Report on **Project: Computer simulations of molecular liquids**

Zhicheng Zhang

June 30, 2023

Abstract

In this project, I applied molecular simulation methods to studying the properties of a 2D system of particles. The methods include molecular dynamics (MD) and Metropolis Monte Carlo (MC) simulation. I modeled the system using different model interaction potentials among the particles, including Lennard-Jones (LJ) potential, model atomic potential and model colloidal potential, all pairwise and with a cut-off distance r_c. With carefully chosen initial conditions and boundary conditions, I was able to realize equilibrium states, calculate average total energies and deviations, calculate the radial distributions functions, evaluate the pressure, etc. **Special notes:**

1 Introduction

This is the final project of the course Introduction to Computational Physics. The project is aimed at testing our ability to implement the algorithms we learned in class through programming. The main programming language I used in this project is Python3.11. I have uploaded all the notes, codes, slides and figures we produced in class to my Github repository Computational Physics.

The ideas of molecular dynamics and Monte Carlo simulation have been discussed extensively in our textbook. In the following "Background" section, I will present some fundamental concepts and equations regarding the simulation methods. In further sections, I will deal with the specific requirements of the simulations and explain the results.

2 Background

Computer simulations allow us to calculate the physical quantities and study the properties of many-particle systems, without conducting experiments in real world. Hence we can generate much more data in a much shorter time. However, it is impossible to completely replicate a real system in a simulation, and the quantities measured in a simulation do not necessarily correpond to the properties studied in a real experiment. For example, we can specify the position and velocity of each molecule in a molecular simulation program, but none of these details can be measured in a real experiment.

In fact, quantities measurable for experiments are usually average quantities over large ensembles of particles, and often also over the long time of the experiments. To extract from a computer simulation data comparable to those measured in a corresponding experiment, we have to know what kind of averages we should compute out of the simulation, and the techniques to numerically compute them. This involves statistical mechanics. Below are some useful results.

Let $\Omega(E, V, N)$ denote the volume of the Γ phase space of the system of N particles with energy E confined in a box of volume V. Or in words of quantum mechanics, $\Omega(E, V, N)$ denotes the number of eigenstates with energy E of a system of N particles in a volume V. The system has entropy

$$S(N, V, E) \equiv k_B \ln \Omega(N, V, E) \tag{2.1}$$

yielding the thermodynamic definition of temperature

$$\frac{1}{T} = \left(\frac{\partial S}{\partial E}\right)_{V,N} \tag{2.2}$$

Conventionally, we use the shorthand notation

$$\beta(E, V, N) \equiv \left(\frac{\partial \ln \Omega}{\partial E}\right)_{V,N} \tag{2.3}$$

which is found to be $\beta = \frac{1}{k_B T}$. Such a system in contact with a thermal bath held at equilibrium temperature T is called a canonical ensemble, and it follows Boltzmann distribution, in which the probability that the system is found in state i is

$$P_j = \frac{e^{-\frac{E_i}{k_B T}}}{\sum\limits_i e^{-\frac{E_j}{k_B T}}} \tag{2.4}$$

We can then calculate the average energy of the system at the given temperature T

$$\begin{aligned} \langle E \rangle &= \sum_i E_i P_i \\ &= \frac{\sum\limits_i E_i e^{-\frac{E_i}{k_B T}}}{\sum\limits_j e^{\frac{-E_j}{k_B T}}} \\ &= -\frac{\partial \ln Z}{\partial \beta} \end{aligned} \tag{2.5}$$

In the last line, we define the partition function

$$Z = \sum_i e^{-\frac{E_i}{k_B T}} \tag{2.6}$$

By comparing equation ?? with the thermodynamic relation between energy E and Helmholtz free energy F

$$E = \frac{\partial(F/T)}{\partial(1/T)} \tag{2.7}$$

we see F can be calculated from the partition function

$$F = -k_B T \ln Z = -k_B T \ln\left(\sum_i e^{-\frac{E_i}{k_B T}}\right) \tag{2.8}$$

For an observable quantity (Hermitian operator) $\mathcal{A}$, its thermal average is computed via its expectation values in the eigenstates

$$\langle \mathcal{A} \rangle = \frac{\sum\limits_i e^{-\frac{E_i}{k_B T}} \langle i|\mathcal{A}|i\rangle}{\sum\limits_j e^{-\frac{E_j}{k_B T}}} \tag{2.9}$$

Let $\mathcal{H}$ denote the Hamiltonian of the system, and noting that $\exp(-E_i/k_B T) = \langle i|\exp(-\mathcal{H}/k_B T)|i\rangle$, we then write

$$\begin{aligned}\langle \mathcal{A} \rangle &= \frac{\sum\limits_i \langle i|\exp(-\mathcal{H}/k_B T)\mathcal{A}|i\rangle}{\sum\limits_j \langle j|\exp(-\mathcal{H}/k_B T)|j\rangle} \\ &= \frac{\mathrm{tr}\left(\exp(-\mathcal{H}/k_B T)\mathcal{A}\right)}{\mathrm{tr}\left(\exp(-\mathcal{H}/k_B T)\right)}\end{aligned} \tag{2.10}$$

where tr means trace. Hamiltonian $\mathcal{H}$ is the sum of kinetic energy and potential energy, $\mathcal{H} = \mathcal{K} + \mathcal{U}$, so the exponential of the Hamiltonian becomes $\exp(-\beta(\mathcal{K}+\mathcal{U}))$. According to Baker-Campbell-Hausdorff formula,

$$e^X e^Y = e^{X+Y+\frac{[X,Y]}{2}+\frac{[X,[X,Y]]}{12}-\frac{[Y,[X,Y]]}{12}+\cdots} \tag{2.11}$$

the exponential is approximated by

$$e^{-\beta\mathcal{K}}e^{-\beta\mathcal{U}} = e^{-\beta(\mathcal{K}+\mathcal{U})+\beta^2\frac{[\mathcal{K},\mathcal{U}]}{2}+\cdots} = e^{-\beta(\mathcal{K}+\mathcal{U})+O([\mathcal{K},\mathcal{U}])} \tag{2.12}$$

To estimate the order of $[\mathcal{K},\mathcal{U}]$, we take the simplest example of a 1-dimensional harmonic oscillator. The Hamiltonian is $\mathcal{H} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \frac{1}{2}kx^2$, and the commutator is

$$[\mathcal{K},\mathcal{U}] = \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}, \frac{1}{2}kx^2\right] = -\frac{\hbar^2 k}{2m} - \frac{\hbar^2 k}{m}x\frac{\partial}{\partial x} \tag{2.13}$$

Put the oscillator in the n'th eigenstate, and by writing the Hamiltonian in terms of the raising and lowering operators, we have

$$\begin{cases}\mathcal{K} = -\frac{\hbar\omega}{4}(a_+^2 - a_-a_+ - a_+a_- + a_-^2) \\ \mathcal{U} = \frac{\hbar k}{4m\omega}(a_+^2 + a_-a_+ + a_+a_- + a_-^2)\end{cases} \tag{2.14}$$

where $\omega \equiv \sqrt{\frac{k}{m}}$ and then

$$\begin{cases} \mathcal{K}\psi_n = -\dfrac{\hbar\omega}{4}\left(\sqrt{(n+1)(n+2)}\psi_{n+2} - (2n+1)\psi_n + \sqrt{n(n-1)}\psi_{n-2}\right) \\ \mathcal{U}\psi_n = \dfrac{\hbar k}{4m\omega}\left(\sqrt{(n+1)(n+2)}\psi_{n+2} + (2n+1)\psi_n + \sqrt{n(n-1)}\psi_{n-2}\right) \end{cases} \tag{2.15}$$

