MSE 203
Introduction to Computational Materials

Assignment 2

Dreamy Jain (23110106)
Panjari Patel (23110236)
Course Instructor: Prof. Raghavan Ranganathan

Department of Material Science and Engineering
IIT Gandhinagar

Computational Studies of Molecular and Statistical Mechanics Using Monte Carlo Simulations

1.Simulation of a Binary Hard Sphere System Using Towhee

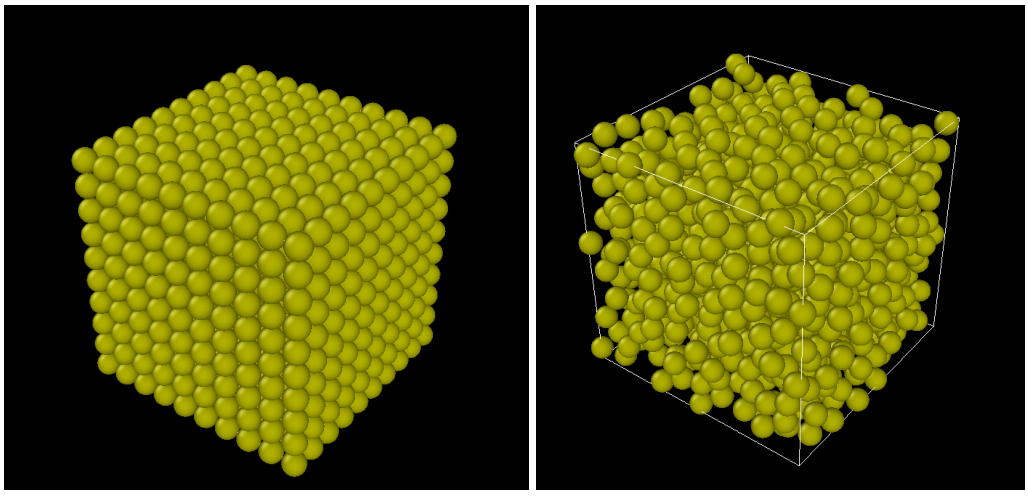
Towhee is an open-source Monte Carlo molecular simulation package designed for studying fluid-phase equilibria, adsorption, and molecular interactions. It supports simulations in canonical (NVT), isothermal-isobaric (NPT), and grand canonical (μVT) ensembles. In this assignment, we use Towhee to conduct an NVT simulation of a binary hard sphere system. The simulation outputs a radial distribution function (RDF), which provides insights into particle correlations, and atomic density profiles, which reveal spatial distributions. We analyze the results and compare the hard sphere system with the Lennard-Jones (LJ) potential.

1.1 Hard Sphere Model

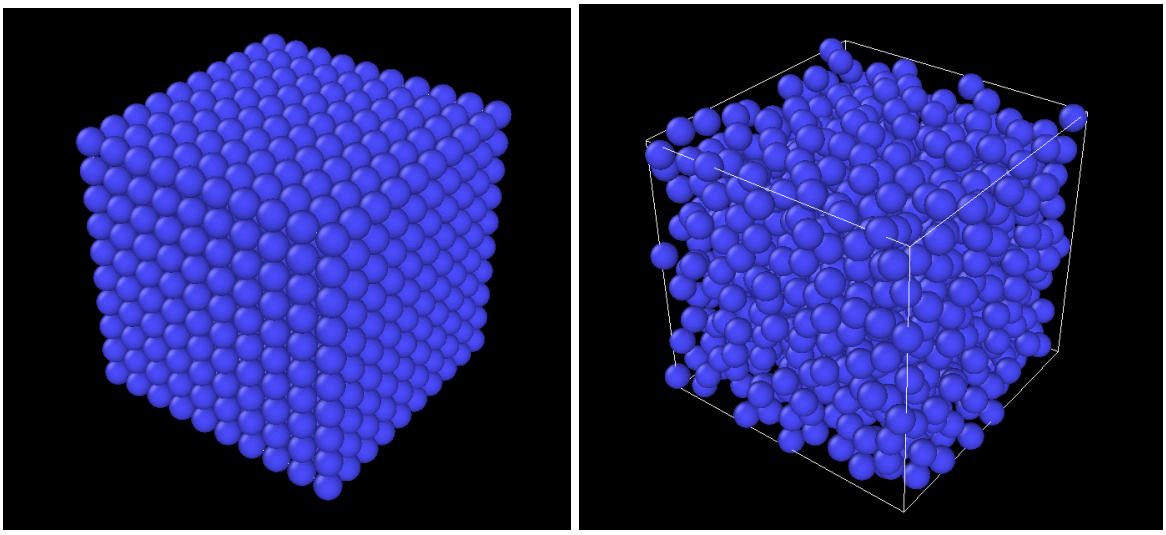
The hard sphere model represents particles as impenetrable spheres with no attractive interactions. The potential energy *U(r)* is given by:

$$U(r) = \begin{cases} \infty & \text{if } r \leq \sigma \\ 0 & \text{if } r > \sigma \end{cases}$$

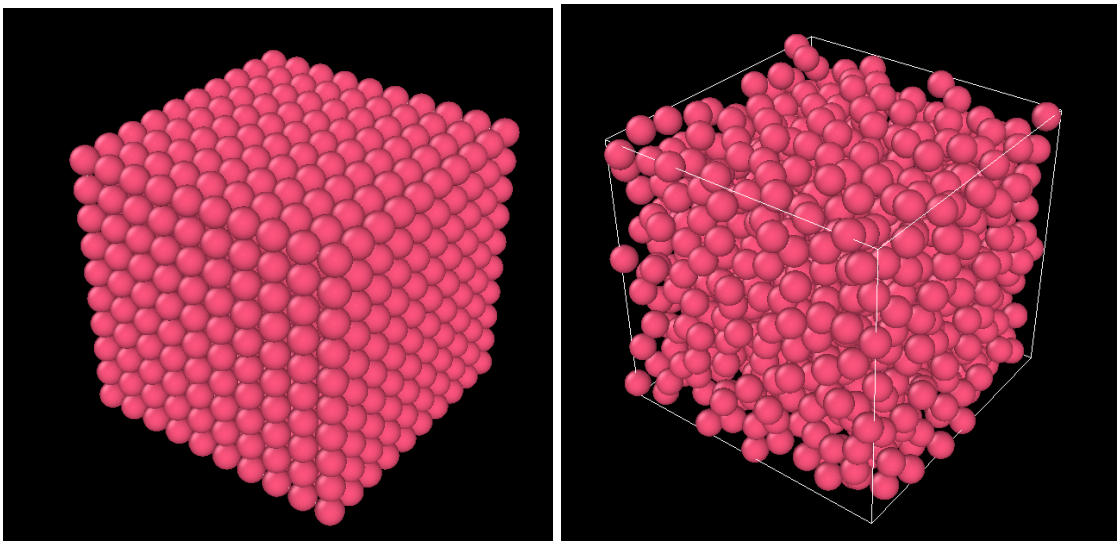
where σ is the diameter of the spheres. The hard sphere model is widely used to study packing, phase transitions, and structural properties of fluids.



