# NVT ensemble for LJ-fluid with Nosé-Hoover Chain

Final Project : Topics in Advanced Computational Modeling

Ho In Jung
*Department of Computational Science & Technology*
*Seoul National University*
Seoul, Korea
hoyin@snu.ac.kr

***Abstract*—To compare the simulation results with the experimental data obtained at a constant temperature, we need to attach some numerical apparatus regulating temperature to our original molecular dynamics simulation to have a canonical ensemble. The simulation results from constant energy, Nosé-Hoover chain NVT indicate this method can evolve the system to equilibrium at a comparable rate.The velocity probability distribution of a system consisting of one free particle obtained using the NHC method agrees with the exact distribution (Maxwell-Boltzmann distribution) very well.**

## I. Introduction

The Molecular Dynamics technique is a scheme for studying the natural time evolution of a classical system of $N$ particles in volume $V$. In such simulations, the total energy $E$ is a constant of motion. It would seem that it is impossible to perform MD simulations in ensembles at first sight, but one approach is completely dynamical in origin. It is based on a reformulation of the Lagrangian equations of motion of the system. This method, first introduced by Andersen in constant-pressure MD simulations, has become one of the most important tricks to extend the applicability of MD simulations. In the Andersen approach to isothermal Molecular Dynamics simulation, constant temperature is achieved by stochastic collisions with a heat bath. Nosé has shown that one also can perform deterministic Molecular Dynamics at a constant temperature. This approach is based on the innovative use of an extended Lagrangian, which contains additional artificial coordinates and velocities. For constant-temperature MD simulations, it is now more common to use the Nosé scheme in Hoover's formulation, so-called Nosé-Hoover thermostat.

The equations of motion in the Nosé-Hoover scheme cannot be derived from a Hamiltonian. This situation implies that one cannot use the standard methods to connect the dynamics generated by solving these equations of motions with Statistical Mechanics, so-called non-Hamiltonian dynamics. In this case, the conventional Nosé-Hoover algorithm only generates the correct distribution if there is a single constant motion. Typically, the total energy is always conserved. Furthermore, this implies that one should not have any other conserved quantity. In most conventional simulations, this is the case if the momentum is not conserved, for example, if there is an external force ; i.e., the sum of the forces $\sum_i \mathbf{F}_i \neq 0$ . Suppose we simulate a system without external forces, $\sum_i \mathbf{F}_i = 0$. which implies we have an additional conservation law. In that case, the Nosé-Hoover scheme is still correct provided that the center of mass remains fixed. This condition can be fulfilled easily if we ensure that during the equilibration, the velocity of the center of mass is set to 0. If we simulate a system using no external field and in which the center of mass is not fixed or if we have more than one conservation law, we have to use Nosé-hoover chains to obtain the correct canonical distribution as introduced in Section2.

To alleviate the restriction for the Nosé-Hoover thermostat, Martyna *et al.* proposed a scheme in which the Nosé-Hoover thermostat is coupled to another thermostat or, if necessary, to a whole chain of thermostats which take into account additional conservation laws. This generalization of the original Nosé-Hoover method still generates a canonical distribution. In this article, we show how the Nosé-Hoover chains are coupled and how the chains are used in Python code. After executing a few exercises of the Nosé-Hoover chain, we discuss the result of simulations from a thermodynamic point of view.

## II. Background

### *A. Constant Temperature*

We can impose a temperature on a system by bringing it into thermal contact with a large heat bath. Under those conditions, the probability of finding the system in a given energy state is given by the Maxwell-Boltzmann velocity distribution.

$$f_v(v_i) = \sqrt{\frac{m}{2\pi kT}} \exp\left[\frac{-mv_i^2}{2kT}\right] \tag{1}$$

And Maxwell-Boltzmann speed distribution.

$$f(v) = 4\pi \left(\frac{m}{2\pi kT}\right)^{3/2} v^2 \exp\left[\frac{-mv^2}{2kT}\right] \tag{2}$$

while *k*=1, *m*=1 in reduced unit. Temperature is a macroscopic quantity. Microscopically it is less well defined due to the low number of particles. However, if we use the kinetic energy

Identify applicable funding agency here. If none, delete this.

of the parameters, we can calculate the temperature. We will use this in order to scale the velocities to maintain a constant temperature.

$$E_K = \frac{1}{2}mv^2 \tag{3}$$

$$k_B T = \frac{2}{3}\sum_N E_K \tag{4}$$

### *B. Velocity Scaling*

Many methods have been developed to control the temperature of molecular dynamics (MD) simulations. A simple, but crude method fixes the system temperature to a desired value to avoid steady energy drifts caused by the accumulation of numerical errors during MD simulation, by rescaling the velocity of each atom at each timestep or every several timesteps by a factor of $\lambda = \sqrt{T_0/T(t)}$ where $T(t)$ is the current temperature as calculated from the kinetic energy and T0 is the desired temperature. One problem with this approach is that it does not allow fluctuations in temperature, which are present in the canonical ensemble. In this article, we use this method only for velocity initialization.

