High Performance Computing : Molecular Dynamics with C++

Final Report

Dwaipayan Roy Chowdhury

5360085

dwaipayan0502@gmail.com

September 21, 2022

Contents

1

Introduction

In simulation, we often come across paths where macroscopic simulations fail to provide precision maths. To further investigate the microscopic or molecular behaviour of any mechanism motivates us to perform molecular dynamics. In molecular dynamics, we substitute a static position of an atom by a dynamical moving atom where the system as a whole is forced into movement. Then we simulate the continuous system by discretizing it and finding out precise thermodynamic and other molecular properties.

In this paper, we explain a comprehensive methodology of performing a molecular dynamics simulations from initially 2-atom system to higher volume gold clusters. Also, we determine different atomic or molecular properties of Au_x (x=13 to 10179) such as Melting point, latent energy, heat capacity. We also present an implementation of parallel computing using clusters to increase computing power of our simulation.

In the beginning, we take a molecular cluster of 54 atoms and try simulating it to test the total energy conservation over time. Then, we use thermostat to visualize the effect of temperature control. Till this point, we have used Lennard jones unit. Following to this, we change our simulation measurements from Lennard Jones unit to actual meaningful units which has been described in the later part of the project.

In this project, we try to solve characteristic behaviors like stress-strain curves for small and large whiskers and visualizing them. We use high-performance parallel computing and observe microcanial (NVE) run where energy is conserved with time. Also, we present different thermodynamic properties like potential energy vs temperature, melting point vs cluster size, heat capacity vs cluster size, latent heat vs cluster size. We use neighbor search list algorithm with a cut-off radius to ignore the very small effect of far-distant atoms. To check if this works fine without neighbor search algorithm, we present a comparative study to that. Also, how cutoff radius makes difference in run-time for our simulation is being presented here by plotting simulation time as a function of size (number of atoms). A thermostat implementation (we use Berendsen Thermostat here) is also shown and to test that, we plot a graph for target and current temperature with time to observe if with long timescale our atom cluster evolves to the specified temperature. Overall, we known that for a molecular dynamic simulation to run successfully, we need to see that the total energy should be constant with time which has also been shown if no additional energy is induced on the system. [1]

We perform some useful tests to ensure that our code works fine as per the MD standards. We also discuss about some of the potential test strategies that help readers understand the results from developer's point of view.

2

Fundamentals of MD

We know that atoms react due to forces. From Feynman's fundamentals, truth of understanding the physical world is "that all things are made of atoms – little particles that move around in perpetual motion, attracting each other when they are a little distance apart, but repelling upon being squeezed into one another." [2]

2.1 Importance of MD

Let us talk about a two atom example of simple salt, for example Na-Cl. From our chemistry knowledge, we can say that this is a strong ionic bond with sodium charge $q_Na = +1|e|$ and Chlorine charge as $q_Cl = -1|e|$. So, from coulomb's law of force between two charge particles, we can deduce the pairwise interaction energy between these two charges and the formula is given by,

$$V_{Coulomb}(r, q_{Na}, q_{Cl}) = \frac{1}{4\pi\epsilon_0}\frac{q_{Na}q_{Cl}}{r} \tag{2.1}$$

Thus, we see if we can find out the charges of these two particles, we can actually perform a molecular study with force vs displacement. [1]

2.2 Applications

Molecular dynamics is now more that 60 years into research and application. This simulation technology got scientists' attention due to its reliability and varied field of applications which includes biochemistry, material science, chemistry, molecular and quantum physics.

In material science, often scientists have a requirement to calculate the equilibrium thermodynamic status of a system, or maybe the phase transitions, surface growth, rough surface contact mechanics, surface defects etc. In bio-science and bio-chemistry, molecular dynamics is used to study drug-molecular conformation dynamics and interactions, then stimuli affect to forces generated by motor proteins, investigate DNA compatibility etc. In chemistry, MD is being used to calculate ionic and chemical equilibrium, intra-molecular and inter-molecular interactions, follow chemical reactions for micro-level time steps, calculate Gibbs free energy etc. This paper focuses more into material science, thus let's put some more light into how MD can be a savior to material scientists. In this category of science, defects is a crucial area of study. Thus, it has always been an area of interest to explore point defects (latent vacancies), line defects (dislocations), planar defects (grain boundaries). Macro experiments just show us about the defects that exist in one macro time frame and the final defect condition in another time frame. But MD helps us to study about the defect transitions and the

trajectory which are based on dynamical behavior of defects and their material properties. [3]

2.3 Limitations

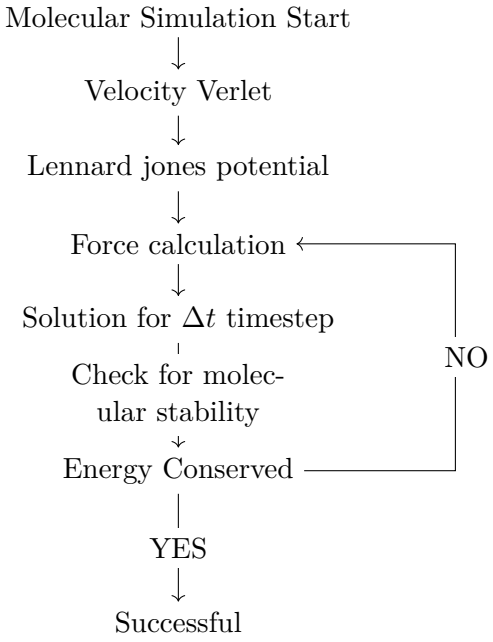
Molecular dynamics is one of the most widely used efficient way of simulation. But this also has some drawbacks and has limitations in flexibility. The main ingredient to molecular dynamic simulation is its potential energy and right description of this potential energy is often a challenge. If the potential function is inaccurate, it will produce an erroneous behavior of trajectories. Usually, potential functions are some semi-empirical formulae that we get from curve-fitting of experimental data (e.g. Gupta EAM potential method is force matching curve fitted function). In this case, we have some constraints in conditions at which the experiments were carried out. For example, we have certain constants like p (the compressible parameter) which are experiment dependent. [3]

Another limitation of molecular dynamics simulation is the dimension scales like length and time. Usually MD simulations stand valid for timescale less than 100 ns and lengthscale less than 100 nm. Most requirements are sufficed by these scales but that's not always the case because of larger relaxation time. In lengthscale, we have some protein molecules that have approx. $10^6 atoms$ and this scale doesn't satisfy the purpose of MD with such small lengths. However, this limitation is partially solved by a mathematical fitting formula, [3]

$$Q(L) = Q_0 + \frac{c}{L^n} \tag{2.2}$$

where, Q_0, c, n are the fitting parameters.

2.4 Methodology



A basic skeleton of MD simulation suggests to first setup the velocity implementation followed by a successful integration of potential and force algorithm. These ingredients help to run a stable multi atom simulation.

3

Theoretical concepts

In this section, we will discuss about the step by step procedure on how to run the first molecular dynamics simulation in our system. This is categorized into 4 sub-sections-

3.1 Velocity Verlet Integration

Velocity and position are the two basic state variables that determines the future state of a molecular system. Acceleration is a dynamic entity that helps us find the evolution of these positions and velocity after a certain time period. Therefore, here we take these 3 variables into account for describing the verlet algorithm. Now, let us consider the variables as 3 dimensional vectors $r_i(t), v_i(t)$, and $a_i(t)$ for N particles in the system. By conventional Newtonian physics, we know $v = dr/dt$ and $a = dv/dt$ and we also know that in a 3D system, we have 6 first order differential equations (3 for position and 3 for velocity) per atom. [4] [1]

$$r_i = \frac{d}{dt} r_i = v_i \tag{3.1}$$

$$v_i = \frac{d}{dt} v_i = a_i = \frac{F_i}{m_i} \tag{3.2}$$

where m_i is the mass of the particle i and F_i is the force of that particle.