300K



500K



800K

1.2 Radial Distribution Function (RDF)

The radial distribution function g(r) describes how particle density varies as a function of distance from a reference particle. It is defined as:

$$g(r) = \frac{\rho(r)}{\rho_0}$$

where ρ(r) is the local density at distance r, and ρ_0 is the bulk density. The first peak in g(r) corresponds to the most probable distance between neighboring particles.

The first peak in a radial distribution function (RDF) represents the most probable nearest-neighbor distance between atoms or molecules. Its position indicates the average bond length or coordination distance in the system.

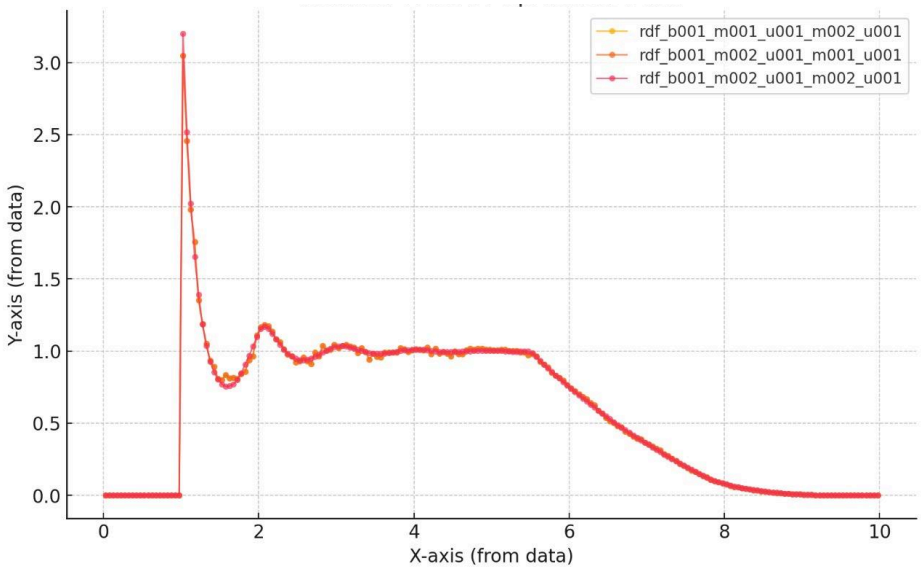
1.2.1 Temperature dependence of RDF

To understand the effect of temperature on the hard sphere system, we conduct simulations at three different temperatures: 300K, 500K, and 800K. Higher temperatures generally reduce ordering and weaken correlations, leading to a broader and less pronounced RDF peak

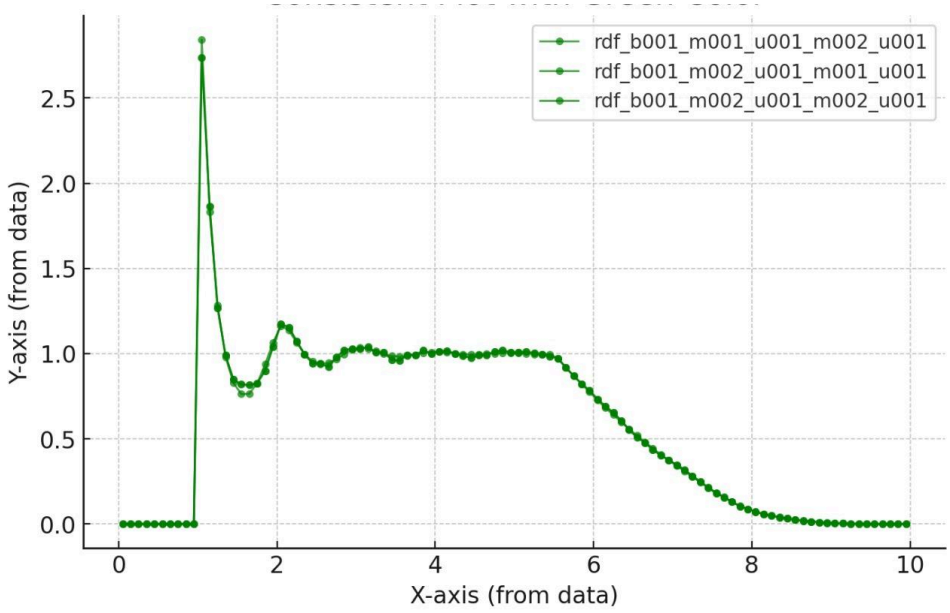
1.2.2 Comparison of RDFs

- The RDF for the hard sphere system shows sharp peaks corresponding to close-packed structures.
- The RDF for the LJ system exhibits additional structure due to attractive forces, leading to a more extended ordering.

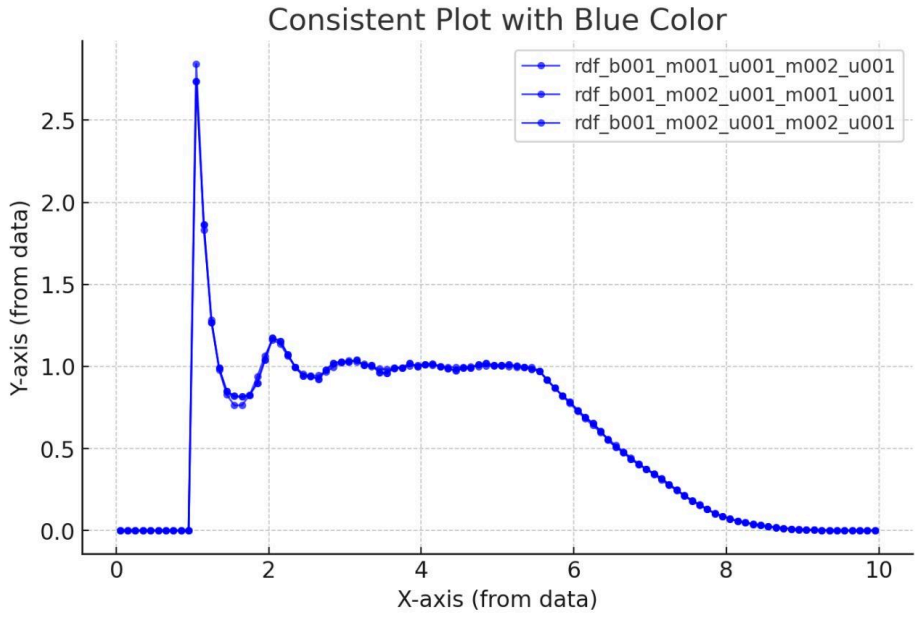
The RDF plots for both the hard sphere and Lennard-Jones (LJ) systems are presented below, comparing the differences in particle correlations.



