Simulating a Non-ideal Gas using Hard Spheres Confined to a Closed Container

S. S. Mall

***Abstract*—The objective of this simulation is to model a two-dimensional non-ideal gas by using "hard spheres" confined to a container and then to compare to the known gas laws. Necessary simplifications such as neglecting inter-molecular forces were made so that the trajectory of each ball was constant up to a collision, making the time to collision between each pair of balls predictable through a simple formula.**

After a large number of collisions, the system is said to be in steady-state (this is confirmed quantitatively) and thermodynamic quantities such as pressure and temperature are extracted and plotted against each other. Each plot resembled what is expected from the ideal gas law but further inspection of the fit coefficients do not agree, suggesting that the system cannot be treated as an ideal gas.

I. Introduction

THIS project aimed to simulate the behaviour of an non-ideal gas by considering a number of hard sphere "balls" confined to a two dimensional container. After reaching a steady state, thermodynamic quantities can be extracted from the system and compared to prediction. The primary reason for carrying out this task is because although explainable by kinetic theory, the gas laws were first discovered empirically and so being able to simulate a gas's behaviour without the compromises and inaccuracies of physical experimentation is useful. Other checks can also be made, such as comparing the speed distribution of the balls with the Maxwell-Boltzmann speed distribution derived from Statistical Mechanics.

However, although it would be ideal to replicate the physical systems on a scale comparable to those of everyday life, this is unfeasible as the number of particles in the system will increase the runtime of the code massively. Instead, systems of around 100 balls will be considered.

II. Simulation

A. Ball-Ball Collision

As with a real gas, the system must reach a steady state if left isolated for a prolonged period of time, and the way that this is done is for the balls to undergo collisions with each other. Whereas interactions in a real gas are governed by several forces (such as the electrostatic force and the Pauli and Van der Waal's forces due to the Lennard-Jones "6-12" potential), the only interactions between the simulated gas particles will be simple hard collisions. As well as simplifying the calculations for the actual collision, this assumption also means that the balls move at constant velocity when not undergoing a collision, hence it is far simpler to predict when any two balls will collide (if at all).

A "Ball" class was created that would initialise with the parameters mass and radius, and position and velocity lists of dimension 1x2 (which would then become numpy arrays for functionality). The methods contained in each ball class would allow the following general steps to be carried out:

1) Measure the time to the next collision with another ball
2) Move the ball at constant velocity for the aforementioned time
3) Carry out the collision, i.e. update the velocity of each ball taking part in the collision.

1) Time to Collisions: This method was based on a formula derived from the simple observation that when two balls collide, the separation distance between their centres is equal to the sum of their radii. Furthermore by considering the balls some known time before the collision, it is easy to relate the position of each ball before and during the collision, as shown in Fig. 1.

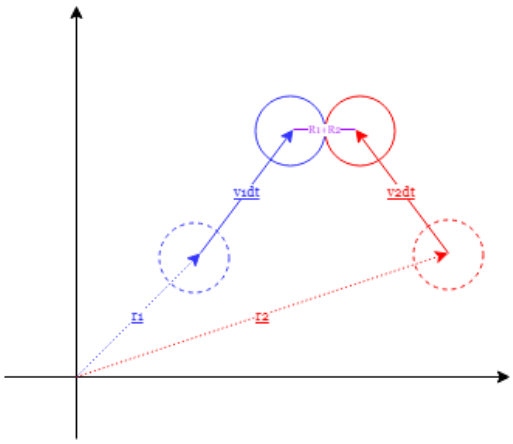

Fig. 1. A diagram to relate the position and velocity of two balls to their position and radii during a collision

This diagram can be used to geometrically derive the formula [1]:

$$((\underline{r}_1 + \underline{v}_1\delta t) - (\underline{r}_2 + \underline{v}_2\delta t))^2 = (R_1 + R_2)^2 \qquad (1)$$

where $\underline{r}_i$ is the position of ball i,
$\underline{v}_i$ is the velocity of ball i,
R_i is the radius of ball i,
and δt is the time to the next collision.