Verlet algorithm helps us discretize the continuous velocity evolution with time. Thus, by applying Taylor expansion for a unit increment in timestep, it becomes

$$\vec{r_i}(t) = \vec{r_i}(t + \Delta t) - \vec{v_i}(t + \Delta t)\Delta t + \frac{1}{2m_i}\vec{f_i}(t + \Delta t)\Delta t^2 - \frac{1}{6m_i}\vec{f_i}(t + \Delta t)\Delta t^3 + O((\Delta t)^4 \tag{3.3}$$

Here we find out the updated position and velocity of an atom using,

$$\vec{r_i}(t + \Delta t) = \vec{r_i}(t) + \vec{v_i}(t)\Delta t + \frac{1}{2m_i}\vec{f_i}(\Delta t^2 \tag{3.4}$$

$$\vec{v_i}(t + \Delta t) = \vec{v_i}(t) + \frac{1}{2m_i}(\vec{f_i}(t) + \vec{f_i}(t + \Delta t))\Delta t \tag{3.5}$$

and this is known as the Velocity-Verlet algorithm. Now, for efficiency purpose, we can restructure the equations into a prediction and a correction step. Prediction step will simply predict the next velocity and the corresponding position. The correction step will include the updated forces thereby precising the velocity equation. We use this method of Verlet integration because it conserves the phase-space volume. [1] [2] [5]

3.2 Pair potential

Any type of Molecular dynamics calculation needs inter-atomic force and for that we use Newtons equation of motion $\vec{f} = m\vec{a}$. To compute a MD simulation we need to take care of 3 models named accuracy, transferability and computational cost. With a fixed computational cost, there's always a trade-off between accuracy and transferability. Interatomic potential (also nown as pair potential) can be expressed as

$$E_{pot}(\vec{r_i}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} V(r_{ij}) = \sum_{i<j} V(r_{ij}) \tag{3.6}$$

where, $r_{ij} = |\vec{r_i} - \vec{r_j}|$ is the linear distance between atom i and atom j and $V(r_{ij})$ is the pair interaction energy or pair potential. Looking at the equation, we can see that at R.H.S there is $\sum_{i<j}$ which means for i^{th} atom, it will loop through all the rest of the atoms for calculating the energy or potential. This can be expressed as,

$$\vec{f_k} = -\frac{\delta E_{pot}}{\delta \vec{r_k}} = -\frac{1}{2}\sum_{ij}\frac{\delta V}{\delta r_{ij}} = -\frac{1}{2}\sum_{ij}\frac{\delta V}{\delta r_{ij}}\hat{r}_{ij}(\delta_{ik} - \delta_{jk}) = \sum_{i}\frac{\delta V}{\delta r_{ik}}\hat{r}_{ik} \tag{3.7}$$

where $\hat{r}_{ik} = \vec{r}_{ik}/r_{ik}$ is the unit vector and δ_{ik} is the knocker/dirac delta which means function of two variables and the value is 1 when $i = j$ else 0 when $i \neq j$. [1]

London dispersion force

This force comes into account when two adjacent atoms are located in really close vicinity. This is an attractive force between two Ionic/covalent atoms and creates a dissymmetric dipole characteristics. This is also a quantum mechanical effect which can be also described as a fluctuation of the atomic dipole moment the magnitude of which is $p = ed$ and the electric field of one dipole atom is proportional to $1/r^3$. If an adjacent atom is placed near this, the attractive force decays with distance at an exponential rate of $(r^{-3})(r^{-3}) = r^{-6}$ at short distances. When we talk about Van-der-waals interaction, we assume this force into consideration. [1] [6]

Pauli repulsion

This theory of Pauli states that no two atoms may simultaneously occupy the same quantum state. Thus when they are brought together in close proximity, they tend to perform a repulsive force and the strength is proportional to the overlap of their wave function. As per Hellmann-Feynman theorem interpretation to Pauli repulsion energy, we can write an equation showing the charge density vs energy as,

$$U_{pauli} = \frac{e^2}{a_0} \tag{3.8}$$

where e is the charge and a_0 is the distance between two atoms. [7]

Lennard jones potential

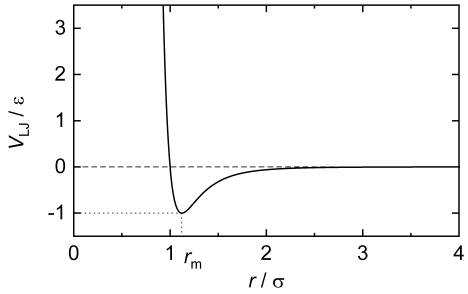
Combining the attractive part (London Dispersion force) and the repulsive part (Pauli Repulsion), Lennard jones potential model is being developed. Thus, it is a function of distance between two molecules that defines the corresponding force by just deriving that once with respect to distance. If the separation or distance between two molecules decrease, the chance of molecular interaction increases. The combined equation is expressed by,

$$V(r) = \frac{1}{2}\sum_{ij} 4\epsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}] \tag{3.9}$$

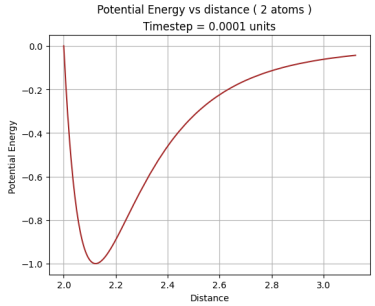
where, V is the interatomic potential, ϵ is a measure to how strongly the atoms attract each other, σ is the distance when interatomic potential is 0 and describes a measure of van-der-waals radius, r is the distance between two atoms. The term r^{-12} comes from Pauli repulsion model and r^{-6} refers to the dispersion theorem. We can also derive the lennard-jones force by,

$$\begin{aligned}\vec{f_r} &= -\nabla_r U(r) \\ &= -\frac{d}{dr}[4\epsilon(\frac{\sigma^{12}}{r^{12}} - \frac{\sigma^6}{r^6})] \\ &= -4\epsilon[-\frac{12\sigma^{12}}{r^{13}} + \frac{6\sigma^6}{r^7}] \\ &= 48\epsilon[\frac{\sigma^{12}}{r^{13}} - \frac{0.5\sigma^6}{r^7}]\end{aligned} \tag{3.10}$$

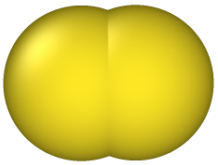
and then for all the atoms, we use $\vec{f_r} = -\sum_i \vec{f_r}$ to calculate the total force where $\vec{f_r}$ is the pair force. [8]



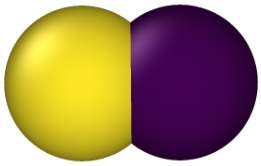
(a) Expected Lennard jones potential plot



(b) Simulated Lennard jones for 2 atoms



(c) 2 atoms initial distance



(d) 2 atoms final distance

Table 3.1: Here we show the plots for potential vs distance simulation in 2 atom system.

3.3 Berendsen Thermostat

Thermostat is a technology used to measure and control the temperature of the system to a specified temperature. By kinetic definition of temperature and also from equipartition theorem we can write,

$$N\frac{3}{2}K_BT = E_{kin} = \sum_i \frac{1}{2}mv_i^2 \tag{3.11}$$

Thermostats in molecular dynamics is used to remove excess energy from the system that is generated from implementing boundary condition for e.g. energy introduced by work performed on our system. Also, we need thermostat for relaxation. We setup a molecular

dynamics simulation initially as we are typically far from the equilibrium. And to achieve this equilibrium we might heat the system. Thus, we provide the system with an additional energy by factoring the velocity and then allow it to relax for some time. Also, based on the purposes, thermostat can be categorized into different types like constraint method (velocity rescaling and Berendsen), stochastic method (Andersen, langevin) and extended systems(Nosé-Hoover).

Berendsen thermostat follows acceleration or exponential deceleration method in controlling the temperature. The Newtonian governing thermostat equation to this can be written as,

$$m\vec{v_i} = \vec{f_i} + \frac{m}{2\tau}(\frac{T_0}{T} - 1)\vec{v_i} \tag{3.12}$$

where τ is the relaxation time and $T = T_0$ is the damping coefficient vanishes thereby deducing it to Newton's law of motion. [1]

Rescaling velocity

From the equipartition theorem, we know the relation between temperature and velocity of atoms in a structure. Thus, for rescaling the velocity, we might use a rescaling factor λ that when multiplied to velocity modifies the velocity such that the whole system reaches to the expected temperature. We can derive the expression of the factor by,

$$N\frac{3}{2}K_BT = \sum_i \frac{1}{2}mv_i^2 \tag{3.13}$$

$$\lambda = \sqrt{\frac{T_0}{T}} \tag{3.14}$$

and we thus rescale the velocity by $\vec{v_i} -> \lambda\vec{v_i}$ after every timestep. This ignores the relaxation time and thus becomes very intrusive way of formulation. From [1] we get a formula of temperature variation with relaxation time which is exponential and can be written as,

$$T(t) = T_0 + (T_1 - T_0)e^{-\frac{t}{\tau}} \tag{3.15}$$

Thus we can remodify the rescaling factor by the given expression 3.15 and be written as,

$$\lambda = \sqrt{\frac{T_0}{T} + (1 - \frac{T_0}{T})e^{-\frac{\Delta t}{\tau}}} \approx \sqrt{1 + (\frac{T_0}{T} - 1)\frac{\Delta t}{\tau}} \tag{3.16}$$

here, T is the current temperature and T_0 is the measured temperature.[1]

3.4 Embedded atom potential

Often, molecular dynamic simulations are used in predicting and assessing the structural defects and transformations, phase transition behaviors etc and the simulations are based on empirical interatomic potentials. While performing lennard jones pair potential, we choose to consider only the adjacent 1 directional effects of adjacent atoms. But, this is not a ideal condition and in molecular cluster, one atom is surrounded by various atoms and charges. In ionic bond, lennard jones potential completely ignores the electron charge and its effect to the neighbors. Thus, researchers proposed another method that relates energy between atoms and distance. The method considers charge density and is based on tight-bonding theory.

