Molecular Dynamics with C++ report

Eslam Salah Zaki Elshiekh

5051183

University of Freiburg

Faculty of Engineering

Department of Microsystems Engineering

September 30$^{\text{th}}$, 2022

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Molecular dynamics simulations are becoming an essential part of modern technology, because they allow to study the behavior of complex systems in a controlled way. The simulation of molecular systems is a very complex task, because the number of atoms in a system can be very large. Therefore, the simulation of a system requires a lot of computational power. In the last years, the computational power of computers has increased dramatically. This development has led to the fact that molecular dynamics simulations are now possible for systems with millions of atoms. However, the simulation of such systems is still a very demanding task. Therefore, the simulation of such systems is usually done on supercomputers or clusters of computers.

In this report, we will discuss the design and Implementation of a molecular dynamics simulations for atoms and molecules. We will also discuss the different potential forces that are used in the simulations, and how they are implemented, initializing atomic systems in random positions and preserve the atoms from evaporating using Thermostates, how to make the simulation run faster by using only neighbor list search, and how to parallelize the simulation using MPI.

2 Methods

Here we will discuss the different methods that we will use in our simulation.

2.1 Velocity-Verlet integrator

The velocity-Verlet algorithm is a numerical method for solving the equations of motion of a system of particles. It is a symplectic integrator, which means that it conserves the total energy of the system. The algorithm is based on the Taylor expansion of the position and velocity of the particles. The algorithm is as follows:

1. Calculate the acceleration of the particles at time t.

2. Calculate the velocity of the particles at time $t + \frac{1}{2}\Delta t$.

3. Calculate the position of the particles at time $t + \Delta t$.

4. Calculate the acceleration of the particles at time $t + \Delta t$.

5. Calculate the velocity of the particles at time $t + \Delta t$.

2.2 Lennard-Jones potential force

The Lennard-Jones potential is a potential that is used to model the interaction between atoms and molecules. It is a potential that is used in molecular dynamics simulations. LJ potential is a function of the distance between two particles. The LJ potential is given by the following equation:

$$V(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right] \tag{1}$$

where ϵ is the depth of the potential well, σ is the distance at which the potential is zero, and r is the distance between the two atoms. The force is given by the following equation:

$$F(r) = -\frac{dV(r)}{dr} = 24\epsilon \left[\frac{2}{r} \left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right] \tag{2}$$

The force is calculated by using the following steps:

1. Calculate the distance between all pairs of atoms.

2. Calculate the potential between all pairs of atoms.

3. Calculate the force between all pairs of atoms.

2.3 Thermostat

The thermostat is an algorithm that is used to preserve the atoms from evaporating. The idea is when we initialize the atoms in random positions, the atoms will try to get to the equilibrium position. This means that the atoms will move very fast and will collide with each other and the temperature of the system will increase very fast. This will lead to the atoms evaporation. The thermostat is used to prevent this from happening. The thermostat is used to slow down the atoms and make them move

slower until they reach the equilibrium position and the temperature of the system is stable. The thermostat is based on the idea that the kinetic energy of the system is conserved. This means that if we change the velocity of the particles, then we must change the velocity of the other particles in the system in such a way that the total kinetic energy of the system is conserved. The algorithm is as follows:

2.4 Neighbor List

The neighbor list is an algorithm that is used to speed up the calculation of the forces that act on the particles. It is based on the idea that the forces between particles are only dependent on the distance between them. This means that if the distance between two particles is larger than a certain threshold value, then the force between them is zero. Therefore, we can calculate the forces between particles only if the distance between them is smaller than the threshold value. The algorithm is as follows:

1. Calculate the distance between all particles.

2. If the distance between two particles is smaller than the threshold value, then calculate the force between them.

3. If the distance between two particles is larger than the threshold value, then do not calculate the force between them.

2.5 Embedded-atom method potential force

The embedded-atom method (EAM) is a method for calculating the forces between atoms. It is based on the idea that the forces between atoms are dependent on the

density of the atoms. EAM is a potential force that is used in the simulation of metals, alloys, and intermetallic compounds. The EAM potential force is as follows:

2.6 Parallelization

The parallelization of the simulation is done using MPI. The algorithm is as follows:

1. Initialize the MPI environment.
2. Get the number of processes.
3. Get the rank of the process.
4. Initialize the atoms.
5. Calculate the forces.
6. Calculate the total energy.
7. Calculate the total momentum.
8. Calculate the temperature.
9. Calculate the pressure.
10. Print the results.
11. Finalize the MPI environment.

