Using a Simulation of Balls in a Container to investigate the limitations of the Ideal Gas formula

Abdul-Wadoud Charbak

***Abstract*—Using python, I coded a simulation of balls in a container. These balls were programmed to have a finite mass and radius within a container of much larger mass and radius. Throughout this Simulation we assumed all collisions were elastic both between the balls and the container. Using the Simulation I saw that kinetic energy of the balls is directly proportional to temperature. The results I achieved agreed with the fundamental principles of the Ideal gas equation, such as Pressure being proportional to Temperature and the Pressure also being proportional to the number of balls in the container. I also discovered that the ball radius actually had no effect on the equation of state, but did effect the quality of the simulation as the smaller the ball radius, the better the simulation ran. Our results did not fit very well to the ideal gas equation as seen in Figure 6, instead the results fitted much better to the Van Der Waals equation of state as seen in figure 8. The code outputted the value of a to be (4.10 ± 2.86) x 10^{-8} Nm^4 and b to be 2.28 ± 1.05 m^3.**

I. Introduction

Our job here was to conduct a thermodynamic investigation. The problem is that as second year undergraduate students, we do not have access to the equipment necessary in order to conduct this experiment in real life. For this reason we decided to code a simulation of the problem and use the simulation to conduct our physics experiments. Simulations like these offer many benefits such as us being able to code idealised or real life situations into any computer that can run Python which is extremely useful in the sharing of scientific discoveries and knowledge across the globe. We can set the Parameters to however we like and scale up the Simulation to the limit of our computers. Most importantly however, there are practically no consequences in doing our experiment via a Simulation in comparison to a real life experiment which requires equipment and incurs costs.

II. Simulation Setup / Method

The first thing I did was setup a Class for the balls with attributes of mass, ball radius, velocity and position. Throughout this Simulation I considered the collisions to be perfectly elastic, so that energy and momentum are fully conserved. The two most important functions within the ball class were "time_to_collide" and "collision". For "time_to_collide", I derived the formula from the quadratic in the beginning of the script which can be seen below:

$$\delta t = \frac{-\vec{r}\cdot\vec{v} \pm \sqrt{(\vec{r}\cdot\vec{v})^2 - \vec{v}^2(\vec{r}^2 - R^2)}}{\vec{v}^2} \tag{1}$$

where $\vec{r} = \vec{r_1} - \vec{r_2}$ and $\vec{v} = \vec{v_1} - \vec{v_2}$ and R is the difference between the radii. Due to the nature of the quadratic, I had to consider the case of 2 positive roots and always return the smallest root. This was key to running the Simulation as explained later. The "collision" function was derived from the formula for kinetic energy and momentum conservation in 2 dimensions.

The way colliding balls in my simulation worked was to firstly see the find the time to collide between the 2 balls you wanted to collide, move them to the position of collision then collide them. I derived a formula from the conservation of KE and Momentum to give me a formula for the velocity after collision:

$$\vec{v_1} = \vec{u_1} - \frac{2m_2}{(m_1+m_2)}\frac{(\vec{u_1}-\vec{u_2})\cdot(\vec{r_1}-\vec{r_2})}{\|\vec{r_1}-\vec{r_2}\|^2}(\vec{r_1}-\vec{r_2}) \tag{2}$$

where $\vec{u_1}$ & $\vec{u_2}$ are the initial vector velocities of ball 1 and 2. This equation is only for the final velocity of ball 1.

This leads onto the Simulation class. This is initialised with attributes of number of balls, temperature coefficient, ball radius and the container used in the simulation. The balls were arranged systematically by placing each ball a distance of half the container's radius away from the centre in angle increments of 2π divided by the number of balls in the simulation. The balls were also given a random velocity with a mean of 0. In order to achieve this, I used pythons in-build "numpy" module to generate a normal distribution of velocities with a mean of 0. The width of this Gaussian is controlled by the Temperature coefficient as the wider the Gaussian, the faster the balls, the higher the temperature of the system.

