Using OOP to Test the Kinetic theory of a Gas

Immanuel Adewumi

4th December 2018

Abstract—**This experiment explores the use of computational methods in simulating the kinetic theory of gases. Particles in a circular container were simulated with random non-overlapping positions and velocities. They were left to run, colliding with each other and the inner walls of the container for a set period. During this process the code was set up to probe the system and take repeat and average measurements for state variables such as pressure and temperature. The volume was a controlled variable. Using this method, we tested the validity of the Van der Waals formula in describing the kinetic theory of gases. A best-fit line of the Pressure versus 1/(V-b) yielded a gradient of 0.30±0.08 which included the expected gradient of NkbT = 0.35 for the simulation, and the pressure versus time yielded a value of 2 ± 5 × 10-23 which included the expected Nkb/(V-b) value of 5.4. The error values are high due to temperature fluctuations.**

I. Introduction

The Kinetic theory of gases [1], first postulated by Daniel Bernoulli 1738, is a model which describes a gas as many point particles in constant random motion. The randomness is a result of their repeated collisions with a in which they are enclosed, as well as with each other. It assumes that particles in a gas have a known mass with negligible size, thus can be modelled as hard point like spheres which undergo elastic collisions. In the Kinetic theory of gases, physically observable macroscopic properties (such as pressure) of gases are a resultant net effect of the individual particles.

This report presents the findings of an object-oriented programme investigating the Kinetic theory of gases. The particles are modelled as 'red' 2-dimensional circles with random motion, enclosed in a larger circular container.

II. Theory

A. Elastic Collisions

One of the assumptions of the kinetic theory of gases is that the particles collide elastically with no overall loss in total momentum or kinetic energy. As our simulation was modelled as a circular container filled with randomized ball particles it followed the kinematic equation (1)[2].

$$(\boldsymbol{r_1} - \boldsymbol{r_2} + \boldsymbol{v_1}\boldsymbol{\delta t} - \boldsymbol{v_2}\boldsymbol{\delta t})^2 = (\boldsymbol{R_1} \pm \boldsymbol{R_2})^2, \quad (1)$$

where $\boldsymbol{r_1}$, $\boldsymbol{r_2}$, $\boldsymbol{v_1}$, $\boldsymbol{v_2}$, $\mathbf{R}_1$ and $\mathbf{R}_2$, are vectors representing the positions, velocities and respective radii of colliding spheres, and $\boldsymbol{\delta t}$ represents the time until the next collision. In this investigation the system is set up in such a way that the container itself was defined as a ball and so the negative component of the ± sign in equation (1) refers to a collision from within the ball (overlapping). By using the magnitude values of the relative position $\| \boldsymbol{r_1} - \boldsymbol{r_2} \|$, velocity $\| \mathbf{v_1} - \boldsymbol{v_2} \|$, and radius $\| \boldsymbol{R_1} \pm \boldsymbol{R_2} \|$, and rewriting equation (1) as a quadratic the values of time to collision $\boldsymbol{\delta t}$ can be found. A negative value represents a collision which may have happened in the past, a positive value represents the time until a future collision, whereas a complex valued $\boldsymbol{\delta t}$ represents a scenario where the balls are not travelling along collision paths.

B. Particle Energy

Ideal Gas-2d

It can be proven, based on the assumptions of the Kinetic theory of gases, that each particle in a gas has a total kinetic energy E_k of $nk_bT/2$ were n is the particle's number of degrees of freedom, k_b is Stefan Boltzmann's constant and T is the temperature of the gas. This is known as the equipartition principle [2] in thermodynamics. In this investigation a 2-dimensional programme was used and so each particle only had 2 degrees of freedom. Thus, each particle had kinetic energy of the form

$$E_k = k_b\mathrm{T}, \quad (2)$$

shown in equation 2. Based on the Kinetic theory model for gases it is assumed that the particles have no interaction forces except during collision. This means the particles have no potential energy, thus, the total internal energy of our system is equivalent to its total kinetic energy.
In the ideal gas equation, shown in (3);

$$PV = Nk_b\mathrm{T}, \quad (3)$$

where P is macroscopic pressure, V is volume, N is the number of particles, k_b is the Stefan Boltzmann constant and T is the temperature. This equation governs the state variables for an ideal gas in thermal equilibrium.

