Report for PHY905 Project 4, Molecular dynamics

Jialin Liu
Thanaphong Phongpreecha

May 4, 2016

Abstract

Molecular dynamics (MD) is the use of classical equation of motion to study structural and properties evolution of the interacting system for a period of time. Although it seems the application of MD is virtually unlimited, users should take caution in results interpretation as its underlying equations overlook many features.In this study, a Python code was employed to study (MD) behavior at different temperatures and melting temperature of argon. Within Python code, object-orienting code style is used. Positions and velocities were updated by Velocity Verlet algorithm. Periodic boundary condition was applied to enable study of a large system, which is 3 x3x3 super cells in this study (accounting to 108 atoms). Since argon is inert and virtually contains no partial charge, this simplifies the force field to just only van der Waals, which was calculated from Lennard-Jones potential. The melting temperatures were determined by exhibition of drastic change in diffusion coefficient. Using the initial FCC structure of lattice constant 5.260 Å, this study concludes that the melting temperature lies in the range of 36-60 K (as opposed to 80 K). Lowering the density of the lattice by changing the constant to 5.405 Åincreases the accuracy of the predicted melting temperature to between 61-91 K instead.

1 Introduction

Introduction to molecular dynamics

Molecular dynamics (MD) is a study of structure and microscopic dynamics of the molecules/atoms according to N-body interactions.Unlike density functional theorem (DFT), which solve the molecular problems through Kohn-Sham equations, MD utilizes classical equation of motion by solving for total force acting on that molecule then update position, velocity, and acceleration through accordingly. This renders the calculations to be orders of magnitude faster and allows users to study larger system, however at an expense of accuracy. By solving the equation step-by-step according to specified time frame, MD yields plethora of useful insight including, molecules' trajectory, potential/kinetic energy. Analyses of these term can further enhance the understanding of particular system through derived parameters, such as radius of gyration, mean-square displacement, radial distribution function, stress tensor, and many more. In this way MD simulations can help us to gain new insight into important processes taking place at the atomic and molecular level, an insight which is often impossible to obtain purely from experiments, as these rarely have sufficient resolution. These include myriad types of problems such as material phase transition [Kremer and Grest, 1990], and thin-film growth [Zu Heringdorf et al., 2001] in physics, or x-ray [Rose-Petruck et al., 1999] or NMR applications [Palmer III, 2001], and modeling reactions [Bergsma et al., 1987] in chemistry, or solvation energy of proteins and nucleic acids [Levitt et al., 1995], drug design [Åqvist et al., 1994],and protein folding [Levitt, 1983] in biology. Of course, MD is no magic wand, apart from inferior accuracy as compared to *ab initio* calculation, due to how its underlying construction, its limitations include negligence of entropic optimization (since it optimizes only forces from energy), exclusion of hydrogen bonds, omission of possible reactions. Although the latter two have been somewhat tackled by including H-bonds into coulombic interaction, and invention of reactive force fields (ReaxFF) [Van Duin et al., 2001], these are crude approximation because as H-bond is of quantum mechanical nature, and ReaxFF requires prior assignment of reactions, which therefore imposes constraints and limits discovery of other unknown possible reactions.

Molecular dynamics allow us to explore the properties of macroscopic level through the view of microscopic simulation. Macroscopic state is defined by number of particles (N), volume (V), temperature (T), pressure (P), energy (E), and chemical potential (μ). Therefore, MD can be performed in several types of ensemble depending on the answer users seeking. For example, NVT ensemble, where N, V, and T are kept constant by different types of thermostats.

Force field

Force field is arguably the most important factors in success of MD. Appropriate force field is required to obtain accurate results. Force field is essentially a summation of all forces considered. Empirical force fields were developed to suit different environment, such as CHARMM and AMBER. These force field requires parametrization in order to obtain the force field constants, which can be done by comparing results with simpler model in DFT calculations. For metals, where electrons are not localized, specific types of force field were developed such as embeded atomic potential.

Present study

In this study, we aim to develop a MD for a simple system of argon atoms.

