Molecular Dynamics with C++

Final Report

Hari Narayan

January 20, 2023

Contents

1

Introduction

Molecular dynamics simulations are used to analyze the state of atoms and molecules and their influence on other atoms and molecules over a period of time. This is done to make inferences, better understand a system and to identify methods to favourably influence the system. The scope of a molecular dynamics simulation system ranges from material science [1] to biological and chemical systems [2]. They usually involve simulating one of the three thermodynamic ensembles, namely

- microcanonical or constant NVE ensemble
- canonical or constant NVT ensemble
- isothermal-isobaric or constant NPT ensemble

Where N stands for number of atoms, V stands for the system volume, E stands for total energy of system, T stands for system temperature and P stands for system pressure.

In this project, NVE and NVT simulations are done. Lennard Jones and Gupta energy models are used to represent inter-atomic potential energy. Berendsen thermostat and velocity rescaling methods are used to manage temperature control. To minimize computation cost, a neighbour list using a cutoff range is used. Gold clusters are periodically heated up to produce their so called heating curves. Towards the end, an NVT simulation of a gold nanowire made of different dimensions, stretched at different temperatures are simulated and their respective force-strain curves are plotted.

2

Theory

2.1 The Velocity-Verlet algorithm

This algorithm is used to calculate the trajectory of the atoms at each time-step. Considering newton's second law $f = ma$ and a Taylor expansion of the positions of atoms, the next position of the atoms after a time-step Δt is calculated using the following equations 2.1 and 2.2.

$$\vec{v}_i(t + \Delta t) = \vec{v}_i(t) + \frac{1}{2m_i}\left(\vec{f}_i(t) + \vec{f}_i(t + \Delta t)\right)\Delta t \tag{2.1}$$

$$\vec{r}_i(t + \Delta t) = \vec{r}_i(t) + \vec{v}_i(t)\Delta t + \frac{1}{2m_i}\vec{f}_i(t)\Delta t^2 \tag{2.2}$$

Where, m_i, $\vec{r_i}$, $\vec{v_i}$ and $\vec{f}_i$ are the masses, positions, velocities and forces respectively applied on them in 3D space.

2.2 Lennard-Jones (LJ) Potential

The simplest form of pair potential energy can be generally written as equation 2.3.

$$E_{\text{pot}}\left(\{\vec{r}_i\}\right) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} V\left(r_{ij}\right) = \sum_{i<j} V(r_{ij}) \tag{2.3}$$

Where, $r_{ij} = |\vec{r_j} - \vec{r_j}|$ is the distance between two atoms i and j. The force acting on atom k is derived as the negative gradient of energy as seen in equation 2.4 .

$$\vec{f_k} = -\frac{\partial E_{\text{pot}}}{\partial \vec{r}_k} = -\frac{1}{2}\sum_{ij}\frac{\partial V}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial \vec{r}_k} = \sum_{i}\frac{\partial V}{\partial r_{ik}}\hat{r}_{ik} \tag{2.4}$$

Where, $\hat{r}_{ik} = \vec{r}_{ik}/r_{ik}$. One version of such a simplified model is the Lennard-Jones [3]. It that has been studied extensively and approximates inter-molecular and inter-atomic forces well for fluid phases and roughly well for solid phases. This potential is given by the equation 2.5.

$$E_{pot} = \frac{1}{2}\sum_{ij} 4\varepsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{2.5}$$

Where ε and σ are variables dependent on the substance being modelled. The force acting on atom k due to this potential is given by the equation 2.6.

$$\vec{f_k} = \sum_{ik} 4\varepsilon\left(12\frac{\sigma^{12}}{r_{ik}^{13}} - 6\frac{\sigma^{6}}{r_{ik}^{7}}\right)\hat{r_{ik}} \tag{2.6}$$

2.3 Embedded Atom Model (EAM) and Gupta Potential

EAM is a family of functions that is used to represent an approximation of the interatomic potential energy in a system particularly for metallic systems. These models contain an attractive and repulsive components in them and take the functional form as seen in equation 2.7.

$$E_{\text{pot}}(\{\vec{r}_i\}) = \sum_i \mathcal{F}(\rho_i) + \frac{1}{2}\sum_{i,j}\phi(r_{ij}) \tag{2.7}$$

Where $\mathcal{F}(\rho_i)$ is an approximated embedding function that defines the relationship between energy and local electron density ρ_i for an atom at position $\vec{r_i}$. The repulsive component $\phi(r_{ij})$ depends on the distance between two atoms r_{ij}. The forces from EAM are derived as seen in equation 2.8 by finding the negative gradient as done for the Lennard-Jones potential.

$$\vec{f_k} = \sum_i \left(\frac{\partial \mathcal{F}(\rho_k)}{\partial \rho_k} + \frac{\partial \mathcal{F}(\rho_i)}{\partial \rho_i}\right)\frac{\partial f}{\partial r_{ik}}\hat{r}_{ik} + \sum_i \frac{\partial V}{\partial r_{ik}}\hat{r}_{ik} \tag{2.8}$$

In this project, Gupta potential [4] is used. They have the repulsive component E_R^i and attractive component E_b^i as shown in the equations 2.9. They are combined together as shown in equation 2.10 to form gupta potential.

$$E_R^i = \sum_j A_{\alpha\beta} e^{-P_{\alpha\beta}(r_{ij}/r_o^{\alpha\beta}-1)} \qquad E_B^i = \{\sum_j \xi_{\alpha\beta}^2 e^{-2q_{\alpha\beta}(r_{ij}/r_o^{\alpha\beta}-1)}\}^{1/2} \tag{2.9}$$

$$E_c = \sum_i (E_R^i + E_B^i) \tag{2.10}$$

The default parameters used by this function are Cleri & Rosato's [5] parameterization for gold.

2.4 Temperature control techniques

The instantaneous temperature of a system is calculated using the kinetic energy and boltzmann constant relation [6] as seen in equation 2.11.

$$\frac{3}{2}Nk_BT = \sum_i \frac{1}{2}mv_i^2 \tag{2.11}$$

This relation is also made use of to modify the temperature of a system by multiplying a scaling factor λ with the velocity of the atoms. In this project, the following two methods are made use of to calculate λ,

2.4.1 Berendsen thermostat

This is a weak coupling method that modifies and sustains the temperature of a system thought the simulation. The velocity scaling factor is calculated as seen in equation 2.12.

$$\lambda = \sqrt{1 + \left(\frac{T_0}{T} - 1\right)\frac{\Delta t}{\tau}} \tag{2.12}$$

Where T_0 and T are the target temperature and system temperature respectively. The relaxation constant τ is used to determine the strength of coupling wherein lower values mean stronger coupling and higher values mean weaker coupling between the thermostat and the system.