### *C. Lennard-Jones Potential*

From Lennard-Jones Potential, we can obtain force and potential energy from each coordinate.

$$U_{LJ}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{5}$$

$$\mathbf{F} = -\frac{1}{r}\nabla U_{LJ}(\mathbf{r}) = -\frac{1}{r}\frac{dU_{LJ}}{d\mathbf{r}} \tag{6}$$

$$= -24\left[2\left(\frac{\sigma}{\mathbf{r}}\right)^{14} - \left(\frac{\sigma}{\mathbf{r}}\right)^{8}\right] \tag{7}$$

And then, equation (5), (6) can be converted to MD unit.

$$U^*_{LJ}(r) = 4\left[\left(\frac{1}{r^*}\right)^{12} - \left(\frac{1}{r^*}\right)^{6}\right] \tag{8}$$

$$\mathbf{F}^* = 48\left[\left(\frac{1}{r^*}\right)^{14} - \frac{1}{2}\left(\frac{1}{r^*}\right)^{8}\right] \tag{9}$$

### *D. Virial Pressure*

The pressure is defined in terms of the virial expression. The virial pressure is commonly used to obtain the pressure from a general simulation. It is particularly well suited to molecular dynamics since forces are evaluated and readily available. For pair interactions, one has

$$p = \frac{k_B T N}{V} + \frac{1}{Vd}\overline{\sum_{i<j}\mathbf{f}_{ij}\mathbf{r}_{ij}} \tag{10}$$

$$p^* = \rho T + \frac{\text{(virial)}}{3V} \tag{11}$$

where $p$ is the pressure, $T$ is the temperature, $V$ is the volume, and $k_B$ is the Boltzmann constant. In this equation, one can recognize an ideal gas contribution and a second term due to the virial.

### *E. Dimesionless parameters*

Dimensionless reduced units can be defined based on the Lennard-Jones potential parameters, convenient for molecular simulations. By choosing our units, we can remove any constants and get a general behavior for all gases. Mass, sigma, epsilon, and the Boltzmann constant are set to equal one. Reduced coordinates are used for the other variables, which are derived from the parameters set to one.

The reduced units may seem complicated, but it allows all of the equations to be written very simply in the program. This also gives us physical insight - for example, the reduced temperature is the ratio of the thermal energy $k_B T$ to the energy of the intermolecular interactions $\epsilon$.

| Property | Symbol | Reduced form |
|---|---|---|
| Length | $r^*$ | $\frac{r}{\sigma}$ |
| Time | $t^*$ | $t\sqrt{\frac{\epsilon}{m\sigma^2}}$ |
| Temperature | $T^*$ | $\frac{k_B T}{\epsilon}$ |
| Force | $F^*$ | $\frac{F\sigma}{\epsilon}$ |
| Energy | $U^*$ | $\frac{U}{\epsilon}$ |
| Pressure | $p^*$ | $\frac{p\sigma^3}{\epsilon}$ |
| Density | $\rho^*$ | $\rho\sigma^3$ |

tab. I: Dimensionless Unit (MD Unit)

## III. Nosé-Hoover Chain

### *A. Introduction*

Nosé introduced an extended system method by coupling the system to a heat bath. In this method, an extra degree of freedom associated with the heat bath is included. The total Hamiltonian of the extended system is conserved, and the density distribution is microcanonical. However, the distribution function projected on the physical system is canonical. Nosé extended the method to include multiple thermostats. Hoover demonstrated that the equations of motion of Nosé dynamics could be derived from the Liouville theorem.

Martyna *et al.* modified the equations of motion by introducing a chain of thermostats and formulated a Nosé-Hoover chain (NHC) algorithm. It is shown that by doing so, one can sample the phase space more extensively. Moreover, Martyna *et al.* have applied the reversible integrator to NHC extended system methods and derived modified velocity-Verlet integrators.

The equations of motion defining the NHC dynamics were constructed using the conservation of the probability in phase space. For the constant-$T$ dynamics of M-chains of thermostats, the equations of motion of $N$ atoms in extended Hamiltonian form is a conserved quantity.

$$\mathcal{H}_{NHC} = \mathcal{H}(\mathbf{r},\mathbf{p}) + \sum_{k=1}^{M} \frac{p_{\xi_k}^2}{2Q_k} + Lk_bT\xi_1 + \sum_{k=2}^{M} k_BT\xi_k$$

Where the $r$'s and $p$'s are the positions and momenta, respectively, $M$ is the number of thermostats on the chains, $\xi$ and $v_\xi$ are the thermostat variable and it conjugated momentum, and $Q_i$ is the mass of the$_{ith}$ thermostat. $k$ is the Boltzmann constant, and $L$ is the number of degrees of freedom.