Since there is no bonds or polarity, the system can be safely described by only van der Waal forces obtained from standard 6-12 Lennard-Jones potential. Herein, we build MD code with NVE dynamic in python from scratch. We run our code on two Argon system with 108 atoms with different cell parameters. The cell length for large cell is 5.405 A and smaller cell 5.260 A. From the dynamics with different initial temperature, the trajectory and diffusion coefficient are calculated to estimated the melting point of Argon.

2 Methods

Code structure

The outline of how our algorithms are connected can be seen from Fig. 1, where the corresponding functions used in Python are printed in red.

Object orientation

The Python code written in this project is separated into 2 classes, Atom and MDSystem.

Atom contains all the properties of argon including lattice parameters as shown in the source code below. Initializing Atom requires specification whether the system will be FCC lattice (mode 2) or randomized argon position (mode 1) as shown below. All of the information in Atom class will be processed and written out from the class MDSystem through main().

```
class Atom:
    def __init__(self, numberOfAtom, cellLength, mass, element, latConst, mode):
        # unit: cell length: A, mass: g/mol
        self.cellLength = cellLength
        if mode == 2:
            self.latConst = latConst
            Atom.createFCC(self, element, cMass(mass,1))
        if mode == 1:
            Atom.createRand(self, numberOfAtom, element, cMass(mass, 1))
```

Periodic boundary condition

In Molecular dynamics, the size of simulation cell is always limited. To get rid of boundary effects and mimic the properties of bulk material, periodic boundary condition(PBC) is implemented in system. We assume our simulation cell is surrounded by 26 of its identical image simulation cell (in 3D) in three directions. When an atom move out of the current simulation cell, its image atom from the other simulation cell will move in. It can be implemented using the following equation:

$$x = x - floor(x/a) \cdot a,$$

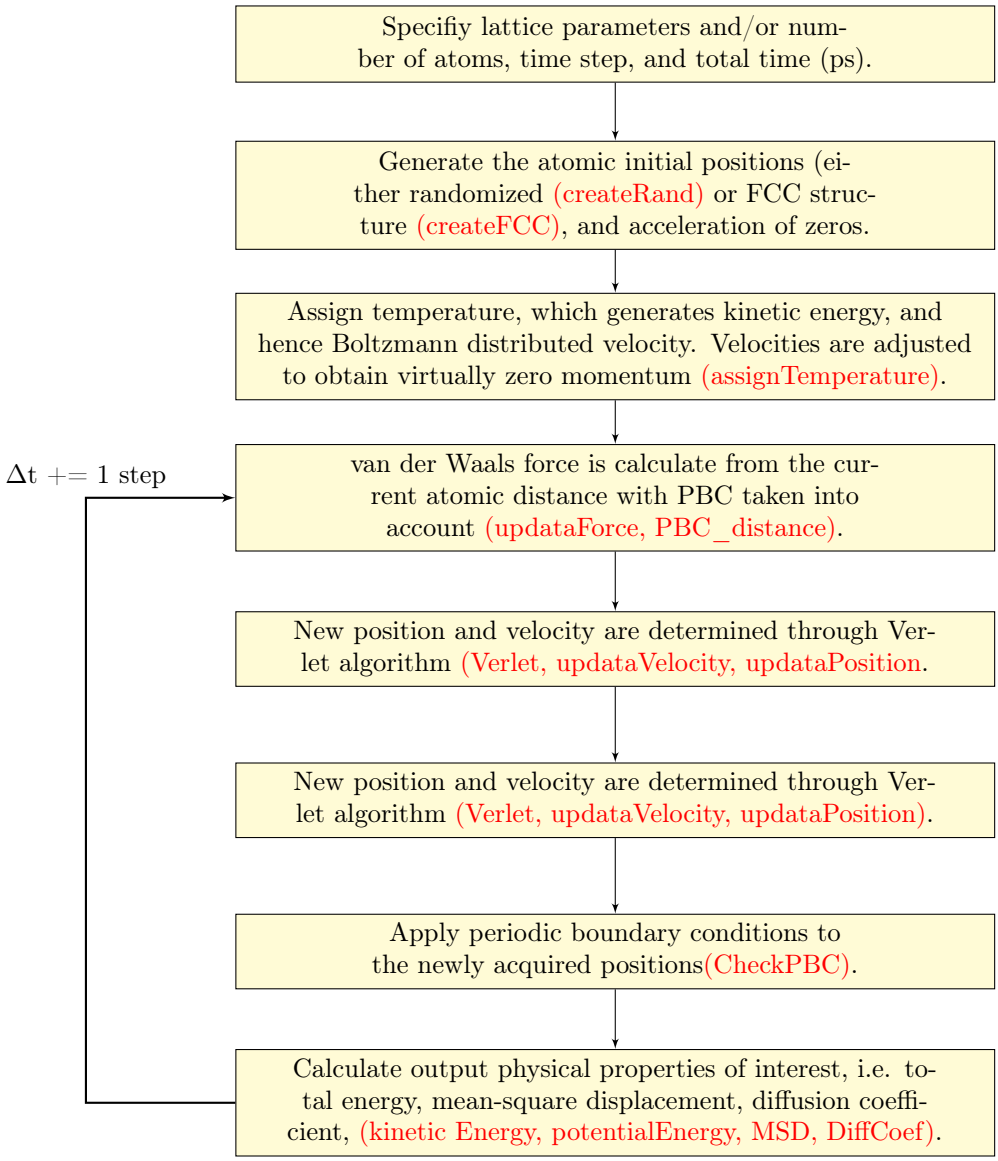

Figure 1: Flowchart of MD simulation in this present study. The corresponding functions used in Python code is shown in red

.

where x is the position vector of an atom, a is the cell length and floor is round towards negative infinity.

Because of PBC, the smallest distance between two atoms may not be the distance in same cell. The distance need to be corrected use the following equation:

$$|d| = |d - floor(2x/a) \cdot a|,$$

where d is the distance vector of two atoms in same simulation cell.

The corresponding codes are implemented in MDSystem.PBCdistance() and MDSystem.checkPBC() function.

```
def PBC_distance(self, x1, x2):
    # Read cellLength from self.cellLength
```

```
        cellLength = self.structure.cellLength
        distance = abs(x1 - x2)
        correctD = np.zeros((3))
        for i in [0, 1, 2]:
            correctD[i] = math.floor(2*distance[i]/cellLength[i])*cellLength[i]
        distance = distance - correctD
        return lg.norm(distance)
    def checkPBC(self):
        cellLength = self.structure.cellLength
        for i in xrange(0, self.structure.totNumberOfAtom, 1):
            currentPosition = self.structure.position[i, :]
            for j in [0, 1, 2]:
                self.structure.position[i, j] -= math.floor(currentPosition[j]/cellLength[j])*cellLength[j]
```

Temperature assignment and corresponding initial velocities

Temperature determines the initial velocity. In order for momentum to be conserved, the initial speed must be equal to zero ($\mu = 0$). The velocity as affected by temperature should be in Boltzmann distribution, which is conceived as normally distributed velocity with a variance of

$$\sigma^2 = \frac{k_B T}{m_i}$$

The corresponding code to generate such velocity given a temperature is therefore,

```
    def assignTemperture(self, t):
        # print 'Assign temperature'
        t = cTemperature(t, 1)
        kEcurrent = 0
        kb = 1  # eV/K
        sigma = (kb * t / self.mass[0]) ** 0.5
        mu = 0
        totMem = 0
        for i in xrange(0, self.totNumberOfAtom, 1):
            nv = 0
            self.velocity[i, :] = np.random.normal(mu, sigma, 3)
            totMem += sum(self.velocity[i, :])
        print totMem
        # Remove momentum
        difference = totMem / (3 * self.totNumberOfAtom)
        for i in xrange(0, self.totNumberOfAtom, 1):
            self.velocity[i, :] = self.velocity[i, :] - [difference, difference, difference]
```