2.4.2 Velocity Scaling

This is a strong coupling method that instantly changes the temperature of the system. This is applied once after every few timesteps to allow the system to equilibrate after scaling. It is useful for the purpose of heating and cooling but not to sustain a temperature since this is inconsistent with the statistical mechanics of an NVT ensemble. The velocity scaling factor is calculated as seen in equation 2.13.

$$\begin{aligned} \Delta Q &= \mathrm{E}'_k - \mathrm{E}_k = (\lambda^2 - 1)\mathrm{E}_k \\ \lambda &= \sqrt{\frac{\Delta Q}{\mathrm{E}_k} + 1} \end{aligned} \tag{2.13}$$

Where E'_k is the desired kinetic energy, E_k is the current kinetic energy of the system and ΔQ is the amount of energy that is to be added to the system.

2.5 Time-step Calculation

It is necessary to use the right time-step for trajectory calculation to ensure accuracy of output from the simulated system. If this is not done right, it would cause the system to get unstable. This can be visualized by the loss of energy when attempting to run an NVE simulation on the atoms. This inaccuracy is caused because the trajectory calculation algorithm fails to interpolate the natural oscillating frequency of the atom. A demonstration of this is shown in the results section of this paper.

Hence considering *Nyquist's theorem* [7] , a time-step lesser than the oscillating frequency of the atoms of the substance being simulated by at least a factor of 2 should be chosen. The upper bound to the time-step can be calculated using the bond frequency of the atoms in consideration. The bond frequency for H-H [8] and Au-Au [9] bonds are 4342 cm^{-1} and 90 cm^{-1}. By converting them to time-period in femtoseconds (fs) using [10] the oscillating frequency is 8 fs and 371 fs respectively, less than half of which is an acceptable time-step. But in this project, a time-step of 2 fs is used to maximize on accuracy [11].

2.6 Unit Calculation

The units used in this project for distance between atoms, atomic mass, energy (in terms of charge of an electron), temperature and time are Armstrong (Å), grams/mol (g/mol), electron Volts (eV), Kelvin and femtoseconds (fs) respectively. But to use this unit for time, a constant needs to be multiplied with one of these parameters. Since the mass is constant throughout the simulation, a scaling factor is derived for the mass. To do this, the dimension of energy ($\mathrm{M}^1\mathrm{L}^2\mathrm{T}^{-2}$) and the existence of proportionality between SI units are used in the following way.

$$\mathrm{E} = \mathrm{M}^1\mathrm{L}^2\mathrm{T}^{-2} \qquad \mathrm{M} = \mathrm{E}^1\mathrm{T}^2\mathrm{L}^{-2} \tag{2.14}$$

The the value of mol and SI units conversion for energy and atomic mass are the following

$$1\mathrm{eV} = 1.6 \cdot 10^{-19}\mathrm{J} \qquad 1\text{Å} = 10^{-10}m \qquad 1\mathrm{fs} = 10^{-15}\mathrm{s} \qquad 1\mathrm{kg} = 10^3 \times 6 \cdot 10^{23}\mathrm{g/mol} \tag{2.15}$$

The SI unit values in 2.15 for energy, distance and time are substituted in the derived dimensions of mass in equatin 2.14 to receive a value in kg. This kg is converted to g/mol by using the conversion in 2.15. This way, a scaling factor of 103.6 is arrived at that should be multiplied with the mass for the simulation to run with a time-step of 1 fs.

2.7 Heating Curve

As seen in figure 2.1 It represents the relationship *total energy* and *temperature* and serves the purpose of characterising a substance. The plateaus are caused due to energy being used towards phase change over temperature increase. in this

- Heat Capacity (eV/K) - It is the quantity of energy that will increase the temperature of the object by one kelvin.
- Melting Point (K) - It is the point at which the temperature of the molecule starts to plateau after the solid phase while adding in more energy.
- Latent Heat (eV) - It is the amount of energy used towards phase change after melting point until the end of the plateau.

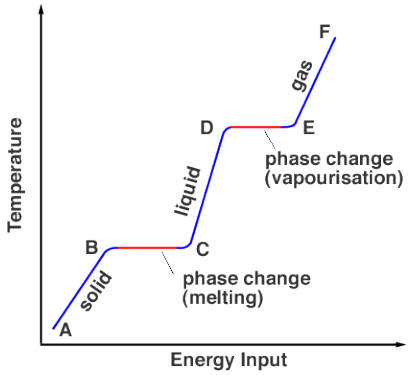


Figure 2.1: Heating curve [12]

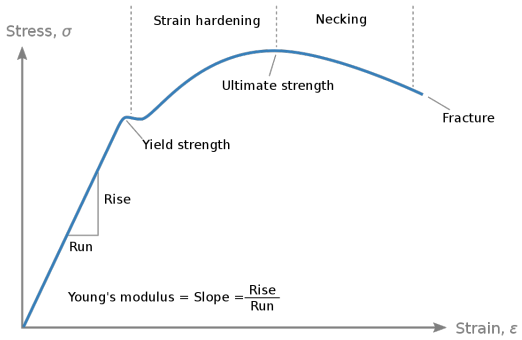
In this project, the heat curve is generated up to the liquid phase.

2.8 Stress-Strain Graph

This graph is used to characterize a material based on deformation produced per unit of force applied on it. The stress and strain are calculated as shown in the set of equations in 2.16

$$stress(\sigma) = \frac{F}{A} \qquad strain(\varepsilon) = \frac{\Delta L}{L} \tag{2.16}$$

The variables σ and ε are not to be confused with the variables of the Lennard-Jones potential.



(a) stress-strain graph [13]

In this project, the force is calculated for the gold nanowire simulation. This is done for different dimensions under different temperatures and strain conditions. From this force, the Force-strain graph is plotted.

2.9 Domain decomposition

This process involves dividing the simulation space into 3D sub-spaces for the purpose of assigning individual CPU cores to process atoms for each subspace. This is handled by the MPI library. When atoms leave the sub-space to a neighbouring sub-space, their calculations get handled by the CPU core of that sub-space. To calculate forces on the atoms at the boundary, these atoms need access to the atoms at the adjacent sub-space. For this purpose a set of so called "ghost atoms" are made accessible to the boundary atoms upto twice the cut-off radius used to calculate the neighbor lists. Twice the sum of all forces along the ghost atoms on one side of the domain is used to represent the force on the nano-whisker.

3

Project Structure

3.1 Directory Structure

The individual simulation modules have their own declaration and definition files as well as the simulation functions. The main `.cpp` files to control the simulations from is present in the `code/` directory. The simulation functions files and the main files are divided into mpi and non-mpi related functions. They are differentiated by the suffix of their file names as `_serial` for non-mpi related simulations and `_parallel` for mpi related simulations.

