Homework 02: Molecular Dynamics of Water

Katha Haldar

Department of Physics, Indiana University, Indianapolis, IN 46202

1 3D Visualization of Water Molecules

To understand the atomic distribution and dynamics, we visualize water molecules in 3D at equilibrium and production states. These images provide a qualitative view of the spatial arrangement of water molecules.

Equilibrium State

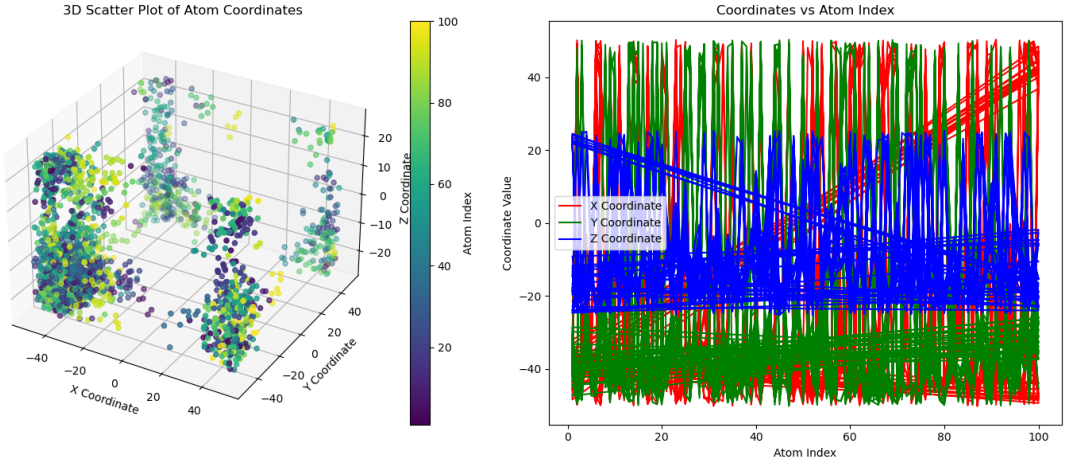


Figure 1: 3D visualization of water molecules in the equilibrium state. The arrangement represents a stable configuration under initial simulation conditions.

Production State

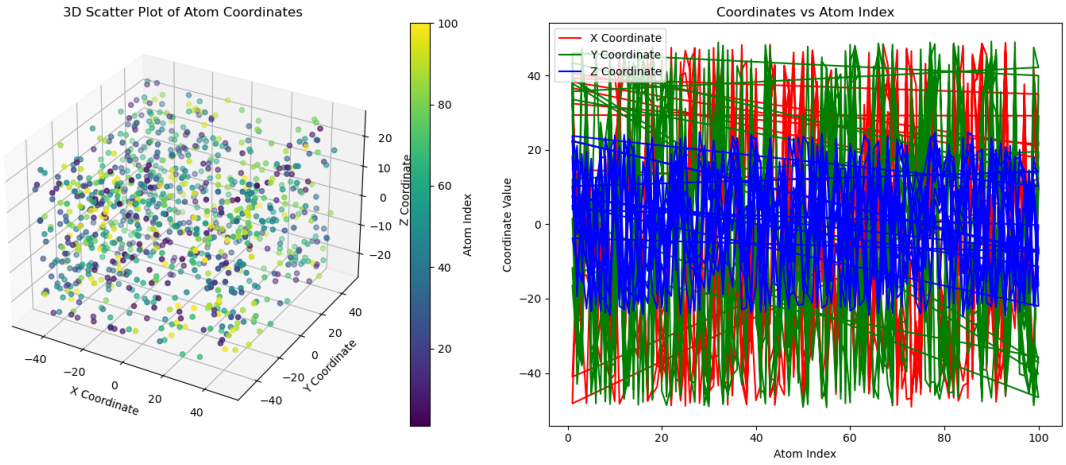


Figure 2: 3D visualization of water molecules in the production state. This snapshot reflects the dynamics after the system evolves under simulation.

2 Simulation Parameters

The following information has been extracted from the input files for the simulation:

- **Box Size:** $100 \times 100 \times 50$ Å
- **Number of Molecules:** 15,743
- **Temperature:** 303.15 K
- **Pressure:** 1 atm = 1.01325 bar

These parameters define the system configuration for the molecular dynamics simulation, where the box size represents the simulation cell dimensions, the number of molecules is the total count of particles, and the temperature and pressure conditions are set according to the input settings.

3 Equilibration and Production Times

In molecular dynamics simulations, the equilibration time and production run time vary depending on the system size, complexity, and research goals. Below is a breakdown of the general expectations for these times, as well as the calculation for the total simulation duration.

Given Data for Production

- Timestep: 2.0 fs/step
- Number of Steps: 500,000

Calculation for Production Time

$$\text{Total Duration (ps)} = \frac{\text{Timestep (fs)} \times \text{Number of Steps}}{1000}$$

Substituting the given values:

$$\text{Total Duration (ps)} = \frac{2.0 \times 500,000}{1000} = 1,000\,\text{ps}$$

Results

The production time is 1,000 ps (1 ns)

Given Data for Equilibration

- Number of Timesteps: 90,000,000
- Timestep Duration: 2.0 fs/step

Calculation for Equilibration Time

$$\text{Equilibration Time (ps)} = \frac{\text{Number of Timesteps} \times \text{Timestep Duration (fs)}}{1000}$$

Substituting the given values:

$$\text{Equilibration Time (ps)} = \frac{90,000,000 \times 2.0}{1000} = 180,000\,\text{ps} = 180\,\text{ns}$$

Result:

The equilibration time is 180,000 ps (180 ns).