RDF at 300K



RDF at 500K



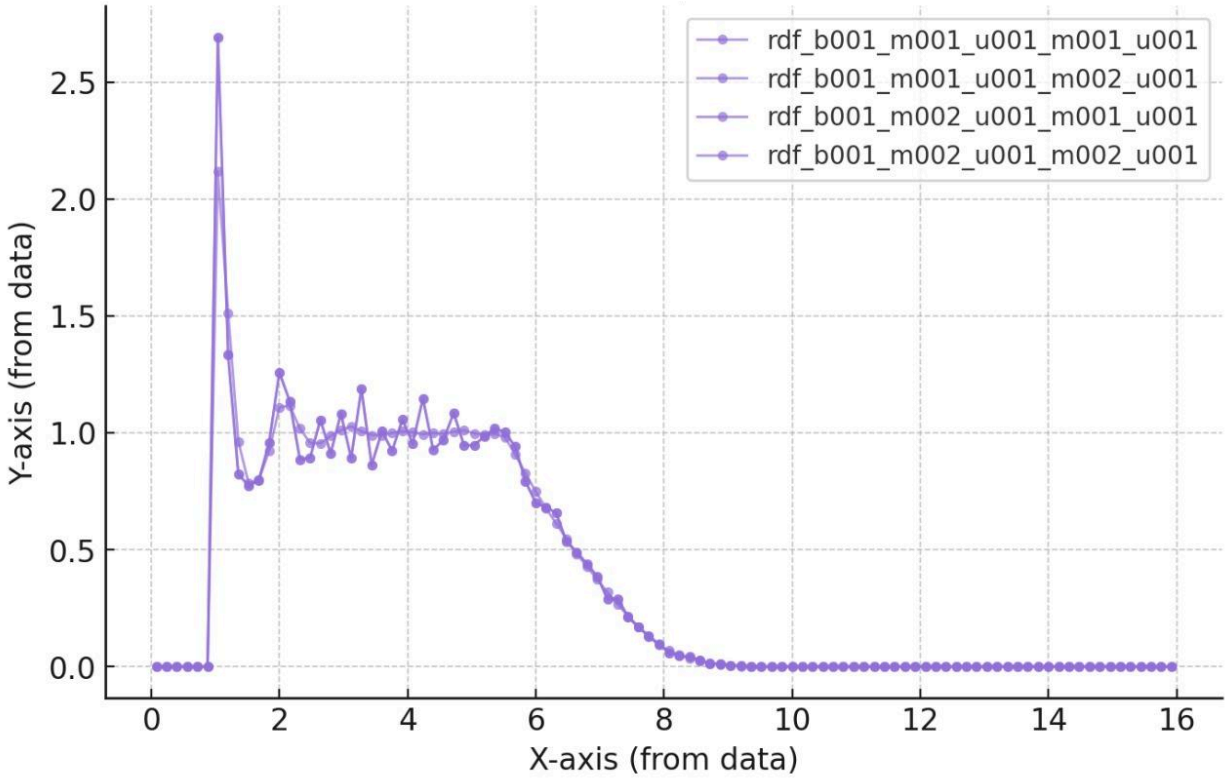
RDF at 800K

1.3 Lennard-Jones (LJ) Potential

The Lennard-Jones potential describes interactions between neutral atoms or molecules and is given by:

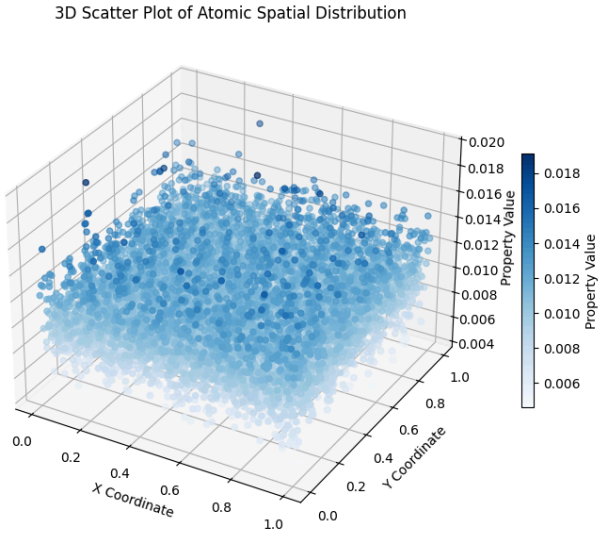
$$U(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right]$$

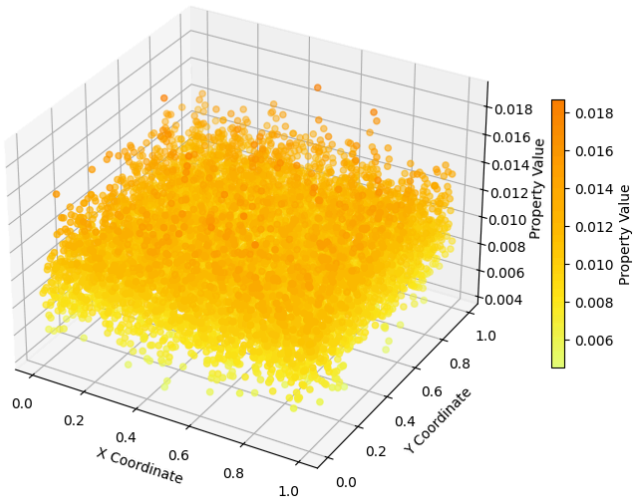
where ϵ is the depth of the potential well, and σ is the distance at which the potential is zero. The LJ potential is used to model van der Waals interactions and is more realistic than the hard sphere model.

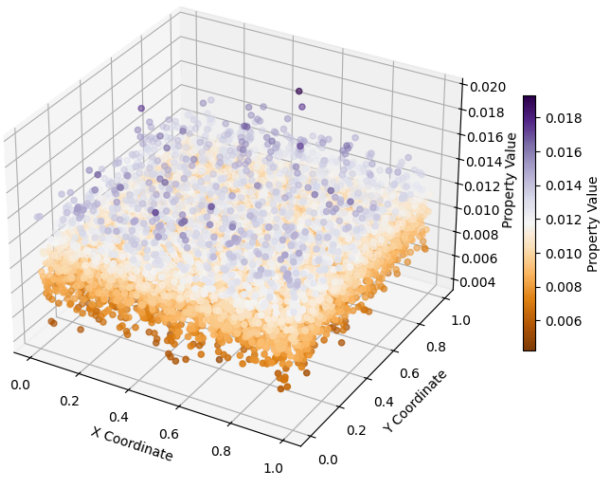


1.4 . Atomic Spatial Distribution

The spatial distribution function provides information about the spatial arrangement of atoms within the simulation box. This is useful for visualizing phase behavior and identifying structural transitions. The post-processing tool analyse_movie is used to extract and visualize spatial distributions from Towhee simulations. The atomic spatial distribution plots and snapshots are provided below to illustrate the system's structure.

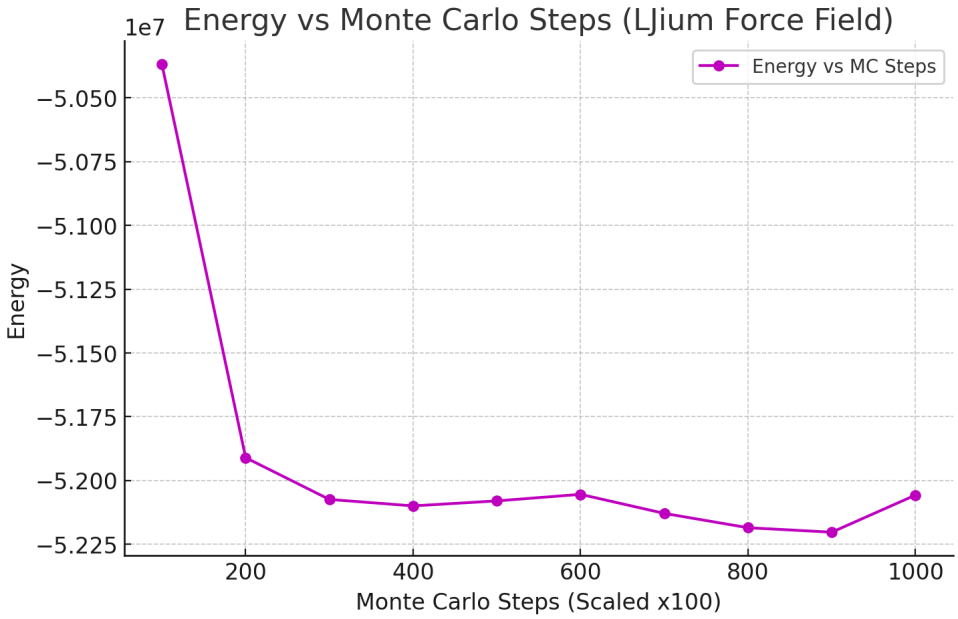

3D Scatter Plot of Atomic Spatial Distribution
X Coordinate
Y Coordinate
Property Value
0.0
0.2
0.4
0.6
0.8
1.0
0.004
0.006
0.008
0.010
0.012
0.014
0.016
0.018
0.020


