Implementation of a Molecular Dynamic Engine in Python: Stability of Integrators and Temperature-Pressure Relationship of Lennard-Jones Gas

Allen Chen, Summer 2024

Abstract

The purpose of this project was to implement a simple molecular dynamics (MD) engine using Python for simulating ideal and Lennard-Jones gas in a fixed box. This MD engine was used to assess Euler and velocity Verlet integrators and the effects of various simulation parameters upon long-term stability of the simulation, as well as the pressure-temperature relationship and Lennard-Jones gas deviation from ideal gas under various temperatures. Interactions among gas particles are modeled using the Lennard-Jones 12-6 potential, and particle positions and velocities are integrated using either Euler or velocity Verlet algorithms. The system could optionally be coupled to a Berendsen thermostat for temperature control. Data is periodically outputted to a text file where particle positions are stored in Protein Data Bank format. To test stability of integration algorithms, 15 picosecond long simulations were run for both integrators using 1, 2, 4, 8, and 16 femtosecond integration timesteps without the effects of a thermostat, and conservation of energy was assessed using standard deviation and drift rate of the total energy of the system. For analysis of pressure-temperature relationship, simulations were run with thermostat using Euler integration at temperatures from 200K to 400K with a 25K increment, and pressure was analyzed in relation to ideal gas. For small timesteps, drift and standard deviation of net energy in Euler and velocity Verlet algorithms were generally on the same order of magnitude. Larger timesteps induced enough error where the energy of the system was no longer conserved regardless of which integrator was used. Pressure and temperature were found to have a linear relationship where the average pressure deviation from ideal gas decreased as simulation temperature increased, which is the expected behavior. This MD engine worked well and behaved as expected in simple simulations of Lennard-Jones or ideal gas under small timesteps, regardless of integration algorithm used.

Introduction

In recent times, computation and simulation has had an increasing role in scientific study. By parameterizing and describing molecular systems or data sets, simulation can be used to compute far more detailed properties of the system or explore various scenarios that may be otherwise difficult or implausible through experiment.

Molecular dynamics is a foundational method of computer simulation that has substantial impact upon a wide variety of fields of scientific research. Implementation and study of a simple molecular dynamics (MD) engine using Python can offer an impactful learning experience. Understanding and implementing the basics of MD served as a useful developmental project with skills that carry over into other deeper subjects and fields. Although Python is not a very efficient computing language and thus likely not suitable for more extensive or complex MD engines, it is an extremely user-accessible language with numerous existing libraries, making it suitable to run small systems and study various aspects of MD simulation.

This MD engine was used to study various aspects of two different algorithms for numerical propagation of the system, and the temperature-pressure relationship was explored for various systems in comparison to ideal gas.

Theory and Methods

Data Structure

Written in Python, the molecular dynamics engine was built step-by-step by adding additional functionality to the existing code. First, data structures were created for gas particle coordinates, velocities, and forces. For this, NumPy arrays were used instead of Python lists for more efficient handling of data, where coordinates, velocities and forces were each stored in an array of dimensions N by 3 where N was the number of gas particles in the system.

For coordinate output and visualization, the PDB format was used for convenience, where each snapshot was recorded as a new model. The resulting PDB file can be visualized using the Visual Molecular Dynamics (VMD) program.

Additional data was recorded simply as columns in a text file, and could be graphed or analyzed in a variety of programs and using a variety of methods.

Generation of initial coordinates and velocities

In order to perform MD simulations, initial coordinates and velocities needed to be assigned. For coordinates, an obvious solution would be to generate particle positions randomly within the box. However, this could potentially lead to overlapping particles, especially for large systems or smaller box sizes. Instead, gas particles were initially placed on a uniform grid within the fixed box. One of such initial configurations is illustrated in Figure 1.

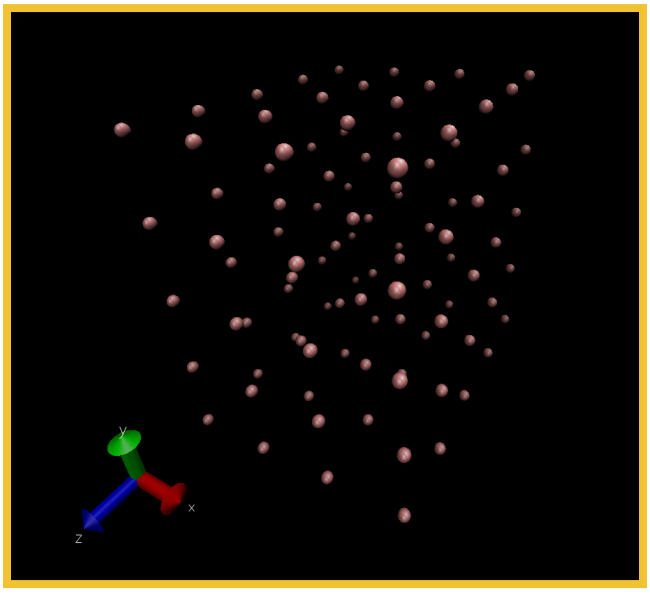

Figure 1: Initial configuration of 100 atoms within a 100 by 100 by 100 angstrom box, generated upon a uniform grid to minimize possibility of initially overlapping atoms.

