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

Zhicheng Zhang

July 1, 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:** Some of my codes were run by PyPy3 interpreter. Documentations can be found here PyPy documentation.

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 2.5 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

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

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

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

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

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

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 2.12, 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

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

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

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

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

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

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. 2.22. 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 [1, 6].

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[1].

2.1.1 Initialization

Algorithm 1 Initialization

```
1: function INIT
2:     sumv=0                                          ▷ velocity of center of mass
3:     sumv2=0                                         ▷ mean squared velocity
4:     procedure SET INITIAL POSITIONS AND VELOCITIES
5:         for i=1: npart do
6:             x(i)=lattice_pos(i)                     ▷ place the particles on a lattice
7:             v(i)=randv()                            ▷ random distributed velocities
8:             sumv=sumv+v(i)                          ▷ sum for total velocity
9:             sumv2=sumv2+v(i)**2                     ▷ sum for total squared velocity
10:        end for
11:        sumv=sumv/npart                             ▷ velocity of center of mass
12:        sumv2=sumv2/npart                           ▷ mean squared velocity
13:    end procedure
14:    procedure RESCALE VELOCITIES AND GET PREVIOUS POSITIONS
15:        fs=sqrt(3*temp/sumv2)                       ▷ scale factor for velocities
16:        for i=1: npart do
17:            v(i)=(v(i)-sumv)*fs     ▷ set desired kinetic energy and fix centre of mass
18:            xm(i)=x(i)-v(i)*dt                      ▷ get position of previous timestep
19:        end for
20:    end procedure
21:    return
22: 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

```
1: function FORCE
2:     procedure SET ENERGY AND FORCES TO 0
3:         en=0
4:         for i=1: npart do
5:             f(i)=0
6:         end for
7:     end procedure
8:     procedure DETERMINE ENERGY AND FORCES
9:         for i=1: npart-1 do
10:            for j=i+1: npart do                          ▷ loop over all pairs
11:                xr=x(i)-x(j)
12:                if xr¡-0.5*box then
13:                    xr=xr+box                            ▷ periodic boundary condition
14:                end if
15:                if xr¿=0.5*box then
16:                    xr=xr-box                            ▷ minimum-image convention
17:                end if
18:                r2=xr**2
19:                if r2¡rc2 then                           ▷ test cutoff
20:                    f(i)=f(i)+force(xr)
21:                    f(j)=f(j)-force(xr)                  ▷ update force
22:                    en=en+potential(r2)                  ▷ update energy
23:                end if
24:            end for
25:        end for
26:    end procedure
27:    return
28: 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`.

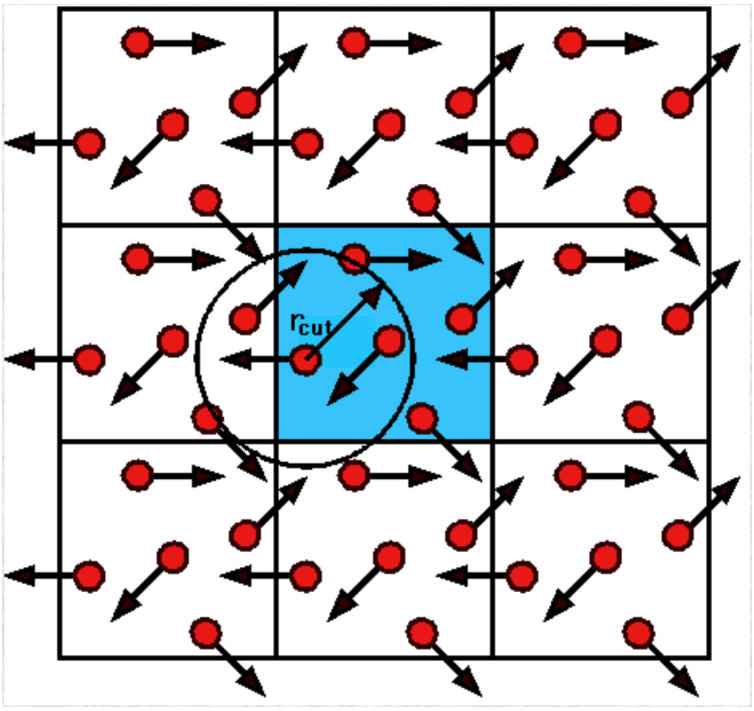

Figure 1: periodic boundary condition and minimum-image convention[3]

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 [5]. The Verlet algorithm is characterized by the following equations

$$\begin{cases} r(t+\delta t) = 2r(t) - r(t-\delta t) + \dfrac{F(t)}{m}\delta t^2 \\ v(t) = \dfrac{r(t+\delta t) - r(t-\delta t)}{2\delta t} \end{cases} \tag{2.24}$$

We only need the first equation to obtain the trajectory, but we still need the velocities to calculate the temperature and energy, thus the second equation.

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

Algorithm 4 Simple MD Program

```
call INIT                                   ▷ initialize
t=0
while t¡tmax do
    call FORCE                              ▷ determine forces
    call INTEGRATE                          ▷ integrate equations of motion
    t=t+delt
    call SAMPLE                             ▷ sample averages
end while
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

$$S = \int_D F(\boldsymbol{R})\mathrm{d}\boldsymbol{R} \tag{2.25}$$

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 $W(\boldsymbol{R})$[4]. Then the integral is estimated by

$$S \approx \frac{1}{M}\sum_{i=1}^{M}\frac{F(\boldsymbol{R}_i)}{W(\boldsymbol{R}_i)} \tag{2.26}$$

It can be shown that the standard deviation of the estimated integral from the exact integral is

$$\sigma_S^2 = \left\langle \left(\frac{1}{M} \sum_{i=1}^{M} \frac{F(\boldsymbol{R}_i)}{W(\boldsymbol{R}_i)} - \left\langle \frac{F(\boldsymbol{R})}{W(\boldsymbol{R})} \right\rangle \right)^2 \right\rangle = \frac{1}{M} \left(\left\langle \frac{F^2}{W^2} \right\rangle - \left\langle \frac{F}{W} \right\rangle^2 \right) \tag{2.27}$$

To minimize σ_S^2, $G(\boldsymbol{R}) = F(\boldsymbol{R})/W(\boldsymbol{R})$ should be nearly a constant. Compare this with the canonical ensemble average

$$\langle A \rangle = \int_D A(\boldsymbol{R}) W(\boldsymbol{R}) \mathrm{d}\boldsymbol{R} \tag{2.28}$$

where the weight $W(\boldsymbol{R})$ is the probability density function

$$W(\boldsymbol{R}) = \frac{e^{-U(\boldsymbol{R}/k_B T)}}{\int_D e^{-U(\boldsymbol{R}'/k_B T)} \mathrm{d}\boldsymbol{R}'} \tag{2.29}$$

Metropolis's method generates a sequence of sample points in the configuration space. The probability of transition from the old sample point to the new sample point obeys the principle of detailed balance

$$W(\boldsymbol{R}) T(\boldsymbol{R} \to \boldsymbol{R}') = W(\boldsymbol{R}') T(\boldsymbol{R}' \to \boldsymbol{R}) \tag{2.30}$$

The ratio of transition probabilities is

$$\frac{T(\boldsymbol{R} \to \boldsymbol{R}')}{T(\boldsymbol{R}' \to \boldsymbol{R})} = \frac{W(\boldsymbol{R}')}{W(\boldsymbol{R})} = e^{-(U(\boldsymbol{R}') - U(\boldsymbol{R}))/k_B T} \tag{2.31}$$

It is easily verified that the following choice of transition probability (rate) arrives at detailed balance

$$T(\boldsymbol{R} \to \boldsymbol{R}') = \min \left(1, e^{-\beta(U(\boldsymbol{R}') - U(\boldsymbol{R}))} \right) \tag{2.32}$$

Algorithm 5 Metropolis algorithm