This equation can be rearranged for δt as follows:

$$\delta t = \frac{-\underline{r}.\underline{v} \pm \sqrt{(\underline{r}.\underline{v})^2 - v^2(r^2 - R^2)}}{v^2} \qquad (2)$$

where $\underline{r}$ is $\underline{r_1} - \underline{r_2}$,

$\underline{v}$ is $\underline{v}_1 - \underline{v}_2$,
and R is $R_1 + R_2$ if neither ball is a container $R_1 - R_2$ if ball$_1$ is a container (visualise ball$_2$ being within ball$_1$).

Other than this subtle difference, the formulae describing the collision between two balls is unchanged if one of the balls is a container. Hence by defining a Container as a derived class of ball with a way of distinguishing it from a normal ball, only one set of calculations has to be made.

Carrying out the same calculation for every ball regardless of whether or not is a container means that there are different types of solution (the positive and negative solution from the square root in the formula) for δt depending on the type of collision.

1) **Non-Real** No collision will occur. This is highly likely for balls of small radii or if v does not lie along r. No solution is taken.
2) **Negative, Negative** Both balls are moving away from each other and will not collide until at least one of their velocities is changed by another collision because they have already collided in the past. No solution is taken.
3) **Positive, Negative** One solution has occurred in the past and one is yet to occur. This can only be the case if one of the balls is overlapped with / inside another, for example if one of the balls is a container. The positive solution is taken and the negative discarded.
4) **Positive, Positive** The balls will collide. The solution with the lowest value corresponds to the first point in time where the radii of the balls meet, and the second is when the balls have finished passing through each other. The solution with the lowest value is taken.

This calculation was included in a method called *timeToColl* in the Ball class and returns the time to collision.

2) Collision: Now that the time to the collision has been calculated, the position of each ball is updated by adding $v_i \delta t$ to their initial so that the two balls are now in contact. A collision formula is then used to update the velocity of each ball and the position of each ball advanced by a very short period of time to prevent them from sticking to each other.

The collision formula is derived by resolving the problem into two new "dimensions": the line parallel to the separation of the two balls and the line perpendicular. This is because only the velocity along the line of separation is affected by the collision while the velocity perpendicular is unchanged.

This collapses the problem from two dimensions into one and the change in velocity along r is easily found using the conservation laws of energy and momentum. Hence, a vector equation can be formed for the velocity after collision ($\underline{v}_i$) as being equal to the initial velocity ($\underline{u}_i$) plus some term multiplied by the unit vector in the $\hat{r}$ direction [2]:

$$\underline{v}_1 = \underline{u}_1 - \frac{2m_2}{m_1 + m_2} . \frac{\underline{u}.\underline{r}}{\underline{r}.\underline{r}} . \underline{r} \tag{3}$$

and similarly,

$$\underline{v}_2 = \underline{u}_2 + \frac{2m_1}{m_1 + m_2} . \frac{\underline{u}.\underline{r}}{\underline{r}.\underline{r}} . \underline{r} \tag{4}$$

where m_i is the mass of ball$_i$ and the other quantities have already been defined. The collisions here are elastic though it would not be difficult to consider non-elastic collisions by scaling the change by a chosen coefficient of restitution.

A method containing the calculation (simply called *collide*) was created in the Ball class that would take a second ball as well as itself, updating the velocity of both balls when it is called.

B. Simulation Code Structure

The next layer is the Simulation class, which has the purpose of generating balls within a Container and colliding them. It is initialised with a Container and contains the following methods, which are run in this order:

1) **generateBalls** Takes the number of balls to be generated, the mass and radius of each ball, and a range in which each component of each ball's velocity is chosen according to a uniform or normal distribution. The balls are generated systematically to form a grid, as shown in Fig. 2.
2) **lowestTime** Iterates through each non-repeating combination of two balls and calculates the time to their next collision using the *timeToColl* method in the Ball class. After comparing to an initial large value, the method compares each time to the current lowest time to collide and stores the new time if it is lower. This method returns the time to the next collision as well as which balls are involved.
   This method was made more efficient by noticing that a ball cannot collide with itself and that the time to collide between ball$_i$ and ball$_j$ is the same as that between ball$_j$ and ball$_i$.
3) **nextCollision** Is passed the lowest time to collision and balls to collide by the $lowestTime$ method. All balls are moved for this time (so that the colliding balls are in contact) and the relevant collision carried out. Each ball is then advanced by a very small amount of time to prevent any from sticking to each other.
4) **drawFrame and Run** *drawFrame* provides a visualisation of the simulation at the current time. *Run* advances the simulation by a specified number of collisions and provides the option of visualising as it does so.