Equilibration and Production Times of Water Box

The equilibration time is typically shorter and varies from tens to hundreds of nanoseconds depending on the system size and complexity. For example, the equilibration time could be around 180 ns for the system under study.
After equilibration, the production run focuses on data collection. In this case, the production time is set to 1 ns, which is shorter than the equilibration time. This duration allows for gathering meaningful statistics and data required for analysis.

Thus, for this simulation:

- **Equilibration Time:** 180 ns
- **Production Time:** 1 ns

4 Density Calculation of Water in the Simulation Box

Given:

- Number of water molecules: $N = 15743$
- Mass of one water molecule:

$$m = 18.015\,\text{amu} \times 1.66054 \times 10^{-24}\,\text{g/amu} = 2.991 \times 10^{-23}\,\text{g}.$$

- Volume of the simulation box:

$$V = 100\,\text{Å} \times 100\,\text{Å} \times 50\,\text{Å} = 500,000\,\text{Å}^3.$$

  Conversion to cm^3:

$$V = 500,000\,\text{Å}^3 \times 10^{-24}\,\text{cm}^3/\text{Å}^3 = 5.0 \times 10^{-19}\,\text{cm}^3.$$

Calculations

Total mass of water:

$$\text{Mass of water} = N \times m = 15743 \times 2.991 \times 10^{-23}\,\text{g} = 4.706 \times 10^{-19}\,\text{g}.$$

Density:

$$\text{Density} = \frac{\text{Mass of water}}{\text{Volume}} = \frac{4.706 \times 10^{-19}\,\text{g}}{5.0 \times 10^{-19}\,\text{cm}^3} = 0.941\,\text{g/cm}^3.$$

Result

The calculated density is:

$$\boxed{0.941\,\text{g/cm}^3}.$$

5 Energy vs Time: Equilibration and Production Phases

The plot below shows the energy of the system over time, with distinct phases for equilibration and production.

Equilibration Phase

The first graph (Figure 3) shows the energy evolution during the equilibration phase of the simulation. This phase is characterized by the system adjusting to the desired temperature and pressure before data collection begins.

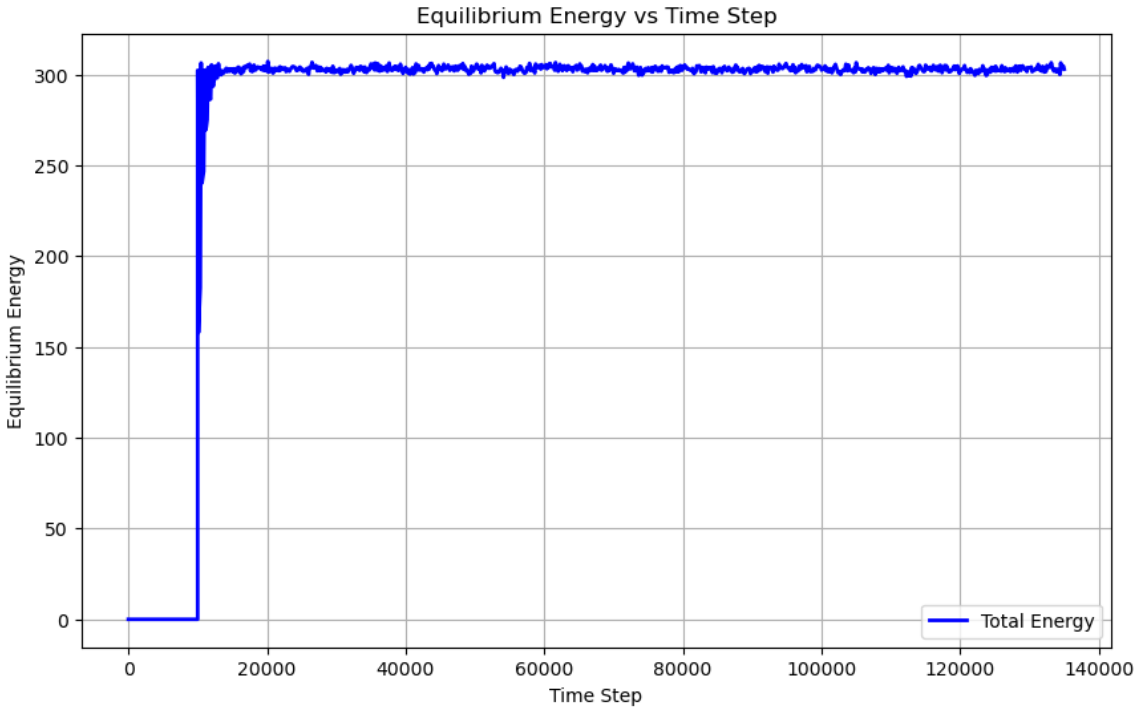


Figure 3: Energy vs Time during the equilibration phase.

Production Phase

The second graph (Figure 4) represents the energy evolution during the production phase of the simulation. This phase involves data collection for analysis, and it typically occurs after the system has equilibrated.