3.4.1 History of EAM

Solid metals react elastically to stress or deformation. In cubic crystalline materials, there are typically 3 elastic constants - C_{11}, C_{12}, C_{44} that basically describes the resistance in volume change, dilation shear and simple shear and in isotropic materials which are rotationally invariant, there are 2 constants. The drawback of pair potential is that it accounts for only 2 elastic constants for crystal. For example Lennard jones potential would consider only 2 elastic constants for FCC cubic lattice. This basically creates an additional pressure called 'Cauchy pressure'. The EAM method is set to overcome this Cauchy pressure $P_c = (C_{12} - C_{44}/2)$. This relation can be balanced by introducing an additional energy term that depends on volume per atom $v = V/N$ [9] [10]

$$E(\vec{r_i}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} V(r_{ij}) + NU(V/N) \tag{3.17}$$

where left part takes care of the pairwise potential and right side handles the cauchy pressure gap by using a functional energy per volume mathematical substitution.

3.4.2 Modification of EAM

In EAM, the attraction part is assumed to be a function of electron density which is caused by the host crystal lattice. It also accounts for the Pauli's repulsion $E_{pot} = F[\rho(\vec{r})] + \phi$ where F is the embedded atom function that relates energy and electron density and ϕ is the Paulis repulsion. The EA function $F(\rho)$ is negative and usually written as,

$$E_{pot}(\vec{r_i}) = \sum_i F(\rho_i) + \frac{1}{2}\sum_{i,j}\phi(r_{ij}) \tag{3.18}$$

Now, we need to identify the charge density ρ_i which can be found or approximated from the density of nuclei. This means each atom in close proximity will contribute to the charge of the atom i. The formula or expression to calculate the charge density is by introducing a step function which is 0 when $r = r_c$ and 1 otherwise.

$$\rho_i = \sum_j f(r_{ij}) \tag{3.19}$$

$$f(r_{ij}) = \left\{ \begin{array}{ll} 0, & \text{if } r = r_c \\ 1, & \text{if } r \neq r_c \end{array} \right\} \tag{3.20}$$

Generally, a step function is not differentiable, but with balanced regularization, one can create a continuous function thereby discretizing it for our simulation.

3.4.3 Gupta potential

As referred in the paper by Cleri and Rosato [11], the cohesive properties of metals and alloys are usually caused by the band density of state (DOS) and is determined by the second order moment of DOS which describes a matrix in Hamiltonian with electron paths from origin. From what is being found in the paper, we can write the band energy (i.e. proportional to this second order moment μ) as follows,

$$E_B^i = \{\sum_j \zeta_{\alpha\beta}^2 e^{-2q_{\alpha\beta}(r_{ij}/r_0^{\alpha\beta}-1)}\}^{\frac{1}{2}} \tag{3.21}$$

here r_{ij} is the distance between two atoms and $r_0^{\alpha\beta}$ is the first neighbor distance in the $\alpha\beta$ lattice. To substitute and stabilize the crystal energy, a counter-energy formulation is made

equivalent to Pauli's repulsion model. This is considered to be pairwise as described by Born-Mayer ion-ion repulsion. This repulsion energy can be written as,[11]

$$E_R^i = \sum_j A_{\alpha\beta} e^{-P_{\alpha\beta}(r_{ij}/r_0^{\alpha\beta}-1)} \tag{3.22}$$

here, parameter p depends on the compressibility of the bulk metal. This pair-wise repulsion term takes care of the pair electostatic interactions and exchange and correlation information. Thus, the total Gupta potential can be deduced as, [11]

$$E_c = \sum_i (E_R^i + E_B^i) \tag{3.23}$$

3.5 Parallel computing

Serialized simulations are often cumbersome as it deals with huge chunk of data and normally there are places where we cannot avoid looping through the iterations. Thus, computer scientists deciphered a way to tackle this expensive way of simulation. Our computers have multiple cores which aren't used completely everytime we run a simulation. We need this technology because we cannot speedup a single processor more than a certain level because the amount of electrons moving through it can melt down the processor due to higher heating range. Thus capitalizing all the cores simultaneously for a single simulation is what we call as parallel computing. This technology is a paradigm due to its scaling flexibility and also be associated to concurrent computing. Even though it eases out the simulation expense, it brings couple of difficulties like More bugs, tasks are independent in processes, ghost nodes need to be taken care of.

In terms of hardware architecture, to perform efficient coding, a developer has to understand the CPUs and core distribution which allows the simulation to run parallely. Usually at first level, we can only think about the multiple core of our own system. The next level comes when we talk about high-performance computing i.e. combining different systems to form a cloud cluster.

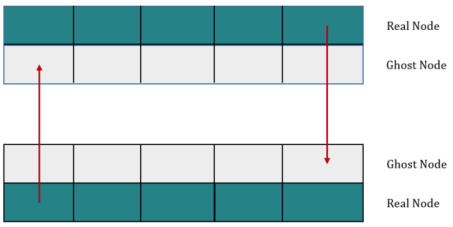


Figure 3.1: A schematic on how interaction works between nodes.

Domain decomposition is another important concept that helps us decide the cluster size. Till this point of time, we didn't have any space constraints in our MD. To perform domain decomposition, at first we need to create a fixed boundary for the simulation. Let's denote the space with Ω. Assuming our vectors to linearly reside in 3 independent vector space $A = (\vec{a}_1, \vec{a}_2, \vec{a}_3)$, this can be expressed as $\vec{r}_i = A\dot{\vec{s}_i}$ where $\vec{s}_i$ is the scaled position of the atom i,

$$\vec{r}_i = s_{i,1}\vec{a}_1 + s_{i,2}\vec{a}_2 + s_{i,2}\vec{a}_2 \tag{3.24}$$

We use rectilinear domain such that $\vec{a}_1 = (L_x, 0, 0), \vec{a}_2 = (0, L_y, 0), \vec{a}_3 = (0, 0, L_z)$ where L_x, L_y, L_z are the linear dimensions. Now we decompose the full domain into linear scales - $L_x/N_x, L_y/N_y, L_z/N_z$. Each domain works on it's own way and gets the final result stored in each rank. Then we use a function to accumulate the results into one process. [1] In the figure 3.1, we can see how all ranks are being surrounded by ghost nodes. It is a communication process amongst different cores of our system which helps us run bulk data-simulations at an efficient manner.

4

Implementation

In chapter 5, we discuss about all the simulations performed. For easy readability and understanding, we also produce graphical simulation results in plots and with physical molecular images. We present the algorithms used to perform the simulation along with their testing strategies. For Velocity verlet and lennard jones potential, we have testing algorithms to test using Googletest and for Berendsen thermostat, we have a graphical test method.

4.1 Velocity verlet integration

This section describes how we have used the Verlet integration algorithm to wrap the velocity movement driven by its forces. From our Newton's law of motion, $v = v_0 + at$ where v is the velocity and a, t are the acceleration and timescale of particle. So, we update the velocity and then update the position accordingly using $x = vt$ where, x is the distance traversed and v, t are the particle velocity and timescale of the particle. Till this point, we had our verlet integrator 1 0 for prediction step as described in 3.1.

Algorithm 1 Verlet integrator 1

```
1: dt ← timestep
2: atoms ← atomsStructure
3: for i = 0 to Number of atoms do
4:     atoms.velocities.col(i) += 0.5 * atoms.forces.col(i) * dt;
5:     atoms.positions.col(i) += atoms.velocities.col(i)*dt;
6: end for
```

Now, we correct the velocity with the updated position of the atom using the prediction velocity. The algorithm is just a simple Newton's law of motion.

Algorithm 2 Verlet integrator 2

```
1: dt ← timestep
2: atoms ← atomsStructure
3: for i = 0 to Number of atoms do
4:     atoms.velocities.col(i) += 0.5 * atoms.forces.col(i) * dt;
5: end for
```

Test

Let's discuss the test strategy for Verlet implementation in this section. We have categorized the test into two classifications - single atom, multiple atom. For single atom, we have

initialized one atom with its positions, velocities and forces defined in our atom structure. Now we provide $1unit$ force to the atom and decide the timestep to be $1unit$ as well. After initialization, we apply the Verlet implementations 0 and 0 for 10 iterations which means for $10unit$ timescale. Going by mathematics, we must have the position value near 50 and velocity value near 10 as,

$$\boxed{\textbf{v = a * t -> 1 * 10 = 10 units.}}$$

$$\boxed{\textbf{x = 0.5 * a * t*t -> 0.5 * 1 * 10*10 = 50 units}}$$

where the notations have their usual meaning. To test the proximity of our implemented code with the analytical solution, we use $ASSERT - NEAR(atoms.positions(0), 50, 1e-6)$ for positions and $ASSERT - NEAR(atoms.velocities(0), 10, 1e-6)$ for velocity.

For multiple atom (here we consider 10 atoms), we perform the similar steps and initialize forces in all 3 dimensions (x,y,z) followed by running the Verlet algorithms for a timescale of 10. We thus perform the similar assertion test and check if analytical and implemented result produce similar results with a buffer of $1e-6$.