Some substitutions yield

$$\begin{cases} \mathcal{K}\mathcal{U}\psi_n = -\dfrac{\hbar^2 k}{16m}\left(\gamma_{n+1,4}\psi_{n+4} - 4\gamma_{n+1,2}\psi_{n+2} + (-2n^2-2n+1)\psi_n + 4\gamma_{n-1,2}\psi_{n-2} + \gamma_{n-3,4}\psi_{n-4}\right) \\ \mathcal{U}\mathcal{K}\psi_n = -\dfrac{\hbar^2 k}{16m}\left(\gamma_{n+1,4}\psi_{n+4} + 4\gamma_{n+1,2}\psi_{n+2} + (-2n^2-2n+1)\psi_n - 4\gamma_{n-1,2}\psi_{n-2} + \gamma_{n-3,4}\psi_{n-4}\right) \end{cases}$$

where $\gamma_{n,m} = \sqrt{(n)\cdots(n+m-1)}$. Therefore, through straightforward calculations, we conclude that

$$[\mathcal{K},\mathcal{U}]\psi_n = -\frac{\hbar^2 k}{2m}\left(\gamma_{n-1,2}\psi_{n-2} - \gamma_{n+1,2}\psi_{n+2}\right) \tag{2.16}$$

In the classical limit, $n \gg 1$, and all the none-zero matrix elements of $[\mathcal{K},\mathcal{U}]$ approach $\pm\frac{\hbar^2 kn}{2m} = \pm\frac{n\hbar^2\omega^2}{2m}$. Taking the Boltzmann distribution into account, $(n+1/2)\hbar\omega \sim k_B T$, and since $n \gg 1$, $\beta^2[\mathcal{K},\mathcal{U}] \ll 1$. Now we see in approximation **??**, the term $O[\mathcal{K},\mathcal{U}]$ can be safely neglected. In that case, we may write

$$\operatorname{tr} e^{-\beta\mathcal{H}} \approx \operatorname{tr}\left(e^{-\beta\mathcal{K}}e^{-\beta\mathcal{U}}\right) \tag{2.17}$$

We choose eigenvetors of the position operator, $|r\rangle$, and eigenvetors of the momentum operator, $|k\rangle$, to represent our equations. Consequently,

$$\begin{aligned} \operatorname{tr} e^{-\beta\mathcal{H}} &= \sum_r \langle r|e^{-\beta\mathcal{K}}e^{-\beta\mathcal{U}}|r\rangle \\ &= \sum_r \sum_k \langle k|e^{-\beta\mathcal{K}}\langle r|k\rangle\, e^{-\beta\mathcal{U}}|r\rangle \end{aligned}$$

Since $\mathcal{K}$ and $\mathcal{U}$ are respectively diagonal under the momentum basis and position basis, we further expand the equation

$$\operatorname{tr} e^{-\beta\mathcal{H}} = \sum_{r,k,r',k'} \langle k|e^{-\beta\mathcal{K}}|k'\rangle\,\langle k'|r'\rangle\,\langle r|k\rangle\,\langle r'|e^{-\beta\mathcal{U}}|r\rangle \tag{2.18}$$

Now the matrix elements can be evaluated directly

$$\langle r'|e^{-\beta\mathcal{U}}|r\rangle = e^{-\beta\mathcal{U}(\boldsymbol{r}^N)}\delta(\boldsymbol{r}^N - \boldsymbol{r}'^N) \tag{2.19}$$

Similarly,

$$\langle k|e^{-\beta\mathcal{K}}|k'\rangle = e^{-\beta\sum_{i=1}^{N}\frac{p_i^2}{2m_i}}\delta(\boldsymbol{k}^N - \boldsymbol{k}'^N)$$

and

$$\langle r|k\rangle\,\langle k'|r'\rangle = \frac{1}{(2\pi)^N}e^{i\boldsymbol{k}^N\cdot\boldsymbol{r}^N - i\boldsymbol{k}'^N\cdot\boldsymbol{r}'^N}$$

where N is the number of particles.

In our classical situation, the sum can be replaced by an integration over the Γ phase space. Thus the final result is

$$\operatorname{tr} e^{-\beta \mathcal{H}} = \frac{1}{h^{dN} N!} \int \mathrm{d}\boldsymbol{p}^N \mathrm{d}\boldsymbol{q}^N e^{-\beta\left[\mathcal{U}(\boldsymbol{r}^N)+\sum_i \frac{p_i^2}{2m_i}\right]} = Z_{classical} \tag{2.20}$$

The above equation defines the classical form of partition function. d is the dimensionality of the system, and the factor $N!$ is in accordance with the indistinguishability of identical particles.

A similar calculation results in

$$\operatorname{tr}\left(e^{-\beta \mathcal{H}} \mathcal{A}\right) = \frac{1}{h^{dN} N!} \int \mathrm{d}\boldsymbol{p}^N \mathrm{d}\boldsymbol{q}^N e^{-\beta\left[\mathcal{U}(\boldsymbol{r}^N)+\sum_i \frac{p_i^2}{2m_i}\right]} \mathcal{A}(\boldsymbol{p}^N, \boldsymbol{q}^N) \tag{2.21}$$

and the classical expression for the thermal average of observable $\mathcal{A}$ becomes

$$\langle \mathcal{A} \rangle = \frac{\int \mathrm{d}\boldsymbol{p}^N \mathrm{d}\boldsymbol{q}^N e^{-\beta\left[\mathcal{U}(\boldsymbol{r}^N)+\sum_i \frac{p_i^2}{2m_i}\right]} \mathcal{A}(\boldsymbol{p}^N, \boldsymbol{q}^N)}{\int \mathrm{d}\boldsymbol{p}^N \mathrm{d}\boldsymbol{q}^N e^{-\beta\left[\mathcal{U}(\boldsymbol{r}^N)+\sum_i \frac{p_i^2}{2m_i}\right]}} \tag{2.22}$$

Unfortunately, the ensemble average of a given quantity is impossible to compute due to the enormous domain of integration. Yet fortunately, with the help of ergodicity, we are able to relate ensemble averages to time averages. For ergodic systems, the properly evaluated time averages along simulation trajectories are equivalent to ensemble averages. That is to say, if we take an observable S, its time average is equal to its ensemble average

$$\langle S \rangle_t = \langle S \rangle_e \tag{2.23}$$

where $\langle S \rangle_t$ denotes the time average $\langle S \rangle_t = \lim\limits_{\tau \to \infty} \int_0^\tau S\left(\boldsymbol{p}^N(t), \boldsymbol{q}^N(t)\right) \mathrm{d}t$, and $\langle S \rangle_e$ denotes the ensemble average as shown in Eq. **??**. Simulation of the experimental trajectories and averaging by time lead to molecular dynamics simulation, while sampling in the phase space and estimation of the ensemble average lead to Monte Carlo simulation

2.1 Molecular Dynamics

Molecular dynamics simulations very much resemble real experiments. We manage to mimic how the particles move in nature by calculating the forces acting on them and performing their motions governed by mechanical equations.