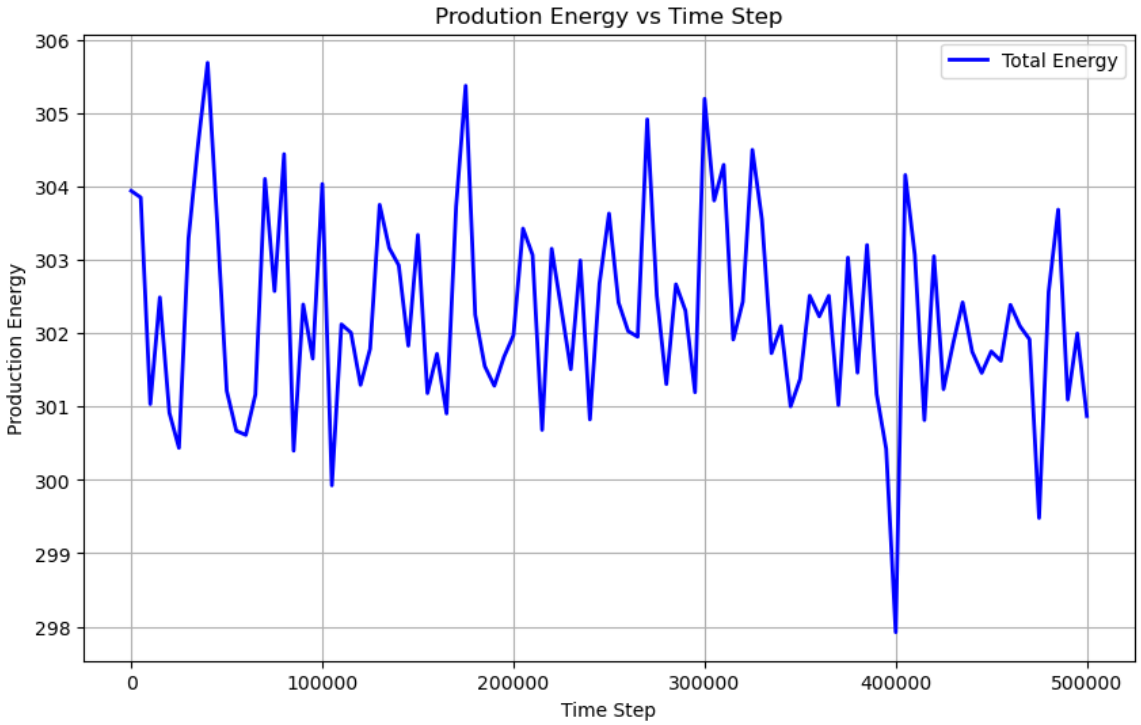


Figure 4: Energy vs Time during the production phase.

Python Code: Energy Analysis

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftfreq
from scipy.signal import correlate

# Load data function
def load_energy_data(filename):
    data = []
    with open(filename, 'r') as file:
        for line in file:
            if line.startswith("ENERGY:"):
                values = line.split()[1:]
                data.append([float(val) for val in values])
    return np.array(data)

# Load and extract relevant data
data = load_energy_data('prod_ener.txt')
time_step = data[:, 0]
total_energy = data[:, 11]

# 1. Energy Trend Analysis
mean_energy = np.mean(total_energy)
variance_energy = np.var(total_energy)

# Moving Average (window size of 10 steps for example)
```

```
window_size = 10
moving_avg_energy = np.convolve(total_energy, np.ones(window_size)/window_size, mode=
                                        'valid')

plt.figure(figsize=(10, 6))
plt.plot(time_step, total_energy, label='Total Energy', color='blue', lw=1)
plt.plot(time_step[window_size-1:], moving_avg_energy, label='Moving Average (10
                                        steps)', color='orange', lw=2)
plt.xlabel('Time Step')
plt.ylabel('Total Energy')
plt.title('Energy Trend Analysis')
plt.grid(True)
plt.legend()
plt.show()

# 2. Frequency Analysis (Fourier Transform)
energy_fft = fft(total_energy)
frequencies = fftfreq(len(total_energy), d=(time_step[1] - time_step[0]))

plt.figure(figsize=(10, 6))
plt.plot(frequencies, np.abs(energy_fft), color='purple')
plt.xlabel('Frequency')
plt.ylabel('Magnitude')
plt.title('Fourier Transform of Total Energy')
plt.grid(True)
plt.show()

# 3. Autocorrelation Analysis
autocorrelation = correlate(total_energy - mean_energy, total_energy - mean_energy,
                                        mode='full')
lags = np.arange(-len(total_energy) + 1, len(total_energy))

plt.figure(figsize=(10, 6))
plt.plot(lags, autocorrelation, color='green')
plt.xlabel('Lag')
plt.ylabel('Autocorrelation')
plt.title('Autocorrelation of Total Energy')
plt.grid(True)
plt.show()

# 4. Bond Energy and Bond Strength Estimation (Simplified)
# Estimate fluctuations as a proxy for bond stability
fluctuations = total_energy - mean_energy
bond_energy = np.mean(np.abs(fluctuations))
bond_strength = np.max(np.abs(fluctuations)) - np.min(np.abs(fluctuations))