Non-Ideal Gas

For non-ideal gases there are intermolecular forces present. The ideal gas law is thus not a fully accurate formula for describing the state-variable relations in a non-ideal gas. Instead, the Van der Waals equation (4) [3] can be used.

$$\left(P - \frac{a}{\mathrm{V}^2}\right)(V - b) = Nk_b\mathrm{T}. \quad (4)$$

In equation (4) new variables *a* and *b* introduced. Variable *a* provides the correction for the presence of intermolecular forces in a non-ideal gas and *b* represents the volume excluded, the total non-negligible volume of the surrounding

container occupied by the particles [4].

In this experiment the intermolecular forces were assumed negligible, however, due to limitations of object-oriented programming in python it was not possible to assume negligible size for the gas particles. So, for our simulation the Van der Waals equation (4) reduces to

$$P\,(V-b) = Nk_b\mathrm{T}, \qquad (5)$$

Maxwell Boltzmann Distribution

The Maxell Boltzmann equation (6)[5] is a probability distribution that defines the distribution of particle speeds in an ideal gas at a given temperature T.

$$f(v) = \frac{mv}{k_bT}e^{-m/2k_bT}, \qquad (6)$$

In the equation *m* represents the mass of each particle and *v* the velocity.

III. Programme Structure

A. Code Structure

In the simulation two major class objects were used; 'Ball' and 'Simulation'. The Ball class contained all the methods required for instantiating a ball object, defining its parameters (mass, radius, position and velocity), finding the time to collision between itself and an incoming object, as well moving the ball and colliding it with the incoming object. Each particle was instantiated as an object within the Ball class with its various parameters. The container surrounding the particles was also instantiated within the ball as a much larger ball with a much larger mass. For this reason, the instance method __init__() for the Ball class took an argument called iscont which checked if the object being instantiated was a Container or particle by checking the truth value of the argument.

The Simulation class took arguments to determine the number of particles used, the total time elapsed during the simulation run, the position and velocity of particles as well as the radii for both the Container and particles. It included attributes finding parameters of the system (total kinetic energy, average kinetic energy of particles, and temperature), methods for creating a ball list of completely randomised and non-overlapping particles, a next_collison() function which finds the time to next collision and moves the simulation to that point in time, a run function that animates and runs the simulation, and a plot function which plots different graphs for analyses.

IV. Investigations

Pressure versus Time

Fig. 1 shows the plot of the pressure against time for a run of the system over 100 frames, using a 100 particles of mass value 0.01 and radius 0.05. At the start of the simulation is started the particles had their respective initial velocities and had not yet undergone any collisions. As the time elapsed they began to undergo more particle to particle collisions, and less container collisions as they became more spread out, and the system approached thermodynamic equilibrium. This is the reason for the initial sharp fall, followed by a stabilisation of pressure shown in Fig. 1.

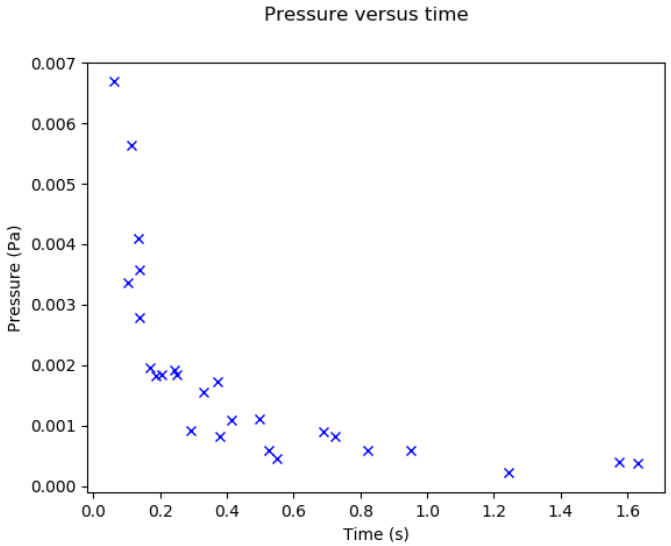

Fig. 1 Shows the plot of Pressure during simulation plotted against Time. The simulation was run over 100 frames using 100 particles of mass with value 0.01 and radius 0.05. Units were not accounted for as only the physical principles were being tested.