Initial velocities for particles would be determined by temperature of the system; however, the particles should not all be generated at an identical velocity. Therefore, a Maxwell-Boltzmann distribution was used to initialize particles with appropriate initial velocities.

Maxwell-Boltzmann velocity distributions were generated using the following algorithm:

$$v = \sqrt{\frac{R \cdot T}{m}} \cdot N(0, 1)$$

where v represents particle velocity in one degree of freedom (e.g., along x, y, or z coordinate), R represents the ideal gas constant, T represents the system temperature, m represents the mass of the particles, and $N(0, 1)$ is a normal distribution with the mean at 0 and width of 1.0. Velocity was assigned independently for each particle.

Molecular Dynamics

The basic idea of MD simulation is to treat all gas particles as classical objects and numerically propagate the state of the system, as defined by the coordinates and velocities, using Newton's classical mechanics. In this project, two numerical integrators were evaluated.

The first one was an Euler-based integrator, which only considers the first derivative and has a local error on the order of $O(\Delta t^2)$:

$$x(t + \Delta t) = x(t) + v(t) \cdot \Delta t$$
$$v(t + \Delta t) = v(t) + \frac{f(t)}{m} \cdot \Delta t$$

where x(t) and v(t) represents particle position and velocity at time t, and Δt is the integration timestep. Note this integration was applied separately to each degree of freedom of each particle.

The second one was a slightly more complex velocity Verlet algorithm, which is widely used in actual MD simulations of molecular systems.

Velocity Verlet integration was done according to the following algorithm:

$$x(t + \Delta t) = x(t) + v(t)\Delta t + \frac{1}{2} a(t) \Delta t^2$$
$$v(t + \Delta t) = v(t) + \frac{a(t) + a(t + \Delta t)}{2} \Delta t$$

Where t is time, x is particle position, v is particle velocity, a is particle acceleration. This integration was done separately in each degree of freedom for each particle.

Velocity Verlet integration has local error on the order of $O(\Delta t^4)$, two orders of magnitude less than Euler integration.

An important consideration for both methods of integration are the bounds of the fixed box. Particle collisions with the walls of the box were treated as perfectly elastic. When integration carried the particle outside the bounds of the box, the velocity of the corresponding direction was reversed and the particle was moved to a position inside the box equidistant to the distance it had exceeded the bounds of the box by.

Thermostat

Due to the use of Maxwell-Boltzmann distributions to generate initial particle velocities in a more realistic manner, the semi-random nature of generation usually meant that the temperatures of the systems were not actually true to the desired values. Similarly, in later simulations of Lennard-Jones gasses, particle interactions could also result in changes in velocity that caused actual system temperature to shift away from the desired value.

Thus, a Berendsen thermostat was implemented to rescale velocities towards a desired temperature; this is the equivalent of coupling the system to an external heat bath. At each integration step, velocities are multiplied by some scaling factor λ, calculated with the following formula:

$$\lambda = \sqrt{1 + \frac{\Delta t}{\tau}\left[\frac{T_{bath}}{T(t)} - 1\right]},$$

where λ is the velocity rescaling factor, τ is the coupling constant, Δt is the timestep, T_{bath} is the temperature of the heat bath, and $T(t)$ is the instantaneous temperature of the system. The coupling constant controls how quickly the system temperature relaxes towards the desired temperature. In this project, τ was set to 50 timeteps.

Through the use of this thermostat to rescale velocities, simulations could be run fairly reliably at a desired temperature for various testing purposes.

Lennard-Jones Gas

To simulate Lennard-Jones gas, the potential energy between any two particles was modeled by the 12-6 Lennard-Jones function, defined as,:

$$V_{LJ}(r) = 4\varepsilon\cdot[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}]$$

where V_{LJ} represents Lennard-Jones potential, r represents inter-particle distance, ε represents the depth of the potential well, and σ is the distance where Lennard-Jones potential is zero.

To calculate the force acting upon each particle in each degree of freedom, partial derivation of the Lennard-Jones potential was used to calculate the force from Lennard-Jones potential acting upon each particle in each of the x, y, and z dimensions, which was summed up to calculate the net force acting upon each particle along each dimension.

One issue with this process of summing through every pair of particles and keeping a running total force for all particles is that computational load increases on the order of N^2 as the system size N grows. Additionally, most of these calculations would be performed on pairs of particles far enough removed from each other that the effective Lennard-Jones potential and corresponding force on both particles was negligible. Therefore, a cutoff was implemented to only calculate and update Lennard-Jones forces for particles within a certain distance of each other. A suitable cutoff distance varies based on the parameters used for Lennard-Jones 12-6. However, for the small system simulations run in this project, there was a small enough number of particles for simulations to be manageable in computational load and so these simulations were run without a cutoff distance.

All particles in the system had identical mass. By dividing the force array by particle mass, the resulting new array contained appropriate acceleration vectors that could be used in the integration algorithms in simulation of Lennard-Jones gas.