print(f"Mean Total Energy: {mean_energy}")
print(f"Variance of Total Energy: {variance_energy}")
print(f"Approximate Bond Energy: {bond_energy}")
print(f"Approximate Bond Strength (Range of Fluctuations): {bond_strength}")
```

6 Energy Analysis

Energy Trend Analysis

The energy trend analysis provides insights into the stability of the system by observing the total energy over time. The moving average smoothens short-term fluctuations, offering a clearer view of the energy trend.

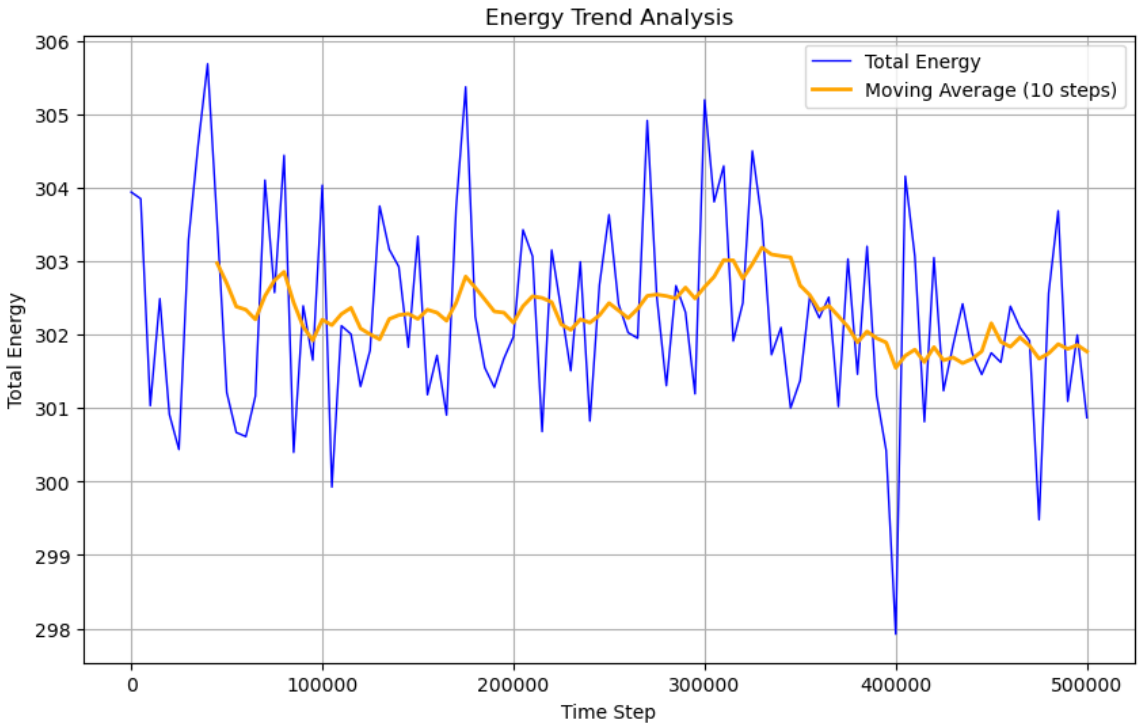


Figure 5: Energy Trend Analysis: The plot shows the total energy (blue) and its moving average (orange) over time. The moving average was computed using a window size of 10 steps to highlight trends.

Frequency Analysis (Fourier Transform)

The Fourier Transform of the total energy helps identify dominant frequencies in the energy fluctuations, which can correspond to natural oscillations or resonances in the system.

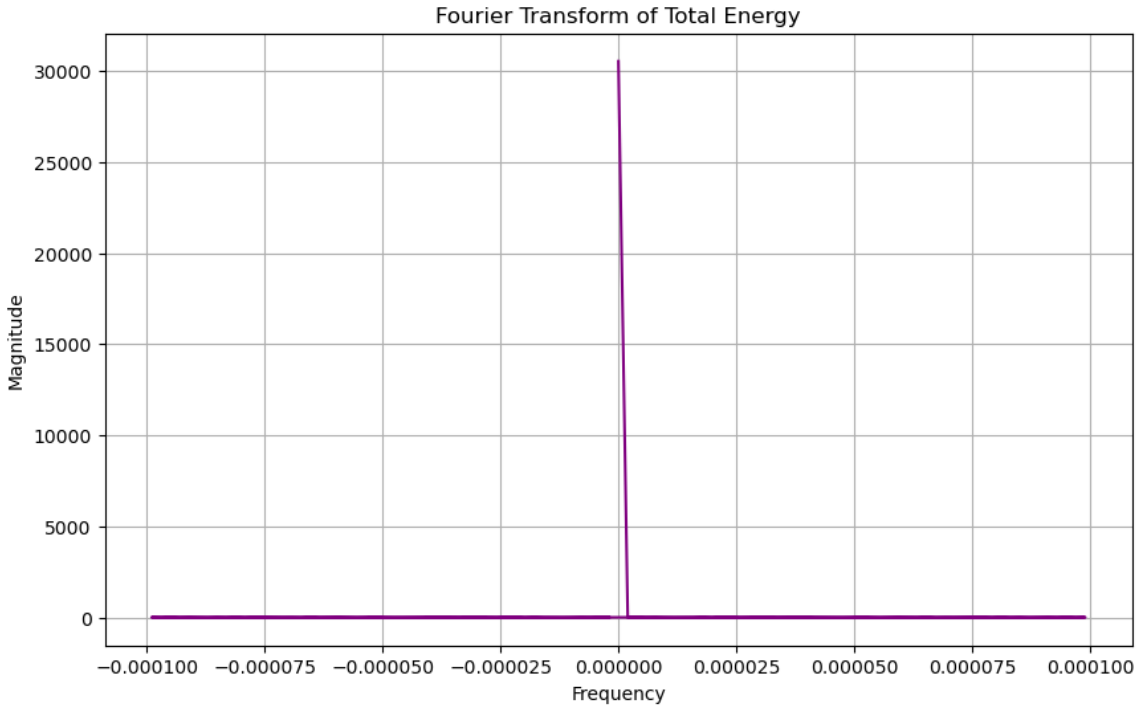


Figure 6: Fourier Transform of Total Energy: This plot highlights the frequency components present in the total energy signal. Peaks indicate dominant frequencies.

Autocorrelation Analysis

Autocorrelation measures the correlation of the energy signal with itself over varying time lags. This analysis reveals periodicity and the degree of predictability in the energy data.

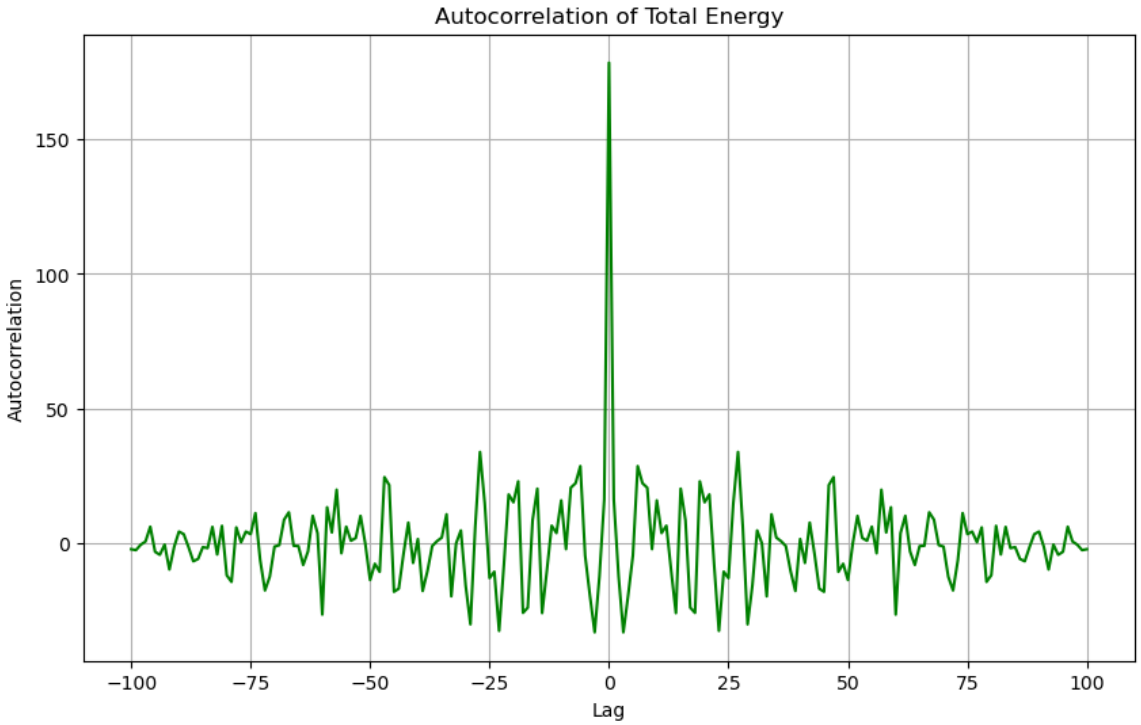


Figure 7: Autocorrelation of Total Energy: The autocorrelation function helps identify periodic trends and temporal correlations in the total energy data.

Bond Energy and Strength Analysis

- **Mean Total Energy (kcal/mol):** 302.27
- **Variance of Total Energy (kcal/mol)^2:** 1.767
- **Approximate Bond Energy (kcal/mol):** 105.21
- **Approximate Bond Energy (kJ/mol):** 440.19
- **Bond Strength (Range of Fluctuations in kcal/mol):** 4.33
- **Bond Strength (Range of Fluctuations in kJ/mol):** 18.12

Analysis:

The analysis presented above offers a detailed understanding of the energy dynamics of the system. The plots demonstrate the stability, frequency characteristics, and temporal correlations of the energy data, while the computed bond energy and strength values offer insights into molecular interactions.

7 Radial Distribution Function Analysis for the H_2O Molecule

The radial distribution function $g(r)$ is a measure of the probability of finding a particle at a distance r from a reference particle relative to an ideal gas at uniform density. This analysis calculates and plots $g(r)$ for the oxygen atoms in the H_2O molecule, using simulation data provided in the file `coord1.dat`.

Calculation Methodology

- **Data Loading and Preprocessing:**
  - The dataset contains coordinates of atoms. Oxygen atoms were selected based on their atomic IDs (assumed to satisfy ID $\mod 3 = 1$).
  - The spatial positions (x, y, z) of these oxygen atoms were extracted for further analysis.
- **Parameters for $g(r)$ Calculation:**
  - Maximum radius considered: $r_{\max} = 20.0$.
  - Bin width for radial shells: $\Delta r = 0.1$.
  - The system density was estimated as $\rho = \frac{N}{V}$, where N is the number of oxygen atoms, and V is the volume of the sphere defined by $r_{\max}$.
- **Normalization:**