A successful molecular dynamics simulation consists of the following procedures: initialization, calculation of the force, and integration of the equations of motion. Their basic algorithms are sketched as follows

2.1.1 Initialization

Algorithm 1 Initialization

```
function INIT
    sumv=0                                              ▷ velocity of center of mass
    sumv2=0                                             ▷ mean squared velocity
    procedure SET INITIAL POSITIONS AND VELOCITIES
        for i=1: npart do
            x(i)=lattice_pos(i)                         ▷ place the particles on a lattice
            v(i)=randv()                                ▷ random distributed velocities
            sumv=sumv+v(i)                              ▷ sum for total velocity
            sumv2=sumv2+v(i)**2                         ▷ sum for total squared velocity
        end for
        sumv=sumv/npart                                 ▷ velocity of center of mass
        sumv2=sumv2/npart                               ▷ mean squared velocity
    end procedure
    procedure RESCALE VELOCITIES AND GET PREVIOUS POSITIONS
        fs=sqrt(3*temp/sumv2)                           ▷ scale factor for velocities
        for i=1: npart do
            v(i)=(v(i)-sumv)*fs                         ▷ set desired kinetic energy and fix centre of mass
            xm(i)=x(i)-v(i)*dt                          ▷ get position of previous timestep
        end for
    end procedure
    return
end function
```

In the above algorithm, `npart` is the number of particles. The function `lattice_pos()` gives the coordinates of lattice position `i`, and `randv()` gives a randomly distributed velocity (with expectation value 0). It does not matter what probability distribution we use. No matter uniform distribution or normal distribution is `randv()`, the velocities will eventually end up in Maxwell-Boltzmann distribution once the system reaches equilibrium.

`temp` means temperature, and as we know the root-mean-square velocity at temperature T is $\sqrt{\frac{3k_BT}{m}}$, with `fs` we rescale the velocity of each particle to the proper value at temperature T. Since the distribution `randv()` has an expectation value of 0, the velocity of center of mass `sumv` must be small compared to `v(i)`, and we need not to change `sumv2`. By subtracting `sumv` from `v(i)`, the center of mass is made still.

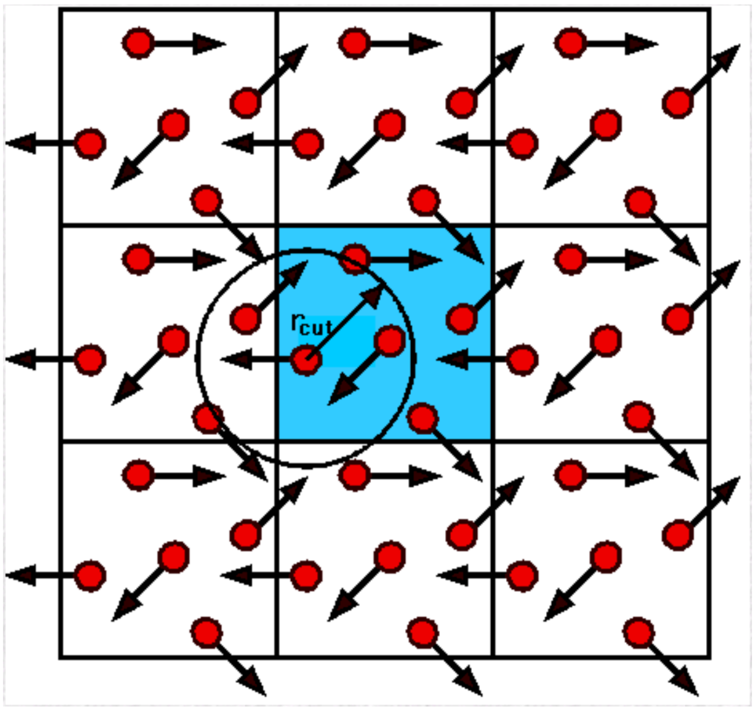
2.1.2 The Force Calculation

Algorithm 2 Calculation of the Forces

```
function FORCE
    procedure SET ENERGY AND FORCES TO 0
        en=0
        for i=1: npart do
            f(i)=0
        end for
    end procedure
    procedure DETERMINE ENERGY AND FORCES
        for i=1: npart-1 do
            for j=i+1: npart do                          ▷ loop over all pairs
                xr=x(i)-x(j)
                if xr<-0.5*box then
                    xr=xr+box                            ▷ periodic boundary condition
                end if
                if xr>=0.5*box then
                    xr=xr-box                            ▷ minimum-image convention
                end if
                r2=xr**2
                if r2<rc2 then                           ▷ test cutoff
                    f(i)=f(i)+force(xr)
                    f(j)=f(j)-force(xr)                  ▷ update force
                    en=en+potential(r2)                  ▷ update energy
                end if
            end for
        end for
    end procedure
    return
end function
```

Here comes the most time-consuming part in a molecular dynamics simulation. We assume the forces acting on every particle are pairwise additive, and the pairwise potential energy only depends on the distance between two particles. We have to evaluate all $N(N-1)/2$ pair distances to figure out the total potential energy, and for every particle we have to add up $N-1$ forces exerted on it by other particles.

When calculating the relative distance between particles `i` and `j`, we use periodic boundary condition and minimum-image convention. As the figure below shows, the intermolecular interaction between `i` and `j` is replaced by the interaction between `i` and the nearest periodic image of `j`.



2.1.3 Integrating the Equations of Motion

Algorithm 3 Integration of Equations of Motion

```
 1: function INTEGRATE
 2:     sumv=0
 3:     sumv2=0
 4:     for i=1: npart do                              ▷ MD loop
 5:         xx=2*x(i)-xm(i)+delt**2*f(i)               ▷ Verlet algorithm
 6:         vi=(xx-xm(i))/(2*delt)                     ▷ velocity
 7:         sumv=sumv+vi                               ▷ velocity of center of mass
 8:         sumv2=sumv2+vi**2                          ▷ total kinetic energy
 9:         xm(i)=x(i)                                 ▷ update previous positions
10:         x(i)=xx                                    ▷ update current positions
11:     end for
12:     temp=sumv2/(3*npart)                           ▷ instantaneous temperature
13:     etot=(en+0.5*sumv2)/npart                      ▷ total energy per particle
14:     sumv=sumv/npart
15:     return
16: end function
```

It is worth noting that throughout the simulation, the total energy `etot` should remain approximately constant, and the velocity of center of mass `sumv` should also remain zero.

The system we are simulating is usually non-dissipative, and we want to obtain a symplectic and reversible integrator, in order to preserve the total energy and improve robustness. Liouville theorem tells us that the Verlet or velocity Verlet algorithm satisfies these requirements. A proof can be found here

Combining the above three algorithms, we can realize a complete molecular dynamics simulation program.

Algorithm 4 Simple MD Program

```
1: call INIT                                              ▷ initialize
2: t=0
3: while t<tmax do
4:     call FORCE                                         ▷ determine forces
5:     call INTEGRATE                                     ▷ integrate equations of motion
6:     t=t+delt
7:     call SAMPLE                                        ▷ sample averages
8: end while
9: return
```

The function `SAMPLE` is not defined in the pseudocodes above. It calculates the average quantities along the simulation trajectory.

2.2 Monte Carlo Simulation

As opposed to molecular dynamics, Monte Carlo simulation does not follow any real or simulated trajectories. We do not need forces or velocities. The kinetic energy or other functions that only depend on the particle velocities have averages that can be easily carried out, because the velocities follow Maxwell-Boltzmann distribution and can be averaged analytically. Therefore we only need to sample in the configuration space $\boldsymbol{r}^N$. The 3N-dimensional integral is written as

$$S = \int_D F(\boldsymbol{R})\mathrm{d}\boldsymbol{R} \tag{2.24}$$

where D is the domain of integration. Let us assume a normalized domain $\int_D \mathrm{d}\boldsymbol{R} = 1$. Following the idea of importance sampling introduced by Metropolis, we sample M random points $\boldsymbol{R}_i$ from a nonuniform probability distribution $W(\boldsymbol{R})$

3 Simulations

The specifications and requisites of our simulations are clearly provided in the assignment. Let us jump right into the tasks.