## *B. Reversible Integrator*

A time-reversible MD integrator can be generated using Trotter factorization of the Liouville propagator as demonstrated by Tuckerman *et al.*

For $M$ chains, the Nosé-Hoover equations of motion and the Liouville operator for the equation of motions are given by

$$\begin{aligned}
\dot{\mathbf{r}}_i &= \mathbf{p}_i/m_i \\
\dot{\mathbf{p}}_i &= \mathbf{F}_i - \frac{p_{\xi_1}}{Q_1}\mathbf{p}_i \\
\dot{\xi}_k &= \frac{p_{\xi_k}}{Q_k} \quad k = 1, ..., M \\
\dot{p}_{\xi_1} &= \left(\sum_i p_i^2/m_i - Lk_BT\right) - \frac{p_{\xi_2}}{Q_2}p_{\xi_1} \\
\dot{p}_{\xi_k} &= \left[\frac{p_{\xi_{k-1}}^2}{Q_{k-1}} - k_BT\right] - \frac{p_{\xi_{k+1}}}{Q_{k+1}}p_{\xi_k} \\
\dot{p}_{\xi_M} &= \left[\frac{p_{\xi_{M-1}}^2}{Q_{M-1}} - k_BT\right] \\
iL &\equiv \dot{\eta}\frac{\partial}{\partial\eta} \quad \text{with}\, \eta = (\mathbf{r}^N, \mathbf{P}^N, \xi^M, p_\xi^M)
\end{aligned}$$

Using the quations of motion, $\mathbf{p}_i = m_i\mathbf{v}_i$ and $p_{\xi_k} = Q_k v_{\xi_k}$, we obtain as Liouville operator for the Nosé-Hoover chains

$$\begin{aligned}
iL_{NHC} &= \sum_{i=1}^{N} \mathbf{v}_i \cdot \nabla_{\mathbf{r}_i} + \sum_{i=1}^{N} \left[\frac{\mathbf{F}_i(\mathbf{r}_i)}{m_i}\right] \nabla_{\mathbf{v}_i} \\
&- \sum_{i=1}^{N} v_{\xi_1}\mathbf{v}_i \cdot \nabla_{\mathbf{v}_i} + \sum_{k=1}^{M} v_{\xi_k}\frac{\partial}{\partial\xi_k} \\
&+ \sum_{k=1}^{M-1} (G_k - v_{\xi_k}v_{\xi_{k+1}})\frac{\partial}{\partial v_{\xi_k}} + G_M\frac{\partial}{\partial v_{\xi_M}}
\end{aligned}$$

with

$$\begin{aligned}
G_1 &= \frac{1}{Q_1}\left(\sum_{i=1}^{N} m_i\mathbf{v}_i^2 - Lk_BT\right) \\
G_k &= \frac{1}{Q_k}\left(Q_{k-1}v_{\xi_k}^2 - k_BT\right).
\end{aligned}$$

Here $iL$ is seperated into three parts that only involves the positions($iL_r$) and the velocities($iL_v$) from the parts that involve the Nosé-Hoover thermostat($iL_C$)

$$iL_{NHC} = iL_r + iL_v + iL_C$$

with

$$\begin{aligned}
iL_r &= \sum_{i=1}^{N} \mathbf{v}_i \cdot \nabla_{\mathbf{r}_i} \\
iL_v &= \sum_{i=1}^{N} \frac{\mathbf{F}_i(\mathbf{r}_i}{m_i}) \cdot \nabla_{\mathbf{v}_i} \\
iL_C &= \sum_{k=1}^{M} v_{\xi_k}\frac{\partial}{\partial\xi_k} - \sum_{i=1}^{N} v_{\xi_1}\mathbf{v}_i \cdot \nabla_{\mathbf{v}_i} \\
&+ \sum_{k=1}^{M-1} (G_k - v_{\xi_k}v_{\xi_{k+1}})\frac{\partial}{\partial v_{\xi_k}} + G_M\frac{\partial}{\partial v_{\xi_M}}.
\end{aligned}$$

Using Trotter factorization, the propagator can be written as

$$e^{iL\Delta t} = e^{iL_C\Delta t/2}e^{iL_v\Delta t/2}e^{iL_r\Delta t}e^{iL_v\Delta t/2}e^{iL_C\Delta t/2} + O(\Delta t^3)$$