Data Output and Recording

The above data structures and equations are capable of modeling and numerically propagating systems of ideal and Lennard-Jones gasses. However, without the ability to record or output data, the simulated system is good for little more than calculating a final set of positions and data arrays.

After every integration step, the particle coordinates were output in PDB format and appended as

a new model to a PDB file; as the simulation ran, it would generate a PDB file that could be visualized in Visual Molecular Dynamics.

Additionally, certain information important to the system was also recorded to be used for later analysis, as summarized below. This data was appended each integration step in columns in a text file, and could be used or interpreted in a variety of ways; the data was recorded in columns alongside the elapsed simulation time.

One important quantity is the total energy; this MD engine should theoretically simulate an isolated system with perfect energy conservation. Thus, at each integration step, the net kinetic energy, net Lennard-Jones potential, and total system energy were calculated and appended to a text file.

The net kinetic energy was calculated with the following equation:

$$E_k = \sum_{i=1}^{n} \frac{1}{2} m v_n^2,$$

where ΣE_k represents net kinetic energy, n represents number of particles, m represents the mass of the particles, and v_n represents the magnitude of the overall velocity vector of the n^{th} particle.

Alongside the system's calculations of Lennard-Jones force for use in numerical propagation, the total Lennard-Jones potential can also be quickly calculated. This total would not include the calculations for pairs of particles outside the implemented cutoff distance, but the contribution from excluded pairs should theoretically be negligible.

Total energy of system was calculated using the following formula:

$$\Sigma E = E_{LJ} + E_k$$

Where ΣE represents total energy, E_{LJ} represents net Lennard-Jones potential, and E_k represents net kinetic energy.

Additionally recorded were instantaneous temperature and pressure of the system, which could be used to assess the behavior of the system at various points along the simulation trajectory. These two values were similarly appended to the data file alongside the elapsed simulation time.

Instantaneous temperature was calculated using the following formula:

$$T = \frac{2}{3NR} E_k$$

where T represents the instantaneous temperature, N represents the number of gas particles, R represents the ideal gas constant, and E_k represents the net kinetic energy.

Instantaneous pressure was calculated using the following formula:

$$P = \frac{NkT}{V} + \frac{1}{3V} \sum_i r_i \bullet f_i$$

P is the pressure, N is the number of atoms, k is the Boltzmann constant, T is the instantaneous temperature of the system, V is the volume of the box, r_i is the position vector of atom i, f_i is the total force vector acting upon atom i, and $\bullet$ denotes the dot product of two vectors.

MD simulations, especially when using very simple integration algorithms, often require an extremely small timestep in order to maintain numerically accurate simulation. In this case, it is often unnecessary to output data every single integration step. Thus, while particles would be numerically propagated every integration step to maintain simulation stability, the above outlined calculations for data recording and output would only be performed every N_{out} integration steps, where N_{out} can be changed based upon desired resolution of output frequency.

Simulation Methodology

To assess stability of integration algorithms, simulations were run without the thermostat enabled. A system of 100 atoms in a 100 by 100 by 100 angstrom box was simulated over 15 picoseconds with the initial temperatures generated at 300K. For Lennard-Jones 12-6 potential, $\sigma = 4$ angstrom was used and the depth of the potential well was $\varepsilon = 1000$ J/mol. This simulation was run with 1, 2, 4, 8, and 16 femtosecond timesteps once using Euler integration and once using velocity Verlet integration. With these simulations, the goal was to assess the stability of the integrators by assessing how effectively total energy in the system was conserved (thus no thermostat). With the output data, linear regression was used to calculate drift rate of net energy in J/mol per picosecond for each timestep and integration algorithm. A representative trace of the total energy as a function of the simulation time is shown in Figure 2. Additionally, the standard deviation of the total energy was used to assess the size of energy oscillations for each timestep and integration algorithm, scaled according to the average calculated instantaneous temperature for each simulated system.

To assess the pressure-temperature relationship, simulations were run with the thermostat imposed. A system of 300 atoms in a 120 by 120 by 120 angstrom box was simulated over 10 picoseconds with temperatures from 200K to 400K with a 25K increment using Euler integration. By running similar system simulations at varying temperatures, the goal was to assess how the system behaved based on the temperature, and determine how well this aligned with theoretical behavior and with ideal gas. With output data, average temperature, average pressure, and standard deviation of the pressure was calculated for each quarter of the simulation and overall to assess how the pressure changed as the system approached the thermostat target temperature.