3.1 TASK1

Different potentials as functions of r.

Table 1: potential functions

Potential	Lennard-Jones $U^{LJ}(r)$	Atomic $\phi^{A}(r)$	Colloidal $\phi^{C}(r)$
Minimum r_{min}	$2^{1/6}$	1.155	1.055
Cut-off r_c	2.5	2.0	1.2
Prefactor α	N.A.	1	114

Their plots are as follows.

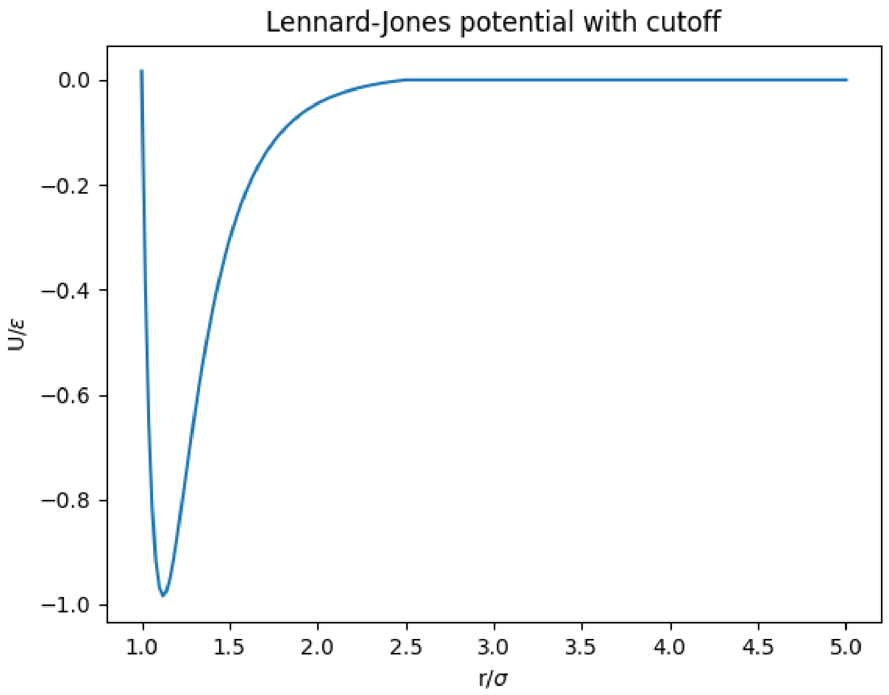


Figure 2: Lennard-Jones potential with cutoff

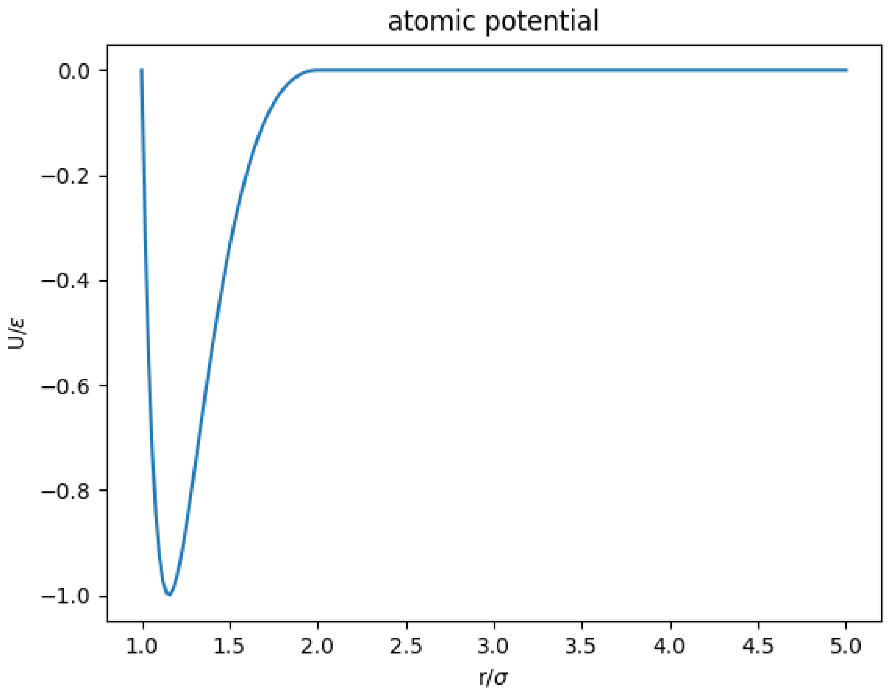


Figure 3: atomic potential

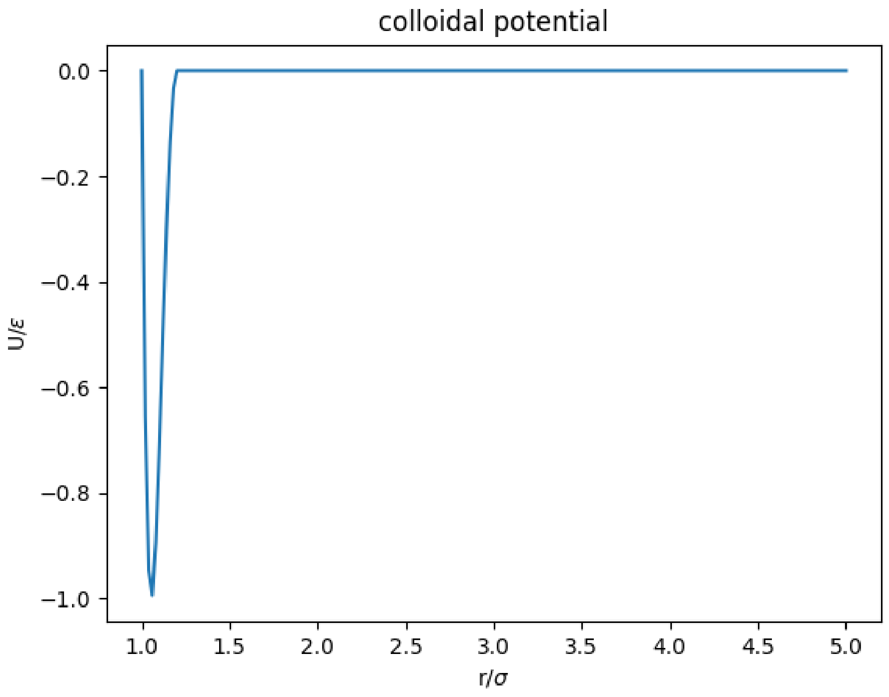


Figure 4: colloidal potential

3.2 TASK2

This task requires me to do the simulation with two initial conditions: random positions and ordered lattice, using MC and MD methods.

