École polytechnique fédérale de Lausanne

CH-420. Understanding Advanced Molecular Simulation

Spring 2022

Exercises block 2

bruno rodriguez carrillo

Sciper 326180

Professor:
Berend Smit

May 31st, 2022

1 Calculation of π

1. From the Block02 problem sheet, we have figure 1.

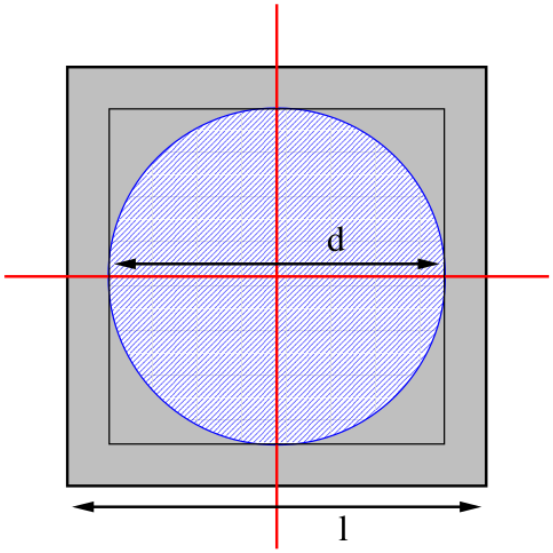


Figure 1: Computation of π via Monte Carlo.

In order to compute the value of π, from picture (1), we have that the ratio of the area of the circle S_{circle} and the area of the square S_{square} can be computed by:

$$\frac{S_{\text{circle}}}{S_{\text{square}}} = \frac{\pi}{4} \cdot \left(\frac{d}{l}\right)^2 \longrightarrow \pi = 4 \cdot \frac{S_{\text{circle}}}{S_{\text{square}}} \cdot \left(\frac{l}{d}\right)^2$$

From the above expression, we observe that we can approximate the value of π by 'counting' how much area of the circle falls inside the square. Then, we can use the Monte Carlo method to count.

It is important to mention that the S_{square} is the total number of points generated. That is S_{square} = points inside the circle + points outside the circle.

From the above formula, we can view the ratio $\frac{S_{circle}}{S_{square}}$ as a probability of any given point to fall inside the circle. Additionally, the ratio $\frac{l}{d} \geq 1$ since the whole circle must be inside the square. As a result, if we were to modify the diameter of the circle, such a ratio will decrease and in the limit when $d = 0$, we have $\pi = 0$: a circle whose radius is zero. However, changing the position of the circle –as long as it is completely contained in the square – will have no effect on the the computed value of π. That is, the smaller d is, the more points we need for our estimation of π to improve: many points should be computed in such a manner that the whole area of the circle contains more points, which increases the computational cost. Such

points are generated in a random fashion and here it is when we make use of the Monte Carlo method.

As a result, we approximate our value of π by generating a set $\mathcal{U}$ of uniform distributed points in the interval $[0, l]^2$. We say that a point falls inside or on the perimeter of the circle if $u_1^2 + u_2^2 \leq (d/2)^2$ for any $u = (u_1, u_2) \in \mathcal{U}$. In a more formal approach, we can view the problem as follows:

$$\text{point\_inside}(u) = \mathbb{1}(u) = \begin{cases} 1, \text{ if } u_1^2 + u_2^2 \leq (d/2)^2 \\ 0, \text{ otherwise} \end{cases}, \text{ for all } u \in \mathcal{U}$$

Then, in fact, we compute the expected value of point_inside to compute the expected value of our approximation to π; that is:

$$\mathbb{E}\left[\text{point\_inside}(u)\right] = \frac{1}{N} \sum_{j=1}^{N} \mathbb{1}(u^{(j)}), \text{ for all } u^{(j)} \in \mathcal{U}$$

Thus our approximation of π can be written as:

$$\tilde{\pi} = 4 \cdot \left(\frac{l}{d}\right)^2 \cdot \mathbb{E}\left[\text{point\_inside}(u)\right] = \frac{4}{N} \cdot \left(\frac{l}{d}\right)^2 \cdot \sum_{j=1}^{N} \mathbb{1}(u^{(j)}), \text{ for all } u^{(j)} \in \mathcal{U} \tag{1}$$

2. To observe the impact of $\frac{l}{d}$ and on N the relative error, we run a few numerical experiments for different values of such quantities, which leads to figure (2).

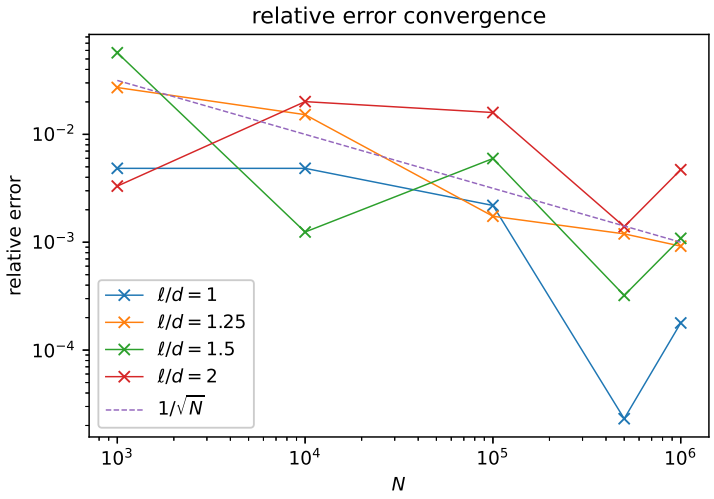


Figure 2: Relative error convergence for distinct values of ℓ/d and N.

From figure (2), we visually corroborate that the relative error converges at a $1/\sqrt{N}$ rate – plotted in dashed lines and discussed in points below. Moreover, we clearly observe that the smallest relative error occurs we use a ratio $\ell/d = 1$. The wavy behavior observed in the same plot is due to the fact that we are working with random numbers but it is important to bear in mind that the theoretical rate holds.

3. Since point_inside$(u) = \mathbb{1}(u)$ can be either 0 or 1, its probability density function follows a binomial distribution and we assume that the probability of 0 and 1 are equal to 0.50; that is:

$$\mathbb{P}\left\{\text{point\_inside}(u) = 1\right\} = \mathbb{P}\left\{\text{point\_inside}(u) = 0\right\} = 0.50 = p = q$$

where p and q are the probabilities of success and fail, respectively.

From point 2), we observe that point_inside(u) follows a Bernoulli distribution; consequently, $Y = \sum_{j=1}^{N} \mathbb{1}(u^{(j)})$ follows a Binomial distribution since it is the sum of N independent Bernoulli random variables. That is, $Y \sim \text{Bin}(N, p)$, where p is the probability of a point hitting inside the circle.