III. Thermodynamics - Defining Quantities

Now that a working simulation of the modelled gas has been created, thermodynamic quantities can be extracted from it. However, it must first be understood how these quantities are measured and relate to each other in order to predict what certain plots should look like. Additionally, certain checks should also be made here regarding the distributions of certain quantities before taking data for the investigation.

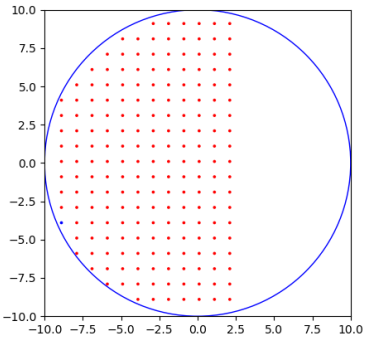

Fig. 2. Initialisation of 100 balls of radius 0.1 in a container of radius 10.

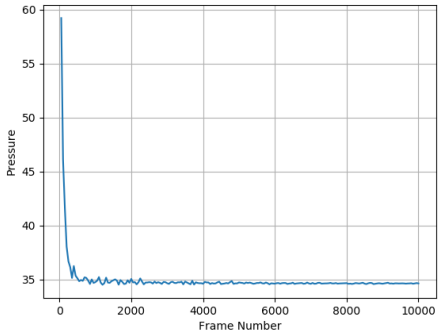

Fig. 3. A plot of pressure as a function of number of collisions occurred since the start of the simulation.

A. *Pressure*

The pressure exerted on a container is given in 3 dimensions by the force exerted on it per unit area, with the 2 dimensional analogue being the force exerted per unit length (this is the reason why arbitrary units are used in the results). The pressure exerted on the container can easily be found because the force exerted on the container during a collision with a ball is the same as the force on the ball. Force is equal to change in momentum divided by time, so by recording the total time a simulation has run for and the momentum change of each ball when colliding with the container, the pressure can be recorded.

This was implemented by simply adding a variable to the Simulation class that updated each time a collision took place, and another variable that added the momentum change of a ball if the other colliding ball was a container. These variables are both initialised to zero when the balls are generated so that generating a new system restarts the counting.

The first test that was carried out was to plot the cumulative pressure as a function of the number of collisions that have taken place, with the pressure being probed every 500 total collisions (not necessarily having to be with the container). 100 balls were initialised with random velocity components in the range $-10 \rightarrow 10$ with radius $r = 0.1$ and spacing $10 \times r = 1$.

It was predicted that the pressure would be low at first as most of the collisions would be between balls if the ball spacing was low. It would then peak and proceed to tail off to a constant value as the system reaches thermal equilibrium. This is somewhat agreed with by the result, shown in Fig 3.

The usefulness of this plot is that it shows that the system seems to reach an equilibrium at around 2000 frames but still has fluctuations of just over 1% in this region. This means that 2000 frames can be used as a minimum runtime before taking data for future plots.

B. *Maxwell-Boltzmann Distribution*

The Maxwell-Boltzmann distribution is a general result of statistical mechanics used to describe the distribution of particle states i in a canonical ensemble, i.e. a system with a constant number of distinguishable particles [3]:

$$f(v_i) = \frac{1}{Z}\, e^{-\frac{\frac{1}{2}mv_i^2}{kT}} \tag{5}$$

where $f(v_i)$ is the probability of a particle having a component of velocity v_i,
Z is a normalisation constant called the Partition Function,
k is the Boltzmann constant,
ϵ_j is the energy of that state,
and T is the temperature.

It is easy to see [4] that the probability of a particle with a speed in the range $v \rightarrow v + \delta v$ is:

$$p(v)\delta v = f(v)\, g(v)\, \delta v \tag{6}$$

where $g(v)$ is the density of states with total speed v. This is an important term because all velocities lying on a circle in a plane in v-space have different velocities but the same speed, hence we need to consider all these velocities. It can be seen then that $g(v)$ is given by:

$$g(v) = v \int_0^{2\pi} d\theta \tag{7}$$

where θ is the angle given by the ratio of the components of the velocities in the $v_x - v_y$ plane, which when integrated over the plane produces a factor of 2π. This produces:

$$p(v)\delta v = \frac{2\pi}{Z}\, v\, e^{-\frac{\frac{1}{2}m}{kT}v^2} \delta v \tag{8}$$