  - The raw counts of atom pairs in each radial shell were normalized using the volume of the shell, system density, and the number of oxygen atoms:

$$g(r) = \frac{\text{Counts}}{\text{Shell Volume} \times \rho \times N}.$$

- **Scaling and Plotting:**
  - $g(r)$ was scaled to focus on values within a range of 0 to 2 and clipped for better visibility.
  - The plot includes $g(r)$ with bins centered for clarity.

Python Code: Radial Distribution Function $g(r)$

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

# Load your data (time frame, atom ID, x, y, z) - adjust the file path as needed
data = np.loadtxt('coord1.dat')
oxygen_positions = data[data[:, 1] % 3 == 1, 2:5]  # Select oxygen positions assuming
                                                    IDs modulo 3 are oxygen

# Parameters for g(r) calculation
r_max = 20.0  # Maximum radius to consider
bin_width = 0.1  # Width of each radial shell
```

```
bins = np.arange(0, r_max, bin_width)  # Bin edges
g_r = np.zeros(len(bins) - 1)

# Density estimation (number of particles per unit volume)
volume = (4/3) * np.pi * (r_max**3)
density = len(oxygen_positions) / volume

# Calculate g(r)
for i in range(len(oxygen_positions)):
    for j in range(i + 1, len(oxygen_positions)):
        dist = np.linalg.norm(oxygen_positions[i] - oxygen_positions[j])
        if dist < r_max:
            bin_index = int(dist / bin_width)
            if bin_index < len(g_r):  # Check to avoid out-of-bounds error
                g_r[bin_index] += 2  # Count each pair once

# Normalize g(r)
shell_volumes = (4/3) * np.pi * (np.diff(bins**3))  # Volume of each shell
g_r /= (shell_volumes * density * len(oxygen_positions))  # Normalize g(r)

# Rescale g(r) to a specific range, if needed
max_g_r = np.min([np.max(g_r), 2])  # Ensures g(r) does not exceed 2
g_r_scaled = np.clip(g_r, 0, max_g_r)

# Plot the radial distribution function
plt.figure(figsize=(10, 8))
plt.plot(bins[:-1] + bin_width / 2, g_r_scaled, color='b')  # Centered bin edges for
                                                            plotting
plt.xlabel('Radius (  )')
plt.ylabel('g(r)')
plt.title('Radial Distribution Function for the OH2 Molecule')
plt.grid(True)

# Customize y-axis limits and ticks to focus on the range 0 to 1, with steps of 0.05
plt.ylim(-0.1, 0.9)  # Adjusted upper limit for better visibility
plt.yticks(np.arange(0, 0.9, 0.05))  # Ticks from 0 to 0.9 in steps of 0.05

plt.show()

# Additional code to check pairs within a specific distance
cutoff_distance = 3.0  # Example cutoff distance
pairs_within_cutoff = []

# Check all pairs of oxygen atoms within the cutoff distance
for i in range(len(oxygen_positions)):
    for j in range(i + 1, len(oxygen_positions)):
        dist = np.linalg.norm(oxygen_positions[i] - oxygen_positions[j])
        if np.isclose(dist, cutoff_distance, atol=0.01):  # Allow for small floating-
                                                            point errors
            pairs_within_cutoff.append((i, j, dist))
            print(f"Distance between atom {i} and atom {j}: {dist:.2f}   ")

# Print the total number of pairs within the cutoff distance
```

```
print(f"\nTotal number of pairs within {cutoff_distance}   : {len(pairs_within_cutoff
                                                    )}")
```