This means that our approximation to π follows also a Binomial distribution as indicated by (2).

$$\tilde{\pi} = \frac{4}{N} \cdot \left(\frac{l}{d}\right)^2 \cdot Y \sim \frac{4}{N} \cdot \left(\frac{l}{d}\right)^2 \cdot \text{Bin}(N, p). \tag{2}$$

Consequently, it can be shown that the expected value, variance, and standard deviation *std* of $\tilde{\pi}$ can be computed as:

$$\mathbb{E}\left[\tilde{\pi}\right] = 4\left(\frac{l}{d}\right)^2 p, \text{ , } \text{Var}\left[\tilde{\pi}\right] = \frac{16}{N}\left(\frac{l}{d}\right)^4 p(1-p), \text{ and } std\left[\tilde{\pi}\right] = \frac{4}{\sqrt{N}}\left(\frac{l}{d}\right)^2 \sqrt{p(1-p)}$$

From the expression for $std\left[\tilde{\pi}\right]$, we observe that for such a quantity we have

$$\frac{4}{\sqrt{N}}\sqrt{p(1-p)} \leq \frac{4}{\sqrt{N}}\left(\frac{l}{d}\right)^2 \sqrt{p(1-p)},$$

since we take into account that the circle must be contained fully inside the square; in other words, $0 \leq d \leq l$, which means that the minimum value of the ratio $\dfrac{l}{d} = 1$. This lead us to stating that the minimum value of the standard deviation is reached when $l = d$.

4. From point 3), we observe that $std\left[\tilde{\pi}\right]$, as a function of the number of generated coordinates, decreases at a rate $1/\sqrt{N}$. This means that if we required a reduction of 50% in the standard deviation from an initial simulation with N samples, we would need to increase the number of samples to $4N$ to achieve such a goal. That is, we would need to quadruply the number of samples to halve the standard deviation.

   As a consequence, having an approximate computation of π with many decimals is not a good idea because too many samples could be required, which means having a higher computational cost.

2 The photon gas

We can compute the average occupancy of state j of the photon gas ($\langle n_j \rangle$). From quantum mechanics, we know that the total energy of the system, denoted by U, can by written a the sum of the energies of harmonic oscillators:

$$U = \sum_{j=1}^{N} n_j \omega_j \hbar = \sum_{j=1}^{N} n_j \epsilon_j$$

where ϵ_j is the characteristic energy of oscillator j, $n_j = 0, 1, 2, \ldots$ is the occupancy number of oscillator j, N is the number of oscillators, ω_j the frequency, and $2\pi\hbar$ is the Planck constant.

In this exercise, we compute $\langle n_j \rangle$ via the Monte Carlo method.

From the lecture notes, we observe that in the Metropolis-Hastings algorithm, we accept the trial move with probability

$$\text{acc}\,(\text{o} \to \text{n}) = \min\left(1, e^{-\beta[U(n)-U(o)]}\right) = \min\left(1, e^{-\beta \Delta U}\right),$$

where $U(o)$ and $U(n)$ correspond to the energy values at old and new states, respectively.

From such an expression, we notice that we accept the move if $\Delta U < 0$. This means that we accept a new state n corresponds to a state such that the energy is lower than the current (old) state o. Moreover, this implies that we move to those states with lower energy.

On the other hand, if $\Delta U > 0$, we accept a new state n with probability $e^{-\beta \Delta U}$. In other words, we accept if $r < e^{-\beta \Delta U}$, where $r \sim U([0, 1])$ follows and uniform distribution in the interval 0 and 1.

As a result, the probability of acceptance a new state n is described by

$$\mathbb{P}_{acc}\,(\text{o} \to \text{n}) = \begin{cases} e^{-\beta[U(n)-U(o)]}, \text{ if } U(n) \geq U(o) \\ 1, \text{ if } U(n) < U(o) \end{cases},$$

1. The proposed scheme obeys detailed balanced if we always reject the move whenever $n_j < 0$. In other words, the acceptance probability of such a move is zero, i.e. $\mathbb{P}_{acc}(o \to n = \{n_j < 0\}) = 0$. From the lecture notes, we know that detailed balance corresponds to $K(o \to n) = K(n \to o)$, where $K(A \to B)$ is the number of systems moving from state A to state B; then we have

$$\begin{gathered} \text{number systems from old o to new state n} = \text{number systems from old n to new state o} \\ \iff N(o) \cdot \mathbb{P}_{\alpha}(o \to n) \cdot \mathbb{P}_{acc}(o \to n) = N(n) \cdot \mathbb{P}_{\alpha}(n \to o) \cdot \mathbb{P}_{acc}(n \to o), \end{gathered}$$

   which is zero if $\mathbb{P}_{acc}(o \to n = \{n_j < 0\}) = 0$. Furthermore, we should mention that such a move is always rejected since such a configuration does not belong to the phase space. Thus, detailed balance holds.

2. The algorithm is correct if trial moves change the occupancy number of state j n_j in an interval $[-5, 5]$. The reason behind this is that even if we restrict ourselves to trial moves within such an interval, it would be possible to visit all occupancy states n_j; that is, we fulfill

ergodicity when sampling each reachable state in a finite number of steps. But this algorithm could take longer time to sample the whole phase space.

However, when the trial moves take either -3 or $+3$, the algorithm would not visit all reachable states since it would be visiting states that are just multiples of ± 3, leaving out all other states. Under this condition, our results are erroneous.

3. From this point on, we assume that we are working with single oscillator, which means that $N = 1$ and $\epsilon_j = \epsilon$.

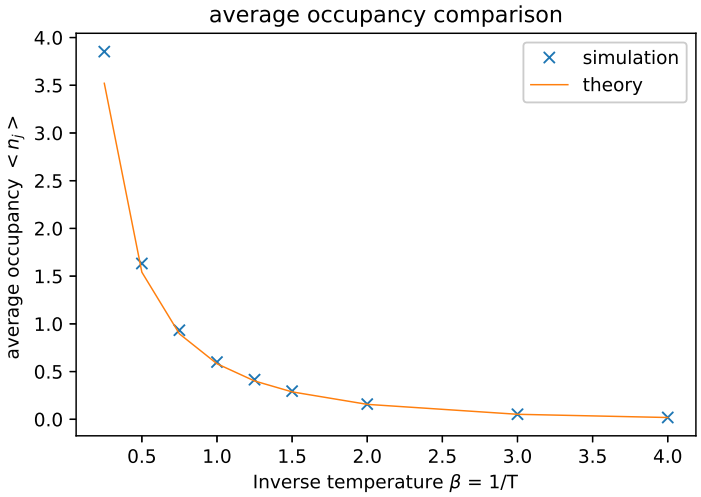


Figure 3: Theoretical and simulated average occupancy of state j, n_j.

Furthermore, the analytical solution for $\langle n \rangle$ is given by:

$$\langle n \rangle = \frac{1}{e^{\beta\epsilon} - 1},$$

which corresponds to the solid line shown in figure 3. We note that, at least, graphically, our results are congruent with what it is predicted by theory. Our sampled average occupancy via simulation matches almost perfectly the theoretical results.

