Molecular Dynamics Simulation

Solid, Liquid & Gaseous Argon

Lukas Welzel, Catherine Slaughter

Faculty of Science, Leiden University
e-mail: welzel@strw.leidenuniv.nl, slaughter@strw.leidenuniv.nl

April 22, 2022

Context. The increase of technology in recent decades gives the scientific community unprecedented access to machines with higher computing power than ever. This access allows Physicists to better tackle analytically challenging problems via computational numerical methods.
Aims. To successfully simulate a physically sound system of 108 argon atoms in various states of matter (solid, liquid, gas), and to systemically and quantitatively analyse the behaviours of each
Methods. We utilise a Velocity-Verlet numerical integrator, truncating the potential to a more realistic sphere of influence in order to run large simulations in a reasonable amount of time. Additionally, minimal image convention is implemented to handle boundary conditions.
Conclusions. We simulate a system of 108 atoms with tentative success. Total system energy is conserved. While pair correlation functions show a general decrease in structure as the simulation temperature increases, they do not quite align with the states of matter expected for a given input temperature and density. Further analysis of the kinetic energies of the simulations would be fruitful. We find the local pressures for the three states of matter to be 0.897 (gas), 0.723 (liquid), and 0.084 (solid).
Key words. molecular dynamics – numerical methods – argon

Contents

1 Introduction and Background **1**

2 Methods **2**

3 Results **5**

4 Summary **7**

A Additional Equations **8**

B Extended Tables **8**

C Extended Figures **8**

1. Introduction and Background

The past decades have seen an unprecedented spike in technological advancement. Now more than ever, physicists have access to faster, cheaper, and more powerful computers. Problems that cannot be solved analytically (and had been previously set aside) can now be studied more closely than ever. New generations of scientists are jumping at the opportunity to learn the simulation methods that will drive the future of computational physics. One of these new methods involves modelling n-body systems via numerical integration. Numerical methods have applications in many subfields, including particle physics, astrophysics, and thermodynamics. For this project, we have been tasked with simulating and analysing the molecular dynamics of a system of Argon atoms using such numerical simulation techniques.

1.1. Physics

In order to study the motions of this set of Argon atoms, we return to first principles. Namely, Newton's Second Law of Motion:

$$m\frac{d^2\boldsymbol{x}}{dt^2} = \boldsymbol{F}(\boldsymbol{x}) = -\nabla U(\boldsymbol{x}) \tag{1}$$

It is from this equation that the interactions can be derived, given a potential. Because Argon atoms are neutral, they do not interact via Coulomb's Law. Instead, the primary function that governs the interactions between such particles is the Lennard-Jones Potential.

$$U_{LJ}(r) = 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right) \tag{2}$$

The first term in the potential represents the quantum mechanical Pauli repulsion at very small distances, while the second term represents an attractive interaction due to induced dipoles in the atoms. A plot of the Lennard-Jones potential (Figure 1) shows that σ corresponds with the radius at which the potential is zero, and ϵ is the depth of the potential well at the function's minimum

The force that accelerates the particles (via Equation 1) is simply the negative derivative of the potential (see Figure 2):

$$-\nabla U = \frac{-24\epsilon\sigma^6\left(r^6 - 2\sigma^6\right)}{r^{13}}\hat{\boldsymbol{r}} \tag{3}$$

In principle, Newton's Second can then be applied to determine the movements of each of the particles. In practice, however, this is extremely difficult to do analytically. The number of atoms needed to produce an interesting science result creates a remarkably complicated system, even in such a relatively simple potential. Instead, we turn to numerical methods.

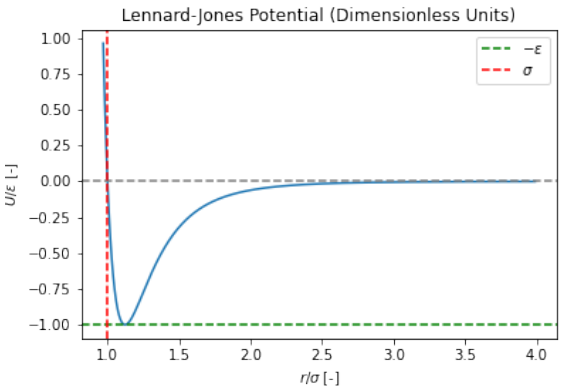

Fig. 1. The Lennard-Jones Potential. It is clear through this plot the physical significance of sigma (red) and epsilon (green).