- `build` - Holds the executable for the tests, and simulations. Also contains a specific version for the eigen library present in "`build/external/`" that's compatible with this project.
- `cmake-skeleton` - Skeleton project provided by the lecture [14] that contains the necessary cmake files. The code present in the "`code/`" directory was built upon it.
- `code` - Holds the main C++ code including the test code in it.
- `data` - Holds the simulation input files and the scripts used to generate the simulation input files.
- `results` - Holds the simulation output files (`.csv` and `.xyz` files), jupyter-notebook to generate plots and the plots.

3.2 Simulation Modules

The following modules used in this project are provided by the lecture [14]

- The directories "`code/header/`" and "`code/src/`" hold `xyz`, `neighbors`, `gupta` and `domain` header and source files respectively. Additionally under "`code/header/`", the `mpi_support.h` file was also provided.
- The directory "`code/tests/`" hold, `test_gupta.cpp`, `test_ljDS.cpp` and `test_neighbors.cpp`.
- The directory "`data/defaults/`" holds the provided `.xyz` files.
- The directory "`data/`" holds `ih.C` and `vector.h` which are modified versions of code from page[1].

[1] http://www.pas.rochester.edu/~wangyt/algorithms/ih/

4

Implementation

Throughout all the simulations, a record is made of the loop, total energy, potential energy, kinetic energy and system temperature and towards the end, simulation variables in `.csv` files. The structure of the simulation code called from the `main_serial.cpp` and `main_parallel.cpp` is the following

```
unsigned long long int m#(struct& V) {
    COUT /* status: location of files stored */
    for(auto var : V.varList) {
        CreateCSV; /* With the filename dependent on variables from V */
        CreateXYZ; /* With the filename dependent on variables from V */
        // start_clock
        for(unsigned long long int loop{0}; loop < V.loops; ++loop) {
            if(condition) COUT /* status: number of loops done */
            verletStep1(arguments);
            CALC /* forces on atoms and potential_energy */
            verletStep2(arguments);
            // temperature control function here
            if(condition) {
                CALC     /* kinetic, total energy and temperature */
                WriteCSV /* loop total_E potential_E kinetic_E temperature */
                WriteXYZ /* positions and velocities */
            }
        }
        // end_clock
        COUT     /* seconds(end_clock - start_clock) */
        WriteCSV /* variables used in simulation */
        CloseCSV;
        CloseXYZ;
    }
    COUT /* status: total execution time of function */
    return /* seconds(end_clock - start_clock) */
}
```

Where the `#` in `m#` represents the milestone number from the lecture that the simulation function is relevant to.

4.1 Atoms

The "atoms `struct`" `code/header/atoms.h` is used for ease of manipulation of atom properties. The constructor of the `struct` has been implemented using various combination of parameters. It has not been displayed in this report to minimize verbosity.

4.2 Velocity Verlet

This is used in the predictor-corrector scheme from 2.1 and 2.2 to 4.1 and 4.2.

$$\vec{v}_i(t + \Delta t/2) = \vec{v}_i(t) + \frac{1}{2m_i}\vec{f}_i(t)\Delta t \tag{4.1}$$

$$\vec{r}_i(t + \Delta t) = \vec{r}_i(t) + \vec{v}_i(t + \Delta t/2)\Delta t \tag{4.2}$$

4.1 and 4.2 is used in the following manner in `src/verlet.cpp`

```
void verletStep1( Atoms& A, const double dt) {
    A.velocities += 0.5 * A.forces * dt / A.mass;
    A.positions += A.velocities * dt;
}
void verletStep2( Atoms& A, const double dt) {
    A.velocities += 0.5 * A.forces * dt / A.mass;
}
```

This is tested in the `tests/tests_verlet.cpp` by comparing the output of the above functions with 2.1 and 2.2 for the same random input.

4.3 Kinetic Energy

The following calculates per-atom and total kinetic energy in `code/src/kineticAndTemp.cpp`.

```
double getKineticE(Atoms &A) {
    A.kin_energies.setZero();
    for (int i{0}; i < A.velocities.cols(); ++i) {
        for (int j{0}; j < 3; ++j)
            A.kin_energies(i) +=  pow(A.velocities(j,i), 2);
        A.kin_energies(i) *= A.mass * 0.5;
    }
    return A.kin_energies.sum();
}
```

4.4 System Temperature

The following function definitions are derived from 2.11.

```
double getSystemT(Atoms &A) {
    return (getKineticE(A) / (A.nb_atoms() * 8.617333262e-5 * 3/2));
}

double getSystemT(const double KE, const unsigned int nb) {
    return (KE / (nb * 8.617333262e-5 * 3/2));
}
```

4.5 Lennard Jones Potential

The following calculates per-atom forces from 2.6, per-atom and total potential energy 2.5 in `code/src/lj.cpp`. Commenting and uncommenting the appropriate lines would change the following code to use direct summation (DS).

```
// double ljDS(Atoms &A, const double epsilon, const double sigma) {    // for DS
double lj(Atoms &A, NeighborList &N, const double epsilon, const double sigma) {
    A.pot_energies.setZero();
    A.forces.setZero();

    // for (int i{0}; i < A.nb_atoms(); ++i) {      // for DS
        // for (int j{0}; j < A.nb_atoms(); ++j) {      // for DS
    for (auto[i, j]: N) {       // for neighbor list
        Eigen::Array3d dist_vectr{A.positions.col(i) - A.positions.col(j)};
        double dist{sqrt((dist_vectr * dist_vectr).sum())};

        // if(dist == 0) continue;      // for DS

        double derivative{4 * epsilon * (12 * pow(sigma, 12) / pow(dist, 13) -
                        (6 * pow(sigma, 6) / pow(dist, 7)))};

        for(int k{0}; k < 3; ++k)
            A.forces(k, i) +=  derivative * dist_vectr(k) / dist;

        A.pot_energies(i) += 2 * epsilon * (pow((sigma / dist), 12) -
                        pow((sigma / dist), 6));
    } // }      // for DS
    return A.pot_energies.sum();
}
```

4.6 Temperature Control

4.6.1 Berendsen Thermostat

This is implemented using 2.12 and is present in `code/src/bthermostat.cpp`.

```
void bThermostat(Atoms &A, const double targetT, const double dt,
                            const double tau) {
    A.velocities *= sqrt(1 + ((targetT/getSystemT(A)) - 1) * (dt/tau));
}
```

This is tested in `tests/test_bThermostat.cpp`, by comparing the temperature increase and decrease rate for 3 different values of tau τ. The expected behaviour is that system temperature reaches the target temperature faster for lower values of τ.

4.6.2 velocity scaling

This is implemented using 2.13 This is used in the milestone function directly and is run at every nth time step.

```
if(loop%relax_time == 0) atoms.velocities *= ((deltaQ/getKineticE(atoms)) + 1);
```

5

Results

5.1 Lennard-Jones Potential

The LJ potential energy characteristic curve is demonstrated in figure 5.1 by logging the potential energy energy in LJ units while increasing the distance between two atoms.