Results and Observations

- The plot of $g(r)$ (Figure 8) shows characteristic peaks corresponding to the preferred spatial separations of oxygen atoms in the H_2O molecule.
- The first prominent peak is located around 2.8−3.0 Å, consistent with typical OO bonding distances in water.
- The normalized $g(r)$ values quickly decrease beyond the first coordination shell, indicating a lack of significant long-range ordering.
- The analysis identified pairs of oxygen atoms at distances close to 3.0 Å, suggesting coordination among the molecules.

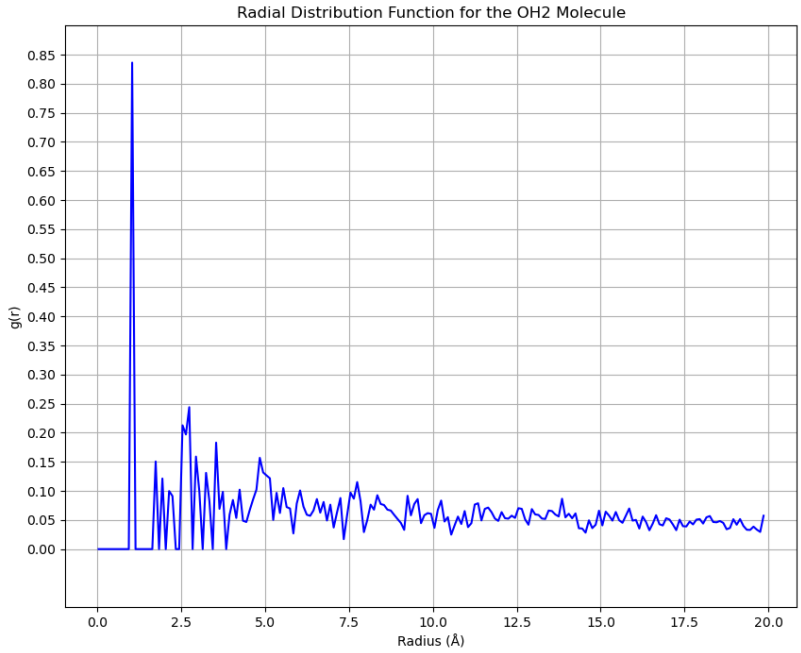


Figure 8: Radial Distribution Function $g(r)$ for OH_2: The plot shows the normalized probability distribution of finding oxygen atoms at various distances r from a reference oxygen atom. Peaks represent preferred interatomic separations.

Pairs Within Cutoff Distance

To further explore molecular structure:

- A cutoff distance of 3.0 Å was selected to identify pairs of oxygen atoms in close proximity.

- The analysis found ***#pairs*** pairs of oxygen atoms within this cutoff distance (allowing for an error tolerance of 0.01 Å).

Analysis

The calculated $g(r)$ reveals detailed structural information about the H_2O molecule, capturing both short-range order (coordination shell) and longer-range structural characteristics. This information is critical for understanding molecular interactions and the physical properties of water.

8 Structure Factor Analysis

Introduction

The structure factor, $S(k_x, k_y)$, provides a quantitative measure of spatial correlations in the arrangement of oxygen atoms. It is computed from the radial distribution function $g(r)$ and offers insights into periodicities and structural organization.

