Assignment 2

Due November 15, 2024, on learn dropbox

Show your work for each question

This a simulation assignment. The required Python scripts and pdb data files are available on the UW Learn page, in the same content section this Assignment PDF file was found.

Software requirements:

The openMM package (can be installed using anaconda or miniconda)

The openMM-Tools package (can be installed using anaconda or miniconda)

Plotting: you can use Matplotlib or MS Excel

**Question 1:** Use the python script water2.py to study the dynamics of two rigid water molecules under NVE (microcanonical) conditions. The parameters are at the beginning of the script. You will need to set the ensemble flag to NVE as shown below:



The water2.py script requires the water2.pdb file which contains the coordinates of the two water molecules.

1. Report all the parameters of your simulations with units.

steps = 1000

skipSteps = 1

T = 150 K

dt = 1 fs

ensemble = ‘NVE’

1. Plot the kinetic, potential, and total energies (stored in files KE.NVE, PE.NVE, and TE.NVE). Note your observations.

A graph of energy and energy

Description automatically generated

A graph showing energy and energy

Description automatically generated

A graph showing a number of energy

Description automatically generated

\*NOTE: x and y labels are swapped, couldn’t run again because data would reset

The kinetic energies are positive and the potential energies are negative. Since the magnitude of the potential energy is higher than the magnitude of kinetic energy, it results in the total energy being very close to -17. The magnitude of potential energy is higher because the force of attraction between the two molecules is higher than their kinetic forces due to hydrogen bonding.

1. Compute the average and variance of the kinetic, potential, and total energies. Vary the timestep (dt) and the number of time steps and note your observations. Which quantity do you expect will be conserved?

Kin mean: 7.5758605

Kin variance: 3.9290402

Pot mean: -22.134796

Pot variance: 3.932365

Total mean: -14.558937

Total variance: 1.6873536e-06

By changing the time step, the potential energy increases and kinetic energy decreases. The quantity that we expect to be conserved is the total energy.

1. Plot the distance between the two oxygen atoms of each water as a function of time (filerOO.NVE). Do you observe oscillations? Report the mean OO distance along with its variance.

A graph of a graph

Description automatically generated

Yes, there are oscillations.

Mean = 0.27264581127

Variance = 2.1292976986368807e-07

**Question 2:** Use the python script water2.py to study the dynamics of two water molecules under NVT (canonical) conditions (change ensemble flag to NVT). Data files will now have the suffix NVT, ie: rOO.NVT, KE.NVT, PE.NVT, and TE.NVT.

1. Report all the parameters of your simulations

steps = 1000

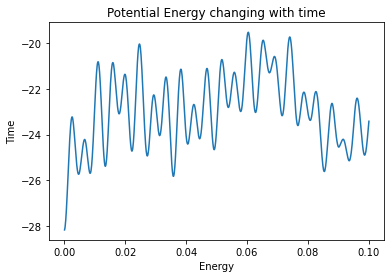
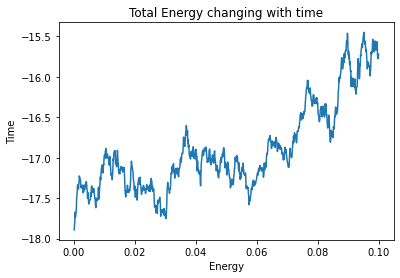
skipSteps = 1

T = 150 K

dt = 1 fs

ensemble = ‘NVT’

1. Repeat the steps of the NVE analysis above, but this time with in the NVT ensemble.A graph showing energy and energy

   Description automatically generated  

A graph of a graph

Description automatically generated

Kin mean: 0.84940034

Kin variance: 0.08371429

Pot mean: -27.669907

Pot variance: 0.053568363

Total mean: -26.820507

Total variance: 0.07164044

Distance mean: 0.27413516102

Distance variance: 2.657043098686558e-06

**Question 3:** Perform an NVT MD simulation of liquid argon using arBox*.*py. Report all parameters of the simulation (ie, the total number of atoms, the temperature, the density, the time step, the number of time steps, etc.). Use the suggested *T* and density values. Increase the number of time steps to improve the quality of your results.

temperature = 300.

steps = 1000

skipSteps = 10

equilSteps = 100

N=200

1. Use the Vanalysis*.*py script to calculate the average potential energy. The script takes the output trajectory file name as a command line argument. The filename is arliquidtraj200*.*h5 for *N*=200. Vary the number of particles in the simulation box (e.g. *N*= 200, 300, 400, 500, 600) and note the change in average potential energy per particle *V/N*. What happens as the system size increases? (Note that the script outputs the total potential energy in the file V). Estimate the standard error bars of the average potential energy per particle (hint, you need the variance of your data along with the total number of steps).