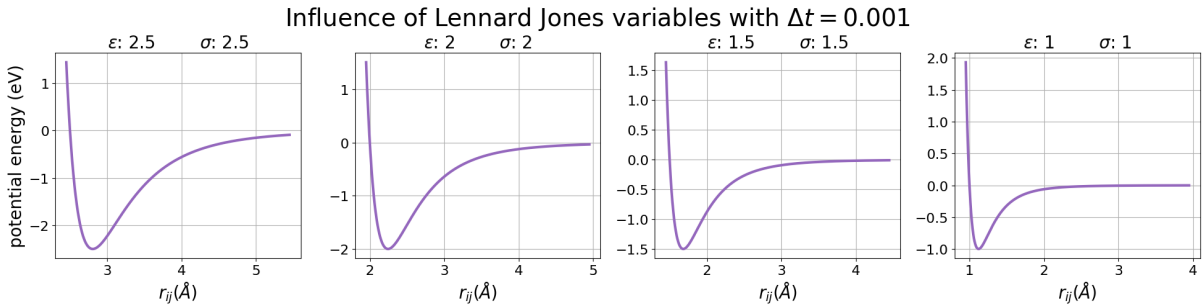


Figure 5.1: lennard jones potential energy characteristic curve

As seen in the figure 5.1, for distances lesser than σ, the potential energy tends towards positive infinity as their distance decreases indicating a strong repulsion. For distances greater than σ, an attractive force is generated with ε LJ units of potential energy.

5.1.1 Influence of time-step

A molecule with 54 atoms that is in equilibrium at time-step $\Delta t = 0.01$ present in `data/defaults/lj54.xyz` is simulated and their positions are visualized as seen in figure 5.2.

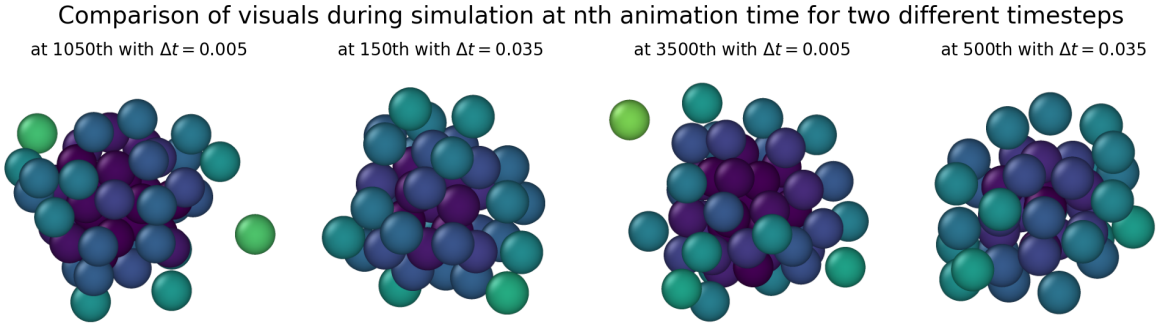


Figure 5.2: simulation of `lj54.xyz`

This simulation is run for different Δt values and their energies are plotted in figure 5.3.

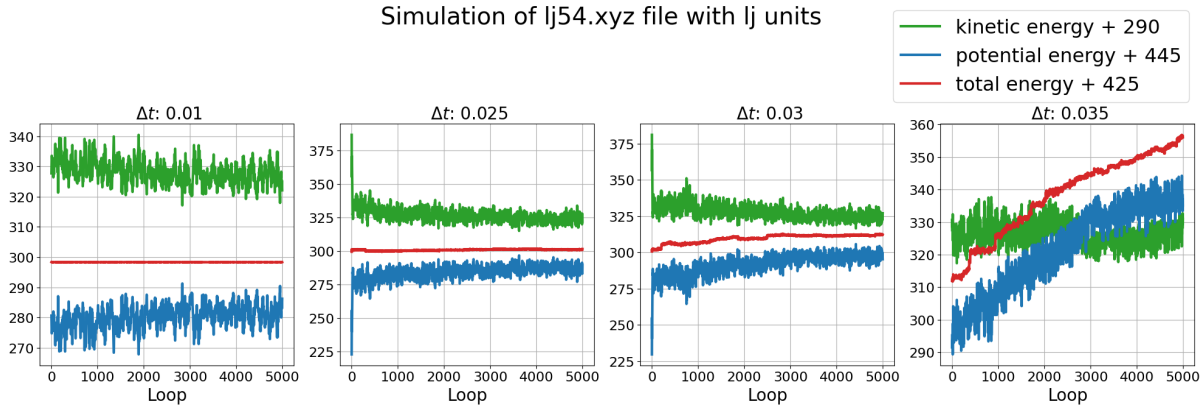


Figure 5.3: stability of `lj54.xyz` for different Δt values

As elaborated in the theory section of this report under the time-step calculation section, it can be seen in figure 5.3, that the total energy is no longer stable for higher values of Δt.

This instability can also be observed in the positions of atoms as shown in figure 5.2. The molecule in the 1^{st} and 3^{rd} positions of figure 5.2 represent a stable molecule. While the molecule in the 2^{nd} and 4^{th} positions of the figure represent an unstable molecule. The 1^{st} and 2^{nd} sub-figure represent the same point in time. The 3^{rd} and 4^{th} sub-figure also represent the same point in time (POT) of simulation where, POT is the relation $\text{POT} = \text{Loop} * \Delta t$.

The animation time in figure 5.2 represents the animation time shown in OVITO. The lighter colors of atoms indicate larger relative distance from other atoms in the system. From this it can be understood that the atoms are more relatively closely packed as time passes in the stable molecule representative and are more loosely packed in the unstable molecule representative as time passes. This is consistent with the increase in total energy as can be seen in figure 5.1. The simulation of the molecule under stable conditions is called an NVE simulation.

5.2 Neighbour List Performance

The time taken by a simulation cycle running an implementation of a neighbour list is demonstrated by the use of different molecules containing an increasing number of atoms that are arranged in the form of a cubic lattice.