4.2 Lennard jones potential energy

We use the formula 3.9 for calculating potential energy of the system. In this algorithm 0 we initialize the interaction factors *sigma* and *epsilon* to 1. Also, we incorporate a neighbor list concept by setting the cutoff radius as described in **??** and then loop through all the remaining atoms for each atom. We also use this formula 3.9 to calculate the force by differentiating the potential once w.r.t distance $\frac{\delta U}{\delta x}$

Algorithm 3 Lennard jones potential energy

```
1: dt ← timestep
2: atoms ← atoms structure
3: sigma ← sigma
4: eps ← epsilon
5: for auto [i, j]:  neighbor list do
6:    for k = 0; k < 3; k++ do
7:       r2 += (atoms.positions(k, i) - atoms.positions(k, j))
   *(atoms.positions(k, i) - atoms.positions(k, j));
8:    end for
9: end for
10: rnorm = sqrt(r2);
11: term1 = pow(quot, 12.);
12: term2 = pow(quot, 6.);
13: Pot += 4 * eps * (term1 - term2);
```

Test

Testing potential energy and corresponding force is very crucial for our molecular dynamics simulation because this particular type of simulation is driven by this energy as described in the limitations section 2.3. Here, we consider multiple atoms with $epsilon = 0.7$ and $sigma = 0.3$ and the difference for numerical finite difference computation $delta = 0.00001$. We set the masses to 0 and positions as random. Then we run a loop for all 10 atoms in all 3

directions to shift the atoms to right by factor *delta* and the move to left by factor $2 * delta$ and calculate corresponding potentials. The analytical force solution can then be derived from the formula $Force = Potential/distance$. A short implementation for test is written here for 1 iteration and one direction,

Step 1. Move the atom to right by distance delta
Step 2. Move the atom to left by shifting to 2 * delta
Step 3. Bring it to its native position
Step 4. Calculate finite difference of force
Step 5. Check with our simulated force result

4.3 Kinetic energy

We need to compute kinetic energy of the system to know if the total energy is conserved in the system. Also, kinetic energy is required for velocity scaling in thermostat. The algorithm for kinetic energy is listed below.

Algorithm 4 Kinetic energy

```
1: dt ← timestep
2: atoms ← atoms structure
3: for i from 0 to atoms.nbatoms() do
4:     for k = 0; k < 3; k++ do
5:         atoms.kinenergy(i) = 0.5 * atoms.masses(i)*atoms.velocities(j, i) *
   atoms.velocities(j, i);
6:     end for
7: end for
8: Return ← atoms.kinenergy.sum()
```

We have a property called *kinenergy* in the class *Atoms* that stores information about the kinetic energy of the system. We simply compute the kinetic energy using general formula $K.E = \frac{1}{2}mv^2$.

4.4 Measurement units

In lennard jones potential model we have no unit specify any unit as it has its own set of unit adjustments. But in real-life, it has no significance and thus we use EAM potential model where unit measurements are crucial for understanding how the simulation behave. In our simulation, we work with length $[l]$, time $[t]$ and mass $[m]$ as our primary dimensions. All other derived dimensions are velocity $[v]=[l]/[t]$, accelerations $[a] = [l]/[t]^2$, forces $[a]= [m][l]/[t]^2$ and energies $[E] = [m][l]^2/[t]^2$. It has been observed that the bond energy in metals and covalent atoms are close to 1 eV and 1Å is approximately the distance between them. Thus, we fix these two units for our calculation. Since we have unit of one primary dimension fixed as Å for length now and unit of one secondary dimension as eV for energy, we can have two possibilities in achieving the other two primary dimensions time $[t]$ and mass $[m]$. We can fix our mass and vary time or vice-versa. Timestep variation is more widely used as this directly impacts the energy drift of a system and since velocity integrator function is a timestep dependent phenomenon, we decide to fix the unit of time and multiply the mass with a factor. Now, in molecular dynamics, time steps are in order of femto-seconds (fs) as this suffice the $dt- > \Delta t$ in Verlet integration.[1]

We decide our timestep unit to be $[t]$ = 1fs and thus we can derive the mass factor $[m]$=$[E][t]^2/[l]^2$. Widely used mass units are in g/mol and the derivation of factor that we use in adjusting the mass can be derived as $[m] = [1eV][1fs]^2/[1\text{Å}]^2 = 1.6 \cdot 10^{-19} \times 10^{-30}s^2/(10^{-20}m) = 1.6 \cdot 10^{-29}$ Kg. To convert this in g/mol, we have to adjust it using avogadro number using $[m] = 1.6 \cdot 10^{-}29Kg \times mol/mol = 0.009649$ g/mol. We can multiply our mass (for gold, m = 196.96657 g/mol) with a factor 1/0.00964 = 103.6 to achieve 1fs.[1]

4.5 Berendsen thermostat

Thermostats are important for driving the molecular simulations to its equilibrium state as explained in section 3.3. Also, we use the formula 3.16 to write our algorithm. Relaxation time here means the time interval that's provided after rescaling the velocity for one time. This function to rescale the velocity is being used after we calculate the velocity using Verlet algorithm and force using either EAM or Lennard jones method. There are no as such testing strategy followed to tally the data with any analytical one. However, we here observe the temperature plot to slightly drift towards the target temperature shown in the image.

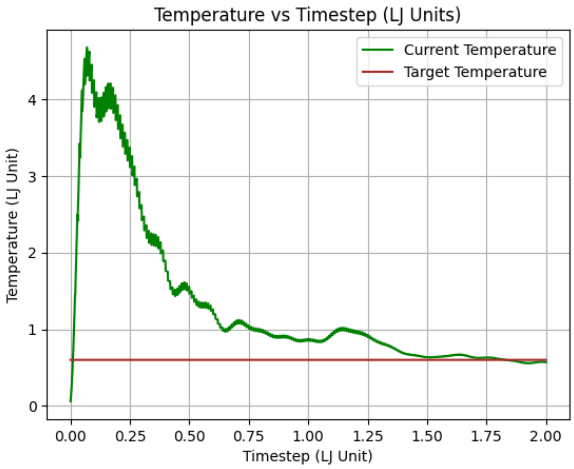


Figure 4.1: This plot is to observe the target and current temperature with timestep for a long scale.

Algorithm 5 Berendsen thermostat

```
timestep ← Timestep
atoms ← Atoms structure
relaxationtime ← Relaxation time
temperature ← Target temperature
for i from 0 to atoms.nbatoms() do
   for k = 0; k < 3; k++ do
      vSqdSum += atoms.velocities(j,i)*atoms.velocities(j,i);
   end for
end for
lambda = sqrt(1+((3*kb*atoms.nbatoms()*temperature/(atoms.masses(0)*vSqdSum))-1)*
timestep/relaxationtime);
for i = 0; k < atoms.nbatoms(); k++ do
   atoms.velocities.col(i) *= lambda;
end for
Return ← atoms.velocities
```

4.6 Parallel computing

In the code implementation, we have a class written as domain.cpp that described certain functions about decomposition of spatial 3D matrix, then accessing and transferring the ghost nodes. Let's discuss them one by one.

1. Domain.enable() - This function is used to decompose the cluster into size of ranks/-cores and allot them in a sequential manner. We distribute the mass, position, velocity and force properties into these domains as energy is calculated as a whole from these information.

2. Domain.disable() - We write this function to wrap up the distributed ranks into serialized one and after executing this, all clusters have the same information (i.e. the total one) in their repository.

3. Domain.exchange() - We need to exchange peripheral information to maintain a communication amongst ranks and this function typically does that by exchanging the atoms.

4. Domain.updateghosts() - Ghost nodes are required that provides a kernel to the atoms in each rank. This is required in computing the embedded atom potential because we have a cut-off radius to calculate the energy. So, for atoms in boundary of each process, we have no information about the positions of atom in other ranks which needs to be used in potential energy calculation. Thus, we use this function to fuse the ghost nodes in the algorithm.

5. MPIReduce() - Like mentioned earlier, we calculate the potential energy by summing up the local energies of all processes without the ghost nodes. This function from MPI library is used to perform this operation. The syntax for this is ,

MPIReduce(local param, global param, MPI SUM)

Algorithm 6 MPI modifications

```
*argc ← argument
*argv[] ← argumentlist
MPIInit(argc, argv)
Decompose domain using Domain()
domain.enable(atoms)
for i = 0 to Step count do
    Verlet Step 1 - Prediction
    domain.exchange-atoms(atoms)
    domain.updateghosts(atoms, 2 * rc)
    localpotential = gupta(atoms, neighborlist, rc)
    Verlet Step 2 - Correction
    localkinetic = Kinetic(atoms, rc, eps, sigma)
    MPIReduce(localkinetic, globalkinetic, MPI SUM)
    MPIReduce(localpotential, globalpotential, MPI SUM)
    domain.disable(atom)
    Save the xyz file and print the potential and kinetic energy for the
timestep
    domain.enable(atoms)
end for
```

5

Results

In this chapter, we discuss about the simulation outcomes of the theories and implemented code described in earlier chapters. We plot the energy conservation graph to ensure that our simulation follows the energy conserved. Other simulations that are performed and produced here are plotting cut-off radius for different distances, latent heat, melting point, potential energy of gold cluster. We also simulate a nanowire whisker to calculate surface defects and stress of the material.

5.1 Energy conservation

We run a simulation to calculate the total energy vs time for a timestep of 0.001 and timescale of 2 Lennard jones unit. The simulation is observed for 54 particles and total energy is calculated as the summation of kinetic energy and potential energy.