4. After modifying the corresponding section in the code, we obtain figure 4.

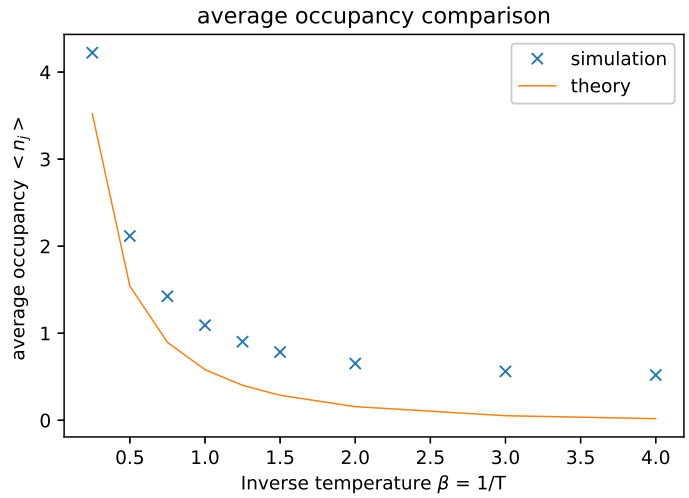


Figure 4: Theoretical and simulated average occupancy of state j, n_j.

It is worth noticing that the error we obtain in the simulation increases as the value of β increases, as can be corroborated visually in figure 5.

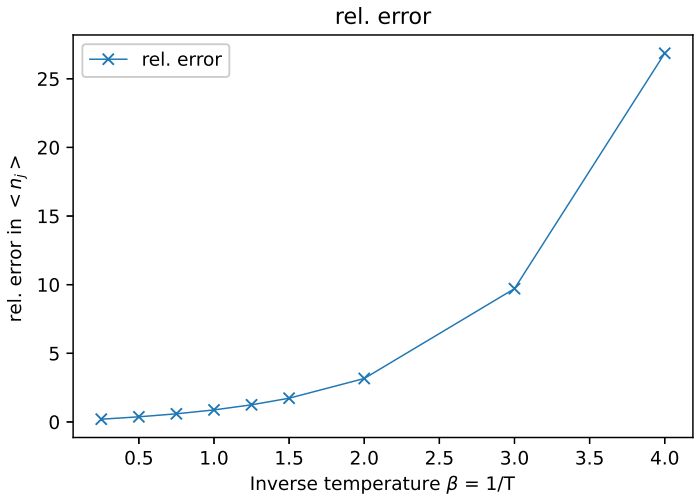


Figure 5: Error behaviour for values of j, n_j during simulation.

When we update the average when a trial move is accepted we are clearly inserting a bias to the result. To illustrate this fact, let us think about what an arithmetic average is in its simplest form. It is a sum of a set of N numbers divided by N itself. We keep this idea in

mind to state that when we update the average only when we accept a given trial move, we are actually leaving out all unaccepted moves, which means that the final average value is wrong since it considers only a fraction of the total samples. In addition, it should noted that when the average value is updated only after an accepted trial move, we intrinsically assume that the probability of not moving is zero, which is not true as shown in [1].

Besides, the errors are more evident as β increases since the higher the value of β, the smaller the value of temperature T is.

Staring from a high T allows the system to explore many different configurations. However, after a few iterations at high T in the MH algorithm, the system will start cooling, which means that the lower T, the more iterations the system will usually need to explore all configurations. This also means that generating a new configuration different from the current one is unlikely.

5. As given in the lab sessions, we have that the analytical solution for the distribution of $n_j$is given by (3).

$$P = \frac{\exp\left(-n_j \beta\epsilon\right)}{Q}, \text{ for } Q = \frac{1}{1-\exp\left(-\beta\epsilon\right)}, \tag{3}$$

where Q is a normalization constant.

After modifying and running the simulation for the same values of $\beta\epsilon$ used in previous points, we have figure, which shows a good agreement between theoretical and simulation results.

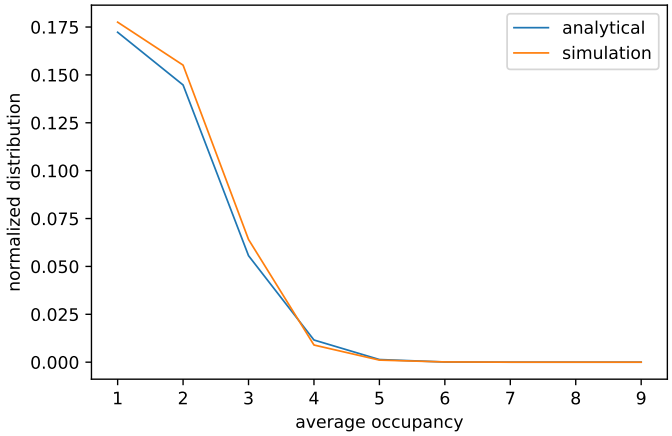


Figure 6: Distribution of n_j.

3 MC of a Lennard-Jones system

In this exercise, we study a 3D Lennard-Jones NVT system. A cubic box of volume V encloses N particles at a given temperature T in any given configuration allowed by the potential energy U, which is defined as

$$U(r) = \begin{cases} 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right], \text{ if } r \leq r_c \\ 0, \text{ if } r > r_c. \end{cases} \tag{4}$$

Where we make use of the following tail corrections,

$$\frac{u^{tail}}{N} = \frac{8}{3}\pi\rho\epsilon\sigma^3 \left[\frac{1}{3}\left(\frac{\sigma}{r_c}\right)^9 - \left(\frac{\sigma}{r_c}\right)^3 \right] \tag{5}$$

$$p^{tail} = \frac{16}{3}\pi\rho^2\epsilon\sigma^3 \left[\frac{2}{3}\left(\frac{\sigma}{r_c}\right)^9 - \left(\frac{\sigma}{r_c}\right)^3 \right] \tag{6}$$

$$\mu^{tail} = \frac{1}{\beta}\frac{16}{3}\pi\rho\epsilon\sigma^3 \left[\frac{1}{3}\left(\frac{\sigma}{r_c}\right)^9 - \left(\frac{\sigma}{r_c}\right)^3 \right] \tag{7}$$

1. We start writing the equation for pressure p as

$$p = \frac{\rho}{\beta} + \frac{\text{vir}}{3V}, \text{ where vir} = -\frac{\partial U}{\partial r} \cdot r. \tag{8}$$

   Using (4) in (8) and doing all computations we obtain the virial equation

$$\text{vir} = 48 \left[\left(\frac{\sigma}{r}\right)^{12} - 0.50 \left(\frac{\sigma}{r}\right)^{6} \right] \tag{9}$$

   Expression (9) corresponds to the modification in the code *energy.c*.

2. When we perform a simulation for a fixed $T = 2$ and density $\rho \in [0.10, 1.20]$ with steps of length 0.10. We obtain figure 7, where the x-axis is in log scale.