3D Scatter Plot of Atomic Spatial Distribution
X Coordinate
Y Coordinate
Property Value
0.0
0.2
0.4
0.6
0.8
1.0
0.004
0.006
0.008
0.010
0.012
0.014
0.016
0.018


3D Scatter Plot of Atomic Spatial Distribution
X Coordinate
Y Coordinate
Property Value
0.0
0.2
0.4
0.6
0.8
1.0
0.004
0.006
0.008
0.010
0.012
0.014
0.016
0.018
0.020

1.5 Energy vs. Monte Carlo Steps for LJium Force Field

The energy of the LJ system fluctuates due to attractive interactions, whereas the hard sphere system has a constant energy as there are no attractive interactions. A plot of energy vs. Monte Carlo steps for the LJ system is included below, demonstrating system equilibration and energy fluctuations.



1.6 Comparison Between Hard Sphere and Lennard-Jones (LJ) Potentials

The hard sphere (HS) model and Lennard-Jones (LJ) potential differ significantly in their interaction characteristics, phase behavior, computational cost, and applications.

Forces: The HS model includes only steric repulsion, meaning particles do not experience attraction beyond direct collisions. In contrast, the LJ model accounts for both short-range repulsion (due to electron cloud overlap) and long-range attraction (from van der Waals forces).

Phase Behavior: The HS model supports only gas and solid phases, as particles cannot attract each other to form liquids. The LJ model, however, allows for gas-liquid-solid transitions, enabling simulations of condensation, evaporation, and adsorption phenomena.

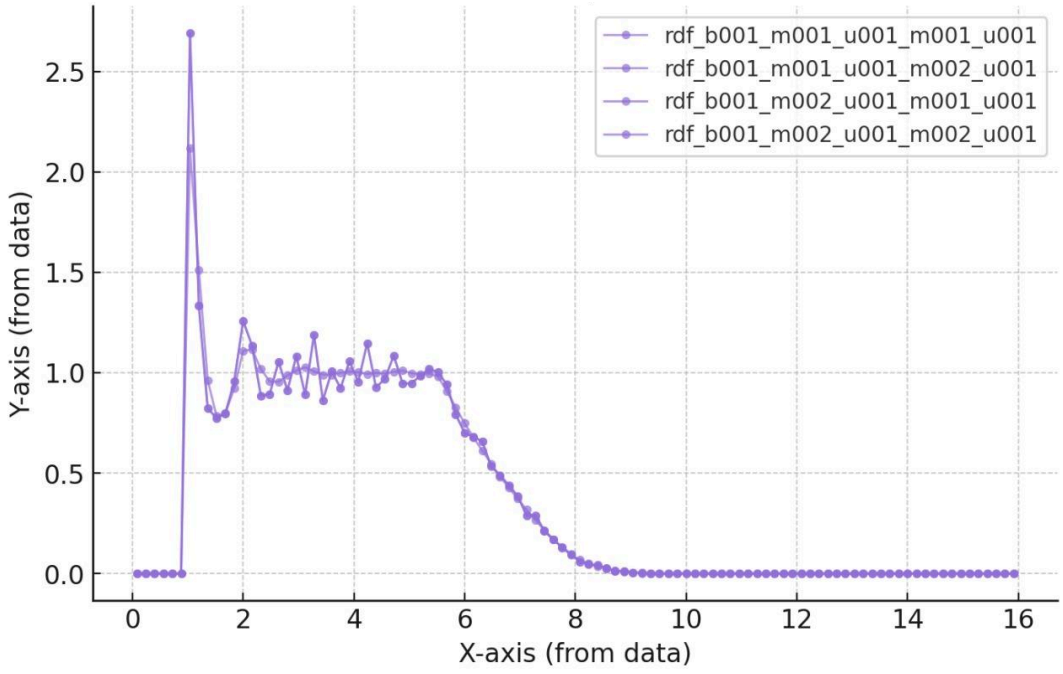
Computational Cost: The HS model is computationally efficient, as interactions occur only during direct collisions, eliminating the need for continuous force calculations. This leads to faster Monte Carlo and event-driven Molecular Dynamics simulations. In contrast, the LJ model requires continuous force evaluations, making it computationally more expensive. The cost of an

LJ simulation scales as O(N^2) without optimizations, whereas HS systems can be simulated in O(N) time using Monte Carlo methods.

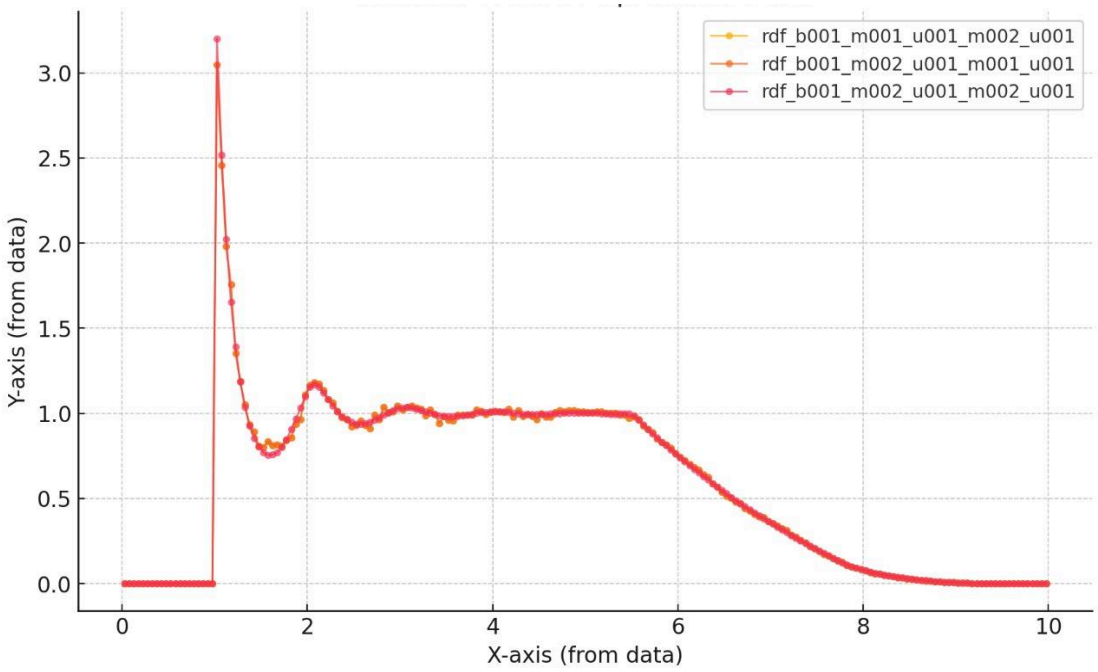
Radial Distribution Function

The radial distribution function (g(r)) describes how particle density varies as a function of distance from a reference particle. It provides insights into molecular structure in fluids and solids. When comparing Hard Sphere (HS) and Lennard-Jones (LJ) potentials, g(r) reflects their fundamental interaction differences.