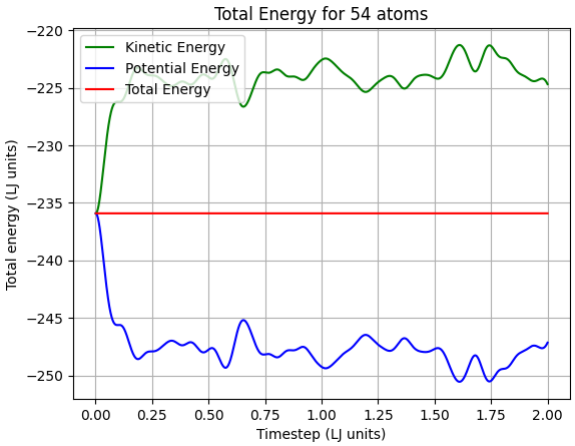


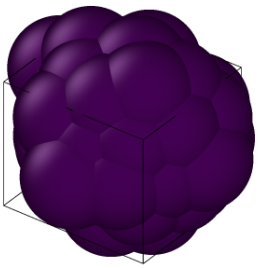
Figure 5.1: This plot shows that our overall energy is conserved. It is calculated with timestep $\Delta t = 0.001$ for 2000 timesteps amd kinetic energy is plotted with an offset of -236 LJ units

Now, we present the simulation snapshots for the energy conserved state. As no external energy is provided on the system, we observe a stability in the cluster. This stability can also be understood by looking at the Radial distribution function (RDF) which can be calculated,

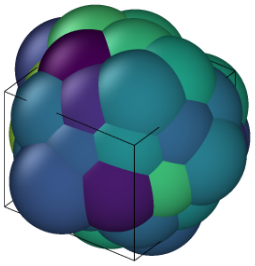
$$g(r) = \frac{dn_r}{4\pi r^2 dr * \rho} \tag{5.1}$$

where dr_n is a probability density function that computes the number of particles within a radius dr. This function gives us information about how many particles are present in a

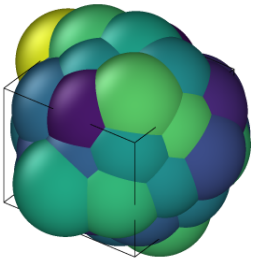
specific radius. If we have a stable count, this means our energy is also conserved and there's no loss or gain in energy or any phase transition.



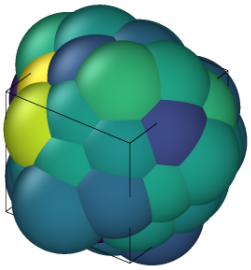
(a) At t = 0 fs



(b) At t = 700 fs

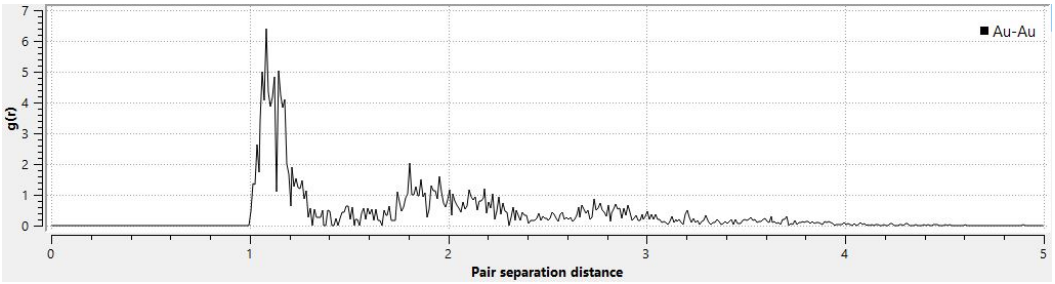


(c) At t = 1400 fs

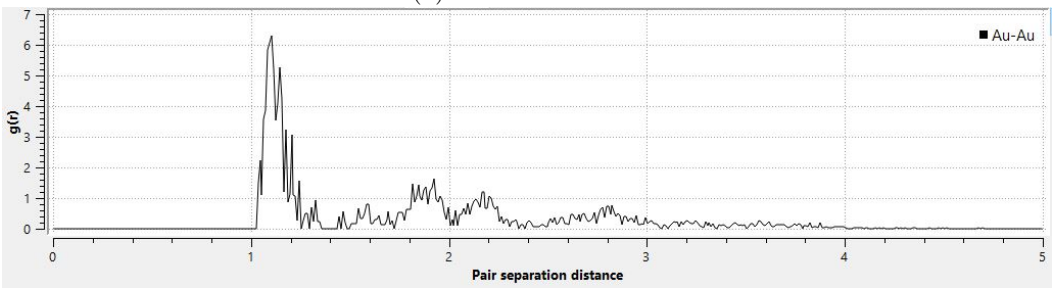


(d) At t = 2000 fs

Table 5.1: Stable MD simulation with time when energy is conserved



(a) RDF at t = 0 fs

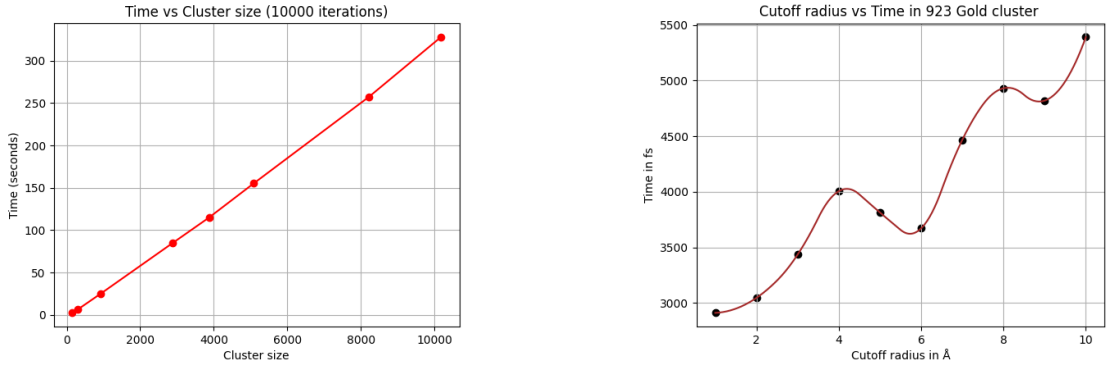


(b) RDF at t = 2000 fs

Table 5.2: Radial distribution function at t=0 and t=2000 fs to show that no atoms are lost or gained in a specific radius.

5.2 Size with time

Cut-off radius is a crucial part in calculating the potential energy is a system. It is not always necessary to calculate the positional distance of all atoms to find the potential of one atom because the lennard jones model shows us that after a certain distance the force between two atoms become negligible 3.1. We here try simulating a 923 gold cluster with 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Å radii. We calculate the time it takes to simulate the code for respective cutoff-radii to show how it scales up with distance. Also, we here present how the time varies with cluster size.



(a) Computation time vs cluster size

(b) Computation time vs cutoff radius

Figure 5.2: This plot shows how computation time scales up with cluster size and cut-off radius

5.3 Gold cluster simulation

5.3.1 Suitable timestep

Timestep decision is a vital step in establishing stable molecular dynamics simulation because too small timestep leads to an unreal system and would have to iterate for higher order of times. Choosing higher values of timestep can create errors in Euler integration formula where for small dt we ideally derived the discrete model. Physicists proposed that the timestep must be less than the time period of the fastest bond vibration by an order or two. We present the calculation for the range of the timestep here,

$$\begin{aligned} T &= 2\pi\sqrt{\frac{\omega}{K}} \\ &= 2 \cdot 3.14 \cdot \sqrt{\frac{196}{(10 \cdot 10^5)(2)(6.023 \cdot 10^{23})}} \text{ seconds} \\ &= 2 \cdot 3.14 \cdot 1.276 \cdot 10^{-14} \text{ seconds} \\ &= 80 \text{ fs} \end{aligned} \tag{5.2}$$

where, T is the time period of fastest oscillation, $\omega = \frac{m_1 \cdot m_2}{m_1 + m_2} = \frac{m}{2}$ $[g]$ is the reduced mass and the bond force $K = (10 \cdot 10^5)$ $[g/sec^2]$ for double bonded Au-Au. Since, we have total time period $T = 80fs$, we can use our timestep in range $\Delta t \leq 8fs$. Taking an approximate middle value between $0.8fs$ and $8fs$ (since timestep should be at least one or two order less than the time period of fastest vibration which is $80fs$), we decide our timestep as $2fs$.

5.3.2 Energy analysis

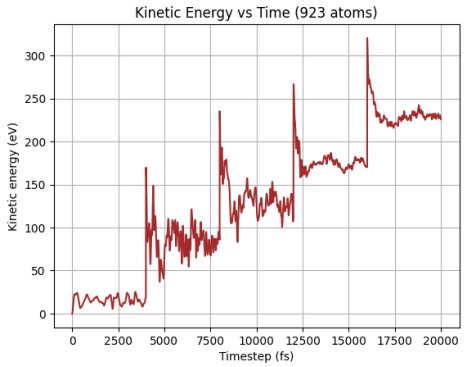
Energy is a very crucial part of simulation and from this primary information, we can arrive at different other properties like melting point, latent heat, heat capacity etc. In this part, let us discuss how different variations of other physical dimensions vary with energy.