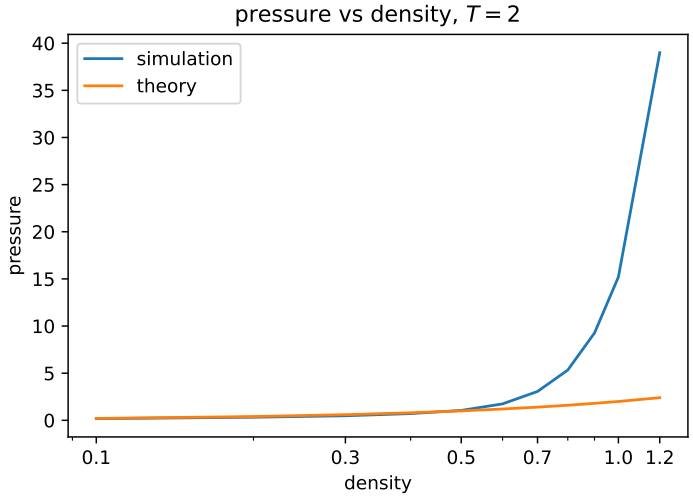


Figure 7: Pressure using the virial equation and the ideal gas law.

In figure 7, equation (10) is plotted in orange and represents the ideal gas law.

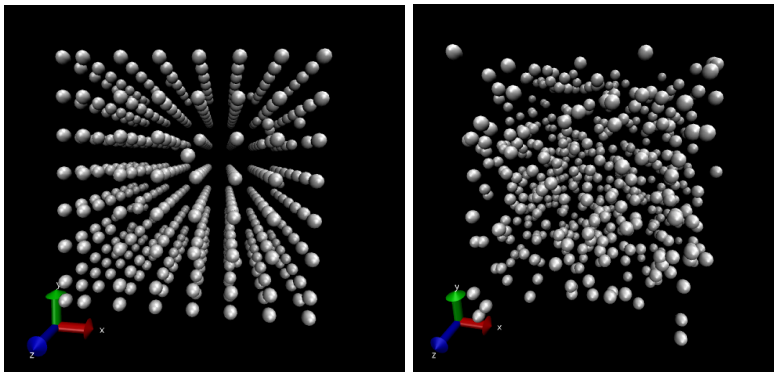
$$\beta p = \rho, \tag{10}$$

where p is the pressure.

From the same figure, we can clearly note that for small values of density, the virial equation is actually able to describe the relation -a linear relation- between the pressure and the density, plotted in blue, as the ideal gas law does, plotted in orange. We could explain this fact as an effect of how particles interact, for a fixed volume, while we increment the value of density. The larger the value of density, the more interactions particles have among them and then, the less we should assume the particles interact obeying the ideal gas equation.

This, in turn, means that if we want to analyze our system using the ideal gas approach, we should be careful when choosing which value of density to use. We may say that for $0 \leq \rho \leq 0.50$, we can think of our system following the ideal gas law.

3. After running our simulation, we have figure 8 .



(a) Initial configuration. (b) Final configuration.

Figure 8: Particles' configuration after running simulation.

4. The heat capacity C_V at constant volume is given by:

$$C_V = \frac{1}{N}\left(\frac{\partial U}{\partial T}\right)_V = \frac{\langle U^2\rangle - \langle U\rangle^2}{Nk_BT^2} \tag{11}$$

From the lecture notes, we know that to compute the ensemble average of a quantity of interest A, we have $\langle A\rangle = \frac{\sum_{i=1}^{N} U_i \exp\left(-\frac{U_i}{k_BT}\right)}{Z}$, where $Z = \sum_{i=1}^{N} \exp\left(-\frac{A_i}{k_BT}\right)$ is the partition function. Using this expression to compute the ensemble average of the energy U, taking the equation for the heat capacity at constant volume given in the problem sheet, and computing derivatives, we obtain

$$\begin{aligned} C_V = \frac{1}{N}\frac{\partial\langle U\rangle}{\partial T} &= \frac{1}{k_BT^2N}\left(\frac{\sum_{i=1}^{N} U_i^2 \exp\left(-\frac{U_i}{k_BT}\right)}{Z} - \frac{\left(\sum_{i=1}^{N} U_i \exp\left(-\frac{U_i}{k_BT}\right)\right)^2}{Z^2}\right) \\ &= \frac{\langle U^2\rangle - \langle U\rangle^2}{k_BT^2N}. \end{aligned} \tag{12}$$

To derive the dimensionless heat capacity, which we call C_V^*, we divide (12) by k_B to get

$$C_V^* = \frac{C_V}{k_B} = \frac{\langle U^2\rangle - \langle U\rangle^2}{T^2N}. \tag{13}$$

5. As we know, when we make a trial move, this is done by picking a particle at random and then we compute the energy change to compare such a change to a given probability. We accept this trial move with probability $\mathbb{P}\left(\text{old} \to \text{new}\right) = \min\left(1, \exp(-\beta\Delta U)\right)$. Now, let us imagine that we pick all particles and perform a trial move of all them randomly. From this

we could state that changing all particles' positions in a trial move is highly unlikely to be accepted since this would lead to abrupt changes of energy from the current to the new state. Furthermore, this means the efficiency in sampling could be significantly inefficient because moving all particles at the same time could never be an actual configuration; that is, our system would hardly move from its current to the proposed state. It is also important to note that when we perform a trial move for all particles, such a move could even lead to particles' overlapping, states with probability zero, system states that are impossible to happen.

6. We start by looking at how a standard normal and a uniform distribution look like in the same interval, figure 9. We observe that the probability of a small displacement is higher that the probability of having a large displacement in the normal distribution. this happens due to the fact that this is centered around 0; 68 % of the trial moves would be within 1 standard deviation from 0. In other words, this means that it could take too many trials to actually displace a particle, which leads to loosing efficiency in sampling.

   However, when we sample using the standard normal distribution, the probability of a large displacement is low and this contributes to not rejecting too large trial moves, which means our sampling is efficient. It is worth noticing that in figure 9, we have a standard Gaussian distribution and then we could change its parameters, mean and variance, to have a "better-behaved" distribution. That is, if we increase the variance to a value such that the displacement is within a given interval, we would expect a similar efficiency as if we used the uniform distribution.

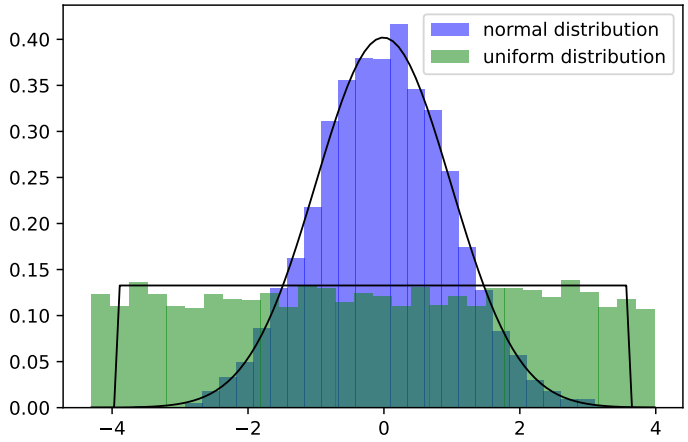


Figure 9: Standard normal and uniform distribution.