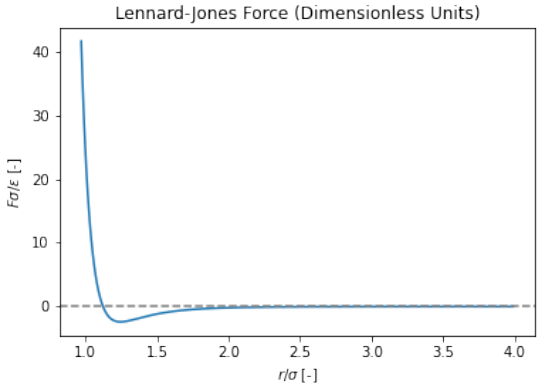

Fig. 2. Lennard-Jones Force

2. Methods

To tackle this problem, we have created an object-oriented program with two primary classes of importance. The first, a meta particle class ("particle_class.py"), holds the relevant data for each individual particle, such as position, velocity, and mass. This is sub-classed by ("argon_class.py") to specifically create Argon atoms. The second class of significant importance defines the simulation itself ("md_simulation_class.py"), and holds global data, such as density, temperature, minimum image convention box length (see subsection 2.2), and timekeeping tools–number of timesteps, timestep length, etc.–as well as handling the initial velocity scaling to reflect the input temperature (see subsubsection 2.5.2). This class essentially *is* the simulation. A single simulation is run through a wrapping function ("md_simulation_func.py") which establishes initial positions and first input velocities (subsection 2.5), and actually runs the simulation. Many simulations can be run simultaneously via a wrapper code ("many_sim_wrapper.py"), though this should be done with care.

2.1. Numerical Integration

In simplest terms, numerical methods differ from analytical calculations by discretizing the time evolution of a given system into small, but finite, time steps. In doing so, the important quantities (position, force) to trace the motion of the system can be calculated at a given time, then propagated to the next, step-by-step, instead of continuously. This reduces the complexity of the problem significantly, while (assuming the timestep is sufficiently small) generally retaining the important physical properties.

The simplest implementation of Numerical integration is via the Euler method. This method comes from the first and second integration of Newton's Second law (which has an acceleration term) to get the velocities and positions of the particles.

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \boldsymbol{v}_n h \tag{4}$$

$$\boldsymbol{v}_{n+1} = \boldsymbol{v}_n + \frac{1}{m}\boldsymbol{F}(\boldsymbol{x}_n)\, h \tag{5}$$

Where n is a given step number.

While this method is acceptable for some numerical simulations, it does not conserve total system energy over time. Therefore, we choose a somewhat more complicated–and accurate–method.

2.1.1. Velocity-Verlet Algorithm

The Velocity Verlet Algorithm is an extension of the typical Verlet Algorithm (Verlet 1967) where the velocities are explicitly used in updating the positions of the particles. The new equations that define this algorithm are calculated by Taylor expanding Equation 45, about $(t + h)$ (where h is the timestep value) then substituting in $\boldsymbol{v}(t) = \dot{\boldsymbol{x}}(t)$ and $\boldsymbol{F}(x)\frac{1}{m} = \ddot{\boldsymbol{x}}(t)$. There is an extra substitution for $\ddot{\boldsymbol{v}}$ that can be found via an additional Taylor expansion.

$$\boldsymbol{x}(t+h) = \boldsymbol{x}(t) + h\boldsymbol{v}(t) + \frac{h^2}{2m}\boldsymbol{F}(\boldsymbol{x}(t)) + O(h^3) \tag{6}$$

$$\boldsymbol{v}(t+h) = \boldsymbol{v}(t) + \frac{h}{\partial m}(\boldsymbol{F}(\boldsymbol{x}(t+h)) + \boldsymbol{F}(\boldsymbol{x}(t))) + O(h^3) \tag{7}$$

In a simple implementation of this algorithm, the force on a given particle is obtained by summing the individual forces (Equation 3) between it and each of the other particles being simulated. A faster implementation is discussed in subsection 2.3. The Velocity-Verlet Algorithm is superior to the original because it is self-starting, meaning it does not require information about the $t = -h$ step to calculate the $t = 0$ step.

2.2. Boundary Conditions

In order to simulate a finite number of particles we implement the minimal image convention (MIC, see Figure 3), a type of periodic boundary condition. In this convention the region of interest that is simulated (typically a cube) is infinitely tiled in all directions. This means that each particle at position $\mathbf{x}$ has identical clones at $\mathbf{x} \pm \mathbf{b} \cdot k$ where $\mathbf{b}$ is the tensor of all unit ordinal directions in the cube scaled by length b, the cube length, and k a positive integer. This in turn means that the particle positions are only dependent on the other particles nearest copy. The cube that is simulated becomes a 3-torus. While there are multiple ways of implementing the MIC, Deiters (2013) finds that so called class-A algorithms have the best performance for MDS without rotational degrees of freedom. Hence, we can implement an efficient boundary transition using implicit integer casting which is resource efficient and can be vectorized. This implementation has the tracked particles remaining within $-b \leq \Delta\mathbf{x} \leq +b$ where $\Delta\mathbf{x}$ is the position difference between two particles. We furthermore define `int:` k, `float:` b, `float:` b_r, `float:` b_2, and `float:` b_{2r}, which are the box length, reciprocal box length, half box length and reciprocal half box length, respectively.

MCI is then implemented for *all* distances that are used internally as shown in Equation 8. This means that particles moving through a boundary and distances are covered by this algorithm.

$$\begin{aligned} &\texttt{int}: \ k = \Delta\mathbf{x} \cdot b_2 \\ &\Delta\mathbf{x}' - = k \cdot b \end{aligned} \tag{8}$$

With this implementation all relevant properties are conserved and the absolute accuracy of `numpy.ndarrays` is kept far below all distances.