Suppose we have an atomic cluster in solid state. From there, we can calculate the total potential energy of the system using the position vectors. Now, if we provide additional amount of heat to the system which has to be constant ΔQ, it gains energy and eventually the bond energy is weakened. Gradually the state of the atoms changes to liquid and then vaporizes eventually. This transition of phases can be observed by plotting the total energy vs temperature.

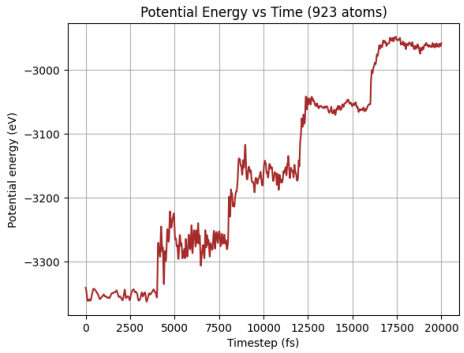
To provide a constant heat to a system, we can't simply multiply the velocities of atoms with a constant factor α. Because with rise in velocity, this will exponentially increase the temperature and energy. Thus, we have formulated it in a different manner,

$$\begin{aligned} \Delta Q &= E_{kin}^{'} - E_{kin} \\ &= (\alpha^2 - 1)E_{kin} \\ Thus, \alpha &= \sqrt{\frac{\Delta Q}{E_{kin}} + 1} \end{aligned} \tag{5.3}$$

Using this formula, we provide an additional heat to the system and thus observe the potential and total energy with time and temperature.



(a) Kinetic energy vs time

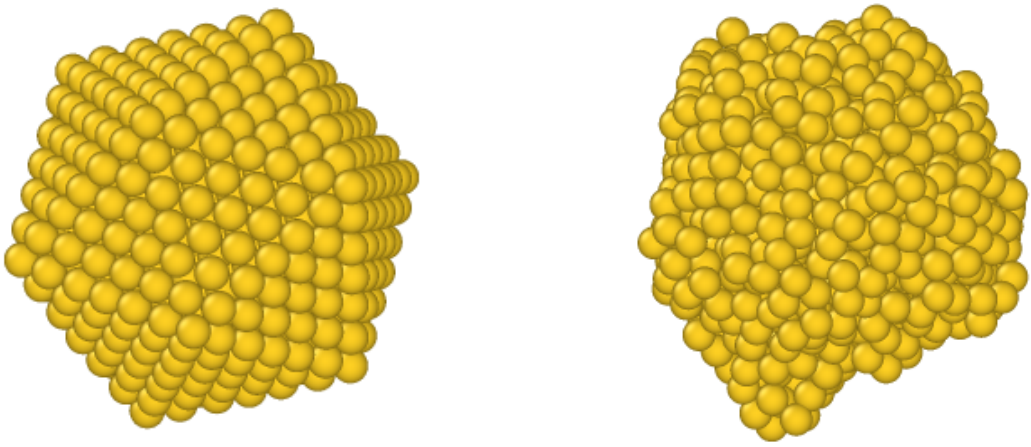


(b) Potential energy vs time

Table 5.3: Energy vs simulation time at $\tau = 4000fs$, $\delta t = 2fs$, $\Delta Q = 150eV$ for 923 atoms.

From the figure 5.3, we notice that when we provide enough heat, the system tends to lose potential energy because the atoms have been separated beyond the cutoff radius. We can observe this in the Ovito simulation figures that's shown below.

Now, we want to observe the melting point, latent heat and heat capacity of the system. With rise in temperature, we know that the energy will rise slowly (rate of increase depends on the heating rate i.e. relaxation time, amount of heat provided). But this transient is noticed at a constant state of molecules. When the atoms reach their melting point, the temperature stops increasing but the energy still increases. This hints us to the fact that this energy is required for phase change which means the latent heat. After the complete phase transition is accomplished, again the slope between energy vs temperature gains a constant value thereby increasing the temperature. As we know, heat capacity (C_p) is defined by the

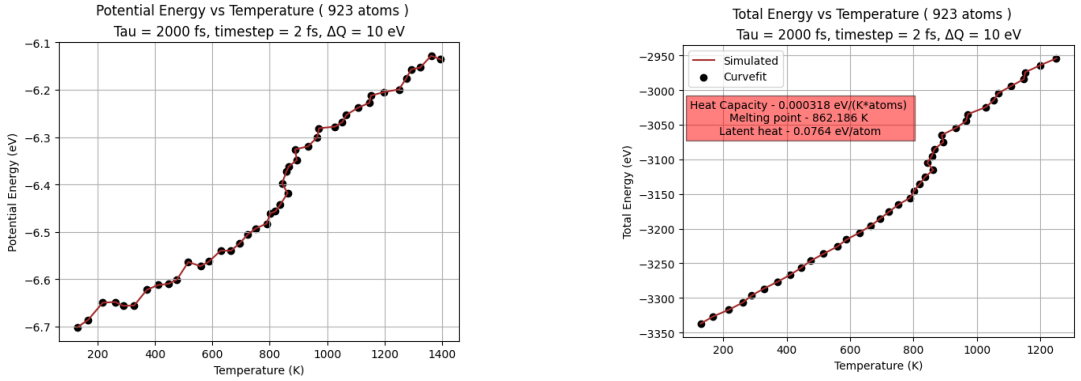


(a) At t = 0 fs (Solid state)

(b) At t = 6000 fs (Molten state)

Table 5.4: Simulation at different time frame generated for $\tau = 2000fs$, $\delta t = 2fs$, $\Delta Q = 10eV$ for 923 atoms.

change in energy w.r.t temperature, we can thereby refer the slope as the Heat capacity. We also study potential energy vs temperature curve to exactly know how the particle distances with increase in temperature.



(a) Potential energy vs temperature

(b) Total energy vs temperature

Table 5.5: Energy with temperature to calculate the melting point, latent heat and heat capacity $\tau = 2000fs$, $\delta t = 2fs$, $\Delta Q = 10eV$ for 923 atoms.

5.3.3 Properties with cluster size

Till this point, we have discussed about the energy conservation and different thermodynamic properties of Gold cluster (Au) with 923 atoms. Now, let us see how these thermodynamic behaviors vary with the cluster size (No. of atoms). To calculate the melting point, we observe the RDF function when drops suddenly. The following is observed in a logarithmic scale with a curve-fitted legend to understand the trend of how this varies.

We can see that the melting point for lower number of atoms have significant variations with number of atoms and thus eventually saturates with number of atoms increasing beyond 923 atoms. This also matches with the results of the research article written by JC Ruiz and L Rincon.[12] For bulk clusters, we observe lower range of temperatures and the numerical value of that is 850 ± 30 K. Also, we have arrived to these melting points from the below

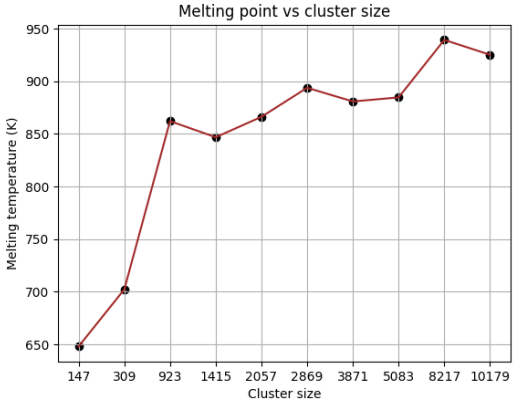
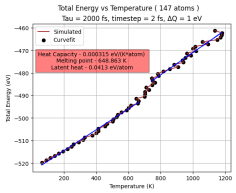
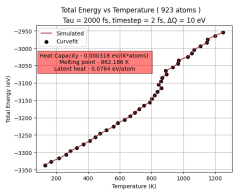


Figure 5.3: Melting point vs cluster size plot for 147, 309, 923, 1415, 2057, 2869, 3871, 5083, 8217, 10179 number of atoms