4 MD of a Lennard-Jones system

1. There are three errors in the code, which once corrected, are shown in figure 10.

```
UPotential+=4.0*r6i*(r6i-1.0)-Ecut;
Ff=48.0*r6i*(r6i-0.5);
Pressure+=Ff;
Ff=Ff*r2i;
```

(a) Error in force computation.

```
NewPositions[i].x=OldPositions[i].x+2*Velocities[i].x*Deltat; // Verlet algorithm
NewPositions[i].y=OldPositions[i].y+2*Velocities[i].y*Deltat;
NewPositions[i].z=OldPositions[i].z+2*Velocities[i].z*Deltat;
```

(b) Error in Verlet algorithm to compute positions.

```
// add the kinetic part of the pressure
Pressure+=2*UKinetic*NumberOfParticles/(CUBE(Box)*(3*NumberOfParticles - 3));
```

(c) Error in computation of pressure due to momentum conservation.

Figure 10: Errors in code

The Verlet integration was wrong since there was a missing 2 factor for the velocity computation. The force implementation was not correct since there was a power of 6 for σ missing in the formula implemented in the code. Finally, when computing the pressure, the conservation of momentum did not hold.

2. In fact, we control the temperature in the given program when we re-scale all velocities such that the desired temperature is reached. This is done in the file init.c as observed in figure 11.

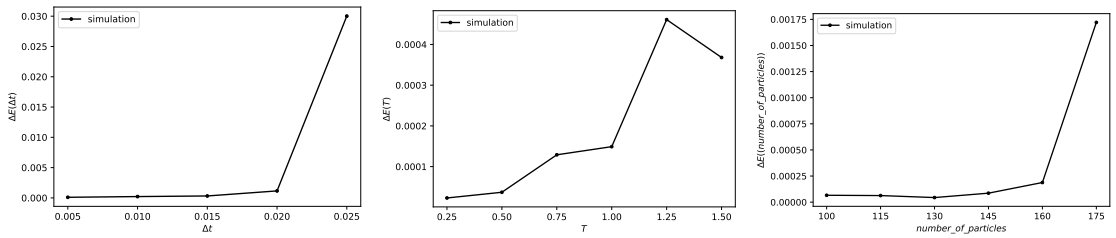
```
// scale all velocities to the correct temperature

scale=sqrt(Temperature*(3.0*NumberOfParticles-3.0)/UKinetic);
for(i=0;i<NumberOfParticles;i++)
{
  Velocities[i].x*=scale;
  Velocities[i].y*=scale;
  Velocities[i].z*=scale;
}
```

Figure 11: Velocity re-scaling.

3. It should be pointed out that we used (14) to compute the energy drift without using the absolute.

$$\Delta E(\Delta t) = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{E(0)-E(i\Delta t)}{E(0)}\right| \tag{14}$$

We then plot (14) as a function of the time step Δt, temperature T, and density ρ (by changing the number of particles in our system); this leads to figure 12.



(a) ΔE as function of Δt. (b) ΔE as function of T. (c) ΔE as function of ρ.

Figure 12: ΔE variation for different parameters.

In figure 12, for a), b) and c), we use $T = 0.50$ and number of particles set to 100, $\Delta t = 0.005$ and number of particles set to 100, and $T = 0.50$ and $\Delta t = 0.005$, respectively.

From figure 12, we clearly observe that a) keeping a certain energy drift depends on having a sufficiently small time step Δt. It is worth mentioning that the energy drift remains around a given value for an interval of Δt; thus we may state that the energy drift remains bounded as a function of Δt. This result follows what it is encountered in theory. We know that the Verlet algorithm is a symplectic, and as it is proven in [3], when integrating the equation of motion using such an algorithm, the energy drift remains bounded by a function Δt. In addition, when we ran the simulation, for $\Delta > 0.30$ approximately, we found that the time step is not small enough and the energy drift was not finite anymore. We observe a similar "bounded" behavior in the same figure c), when we see how the energy drift varies as a function of ρ.

On the other hand, in figure 12, b), we can notice that even thought the energy drift remains bounded as T increases, we do not observe a clear value of or interval of T around which ΔE remains fluctuating. We could then say that the larger the value of temperature is, the smaller the time step should be; that is, if we plan on viewing a "flat" trend, like the one observed when changing density and time step, a) and c) respectively, for a temperature interval, we would be required to decrease the time step as we increase temperature. In other words, we do not certainly see a temperature interval for which ΔE remains moving around.

4. In the code, the implementation of the periodic boundary conditions is written as $x = x - \text{box} \cdot \text{nint}(x \cdot \text{ibox})$, where ibox is used instead of $1/\text{box}$, due to efficiency when executing the code. That is, multiplication is cheaper than division in terms of CPU usage. In programming, we can split how expensive mathematical operation on the CPU are into three categories: basic operations, modulus and division operations, and complex functions such as trigonometric or power functions.

   Multiplication is a basic operation on the CPU, and as a result, it is cheaper to compute than a division. Then, we shall implement multiplications instead of divisions when possible.

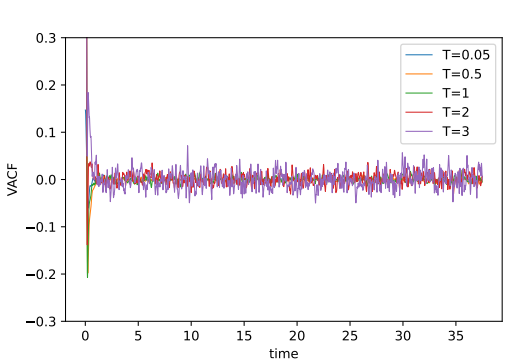
   However, it it quite important to keep in mind that nowadays the CPU power required to compute a division or a multiplication is practically the same due to modern CPUs. See [2] for more information.

5. From the problem description sheet, we have that we can compute the diffusivity coefficient D in two manners. The first one is through integrating the velocity auto-correlation function (15).

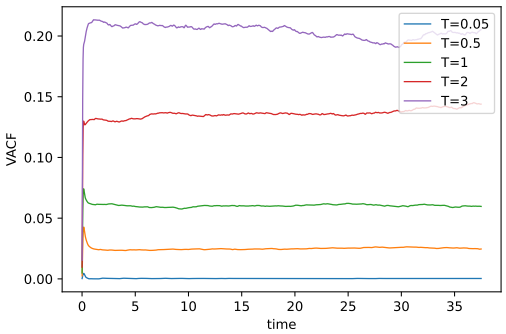
$$D = \frac{1}{3}\int_0^\infty \langle \mathbf{v}(t)\cdot\mathbf{v}(t+t')\rangle dt' = \frac{1}{3N}\int_0^\infty \sum_{i=1}^{N}\langle \mathbf{v}(i,t)\cdot\mathbf{v}(i,t+t')\rangle dt', \tag{15}$$

where N is the number of particles and $\mathbf{v}(i,t)$ is the velocity of particle i at time t. The time t is chosen in such a way that independent time origins are taken, i.e. $t = ia\Delta t, i = 1, 2, \ldots$ and $\langle \mathbf{v}(t)\cdot\mathbf{v}(t+a\Delta t)\rangle \approx \mathbf{0}$. The reason behind this is that we want D to be independent from the time origin that we pick. By doing this, we can actually rely more on our value of D computed via the VACF. From (15), we observe that D is actually computed as an average over multiple time origins, thus by having time-origin-independent samples would ensure having a better statistical error. It is also relevant to mention that the expression $\langle \mathbf{v}(t)\cdot\mathbf{v}(t+a\Delta t)\rangle \approx \mathbf{0}$ means having a zero correlation between two consecutive velocity values; this, in turn, means that our quantity is independent from time and time-origin.