Radial Distance from Centre

Fig. 2 shows a plot of the radial particle distribution from the centre of the container. The distribution increases, on average, uniformly with distance from the centre as the circumference spanned out by a given radial distance from the centre increases uniformly with R with the relation Circumference = $2\pi R$. This means more space for particles to potentially occupy.

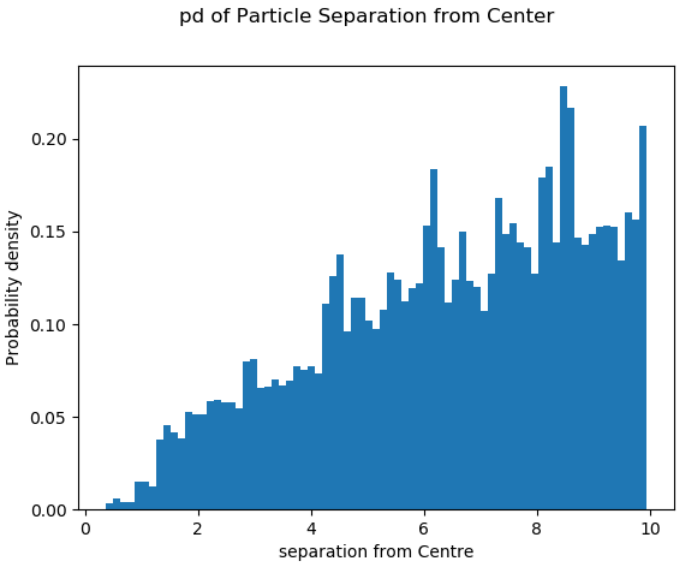

Fig. 2 Shows the distribution of radial particle separation from the container centre. The simulation was run over 100 frames using 100 particles of mass with value 0.01 and radius 0.05. Units were not accounted for as only the physical principles were being tested.

Checking for Energy Conservation

The program was defined with a Kinetic Energy check function which calculated the total kinetic energy of the system after each collision. Fig. 3 shows the plot of Kinetic energy against collision count. As is expected for a system total kinetic energy conserved the plot shows an almost perfectly horizontal line.

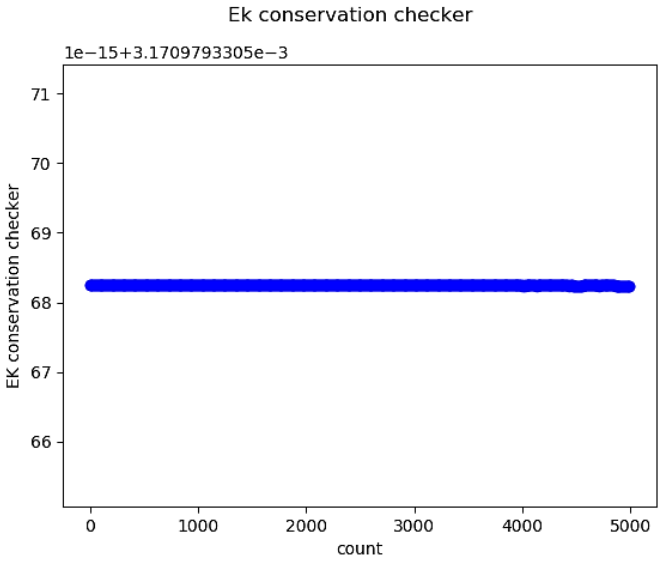

Fig. 3 Shows the plot of total Kinetic energy against collision-count checked over each iteration of the simulation. It is horizontal which shows that the total kinetic energy of the system remains constant during the simulation. The simulation was run over 100 frames using 100 particles of mass with value 0.01 and radius 0.05. Units were not accounted for as only the physical principles were being tested.

Testing for Van der Waals relations

As previously explained, the form of Van der Waals relation which matches our simulation set-up is equation (4). Two simulation plots to test this relation were done. The first Fig. 4 was a plot of *P* against 1/(*V-b)* keeping T relatively stable, and the second Fig. 5 was a plot of *P* against *T* keeping *V* constant. T was not kept completely constant over all iterations of the simulation because on each iteration the particles were generated with random velocities and position, but within a fixed range. Thus, the temperature fluctuations are expected to average out over multiple iterations. This set-up I feel more closely resembles reality.