3.2.1 Random positions

(a) Monte Carlo simulation

The initial configuration is generated like this

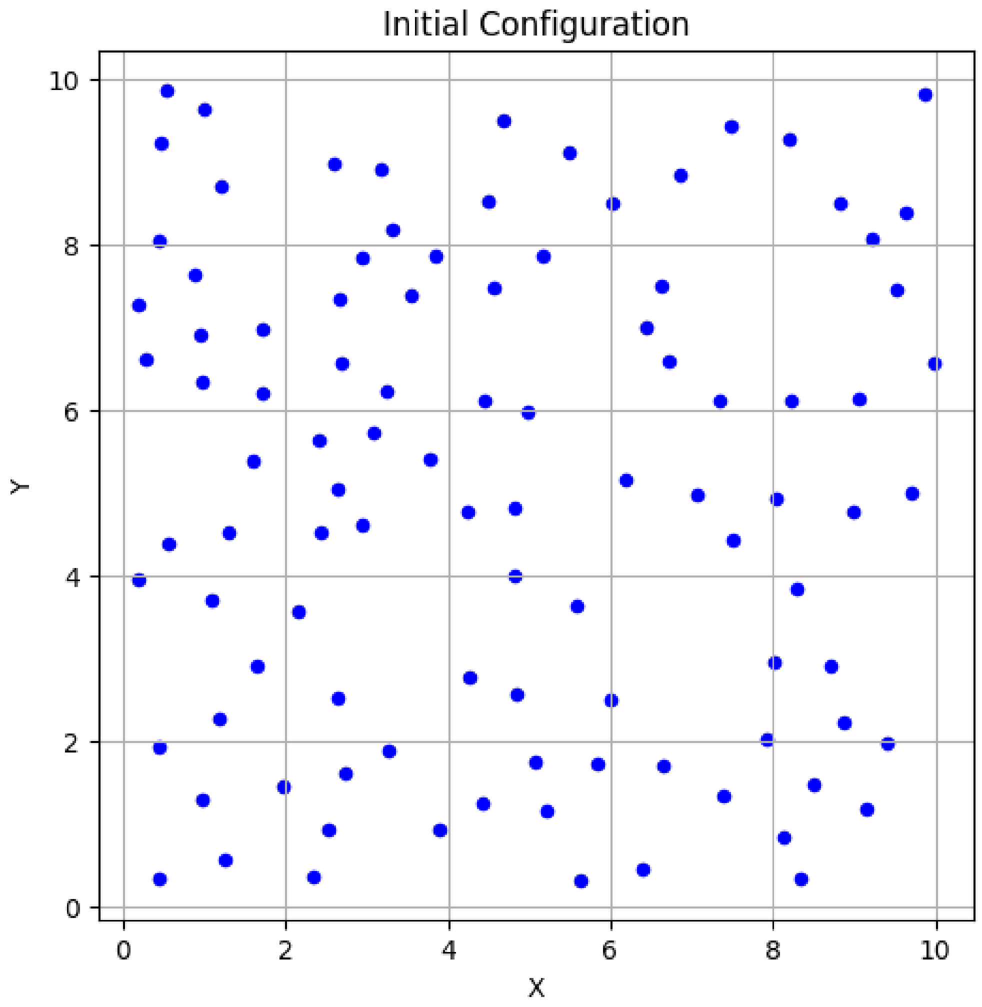


Figure 5: initial configuration: random positions

I should make sure every two particles do not overlap with each other. Using the atomic potential, and fixing the temperature and density to $T_1 = 0.728, \rho_1 = 0.8442$, I reach a final configuration like this

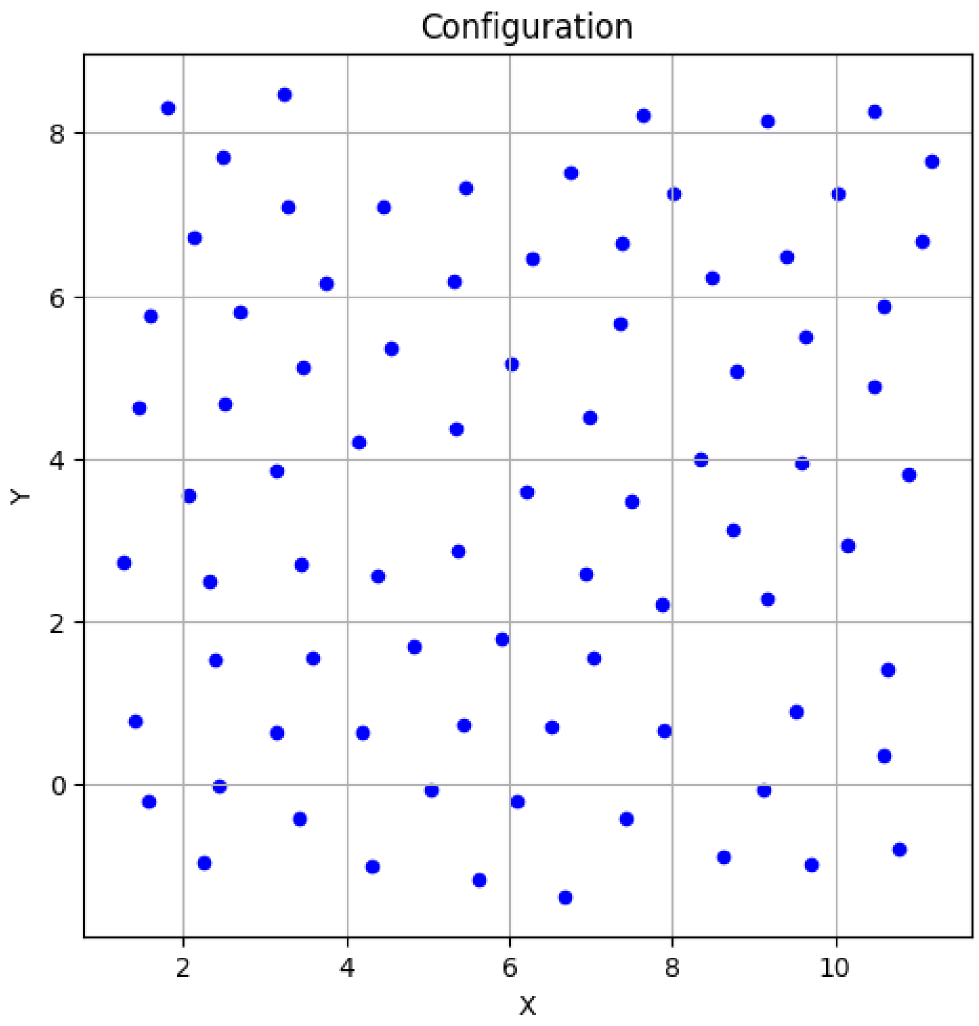


Figure 6: MC final configuration 1

This is after 10000 steps of simulation. The total potential energy as a function of time is plotted as follows

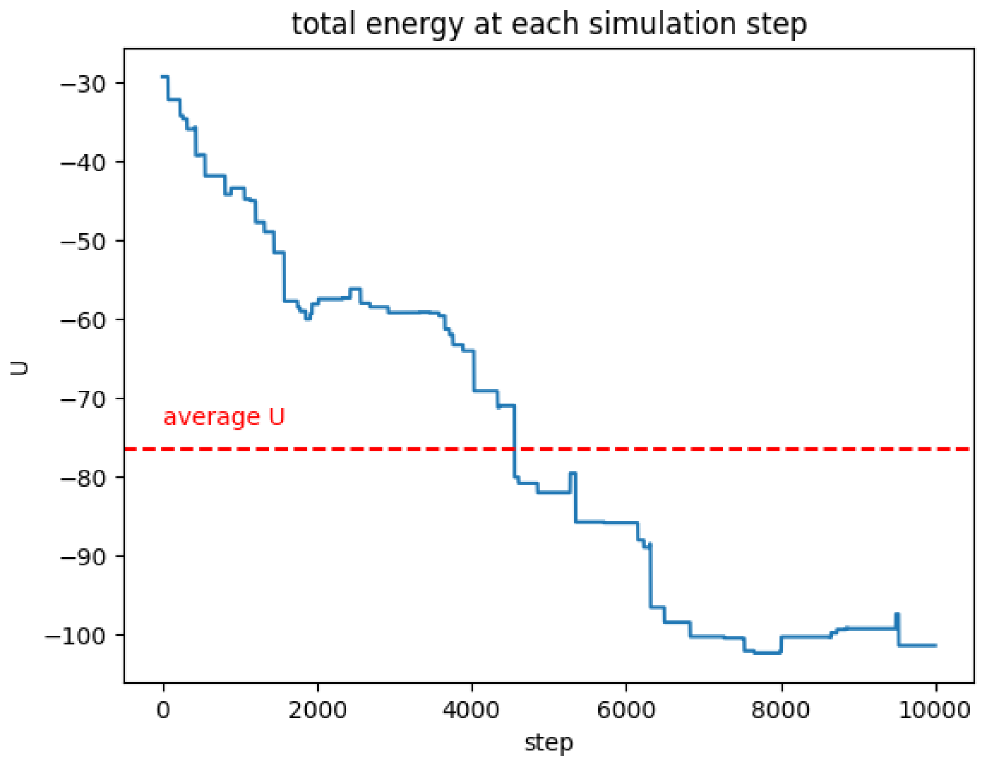


Figure 7: MC energy versus step 1

My citerion for testing the equilibration is:

- The energy stays negative.
- The energy only fluctuates within a relatively small range, without showing a trend of increasing or decreasing.