FCC structure construction

FCC structure is constructed by collapsing 12 different partial atomic positions of a single unit cell into equivalent 4 full atomic positions (when periodic boundary condition is applied) as shown in Fig. 2 . The single unit cell is then replicated to achieve specified cell length (super cell) according to its cell parameter, which is 5.620 Åfor argon. The function createFCC details how this unit cell is expanded to the designated super cell.

```
    def createFCC(self, element, mass):
        #CellLength adjustment
        self.supercell = np.divide(self.cellLength, self.latConst)
```

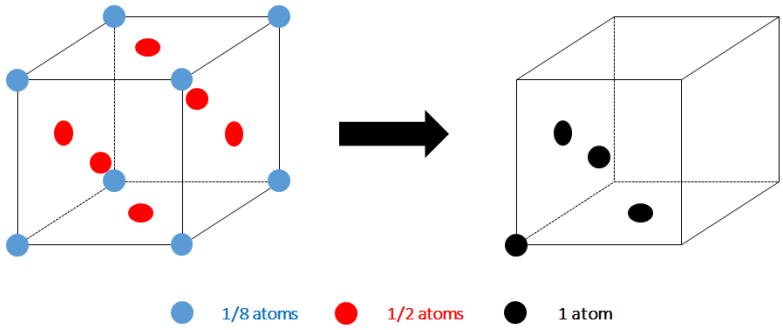

Figure 2: Construction of FCC lattice by converting partial atomic positions to full atomic positions.

```
self.supercell = self.supercell.astype(int)
self.cellLength = self.supercell * self.latConst
#Creating supercell
FCCPosition = np.array([[0, 0, 0], [0, 0.5, 0.5], [0.5, 0.5, 0], [0.5, 0, 0.5]]) * self.latConst
idx = 0
self.totNumberOfAtom = reduce(mul, self.supercell)*4
for i in xrange(0, self.supercell[0], 1):
    for j in xrange(0, self.supercell[1], 1):
        for k in xrange(0, self.supercell[2], 1):
            FCCPosition[:, 0] += i * self.latConst
            FCCPosition[:, 1] += j * self.latConst
            FCCPosition[:, 2] += k * self.latConst
            Atom.addAtom(self, idx+1, FCCPosition[0], [0, 0, 0], mass, element)
            Atom.addAtom(self, idx+2, FCCPosition[1], [0, 0, 0], mass, element)
            Atom.addAtom(self, idx+3, FCCPosition[2], [0, 0, 0], mass, element)
            Atom.addAtom(self, idx+4, FCCPosition[3], [0, 0, 0], mass, element)
            idx += 4
            FCCPosition = np.array([[0, 0, 0], [0, 0.5, 0.5], [0.5, 0.5, 0], [0.5, 0, 0.5]]) * self.latConst
```

Force field implementation

Force field is the function that describe the interactions between atoms. The Lennaard - Jones (LJ) potential is implemented in the code using MDSystem.force function. The LJ potential is expressed with following equation:

$$U(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right], \tag{1}$$

where $r_{ij} = |\vec{r}_i - \vec{r}_j|$ is the distance from atom i to atom j, ϵ is the depth of the potential well (units energy) and σ is the distance at which the potential is zero. For argon, optimal values of the parameters are:

$$\frac{\epsilon}{k_B} = 119.8 \text{ K}, \sigma = 3.405 \text{ Å}. \tag{2}$$

Therefore, the van der Waals force derived from LJ is,

$$F_{LJ}(r_{ij}) = -24\epsilon \left[\left(\frac{2\sigma^{12}}{r_{ij}^{13}} \right) - \left(\frac{\sigma^{6}}{r_{ij}^{7}} \right) \right], \tag{3}$$

In LJ potential, only van de Waals interaction is captured, which is also the major interaction between Argon atoms.