```
call INIT_POSITION                          ▷ initialize starting positions of all particles
for icycl=1: ncycl do
    o=randint(1, npart)                     ▷ randomly select a particle
    eno=ENER(x(o))                          ▷ old energy
    xn=x(o)+randx()                         ▷ random displacement
    enn=ENER(xn)                            ▷ new energy
    if ranf()¡exp(-beta*(enn-eno)) then     ▷ Metropolis acceptance rule
        x(o)=xn
    end if
    call SAMPLE
end for
return
```

Finally, a version of Metropolis Monte Carlo simulation program is established.

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.

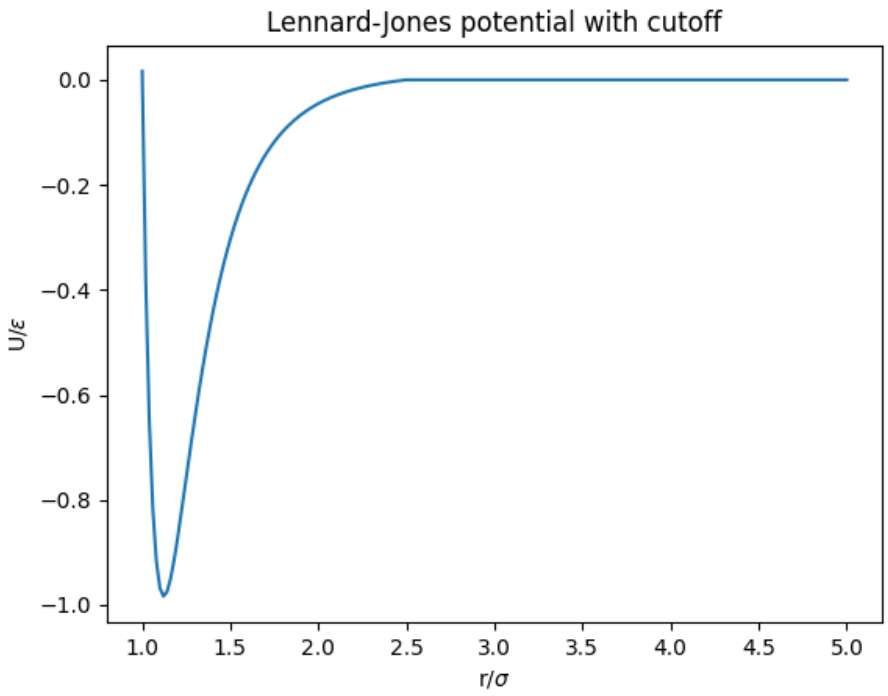

Figure 2: Lennard-Jones potential with cutoff

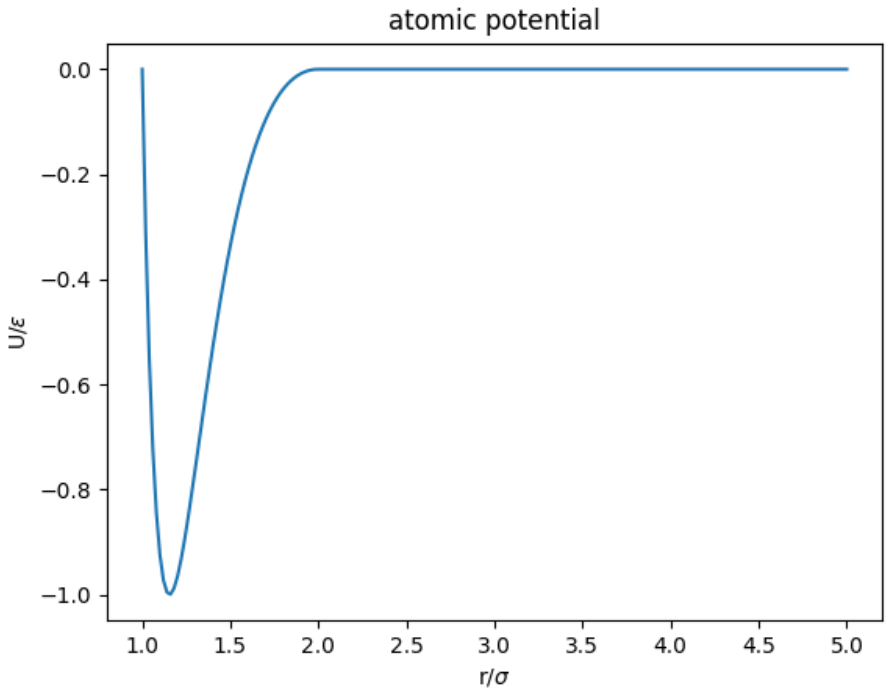

Figure 3: atomic potential

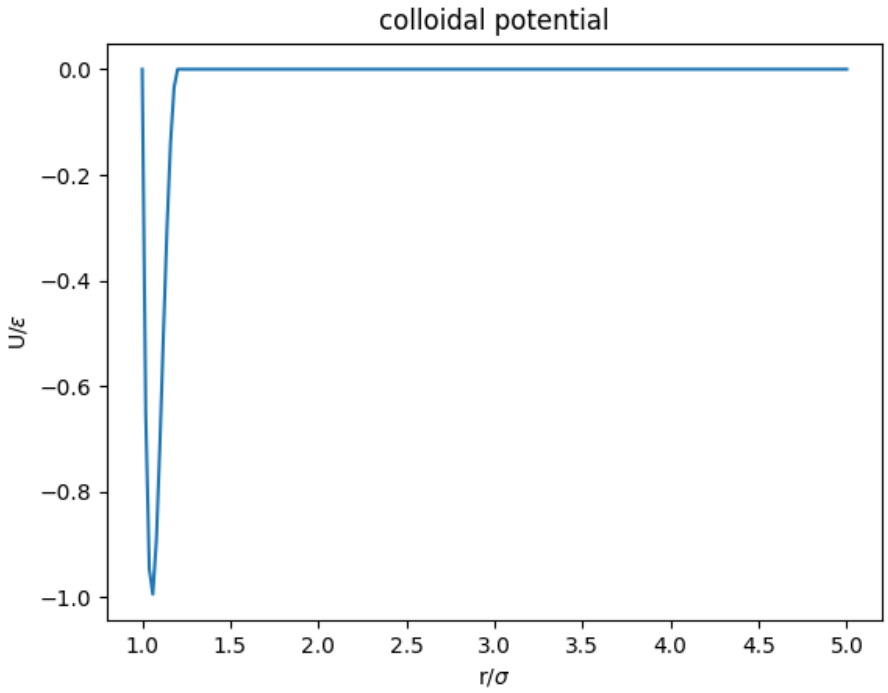

Figure 4: colloidal potential

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

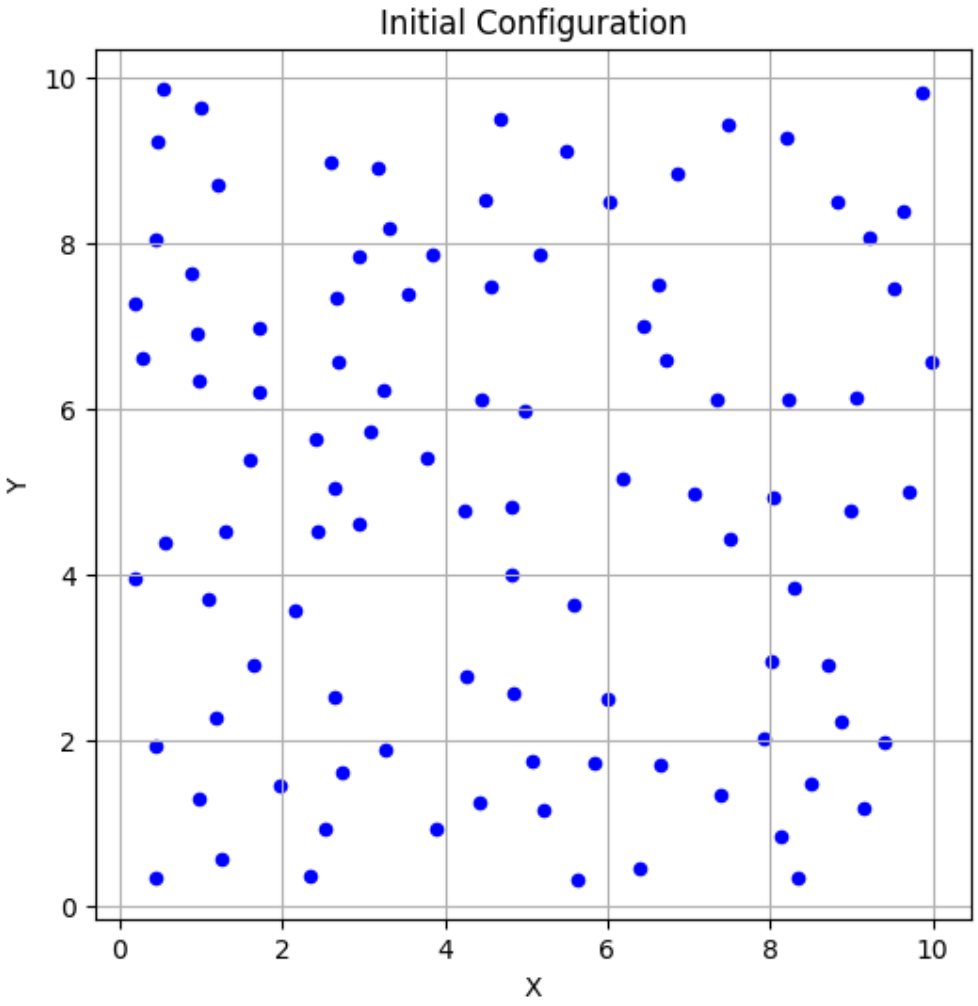

Figure 5: initial configuration: random positions

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

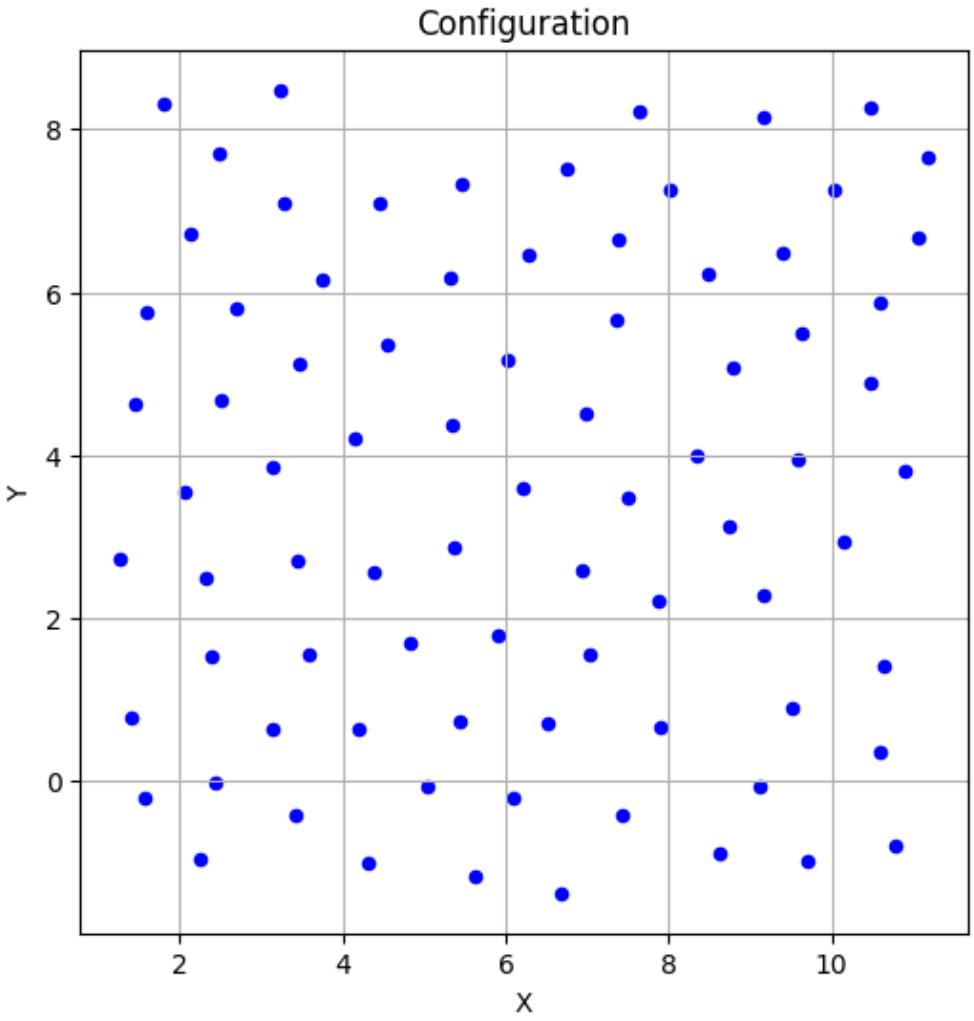

Figure 6: MC final configuration 1

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

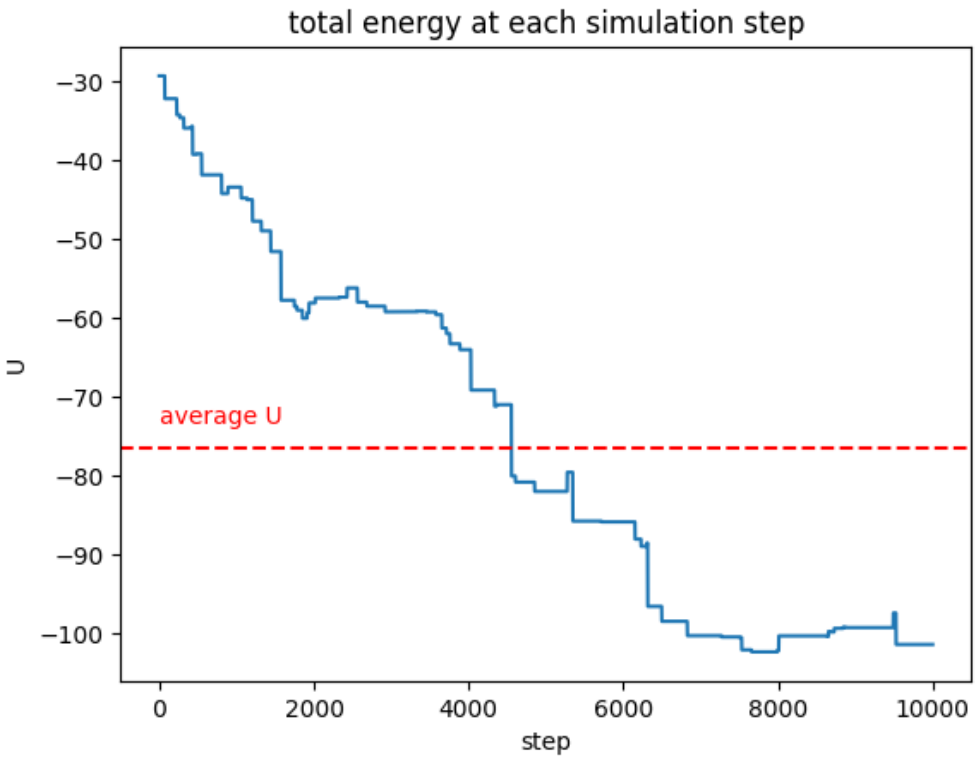

Figure 7: MC energy versus step 1

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

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

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

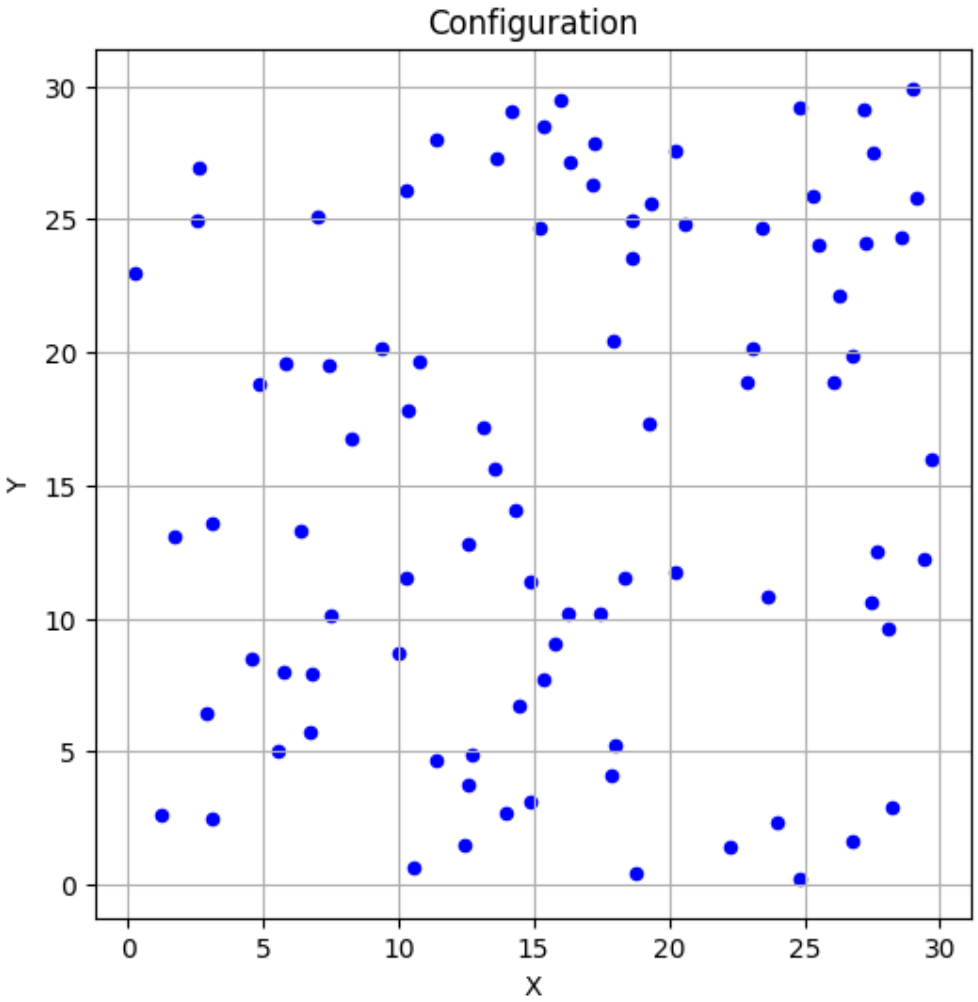

Figure 8: MC final configuration 2

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

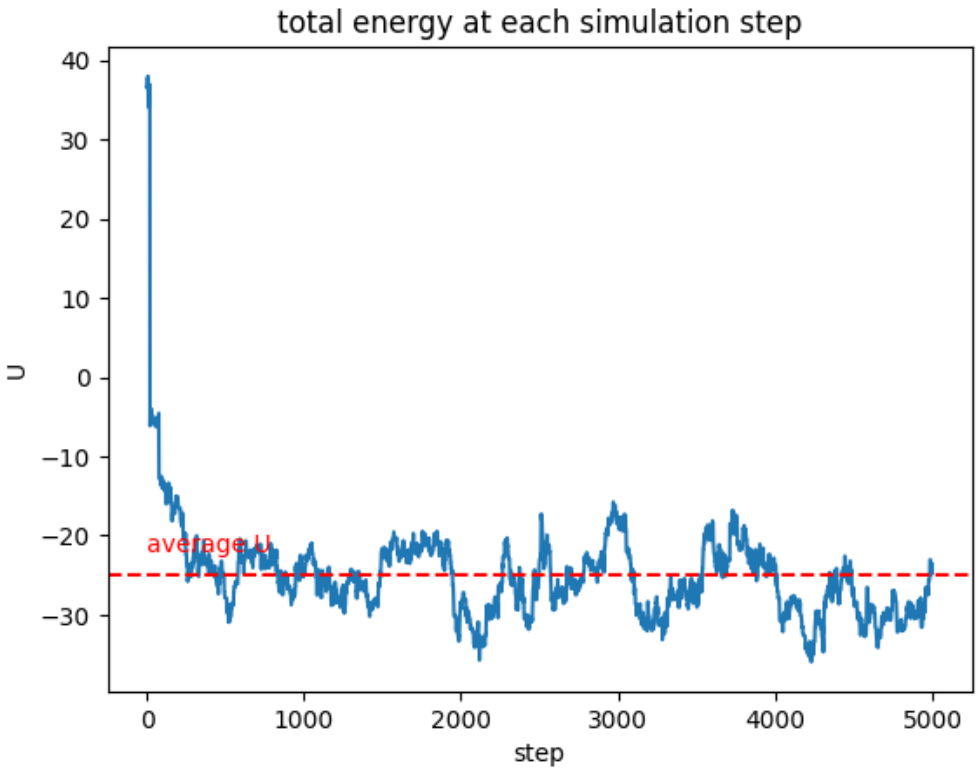

Figure 9: MC energy versus step 2

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

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

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, $T_1 = 0.728, \rho_1 = 0.8442$, I run the simulation for 5000 steps. I have chosen $\lambda = 3$, each time step $\delta t = 0.014$, and initial temperature $T_0 = 0.03 * T_1$, getting a final configuration as the figure below

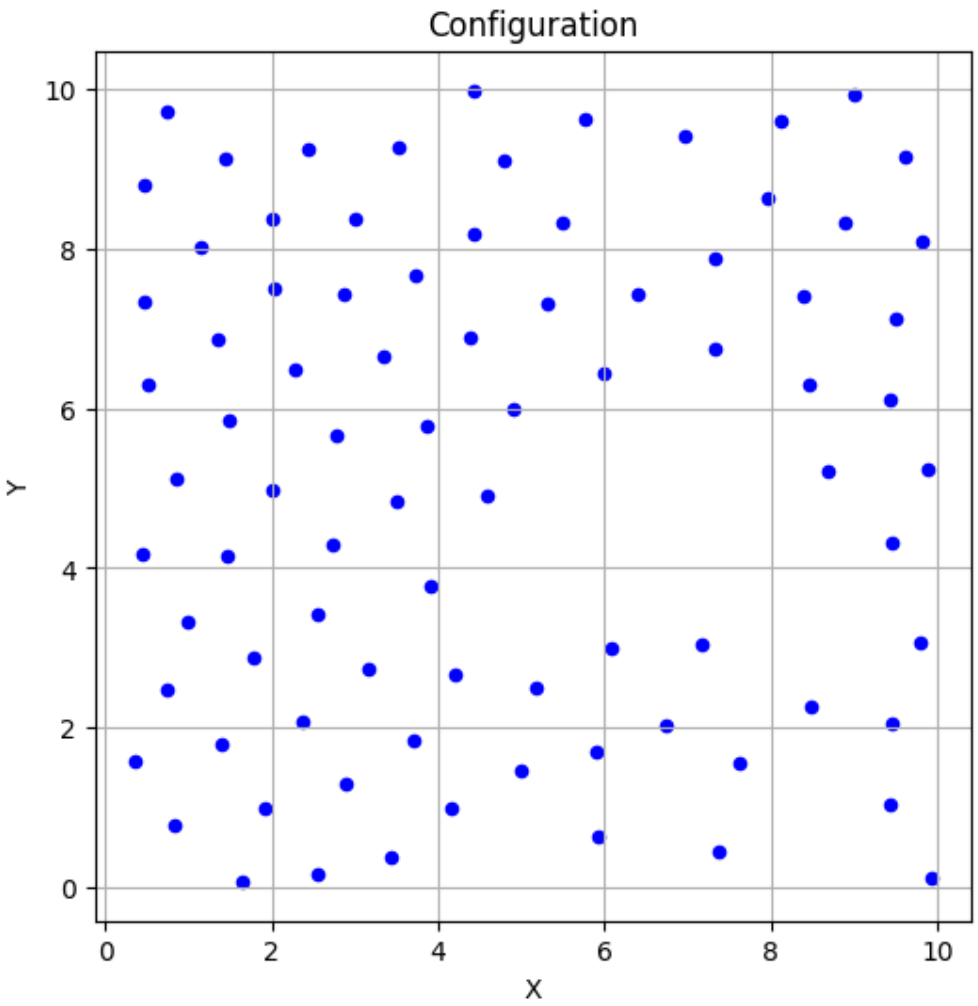

Figure 10: MD final configuration 1

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

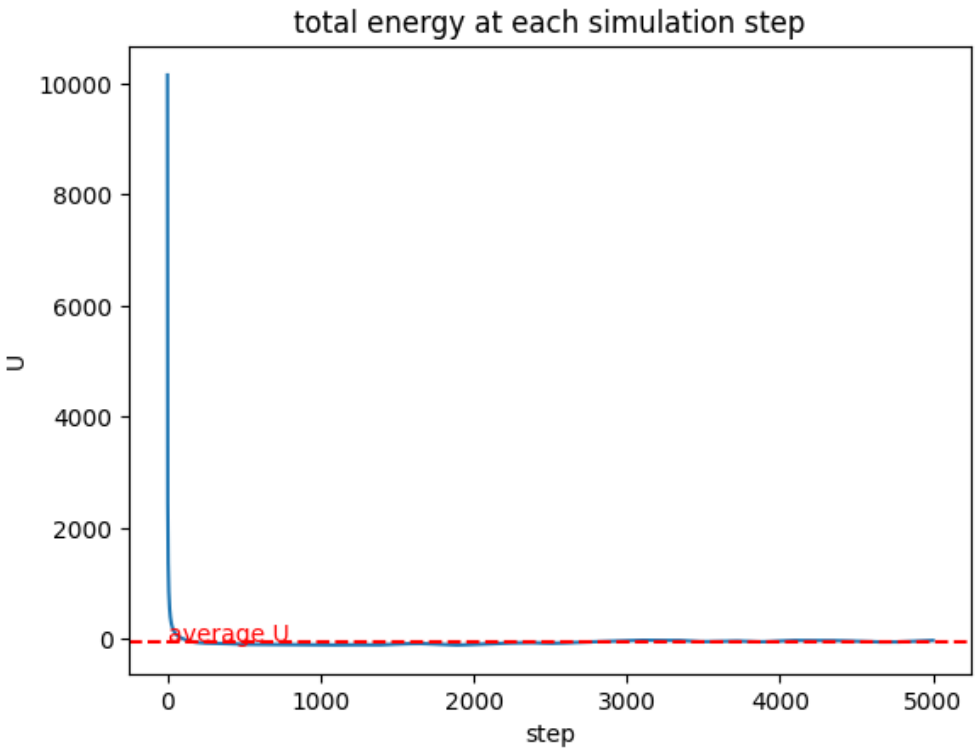

Figure 11: MD energy versus step 1

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

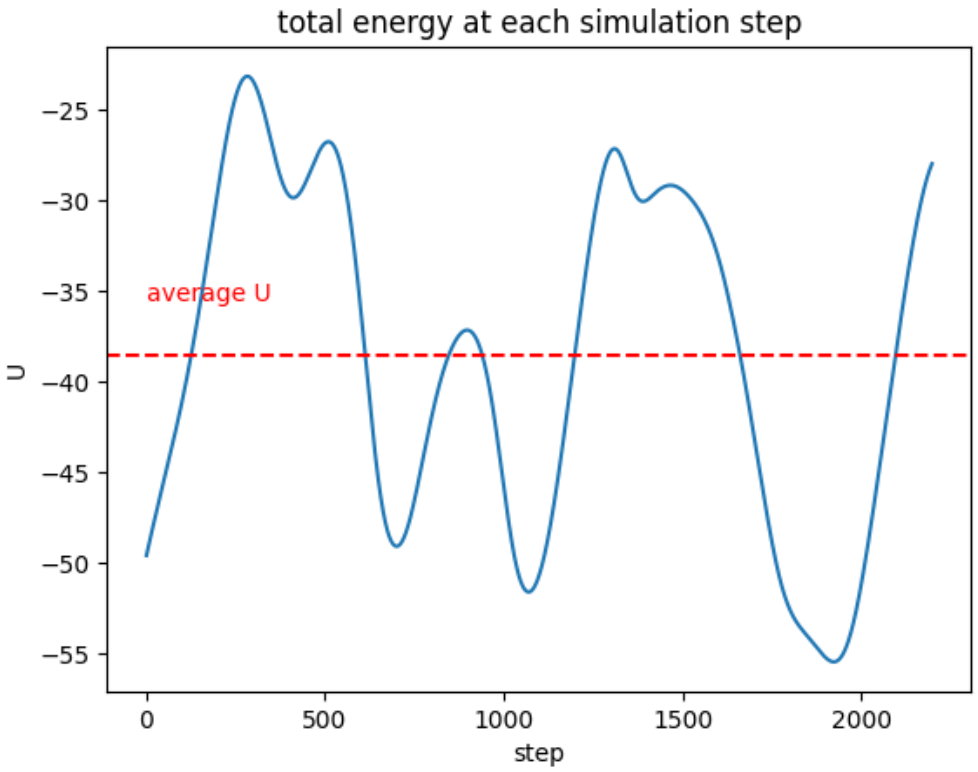

Figure 12: MD energy versus step 2

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 $\lambda = 3$, $\delta t = 0.014$, and most importantly, $T_0 = 0.001 * T_1$, and manage to find a better equilibrium. After 10000 simulation steps, I indeed get a prettier equilibrium. The energy-versus-time figure is

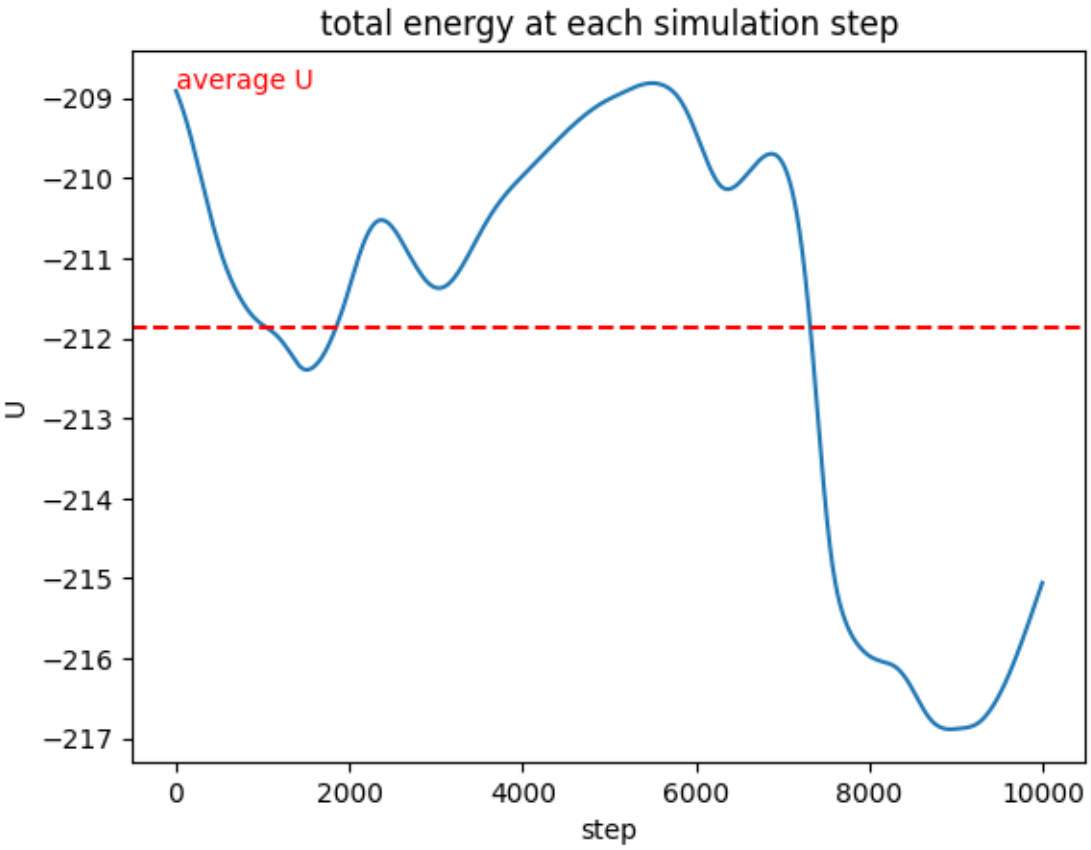

Figure 13: MD energy versus step 3

Calculating the statistical values,