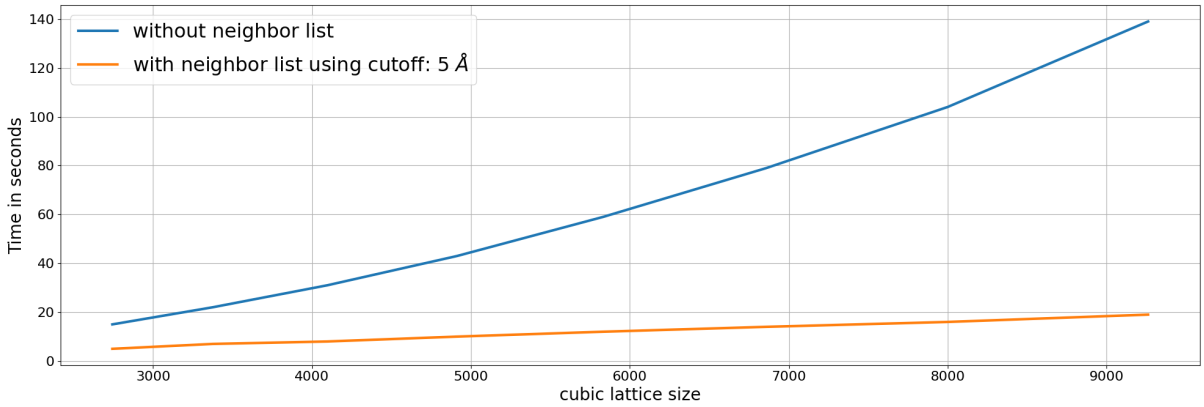


Figure 5.4: Simulation time reduced with the use of a neighbour list

As seen in figure 5.4, the time taken for the simulation to complete with a neighbour list implementation is much lesser compared to the time taken without the use of a neighbour list. This the case as the former does not calculate the influence of atoms beyond 5Å on an atom since their influences are assumed to be insignificant. These simulation were done using the LJ variables $\varepsilon = 1$ and $\sigma = 1$. The molecules used can be found under the directory `data/cubes/`.

5.3 Heating of Icosahedral Gold Cluster

This is done to generate a heating graph as elaborated in the theory section of this report. Velocity scaling is done periodically to heat up the cluster and allow the molecule to equilibriate.

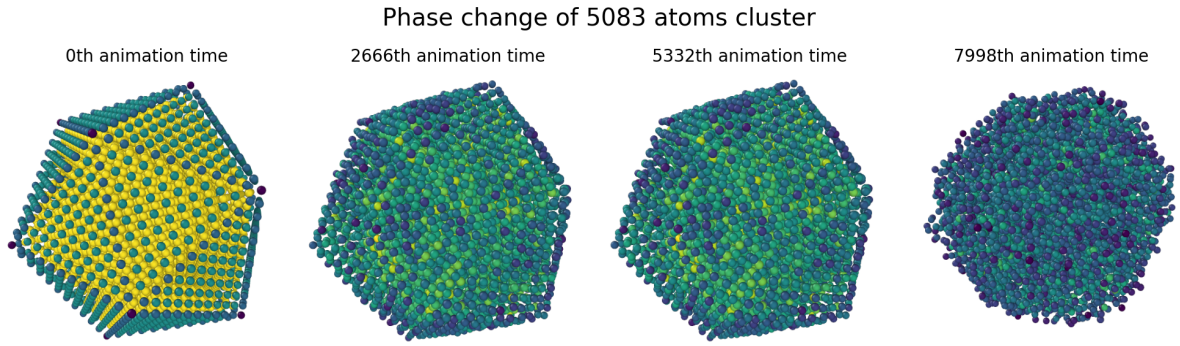


Figure 5.5: snapshots from simulation during conversion solid to molten stage

As time passes, it can be seen in figure 5.5 that the solid icosahedral structure looses form indicating a phase change from solid to molten state. The animation time is used to represent the frame in OVITO. The simulation was run for each cluster present in the directory `data/clusters/` for 10000 loops with each loop representing a $\Delta t = 2fs$. The lighter colors of atoms indicate lesser relative distance from other atoms in the system in figure 5.5.

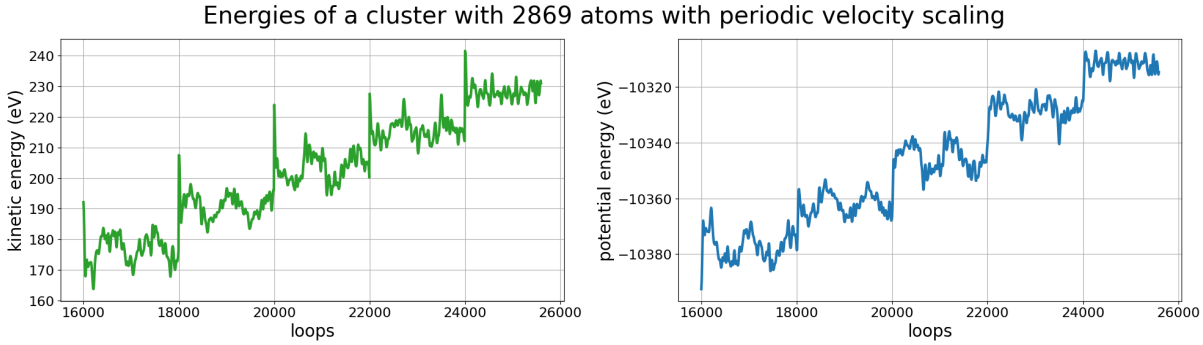


Figure 5.6: Adding a defined amount of ΔQ every 4000 fs

The influence of periodically introducing ΔQ in the cluster on the potential and kinetic energy can be observed in the figure 5.6 in the form of impulses. It can also be observed that with every impulse, the potential energy gets more unstable.

At some point, the potential energy increases while the kinetic energy remains relatively stable, indicative of the molecule being present in a phase changing state. This phase change is more easily observable on a graph.