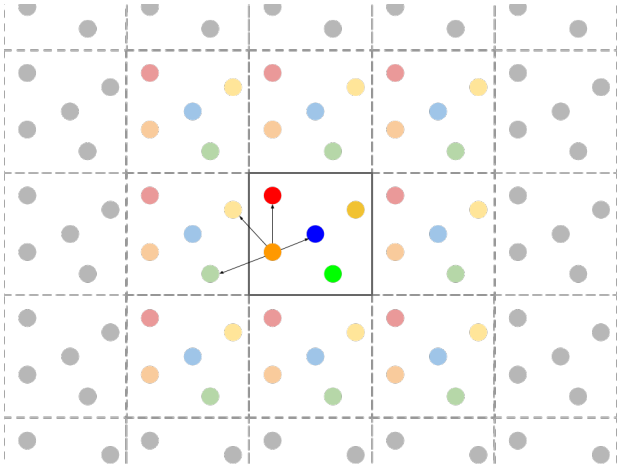

Fig. 3. A 2D representation of the minimal image convention with 5 particles. The solid box and brightly coloured particles represent the actual simulation, the other boxes are the other virtual "images". The black arrows represent the interaction radii used to calculate the potential for the orange particle.

2.3. Sphere of Influence & Large Simulations

Large simulations have the problem of the number of interactions increasing with the number of particles as $O(n^2)$, which is unsustainable. Hence, we implement two methods to decrease the computation effort without decreasing accuracy of the simulations. Firstly, we evaluate each particle pair interaction only once since the forces on the particles are equal but opposite and the potential is the same. This lower the interactions to $O(n(n/2))$, as the symmetric, square pair matrix becomes a triangle matrix. Secondly, we implement a sphere of influence (SOI) for each particle, by defining a cut-off distance $r_{threshold}$ beyond which we assume negligible force from particles on the SOI-centred particle[1]. For large enough $r_{threshold}$ this is a good assumption since the Lennard-Jones potential becomes quickly asymptotic towards zero at increasing separations. Nevertheless, the potential needs to be shifted by $-U(r_{threshold})$ since particles would otherwise experience a "jump" when entering or leaving a SOI. The resulting truncated and shifted Lennard-Jones potential (LJTS) preserves the features of a full LJ potential, like the triple-point, but does shift some statistical properties of the simulated medium. (Stephan et al. 2018) Both methods combined reduce the number of interactions that need to be calculated dramatically, and make large simulations feasible. The influence of the LJTS is dependent on the state of the medium and grows with the number of particles. The limiting best case that can be archived is $O(n)$ interactions, however in our simulations the interactions grow as $O(n) << N < O(n(n/2))$ with a larger reduction for low densities.

We implement these two methods through a pair matrix in which we assign `True` values to pairs that can be masked out due to negligible interactions. This matrix is a quasi-triangular matrix with `True` values along the diagonal. Each m steps the pair distances are calculated again and cells in which $r_{i,j} > r_{threshold}$ are set to `True`. In each simulation step the row corresponding to a particle is then selected when the interactions are calculated and all particles with `True` cell values in the pair matrix are skipped. We use `np.ma.maskedarray` for this which is very efficient and compresses masked particles out of the iterator.

One of the most obvious changes we expect from the potential shift is the shift of the peaks in the radial distribution function (RDF, see subsection 3.2). Especially for solids we expect stark peaks due to sphere packing. The position of the peaks follows directly from the distances between tightly packed spheres with effective radius of σ, see Equation 9.

$$r_{peak\ RDF}(k) = \sqrt{k}\sigma, \tag{9}$$

where k the (positive) integer number of the coordination shell. However, since the potential and hence the "effective equilibrium distance" for two particles is shifted the RDF peak also must shift. For a shift where the LJ potential is truncated at zero the new RDF peak is then also a function of the shift as shown in Equation 10.

$$r_{peak\ RDF}(k) \approx \sqrt{\frac{k\sigma}{1 - |U(r_{threshold})|}} \quad \text{for small } U(r_{threshold}) \tag{10}$$

2.4. Dimensionless Units

In SI units, the important values in this simulation often end up being on very different–and very small–orders of magnitude. This makes the simulation unmanageable to track, and more prone to negative computational effects like round-off errors. Therefore, we define a new set of "dimensionless units" to work in. This is done by demanding some physical constants (Boltzmann's constant, argon mass, σ, and ϵ) be equal to 1, and using these same constants to rescale other properties accordingly. In the case of our simulation, the input values (density, temperature, and timestep length) are already entered in these dimensionless units.

In this unit system, Newton's Second Law (Equation 1) becomes

$$\frac{d^2\tilde{\mathbf{x}}}{d\tilde{t}^2} = -\tilde{\nabla}\tilde{U}(\tilde{r}), \tag{11}$$

the Lennard-Jones potential (Equation 2) becomes

$$\tilde{U}(\tilde{r}) = 4\left(\tilde{r}^{-12} - \tilde{r}^{-6}\right), \tag{12}$$

and the force (Equation 3) acting between two particles becomes

$$-\nabla\tilde{U} = \frac{-24\left(\tilde{r}^6 - 2\right)}{\tilde{r}^{13}}\hat{\tilde{\boldsymbol{r}}}. \tag{13}$$

Additional equations for unit conversion can be found in Table B.1.

[1] As well as the inverse, however this is already implemented through the point above.

