000000001
0.8579001987199845
---------
173.28000000000011
0.7534609432840355
---------
21.440000000000012
0.7345364455177567
---------
17.200000000000003
0.667894938026647
---------
-63.2000000000001
0.5754014372639786
---------
-8.880000000000024
0.5481663101998081
---------
-40.32
0.46079759047970803
---------
18.00000000000002
0.3025875226176675
---------
65.52000000000011
0.30713590068416136
---------
-27.040000000000003
0.2816238566677891
---------
-2.8000000000000096
0.22714875992025152
---------
-21.679999999999968
0.20878728433654123
---------
-23.759999999999994
0.14307988171660715
---------
21.600000000000005
0.12418177927139604
---------
-5.439999999999996
```

```
0.07575787885763702
---------
11.839999999999996
0.0710889439159676
---------
-17.919999999999977
0.06222361042795863
---------
9.599999999999998
0.05017172933130338
---------
-0.08000000000000212
0.04379015250008547
---------
-1.6800000000000144
0.03850104903482019
---------
6.160000000000001
0.02610517333333333
---------
```



Specific Heat v.s. Temperature
Specific Heat (C_v)
Temperature (T)
0.0
0.2
0.4
0.6
0.8
2
4
6
8
10
12
14

Example

In [ ]:

```
beta_arry=np.linspace(1/1,1/15,150)

T_array=1/beta_arry
L_size =5

spins = generate_random_spin_configuration(L_size)
sample_C=Gibbs_step_with_E(spins_init=spins,beta_arry=beta_arry,J=J_value, B=B_value,grid_size=L_size
# Create scatter plot
plt.scatter(T_array, (sample_C), color='tab:blue', marker='o' )

# Add labels and title
plt.xlabel('Temperature (T)')
plt.ylabel(' Specific Heat ($C_v$)')
plt.title(' Specific Heat v.s. Temperature')


# Show the plot
plt.show()
```

Magnetic Susceptibility of the 2D Ising model

Similarly, the magnetic susceptibility quantifies the material's ability to become magnetized per unit change in the external magnetic field strength,

$$\chi = \frac{\beta}{N}\left(\langle M^2\rangle - \langle M\rangle^2\right).$$

In [48]:

```python
# Function to perform a Gibbs stepping with magnetization
def Gibbs_step_with_MS(spins_init, beta_arry, J, B, grid_size,burnin_=13000,sample_size=5000):
    sample_M=[]
    sample_Chi=[]
    for f in range(0,len(beta_arry)):
        beta=beta_arry[f]
        spins=spins_init
        burnin_step=0
        s=0
        M=0
        MS=0 #define average energy
        MS_square=0
        while burnin_step<=burnin_:
            for i in range(grid_size):
                for j in range(grid_size):
                    energy_c,prob_c,new_spin_val_c=Gibbs_sampler(spins,spin_index_x=i,spin_index_y=j,
                    spins[i,j]=new_spin_val_c
            burnin_step+=1

        while s<sample_size:
            for i in range(grid_size):
                for j in range(grid_size):
                    energy_c,prob_c,new_spin_val_c=Gibbs_sampler(spins,spin_index_x=i,spin_index_y=j,
                    spins[i,j]=new_spin_val_c

            M+=np.sum(spins)/(grid_size**2)

            s+=1

            MS+=np.abs(np.sum(spins) )
            MS_square+=((np.sum(spins))**2)
        print(M)
        print(MS)
        print(MS_square)
        sample_M.append(M/sample_size)

        MS_2=(MS/sample_size)**2
        MS2=MS_square/sample_size

        Chi=1/(grid_size**2)*beta*(MS2-MS_2)
        print(Chi)
        sample_Chi.append(Chi)
```

```
        print('---------')

    return sample_Chi
```

Example

In [50]:

```
beta_arry=np.linspace(1/1,1/15,150)

T_array=1/beta_arry
L_size =5

spins = generate_random_spin_configuration(L_size)
sample_Chi=Gibbs_step_with_MS(spins_init=spins,beta_arry=beta_arry,J=J_value, B=B_value,grid_size=L_s
# Create scatter plot
plt.scatter(T_array, (sample_Chi), color='tab:red', marker='o' )

# Add labels and title
plt.xlabel('Temperature (T)')
plt.ylabel(' Magnetic Susceptibility ($\chi$)')
plt.title('Magnetic Susceptibility v.s. Temperature')


# Show the plot
plt.show()
```

```
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
```

```
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
```

```
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
```

```
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
499.92
12498
312404
0.0002573455749466049
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
```

```
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
499.92
12498
312404
0.00023534045637815562
---------
```

```
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
499.92
12498
312404
0.00022733859508053771
---------
499.92
12498
312404
0.00022533812975613325
---------
499.92
12498
312404
0.00022333766443172876
---------
499.92
12498
312404
0.00022133719910732427
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
```