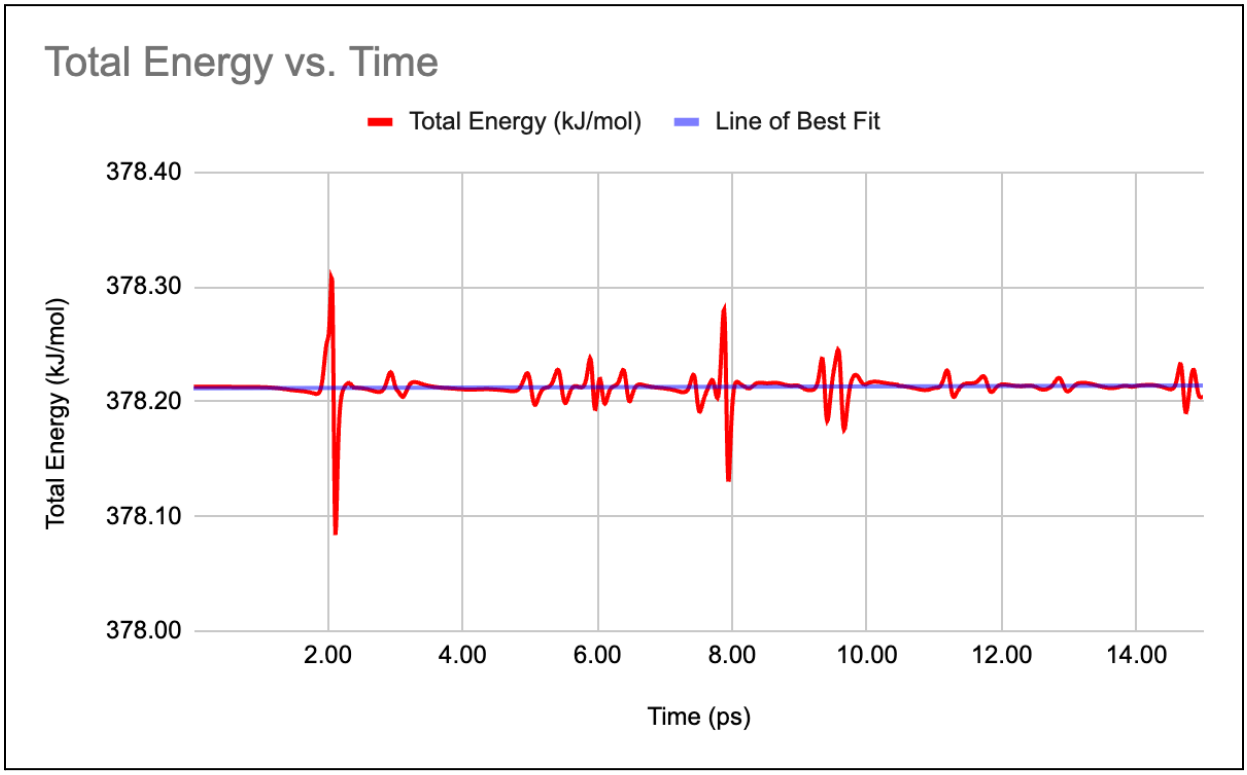

Figure 2: A representative time trace of the total energy, from the 15-ps simulation using the Euler integration algorithm with 1-fs timestep.

Results

Stability of Euler and Velocity Verlet Integrators

In an ideal simulation, energy conservation would be perfect and the total energy would not have any fluctuation or drift. Linear regression was used to calculate the rate at which total energy drifted for the integration algorithms at different timesteps (Table 1, Figures 3 and 4). Generally, greater integration timesteps led to larger drift and therefore less accurate simulation. At the smallest timestep of one femtosecond, Euler integration drifted at a rate of 1.60E-01 kJ/mol per ps and velocity Verlet integration drifted at a rate of 4.90E-01 kJ/mol per ps. At the largest timestep, sixteen femtoseconds, Euler integration drift rate was 9.65E+00 kJ/ps*mol and velocity Verlet integration drift rate was 1.21E+00 kJ/ps*mol, which is slightly larger. Generally, both integration algorithms tended to drift at a rate on the same order of magnitude and neither was clearly more accurate than another. While the graph (Figure 3) visually seems to imply that velocity Verlet might have smaller drift and better long-term stability than Euler at larger timesteps, it is also important to recognize that these single simulation trajectories might not be representative of aggregate trends. Generally, though, drift rate for both integration algorithms increased by an order of magnitude for timesteps larger than two femtoseconds. Overall, both algorithms seemed to perform similarly; at larger timesteps, the error introduced by numerical propagation is enough that the integration algorithm is immaterial.

To measure the magnitude of oscillations of the total energy (e.g, see Figure 2), the standard deviation was calculated for both integration algorithms at different timesteps (Table 1, Figure 4). Greater integration timesteps led to larger standard deviation and larger fluctuations in the total energy. The data suggests that velocity Verlet integration had consistent and slightly greater standard deviation than Euler integration, with velocity Verlet standard deviation (1.40E+02 kJ/mol) being about double that of Euler standard deviation (5.29E+01 kJ/mol) at a four femtosecond timestep. However, as earlier established, timsteps greater than two femtoseconds started leading to drift rates and errors at greater orders of magnitude; at timesteps equal to or less than two femtoseconds, standard deviation of total energy for velocity Verlet and Euler integration were on the same order of magnitude.

It should also be noted that for this relatively short simulation on a relatively small and dilute gas system, the average drift rates were also impacted by the collision frequency and temperature of each system; a system that generated with more intermolecular interaction would have greater drift and standard deviation than a different, more subdued system would. Nonetheless, the timestep seemed to have a much greater effect on total energy drift and fluctuations than integration algorithms did; at small timesteps, both integration algorithms seemed to function similarly and had total energy drift on the same order of magnitude.