We run our simulation for various values of temperature to obtain figure 13.



(a) VACF as a function of simulation time.



(b) Integral of VACF (value of D) as a function of simulation time.

Figure 13: Computation of D by the VACF.

Similarly, the values of D for T at final simulation time are summarize in table 1.

T	0.05	0.5	1	2	3
D	3.37e-04	2.4617e-02	5.9678e-02	1.4391e-01	2.053240e-01

Table 1: Value of D for different T via the VACF.

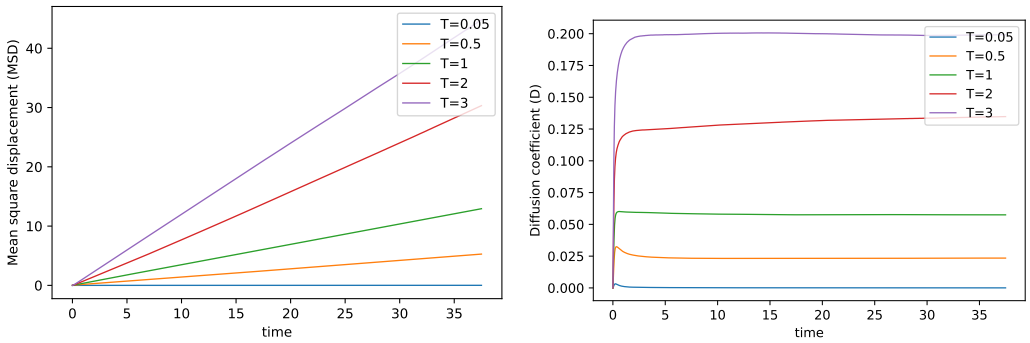
It is also worth discussing that the values for the VACF at initial simulation time are larger that the the ones computed afterwards since at the beginning of the simulation our particles are interacting more or our system is more chaotic and as time goes by, the system reaches a stable state. Furthermore, the VACF goes to zero since we expect our particles' motion to be more uncorrelated as time evolves, which implies a correlation function almost 0.

On the other hand, we can obtain D by computing the means squared displacement as given in (16).

$$D = \lim_{t' \to \infty} \frac{\langle (\mathbf{x}(t+t') - \mathbf{x}(t))^2 \rangle}{6t'}. \tag{16}$$

When we use (16) to compute D, we should be careful with periodic boundary conditions since when a particle leaves the box that represents our system, it does not come back and then, if this particle's position is again used to compute D, it does not make any sense to use its information since it would not be even inside our system. In addition, when periodic boundaries are used there are multiple copies of our system, thus we need to have all our particles involved in the computation D inside our system.

As previously done with the VACF, we create figure 14 for various temperatures.



(a) Mean squared displacement as function of simulation time.

(b) Diffusion coefficient as function of simulation time.

Figure 14: Computation of D by MSD.

And we have table 2 where we show the values of D for different T.

T	0.05	0.5	1	2	3
D	3.8e-05	2.346e-02	5.7486e-02	1.3477e-01	1.9917e-01

Table 2: Value of D for different T via the MSD.

From tables 1 and 2, we observe that the values of D computed by both methods are similar and then we may state that our results follow what we expected from theory.

From [1], we know that the macroscopic law that describes diffusion is called the Fick's law, which states that the flux $\mathbf{j}$ of a diffusing material is proportional to the negative gradient of the concentration $\mathbf{c}$ of the material, i.e, $\mathbf{j} = -D\nabla\mathbf{c}$, where the constant of proportionality is the so-called diffusion coefficient D. From the same reference, we have that D is measured in SI as m^2/s (length^2/time).

As shown in [1], suitable reduced units could be written in terms of ϵ, σ and m, for energy, position and mass, respectively. Then to come with a dimensionless D^* of D, we have (17). Defining, as given in the lab sessions, $\tau = \sqrt{\sigma^2 m/\epsilon}, L = \sigma$ and $t^* = t/\tau, x^* = x/L$, we have

$$D^* = D\tau/L^2 = D\sqrt{m/\sigma^2\epsilon}, \tag{17}$$

where ϵ is a unit of energy and m is the mass of the atoms in the system.

6. We confirm (18) as reported by Naghizadeh and Rice.

$$\log_{10}(D^*) = 0.05 + 0.07p^* - \frac{1.04 + 0.1p^*}{T^*}, \tag{18}$$

for $T^* < 1.0$ and $p^* < 3.0$.

When we vary $T^* \in [0.10, 1.0]$ for temperature steps $\Delta T = 0.10$, we compute a value of p^* for each T, then we directly use (18) to plot figure 15. Afterwards, we compute the value of D^* via the MSD for each T; then, we compute $\log(D^*)$; we get figure 15 as a result. We observe that the values of D^* computed by either of the methods are close to each other for values of temperature $T \geq 0.20$. We could say that for small values of T, the correlation between two consecutive positions in the simulation is high since the system evolves slowly and this may be the reason why the values for small T differ significantly.

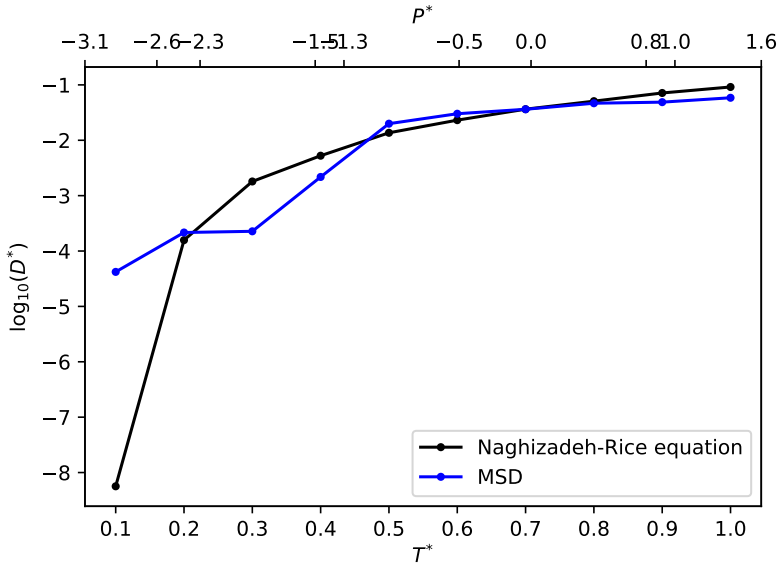


Figure 15: Graphical confirmation of Naghizadeh and Rice expression.