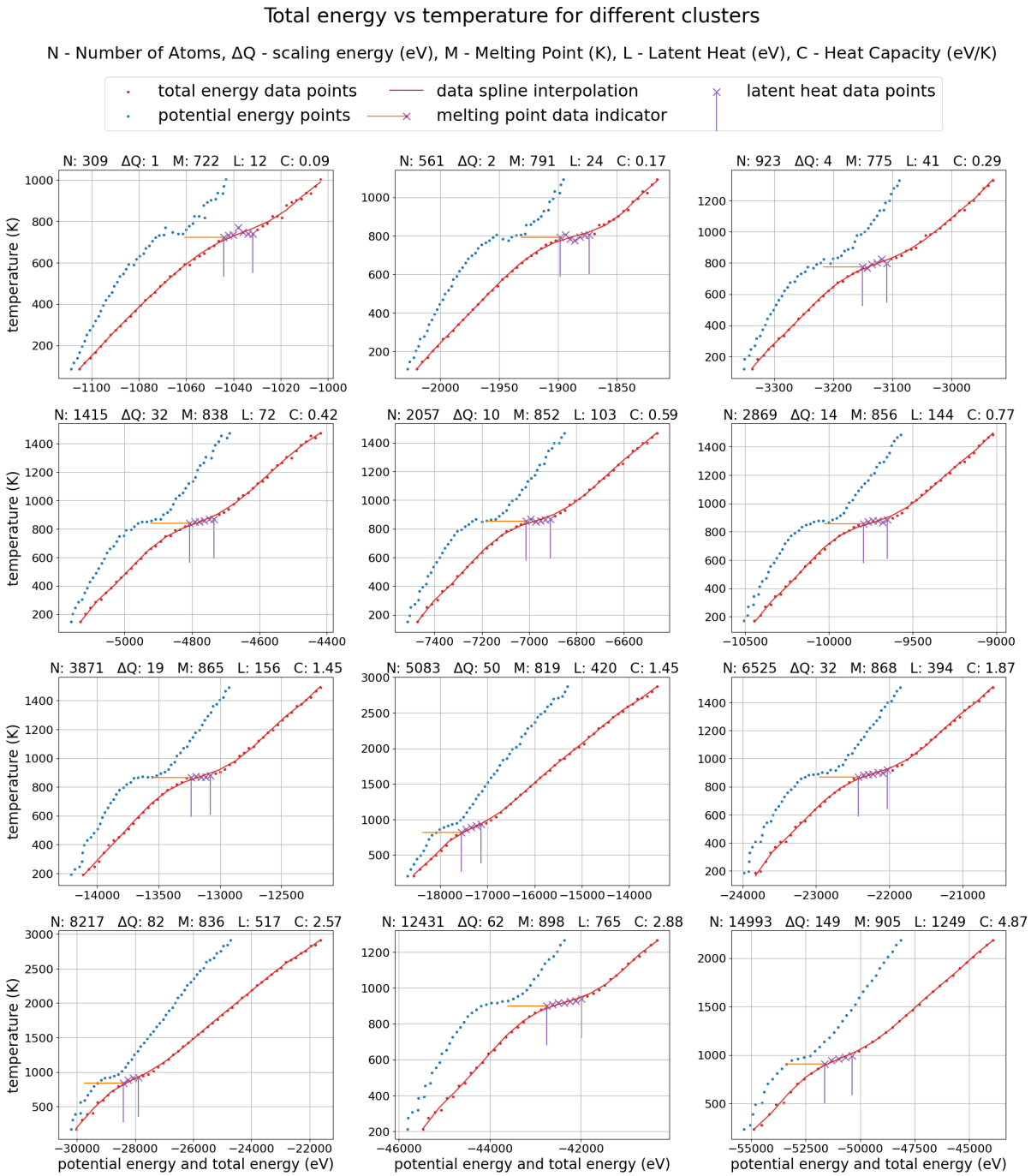


Figure 5.7: Heating graph for cluster of different sizes

As observable in figure 5.7, the difference in slope of the total energy is used to indicate a phase change of the cluster. Each data point plotted is the average of the the final 400 loop values just before the next velocity scaling takes place. A line is fit to these data points to obtain a relatively noise free heating graph. From this, the melting point, latent heat and heat capacity can be identified for each cluster and their dependence of the cluster size can be plotted.

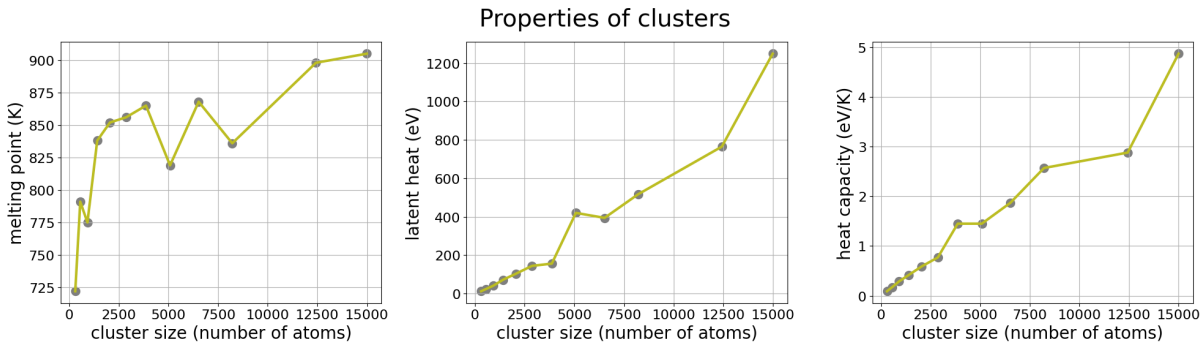


Figure 5.8: cluster properties

As observable in figure 5.8, the dependence of melting point, latent heat and heat capacity can be identified.

5.4 Parallelization

To validate the parallelized version of the code with its serial representative, an NVE simulation is run on a cluster and the energies between them are compared.

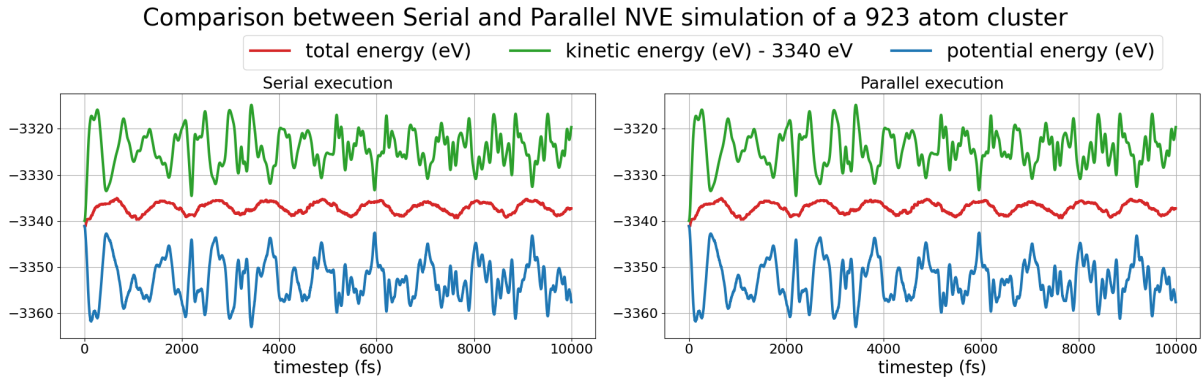


Figure 5.9: parallelization comparison

From figure 5.9, it can be confirmed that the parallel execution of the simulation is similar to the serial execution of the simulation.

5.5 Stretching of gold whisker

This is done to generate a force-strain graph for different types of whiskers. Three different whiskers of different dimensions are stretched at different strain rates and temperatures. The temperatures being 0 kelvin, 3 kelvin and 5 kelvin. The strain rates being 1%, 1.5% and 1.8%. These whiskers are present in `data/whiskers/`.

whisker number	dimensions		
	x-axis	y-axis	z-axis
1	14.43	20.40	57.70
2	23.08	32.64	57.70
3	28.85	40.80	57.70

Different combinations of these parameters are used to run the NVT simulation on the whisker for 1000000 fs with a strain that is applied every 40000 fs. While the temperature of the system is controlled using a Berendsen thermostat.