According to my criterion, the equilibrium is reached at around step 6500, so if I want to evaluate the real average energy, I can only sum up those steps after equilibrium. The result is

$$\langle U \rangle = -100.346 \tag{3.1}$$

The average energy per particle is

$$\langle u \rangle = \frac{\langle U \rangle}{N} = -1.239 \tag{3.2}$$

I can also analyze the fluctuation of energy

$$\delta U^2 = \left\langle U^2 \right\rangle - \left\langle U \right\rangle^2 = 1.330 \tag{3.3}$$

The relative fluctuation of the energy is

$$\left| \frac{\delta U}{\langle U \rangle} \right| \approx 1.15\% \tag{3.4}$$

Choosing another fixed condition $T_2 = 1.0,\ \rho_2 = 0.1$, because the density is far smaller than the previous condition, the equilibrium is sooner reached.

The final configuration is

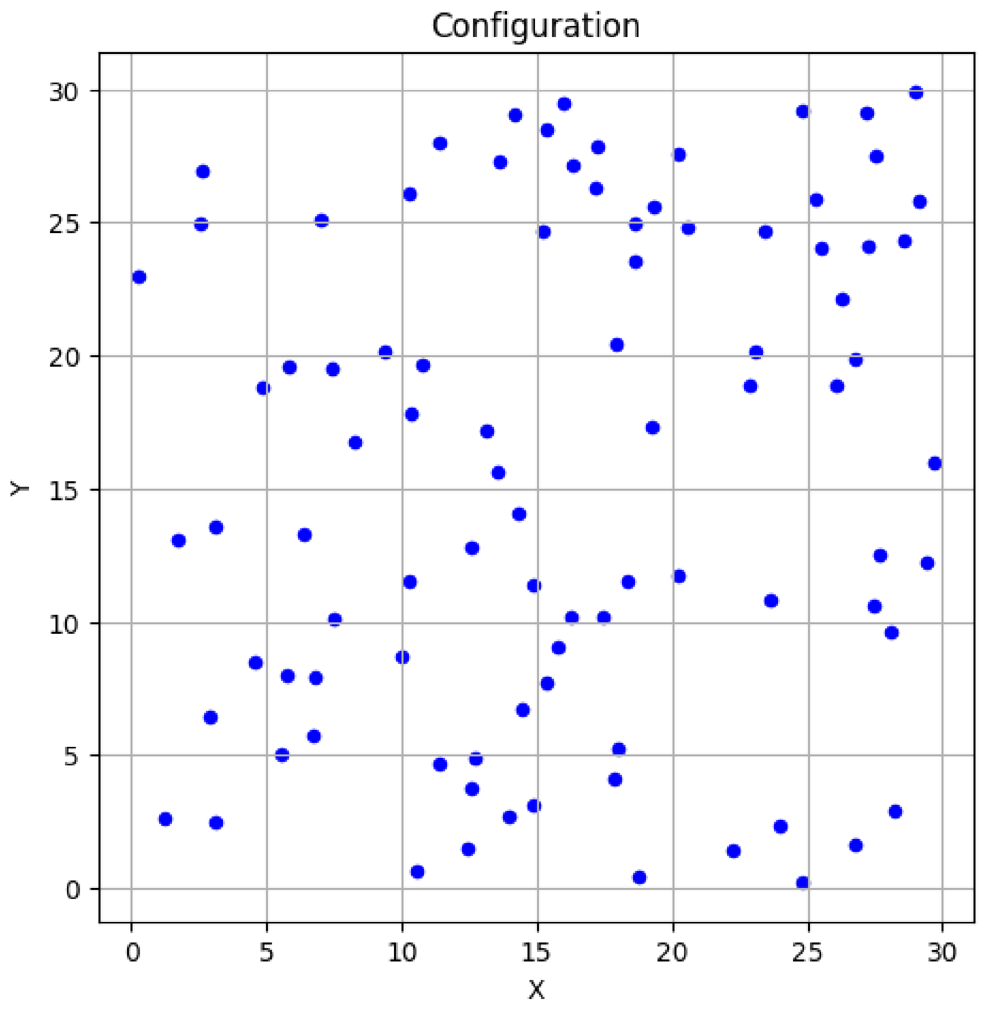


Figure 8: MC final configuration 2

This is after 5000 steps of simulation. The total potential energy as a function of time is plotted as follows

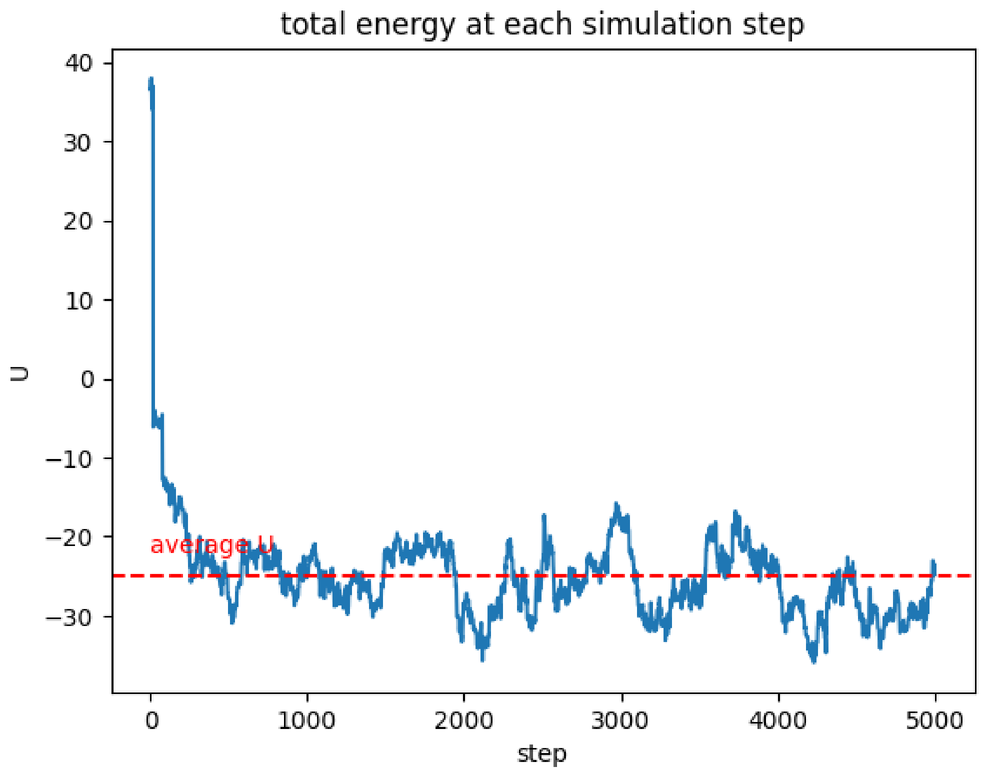


Figure 9: MC energy versus step 2

The average energy is

$$\langle U \rangle = -25.893 \tag{3.5}$$

The average energy per particle is

$$\langle u \rangle = \frac{\langle U \rangle}{N} = -0.2877 \tag{3.6}$$

I can also analyze the fluctuation of energy

$$\delta U^2 = \left\langle U^2 \right\rangle - \left\langle U \right\rangle^2 = 14.90 \tag{3.7}$$

The relative fluctuation of the energy is

$$\left| \frac{\delta U}{\langle U \rangle} \right| \approx 14.9\% \tag{3.8}$$

Obviously, higher temperature and lower density result in higher average energy and greater fluctuation in energy.