$$\begin{aligned} \langle U \rangle &= -211.875 \\ \delta U^2 &= 7.163 \end{aligned} \tag{3.12}$$

both are absolutely larger than those in MC simulation. Nonetheless, the relative fluctuation is

$$\left| \frac{\delta U}{\langle U \rangle} \right| \approx 1.26\% \tag{3.13}$$

which is not far away from that in MC simulation (Eq. 3.4). Why do I get a lower equilibrium energy, larger energy fluctuation and comparable relative fluctuation to what I get in MC simulation? I cannot explain right now. Besides, I find my molecular dynamics simulation very sensitively depending on the initial temperature. Slightly changing the initial temperature will greatly affect the equilibrium energy. This poses serious concerns on the reliability of my MD algorithm. It is also tricky to determine the simulation time step. A large time step causes particles to come too close together, and the energy may explode; while a small time step makes the simulation much longer or even not converge.

After all, I can successfully reach long-time equilibrium, and this proves my algorithm right. There are so many parameters free for adjustment, and my choice is far from the best, so there is substantial space for improvement of efficiency and reliability.

For the second set of fixed conditions, $T_2 = 1.0, \rho_2 = 0.1$, I run the simulation for 5000 steps, with $\lambda = 3$, $\delta t = 0.015$, and initial temperature $T_0 = 0.01T_2$. The final configuration is

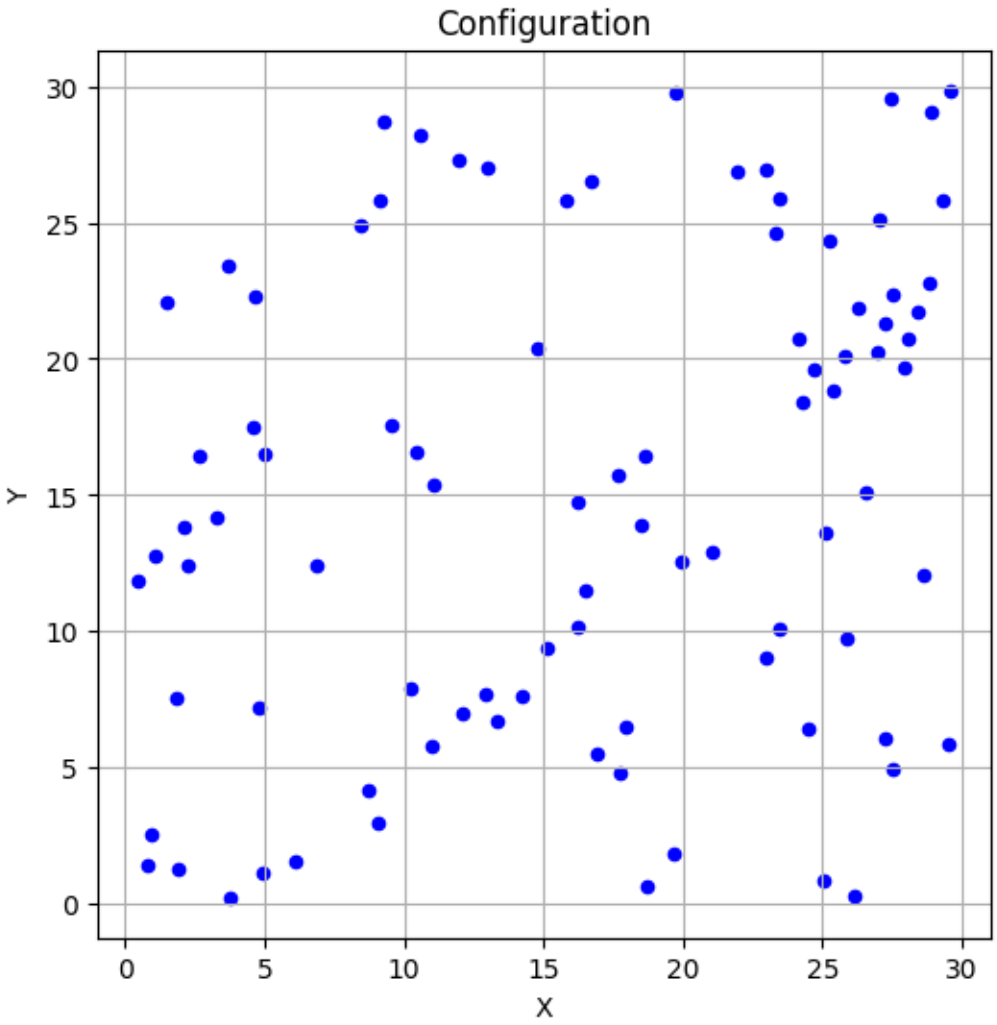

Figure 14: MD final configuration2

The energy-versus-time figure is

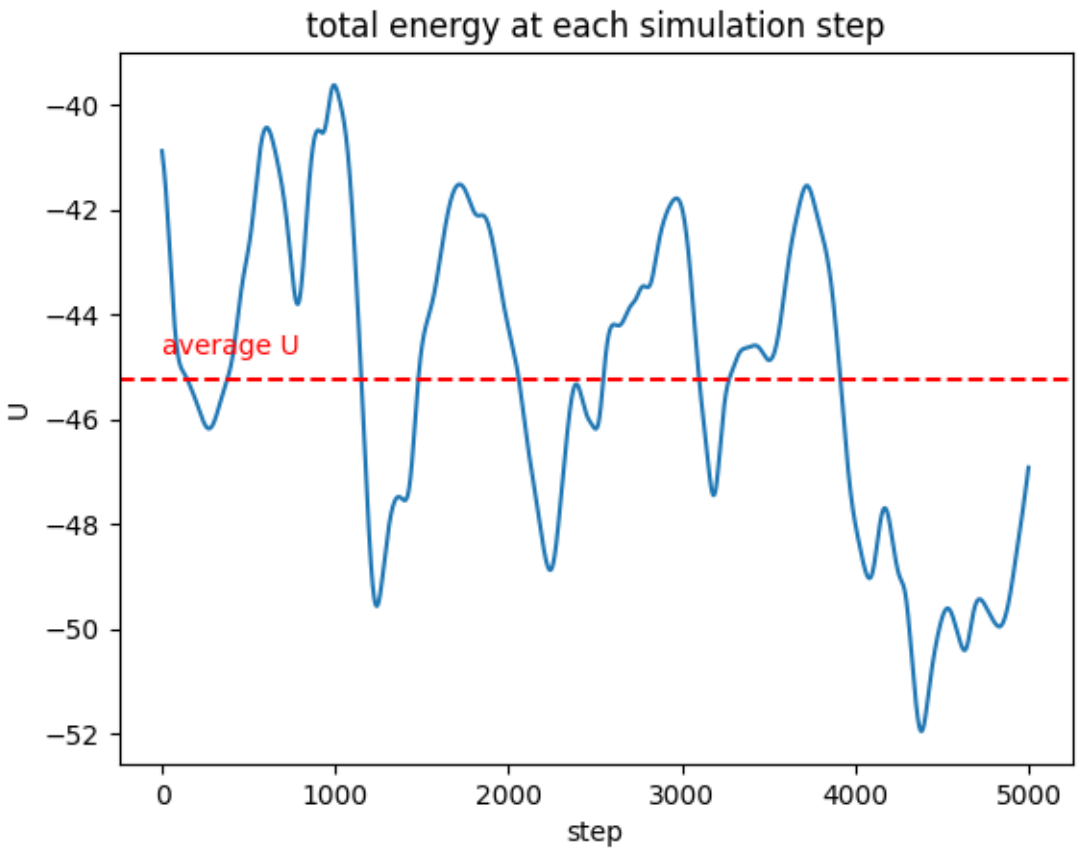

Figure 15: MD energy versus time 4

The equilibrium is reached instantly, and the statistical values are

$$\begin{aligned} \langle U \rangle &= -45.237 \\ \langle u \rangle &= \frac{\langle U \rangle}{N} = -0.503 \\ \delta U^2 &= 8.871 \\ \left| \frac{\delta U}{\langle U \rangle} \right| &\approx 6.58\% \end{aligned} \tag{3.14}$$

Still, average energy and energy fluctuation are absolutely larger than those in MC simulation, but relative energy fluctuation is smaller.

3.2.2 Ordered lattice

In section 3.2.1 we have seen that equilibrium in condition (T_1, ρ_1) exhibits lattice symmetries more, so in the present section, I focus on the simulation of ordered lattice in condition (T_1, ρ_1). The initial configuration looks like

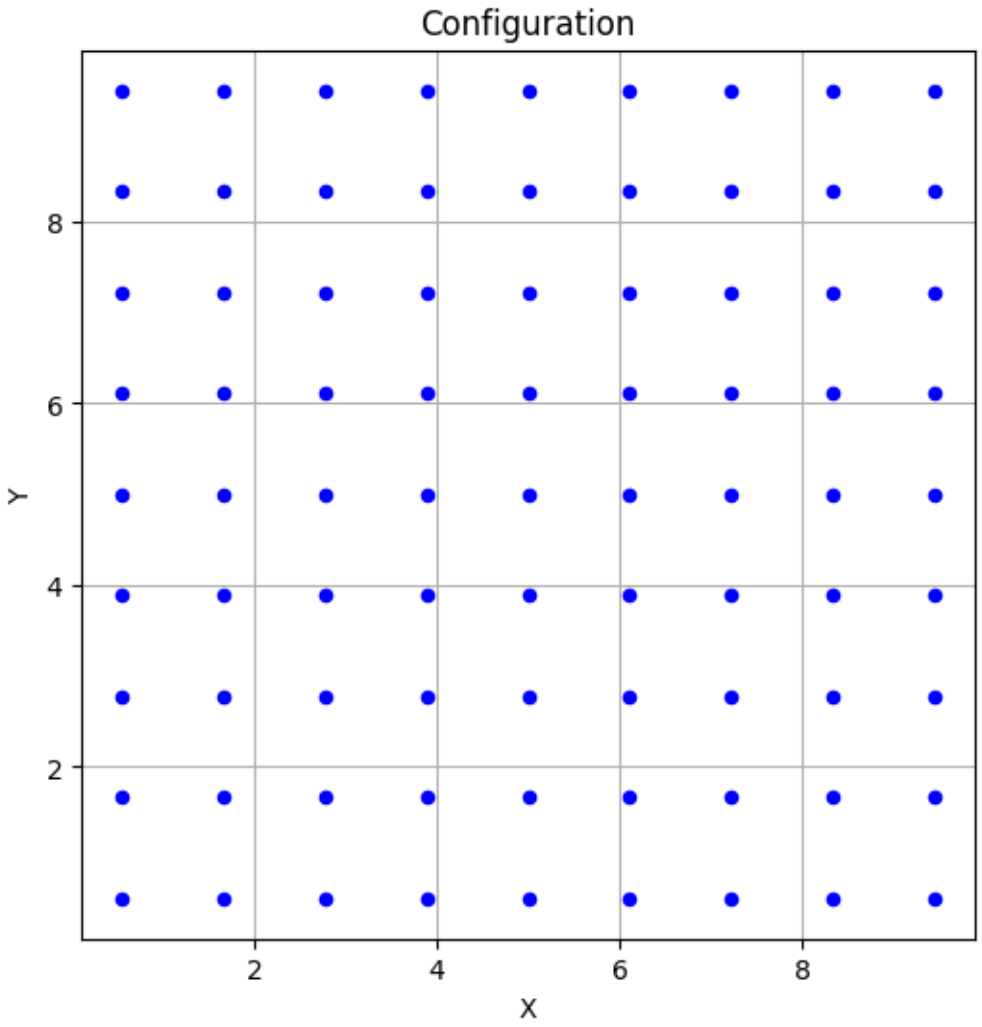

Figure 16: initial lattice configuration 1

(a) Monte Carlo simulation

I run the simulation for 20000 steps, with normally-distributed random displacement $\sigma = 0.5$, and got the final configuration

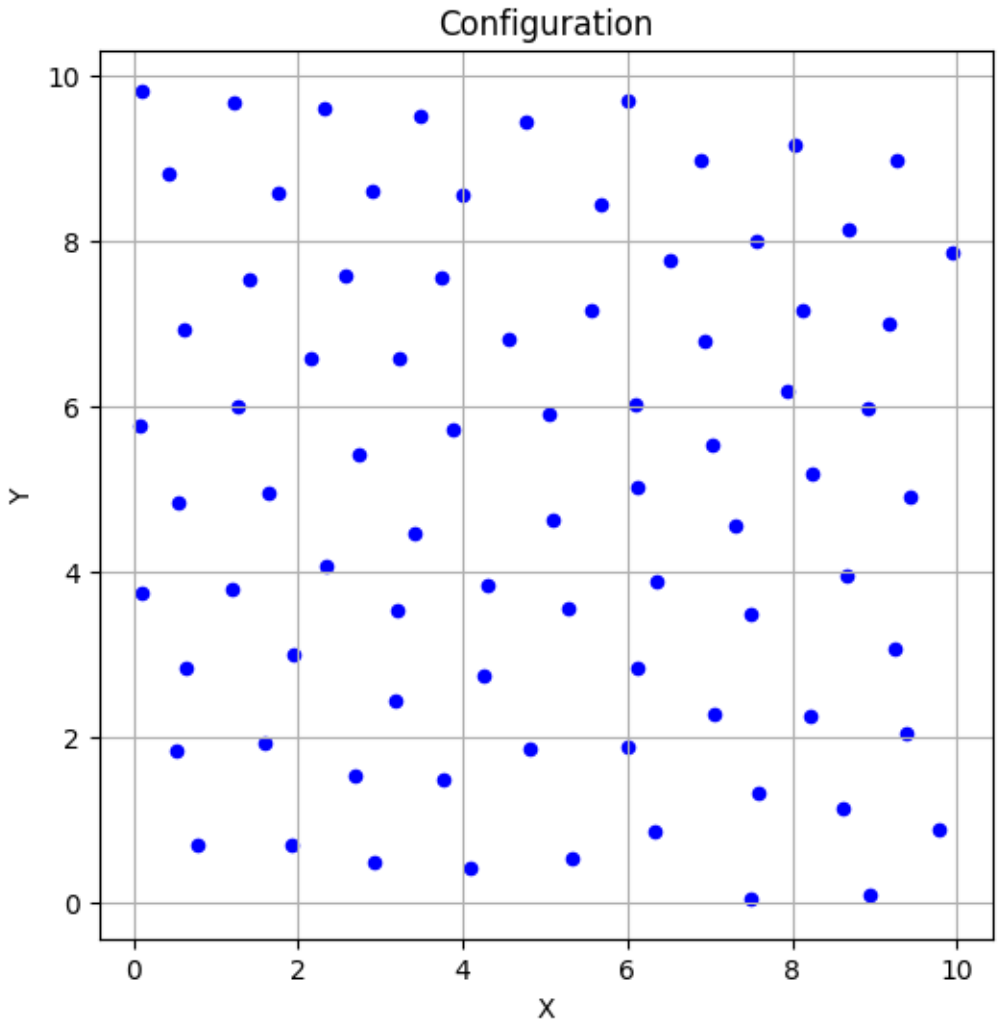

Figure 17: MC final lattice configuration 1

The lattice structure is significantly damaged. The energy-versus-time figure shows

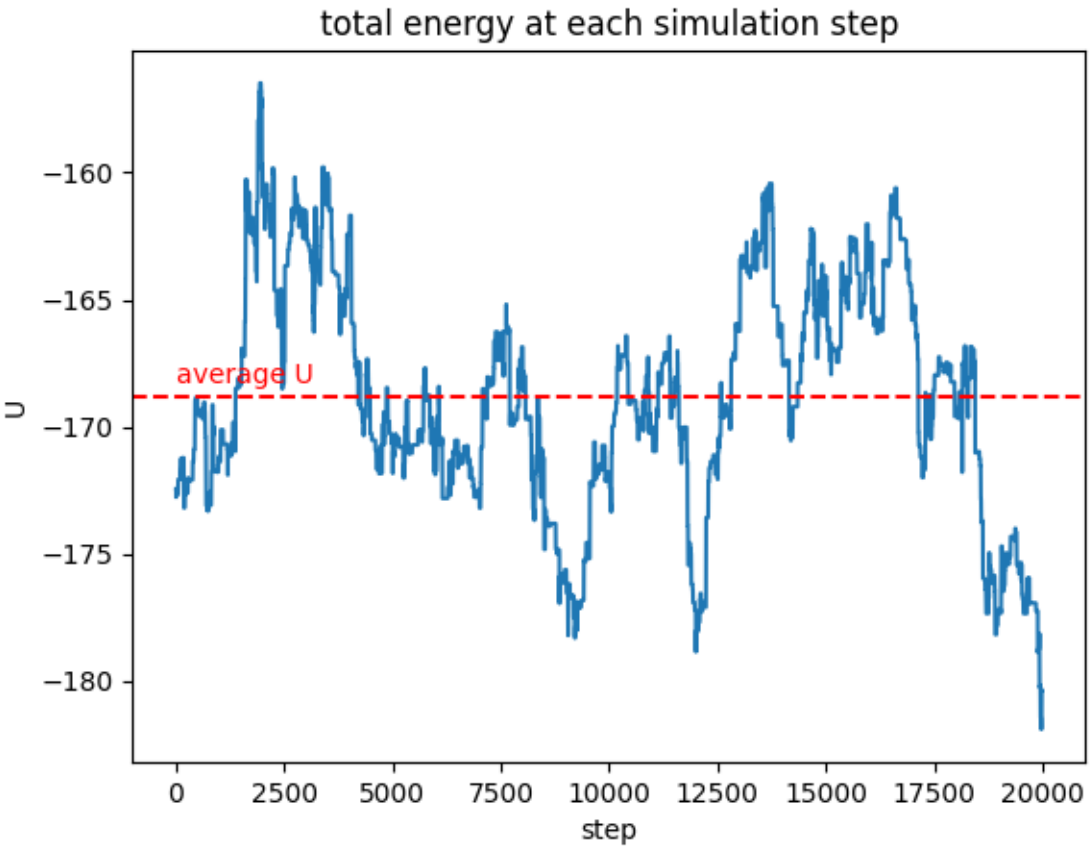

Figure 18: MC lattice energy versus time 1