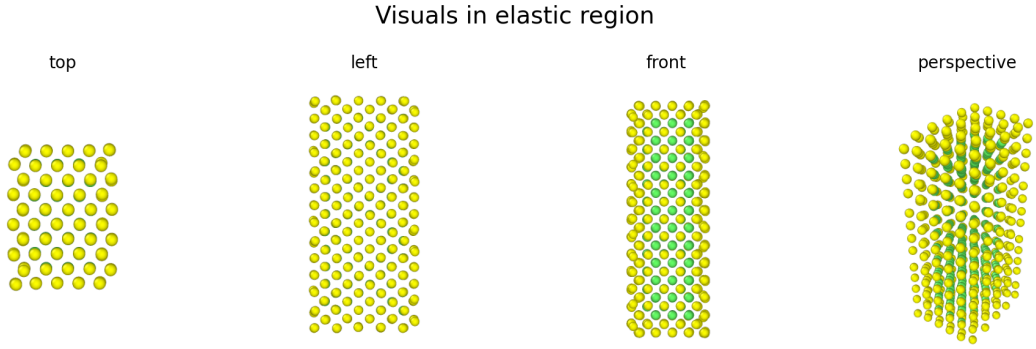


Figure 5.10: before yield strength point with 40.9% FCC

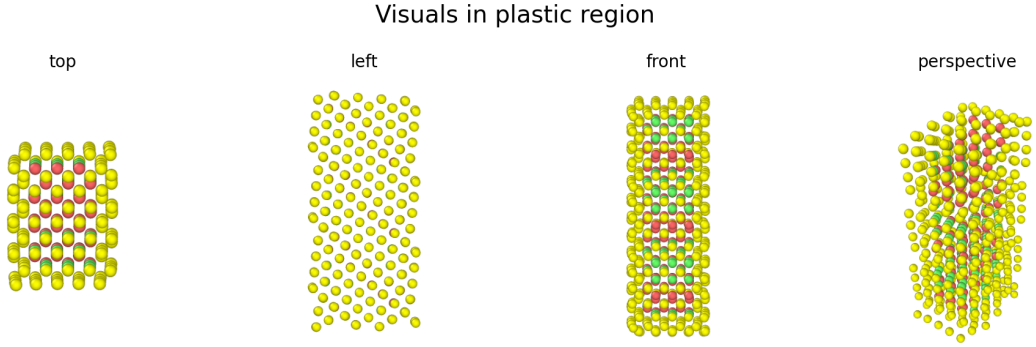


Figure 5.11: after yield strength point with 20.9% FCC and 18.2% HCP

The figures 5.10 and 5.11 are the positions of atoms just before the yield strength point and just after the yield strength point for whisker 1 at 5K temperature with 1% strain rate every relaxation time period. The red atoms represent HCP (Hexagonal Close Packed) atoms and the green atoms represent FCC (Face Cubic Centered) atoms.

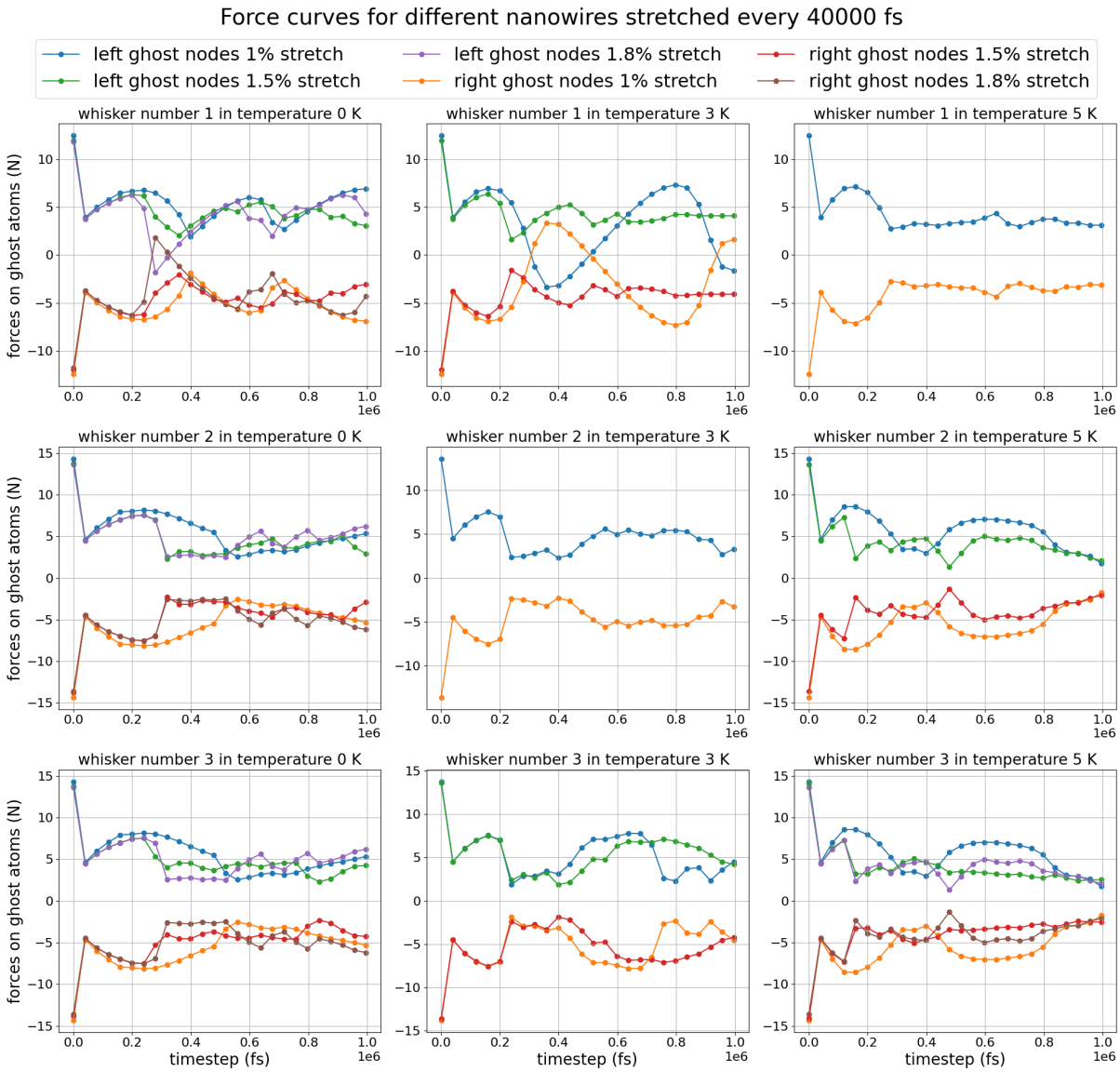


Figure 5.12: Force on ghost atoms of whiskers under different conditions

As seen in figure 5.12, the elastic deformation range is characterized as the initial curve in the graph followed by the bumpy region representing the plastic region.