2.5. Initial Conditions

While there are a number of various initial parameters that can be passed into our simulation, there are three of particular importance. The density, and temperature are the parameters that ultimately determine the phase we are simulating, while the number of atoms helps define the physical simulations size. It should be noted that density and temperature are input into the code in dimensionless units as described in subsection 2.4. From these input parameters, we are able to establish the initial positions and velocities of the particles in the simulation.

2.5.1. Positions

We first enforce the input density by scaling the side length of our simulation box. The box length of the minimum image convention bounding box can be found via the definition of density and the number of particles being simulated. In SI units:

$$\rho = \frac{m_{\text{argon}}\, n_{\text{particles}}}{L^3}$$

Of course, in dimensionless units, $m_{\text{argon}} = 1$, therefore, we can ensure the density of the system by setting the bounding box length to:

$$L = \left(\frac{n_{\text{particles}}}{\rho}\right)^{1/3}$$

We also want to ensure that no two atoms are initialised too close to one another, to prevent a nonphysical acceleration due to the r^{-12} term in the potential. Additionally, the use of the minimal image convention means that a consistent, repeating structure is useful. As such, we initialise the particle positions according to a relatively simple face-centred cubic lattice (FCC). This structure can also be described by a representative unit of 4 particles in a triangular-pyramid formation (Figure 4). By repeating these representative units throughout space, the greater lattice is formed with relative ease.

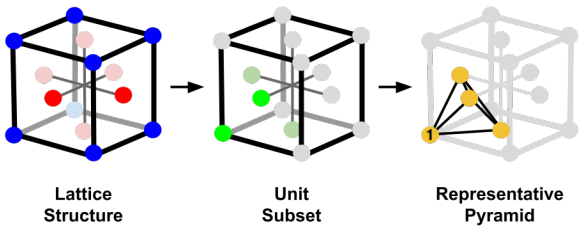

Fig. 4. Finding the representative unit from the face-centred cubic lattice. Note that for the representative pyramid, the anchor particle is labelled "1".

For our purposes, we build a $3 \times 3 \times 3$ array of the representative units. With 4 particles per unit, this gives a default of 108 particles. The code is generalised such that one could (in principle) run it with $n^3 \times 4$ particles (for some positive integer n), creating an $n \times n \times n$ box of 4-particle units. In practice, the efficacy of a such a run is limited by the user's computing power.

In our code, this is implemented by establishing one particle of the unit to serve as the "anchor". The anchor particles are evenly spaced out across the simulation box (n units/side). For each unit, the other 3 particles' positions are shifted in 2 dimensions from their anchor's. For a box length L, the shifts are of length $L/2/n$ (half the distance between anchors). One particle is shifted in x and y, another in x and z, and the third in y and z.

2.5.2. Velocities

While the initial positions are well defined for particles with spherical potentials like Argon, the velocities are less so. Generally, there are four constraints on the velocities before the simulation can begin:

1. The movement of the particles is due to thermal motion and hence random. The simulation must reflect this, and, to be able to meaningfully analyse the system, the simulations runs must be independent and hence initialised in a random state.
2. The total momentum of the system must be zero or the resulting kinetic energy of the system must be accounted for in the analysis. If the entire ensemble moves together there is an inertial frame of reference which we should adapt for convenience.
3. The system must be in a (quasi) equilibrium state for us to be able to meaningfully analyse the system. This means that the initial state must be relaxed or very close to being so.
4. Finally, we want to analyse the system in a specific state, specifically at (or very near) a certain density and temperature.

In total this means that we need to relax the system into an equilibrium state by drawing random velocities, setting the total momentum $\mathbf{p}$ of the system to zero and re-scaling the system until it is both near an equilibrium state and at the correct density and temperature. The (global) density is constant even if there are local perturbations since there is no mass transport through system boundaries. The kinetic energy of a system in equilibrium with N degrees of freedom (here 3 translation directions of which one is blocked due to conservation of momentum) is given in Equation 14.

$$E_{\text{kin}}^{\text{target}} = (N-1)\frac{3}{2}k_B T_{target} \tag{14}$$

Using the total kinetic energy in the system the velocities can then be re-scaled using Equation 15.

$$\mathbf{v}_i \to \lambda \mathbf{v}_i \text{ where } \lambda = \sqrt{\frac{(N-1)3k_B T_{target}}{\sum_i m^\star \hat{\mathbf{v}}_i^2}} \tag{15}$$

This is an iterative process since the system is moved out of the potential equilibrium state into which it relaxed after the re-scaling. The process needs to be repeated until it is certain that the system is close to its equilibrium state. This is especially challenging for high temperature or low density fluids (gas) since the Verlet integrator does not (locally) conserve kinetic energy, see Figure 8. In order to guarantee relaxation we implement the following algorithm:

1. Initialise and normalise simulation
2. Set `int: relaxation_steps`. This should be larger for high temperature or low density fluids since reaching equilibrium and propagating information through them takes longer respectively. We guess the number of `relaxation_steps` needed based on Equation A.1, which works sufficiently well.
3. Set a relaxation threshold $\mathcal{E}_{threshold}^{\lambda}$, which we set to $0.01 T_{target}$ due to the same reasons as above.
4. Run the simulation for `relaxation_steps`