Table 1: Energy fluctuation and drift with different timesteps. The results were derived from 15 ps simulations of 100 atoms in a 100 by 100 by 100 angstrom box. The initial velocities were generated following the Maxwell-Boltzmann distribution at 300K. “Verlet” represents “velocity Verlet” for brevity. SD stands for standard deviation.

Timestep (fs)	Euler Drift Rate (kJ/mol per ps)	Verlet Drift Rate (kJ/mol per ps)	Euler SD (kJ/mol)	Verlet SD (kJ/mol)
1	1.60E-01	4.90E-01	1.21E+01	2.01E+01
2	2.90E-01	2.80E-01	2.46E+01	4.38E+01
4	6.49E+00	4.19E+00	5.29E+01	1.40E+02
8	7.00E-02	1.07E+00	1.17E+02	1.99E+02
16	9.65E+00	1.21E+00	2.07E+02	7.78E+02

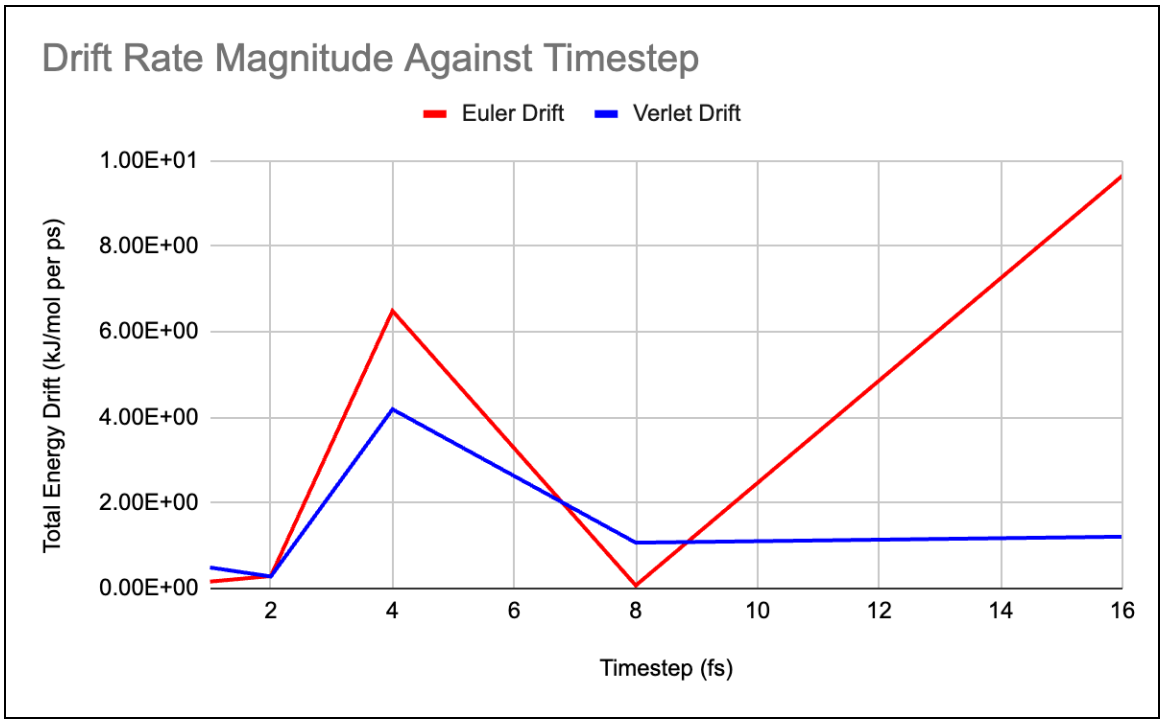

Figure 3: Drift rate of the total energy with Euler and velocity Verlet integration algorithms with various MD timesteps.