The 2 most important functions of the simulation class were "next_collision" and "run". The collisions could have been approached in 2 ways: The first method is to have your simulation run (and be animated) at a constant time increment δt. For example, δt could be set to 1/30 so you could run your simulation at 30 frames per second. This may look nicer, but is incredibly inefficient and severely increases the chances of ball overlap and intersection which breaks the simulation. The second way is to check the time to collide of each ball with every other ball and the container and find the minimum positive time and move all the balls by that time and collide the 2 balls which are colliding with each other. This is the method I used and the basis of how "next_collision" worked.

The way I achieved this was by setting up a large 2 Dimensional time array with an array for every ball. I then found the time to collide of each ball with each other ball. If the program was attempting to compare one ball with itself, it would instead append 10000s to the time array. The value in each array was the collision with the container. Then by finding the value and position of the minimum positive time, the program would tell me by how much to move each ball by and which balls collided. Table I illustrates this. The fact we used the minimum time to collide out of every ball combination ensures that no other ball will collide with another ball or the container in that given time. The program then collides the 2 colliding objects. The program also uses a class variable to store the total time the simulation is running for, which is used to calculate Pressure. The ”run” function runs the simulation by running ”next_collision” for the specified number of frames. It is also responsible for animating the balls and creating ball data plots. It outputs the frame number and associated values for Temperature, KE and Pressure throughout the simulation.

III. Results and Discussion

From our results, we can clearly see in figures 3 and 4 that Kinetic energy and total momentum are conserved throughout the simulation. This makes complete sense as the balls are colliding elastically which means that the momentum and kinetic energy are conserved. In fact, the equations of collision for the balls were derived based on this very fact.

As you can see in Figure 1, The simulation was ran for 10000 frames and produced the following graph. The graph shows a linear relationship between the distance from the centre and the count. This is because all the balls are randomly and equally distributed after this long time which means P(r) = 2πr dr. Our bins here, represent the ”dr”s in this equation. Figure 2 shows the inter-ball separation. This graph works as over time the balls go into random positions but are spread out, meaning in general most of the balls are half the diameter of the container apart.

As you can see in Figure 5, I discovered that the Pressure of the system is indeed directly proportional to the Temperature of the system. This makes sense as the higher the temperature, the larger the width of the normal distribution of the velocities of the balls, the faster the balls are in general. This means the balls have a higher kinetic energy and higher momentum which increases their impulse when in contact with the container walls meaning the pressure on the container walls is higher. However, this only follows very well for low values of pressure and temperature. The points start to hover around the line for higher values of pressure and temperature. We can be confident with this result as it agrees with the equation of state.

In Figure 6 I ran many different Simulations varying temperature across all of them keeping the number of balls a constant and did this with balls of 3 different radii, varying in orders of magnitude. I then fitted the data for each radii to the equation of state and produced 3 straight lines. As you can see these lines are extremely close to one another and accounting for the random fluctuations of the Simulation, we can safely say that according to the data the radius of the ball has no effect on the equation of state. This is one of the results of my simulation that does make me have slightly less confidence in my results as it disagrees with the theory. The theory suggests as the balls get bigger, the area left in the container for the balls to move around decreases so the lines should differ in gradient. As the ball radius decreases, the gradient of the line should increase as it should be proportional to $\frac{1}{V}$. However, my results do not show this. The solution to this problem may be to factor in the decrease in Area as the ball radius gets bigger and fit the data to a model of container area minus the total area of all the balls combined as the volume.

As you can see in Figure 7 the speed distribution of the balls after the simulation was ran follows very closely to that of the Maxwell Boltzmann (MB) distribution. This makes sense as a random distribution of speeds in a container will eventually tend to a MB Distribution from the theory. This gives me further confidence in my results and in the Simulation as this data s with the theory. If we let the Simulation run for an infinite number of frames, then eventually the speed distribution of the balls would tend towards a MB Distribution. At around the 1.5 m/s mark there is a dip in the data. This has consistently appeared when re-running this Simulation again and again. The most likely cause for this dip is most likely due to the systematic initialisation of the position of the balls. Again, if we could run this Simulation for an infinite number of frames, this dip should disappear. The uncertainties on this graph follow $\sqrt{N}$ where N is the Relative Intensity or Number of Balls. This is a standard error calculation used for Histograms.