Integrating over v using standard integrals and the formula for Z produces the general form for the two-dimensional Boltzmann speed distribution:

$$p(v) = \frac{m}{kT}\, v\, e^{-\frac{\frac{1}{2}m}{kT}v^2} \tag{9}$$

A test to see if the simulation acts as predicted is to initialise the balls with a very narrow speed distribution and then run the simulation for a large number of collisions. The Maxwell-Boltzmann speed distribution is valid for when the

gas is in thermal equilibrium so if the distribution is observed after running the simulation, the gas must be in equilibrium. However, it is easy to over-fit to the distribution so this is not a good method of determining if a gas is in equilibrium but a good check to see if it is behaving as expected.

A simulation was initialised with 200 balls and a narrow velocity-component range of $10.0 \rightarrow 10.1$, producing a speed distribution shown in Fig. 4

After running the simulation for 3000 collisions, the distribution was far closer to a Maxwell-Boltzmann distribution. The temperature of the simulation was found and used to determine the fit coefficients to overlay the Maxwell-Boltzmann curve over the histogram (with some scaling). Although this produced an adequate fit, the simulation may not have reached a complete thermal equilibrium, hence why the fit does not match the histogram completely.

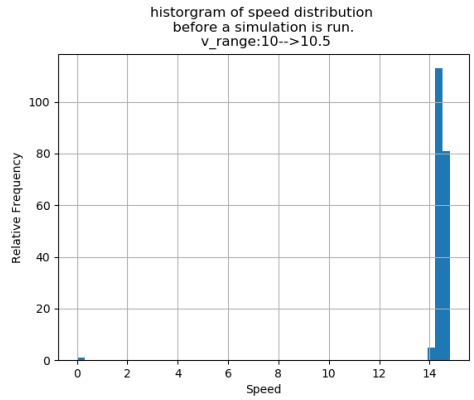

Fig. 4. Speed distribution of 200 balls in narrow velocity range before running simulation.

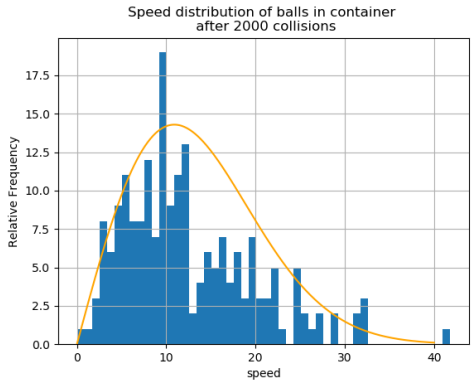

Fig. 5. Speed distribution of 200 balls in narrow velocity range after running simulation for 3000 collisions. The balls have reached thermal equilibrium and are distributed according to the Maxwell-Boltzmann speed distribution.

The average speed should remain constant for balls undergoing elastic collisions because there is no change in the kinetic energy over time and the kinetic energy is directly related to the mean speed. Due to the skew of the distribution, the average speed is always slightly higher than the modal speed which is consistent with what is seen in the result.

C. Energy and Temperature

Because the balls only interact through collisions and do not exert any other forces on each other, the potential energy of the system is zero and the only energy is kinetic. Because of this, it is very easy to relate the temperature to the energy of the system.

Because the Maxwell-Boltzmann speed distribution is a probability distribution, it can be used to find the expectation of functions dependent on the speed. This is particularly useful because the kinetic energy of each particle is related to its speed such that we can say [5]:

$$U =< E_k >= \frac{1}{2}mN < v^2 >= \frac{1}{2}mN \int_0^\infty v^2 p(v) dv \quad (10)$$

Using standard integrals to evaluate the expression leads to the expression for internal energy in two dimensions:

$$U = k_B T = \frac{1}{2} m \sum_n^N v_n^2 \quad (11)$$

which can easily be be rearranged to define the kinetic temperature.

1) Energy Test: One way to test if the simulation is behaving properly is if kinetic energy is conserved. The energy was measured using a function that took a Simulation class and simply iterated over each ball and calculating its individual kinetic energy. The value was then added to the total kinetic energy of the system.