5. Estimate the current kinetic energy of the system:
   (a) Try to fit a sine curve to the last local minimum and maximum and find the kinetic energy using the offset and amplitude of the curve. This is a good solution, especially for systems close to their equilibrium since we observed that the initial dip and jump in kinetic energy generally bounds the final equilibrium kinetic energy of the system. This means that we do not need to propagate the system very far to get a very good estimate of the final scaling. See Figure 5 for this method in practice.
   (b) If above fails estimate the current kinetic energy as the mean between the last local minimum and maximum
6. Rescale the velocities of the particles according to Equation 15
7. Reset the current simulation progress
   (a) If we estimate that we simulate a solid or liquid we know that the equilibrium state will be with particles near the FCC described positions. We reset the particle positions to their initial positions and scale the initial velocity (consecutively) by the λ we found in the previous run(s)
   (b) If we estimate that the system is in a gaseous state there is no clearly defined positions in which the particles "should" be. Hence, we simply continue the simulation with rescaled velocities. This results in a progressive relaxation, shown in Figure 6, which takes much longer than for liquids and solids but is also guaranteed to converge.
8. Return to 4. unless the system is sufficiently relaxed.

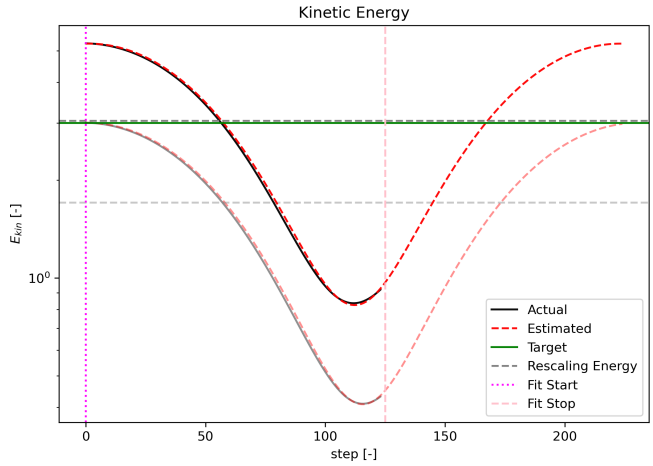

Fig. 5. Relaxation finding algorithm demonstrated on a liquid. We accurately evaluate the equilibrium kinetic energy after two short runs with high certainty. See Figure 8 and Figure 7 for an estimate of the effectiveness of this method for liquids and solids. The transparency of the lines shows how close the estimate is to the correct relaxation value.

3. Results

3.1. Conservation of Energy

Given that our aim is to simulate real physics, it would serve us well to make sure our simulation conserves energy. Figure 7 shows the total energy of a simulation for all three states of matter. Note that the scale on the y axis is very small, and the total energy is conserved within a reasonable margin for a numerical simulation.

The error on the energy values is considerably larger for the gas state than it is for the liquid, and similarly for the liquid and the solid. These errors are a result of the errors in kinetic energies (Figure 8). The gas case has a wider range of relaxation states it scales into than the other two have. This makes sense, the gas case should have a higher entropy, and therefore more possible states. That being said, it appears that total energy the gas state also diverges farthest from the initial total energy, perhaps indicative of an issue with the velocity re-scaling or our choice of timestep.

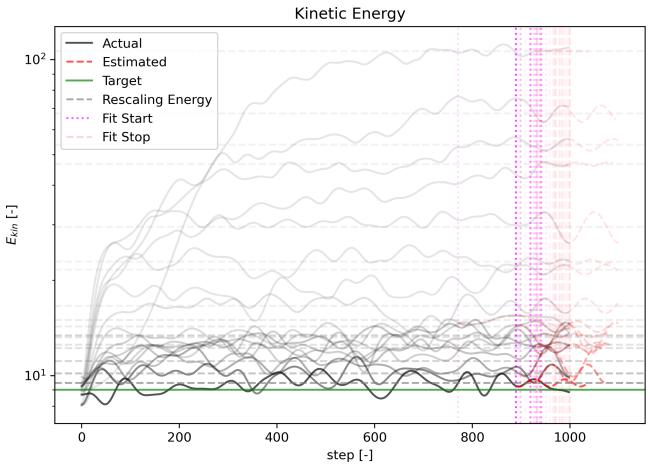

Fig. 6. Relaxation finding algorithm demonstrated on a gas. We accurately evaluate the equilibrium kinetic energy after multiple runs, while this method is guaranteed to converge the relaxation of a gaseous medium took a long time in our simulations, hence many Gas simulations have a small drift in kinetic energy since they are not completely relaxed when the simulation is started. See Figure 8 and Figure 7 for an estimate of the effectiveness of this method for gases. The transparency of the lines shows how close the estimate is to the correct relaxation value. It is clear that this method could be improved by scaling the `relaxation_steps` by how relaxed the system already is.