As *N* and k_b are known constants the accuracy of the Van der Waals law in describing OOP system can be tested using the gradients of the plots. Both plots were initiated with 100 particles of mass 0.01 and radius 0.05. They were run over a hundred frames.

With these settings the linear correlation is not as strongly seen as with higher mass and particle radii values. However, with larger values, the best fit lines of the P versus T and P versus 1/(V-b) deviate from the values they are expected to be. By observing the nature of their Pressure versus time graphs I could deduce that this was likely due to the system not reaching thermal equilibrium quickly enough before the code began taking readings.

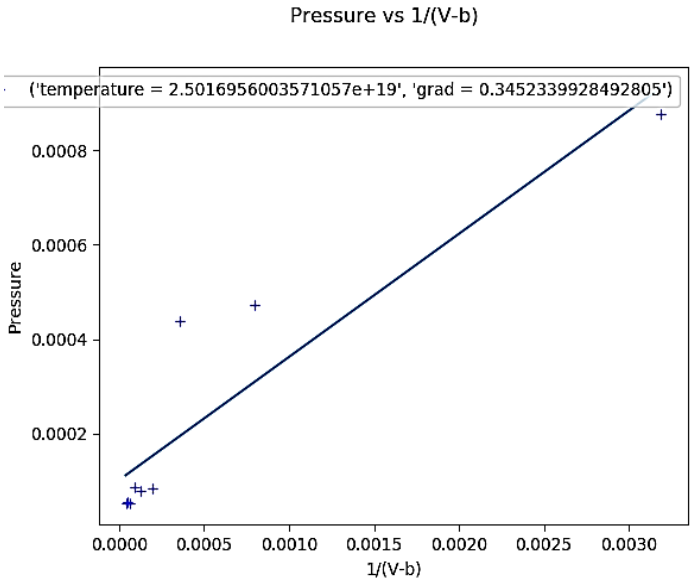

Fig. 4 Shows the plot of pressure against 1/ (Volume – b), taken over simulation iterations in the region where pressure has stabilised. Here b is the volume excluded. The legend shows the average temperature calculated over the simulation run period and the grad value shows the expected gradient value Nk_bT calculated directly using the temperature T and the known values N for particle number and kb Bolttzmann's constant. A best fit line is also found using the Python's Scipy Optimize function.

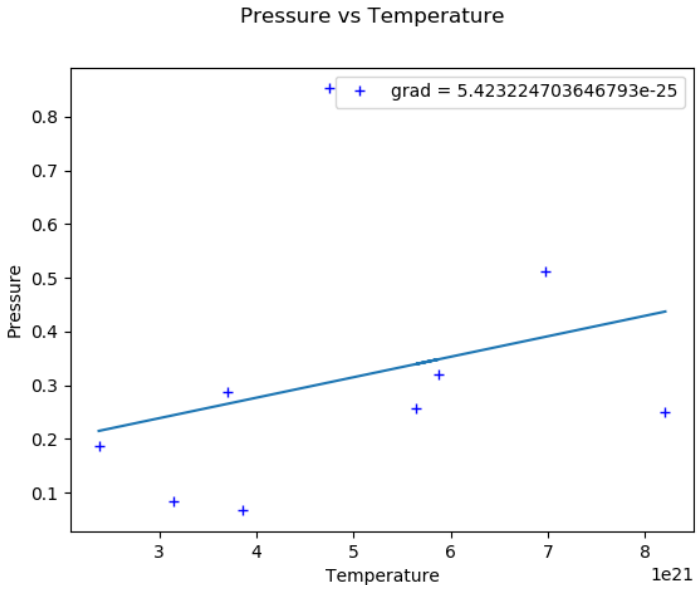

Fig. 4 Shows the plot of pressure against temperature, with fixed volume, taken over simulation iterations in the region where pressure has stabilised. Here b is the volume excluded. The legend shows a grad value which is the expected gradient value $Nk_b/(V-b)$ calculated directly using the set volume and known values N for particle number and kb for Bolttzmann's constant. A best fit line is plotted using the Python's Scipy Optimize fumction.

Maxwell Boltzmann distribution

The code was set to plot a histogram of the particle speed distribution. As is seen in Fig. 6 its distribution follows the Maxwell-Boltzmann equation (5) to a reasonable extent. The simulation was initiated with 100 particles of mass 0.01 and radius 0.05. They were run over a hundred frames.