(b) molecular dynamics simulation

In MD simulation, I need to keep an eye on both the positions and velocities of the particles. In addition to solving the equations of motion, I also have to maintain a constant temperature for the NVT ensemble. An Anderson thermostat is applied in my simulation. To perform the simulation with Anderson thermostat, I need two parameters: T and ν. T is the controlled temperature of the system, and ν is the frequency of stochastic collisions which determine the strength of the coupling to the heat bath. If successive collisions are uncorrelated, then the distribution of time intervals between two successive stochastic collisions is of Poisson form

$$P(X = k) = f(k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{3.9}$$

Its expectation value is $E(X) = \lambda$, which means after every collision, the next collision is expected to come in λ simulation steps. The frequency of collisions is $\nu = \frac{1}{\lambda}$.

In a constant temperature simulation, I first specify the initial positions and momenta. Then I will integrate the equations of motion until the first stochastic collision, which happens at the time determined by a random variable following Poisson distribution. I randomly select a particle which suffers this collision, and its momentum is instantaneously switched to a value generated from Maxwell-Boltzmann distribution under temperature T. The Newtonian equations are then integrated until the next collision. This process is repeated to the end of the simulation.

For the first set of fixed conditions, $T1 = 0.728, \rho_1 = 0.8442$, I run the simulation for 5000 steps. I have chosen $\lambda = 3$, and each time step $\delta t = 0.014$, getting a final configuration as the figure below

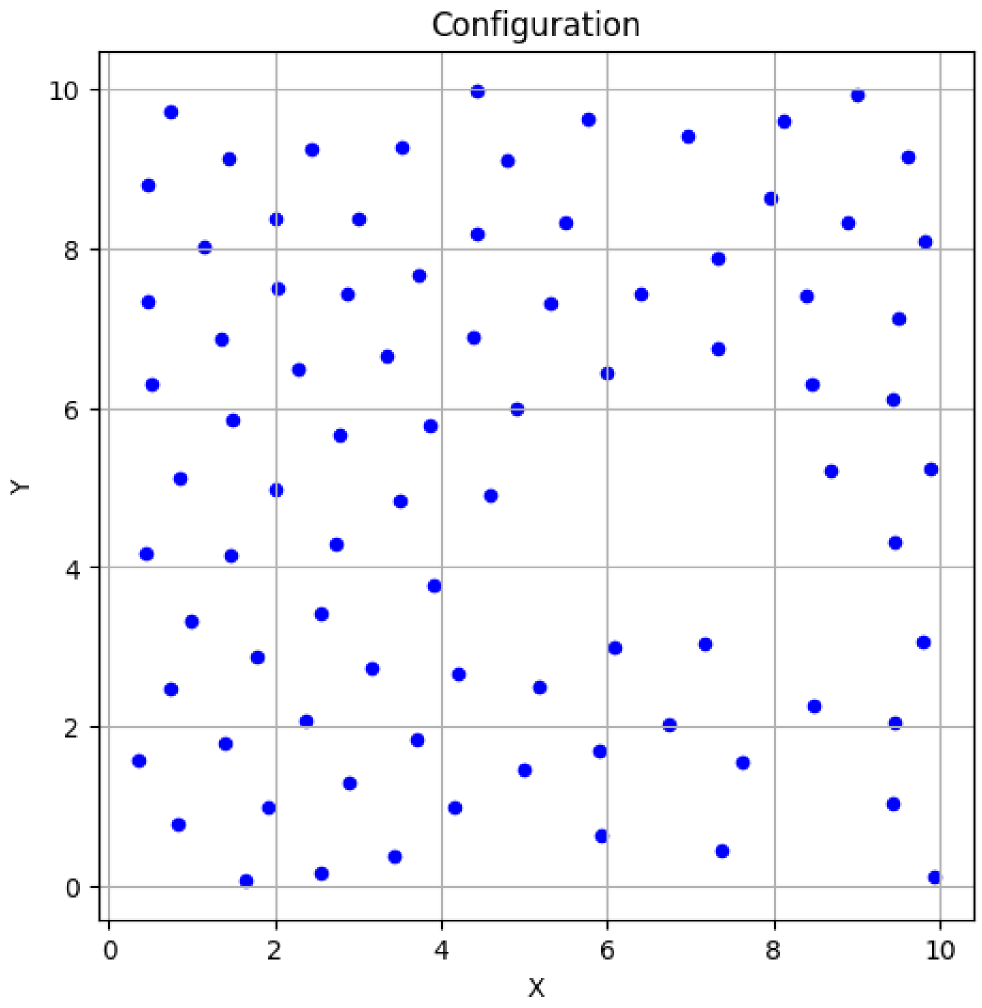


Figure 10: MD final configuration 1

From the final configurations of MC and MD simulations, one may be surprised to discover that the particles tend to be evenly and regularly placed in the box – some even exhibit hexagonal or square lattice structure! But these symmetries are not presupposed. This may explain the formation of solids, especially crystals.

The total potential energy as a function of time is plotted as follows

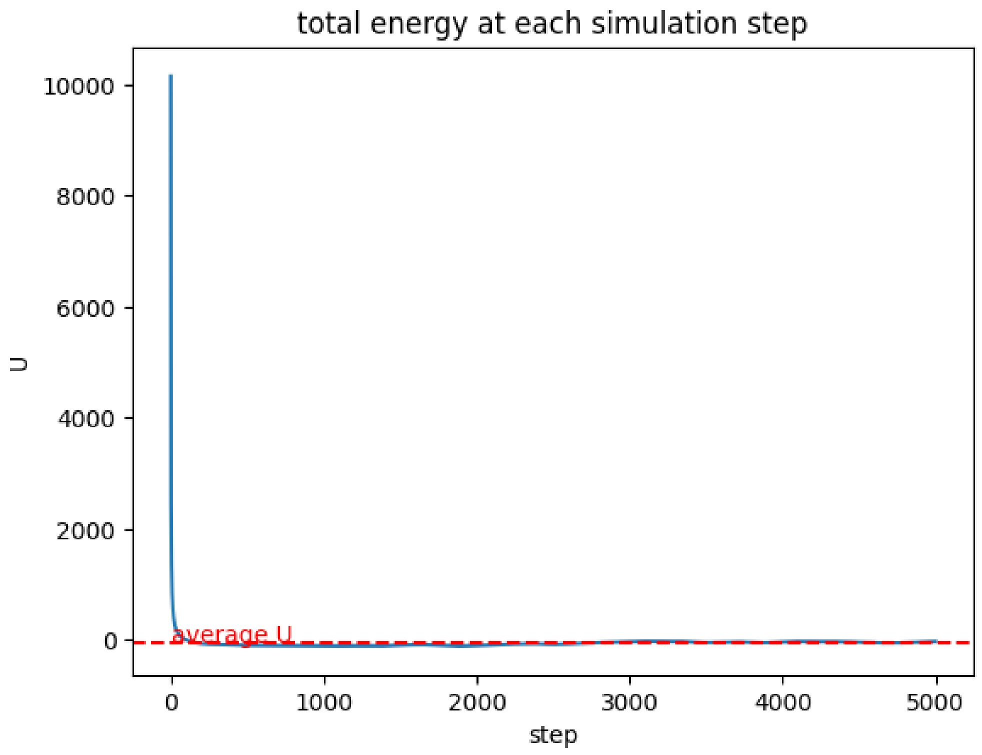


Figure 11: MD energy versus step 1

Since the initial energy is too large, I should truncate the list of energies, leaving only the equilibrium states.