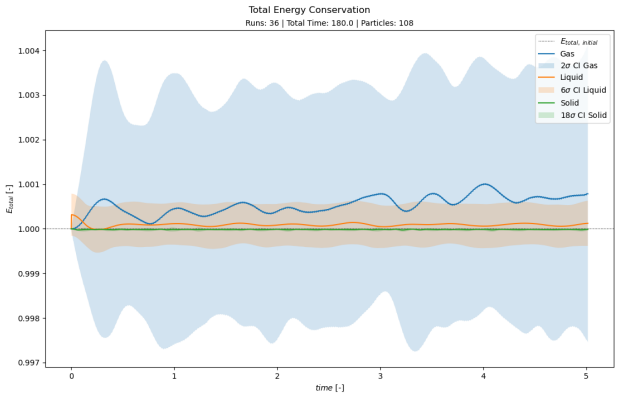

Fig. 7. Total energies over simulation time for all 3 states of matter. Note the small y axis scale. The grey dotted line represents the initial total energy of the system.

3.2. Pair Correlation

One convenient way to distinguish between different states of matter in the system is by using a pair correlation function. Pair correlation reflects patterns in data distributions. Simply put, given a reference particle, the pair correlation function gives the relative probability of finding a second particle at a given distance from the reference. Given a histogram distribution of inter-particle distances $n(r)$, the pair correlation is calculated as follows:

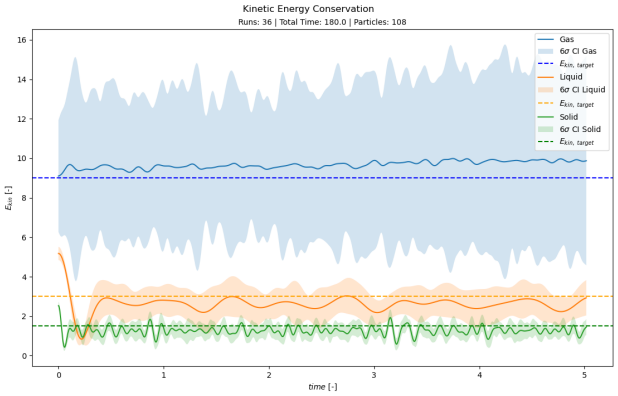

Fig. 8. Kinetic energies of the 3 states of matter over simulation time.

$$g(r) = \frac{2V}{N(N-1)} \frac{\langle n(r) \rangle}{4\pi r^2 \Delta r} \tag{16}$$

Where V is the simulation volume, $\langle n(r) \rangle$ is an average of histograms over several runs, N is the number of particles, r is the starting location for a given bin, and Δr is the bin width. The denominator has a factor of $N(N-1)$ instead of N^2 because the particle does not interact with itself. It should be noted that for our simulations, this is calculated in dimensionless units, so $r = \tilde{r}$, and $V = \tilde{r}^3$.

The pair correlation function can give us a sense of the state of matter we are simulating by showing us how "ordered" the argon atoms in our simulation are. In a system that is totally homogeneously distributed, the pair correlation function would equal 1 at all r. An example of the expected relative shapes for our pair correlation in each state of matter can be found in Figure 9.

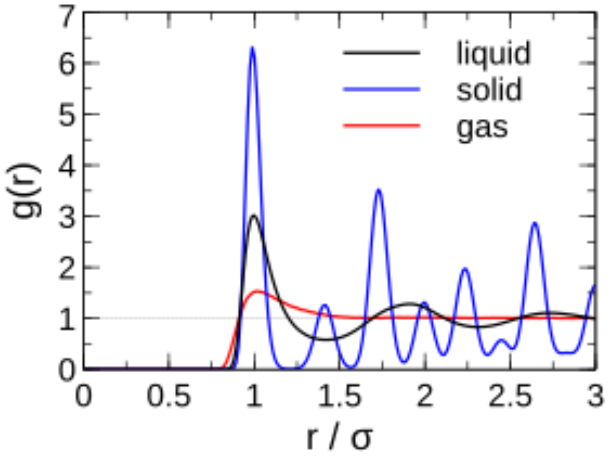

Fig. 9. Example expected pair correlation distributions for solid, liquid, and gaseous argon. Specific input values differ from those used in our simulations, but general trends regarding pair correlation and structure still hold. Rowley (2015)

In this plot, the solid case shows distinct peaks at values of $\sqrt{n}\sigma$ for positive integers n. In the case of our simulation, these positions are shifted slightly due to the truncated potential as outlined in subsection 2.3. The spread in these peaks are a result of the atoms vibrating around their lattice position (a model at absolute zero would have peaks of zero width).

In the liquid case, the solid peaks become broader and less well defined. As atoms begin to break from the lattice and slide past each other, the probability of finding them at "in-between" distances increases.

The gas case shows how the system trends toward the completely homogeneous limit (uniform distribution with value 1) at high enough temperatures. The primary features expected in the gas case are a peak just beyond 1σ that trends to 1 as r increases. This peak is the result of the minimum in the Lennard-Jones potential.

An average of 36 of our simulations (Figure 10) shows much the same trend in $g(r)$ as seen in Figure 9. The solid ($\rho = 1.2$, $T = 0.5$) case shows very thin peaks at the expected locations based on the truncated potential (subsection 2.3). The liquid ($\rho = 0.8$, $T = 1.0$) case shows the peaks beginning to shift and broaden. Comparing the gas ($\rho = 0.3$, $T = 3.0$) case to Figure 1, we can see that the strong initial peak is at the same location as the Lennard-Jones potential minimum. The distribution then quickly drops off, generally trending toward $g(r) = 1$.