3 Implementation

3.1 Velocity-Verlet integrator

3.1.1 Test strategy for Verlet integrator

We test the Verlet integrator by comparing the results of the Verlet integrator with the results of the analytical solution of the equations of motion of a particle. We use the following equations of motion for a particle:

$$\frac{d^2x}{dt^2} = -\frac{k}{m}x \tag{3}$$

$$\frac{d^2v}{dt^2} = -\frac{k}{m}v \tag{4}$$

where x is the position of the particle, v is the velocity of the particle, m is the mass of the particle, and k is the spring constant. We use the following initial conditions:

$$x(0) = 1 \tag{5}$$

$$v(0) = 0 \tag{6}$$

$$x'(0) = 0 \tag{7}$$

$$v'(0) = 1 \tag{8}$$

where x' is the velocity of the particle, and v' is the acceleration of the particle. We use the following values for the mass of the particle and the spring constant:

$$m = 1 \tag{9}$$

$$k = 1 \tag{10}$$

We use the following values for the time step:

$$\Delta t = 0.01 \tag{11}$$

We use the following values for the number of steps:

$$N = 1000 \tag{12}$$

We use the following values for the initial time:

$$t_0 = 0 \tag{13}$$

We use the following values for the final time:

$$t_f = 10 \tag{14}$$

We use the following values for the initial position:

$$x_0 = 1 \tag{15}$$

We use the following values for the initial velocity:

$$v_0 = 0 \tag{16}$$

We use the following values for the initial acceleration:

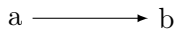

Figure 1: Use tikz to draw nice graphs!

3.2 Lennard-Jones potential force

3.2.1 Derivation of the analytical expression for the forces of the Lennard-Jones potential

3.3 Berendsen thermostat

3.3.1 Test strategy for Berendsen thermostat

3.4 Embedded atom method

3.5 units and specification of the time unit

3.5.1 time step for the gold potential

3.6 Neighbor List

3.7 Parallelization using MPI

Explain the math and introduce notation.

Algorithm 1 Stochastic Gradient Descent: Neural Network

Create a mini batch of m samples $\mathbf{x}_0 \dots \mathbf{x}_{m-1}$
foreach sample $\mathbf{x}$ **do**
  $\mathbf{a}^{\mathbf{x},0} \leftarrow \mathbf{x}$ ▷ Set input activation
  foreach Layer $l \in \{1 \dots L-1\}$ **do** ▷ Forward pass
    $\mathbf{z}^{\mathbf{x},l} \leftarrow \mathbf{W}^l \mathbf{a}^{\mathbf{x},l-1} + \mathbf{b}^l$
    $\mathbf{a}^{\mathbf{x},l} \leftarrow \varphi(\mathbf{z}^{\mathbf{x},l})$
  end for
  $\boldsymbol{\delta}^{\mathbf{x},L} \leftarrow \nabla_{\mathbf{a}} C_{\mathbf{x}} \odot \varphi'(\mathbf{z}^{\mathbf{x},L})$ ▷ Compute error
  foreach Layer $l \in L-1, L-2 \dots 2$ **do** ▷ Backpropagate error
    $\boldsymbol{\delta}^{\mathbf{x},l} \leftarrow ((\mathbf{W}^{l+1})^T \boldsymbol{\delta}^{\mathbf{x},l+1}) \odot \varphi'(\mathbf{z}^{\mathbf{x},l})$
  end for
end for
foreach $l \in L, L-1 \dots 2$ **do** ▷ ▷ Gradient descent
  $\mathbf{W}^l \leftarrow \mathbf{W}^l - \frac{\eta}{m} \sum_{\mathbf{x}} \boldsymbol{\delta}^{\mathbf{x},l} (\mathbf{a}^{\mathbf{x},l-1})^T$
  $\mathbf{b}^l \leftarrow \mathbf{b}^l - \frac{\eta}{m} \sum_{\mathbf{x}} \boldsymbol{\delta}^{\mathbf{x},l}$
end for

4 Results

4.1 Total energy as a function of time for different time steps

4.2 LJ sequence of snapshots (no more than 5) from your simulation

4.3 simulation time as a function of the size (number of atoms) without neighbor list

4.4 simulation time as a function of the size (number of atoms) with neighbor list

4.5 total energy vs temperature

4.6 melting point versus cluster size

4.7 heat capacity versus cluster size

4.8 latent heat versus cluster size

4.9 Energy conservation with MPI parallelization

4.10 Nanowire defects

5 Conclusion

Bibliography