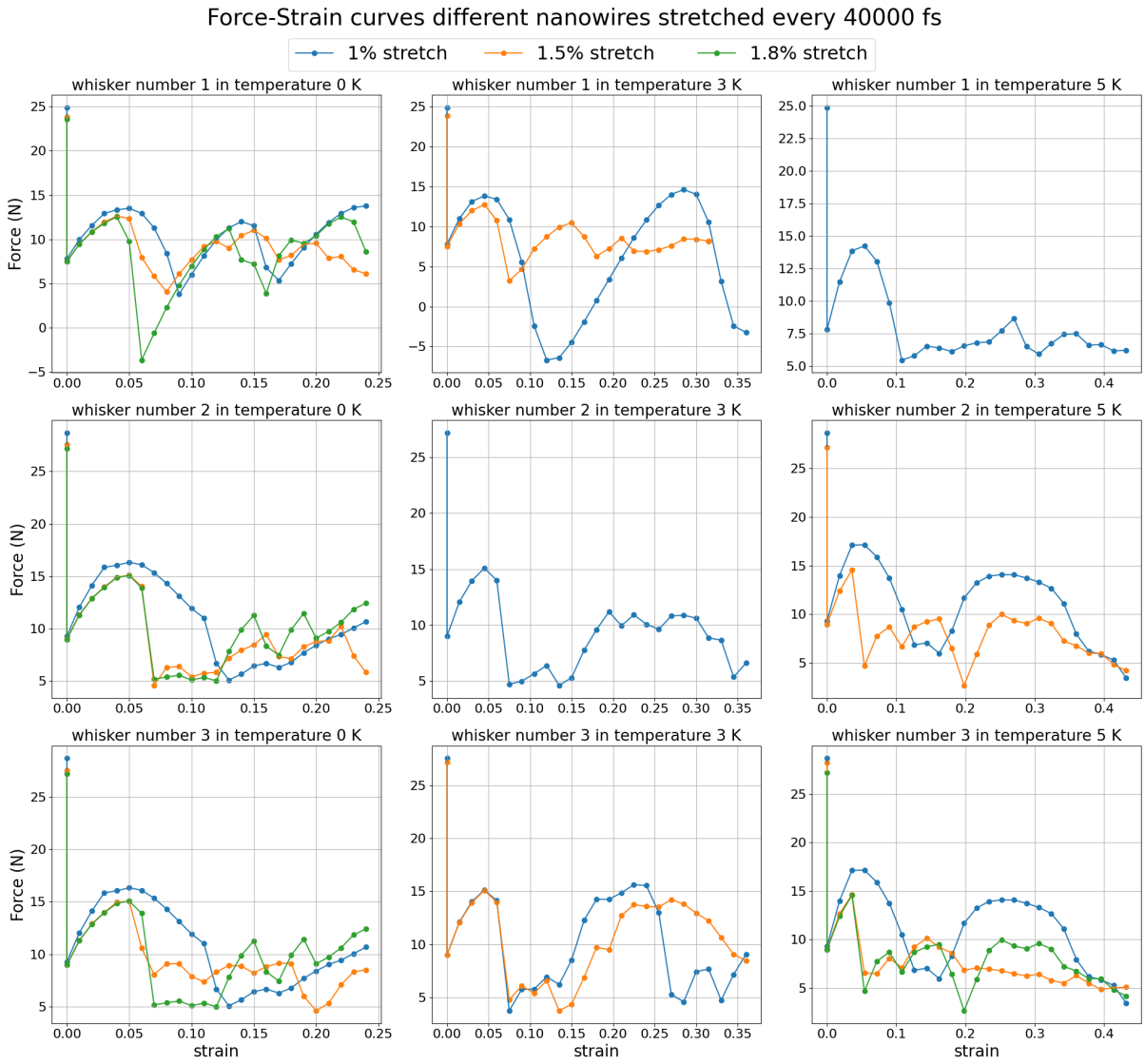


Figure 5.13: Force-strain curves

The force values in figure 5.13 are obtained by summing the absolute values of left and right ghost atoms or twice the absolute values of the ghost atoms on one of the sides. The latter is also valid because of newton's 3rd law.

6

Conclusions

By reading the contents of this paper and the code, one can understand the aspects to implement NVE and NVT ensembles simulation using C++. A demonstration of the individual components functioning was done and their results were shown. A simple simulation of atom clusters using lennard-jones potential was done. Using gupta potential the following simulations were done. A "heating curve" was made for different sized simulated gold structures and the "force strain graph" was made for different sized simulated gold nanowires of varying dimensions and under varying temperature conditions.

Bibliography

[1] Liang Sun, Yu-Xing Zhou, Xu-Dong Wang, Yu-Han Chen, Volker L. Deringer, Riccardo Mazzarello, and Wei Zhang. Ab initio molecular dynamics and materials design for embedded phase-change memory. *npj Computational Materials*, 7(1), February 2021.

[2] Josep Gelpi, Adam Hospital, Ramón Goñi, and Modesto Orozco. Molecular dynamics simulations: advances and applications. *Advances and Applications in Bioinformatics and Chemistry*, page 37, November 2015.

[3] Lennard jones potential. `https://en.wikipedia.org/wiki/Lennard-Jones_potential`. Accessed: 2023-01-19.

[4] Raju P. Gupta. Lattice relaxation at a metal surface. *Physical Review B*, 23(12):6265–6270, June 1981.

[5] Fabrizio Cleri and Vittorio Rosato. Tight-binding potentials for transition metals and alloys. *Physical Review B*, 48(1):22–33, July 1993.

[6] Boltzmann constant. `https://www.chemie-schule.de/KnowHow/Boltzmannkonstante`. Accessed: 2023-01-19.

[7] Nyquist-shannon sampling theorem. `https://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem#Introduction`. Accessed: 2023-01-19.

[8] Hydrogen bond frequency. `https://www.chem.purdue.edu/gchelp/vibs/h2.html`. Accessed: 2023-01-19.

[9] Hikaru Kuramochi, Satoshi Takeuchi, Munetaka Iwamura, Koichi Nozaki, and Tahei Tahara. Tracking photoinduced au–au bond formation through transient terahertz vibrations observed by femtosecond time-domain raman spectroscopy. *Journal of the American Chemical Society*, 141(49):19296–19303, November 2019.

[10] unit conversions. `https://sherwingroup.itst.ucsb.edu/internal/unit-conversion/`. Accessed: 2023-01-19.

[11] Molecular dynamics timestep. `https://en.wikibooks.org/wiki/Molecular_Simulation/Molecular_Dynamics#Time_step`. Accessed: 2023-01-19.

[12] Heating curve. `http://www.splung.com/content/sid/6/page/latentheat`. Accessed: 2023-01-19.

[13] Stress-strain graph. `https://en.wikipedia.org/wiki/Stress%E2%80%93strain_curve`. Accessed: 2023-01-19.

[14] Lars Pastewka and Lucas Frérot. Lecture notes of "molecular dynamics with c++", 2022.