That being said, while the general trend we see from one state of matter to another is promising, our pair correlation functions are not exactly as we'd expect. The peaks in the solid case are very, almost unexpectedly, thin, approaching what we would expect at absolute zero. While not necessarily a red flag on its own, this becomes suspect in the context of the other two plots. First, the spread of the peaks in the liquid case is not wide enough to indicate the particles are sliding past each other, moving freely from one potential well to the next. Additionally, while the gas case does spread out toward 1 at increased r, there is still more peak structure than would be expected. Beyond the initial peak at the Lennard-Jones minimum radius, we would generally expect little to no preferential organisation in a true gas. While the exact cause of these issues cannot be pin-pointed for certain, we believe they could all be caused by a relative lack of kinetic energy in the system. Given more time, our next steps would be to reevaluate the way we relax the system and scale the initial velocities (as outlined in subsubsection 2.5.2), and to see if there is any impact from the truncated shifted Lennard-Jones.

3.3. Pressure

Another assessment of the different states of matter is the system pressure. For our simulations, an equation for the local system pressure can be derived from the virial theorem as outlined in Lion and Allen (2012).

$$\frac{P}{k_B T \rho} = 1 - \frac{1}{3Nk_B T} \left\langle \frac{1}{2} \sum_i \sum_{j>i} r_{ij} \frac{\partial U\left(r_{ij}\right)}{\partial r} \right\rangle \tag{17}$$

Where N is the number of particles, k_B is the Boltzmann constant ($k_B = 1$ in our dimensionless units), T is the temperature, ρ is the density, and the term in the angled brackets is an average over several simulation runs. In this average, each pair of particles is summed over (the factor of 0.5 input to avoid double-counting) where r_{ij} is the distance between the particles in a given pair and the partial derivative of U with respect to r is the force between the two.

The first term in the right-hand-side of the pressure equation represents the value for an ideal gas, while the second term represents correction factors to account for particle interactions. Because of this, we expect the value to go to 1 as the system becomes more like an ideal gas. This also means we expect the second term to increase and the pressure to trend down as the system becomes less like an ideal gas.

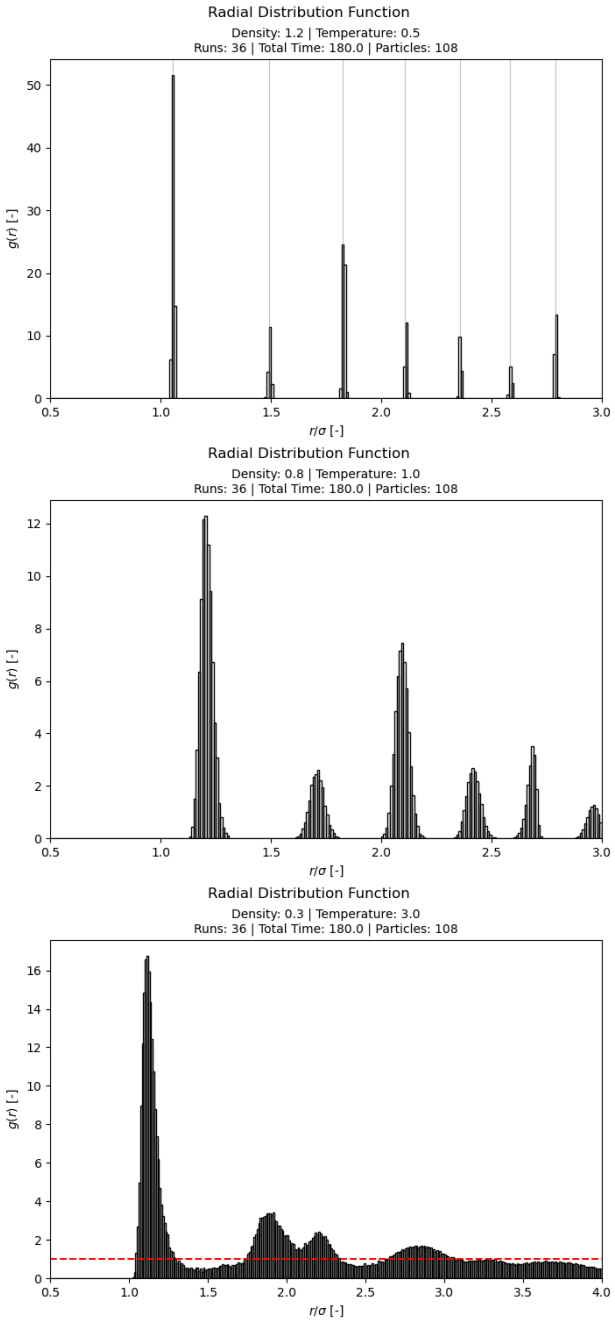

Fig. 10. Simulated radial distribution functions (averaged over 36 runs) for initial conditions defining solid (top), liquid (middle), and gas (bottom) cases. The vertical lines on the solid plot represent the expected peak locations based on the truncated potential outlined in subsection 2.3. The horizontal red line in the gas plot represents the uniform distribution of a perfectly homogeneous case.