Feature	**Hard Sphere (HS)**	**Lennard-Jones (LJ)**
Short-range behavior	Strictly zero for r<σ	Small but nonzero for r<σ
First peak position	Exactly at r=σ	Slightly beyond σ(~1.1σ)
Decay of oscillations	Faster (short-range ordering)	Slower (long-range correlations due to attraction)
Effect of density	Stronger peaks with increasing density	Similar trend, but attraction broadens peaks



LJ Pontential



HS Potential

We can see the difference between the HS and LJ potentials form the above graphs.

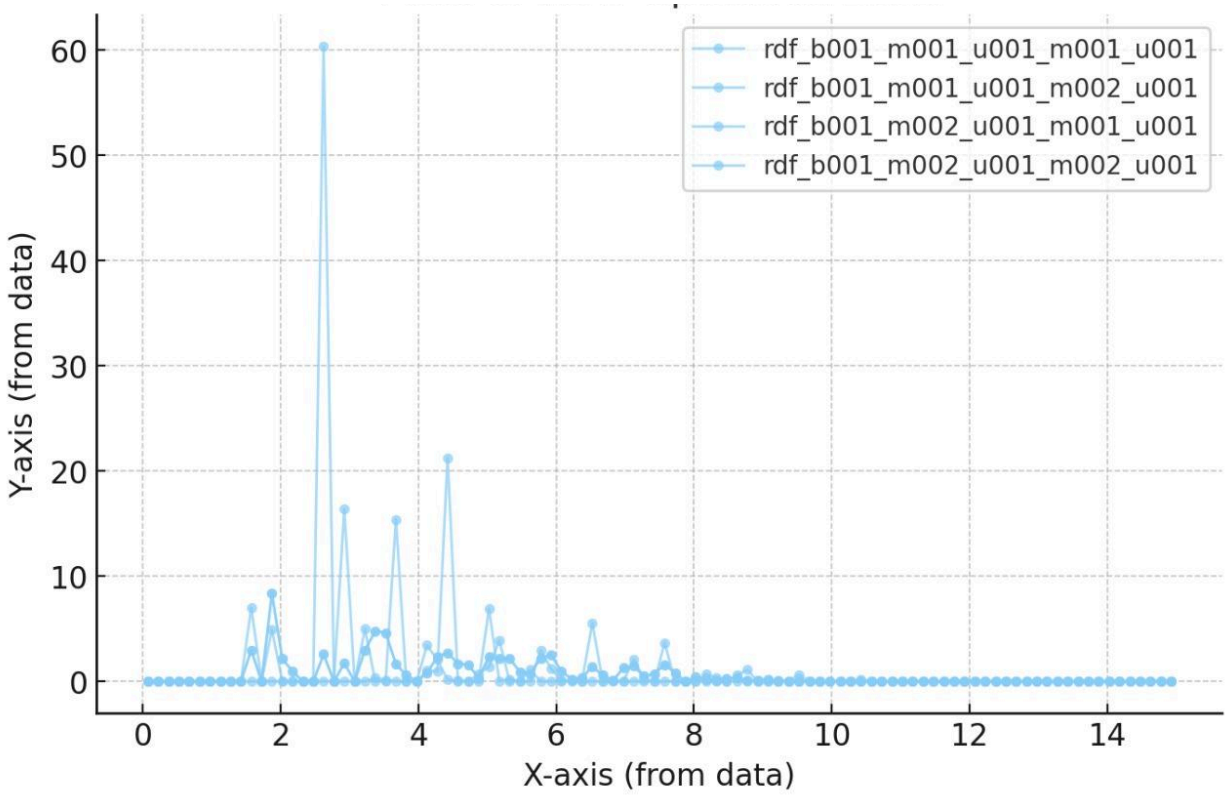
Practical Applications:

- The HS model is widely used in studying colloidal suspensions, granular materials, and high-density simple liquids. It is also valuable for investigating random packing and jamming transitions.
- The LJ model is essential for simulating realistic molecular fluids (such as argon and methane), predicting liquid-gas phase behavior, and modeling biomolecular systems.

2. NPT Simulation of an Au-Cu Mixture Using Towhee

2.1 Atomic Spatial Distribution and RDF Analysis

The atomic position data from the Towhee output files are analyzed to generate spatial distribution plots and RDFs for the Au-Cu mixture. These results are compared with the hard sphere system studied in Section 1.



RDF

2.2 Influence of the Center-of-Mass Switch Move

The center-of-mass switch move plays a crucial role in the mixing behavior of Au and Cu atoms. This move enhances atomic rearrangements, allowing efficient sampling of phase space and leading to a more thermodynamically favorable mixture. It improves diffusion and mixing, ultimately affecting the system's structural and energetic properties.

2.3 Temperature Effects on Phase Behavior

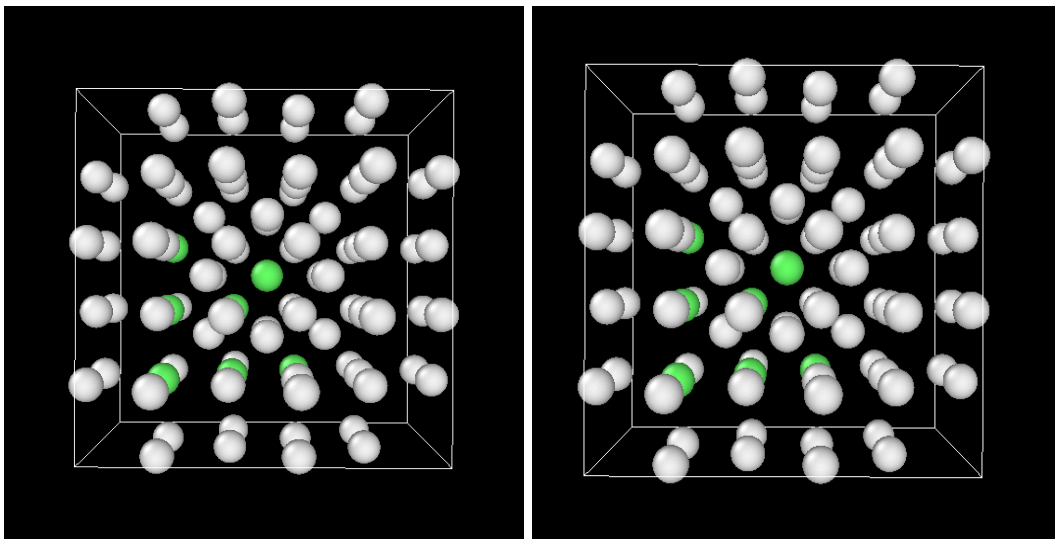
The Au-Cu system is known for its complex phase behavior, which is strongly temperature-dependent. At high temperatures, the mixture tends to form a disordered solid solution due to the increased thermal energy overcoming atomic ordering tendencies. As the temperature decreases, ordered intermetallic phases such as AuCu, $AuCu_3$, and Au_3Cu can form, depending on the composition. These ordered structures arise due to differences in atomic size, electronegativity, and enthalpic contributions to the Gibbs free energy.