7. We know that the general correlation function $g^{(n)}$ for a system of N particles can be written as

$$g^{(n)}(\mathbf{r}_1, \ldots, \mathbf{r}_n) = \frac{\rho^n(\mathbf{r}_1, \ldots, \mathbf{r}_n)}{\rho^n} = \frac{V^n N!}{Z_N N^n (N-n)!} \int d\mathbf{r}_{n+1} \ldots d\mathbf{r}_N \exp\left(-\beta U(\mathbf{r}_1, \ldots, \mathbf{r}_N)\right), \tag{19}$$

where Z_N is the partition function, V the volume, and $\mathbf{r}_i$ the position of the i-th particle. We can compute the probability densities of N particles and $N(N-1)$ by realizing that we can actually count how many of them we have, i.e.

$$\rho_N^{(1)}(\mathbf{r}) = \left\langle \sum_{i=1}^{N} \delta\left(\mathbf{r} - \mathbf{r}_i\right) \right\rangle \text{ and } \rho_N^{(2)}(\mathbf{r}, \mathbf{r}') = \left\langle \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \delta\left(\mathbf{r} - \mathbf{r}_i\right) \delta\left(\mathbf{r}' - \mathbf{r}_j\right) \right\rangle. \tag{20}$$

Moreover, we have

$$\int \rho_N^{(1)}(\mathbf{r}^1) d\mathbf{r}^1 = N \text{ and } \int \rho_N^{(2)}(\mathbf{r}^2) d\mathbf{r}^2 = N(N-1), \tag{21}$$

that is, we can find N particles and $N(N-1)$ pairs of particles in the total volume, respectively. When $n = 2$ in (19), we have the so-called radial distribution function given by

$$g^{(2)}\left(\mathbf{r}_1, \mathbf{r}_2\right) = \frac{N(N-1)}{Z_N \rho^2} \int d\mathbf{r}_3 \ldots d\mathbf{r}_N \exp\left(-\beta U\left(\mathbf{r}_3, \ldots, \mathbf{r}_N\right)\right), \tag{22}$$

for an homogeneous isotropic system, one has that the radial distribution function depends on the particles' relative distance only (23).

$$g^{(2)}\left(\mathbf{r}_1, \mathbf{r}_2\right) = g\left(\left|\mathbf{r}_1 - \mathbf{r}_2\right|\right) = \frac{\rho_N^{(2)}\left(\left|\mathbf{r}_1 - \mathbf{r}_2\right|\right)}{\rho^2} \tag{23}$$

Now, we recall how the energy is related to the partition function Z_N through the function $Q(N, V, \beta)$ as

$$E = -\frac{\partial}{\partial \beta} \ln Q\left(N, V, \beta\right), \tag{24}$$

where $Q\left(N, V, \beta\right) = \frac{Z_N}{\lambda^{3N} N!} = \frac{1}{\lambda^{3N} N!} \int_0^V dr^N \exp(-\beta U(r^N))$ and λ comes from the integration of the momentum p in the phase space given by: $\lambda = \left[\frac{\beta h^2}{2\pi m}\right]^{1/2}$, which leads to

$$\ln Q\left(N, V, \beta\right) = \ln Z_N - 3N \ln \lambda(\beta) - \ln N! \rightarrow E = \frac{3N}{\lambda} \frac{\partial \lambda}{\partial \beta} - \frac{1}{Z_N} \frac{\partial Z_N}{\partial \beta},$$

where it is easily seen that $\frac{\partial \lambda}{\partial \beta} = \frac{\lambda}{2\beta}$. Thus we have

$$\begin{aligned} E &= \frac{3Nk_BT}{2} + \frac{1}{Z_N} \int U\left(\mathbf{r}_1, \ldots, \mathbf{r}_N\right) \exp\left(-\beta U\left(\mathbf{r}_1, \ldots, \mathbf{r}_N\right)\right) d\mathbf{r}_1 \ldots d\mathbf{r}_N \\ &= \frac{3Nk_BT}{2} + \langle U \rangle \end{aligned}. \tag{25}$$

Then, we require to compute the average potential $\langle U \rangle$, where we assume a pairwise potential, so that:

$$U\left(\mathbf{r}_1, \ldots, \mathbf{r}_N\right) = \frac{1}{2} \sum_{i,j,i \neq j} u\left(\left|\mathbf{r}_i - \mathbf{r}_j\right|\right) \equiv U_{pair}\left(\mathbf{r}_1, \ldots, \mathbf{r}_N\right),$$

where we have a sum of terms depending on two terms only and then this sum contains $N(N-1)$ terms. This leads us to

$$\langle U \rangle = \frac{1}{2} \int u\left(\left|\mathbf{r}_1 - \mathbf{r}_2\right|\right) d\mathbf{r}_1 d\mathbf{r}_2 \left[\frac{N(N-1)}{Z_N} \int \exp\left(-\beta U_{pair}\left(\mathbf{r}_1, \ldots, \mathbf{r}_N\right)\right) d\mathbf{r}_3 \ldots d\mathbf{r}_N\right], \quad (26)$$

which contains a two-body probability distribution function so that $\langle U \rangle$ can be rewritten as

$$\begin{aligned}
\langle U \rangle &= \frac{1}{2} \int u\left(\left|\mathbf{r}_1 - \mathbf{r}_2\right|\right) d\mathbf{r}_1 d\mathbf{r}_2 \left[\frac{N(N-1)}{Z_N} \int \exp\left(-\beta U_{pair}\left(\mathbf{r}_1, \ldots, \mathbf{r}_N\right)\right) d\mathbf{r}_3 \ldots d\mathbf{r}_N\right] \\
&= \frac{1}{2} \int u\left(\left|\mathbf{r}_1 - \mathbf{r}_2\right|\right) \rho^{(2)}\left(\mathbf{r}_1, \mathbf{r}_2\right) d\mathbf{r}_1 d\mathbf{r}_2 \\
&= \frac{N^2}{2V^2} \int u\left(\left|\mathbf{r}_1 - \mathbf{r}_2\right|\right) g^{(2)}\left(\mathbf{r}_1, \mathbf{r}_2\right) d\mathbf{r}_1 d\mathbf{r}_2 \\
&= \frac{N^2}{2V} \int_0^\infty 4\pi r^2 u(r) g(r) dr
\end{aligned} \quad . \quad (27)$$

As a result, we obtain

$$E = \frac{3Nk_BT}{2} + 2N\pi\rho \int_0^\infty r^2 u(r) g(r) dr \quad (28)$$

On the other hand, to compute the pressure P, we use one of the Maxwell relations (29).