Averaging over 36 simulations, we find the following pressure values (rounded to 3 decimal places):

Solid:	0.084
Liquid:	0.723
Gas:	0.897

As expected from Equation 17, the solid is the furthest from the ideal gas case, and the gas is the nearest. Figure 11 also shows that our pressures do not fluctuate much over time as the simulation runs.

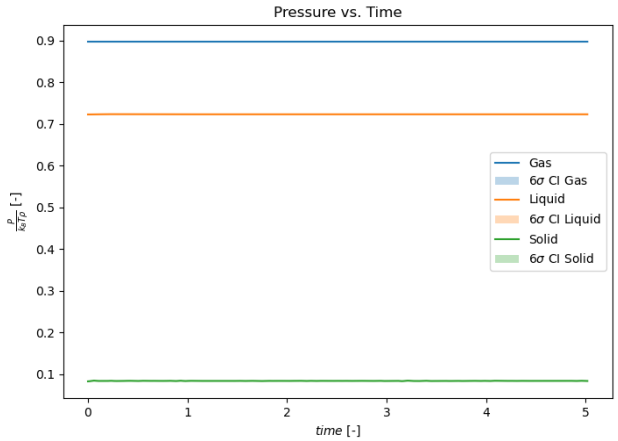

Fig. 11. Pressures for the three states of matter averaged over many runs

4. Summary

Using a Velocity-Verlet numerical integrator (adjusting the sphere of influence for large simulations), and minimal image convention, we simulate a system of argon atoms in three states of matter with tentative success. We initialise our simulation creating a Maxwell velocity distribution re-scaled to a given input temperature. We correctly conserve system energy within a 2σ range $\pm < 4e-3$ in the most extreme case. We show that the pair-correlation function spreads out and trends toward the totally homogeneous limit as we shift from solid to liquid to gas. There is some concern with the amount of structure of the gas and liquid pair correlation functions, most likely caused by an imbalance in kinetic and potential energies. Future inquiry would certainly begin with an assessment of this. Finally, we show that (averaged over many simulations), the pressure of the gas is closest to the ideal gas case at 0.897, the liquid less so at 0.723, and the solid the least at 0.084.

References

Deiters, U. K. (2013). Efficient coding of the minimum image convention. *Zeitschrift für Physikalische Chemie*, 227(2-3):345–352.

Lion, T. W. and Allen, R. J. (2012). Computing the local pressure in molecular dynamics simulations. *Journal of Physics: Condensed Matter*, 24(28):284133.

Rowley, C. (2015). Simulated radial distribution functions for solid, liquid, and gaseous argon.

Stephan, S., Liu, J., Langenbach, K., Chapman, W. G., and Hasse, H. (2018). Vapor- liquid interface of the lennard-jones truncated and shifted fluid: Comparison of molecular simulation, density gradient theory, and density functional theory. *The Journal of Physical Chemistry C*, 122(43):24705–24715.

Verlet, L. (1967). Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Phys. Rev.*, 159:98–103.

Appendix A: Additional Equations

Appendix A.1: Relaxation & Rescaling

$$\texttt{int:}\ \min\left(100 \cdot T_{target}\rho_{target}^{-1}\right) \tag{A.1}$$

Appendix B: Extended Tables

Property	Dimensionless Form
Distance	$\tilde{r} = \frac{r}{\sigma}$
Time	$\tilde{t} = t/\sqrt{\frac{m\sigma^2}{\epsilon}}$
Temperature	$\widetilde{T} = \frac{k_B T}{\epsilon}$
Force	$\tilde{F} = \frac{F\sigma}{\epsilon}$
Potential	$\tilde{U} = \frac{U}{\epsilon}$
Pressure	$\tilde{P} = \frac{P\sigma^3}{\epsilon}$
Density	$\tilde{\rho} = \rho\sigma^3$

Table B.1. Equations for converting important properties for this simulation from SI into dimensionless units.

Appendix C: Extended Figures

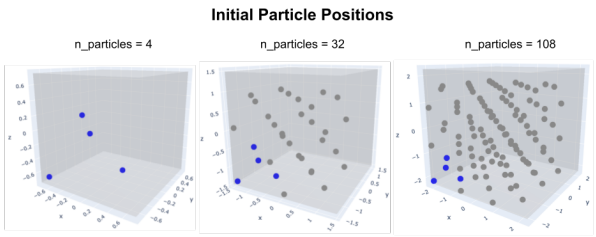

Fig. C.1. Example of the initial position FCC for 4, 32, and 108 particles. Output from the simulation code. The first representative unit is plotted in blue for convenience.

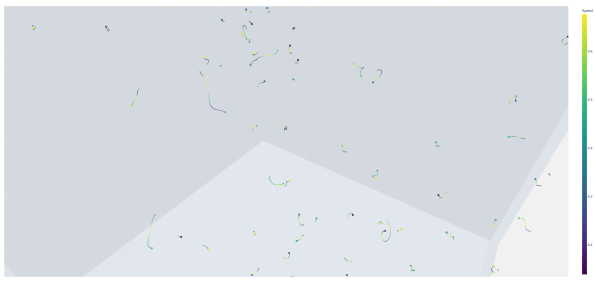

Fig. C.2. An example gas simulation, zoomed in to show paths.