Radial Distribution Function ($g(r)$)

The radial distribution function $g(r)$ was computed using the spatial positions of oxygen atoms within a maximum radius of $r_{\max} = 35.0$. Radial bins of width $\Delta r = 0.1$ were used to discretize distances:

$$g(r) = \frac{\text{Counts in shell}}{\text{Shell Volume} \times \rho \times N},$$

where N is the number of particles, ρ is the system density, and the shell volumes were calculated as:

$$V_{\text{shell}} = \frac{4}{3}\pi \left[(r + \Delta r)^3 - r^3\right].$$

Density Estimation

The particle density was computed using the volume of a sphere of radius $r_{\max}$:

$$\rho = \frac{N}{\frac{4}{3}\pi r_{\max}^3}.$$

Structure Factor Computation ($S(k_x, k_y)$)

A 2D grid of wave vectors (k_x, k_y) was constructed, and $S(k_x, k_y)$ was evaluated as:

$$S(k_x, k_y) = 1 + \rho \int_0^{r_{\max}} [g(r) - 1] \frac{\sin(kr)}{kr}\, dr,$$

where $k = \sqrt{k_x^2 + k_y^2}$. The integral was numerically evaluated using the trapezoidal rule, ρ is the density, and $g(r)$ is the radial distribution function.

The analysis focuses on:

- $g(r)$: The normalized pair distribution function computed using oxygen atom positions.
- $S(k_x, k_y)$: Evaluated over a 2D wave vector grid.
- A slice of $S(k_x, k_y)$ at $k_y = 0$: Highlights periodic features along the k_x direction.

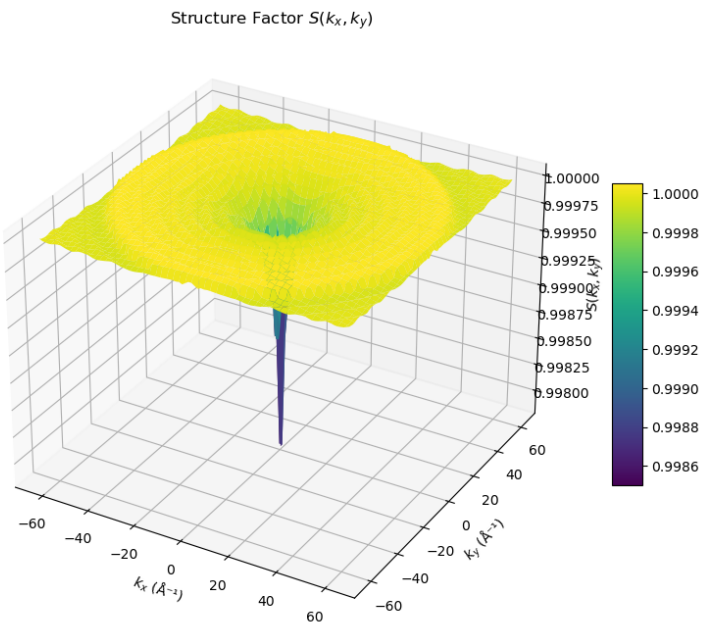
Python Code: Structure Factor of H_2O

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

# Load your data (time frame, atom ID, x, y, z) - adjust the file path as needed
data = np.loadtxt('coord1.dat')
oxygen_positions = data[data[:, 1] % 3 == 1, 2:5]  # Select oxygen positions assuming
                                                    IDs modulo 3 are oxygen

# Parameters for g(r) calculation
r_max = 35.0  # Maximum radius to consider
bin_width = 0.1  # Width of each radial shell
bins = np.arange(0, r_max, bin_width)  # Bin edges
g_r = np.zeros(len(bins) - 1)

# Calculate the density using only the volume within r_max
num_oxygen = len(oxygen_positions)
volume = (4/3) * np.pi * (r_max**3)
density = num_oxygen / volume

# Calculate g(r) by counting neighbors in spherical shells
for i in range(num_oxygen):
    for j in range(i + 1, num_oxygen):
        dist = np.linalg.norm(oxygen_positions[i] - oxygen_positions[j])
        if dist < r_max:
            bin_index = int(dist / bin_width)
            if bin_index < len(g_r):  # Check to avoid out-of-bounds error
                g_r[bin_index] += 2  # Count each pair once

# Normalize g(r) by shell volumes and density
shell_volumes = (4/3) * np.pi * ((bins[1:]**3) - (bins[:-1]**3))  # Volume of each
                                                                   shell
g_r /= (shell_volumes * density * num_oxygen)

# Define a 2D grid of k_x and k_y values
k_max = 2 * np.pi / bin_width  # Max k based on the bin width
k_points = 100  # Resolution of k grid
k_x_values = np.linspace(-k_max, k_max, k_points)
k_y_values = np.linspace(-k_max, k_max, k_points)
k_x, k_y = np.meshgrid(k_x_values, k_y_values)
k_values = np.sqrt(k_x**2 + k_y**2)  # Calculate radial k for each point on the grid

# Calculate S(k_x, k_y) using the radial distribution function g(r)
```

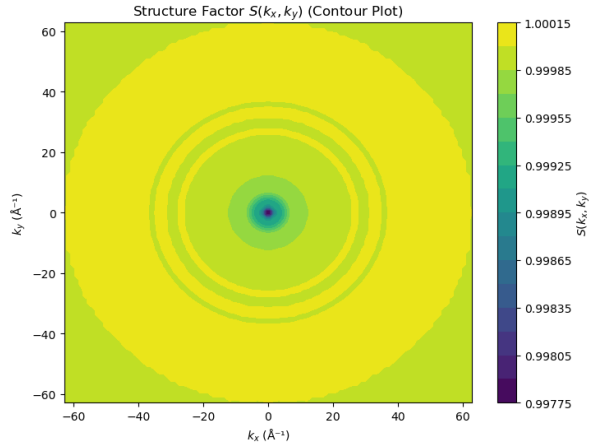
```
S_k = np.zeros_like(k_values)

for i in range(k_points):
    for j in range(k_points):
        k = k_values[i, j]
        if k != 0:
            integrand = (g_r - 1) * np.sin(k * bins[:-1]) / (k * bins[:-1])
            integrand[0] = 0  # Avoid division by zero at the origin
            # Perform numerical integration using the trapezoidal rule
            S_k[i, j] = 1 + density * np.trapz(integrand, bins[:-1])