Velocity verlet method

In most MD programs, Verlet method is implemented for time integrator. Given the position, force and velocity at time t_i, the first step is calculating of the velocity at $t_(i+1/2)$ from force by using following equation:

$$\vec{v}(t+\Delta t/2) = \vec{v}(t) + \frac{\vec{F}(t)}{m}\frac{\Delta t}{2} \tag{4}$$

where v is the velocity, r is the position, F is the force and m is the mass of atoms. After we have the velocity at $t_(i+1/2)$ step, the corresponding position can be calculated by following equation:

$$\vec{r}(t+\Delta t) = \vec{r}(t) + \vec{v}(t+\Delta t/2)\Delta t \tag{5}$$

At last, the velocity at $t_(i+1)$ step can be calculated for the force at $t_(i+1/2)$ step by following equation:

$$\vec{v}(t+\Delta t) = \vec{v}(t+\Delta t/2) + \frac{\vec{F}(t+\Delta t)}{m}\frac{\Delta t}{2}. \tag{6}$$

In our code, the verlet method is implemented in MDSystem.verlet function. This function will call other three functions: MDSystem.updateVelocity, MDSystem.updatePosition and MDSystem.updateForce to do each of the three steps. In the end, the MDSystem.checkPBC will be called to check the periodic boundary condition.

Unit conversion

To prevent roundup error from operations of either to large of a number or too small of a number, the following units were used during the calculation, but the outputs were converted to SI unit using the following correlation shown in Table. 1.

Table 1: Table of unit conversion

Calculating Unit	Output Unit
1 Mass unit	1.661×10^{-27} kg
1 Length unit	10^{-10} m
1 Energy unit	1.651×10^{21} J
1 Temp. unit	119.735 K
1 Time unit	1.00224×10^{-13} s
1 Force unit	1.6536×10^{-11} N
1 Pressure unit	0.16536 Pa
1 Velocity unit	997.765 m/s

3 Results and discussion

Significance of PBC

In MD simulation, we have a simulation cell with finite sizes. Without periodic boundary, the atoms on the edge of cell will behave like a surface. If we want to eliminate the impact of surface and predict properties in bulk structure, the periodic boundary condition need to be introduced. In PBC, the current simulation cell is surrounded by 26 on its neighbors (3D) in three directions. Once one atom move to one of the adjacent cell, its mirror image from another simulation cell will moves in and maintain the bulk structure.

Temperature evolution

The temperature of the system, which supposes to plateau at certain equilibrium, kept increasing especially at the longer run time. this behavior is seen throughout all the temperatures with diminished effects in lower temperatures (Fig. 3). As temperature is directly related to velocity, we went back to double check if our momentum remains at zero or not. As shown in Fig. 5, the net momentum fluctuates around zero, therefore, we believe that this temperature keep increasing behavior should stem from rounding error somewhere in our code. This rounding error inadvertently added energy into our system as can be seen in the Fig. **??**, that the total energy also has the same behavior as temperature. We have been spending a long time to look where it could go wrong, but could not identify it with confidence. While this may obstruct us from using results from a longer range of production time, the results from below 5 ps is usable as it still remain in equilibrium, implying the rounding error is still relatively small.

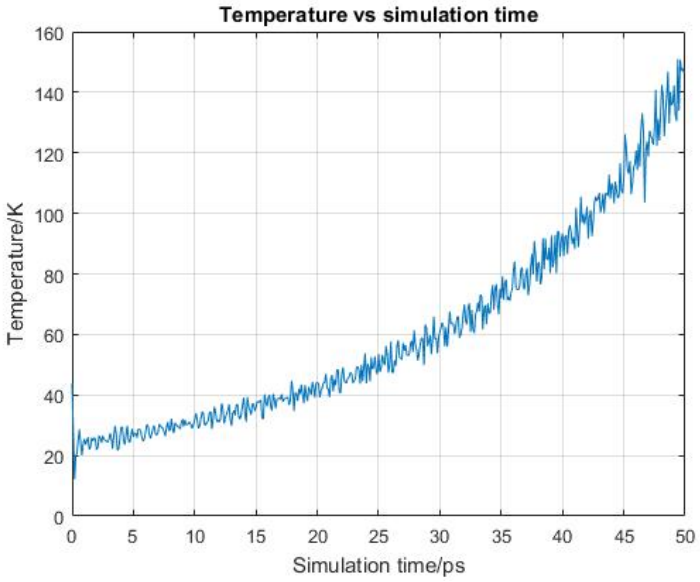

Figure 3: Temperature evolution throughout 50 ps.