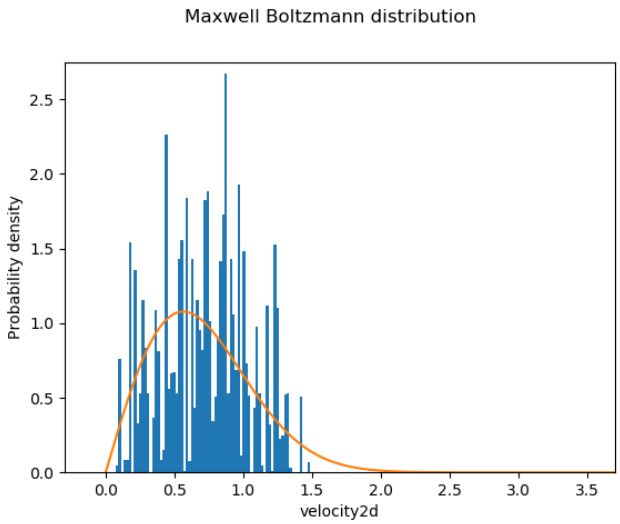

Fig. 6 Shows the plot of the speed distribution of the particles in the system. It is fitted with a Maxwell Boltzmann curve which took average velocity values from the simulation itself.

With larger particle mass and radii values the Maxwell Boltzmann relation is more clearly seen. This is possibly due to less relative fluctuations in Temperature due to the way the code was set up.

V. Result Analysis

During experimentation with the code an interesting observation was made. For relatively low values of particle velocity, and large values of particle radius and mass, the simulation was a poor model for demonstrating the Van der Waals corrections, as it took a much longer time to reach thermodynamic equilibrium (the point at which the pressure in Fig. 1 stabilises). However, using smaller radius and mass values for the particle yielded results which agreed with expectations. This is likely because large particle radii and masses would violate some of the underlying assumptions of the kinetic theory of gasses, one of which is they must have almost negligible size in comparison to their particle to particle separations.

When the code was run over 100 frames with 100 particles of mass 0.01 and radius 0.05, the results shown in Fig. 4 and Fig. 5 were gotten. The best-fit line of the Pressure versus 1/(V-b) yielded a gradient of 0.30±0.08 which includes the expected gradient of $Nk_bT = 0.35$, and the pressure versus time yielded a value of $2 \pm 5 \times 10^{-23}$ which includes the expected Nkb/(V-b) value of 5.4. The error values are high due to temperature fluctuations.

VI. Conclusions

Based on the results of this OOP exercise it can be said with confidence that an object-oriented programming approach, using Python 3.7, can be used successfully to model the kinetic theory of gases. The best-fit line of the Pressure versus 1/(V-b) yielded a gradient of 0.30±0.08 which includes the expected gradient of $Nk_bT = 0.35$, and the pressure versus time yielded a value of $2 \pm 5 \times 10^{-23}$ which includes the expected Nkb/(V-b) value of 5.4. The error values are high due to temperature fluctuations.

Possible extensions might have been to model the system after a real monatomic molecule like hydrogen. However, as real-life gas systems are highly complex (having very high numbers of components), they quickly become computationally infeasible to perform with our system, due to things such as rounding point errors in python.

References

[1] Maxwell, J.C. (1860) "Illustrations of the dynamical theory of gases. Part I. On the motions and collisions of perfectly elastic spheres,"
Clerk Maxwell, James. "On Physical Lines of Force"

[2] Website: available online on imperial blackboard bb.imperial.ac.uk. Year 2 Computing Project: Thermodynamics

[3] Website. Available online: http://hyperphysics.phy-astr.gsu.edu/hbase/Kinetic/eqpar.html [accessed: 05/02/2019]

[4] Silbey, Robert J.; Alberty, Robert A.; Bawendi, Moungi G. (2004). Physical Chemistry (4th ed.). Wiley. ISBN 978-0471215042.

[5] Chang, Raymond (2014). Physical Chemistry for the Chemical Sciences. University Science Books. p. 14. ISBN 978-1-891389-69-6.

[6] Statistical Physics (2nd Edition), F. Mandl, Manchester Physics, John Wiley & Sons, 2008, ISBN 9780471915331