Here, we will do this for a chain of length $M = 2$ , the Nosé-Hoover part of the Liouville operator for this chain length can be separated into five terms :

$$iL_C = iL_\xi + iL_{C_v} + iL_{G_1} + iL_{v\xi_1} + iL_{G_2},$$

where the terms are defined as

$$\begin{aligned}
iL_\xi &\equiv \sum_{k=1}^{2} v_{\xi_k}\frac{\partial}{\partial\xi_k} \\
iL_{C_v} &\equiv -\sum_{i=1}^{N} v_{\xi_1}\mathbf{v}_i \cdot \nabla_{\mathbf{v}_i} \\
iL_{G_1} &\equiv G_1\frac{\partial}{\partial v_{\xi_1}} \\
iL_{v\xi_1} &\equiv -(v_{\xi_1}, v_{\xi_2})\frac{\partial}{\partial v_{\xi_1}} \\
iL_{G_2} &\equiv G_2\frac{\partial}{\partial v_{\xi_2}}
\end{aligned}$$

The factorization for the Trotter equation that we use is

$$\begin{aligned}
e^{(iL_C\Delta t/2)} &= e^{(iL_{G_2}\Delta t/4)}e^{(iL_{v\xi_1}\Delta t/4 + iL_{G_1}\Delta t/4)}e^{(iL_\xi\Delta t/2)} \\
&\times e^{(iL_{C_v}\Delta t/2)}e^{(iL_{G_1}\Delta t/4 + iL_{v\xi_1}\Delta t/4)}e^{(iL_{G_2}\Delta t/4)} \\
&= e^{(iL_{G_2}\Delta t/4)}\left[e^{(iL_{v\xi_1}\Delta t/8}e^{(iL_{G_1}\Delta t/4}e^{(iL_{v\xi_1}\Delta t/8)}\right] \\
&\times e^{(iL_\xi\Delta t/2)}e^{(iL_{C_v}\Delta t/2)} \\
&\times \left[e^{(iL_{v\xi_1}\Delta t/8)}e^{(iL_{G_1}\Delta t/4)}e^{(iL_{v\xi_1}\Delta t/8)}\right] \\
&\times e^{(iL_{G_2}\Delta t/4)}
\end{aligned}$$

where

$$\begin{aligned}
e^{(iL_{G_2}\Delta t/4)}:&\quad v_{\xi_2} \to v_{\xi_2} + G_2\Delta t/4\\
e^{(iL_{v\xi_1}\Delta t/8)}:&\quad v_{\xi_1} \to \exp\left[-v_{\xi_2}\Delta t/8\right] v_{\xi_1}\\
e^{(iL_{G_1}\Delta t/4)}:&\quad v_{\xi_1} \to v_{\xi_1} + G_1\Delta t/4\\
e^{(iL_{\xi}\Delta t/2)}:&\quad \xi_1 \to \xi_1 - v_{\xi_1}\Delta t/2\\
&\quad \xi_2 \to \xi_2 - v_{\xi_2}\Delta t/2\\
e^{(iL_{C_v}\Delta t/2)}:&\quad v_i \to \exp[-v_{\xi_1}\Delta t/2]v_i.
\end{aligned}$$

Applying the Nosé-Hoover part of the Liouville operator changes $\xi_k$, $v_{\xi k}$ and $v_i$ . The other two Liouville operators change $\mathbf{v}_i$ and $\mathbf{r}_i$ . This makes it convenient to separate the algorithm into two parts where the positions and velocities of the particles and the NHC are considered separately.

### C. Application chain to NVT ensemble

The fundamental steps to propagate the chain and the entire process are introduced FrenkelSmit‘s book, as shown below Table II for Algorithm 31 (Propagating the chain), Table III for combining process of Algorithm 30 (Equations of Motion : Nosé-Hoover) with Algorithm 32 (Propagating the Positions and Velocities)

```
G2=(Q1*vxi1*vxi1-T)
vxi2+=G2*delt4
vxi1*=exp(-vxi2*delt8)
G1=(2*ke-L*T)/Q1
vxi1+=G1*delt4
vxi1*=exp(-vxi2*delt8)
xi1+=vxi1*delt2
xi2+=vxi2*delt2
s=exp(-vxi1*delt2)
v=s*v
ke=s*s*ke
vxi1*=exp(-vxi2*delt8) G1=(2*ke-L*T)/Q1
vxi1+=G1*delt4
vxi1*=exp(-vxi2*delt8)
G2=(Q1*vxi1*vxi1-T)/Q2
vxi2+=G2*delt4
return ke
```

tab. II: Propagating the chain

```
for i in range(steps):
    ke = chain($Q$,dt,$N$,ke,vel)
    for j in range(N):
        coord += vel * dt/2
    pe, force = LJ(rc, L, N, coord)
    for j in range(N):
        vel += dt * force
        coord += vel * dt/2
    ke = vel*vel/2
    ke = chain(Q,dt,N,ke,vel)
```