The statistical values are

$$\begin{aligned} \langle U \rangle &= -168.792 \\ \langle u \rangle &= \frac{\langle U \rangle}{N} = -2.084 \\ \delta U^2 &= 20.126 \\ \left| \frac{\delta U}{\langle U \rangle} \right| &\approx 2.66\% \end{aligned} \tag{3.15}$$

which are all greater than those calculated in the random positions case. The reason may lie in the fact that the ordered lattice gains an extremely low energy at the beginning (perhaps close to the global minimum), and since the temperature is low, it is not likely for the system to jump out of this "potential well". In contrast, when I initialize the system with random positions, the energy is expected to be relatively high, and again since the temperature is low, the random displacements may bring the system to some local minima and it will never get out. The local minima and global minimum of the energy have relations unsolvable for me, and the fluctuation of energy might depend on the potential landscapes around the minima. It is hard to say how much my calculated statistical values represent real physics.

(b) molecular dynamics simulation

I run the simulation for 5000 steps, with time step $\delta t = 0.013$, $\lambda = 3$, and $T_0 = 0.1T_1$. The final configuration is

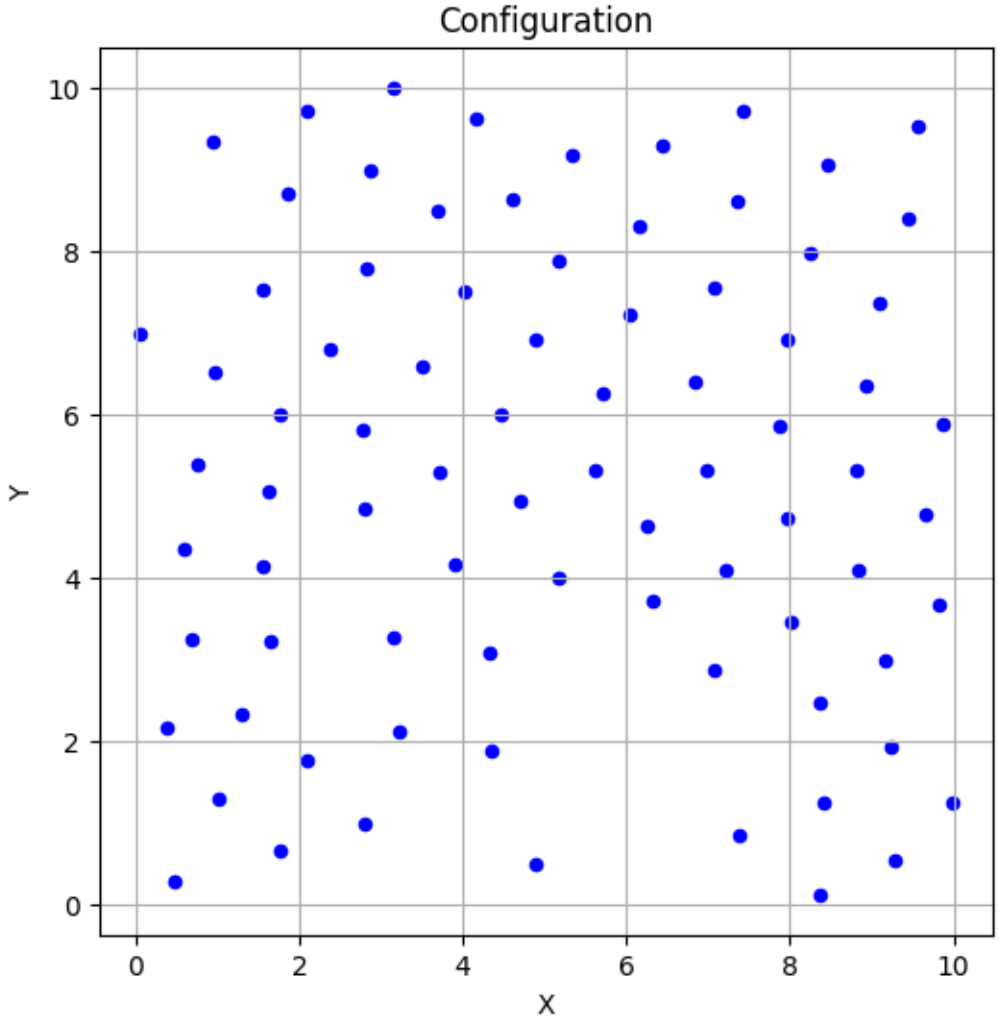

Figure 19: MD lattice final configuration 1

The statistical values are

$$\begin{aligned} \langle U \rangle &= -125.847 \\ \langle u \rangle &= \frac{\langle U \rangle}{N} = -1.554 \\ \delta U^2 &= 263.118 \\ \left| \frac{\delta U}{\langle U \rangle} \right| &\approx 12.9\% \end{aligned} \tag{3.16}$$

The relative energy fluctuation is of the same magnitude as that in random positions condition. We can see the lattice structure is crushed, and the low average energy may be also due to the low initial energy. The large fluctuation in energy may be caused by the higher initial temperature $T_0 = 0.1T_1$.

3.3 TASK3

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

I simulate for $N = 9, 16, 25, 36, 49, 64, 81, 100, 121, 225$, the plots are as follows

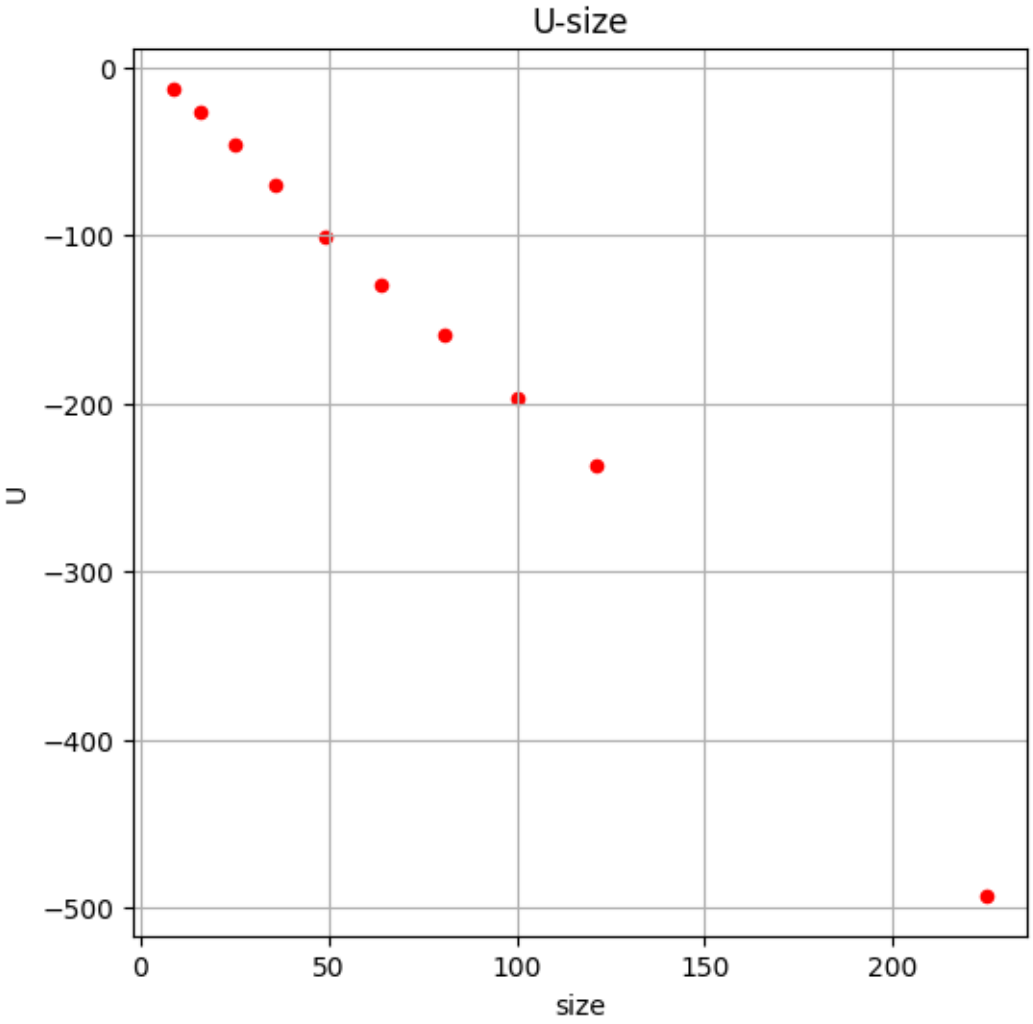

Figure 20: average energy versus N

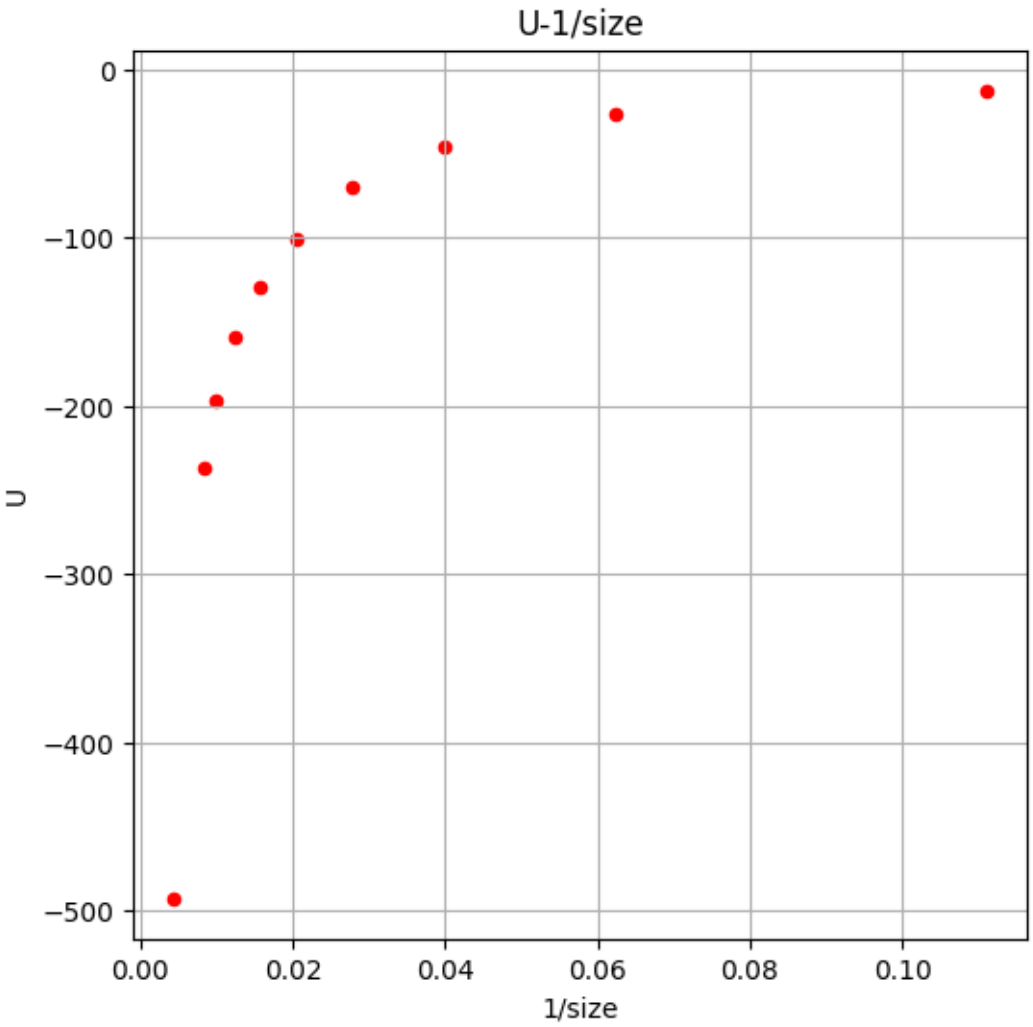

Figure 21: average energy versus 1/N

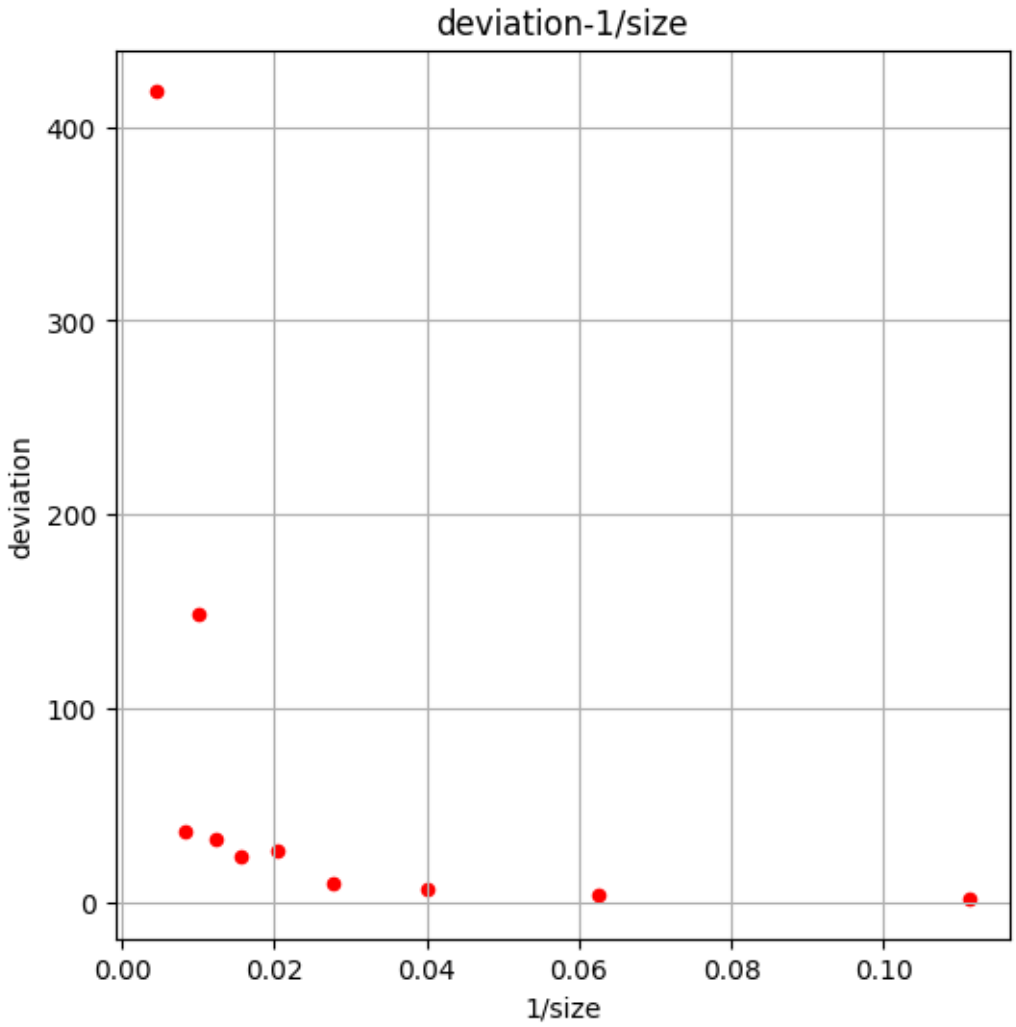

Figure 22: energy mean square deviation versus 1/N

Despite some instabilities, from the two figures we clearly notice $U(N \to \infty) = -\infty$, and $\delta U(N \to \infty) = \infty$. I did not use hard-boundary condition because it is not suitable for macroscopic simulations. Fig. 20 demonstrates apparent linear dependence of U on N, which hard-boundary condition cannot offer. In a hard box, as the proportion of boundary particles to all particles becomes smaller when the cluster becomes larger, the $U - N$ plot will eventually become steeper and more linear.