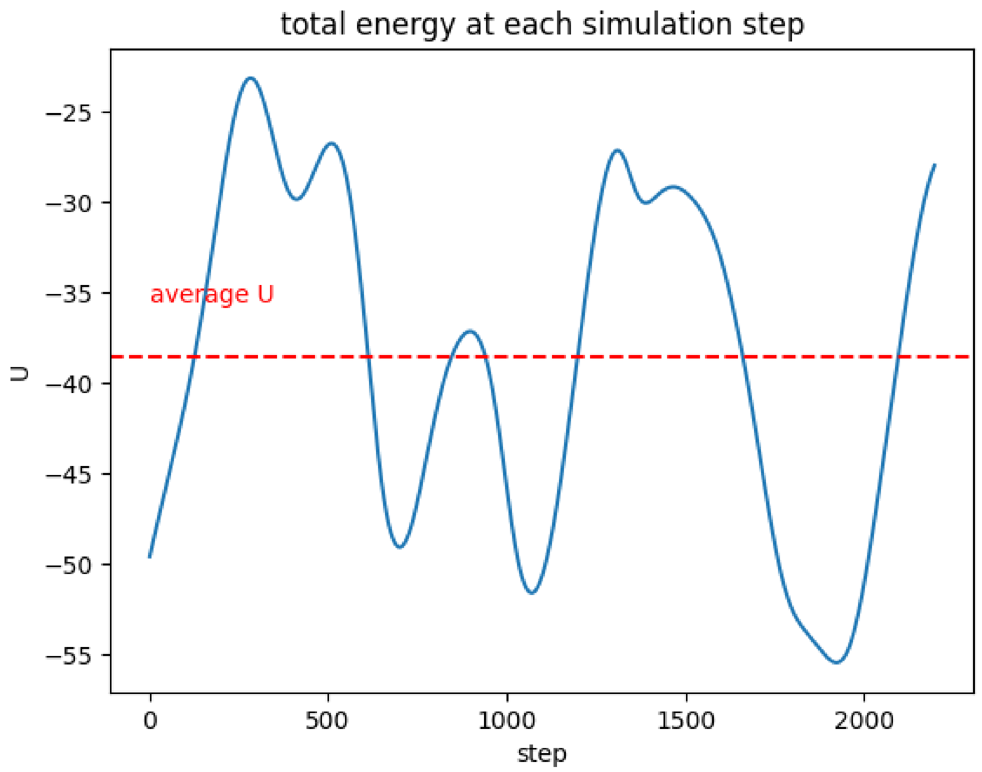


Figure 12: MD energy versus step 2

I can now calculate the statistical values

$$\langle U \rangle = -38.557 \tag{3.10}$$

$$\delta U^2 = 87.544 \tag{3.11}$$

It can be seen that the average energy is a lot higher than in MC simulation (for the same fixed condition), and the variance in U is also larger. Although the final configuration resembles that in MC simulation, I wonder if I have not reached the real equilibrium.

Therefore I reset λ and δt, and manage to find a better equilibrium.

3.2.2 Ordered lattice

3.3 TASK3

The aim of this task is to study the relation between $\langle U \rangle$, δU and the system size N.

3.4 TASK4

This task asks me to calculate the radial distribution function $g(r)$ from the simulations. Suppose we have N particles, and $\rho_N(\boldsymbol{r}_1, \cdots, \boldsymbol{r}_N)$ is called the configuration probability density, meaning that the probability for the N particles being respectively in $\mathrm{d}^3\boldsymbol{r}_1, \cdots, \mathrm{d}^3\boldsymbol{r}_N$ is $\rho_N(\boldsymbol{r}_1, \cdots, \boldsymbol{r}_N)\mathrm{d}^3\boldsymbol{r}_1 \cdots \mathrm{d}^3\boldsymbol{r}_N$. For a simulation box with volume V, the average density of the particles is $n = \frac{N}{V}$.

We then define the pair distribution function

$$F_2(\boldsymbol{r}_1, \boldsymbol{r}_2) \equiv N(N-1) \int \cdots \int \rho_N(\boldsymbol{r}_1, \cdots, \boldsymbol{r}_N)\mathrm{d}^3\boldsymbol{r}_3 \cdots \mathrm{d}^3\boldsymbol{r}_N \tag{3.12}$$

As a homogeneous system has translational invariance, $F_2(\boldsymbol{r}_1, \boldsymbol{r}_2) = F_2(\boldsymbol{r}_2 - \boldsymbol{r}_1)$, we introduce the radial distribution function

$$F_2(\boldsymbol{r}_2 - \boldsymbol{r}_1) \equiv n^2 g(\boldsymbol{r}_2 - \boldsymbol{r}_1) \tag{3.13}$$

Integrating Eq. **??**, we get

$$\begin{aligned} N(N-1) &= \iint F_2(\boldsymbol{r}_2 - \boldsymbol{r}_1) \\ &= n^2 \iint g(\boldsymbol{r}_2 - \boldsymbol{r}_1)\mathrm{d}\boldsymbol{r}^3 \end{aligned} \tag{3.14}$$

3.5 TASK5

and eventually arrived to the

4 Results and Conclusions

Here you briefly summarize your findings. Did you learn any new physics? Was everything as expected?

In this section you will need to show your experimental results. Use tables and

In this project, TASK 2 is the most crucial part. It includes two sets of initial conditions: random positions and ordered lattice, involves two sets of fixed conditions: T_1, ρ_1 and T_2, ρ_2, and applies both the MC and MD simulation methods. To finish this task, I must have thorough understanding for both methods. Moreover, comparing results obtained from different conditions and different simulation methods, I can get a glimpse of the physics inside.

Table 2: Every table needs a caption.

x (m)	V (V)
0.0044151	0.0030871
0.0021633	0.0021343
0.0003600	0.0018642
0.0023831	0.0013287

For example, it is easy to conclude that the experiment and theory match each other rather well if you look at

5 Future Work

Since you had limited time to work on this project, what questions are left outstanding? What would be your next steps?

References

[1] Daan Frenkel and Berend Smit. *Understanding Molecular Simulation: From Algorithms to Applications*. Academic Press, Cambridge, the United States, 2002.

[2] University College London. Periodic boundary condition. https://www.ucl.ac.uk/~ucfbasc/Theory/pbc-mi.html#:~:text=In%20other%20cases%20the%20interaction,image%20of%20any%20given%20atom. Accessed: 2023-06-25.

[3] Tao Pang. *An Introduction to Computational Physics*. Cambridge University Press, New York, the United States, 2006.

[4] Piero Procacci and Massimo Marchi. Liouville formalism: a tool for building symplectic and reversible integrators. http://www1.chim.unifi.it/orac/MAN/node7.html. Accessed: 2023-06-25.

[5] Martin O. Steinhauser. *Computational Multiscale Modeling of Fluids and Solids: Theory and Applications*. Springer-Verlag, Berlin, Germany, 2008.