tab. III: Program NVT-NHC for entire process

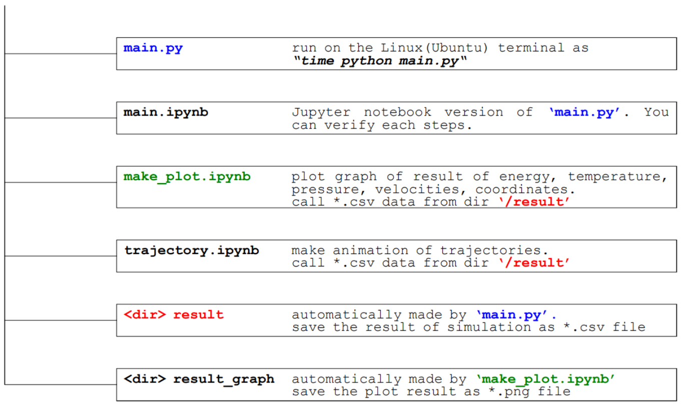
## IV. Programming Code (Python)

### A. Programming Environment

To execute our programm and visualize the result, fundamental Python environment is need.
- Python 3.7 with Ubuntu 16.04.6. for simulation
- Jupyter Notebook is recommended for visualization
- Libraries

- numpy
- sys, os
- math
- pandas
- matplotlib (for visualizing)
- mpl toolkits(for trajectory animation)

### B. Structure of programs



| | |
|---|---|
| main.py | run on the Linux(Ubuntu) terminal as "*time python main.py*" |
| main.ipynb | Jupyter notebook version of 'main.py'. You can verify each steps. |
| make_plot.ipynb | plot graph of result of energy, temperature, pressure, velocities, coordinates. call *.csv data from dir '/result' |
| trajectory.ipynb | make animation of trajectories. call *.csv data from dir '/result' |
| <dir> result | automatically made by 'main.py'. save the result of simulation as *.csv file |
| <dir> result_graph | automatically made by 'make_plot.ipynb' save the plot result as *.png file |

1: Structure of programs to simulate NVT ensemble and visualize the results

### C. Composition of main programs

*1) Initial condition:*

- cutoff : $r_c = 8.5$
- time step : $dt = 0.001$
- initial temperature : $T = 1.0$
- density : $\rho = 0.8442$
- volume : $V = 1,000$
- number of particles : $L = 10$
- boundary condition : periodic
- thermostat mass : $Q = 0.1,\ 0.5,\ 1.0,\ 2.0,\ 5.0$
- number of steps : $N_s teps = 10,000$
- saving frequency : freq $= 10$

*2) Velocity initializaion:* `(def init)` Define initial velocity by uniform distribution, and eliminate center of mass drift. And then, scale the velocity.

*3) Create box:* `(def create_box)` Create a cubic lattice box and initialize coordinates considering minimum image criterion.

*4) Nosé-Hoover Chain:* `(def chain)` This follows the NHC algorithm from Frenkel & Smit’s book. Calculate the scale factor and multiply it to velocity. And update kinetic energy as a return of this function.

*5) Calculate potential energy and force:* `(def LJ_pe_f)` Calculate Lennard-Jones potential energy and force for all pair-particles. And update potential energy, force, virial pressure.

*6) Main loop:* This follows the 'entire process' algorithm.

$1^{st} - half$ : Calculate kinetic energy from Nose Hoover chain first, and update coordinates. And we can obtain new potential energy and force from these coordinates by Lennard-Jones function.

$2^{nd} - half$ : Calculate new velocity and coordinates from the force. And update kinetic energy. And then, input kinetic energy to Nose Hoover chain again to update the final kinetic energy of iteration.

*7) Saving result to analyze:* Save all the result in directory `'/result/'` ; potential energy, kinetic energy, total energy, temperature, the pressure of system and coordinates, velocities of particles for every ten steps,(We can change saving frequency by changing `'freq'`. The default is 10).

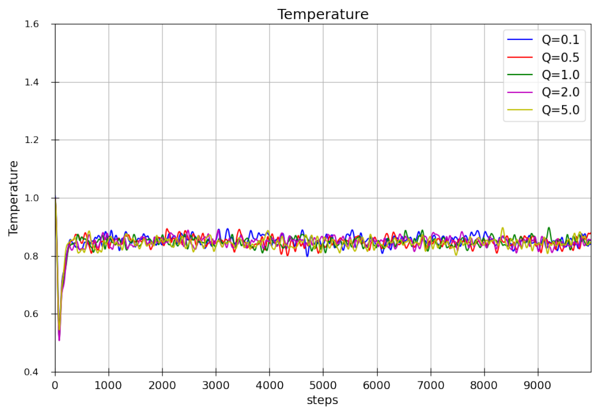
## V. Experiments

In this part, the results of the simulation will be compared following the change of thermostat mass $Q$ and initial temperature $T$.