Finally, we have figure 8. This plot shows that as Pressure increases as the the number of balls increases keeping temperature constant which perfectly matches the theory as the more balls you have, the more force of the balls on the container, the higher the pressure. I chose to compare against number of balls instead of temperature due to time constrains as it far less time to use simulations in a range starting at 0, then stay at 100 balls the entire time. However, as you can see fitted to a Van-Der-Waals line with parameters a and b, the line fits almost perfectly. This shows that there are limitations of the Ideal Gas model and that the Van-Der-Waals line was a more accurate equation. We achieve values of a to be (4.10 ± 2.86) x 10^{-8} Nm^4 and b to be 2.28 ± 1.05 m^3. This gave further confidence in my results as according to the theory, a is supposed to be 0 (as it is due to inter-molecular forces) and b is in the correct order of magnitude. The code may have been improved here if we could also compare to a graph of P vs T.

IV. Conclusion

In conclusion, the Simulation provided many physical discoveries that successfully agreed directly with the theory. This gives me a lot of confidence in my results and is a key success. However, we had one result that disagreed with the theory. To improve this, I believe it must be necessary to change how the balls are initialised in their positions to be able to include more bigger balls. This could be done by using multiple rings on balls instead of one giant ring. This would allow us to have balls that cover a lot more of the area of the container, which may give us results that agree more with the theory. I also believe that with very large balls, the simulation is not extremely stable, with roughly 20% of attempts with a simulation of 3 balls each with a radius of 6 m ending in a not real Simulation (i.e. the ball escapes the container). The way this can be fixed is to code for these specific circumstances of larger balls. In general with my Simulation, the smaller the balls the more stable my Simulation is.

TABLE I
TABLE TO DEMONSTRATE HOW THE TIME STORAGE WORKS. THIS IS WITH 5 BALLS.

Times / s	Ball 0	Ball 1	Ball 2	Ball 3	Ball 4
Ball 0	10000	1.56	2.65	1.24	10.29
Ball 1	1.56	10000	5415.25	-78.17	45.3
Ball 2	2.65	5415.25	10000	6.24	**0.56**
Ball 3	1.24	-78.17	6.24	10000	3
Ball 4	10.29	45.3	**0.56**	3	10000

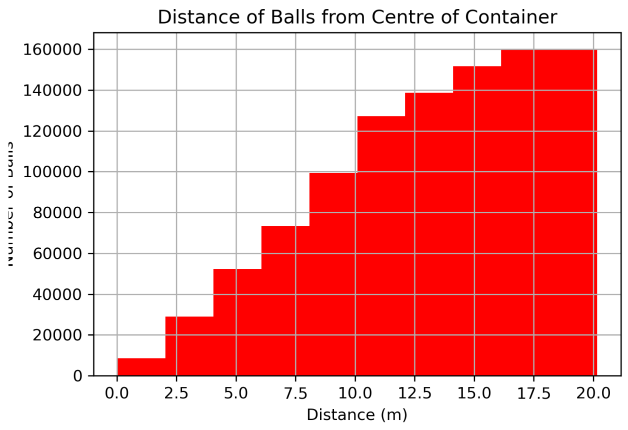

Fig. 1. These were the distances of the balls from the centre of the container after running a Simulation of 100 balls of radius 0.01m for 10000 frames in a container with a radius of 20m (The x-axis should say Count). The graph shows a linear relationship. This is because all the balls are randomly and pretty much equally distributed which means P(r) = 2πr dr. Our bins here, represent the ”dr”s in this equation.