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

$$F_2(\boldsymbol{r}_1, \boldsymbol{r}_2) \equiv N(N-1) \int \cdots \int \rho_N(\boldsymbol{r}_1, \cdots, \boldsymbol{r}_N) \mathrm{d}^3\boldsymbol{r}_3 \cdots \mathrm{d}^3\boldsymbol{r}_N \tag{3.17}$$

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

$$F_2(\boldsymbol{r}_2 - \boldsymbol{r}_1) \equiv n^2 g(\boldsymbol{r}_2 - \boldsymbol{r}_1) \tag{3.18}$$

Integrating Eq. 3.17, we get [2]

$$\begin{aligned} N(N-1) &= \iint F_2(\boldsymbol{r}_2 - \boldsymbol{r}_1) \\ &= n^2 \iint g(\boldsymbol{r}_2 - \boldsymbol{r}_1) \mathrm{d}\boldsymbol{r}^3 \\ &\approx n^2 V \int g(\boldsymbol{r}) \mathrm{d}^3\boldsymbol{r} \end{aligned} \tag{3.19}$$

Just as in previous tasks, simulations with periodic boundary conditions result in final configurations like Fig. 17. The radial distribution function is closely related to the probability density of finding a particle somewhere relative to another particle. I select a particle and count the number $N(r)$ of other particles around it within the distance r. Hence I can plot a cumulative distribution function like this

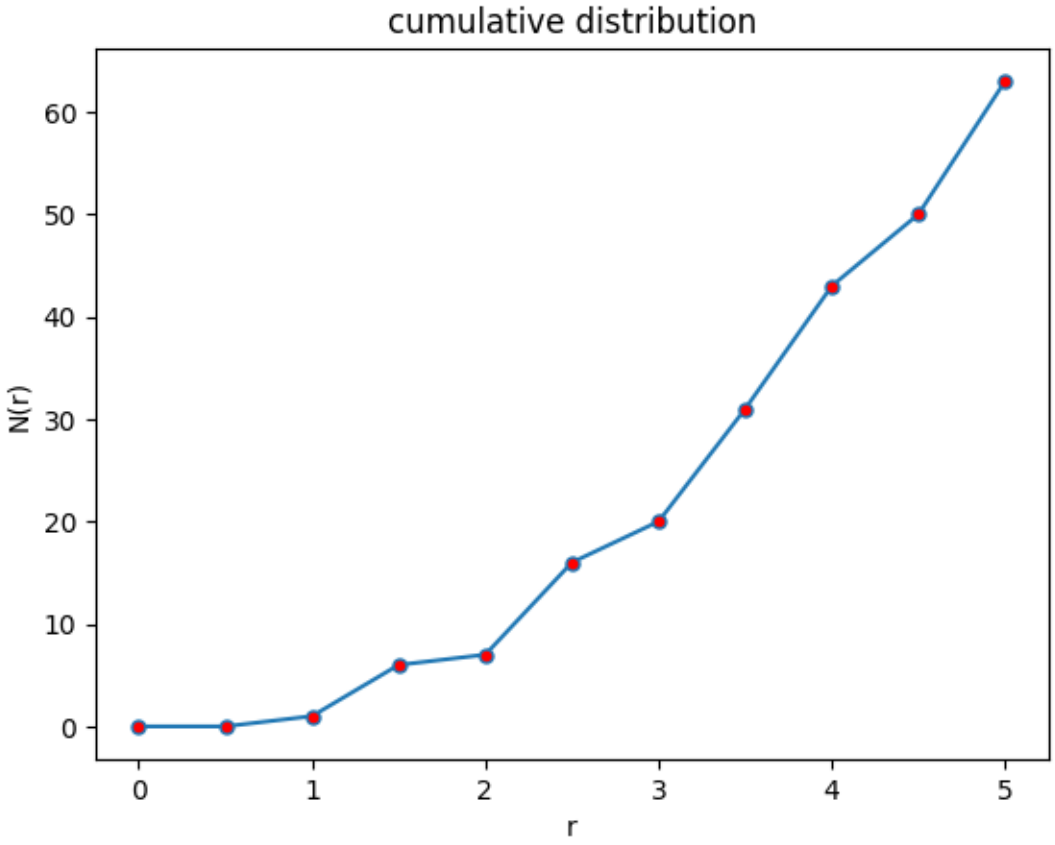

Figure 23: radial cumulative distribution 1

In consideration of minimum-image convention, r cannot be larger than half the diameter of the simulation box, which is 5 in this case. $N(r)$ can be fitted very well to a quadratic function. Here is the result

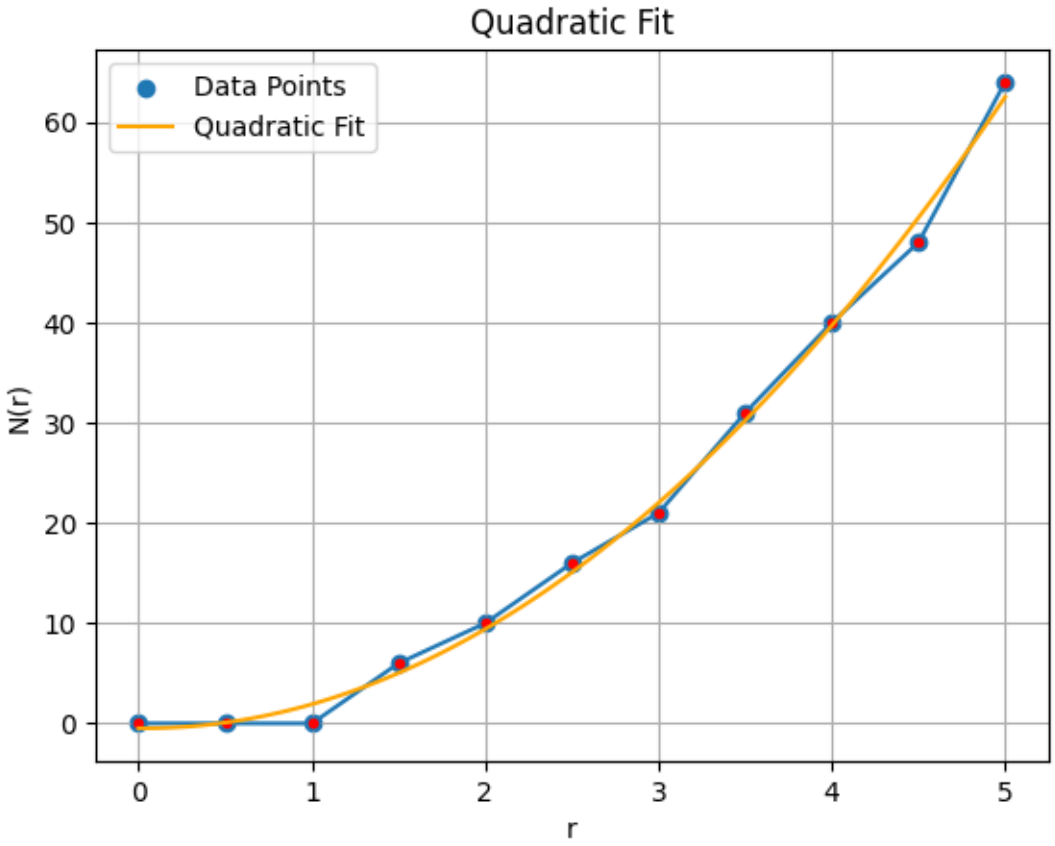

Figure 24: radial cumulative distribution fitted

The fitted quadratic function is

$$N = 2.550r^2 - 0.151r - 0.483 \tag{3.20}$$

with R-squared value $R^2 = 0.9966$. From the definition of pair distribution function $F_2(r)$, we know that if we select two particles 1 and 2, then the probability of the two particles are respectively in the infinitesimal boxes $\mathrm{d}^2\boldsymbol{r}_1$ and $\mathrm{d}^2\boldsymbol{r}_2$ is

$$\begin{aligned} \mathrm{d}^2\boldsymbol{r}_1\mathrm{d}^2\boldsymbol{r}_2 \int_3 \cdots \int_N \rho_N(\boldsymbol{r}_1, \cdots, \boldsymbol{r}_N)\mathrm{d}^2\boldsymbol{r}_3 \cdots \mathrm{d}^2\boldsymbol{r}_N &= \frac{F_2(\boldsymbol{r}_2 - \boldsymbol{r}_1)}{N(N-1)}\mathrm{d}^2\boldsymbol{r}_1\mathrm{d}^2\boldsymbol{r}_2 \\ &= \frac{n^2 g(\boldsymbol{r})}{N(N-1)}\mathrm{d}^2\boldsymbol{r}_1\mathrm{d}^2\boldsymbol{r} \end{aligned} \tag{3.21}$$

where $\boldsymbol{r} = \boldsymbol{r}_2 - \boldsymbol{r}_1$. In the second line of the above equation, I have transformed the coordinates from $(\boldsymbol{r}_1, \boldsymbol{r}_2)$ to $(\boldsymbol{r}_1, \boldsymbol{r})$. Let us focus on particle 1. The probability that particle

2 appears in the infinitesimal box $\mathrm{d}^2\boldsymbol{r}$ relative to particle 1 is

$$\int_1 \frac{n^2 g(\boldsymbol{r})}{N(N-1)} \mathrm{d}^2\boldsymbol{r}_1 \mathrm{d}^2\boldsymbol{r} = \frac{ng(\boldsymbol{r})}{(N-1)} \mathrm{d}^2\boldsymbol{r} \tag{3.22}$$