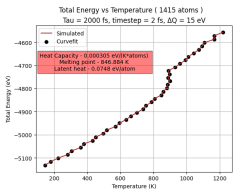
figures using RDF function,



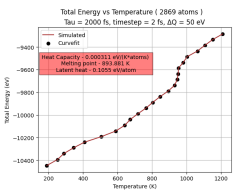
(a) 147 atoms $\Delta Q =$ 1 eV



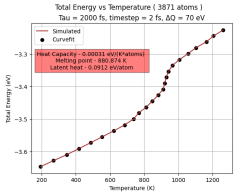
(b) 923 atoms $\Delta Q = 10$ eV



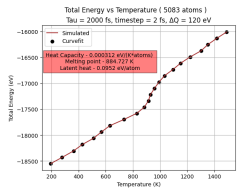
(c) 1415 atoms $\Delta Q = 15$ eV



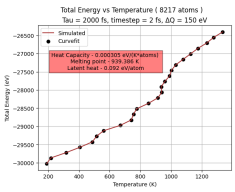
(d) 2869 atoms $\Delta Q = 50$ eV



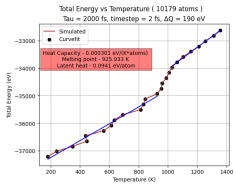
(e) 3871 atoms $\Delta Q = 70$ eV



(f) 5083 atoms $\Delta Q = 120$ eV



(g) 8217 atoms $\Delta Q = 150$ eV



(h) 10179 atoms $\Delta Q = 190$ eV

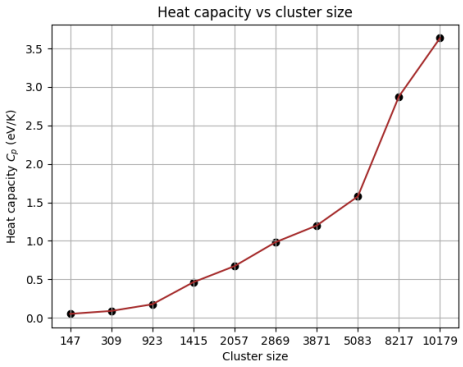
Figure 5.4: Energy vs temperature plot for different cluster sizes. Relaxation time is kept $\tau = 2000fs$, timestep is $dt = 2fs$

These plots 5.4 also show that small clusters have lesser stability in terms of heat capacity and thus with small heat, it reaches to melting point. We observed this keeping the heating rate τ constant, just by varying the heat source ΔQ. Then, the variation of heat capacity (C_p) is observed,

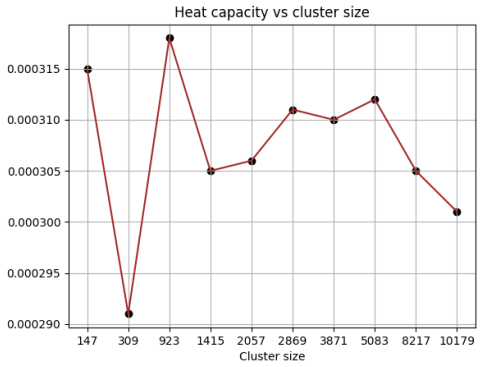
We have calculated the heat capacity 5.5 using finite difference between two time frames. After adding heat, we have determined the energy and temperature for second iteration and again store the data in third iteration. Then we compute the difference between these two energies and plot it with the number of atoms.

Finally, we want to see how the latent heat varies with cluster in the fig 5.6. We simulate the same atoms and observe the latent energy from the plots shown earlier.

The results of latent heat and heat capacity gives us a good idea about the fact that these

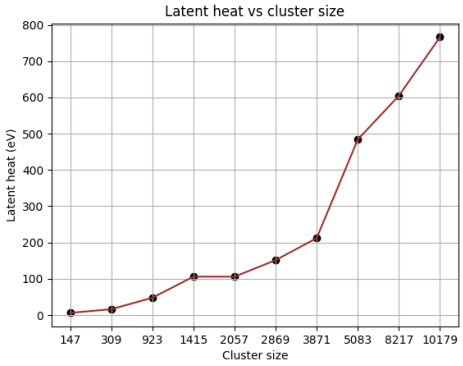


(a) Cluster C_p

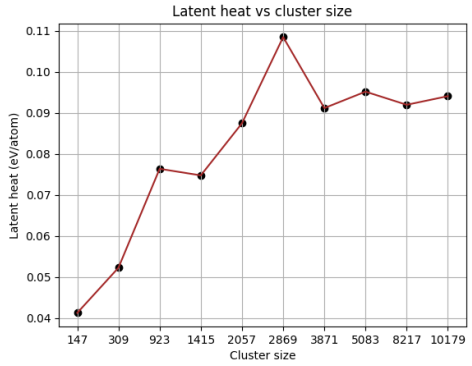


(b) C_p per atom

Figure 5.5: This plot shows how C_p varies with cluster confinements in linear and logarithmic scale for 147,309,923,2879,3871,5083,8271,10179 atoms.



(a) Cluster latent heat



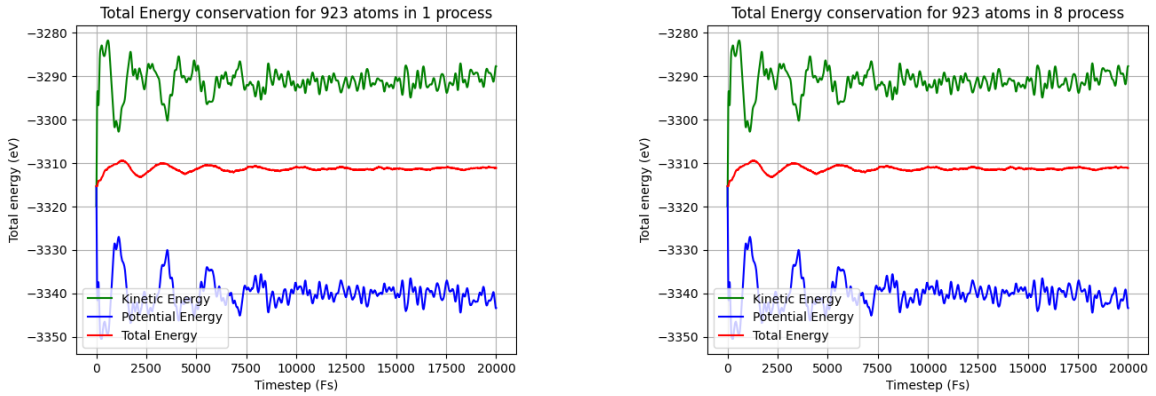
(b) Latent heat per atom

Figure 5.6: This plot shows how latent heat varies with cluster confinements in linear scale for 147,309,923,2879,3871,5083,8271,10179 atoms.

are intrinsic properties and thus, vary with the mass of the system. So, we plot for both properties for the complete cluster as well as per atom. Per atom shows us approximately same values where as this when multiplied with the number of atoms show a gradual increase in magnitude.

5.3.4 Energy simulation with MPI

Calculating energy in serialized version is inefficient when it comes in handling bulk atoms because one processor iterates stuff one by one. Thus, we follow the algorithm 0 to calculate the total energy using 4 processes. Now we are interested in testing if our MPI code is running properly i.e. we are not loosing any atom or if our total energy is being conserved if no additional heat is given to the system. For that, we ran a simulation with the algorithm mentioned above and plot the energy vs time which is shown below 5.7,



(a) 923 atoms, 1 process

(b) 923 atoms, 8 processes

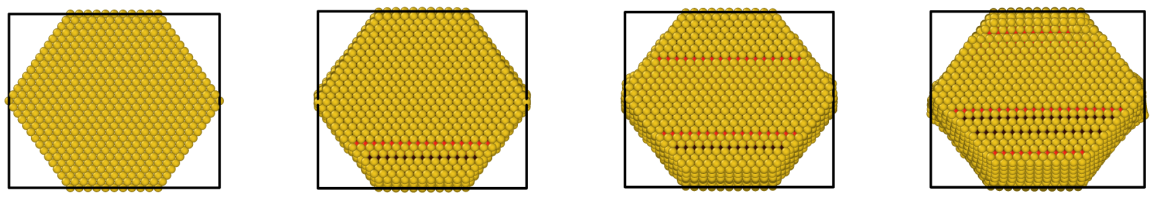
Figure 5.7: A comparative study of total energy vs time in serialized vs parallel processes for 10000 timesteps with $\Delta t = 2fs$. Kinetic energies are plotted with an offset of -3320 eV.

5.4 Nanowire simulation

Our atomic cluster is bounded with a simulation domain after we implement the MPI module. Here, we discuss about the theories that help us find the stress and strain of an atomic cluster. We enclose the cluster with periodic boundary conditions and let's put small deformation to the system. Thus we can write use ghost atoms to calculate the stress by the conservation of moment of inertia.

$$\begin{aligned} I &= \Sigma_i m_i r_i^2 \\ \frac{1}{2}\frac{dI}{dt} &= \Sigma_i m_i \vec{v}_i \cdot \vec{r}_i \\ \frac{1}{2}\frac{d^2I}{dt^2} &= 2E_{kin} + \Sigma_i \vec{f}_i \cdot \vec{r}_i = 0 \end{aligned} \tag{5.4}$$

Now, we use ghost atoms to calculate the total force onto the system due to the deformation caused externally. This conservation of moment of inertia gives us the idea that we can just calculate the net force acting on the system. Now, our domain is excluded from ghost atoms. So, net force can look like $\vec{f}_i = \vec{f}_i^{domain} + \vec{f}_i^{G}$ and thus can be rewritten from 5.4 as $\Sigma_i \vec{f}_i^{G} \cdot \vec{r}_i = -2E_{kin} - \Sigma_i \vec{f}_i^{domain} \cdot \vec{r}_i$. Now, we perform the simulation for small and large whiskers using MPI to calculate the forces of the ghost atoms and plot the results below. We provide nanowire force versus strain at varying temperature plots and some snapshots of how the defects look like. This is visible by using Common Neighbor Analysis (CNA).[1]