Key temperature-dependent phase behavior includes:

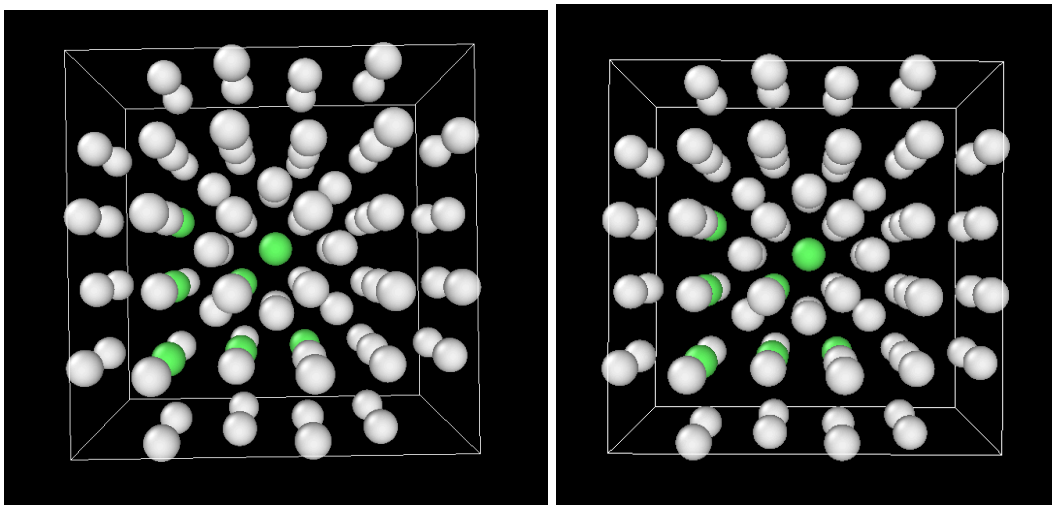
- **Above the melting point (~1000°C)**: The mixture is in a liquid state, with complete miscibility.

- **Moderate temperatures (500-800°C)**: The system forms a face-centered cubic (FCC) disordered solid solution.
- **Low temperatures (<500°C)**: Ordered intermetallic phases (such as AuCu, which adopts an $L1_0$ structure) can form due to ordering tendencies.

Snapshots of the initial and final configurations, obtained using Ovito, demonstrate the evolution of atomic arrangements.



INITIAL MODEL

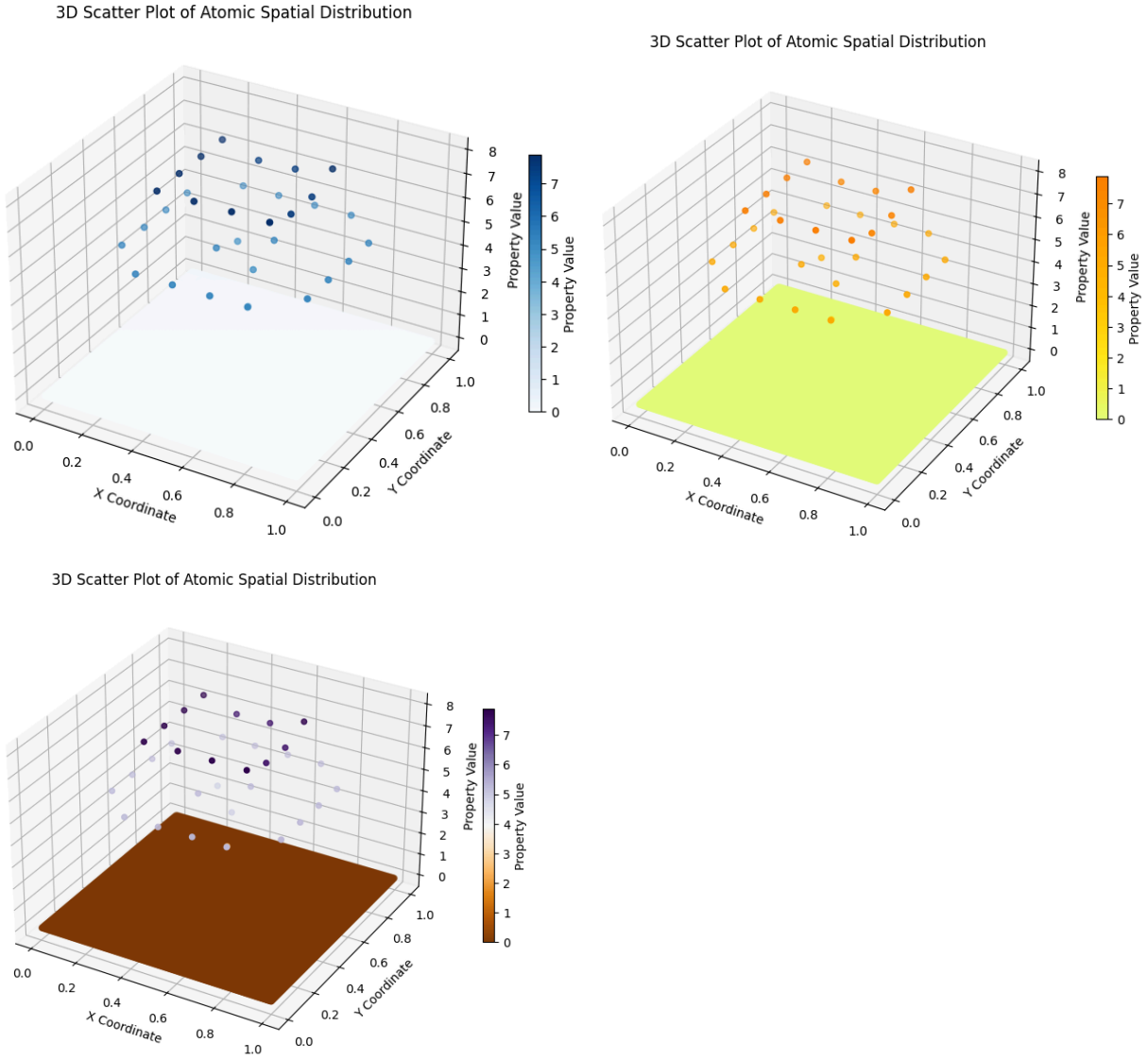


FINAL MODEL

2.4 Hard Sphere (HS) potential instead of the Embedded Atom Method (EAM)

Using a Hard Sphere (HS) potential instead of the Embedded Atom Method (EAM) for Au and Cu would remove metallic bonding, leading to zero cohesion and incorrect material behavior. The mechanical properties (elasticity, plasticity, bulk modulus) would be unrealistic, as HS lacks attractive forces. The radial distribution function (g(r)) would show only short-range order, missing the long-range metallic structure. Thermal effects like expansion and melting wouldn't be captured, as HS does not model interatomic forces. Diffusion would be purely collisional rather than vacancy-driven, making HS unsuitable for metals.