Temperature change in NVE dynamic

Initial temperature drop

From beginning of Fig. 6, the temperature drop is observed in NVE dynamic. This can be explained by increasing of potential energy. In the initial structure, the cell length comes from experimental data and the atom position stays at its equilibrium position. In perfect crystal, the potential is minimized. Once the dynamic starts, atoms will move away and oscillate around the equilibrium position, which will lead to increasing in the potential energy. Because the total energy is conserved in NVE dynamic, the kinetic energy have to drop.

Ratio of initial temperature and final temperature

The ratio of initial temperature and final temperature is plotted in Fig. 7. As temperature increasing, this ratio slightly dropped. The average value is about 1.6. The experimental value for melting point is 80K. So we choose initial temperature of 150K of starting point in finding melting point.

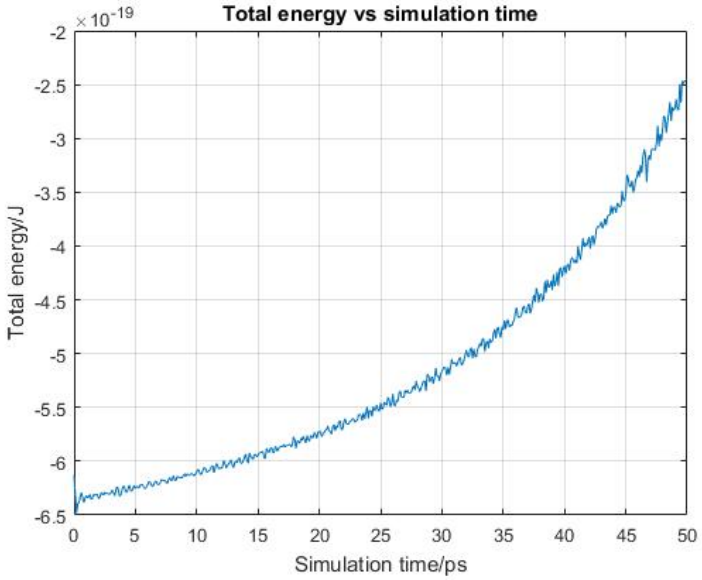

Figure 4: Momentum of the system at different temperature

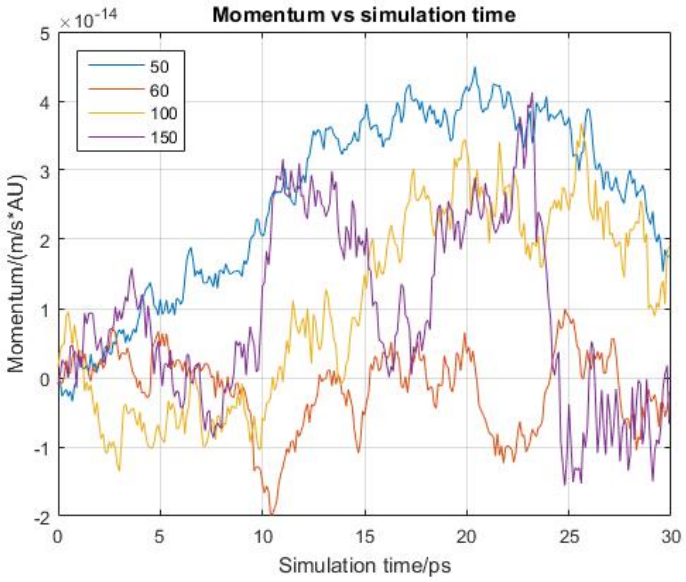

Figure 5: Momentum of the system at different temperature

Melting temperature

The diffusion coefficient is plotted in Fig 8. The sudden increase in diffusion coefficient is observed in the curve of initial temperature of 100K for smaller cell. Due to the limitation of production time, we use this observation as an indication of melting temperature. According to the final temperature calculated above, the melting temperature in our system should between 36K and 60K. For larger cell, the melting temperature should between 61K and 91K, which agrees more with experimental value.