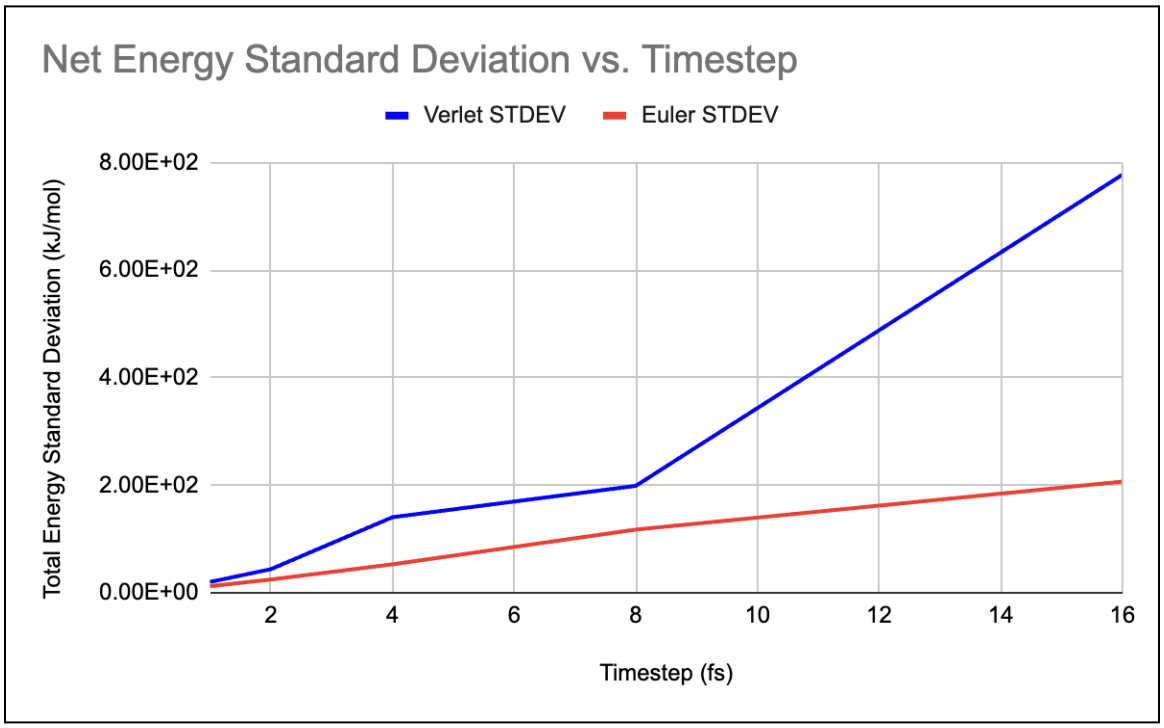

Figure 4: Standard deviation of the total energy with Euler and velocity Verlet integration algorithms with various timesteps.

Thermostat, Pressure-Temperature Relationship, and Deviation from Ideal Gas

Maxwell-Boltzmann distributions generated an initial net temperature not exactly equivalent to the desired temperature for each simulation (due to random assignment via the normal distribution), though this temperature stabilized towards the desired temperature as the simulation went on.

The average difference between calculated temperature and desired temperature during the first quarter of the simulation trajectory among all nine simulated systems was 3.46K, and average standard deviation of average temperatures among the nine systems during the first quarter was 2.45K. By the fourth quarter of the simulation trajectory, average difference between calculated and desired temperature had decreased to 0.19K and average standard deviation of average temperatures had decreased to 0.81K (Table 1). This change reflected the Berendsen thermostat bringing the system to the desired temperature and the system stabilizing.

This apparent stabilization of the system towards equilibrium was not reflected in the standard deviation from average pressure in each quarter. The four quarters of the simulation trajectory had an average pressure of 7.18E+5 Pa, with a SD of 3.38E+03 Pa. The four quarters had an average pressure SD of 1.72E+04 Pa, with an extremely minimal SD 3.33E+02 Pa (Table 2). In other words, there was relatively little variance in average pressure or fluctuations of pressure (SD of average pressure) throughout the entire simulation trajectory. This is because the zero order of magnitude kelvin fluctuations in temperature over the course of the simulation (which represents <1% of the temperature), while noticeable in regards to the Berendsen thermostat, would induce very minimal amounts of change in overall pressure; instead, the fluctuations in pressure would receive much greater impact from particle collisions and the instantaneous state of the system.

The relationship between temperature and pressure was highly linear, which is the expected behavior. Using the average pressures and average temperatures for each system trajectory, the temperature-pressure relationship was fit using linear regression. The average of the residuals for this line of best fit was equal to 0.0 Pa and the average of the absolute values of the residuals for this line was 1.68E+3 Pa, which is 0.23% of the average pressure (Figure 5). The simulated temperature and pressure relationship had extremely minimal deviation from a linear relationship.

The pressure-temperature relationship of the simulated Lennard-Jones gas systems were graphed and compared to the pressure-temperature relationship of ideal gas (Figure 6). The average deviation between simulated LJ gas pressure and ideal gas pressure over all temperatures was 3.21E+3 Pa, which is less than 1% of total pressure. The deviation between Lennard-Jones gas

pressure and ideal gas was relatively small. This deviation was also graphed against the average temperature of the system (Figure 4). As the temperature of the system increased, the deviation between LJ gas pressure and ideal gas pressure seemed to decrease. This is the expected behavior; as particle velocities generally increase, particle interactions proportionally have a smaller effect upon the system and thus the gas particles become a better approximation of ideal gas. It is worth noting that the parameters chosen for Lennard-Jones 12-6 potential ($\sigma = 4A, \varepsilon = 1000J/mol$) are close to the parameters for the noble gas Argon, which should theoretically behave similarly to ideal gas. In short, under the effects of a thermostat and using Lennard-Jones gas parameters close to that of a noble gas, the system behaved as expected.

Table 2: Summary of system temperature and pressure. The simulated system contains 300 atoms in a 1728000 cubic angstrom cube box and the simulation was run for 10 ps with 2 fs timesteps. The properties are shown from each of the four quarters of the simulations.