2.5 Atomic Spatial Distribution



2.6 Effects of Using a Hard Sphere Potential Instead of an Embedded Atom Method (EAM) Potential

The Embedded Atom Method (EAM) is typically used for modeling metallic systems because it captures many-body interactions, cohesive energy, and metallic bonding effects. If a hard sphere

(HS) potential were used instead, significant differences in the predicted phase behavior would arise:

1. **Absence of Bonding and Cohesion**
    - The HS potential only considers excluded volume effects and lacks attractive interactions, which are crucial for describing metallic bonding.
    - Without cohesive forces, the model would fail to predict ordered intermetallic phases such as AuCu or AuCu3_33.
2. **Overestimation of Mixing and Entropic Contributions**
    - Hard sphere systems tend to exhibit ideal mixing behavior at all temperatures since they do not account for electronic effects or bond formation.
    - This would lead to an overprediction of disorder and prevent phase separation or ordering transitions.
3. **No Thermal Expansion Effects**
    - The EAM potential accounts for changes in atomic interactions with temperature, allowing for the prediction of thermal expansion and phase transitions.
    - The HS potential assumes fixed atomic radii, meaning it cannot model temperature-induced structural changes accurately.
4. **Phase Diagram Differences**
    - The actual Au-Cu phase diagram includes ordered phases and phase boundaries, which the HS potential would fail to capture.
    - A hard sphere model would predict only simple mixing behavior and might inaccurately suggest complete miscibility in the solid state.

2.7 Conclusion

This study highlights the structural differences between hard sphere and Lennard-Jones fluids using Monte Carlo simulations in Towhee. The RDF analysis and energy plots provide insights into interparticle interactions, phase behavior, and computational efficiency. The Au-Cu mixture simulation demonstrates the role of the center-of-mass switch move in enhancing atomic mixing. Understanding these potentials is crucial for applications in material science, chemical engineering, and condensed matter physics.

3. Simulation of the 2D Ising Model Using the Metropolis Algorithm

The 2D Ising model is a mathematical model used in statistical mechanics to study phase transitions in magnetic systems. It consists of a lattice of spins that can be either +1 or -1. The Metropolis algorithm is a Monte Carlo method used to simulate the behavior of such systems by randomly flipping spins and accepting or rejecting the flips based on energy changes. In this assignment, we simulate the 2D Ising model using the Metropolis algorithm and analyze the

results, including magnetization, spin configurations, and specific heat as functions of temperature.

3.1 The 2D Ising Model

The Ising model is defined on a 2D lattice where each site has a spin S_i that can take values of +1 (up) or -1 (down). The energy of the system is given by:

$$E=-J\langle i,j\rangle\sum Si\,S_j$$

where J is the interaction strength (set to 1 for simplicity), and the sum is over nearest-neighbor pairs. The system exhibits a phase transition at a critical temperature T_c, where it transitions from an ordered (ferromagnetic) phase to a disordered (paramagnetic) phase.

3.2 The Metropolis Algorithm

The Metropolis algorithm is a Monte Carlo method used to simulate the Ising model. At each step:

1. A random spin is selected, and its flip is proposed.
2. The change in energy ΔE is calculated.
3. If $\Delta E \leq 0$, the flip is accepted. If $\Delta E > 0$, the flip is accepted with probability $\exp\{-\Delta E/k_B T\}$ where k_B is the Boltzmann constant and T is the temperature.

This process is repeated for many steps until the system reaches equilibrium.

3.3 Quantities of Interest

- **Magnetization (M)**: The average magnetization per spin is given by:

$$\langle M\rangle = \frac{1}{N}\left\langle \sum_i S_i \right\rangle$$

  where N is the total number of spins. Magnetization measures the degree of alignment of spins.

- **Specific Heat (C_v):** The specific heat is calculated as:

$$C_v = \frac{\langle E^2\rangle - \langle E\rangle^2}{k_B T^2}$$

  It indicates the system's heat capacity and peaks at the critical temperature T_c, signaling a phase transition.

3.4 Simulation Setup

- A 2D lattice of size 20×20 was initialized with random spins.
- The Metropolis algorithm was implemented to update the spins at different temperatures.

- The simulation was run for 10,000 Monte Carlo steps at each temperature, and data was collected for magnetization, energy, and specific heat.

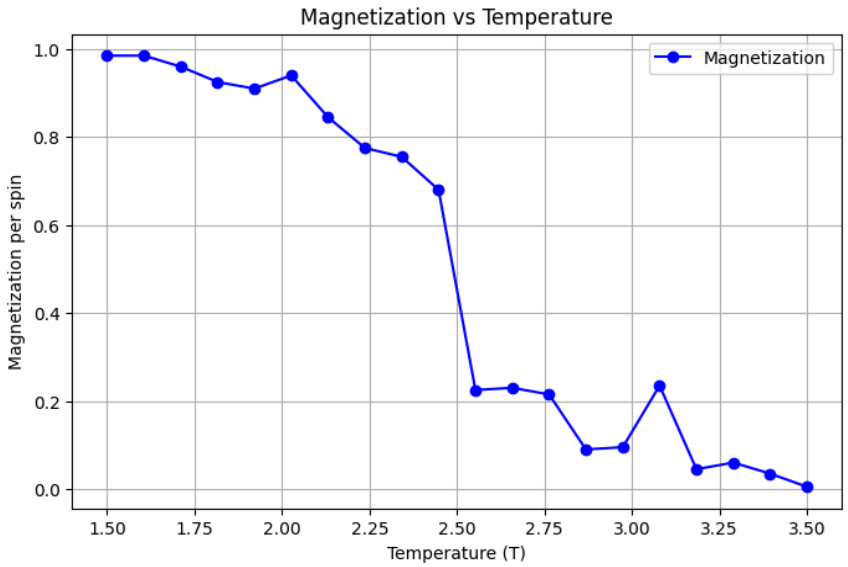
3.5 Data Collection

- The average magnetization per spin was calculated for each temperature.
- The specific heat was computed using the fluctuations in energy.
- Spin configurations were visualized at different temperatures to observe the ordering of spins

3.6 Average Magnetization vs Temperature

The average magnetization per spin was computed as a function of temperature. Near the critical temperature T_c, the magnetization drops sharply, indicating a phase transition from an ordered to a disordered state.