Theoretically, in a perfect crystal, the melting temperature should be overestimated comparing with experimental value. However, due to the uncontrollable increasing total energy, we can only

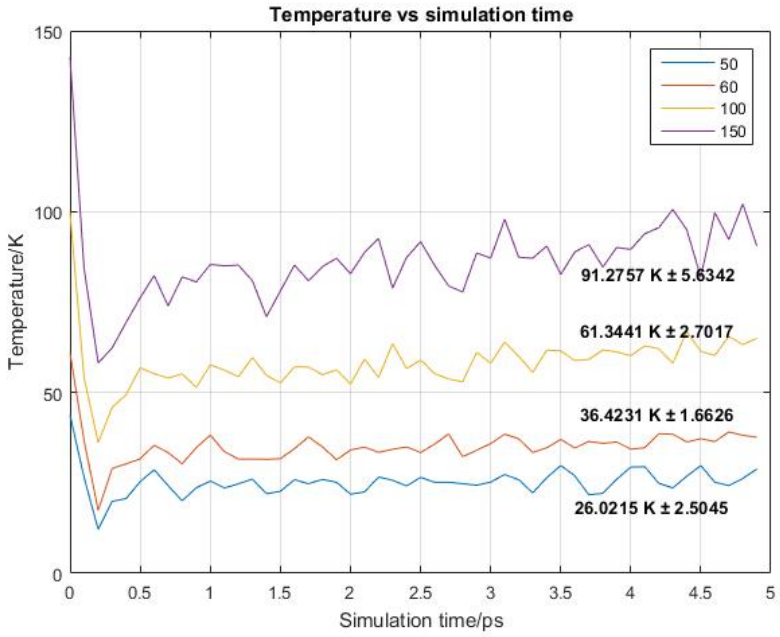

Figure 6: Temperature vs simulation time for large cell

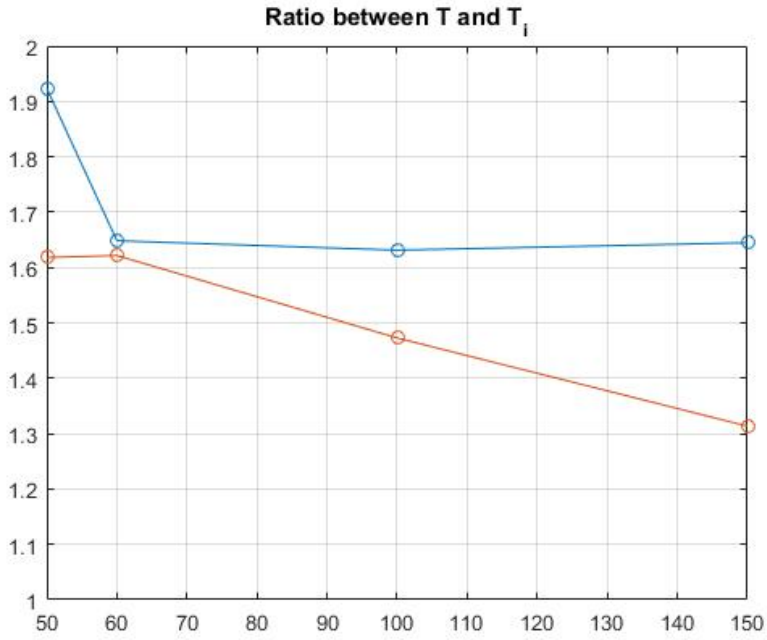

Figure 7: Ratio of initial temperature and final temperature

roughly estimate the melting temperature in such a way. Further work is needed to figure out the convergence problem of total energy.