Therefore, the expected number of particles in the box $\mathrm{d}^2\boldsymbol{r}$ relative to particle 1 is

$$\frac{Nng(\boldsymbol{r})}{(N-1)} \mathrm{d}^2\boldsymbol{r} \tag{3.23}$$

and the expected number of particles that are in the range of distance $r \sim r + \mathrm{d}r$ from particle 1 is

$$\frac{2\pi Nng(r)}{(N-1)r} \mathrm{d}r \tag{3.24}$$

which yields the relation between the cumulative distribution and radial distribution function

$$\frac{\mathrm{d}N(r)}{\mathrm{d}r} = \frac{2\pi Nng(r)r}{(N-1)} \tag{3.25}$$

Differentiating the fitted quadratic function $N(r)$, I estimate $g(r)$ and plot as follows

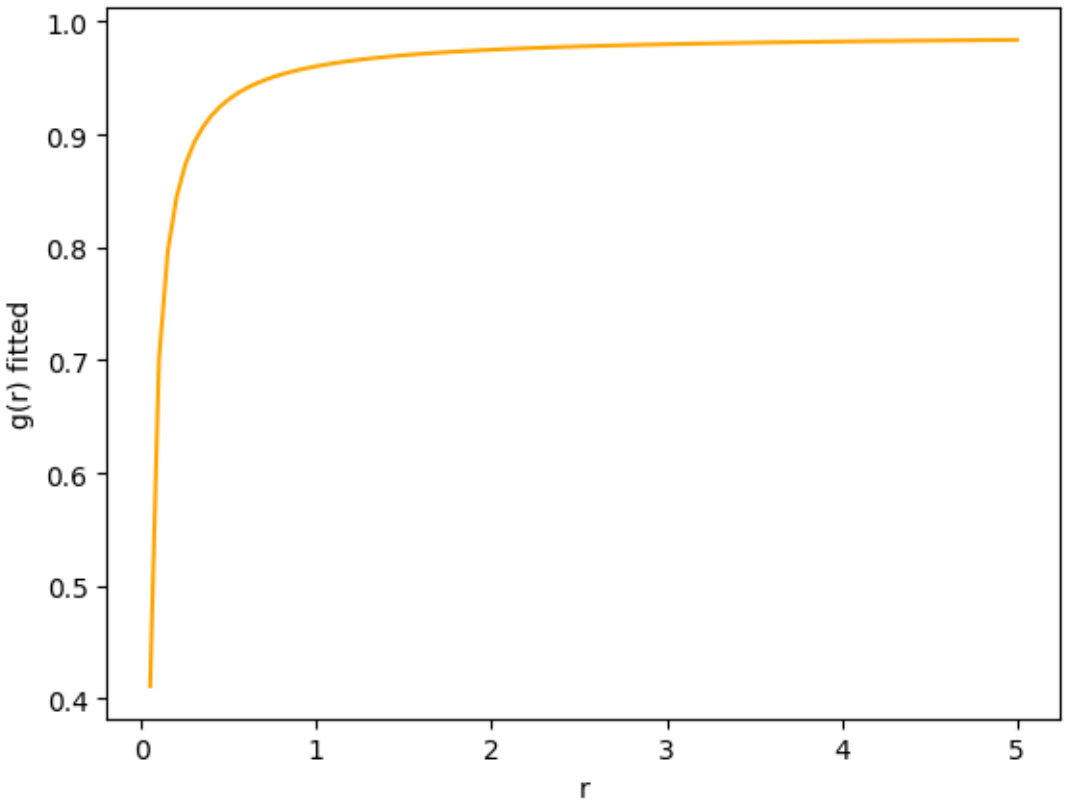

Figure 25: $g(r)$ fitted

For r close to zero, the vertical intercept of $\mathrm{d}N/\mathrm{d}r$ divided by r tends to infinity. But this contributes little to the integrations involving $g(r)$, since $g(r)$ is multiplied by the close-to-zero r, resulting in the same magnitude as the small vertical intercept. When r is large, the estimated $g(r)$ is 0.984. Using equation (4.5.1) in textbook [1],

$$\frac{U}{N} = \pi\rho \int_0^\infty ru(r)g(r)\mathrm{d}r \tag{3.26}$$

I estimate the average energy per particle

$$\frac{U}{N} = \pi\rho_1 \int_0^{r_c} r\phi^A(r)g(r)\mathrm{d}r \tag{3.27}$$

Sadly, we cannot take $g(r)$ as a constant 0.984, because the integrand $r\phi_A(r)$ explodes at $r \to 0$. The aforementioned estimation only works for large r, where particles are seen to be evenly distributed. For small r, I guess I need a Boltzmann factor to suppress the integrand. Thus I get

$$\frac{U}{N} = \pi\rho_1 \int_0^{r_c} r\phi^A(r)g(r)e^{-\phi_A(r)/T}\mathrm{d}r \approx -2.674 \tag{3.28}$$

which agrees in terms of order of magnitude with the result of the simulation (Eq. 3.15). The success of adding a Boltzmann factor implies that the real radial distribution function $g(r)$ may be the estimated $g(r)$ times a Boltzmann factor. It looks like

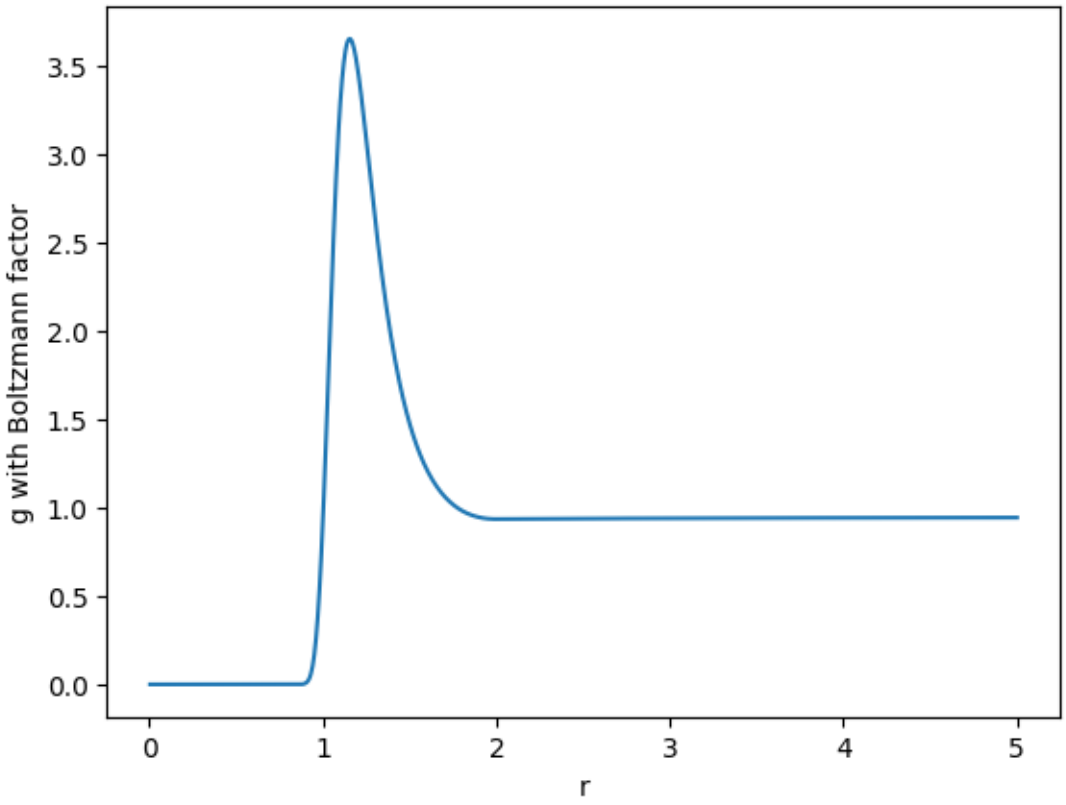

Figure 26: $g(r)$ fitted times $e^{-\phi(r)/T}$

Intuitively, the Boltzmann factor *should* appear in the expression of $g(r)$ at thermal equilibrium, but I am still not sure why adding it can make the result reasonable. Perhaps there is a better way than counting particles within distance r to estimate the distribution function $g(r)$.

3.5 TASK5

In this task, I evaluate the pressure and study the equation of state of the system. Virial theorem states that the average kinetic energy of the system has this relation with the forces and configuration

$$\langle 2K \rangle = -\left\langle \sum_{i=1}^{N} \boldsymbol{F}_i \cdot \boldsymbol{r}_i \right\rangle \tag{3.29}$$