Simulation Quarter	System ID	System Target Temp. (K)	Avg. Temp. (K)	Temp. SD (K)	Avg. Pressure (Pa)	Pressure SD (Pa)
First	1	200	205.52	2.93	493,488.60	10,894.31
First	5	300	302.98	0.90	728,654.79	12,648.96
First	9	400	404.43	1.54	972,047.16	14,920.35
Second	1	200	201.53	1.05	477,913.02	14,771.12
Second	5	300	301.14	0.78	708,363.23	18,767.48
Second	9	400	400.85	1.38	963,876.58	22,160.93
Third	1	200	200.42	0.83	475,958.14	12,364.69
Third	5	300	300.11	0.96	717,519.32	19,985.17
Third	9	400	400.39	1.11	958,873.79	25,173.04
Fourth	1	200	199.98	0.89	471,635.08	19,459.62
Fourth	5	300	300.02	0.76	716,645.27	19,747.44
Fourth	9	400	399.87	1.02	960,557.73	23,485.89

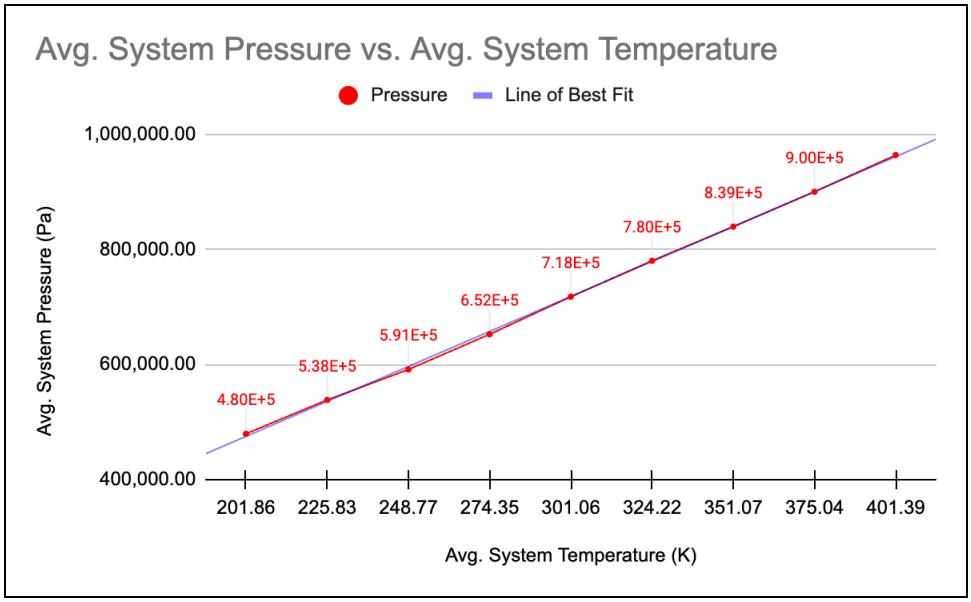

Figure 5: Average pressure graphed against average temperature for each system, with line of best fit to illustrate the linear relationship between pressure and temperature.

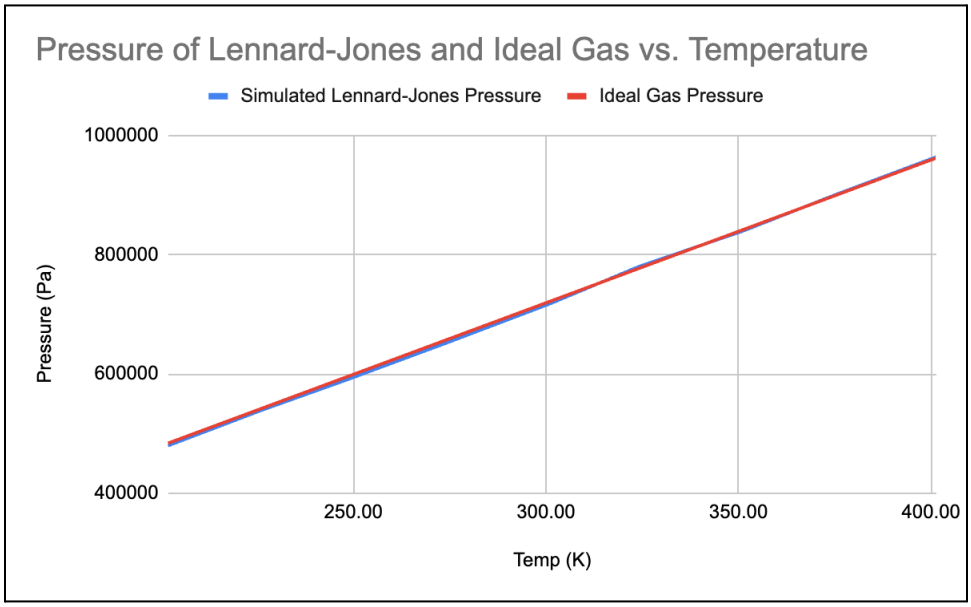

Figure 6: Simulated system average pressures and ideal gas pressures graphed against temperature.