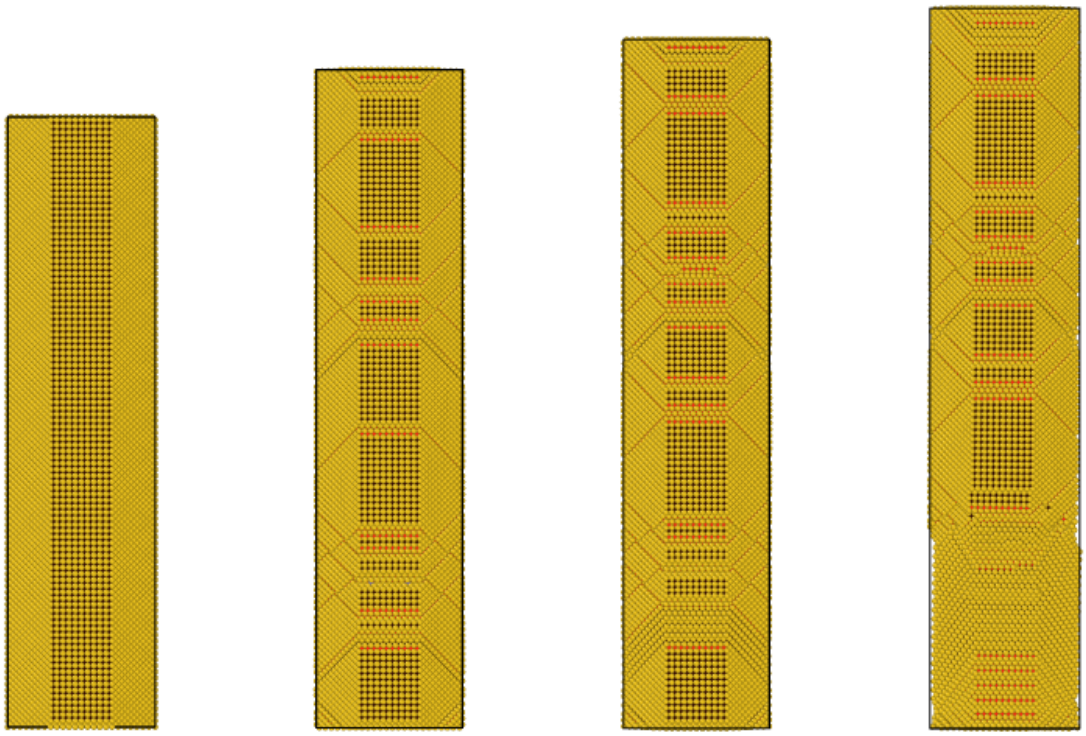
(a) At $t = 0fs$

(b) At $t = 600Kfs$

(c) At $t = 1000Kfs$

(d) At $t = 1600Kfs$

Figure 5.8: We present the snapshots of the top view of the large nanowire with 51500 atoms and calculated at temperature = $0K$. The red color denotes the HCP lattice, brown being the FCC and yellow denotes the other atoms.



(a) At $t = 0Kfs$ (b) At $t = 600Kfs$ (c) At $t = 1000Kfs$ (d) At $t = 1600Kfs$

Figure 5.9: We present the snapshots of the front view of the large Nanowire with 51500 atoms and calculated at temperature = $0K$. The red color denotes the HCP lattice, brown being the FCC and yellow denotes the other atoms.

From the plot below 5.10, we see that the elasticity modulus matches with the order cited by the paper [13]. Also, we see that till 0.2 strain we have the linear slope (yield point) and the material fracture starts from 0.14 strain. Here, we first allow the system to reconfigure at 0K for 20k fs and then we impart scaling operation at varied dimensions that decides the strain rate of the system. The approx area considered is 1600 $Å^2$.

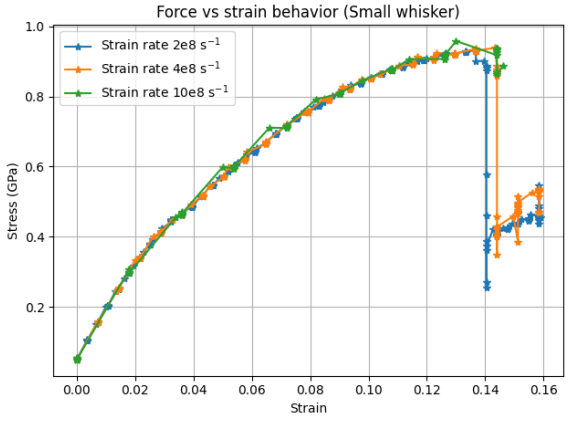


Figure 5.10: The plot presents stress vs strain plots at different strain rates - $2 \cdot 10^8 sec^{-1}, 4 \cdot 10^8 sec^{-1}, 10 \cdot 10^8 sec^{-1}$ for 3050 atoms gold nanowire

6

Conclusion

This report mainly talks about the development of molecular dynamics simulation using C++ code how the theories of physics can be implemented and visualized writing scientific codes. Molecular dynamics as described in the introduction section relates to particle physics and is the best way to study science at such low-scale level. Thus, we routed our report and project through describing the fundamentals of this simulation by categorizing this into its' importance, applications, limitations and methodologies. We augment the reasoning of crucial parameters and how this simulation strategy has minimal errors.

Next chapter includes theoretical concepts of our simulation and the physics that we have used in our method. This also paves the path in expanding continuous velocity profile into discrete points by the grace of Taylor's expansion - Verlet integration. Then we talk about one of the most important parameters - the potential and make a comparative analysis between lennard jones and embedded atom method potential, their ingredients and formulation. We also discuss about the thermostat and how we use this to equilibrate a system.

In the following chapter, we discuss about the implementation techniques, algorithms, and different test strategies of small functions like Verlet algorithm, thermostat implementation, timestep selection, potential test etc. This section gives a fair idea about how the codes are implemented in the project. We also discuss about the measurement units and how parallel computing using message passing interface (MPI) has been incorporated.

In the results section, we present all the graphical plots and snapshots of the molecular clusters generated by simulation code. Here, we first present if our simulation is conserving energy for a small molecular cluster. We study them by matching the RDF functions and present some snaps to prove the same. Then we see how the simulation time varies with cluster size and cut-off radius. Then, we discuss on the energy analysis and perform several simulations for kinetic energy and potential energy with time, total energy with temperature. We also extract the heat capacity, melting point and latent heat from energy vs temperature plot for different clusters and verify the property type i.e. extrinsic/intrinsic. Then we perform the MPI simulation and check if our energy is conserved after implementing the parallel computing with 8 cores and executing the periodic boundary conditions. Finally, we take two nanowires and study the deformation at varying strain rates. We also present a stress strain curve to show the elastic range and if this matches with the expected results.

This project is guided by the simulation department at Albert ludwigs university of Freiburg and some portion of code is being generated by Prof. Lars Pastewka from this university. The project is made available on Github repository

Bibliography

[1] Lucas Frérot Lars Pastewka, Wolfram Nöhring. Molecular dynamics, 2021.

[2] Richard P Feynman, Robert B Leighton, and Matthew Sands. The feynman lectures on physics; vol. i. *American Journal of Physics*, 33(9):750–752, 1965.

[3] Kun Zhou and Bo Liu. *Molecular Dynamics Simulation: Fundamentals and Applications*. Academic Press, 2022.

[4] Q Spreiter and M Walter. Classical molecular dynamics simulation with the velocity verlet algorithm at strong external magnetic fields. *Journal of Computational Physics*, 152(1):102–119, 1999.

[5] Nikola Tchipev, Steffen Seckler, Matthias Heinen, Jadran Vrabec, Fabio Gratl, Martin Horsch, Martin Bernreuther, Colin W Glass, Christoph Niethammer, Nicolay Hammer, et al. Twetris: Twenty trillion-atom simulation. *The International Journal of High Performance Computing Applications*, 33(5):838–854, 2019.

[6] RV Mantri, R Sanghvi, et al. Solubility of pharmaceutical solids. In *Developing Solid Oral Dosage Forms*, pages 3–22. Elsevier, 2017.

[7] Joshua A Rackers and Jay W Ponder. Classical pauli repulsion: An anisotropic, atomic multipole model. *The Journal of chemical physics*, 150(8):084104, 2019.

[8] LibreTexts libraries. Lennard-jones potential.

[9] MW Finnis and JE Sinclair. A simple empirical n-body potential for transition metals. *Philosophical Magazine A*, 50(1):45–55, 1984.

[10] V Vitek. Pair potentials in atomistic computer simulations. *MRS Bulletin*, 21(2):20–23, 1996.

[11] Fabrizio Cleri and Vittorio Rosato. Tight-binding potentials for transition metals and alloys. *Physical Review B*, 48(1):22, 1993.

[12] JC Ruiz Gómez and L Rincon. Melting of intermediate-sized gold nanoclusters. *Revista Mexicana de Física*, 53(7):208–211, 2007.

[13] Andreas Sedlmayr, Erik Bitzek, Daniel S Gianola, Gunther Richter, Reiner Mönig, and Oliver Kraft. Existence of two twinning-mediated plastic deformation modes in au nanowhiskers. *Acta Materialia*, 60(9):3985–3993, 2012.