```
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
500.0
12500
312500
0.0
---------
499.91999999999996
12498
312404
0.00020533347651208842
---------
499.92
12498
312404
0.00020333301118768396
---------
500.0
12500
312500
0.0
---------
500.0
```

```
12500
312500
0.0
---------
499.92
12498
312404
0.00019733161521447048
---------
500.0
12500
312500
0.0
---------
499.92
12498
312404
0.00019333068456566156
---------
499.92
12498
312404
0.0001913302192412571
---------
499.76000000000005
12494
312212
0.0005657127516784076
---------
499.84000000000003
12496
312308
0.0003739077583868432
---------
500.0
12500
312500
0.0
---------
499.91999999999996
12498
312404
0.00018332835794363913
```

```
---------
499.92
12498
312404
0.00018132789261923466
---------
499.84000000000003
12496
312308
0.0003579361073802352
---------
499.92
12498
312404
0.0001773269619704257
---------
499.84000000000003
12496
312308
0.00034995028187693094
---------
499.92
12498
312404
0.00017332603132161673
---------
499.92
12498
312404
0.00017132556599721226
---------
499.68000000000006
12492
312116
0.0006732284563734985
---------
499.68000000000006
12492
312116
0.0006652747024585373
---------
499.84
12496
```

```
312308
0.0003299857181186 7095
---------
499.6
12490
312028
0.001137374496644842
---------
499.6000000000001
12490
312020
0.0008001503355694663
---------
499.84000000000003
12496
312308
0.0003180069798637 1487
---------
499.36000000000007
12484
311732
0.0012409230604036455
---------
499.68000000000006
12492
312116
0.0006175521789687702
---------
499.20000000000005
12480
311540
0.0015055606263979764
---------
499.6
12490
312020
0.0007505395973144 95
---------
499.68
12492
312124
0.0008929302908271052
---------
```

```
499.2000000000001
12480
311540
0.0014466290827738243
---------
498.8000000000001
12470
311084
0.00299230067113888
---------
498.7200000000001
12468
310972
0.00251138634451904
---------
498.9600000000001
12474
311268
0.00235936902013375
---------
499.36000000000007
12484
311740
0.0013781046979867773
---------
498.56000000000006
12464
310820
0.004038625073823951
---------
498.72000000000014
12468
310964
0.0020999926263988652
---------
498.08000000000015
12452
310220
0.003853628134229609
---------
498.0000000000001
12450
310132
```

```
0.004177646532440662
---------
498.24000000000007
12456
310388
0.002725220223714329
---------
497.7600000000001
12444
309852
0.004647647355701697
---------
498.56
12464
310780
0.0024299531454118114
---------
497.04000000000013
12426
308964
0.004727710926174884
---------
496.32
12408
308164
0.0075083356778524344
---------
497.0400000000001
12426
308980
0.00505253082774145
---------
496.5600000000001
12414
308436
0.0065006449395970224
---------
496.1600000000001
12404
307924
0.005938017932885071
---------
-495.0400000000002
```

```
12376
306692
0.010253947704699291
---------
-494.40000000000015
12360
305844
0.008498942281879198
---------
-494.80000000000007
12370
306380
0.009479838926174777
---------
-494.56000000000023
12364
306084
0.009328071874719278
---------
-494.7200000000002
12368
306228
0.00773344157494461
---------
491.1200000000001
12278
302276
0.020119314040267837
---------
492.40000000000015
12310
303604
0.013496060850111237
---------
490.0000000000002
12250
300860
0.018284563758389598
---------
489.2800000000002
12232
300332
0.026529493046980913
```