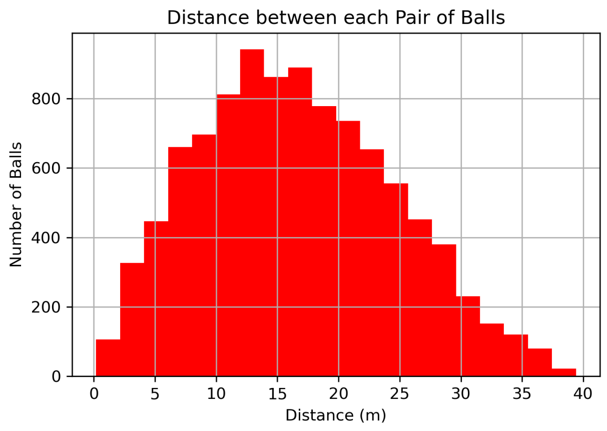

Fig. 2. A histogram of the distances between each pair of balls from each other. This was left to leave for 10000 frames in a container with a diameter of 40m. This histogram makes sense as if the gas is uniformly, randomly distributed across the container, the distance between each pair of balls would maximise at around half the diameter of the container.

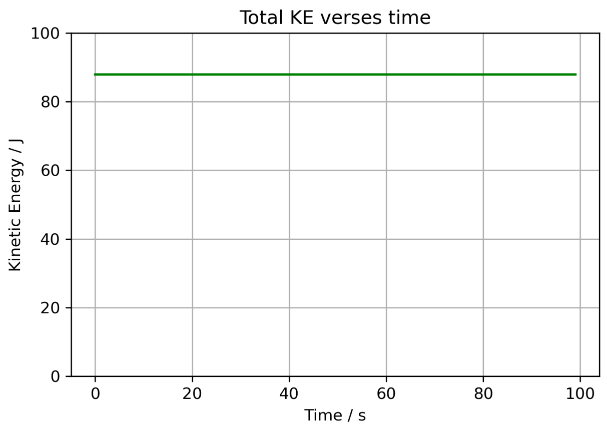

Fig. 3. Total Kinetic Energy (KE) vs the time the simulation has been running. As you can see this remains constant throughout the entire simulation. This makes sense as the collisions are elastic which means the total KE after a collision much equal the total KE before.

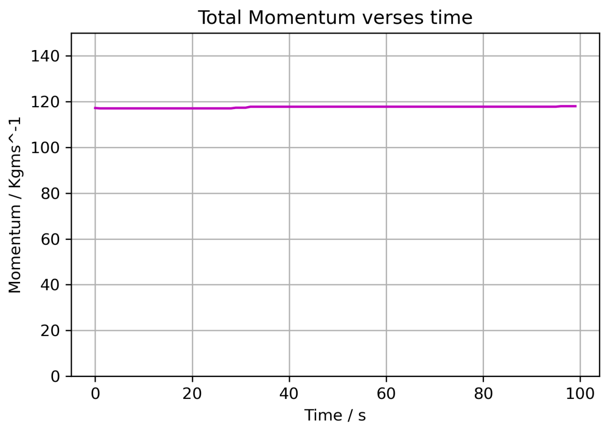

Fig. 4. Total Kinetic Momentum vs the time the simulation has been running. As you can see this remains constant throughout the entire simulation. This makes sense as the collisions are elastic which means momentum after a collision much equal momentum before. The very slight bending of the line around the 30 second mark is due to rounding errors of the Simulation.

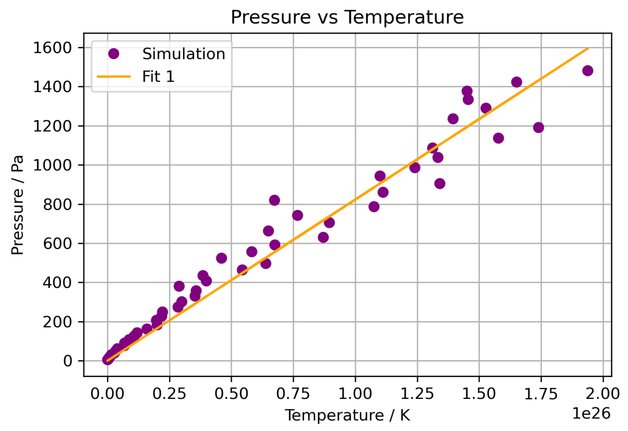

Fig. 5. Pressure vs Temperature graph. These Simulations each had 100 balls. The line ”Fit 1” is a line generated from the theory. As you can see our data follows the line in general. However, for small values of Pressure and Temperature, the line is almost exactly followed, whereas for higher Temperatures and Pressures the data starts to deviate from the line slightly.