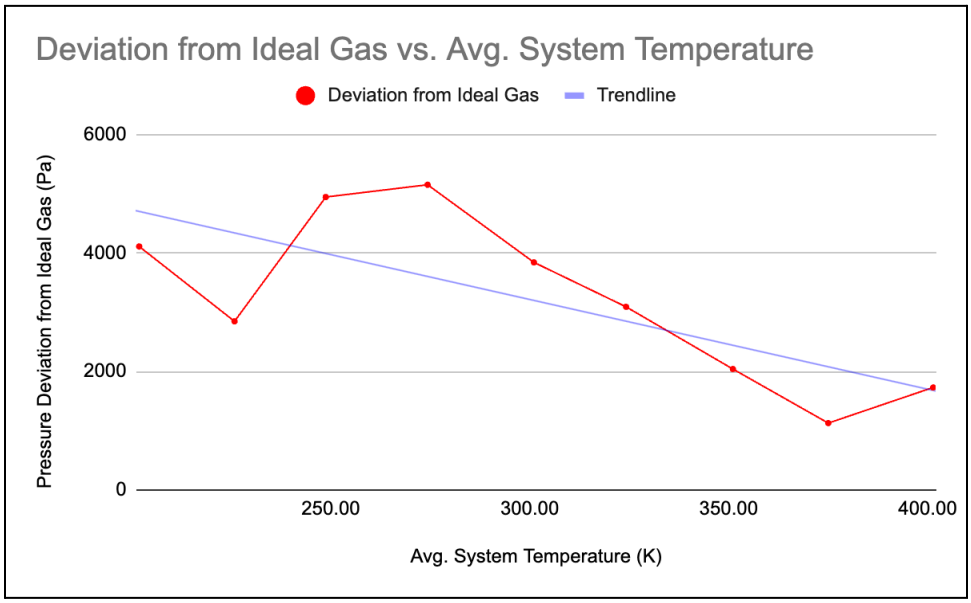

Figure 7: Magnitude of the deviation of simulated Lennard-Jones gas pressures from ideal gas pressures graphed against temperature with line of best fit.

Conclusion

With the increasing role of computation in modeling, molecular dynamics similarly grows in impact and importance. This project, which aimed to implement and study a simple molecular dynamics engine in Python, used different integration algorithms to model Lennard-Jones 12-6 gas in a fixed box. Python, being very user-accessible, served as a good language to learn the foundational aspects of molecular dynamics.

In simulations numerically propagating using either implicit Euler integration or velocity Verlet integration at various timesteps, drift rate and standard deviation of net energy for both integration algorithms did not deviate by any statistically significant margin for timesteps less than 4 femtoseconds. At greater timesteps, both integration algorithms had a far high enough drift rate and standard deviation that differences between the algorithms were immaterial. Overall, in simulations of these relatively small and ideal systems of gas particles, both algorithms seemed to function similarly in terms of long-term stability.

In simulations run with the system set at varying temperatures, the relationship between pressure and temperature was found to increase linearly, with slight deviations from ideal gas. Additionally, as temperature increased, the deviation from ideal gas decreased. The system's deviation from the thermostat temperature also decreased throughout the simulation trajectory, reflecting the system relaxing towards the desired temperature. All of this is the expected behavior.

Despite the relative simplicity of this MD engine, it was effective in modeling trajectories of small systems of ideal and Lennard-Jones gas under various temperature conditions. Regardless of which integration algorithm was used, the engine is able to maintain relative long-term stability and simulate particle behavior at varying temperatures.

Data Availability

The Python script of this MD engine, documentation on script functionality and instruction for running new simulations, and the simulation output data files used in this report are available at the following link: https://github.com/nAmnesiac/py_basicMD/tree/main.

Acknowledgements

The completion of this project and report would not have been possible without the support, feedback, and guidance of Professor Jianhan Chen, University of Massachusetts Amherst.

I would also like to thank the members of the Chen Research Group, University of Massachusetts Amherst, for their assistance throughout this process.

Lastly, thank you to Christoph Wehmeyer and others on Matter Modeling Stack Exchange and Stack Overflow for assisting in the modeling of force fields and in Python syntaxing.

References

Berendsen thermostat. (2010). SklogWiki. Retrieved October 10, 2024, from http://www.sklogwiki.org/SklogWiki/index.php/Berendsen_thermostat

Hollingsworth, S. A., & Dror, R. O. (2018). Molecular Dynamics Simulation for All. Neuron, 99(6), 1129–1143. https://doi.org/10.1016/j.neuron.2018.08.011

van Gunsteren, W. F., Bakowies, D., Baron, R., Chandrasekhar, I., Christen, M., Daura, X., Gee, P., Geerke, D. P., Glättli, A., Hünenberger, P. H., Kastenholz, M. A., Oostenbrink, C., Schenk, M., Trzesniak, D., van der Vegt, N. F., & Yu, H. B. (2006). Biomolecular modeling: Goals, problems, perspectives. Angewandte Chemie (International ed. in English), 45(25), 4064–4092. https://doi.org/10.1002/anie.200502655

Nikolic, B. K. (2003). Verlet Method. University of Delaware. Retrieved October 10, 2024, from https://www.physics.udel.edu/~bnikolic/teaching/phys660/numerical_ode/node5.html