The right-hand-side can be separated to two terms—one contains pressure, the other contains pair-wise interactions. In our 2D model, $\langle K \rangle = N k_B T$, and $-\sum_{i=1}^{N} \boldsymbol{F}_i \cdot \boldsymbol{r}_i = p \oint \hat{n} \cdot \boldsymbol{r} \mathrm{d}l + \sum_{i=1}^{N} \sum_{j \neq i} \frac{\partial v_{ij}}{\partial \boldsymbol{r}_i} \cdot \boldsymbol{r}_i$. Using Gaussian divergence theorem,

$$p \oint \hat{n} \cdot \boldsymbol{r} \mathrm{d}l = p \iint_S \nabla \cdot \boldsymbol{r} \mathrm{d}x \mathrm{d}y = 2pS \tag{3.30}$$

and Virial theorem becomes

$$\begin{aligned} 2N k_B T &= 2pS - \sum_{i=1}^{N} \sum_{j \neq i} \boldsymbol{F}_{ji} \cdot \boldsymbol{r}_i \\ &= 2pS - \sum_{i=1}^{N} \sum_{j > i} \boldsymbol{F}_{ij} \cdot \boldsymbol{r}_{ij} \\ &= 2pS + \sum_{i=1}^{N} \sum_{j > i} \frac{\mathrm{d}v(r_{ij})}{\mathrm{d}r_{ij}} r_{ij} \end{aligned} \tag{3.31}$$

from which we can basically evaluate the pressure p. The textbook [1] provides two kinds of correction for the potential energy and pressure: tail correction and impulsive correction. The former sums up all contributions of particles beyond the truncation distance r_c, which we ignored when performing the simulation; while the latter takes into account the discontinuity in the potential caused by the truncation and arrives at an extra pressure ΔP^{imp}. However, the atomic potential ϕ_A we use is well-behaved at all points including r_c, i.e. both its value and its derivative are continuous. I do not think it is necessary to consider tail or impulsive correction. Anyway, I calculate the tail correction just to compare its magnitude with the Virial result.

From Eq. 3.31, we explicitly derive the equation of state

$$p = \rho T - \sum_{i=1}^{N} \sum_{j > i} \frac{1}{2S} \frac{\mathrm{d}v(r_{ij})}{\mathrm{d}r_{ij}} r_{ij} \tag{3.32}$$

Changing ρ and running the simulations, I get the phase diagram of $T_3 = 0.9$

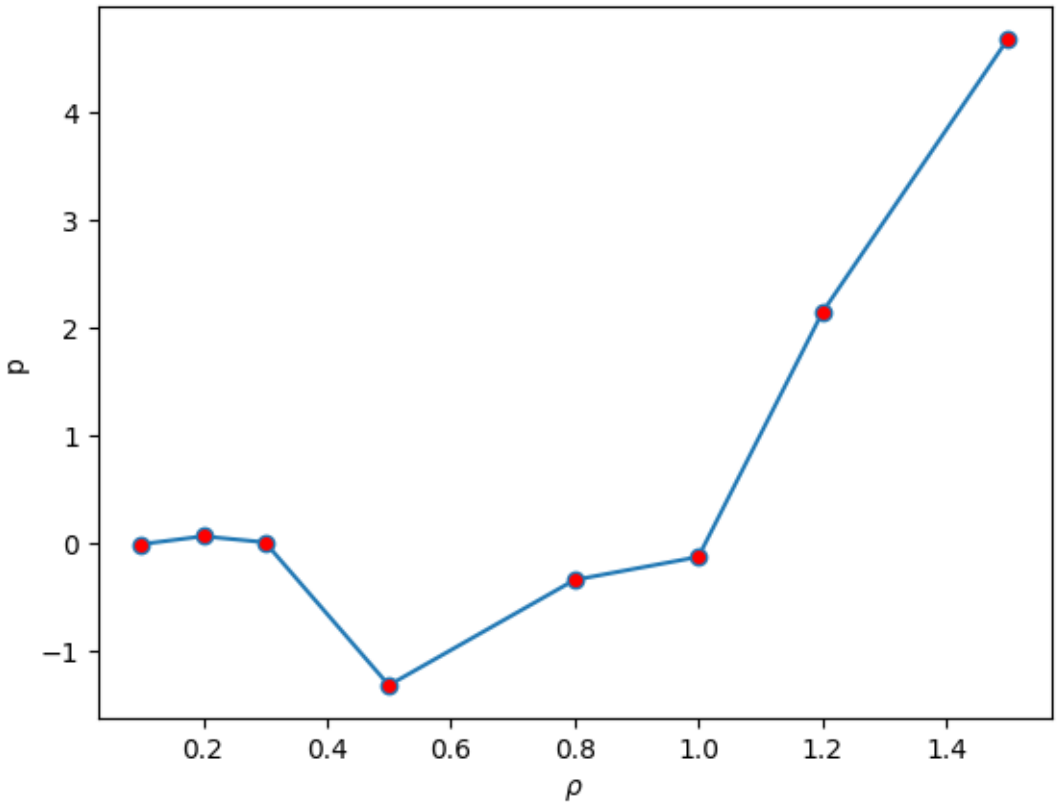

Figure 27: phase diagram at T_3

Comparing this phase diagram to Figure 3.5 in textbook [1], we see T_3 is below the critical temperature. There are some errors in the calculation of p. From simulations above we know that the system is most stable when ρ is around ρ_1, where p will be smallest. But in this diagram p reaches minimum elsewhere.

I also get the phase diagram for $T_4 = 2.0$

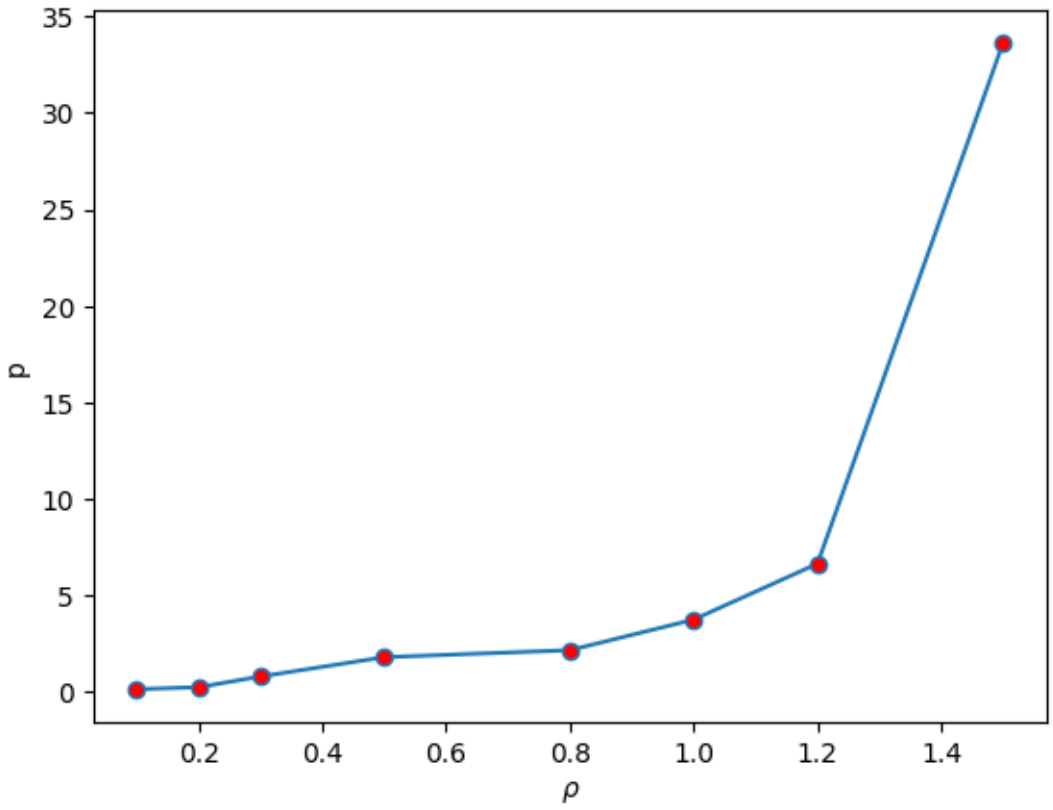

Figure 28: phase diagram at T_4

Here the pressure never goes down below zero. I suspect that the system undergoes a phase transition from $T = T_3$ to $T = T_4$. At $T = T_3$, there are some forces that bring the particles together, forming something like solid; while at $T = T_4$, there is no such force, and the system keeps expanding to something like liquid or gas.

Finally, the tail correction of pressure (for Lennard-Jones potential):