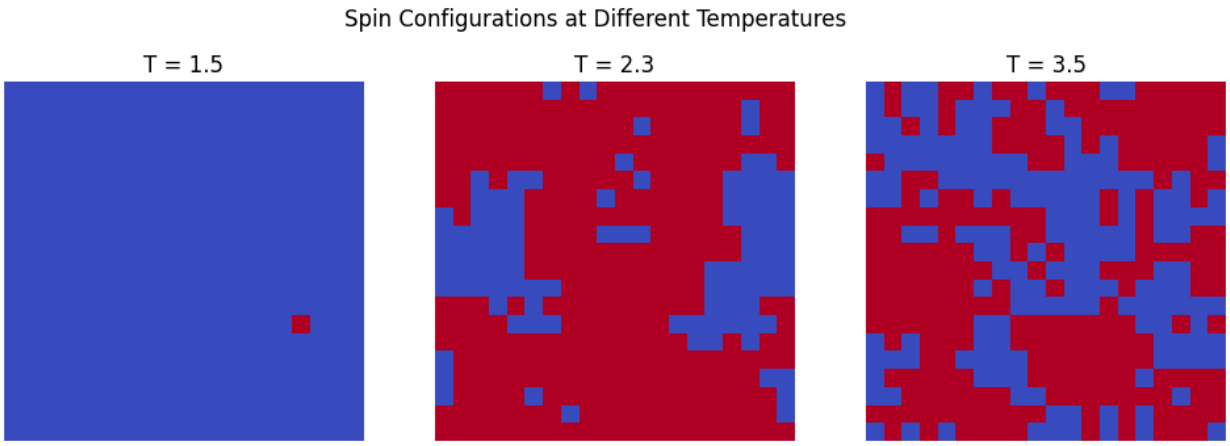
The T_c is at 2.69



3.7 Spin Configurations

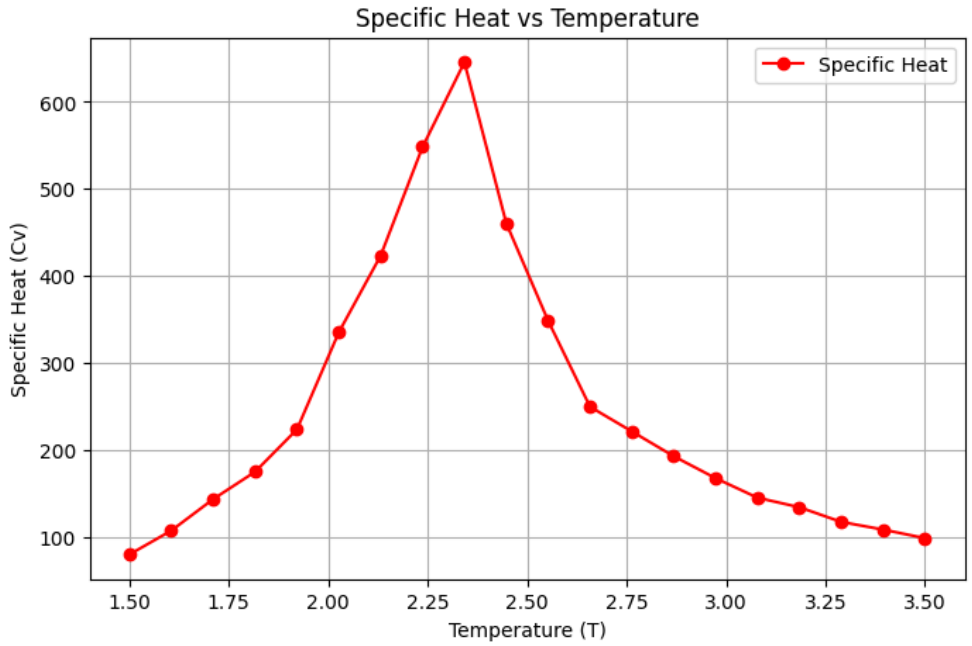
Spin configurations were visualized at different temperatures:

- **Below** *Tc* **:** Spins are mostly aligned, showing large domains of the same spin value.
- **Near** *Tc***:** Domains of different sizes appear, indicating the onset of disorder.
- **Above** *Tc***:** Spins are randomly oriented, showing a disordered state.



3.8 Specific Heat vs Temperature

The specific heat was calculated as a function of temperature. A peak in the specific heat is observed at the critical temperature T_c , indicating a phase transition.



Phase Transition Indication: The specific heat shows a peak at the critical temperature *Tc*, indicating a phase transition. This peak corresponds to the system absorbing energy to transition from an ordered to a disordered state

The self written code:

```
import numpy as np
import matplotlib.pyplot as plt
```

```
# Parameters
L = 20  # Lattice size (L x L)
J = 1   # Interaction strength
kb = 1  # Boltzmann constant
steps = 10000  # Monte Carlo Steps per temperature
temps = np.linspace(1.5, 3.5, 20)  # Temperature range

# Function to initialize a random spin lattice
def initialize_lattice(L):
    return np.random.choice([-1, 1], size=(L, L))

# Function to compute total energy of the system
def compute_energy(lattice):
    energy = 0
    for i in range(L):
        for j in range(L):
            S = lattice[i, j]
            neighbors = lattice[(i+1)%L, j] + lattice[i, (j+1)%L] +
lattice[(i-1)%L, j] + lattice[i, (j-1)%L]
            energy += -J * S * neighbors
    return energy / 2  # Each bond is counted twice

# Function to compute magnetization
def compute_magnetization(lattice):
    return np.abs(np.sum(lattice)) / (L * L)

# Metropolis algorithm for Monte Carlo simulation
def metropolis_step(lattice, T):
    for _ in range(L * L):  # Try L^2 spin flips per step
        i, j = np.random.randint(0, L, size=2)
        S = lattice[i, j]
        neighbors = lattice[(i+1)%L, j] + lattice[i, (j+1)%L] +
lattice[(i-1)%L, j] + lattice[i, (j-1)%L]
        dE = 2 * J * S * neighbors  # Energy difference

        if dE < 0 or np.random.rand() < np.exp(-dE / (kb * T)):
            lattice[i, j] *= -1  # Flip spin

# Running simulation and collecting data
```

```
magnetizations = []
specific_heats = []
energy_vals = []

for T in temps:
    lattice = initialize_lattice(L)
    energy_list = []

    # Equilibrating the system
    for _ in range(steps):
        metropolis_step(lattice, T)

    # Measuring properties
    for _ in range(steps):
        metropolis_step(lattice, T)
        energy = compute_energy(lattice)
        energy_list.append(energy)

    avg_energy = np.mean(energy_list)
    avg_energy_sq = np.mean(np.array(energy_list)**2)
    Cv = (avg_energy_sq - avg_energy**2) / (kb * T**2)

    magnetization = compute_magnetization(lattice)

    magnetizations.append(magnetization)
    specific_heats.append(Cv)
    energy_vals.append(avg_energy)

# Plotting Magnetization vs Temperature
plt.figure(figsize=(8, 5))
plt.plot(temps, magnetizations, 'bo-', label='Magnetization')
plt.xlabel('Temperature (T)')
plt.ylabel('Magnetization per spin')
plt.title('Magnetization vs Temperature')
plt.legend()
plt.grid()
plt.show()

# Plotting Specific Heat vs Temperature
plt.figure(figsize=(8, 5))
```

```
plt.plot(temps, specific_heats, 'ro-', label='Specific Heat')
plt.xlabel('Temperature (T)')
plt.ylabel('Specific Heat (Cv)')
plt.title('Specific Heat vs Temperature')
plt.legend()
plt.grid()
plt.show()

# Plotting Spin Configuration at Different Temperatures
plt.figure(figsize=(12, 4))
for i, T in enumerate([1.5, 2.3, 3.5]):  # Low, critical, and high temp
    lattice = initialize_lattice(L)
    for _ in range(steps):
        metropolis_step(lattice, T)

    plt.subplot(1, 3, i+1)
    plt.imshow(lattice, cmap='coolwarm')
    plt.title(f'T = {T}')
    plt.axis('off')

plt.suptitle('Spin Configurations at Different Temperatures')
plt.show()
```