### A. Temperature& Pressure

Temperature is obtained by kinetic energy at the end of the iteration.

$$T = \frac{KE \times 2}{3N}$$



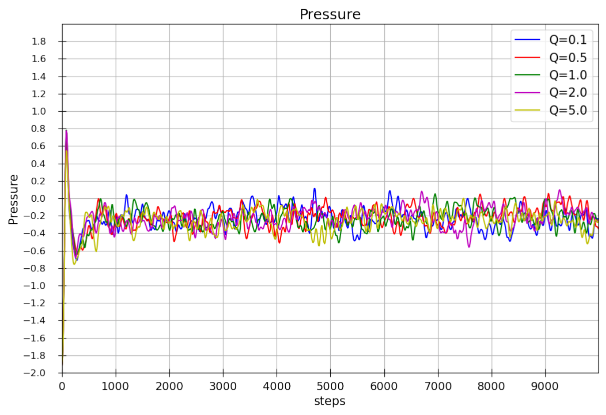
2: Temperature result of 10,000 steps simulation at $T = 1.0$

And Pressure is obtained by Temperature and virial term.

$$p^* = \rho T + \frac{\text{(virial)}}{3V}$$

Temperature and pressure become stable after about 500 steps, regardless of the quantity of $Q$ (Fig.2, Fig.3).

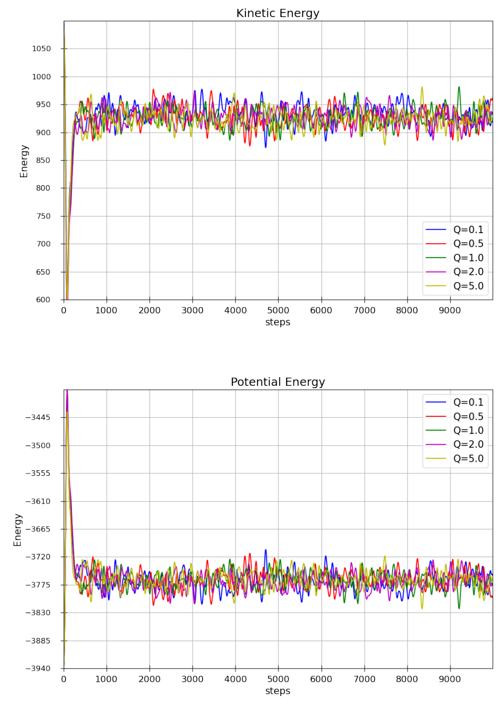
This result makes sense because maintaining the temperature is the main purpose of the thermostat while pressure is derived by temperature.



3: Pressure result of 10,000 steps simulation at $T = 1.0$

### B. Energy

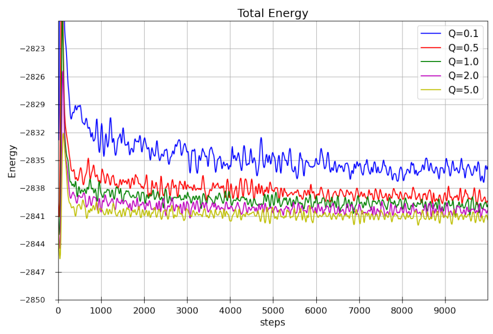
Kinetic energy is obtained by the second half of the Nosé-Hoover chain. And potential energy is derived by Lennard-Jones potential. The total energy is a sum of kinetic energy and potential energy. As for the results of kinetic energy,



4: Kinetic Energy & Potential Energy result of 10,000 steps simulation at $T = 1.0$

its tendency is perfectly the same as the ones of temperature; They are slightly different from each other, and there is no tendency related to a quantity of $Q$. Also, Potential energy seems to have no relation with $Q$. However, the total energy, sum of kinetic energy and potential energy, is proportional to

$Q$. This result makes sense recalling the Hamiltonian form of the equation of motions(Fig.5). The time average of the simulation is shown at Table IV.



5: Total Energy result of 10,000 steps simulation at $T = 1.0$

| | PE | KE | TE | T | P |
|---|---|---|---|---|---|
| Q=0.1 | -3764.96 | 930.69 | -2834.27 | 0.851 | -0.243 |
| Q=0.5 | -3762.01 | 924.06 | -2837.95 | 0.845 | -0.222 |
| Q=1.0 | -3762.93 | 923.81 | -2839.11 | 0.845 | -0.239 |
| Q=2.0 | -3764.55 | 924.57 | -2839.98 | 0.846 | -0.230 |
| Q=5.0 | -3761.48 | 920.73 | -2840.75 | 0.842 | -0.261 |