```
---------
487.44000000000017
12186
297860
0.020599299722595783
---------
-487.44000000000005
12186
298100
0.02577659504250649
---------
477.5199999999998
11938
287540
0.057371324134227936
---------
121.20000000000016
11902
286660
0.07482756152125371
---------
478.0
11950
288412
0.06138978076062675
---------
473.0400000000002
11826
282044
0.049906486120805046
---------
-122.8000000000001
11460
269324
0.13899763400447418
---------
463.20000000000016
11580
272812
0.09407874004474341
---------
-395.9999999999996
11250
```

261020
0.15684026845637547

-440.8800000000001
11026
252996
0.19075438274720363

-430.3199999999997
10844
247284
0.22823609327964225

-44.079999999999956
10444
231108
0.23786174367785154

76.48000000000012
10626
236924
0.1982648322863538

387.19999999999965
10444
232148
0.2429336225503346

-197.6799999999998
9876
210684
0.26322469111409436

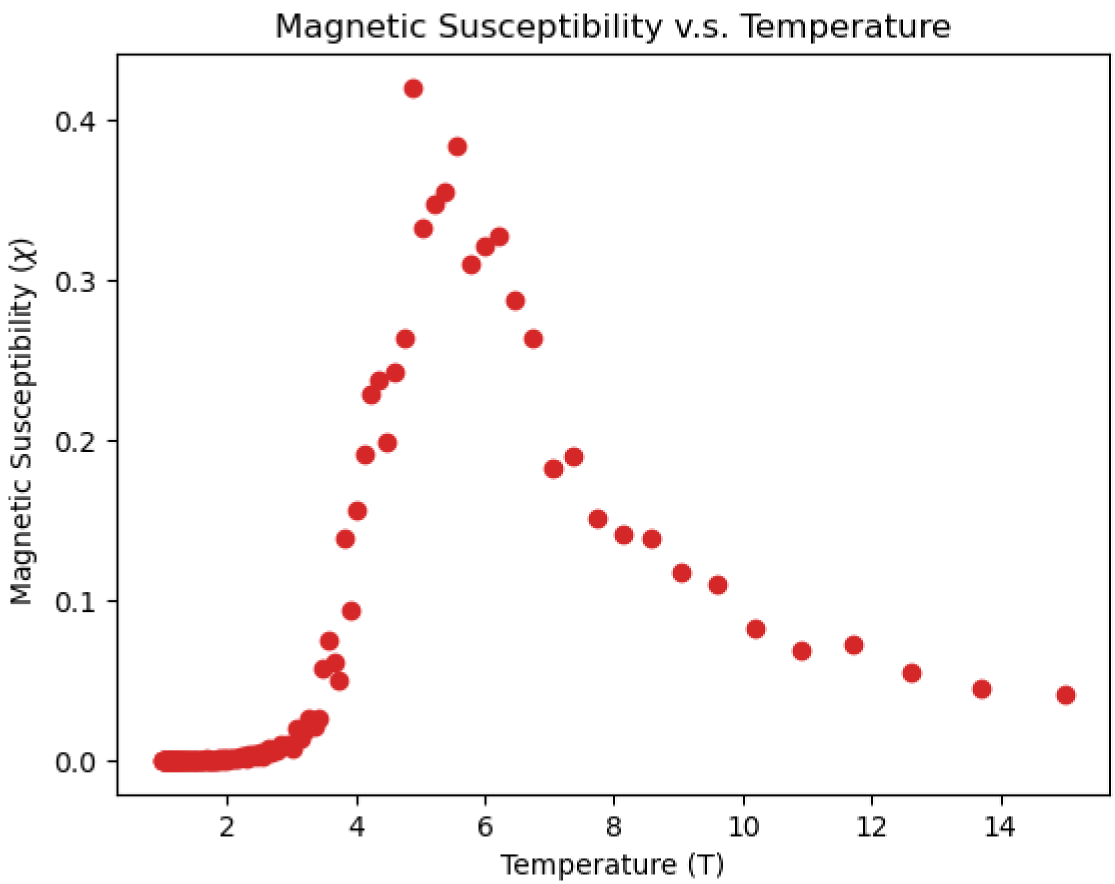
71.36
8550
171892
0.4201864519015655

163.12000000000003
9066
185396
0.33317228041163227

```
-154.88000000000002
8416
164324
0.3480506828456375
---------
11.439999999999984
7742
143772
0.3549484341834449
---------
142.08000000000018
7768
147396
0.3834148779239372
---------
118.56000000000016
7490
134604
0.310345256375839
---------
-13.520000000000016
5780
90892
0.3214335427293063
---------
36.72000000000004
6178
101804
0.32727477004026834
---------
7.119999999999987
5706
88412
0.28767137932885894
---------
-29.280000000000022
5560
84116
0.2640749100671141
---------
-9.520000000000007
4528
57108
```

```
0.182710369360179
---------
-4.080000000000007
4518
58356
0.19013869014765092
---------
-18.159999999999986
4064
47596
0.15065558879642046
---------
-3.919999999999997
4078
47636
0.14150707114093952
---------
-27.360000000000028
3858
44668
0.1391969357315435
---------
-23.200000000000003
3768
41660
0.11727230224608494
---------
-13.439999999999994
3660
40004
0.11019534317673371
---------
14.000000000000002
3160
30452
0.08215821744966434
---------
-2.880000000000008
3030
27748
0.06887412975391496
---------
-14.719999999999985
```

```
3060
29236
0.07184539776286353
---------
15.439999999999998
2964
26260
0.05505235672483215
---------
3.2000000000000046
2696
22316
0.04538718153020129
---------
-11.199999999999983
2816
23516
0.040833536
---------
```


Magnetic Susceptibility v.s. Temperature
Magnetic Susceptibility (χ)
0.4
0.3
0.2
0.1
0.0
2
4
6
8
10
12
14
Temperature (T)

```
In [ ]:
```