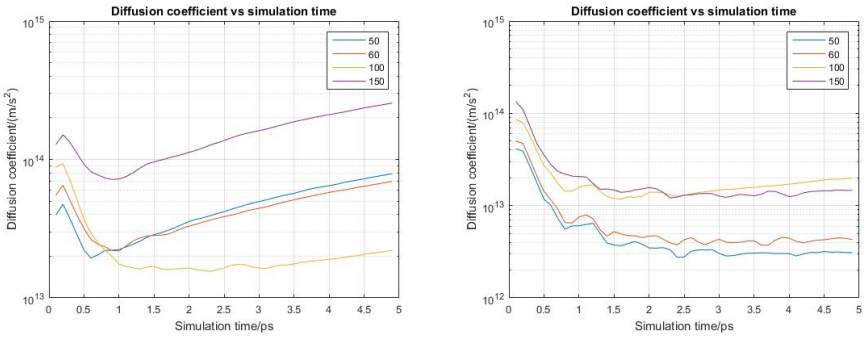

Figure 8: Diffusion coefficient vs simulation time for lattice parameters equal to 5.405A (left) and 5.260 A (right)

4 Conclusion

In conclusion, this study developed a Python code to study dynamics of argon at different temperature using force from LJ potential, periodic boundary condition, and Velocity Verlet algorithm. The issue we faced include rounding error, which gradually increases the total energy of the system (net momentum is checked to be zero). This limited us from acquire more accurate temperature at longer production time.

The melting temperature as determined by violent changes in diffusion coefficient lies between 36-61 K, when lattice constant of 5.260 Åwas used. This is counter-intuitive as typically NVE ensemble should overestimate the melting point and could be contributed to the fact that we cannot use the temperature at longer production time. The results become 61-91 K range, which is closer to experimental value of 80 K, when lattice constant of 5.405 Åis used instead.

References

[Åqvist et al., 1994] Åqvist, J., Medina, C., and Samuelsson, J.-E. (1994). A new method for predicting binding affinity in computer-aided drug design. *Protein engineering*, 7(3):385–391.

[Bergsma et al., 1987] Bergsma, J. P., Gertner, B. J., Wilson, K. R., and Hynes, J. T. (1987). Molecular dynamics of a model sn2 reaction in water. *The Journal of chemical physics*, 86(3):1356–1376.

[Kremer and Grest, 1990] Kremer, K. and Grest, G. S. (1990). Dynamics of entangled linear polymer melts: A molecular-dynamics simulation. *The Journal of Chemical Physics*, 92(8):5057–5086.

[Levitt, 1983] Levitt, M. (1983). Protein folding by restrained energy minimization and molecular dynamics. *Journal of molecular biology*, 170(3):723–764.

[Levitt et al., 1995] Levitt, M., Hirshberg, M., Sharon, R., and Daggett, V. (1995). Potential energy function and parameters for simulations of the molecular dynamics of proteins and nucleic acids in solution. *Computer physics communications*, 91(1):215–231.

[Palmer III, 2001] Palmer III, A. G. (2001). Nmr probes of molecular dynamics: overview and comparison with other techniques. *Annual review of biophysics and biomolecular structure*, 30(1):129–155.

[Rose-Petruck et al., 1999] Rose-Petruck, C., Jimenez, R., Guo, T., Cavalleri, A., Siders, C. W., Rksi, F., Squier, J. A., Walker, B. C., Wilson, K. R., and Barty, C. P. (1999). Picosecond–milliångström lattice dynamics measured by ultrafast x-ray diffraction. *Nature*, 398(6725):310–312.

[Van Duin et al., 2001] Van Duin, A. C., Dasgupta, S., Lorant, F., and Goddard, W. A. (2001). Reaxff: a reactive force field for hydrocarbons. *The Journal of Physical Chemistry A*, 105(41):9396–9409.

[Zu Heringdorf et al., 2001] Zu Heringdorf, F.-J. M., Reuter, M., and Tromp, R. (2001). Growth dynamics of pentacene thin films. *Nature*, 412(6846):517–520.