tab. IV: Time Average of Simulation

## C. Velocity

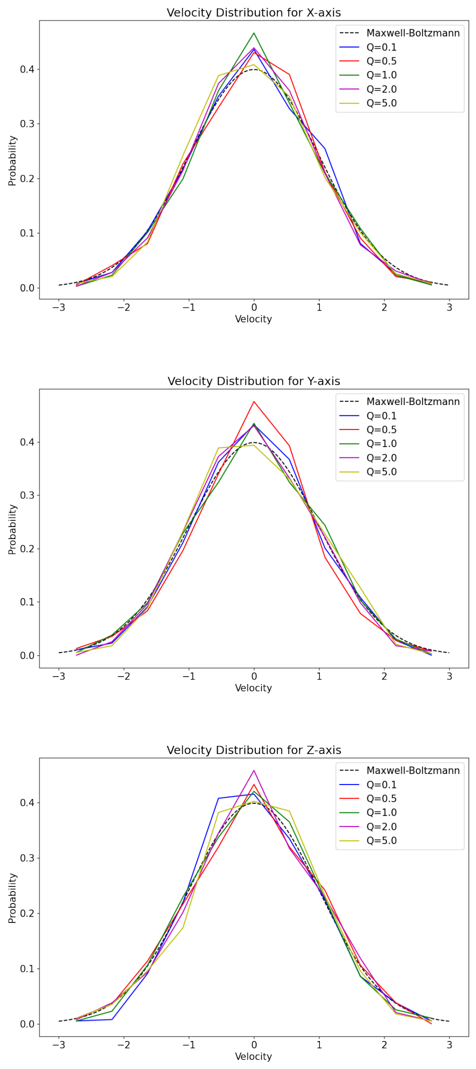
In the classical theory of ideal gases, the velocity distribution function is derived from the Boltzmann transport equation based on the assumption of molecular chaos and a dilute enough gas so that ternary and higher collisions can be ignored. In an ideal gas trajectories of the particles are straight lines, meaning that the particles interact through short range potentials, such as hard-sphere or Lennard-Jones potentials. Under these conditions, the velocity distribution of the particles is described by the Maxwell-Boltzmann distribution function.

Maxwell-Boltzmann velocity distribution for $T = 1.0$ by Eq.(1) and Eq.(2)is

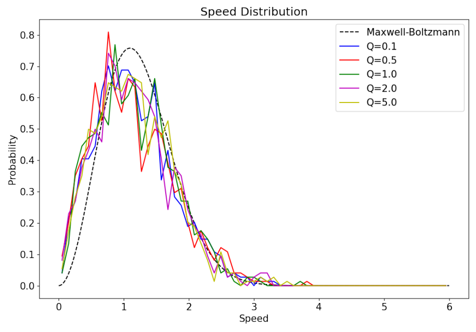
$$\text{velocity} : f(v) = \frac{1}{\sqrt{2\pi}} e^{-v^2/2}$$

$$\text{speed} : f(v) = \sqrt{\frac{2}{\pi}} v^2 e^{-v^2/2}$$

In this simulation, the velocity probability distribution follows the Maxwell-Boltzmann distribution for all directions and all $Q$ (Fig. 5). Although the velocity distribution follows Maxwell-Boltzmann distribution perfectly at $T = 1.0$, the speed distribution is quite different from expected. The probability distribution is more adaptable for $T = 0.6$.



6: Velocity distribution at final step

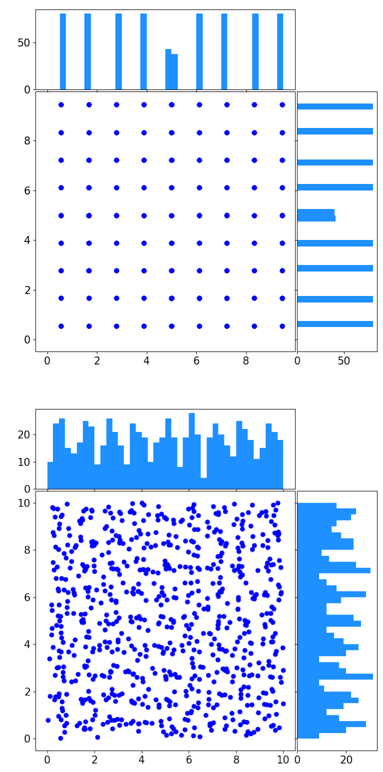


7: Speed distribution at final step

## D. Position

To make sure that the experiment was successful, we can visualize the movement of particles.

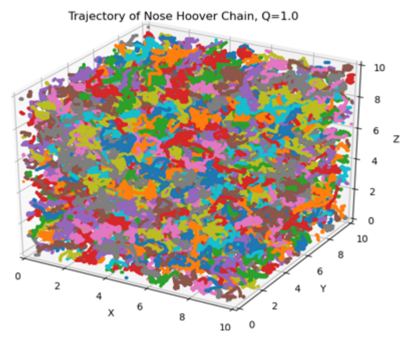
First, we can check the distribution of particles for each step by visualizing, not for the quantitive method. At the beginning part of the steps, the particles are arranged regularly as we initialize the coordinates. After several steps, the distributions begin to spread as the particles move. Finally, the particle distribution is become uniformly after about 2000 steps.