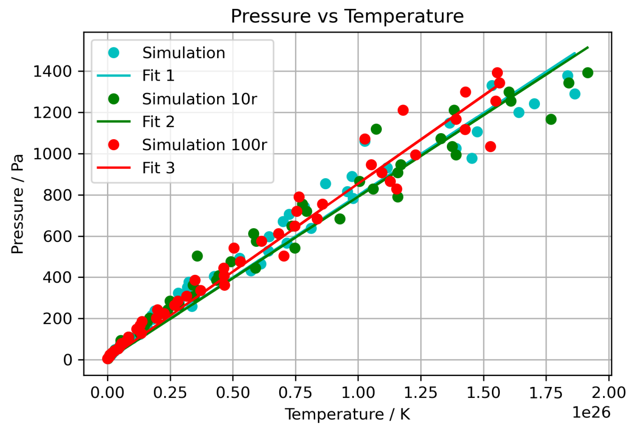

Fig. 6. Pressure vs Temperature graph again but this time including plots of with other ball radius. r = 0.01m. These Simulations each had 100 balls. As you can see the ball radius has very little to no effect on the equation of state.

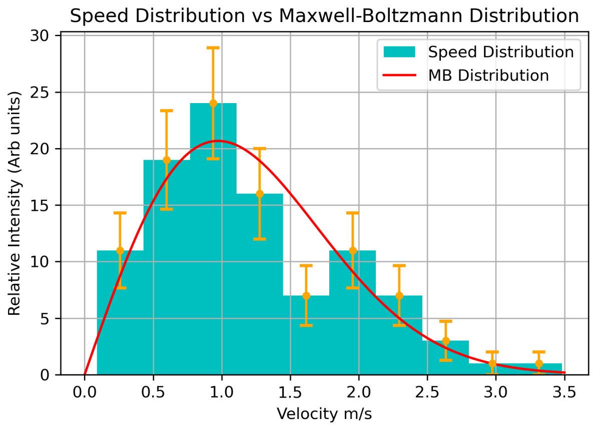

Fig. 7. This is a histogram of absolute speed of the balls. The relative intensity here is the number of Balls in the Simulation. This was ran for 1000 frames. As you can see, this follows extremely closely to the Maxwell Boltzmann Distribution. The uncertainties here follow $\sqrt{N}$.

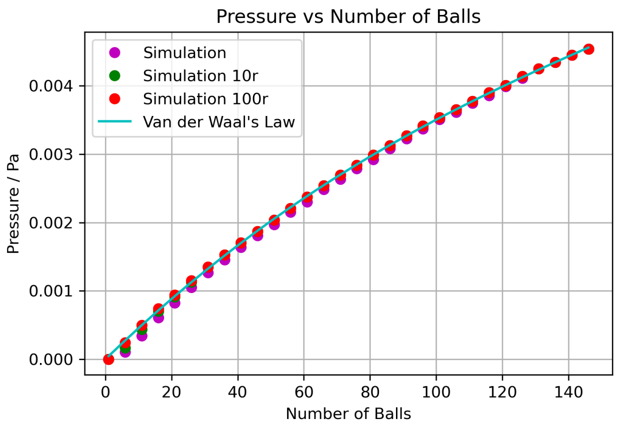

Fig. 8. The Pressure vs the Number of Balls in the Simulation. As you can see this follows almost perfectly with the Van Der Waals theory. This graph also shows that this is independent of ball radius.