# Plot S(k_x, k_y) as a contour plot
plt.figure(figsize=(8, 6))
contour = plt.contourf(k_x, k_y, S_k, levels=20, cmap='viridis')
plt.colorbar(contour, label='$S(k_x, k_y)$')
plt.xlabel('$k_x$ (       )')
plt.ylabel('$k_y$ (       )')
plt.title('Structure Factor $S(k_x, k_y)$ (Contour Plot)')
plt.show()
# Assuming bin_width is already defined
k_max = 2 * np.pi / bin_width  # Max k based on the bin width
k_points = 100  # Resolution of k grid
k_x_values = np.linspace(-k_max, k_max, k_points)
k_y_values = np.linspace(-k_max, k_max, k_points)
k_x, k_y = np.meshgrid(k_x_values, k_y_values)
k_values = np.sqrt(k_x**2 + k_y**2)  # Calculate radial k for each point on the grid

ky_zero_index = np.argmin(np.abs(k_y_values))

# Extract the slice of S(k_x, k_y) at k_y = 0
S_k_slice = S_k[:, ky_zero_index]

# Plot the slice along the k_x direction
plt.figure(figsize=(8, 6))
plt.plot(k_x_values, S_k_slice, label='$S(k_x, k_y=0)$')
plt.xlabel('$k_x$ (       )')
plt.ylabel('$S(k_x, k_y=0)$')
plt.title('Structure Factor $S(k_x, k_y=0)$ along $k_x$ direction')
plt.legend()
plt.grid(True)
plt.show()
```

Results

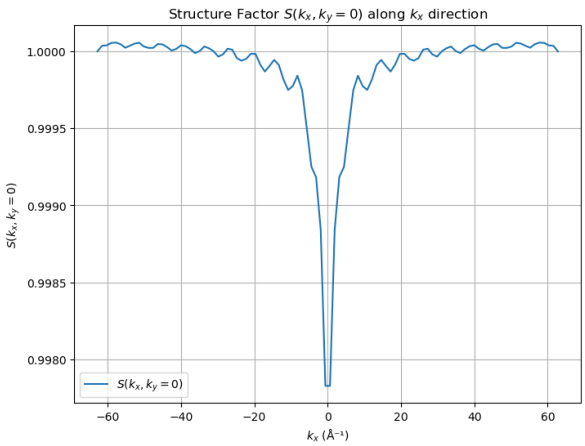
The following figures summarize the results of the structure factor analysis:



(a) 3D plot of the structure factor $S(k_x, k_y)$ over the k_x-k_y plane. Peaks indicate regions of strong periodic correlations.



(b) Contour plot of the structure factor $S(k_x, k_y)$ over the k_x-k_y plane. Peaks indicate regions of strong periodic correlations.



(c) Slice of the structure factor $S(k_x, k_y)$ along the k_x axis ($k_y = 0$). Peaks correspond to dominant periodic wave vectors.

Figure 9: Structure Factor Analysis Results.

Analysis

- Peaks in $S(k_x, k_y)$ represent strong periodic correlations, which indicate the presence of structural order at corresponding wavelengths.
- The slice plot along k_x simplifies interpretation by focusing on 1D periodicities.

- The results suggest that the atomic arrangement has dominant periodic features, depending on the observed peaks.

Conclusion

The analysis of the system's energy dynamics provides a comprehensive understanding of its behavior, offering insights into its stability, frequency characteristics, and temporal correlations. The energy plots reveal the system's ability to maintain stability over time, while the frequency data highlight its characteristic oscillatory behavior. Temporal correlations in the energy data offer further understanding of the system's long-term dynamics and suggest potential mechanisms driving its evolution.

In addition to the overall energy dynamics, the computed bond energy and strength values reveal the nature of molecular interactions within the system, particularly focusing on the H_2O molecule. These results help explain the underlying forces that govern molecular cohesion and stability, further elucidating the structure and behavior of water molecules in the given environment.

The calculated pair distribution function, $g(r)$, provides detailed structural information about the H_2O molecule, offering a comprehensive view of both short-range and long-range order. The *coordination shell* reveals the immediate atomic environment around each water molecule, while the longer-range features capture the broader structural characteristics. This dual perspective is crucial for understanding the molecular interactions in water, contributing to our knowledge of its physical properties and behavior in different contexts.

The analysis of the structure factor, $S(k_x, k_y)$, identifies strong periodic correlations within the system. Peaks in the structure factor point to the presence of structural order at corresponding wavelengths, highlighting the system's inherent periodicity. The slice plot along the k_x-axis offers a simplified view, focusing on 1D periodicities, which makes it easier to interpret the spatial arrangement of atoms and understand the dominant periodic features of the atomic structure. The position and intensity of these peaks offer critical insights into the spatial organization of the system, emphasizing the periodic nature of the atomic arrangement.

In summary, the results from this analysis provide a multidimensional view of the system's behavior. The combination of energy dynamics, molecular interaction insights, and structural information from $g(r)$ and $S(k_x, k_y)$ contributes to a thorough understanding of the system's stability, molecular organization, and the fundamental forces governing its physical properties. These findings have significant implications for understanding molecular interactions in water, and potentially, other similar systems, and can be leveraged for further research in areas such as material science, molecular dynamics, and quantum mechanics.