$$P = \frac{1}{\beta} \frac{\partial Q}{\partial V} = \frac{1}{\beta} \frac{\partial Z_N}{\partial \beta}. \quad (29)$$

The volume dependency can be made explicit by changing the variables $\mathbf{s}_i = V^{-1/3}\mathbf{r}_i$ so that we compute the derivative of the partition function Z_N with respect to the volume as follows:

$$\begin{aligned}
Z_N &= V^N \int \exp\left(-\beta U\left(V^{1/3}\mathbf{s}_1, \ldots, V^{1/3}\mathbf{s}_N\right)\right) d\mathbf{s}_1 \ldots \mathbf{s}_N; \text{ then} \\
\frac{\partial Z_N}{\partial V} &= NV^{N-1} \int_0^1 \exp\left(-\beta U\left(V^{1/3}\mathbf{s}_1, \ldots, V^{1/3}\mathbf{s}_N\right)\right) d\mathbf{s}_1 \ldots \mathbf{s}_N + , \\
&\int_0^1 \frac{\partial U}{\partial V} \exp\left(-\beta U\left(V^{1/3}\mathbf{s}_1, \ldots, V^{1/3}\mathbf{s}_N\right)\right) d\mathbf{s}_1 \ldots \mathbf{s}_N
\end{aligned} \quad (30)$$

where the derivative of the potential with respect to the volume in the original coordinates can be computed by

$$\frac{\partial U}{\partial V} = \sum_{i<j} \frac{dU(\mathbf{r}_{ij})}{d\mathbf{r}_{ij}} \frac{d\mathbf{r}_{ij}}{dV} = \sum_{i<j} \frac{dU(\mathbf{r}_{ij})}{d\mathbf{r}_{ij}} \frac{\mathbf{r}_{ij}}{3V}. \tag{31}$$

When we change to the original coordinates, we get

$$\frac{\partial Z_N}{\partial V} = \frac{N}{V} - \frac{1}{6Vk_BT} \int \mathbf{r}_{12} \frac{dU(\mathbf{r}_{ij})}{d\mathbf{r}_{ij}}\Big|_{\mathbf{r}_{ij}=\mathbf{r}_{12}} \rho^2 g\left(\mathbf{r}_1, \mathbf{r}_2\right) d\mathbf{r}_1 d\mathbf{r}_2. \tag{32}$$

Therefore,

$$P = \rho k_B T - \frac{2}{3}\pi\rho^2 \int_0^\infty r^3 \frac{dU(r)}{dr} g(r) dr \tag{33}$$

As we did previously with the MSD and VACF, we run our simulation for various values of T to plot the radial distribution function (RDF), which leads to figure 16.

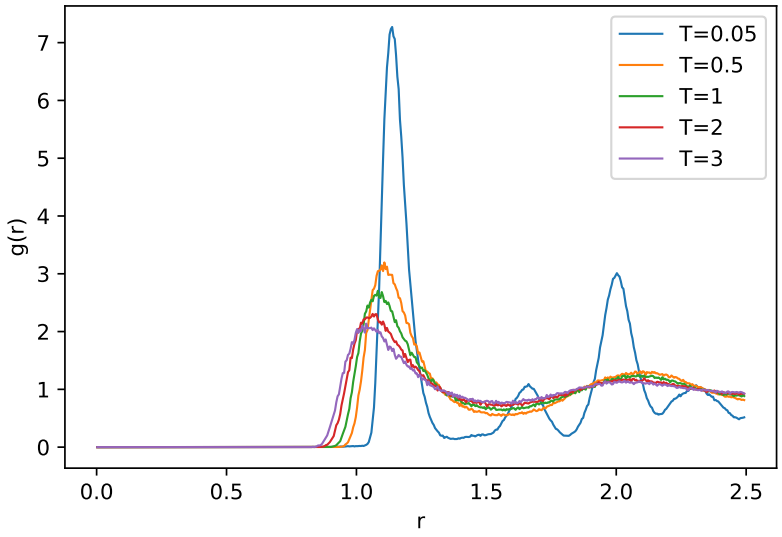


Figure 16: RDF for various T.

8. We observe in figure 17 that the energy drift in the Euler algorithm is significantly larger than the one corresponding to the Verlet algorithm. Moreover, the energy drift computed from the velocity Verlet algorithm has a similar behaviour as the Verlet's since both are equivalent as shown in [3].

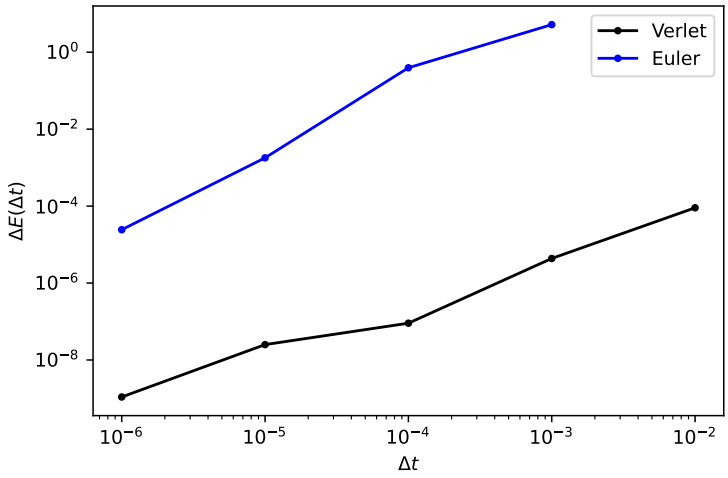


Figure 17: Energy drift comparison.

We comment on a quite interesting property of the Verlet type algorithms. A striking property of the energy determined from the Verlet/leap-frog solution is that it does not show any drift in the total energy (in exact arithmetic). This stability follows directly from the fact that the Verlet algorithm is time-reversible, which excludes steady increase or decrease of the energy for periodic motion. In a molecular dynamics simulation, however, the integration time, which is the duration of the simulation, is much smaller than the period of the system, which is the Poincaré time, that is the time after which the system returns to its starting configuration. The error in the energy might therefore grow steadily during the simulation. It turns out, however, that the deviation of the energy remains bounded in this case also, as the Verlet algorithm possesses an additional symmetry, called symplecticity. This gives rise to conserved quantities, and in particular, it can be shown that a discrete analogue of the total energy is rigorously conserved (in exact arithmetic) [3]. It turns out that this discrete energy deviates from the continuum energy at most an amount of order Δt^k, for some positive integer k. [3]

When we ran our simulations using the Euler algorithm, for values of $\Delta t \geq 0.05$ the energy was not bounded anymore. That is, the Euler algorithm requires a particularly small time step to converge.

References

[1] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Vol. 1. Elsevier, 2001.

[2] Vincent Hindriksen. *How expensive is an operation on a CPU?* URL: `https://streamhpc.com/blog/2012-07-16/how-expensive-is-an-operation-on-a-cpu/`. (accessed: 17.0.2022).

[3] Jos Thijssen. *Computational physics*. Cambridge university press, 2007.