A quick test to ensure the simulation is behaving as expected is to check that the total kinetic energy of the system is being conserved between collisions. A simple function was written that would take a Simulation class and advance it by 5000 collisions, recording the total kinetic every 10 collisions using the function outlined above.

This was run several times but there was always a seemingly random variation of around 3% which is significant considering the value should be completely unchanging. The reason for the variation is unclear but the 3% has been taken as an uncertainty in the energy and by extension, the temperature.

IV. Thermodynamics - Investigation

The purpose of the following investigations is to look at the relationships between the state variables of the system - notably pressure, temperature, and volume. The relationship between these variables should agree with the Ideal Gas Law, which is given in 3 dimensions by:

$$PV = Nk_B T \quad (12)$$

where P is the pressure exerted on the container,
V is the volume of the container,
N is the number of balls in the container,
k_B is the Boltzmann constant,
and T is the temperature of the system.

An interesting note here is that collapsing to two dimensions may require scaling by a factor of $\frac{3}{2}$ though this is only an idea.

When moving from 3 dimensions to 2, the pressure goes from being force imparted (F) divided by the surface area of the container to force over perimeter, while volume simply goes to area. This is shown below for the container being a sphere in 3 dimensions (equation 13) and a circle in 2 dimensions (equation 14).

$$F \cdot \frac{1}{4\pi r^2} = N\, k_B\, \frac{4}{3}\pi r^3 \tag{13}$$

$$F \cdot \frac{1}{2\pi r} = N\, k_B\, \pi r^2 \tag{14}$$

It can be seen that the left hand side of the equation is being multiplied by $\frac{1}{2r}$ whereas the right hand side is being multiplied by $\frac{4}{3r}$, therefore it may be justified to say that the ideal gas law in 2 dimensions is actually given by

$$P\,V = \frac{3}{2}\, N\, k_B\, T \tag{15}$$

The ideal gas law was originally built from several empirical laws and later explained through kinetic theory [6]. Further developments include the addition of correction terms leading to the van der Waals equation of state [7]:

$$\left(P\, +\, a\left(\frac{N^2}{V^2}\right)\right)\left(V - Nb\right) =\, Nk_BT \tag{16}$$

The b term is easiest to explain as it is simply equal to the area of each individual ball (as all balls are assumed to have the same radius) while the a term accounts for the van der Waals force between the particles, which is harder to quantify.

Each of the following experiments was carried out in a similar way: a function was created in a separate module that is initialised with the parameters required to initialise a simulation, e.g. container radius, ball radius, number of balls, etc. The function will vary one of the parameters based on the experiment and run a simulation, collecting data (e.g. Pressure and Volume) at the end. This is continued until enough data points are gathered to build a plot.

Each experiment was initialised with container radius 10, ball number 100, ball mass and radius of 1 and 0.1 respectively, and velocity-component range $-10 \rightarrow 10$.

A. *Pressure-Temperature*

The first experiment to be run was for pressure as a function of temperature. The way that temperature was varied was by moving the velocity-component range by a value of 10 for each simulation so that the system will thermalise at a higher temperature.

Two different ways of calculating the pressure were used: firstly, the pressure was calculated cumulatively so that the counters for the total time elapsed and the momentum imparted on the container wall were not reset. Each simulation was run for 2500 frames before recording pressure.

The second experiment had each simulation run for 2000 collisions, before resetting the counters and only taking data for the next 500 collisions. The reasoning behind this was that "forgetting" the first frames would increase the accuracy of the pressure value because these frames are not in steady-state. However, the second method only takes data for 500 collisions and the number of collisions involving the container is likely to be far lower than that. This allows for far more variation as each collision with the container will become more significant with less opportunity for the pressure to average out.

The agreement between both methods just provides another check to see if the system is in equilibrium, as the two methods should provide the same result if correct.

Fig. 6 shows the cumulative PT plot. The second plot is not shown because it is visibly very similar to the first and so will be compared by its fit coefficients. The fit coefficients for the first were calculated using the *scipy.polyfit* function and gave a gradient of $3.73 \times 10^{-24} \pm 2.9\%$ and an intercept of $324.3 \pm 19.4\%$.

The second experiment gave a very similar gradient of $3.72 \times 10^{-24} \pm 3.6\%$ but with an intercept of $423.2 \pm 18.6\%$.