$$\Delta P^{\text{tail}} = \pi\rho^2 \int_{r_c}^{\infty} r^2 f(r) \mathrm{d}r = \frac{8\pi\rho^2}{3}\left(\frac{3}{5r_c^{10}} - \frac{3}{4r_c^4}\right) \tag{3.33}$$

Plugging in the data, $r_c = 2,\ \rho = \rho_1$, then

$$\Delta P^{\text{tail}} = -0.276 \tag{3.34}$$

Not a lot, but also not negligible.

4 Results and Conclusions

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: $T_1,\ \rho_1$ and $T_2,\ \rho_2$, 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.

TASK 3 is to look at the affect of system size N on the result. I find periodic boundary condition works well with varying sizes.

In TASK 4, I struggle to estimate the radial distribution function, and my estimation hardly agrees with the theory and examples provided in the textbook. I think my main obstacle is that I do not have a good approach to calculating $g(r)$. My minimum resolution distance is the interatomic distance, which is too large and inaccurate.

In TASK 5, I calculate the pressure and obtain phase diagrams. Problems in other tasks remain in this task, such as inconsistency among repeated simulations, sensitive dependence on initial conditions, discrepancy with theoretical predictions, etc. At least, I find that increasing the temperature over the critical temperature causes phase transition.

5 Future Work

Since I had limited time to work on this project, there must be many imprecisions, errors and oversights in this project. Apart from knowing that the particle's displacement at each step must be significantly smaller than the interatomic distance, I am still unclear about how to choose the random displacement vector for Monte Carlo simulations, the initial temperature for molecular dynamics simulations, the strength of thermal coupling, and the time step. There are also numerous physical models, equation-solving algorithms, particle configurations, potential functions, media, phases, etc., most of which I have never heard of or have little knowledge.

In future research and studies, I may have the opportunity to explore more diverse and comprehensive simulation methods for investigating materials with different scales and structural properties. I may also enhance my algorithm and programming skills, and have access to greater computational resources to tackle more extensive computational physics problems. At that time, many of these current challenges will be resolved, but new ones will undoubtedly arise. The exploration of computational physics simulations is an endless journey.

References

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

[2] Zonghan Lin. *Thermodynamics and Statistical Physics*. Peking University Press, 2018.

[3] 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.

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

[5] 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.

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