8: Distribution of particles at step 1(top), step 5000(bottom)

Next, we can make a movie of the trajectory of particles. As shown above, the particles are arranged regularly for the first time. And they begin to move irregularly.

A program to make a movie trajectory is attached, and we can see the particles are moving and mixing.

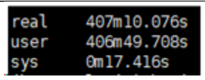
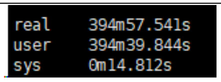
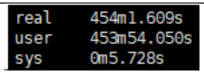
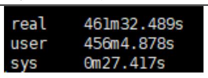
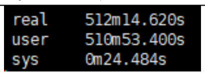
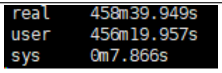
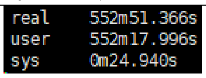


9: Trajectory of particles

## E. *Simulation time*

To save the total experiment time, the experiments have separated the simulations into two computers. Two computers have different computing specifications. All protocol was executed under CPU condition. The real-simulation-time is computed on Linux, executed by command `'time python {filename}'`.

As shown Table V, simulation time spends 394$\tilde{5}$53 minutes depends on CPU performance.

| Computer A | Computer B |
|---|---|
|  real 407m10.076s<br>user 406m49.708s<br>sys 0m17.416s<br>$Q = 0.1, T = 1.0$ |  real 394m57.541s<br>user 394m39.844s<br>sys 0m14.812s<br>$Q = 1.0, T = 1.0$ |
|  real 454m1.609s<br>user 453m54.050s<br>sys 0m5.728s<br>$Q = 0.5, T = 1.0$ |  real 461m32.489s<br>user 456m4.878s<br>sys 0m27.417s<br>$Q = 5.0, T = 1.0$ |
|  real 512m14.620s<br>user 510m53.400s<br>sys 0m24.484s<br>$Q = 2.0, T = 1.0$ |  real 458m39.949s<br>user 456m19.957s<br>sys 0m7.866s<br>$Q = 1.0, T = 1.5$ |
|  real 552m51.366s<br>user 552m17.996s<br>sys 0m24.940s<br>$Q = 1.0, T = 0.75$ | |

tab. V: Simulation time

## VI. Conclusion

We have implemented the Nosé-Hoover chain constant temperature method in conjunction with a reversible integrator algorithm to study the canonical ensemble of Lennard-Jones fluid. The simulation results show that the NHC algorithm can evolve the system to an equilibrium state with further temperature control performance in the NVT ensemble. However, the equilibrium temperature is not equal to the initial temperature. This phenomenon is thought that the initial positions of particles distributing homogeneously are the main reason. The regularly arranged initial position leads to a drift in temperature stability and causes a fluctuation in early steps.

The results of simulation in total energy, potential energy, kinetic energy are very reasonable. The energy fluctuations become stable following the temperature, and we can verify the relationship between thermostat mass $Q$ and total energy is proportional.

When it comes to velocity distribution, the results follow the Maxwell-Boltzmann distribution very well. But for speed distribution, a drift occurred. The drift of speed distribution is caused by temperature drift and truncation error in each velocity direction. Drift phenomena in equilibrium temperature and particle speed distribution are expected adjusted when the tail-correction method is used.

Some future work should be studied after this article. Removing any drifts which are different from analytic solution is accomplished. Next, optimize the system and parameters to reduce truncation error and waste time researching the proper number of steps or thermostat mass. Based on this project, the Python code will be gradually revised to cover several problems and applications to other MD project.

## References

[1] G. J. Martyna, M. L. Klein, M. Tuckerman. “Nose-Hoover chains: The canonical ensemble via continuous dynamics”.The Journal of Chemical Physics. 97, 2635(1992).

[2] D. C. Rapport, “The Art of Molecular Dynamics Simulation” (2nd edition).

[3] D. Frenkel, B. Smit, “Understanding Molecular Simulation From Algorithms to Applications” (2002).

[4] P. Mohazzabi, S. P. Shankar. “Maxwell-Boltzmann Distribution in Solids”.Journal of Applied Mathematics and Physics, 2018, 6, 602-612.

[5] A. Cheng  K. M. Merz, “Application of the Nose-Hoover Chain Algorithm to the Study of Protein Dynamics” (1995) The Art of Molecular Dynamics Simulation” (2nd edition)

[6] P.H. H¨unenberger, “Thermostat algorithms for molecular dynamics simulations”, Adv. Polymer. Sci., 173, 105-149 (2005).