Comparing this to the ideal gas law, the gradient should be equal to $\frac{Nk_B}{V}$, which gives a value of 4.39×10^{-24}. Again, the experiment agrees with the calculated value only to an order of magnitude but not any further. This is because the gas is not ideal, as is suggested by the intercept and non-linear portion as the temperature approaches zero.

The value of the intercept is related to the a term in the Van der Waals non-ideal equation of state, however the uncertainty in the value is too large to make any confident conclusion. Furthermore, the a term is used to account for the "$6-12$" potential between particles but this potential has not been considered in the simulation. For this reason, it is unlikely that the simulation will even agree with the non-ideal gas law so the term is not calculated here.

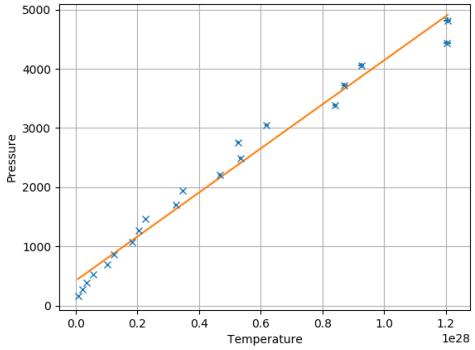

Fig. 6. Plot of cumulative pressure exerted on a container as a function of temperature.

B. *Pressure-Volume*

An experiment for (cumulative) pressure as a function of the container "volume" (reduced down in two dimensions to the area of the container) was carried out. As before, all parameters required to initialise a simulation were passed to a function which increased the container radius by a value of 1 for each simulation. Each of 21 simulations was run

for 3000 frames and the pressure recorded at the end. The results were plotted, shown in Fig. 7 and analysed using the $scipy.optimise.curve\_fit$ function.

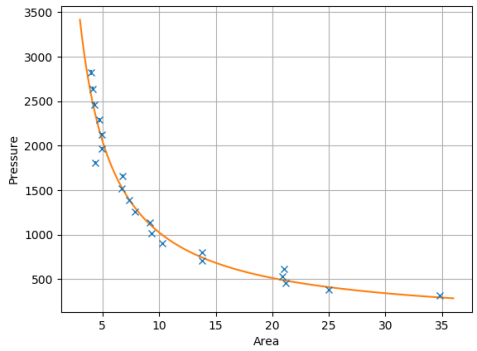

Fig. 7. A plot of pressure exerted on a 2-dimensional container as a function of its area. Each simulation was run with 100 balls of mass 1 and radius 0.1 in a container of initial radius 10 for 3000 frames.

The linear coefficient found relating pressure to inverse area was equal to 10432. This value should correspond to Nk_BT which can be found by measuring using the temperature probe code. This produced a value of around 13800 which has a percentage difference to the result of just over 24%. Again, the experiment produces an appropriate plot and agrees with the ideal gas law only to an order of magnitude.

C. Pressure - Ball Number

The final plot is of pressure exerted on the container as a function of the number of balls contained. This experiment was carried out by running 11 simulations all with the same initial conditions aside from the number of balls generated, which increased by 15 balls each run. Treating the simulation as an ideal gas, the plot should produce a straight line; however, letting the a term go to zero shows that for a system where the area of the balls are non-negligible:

$$P\left(V - Nb\right) = Nk_BT \tag{17}$$

so that

$$P = \frac{k_BT}{\frac{V}{N} - b} \tag{18}$$

which will produce a positive-slope curve and will tend to a liner plot for $b \to 0$. Forcing this using *scipy.optimise.curve_fit* produced the plot in Fig. 8 for a simulation with ball radius 0.2.

The fit coefficients here are useful as the formula can be verified using the area of the container as well as the area of each ball and the temperature of the system. The results from the coefficients are as follows:

1) $k_BT \approx 138$ which when compared to the calculated pressure, was correct only to an order of magnitude.
2) container area ≈ 428. A container of radius 10 has an area of 314 so again, this only agrees to an order of magnitude.

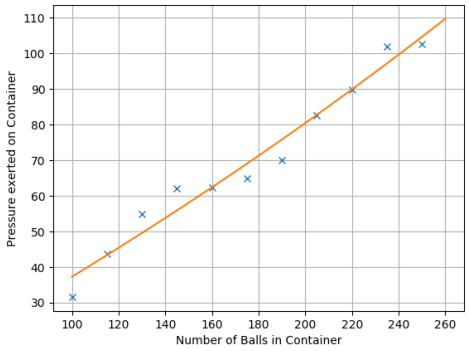

Fig. 8. A plot of the pressure as a function of the number of balls contained within it.