Standard error bars = √(Variance)/√(step size)

|  |  |  |  |
| --- | --- | --- | --- |
| Number of particles | Average potential energy | Variance potential energy | Standard Error bars |
| 200 | -916.81635 | 1282.2538 | 1.1323665 |
| 300 | -1374.3698 | 1942.9742 | 1.3939061 |
| 400 | -1839.971 | 2093.8494 | 1.447014 |
| 500 | -2304.5215 | 3170.7925 | 1.7806719 |
| 600 | -2772.708 | 4136.1245 | 2.0337464 |

As the system size increases, the average potential energy increases in magnitude. However, the variance also increases, which shows that the potential energy per particle varies more.

1. Use genPairDistancesArgon*.*py to calculate the *g*(*r*) for the simulations above. The script takes the output trajectory file name as a command line argument. What happens to the *g*(*r*) as the system size increases?

A graph with a line

Description automatically generated

A graph of a graph

Description automatically generated

A graph with a line

Description automatically generated

A graph with a line

Description automatically generated

A graph with a line

Description automatically generated

As the number of particles increases, the probabilities of finding two argon atoms at a higher distance increase. These probabilities are higher than 1, which means that there was an error in the code. However, ignoring this error, we can see that with each increasing number of particles, there are more peaks on each graph.

1. Calculate the number of neighbours in the first shell using the *g*(*r*). Show all your work.

Delta

NN = Σ g(r)\*r

number of neighbors = (delta r \*4\*π\*(N/volume) \*NN

|  |  |  |
| --- | --- | --- |
| Number of particles | Delta r | Number of neighbors |
| 200 | 1.0204046926498413 | 12.27870224508467 |
| 300 | 1.1679270145893097 | 12.238180000000002 |
| 400 | 1.2853693494796752 | 12.06939 |
| 500 | 1.3845449826717378 | 11.841707999999999 |
| 600 | 1.4712359306812286 | 11.823420000000002 |

**Question 4:** Perform an NVT MD simulation of liquid water using the waterBox*.*py. Report all the parameters of the simulation (ie, the total number of atoms, the temperature, the density, the time step, the number of time steps, etc.). Use the suggested *T* and density values. Increase the number of time steps to improve the quality of your results. Also increase the size of the simulations box.

temperature = 300 K

steps = 1000

skipSteps = 10

equilSteps = 100

Box\_edge=1 nm

1. Use the genPairDistancesWater*.*py script to calculate the *gOO*(*r*) and *gOH*(*r*) for the simulations above. The script takes the output trajectory file name as a command line argument. Describe your results. What happens to the *g*(*r*) as the system size increases?

A graph of a graph

Description automatically generated

A graph of a graph

Description automatically generated

These graphs show that between oxygen atoms, there is only one distance where the probability is the highest, which is around 0.28. But for the distance between oxygen and hydrogen atoms, there are two distances where the probability is high, which is at around 0.17 and 0.32. This is because there are two hydrogen atoms on each oxygen, so for each distance between two oxygen atoms, there are two distances between an oxygen atom and the hydrogen atom on the neighboring atom.

\*was unable to increase size of the system since code stopped working after these two graphs were given

1. Calculate the number of neighbours in the first shell using the *gOO*(*r*). Show all your work.

NN = Σ g(r)\*r

number of neighbors = (delta r \*4\*π\*(N/volume) \*NN

|  |  |  |
| --- | --- | --- |
| Type of distance | Delta r | Number of neighbors |
| OO | 0.496 | 15.935187500000003 |