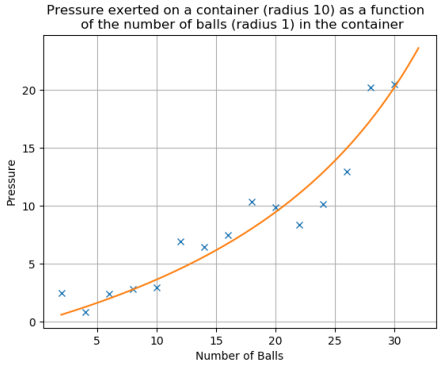

Fig. 9. A plot of pressure on a container as a function of the number of balls contained. The container is of radius 10 and the balls a non-negligible 1.0.

3) $b = 2.89$. For balls of radius 0.2, this value should be 0.13 .

A second simulation was run with decreased ball spacing so that balls of a more significant radius of 1.0 could be tested to give a more pronounced curve, as shown in Fig. 9

V. Conclusion

In conclusion, the simulation displays the general characteristics of a non-ideal gas but does not agree with the governing equations completely. This is likely to be because the number of balls considered is far too small to represent a real gas. This number can be increased by streamlining the code to only carry out *timeToCollide* calculations for the most recently collided balls and subtracting the previous *timeToCollide* for the rest.

It can be shown that the number of calculations carried out for N balls is $\sum_n^N n = \frac{1}{2}N(N+1)$ so the number of calculations scales approximately with N^2. Streamlining will reduce this to an order-N problem and decrease runtime so that simulations with more balls can be run (though of course this is likely to still be insufficient).

Other minor improvements include a more explicit overlap-prevention algorithm (though overlap was never experienced) and in the PN investigation, checking to make sure that the balls in the highest-N simulation can be contained before the simulations are run as this is only checked at the start of each simulation.

VI. Bibliography

1) MacKinnon, A. (2016). Project B: Thermodynamics Snookered. [online] Imperial College Physics Y2 Computing. Available at: https://bb.imperial.ac.uk/bbcswebdav/pid-1446725-dt-content-rid-4655438_1/courses/COURSE-PHY2_LAB-18_19/Y2Computing/python_year2/Projects/.build/html/Snooker.html [Accessed 1 Feb. 2019].
2) Berchek, C. (2009). 2-Dimensional Elastic Collisions without Trigonometry. [online] Vobarian.com. Available at: http://www.vobarian.com/collisions/2dcollisions2.pdf [Accessed 1 Feb. 2019].
3) Patterson, C. (n.d.). Statistical Physics (Nov-Dec 2018) Outline Notes: Lectures 4-6. [online] Available at: https://bb.imperial.ac.uk/bbcswebdav/pid-1457504-dt-content-rid-4601744_1/courses/DSS-PH2_SP-18_19/Outline%20Notes/SummaryNotes-L4-6.pdf [Accessed 2 Feb. 2019].
4) Adjodah, D. (2011). MAS.864: Derivation of 2D Boltzmann Distribution. [online] Fab.cba.mit.edu. Available at: http://fab.cba.mit.edu/classes/864.11/people/dhaval_adjodah/final_project/write_up_boltzmann.pdf [Accessed 2 Feb. 2019].
5) Vallance.chem.ox.ac.uk. (n.d.). The Equipartition Theorem. [online] Available at: http://vallance.chem.ox.ac.uk/pdfs/Equipartition.pdf [Accessed 3 Feb. 2019].
6) Anon, (2017). structure of matter lecture notes - part 1. [online] Available at: https://bb.imperial.ac.uk/bbcswebdav/pid-1189708-dt-content-rid-3896615_1/courses/DSS-PH1_SMAT-17_18/2017_SoM_Wk1.pdf [Accessed 3 Feb. 2019].
7) Anon, (2017). structure of matter lecture notes - part 3. [online] Available at: https://bb.imperial.ac.uk/bbcswebdav/pid-1189710-dt-content-rid-3896619_1/courses/DSS-PH1_SMAT-17_18/2017_SoM_Wk3.pdf [Accessed 3 Feb. 2019].