# Thermodynamics Snookered

Ewan Saw

*Abstract*— **in this computational experiment, we used object-oriented programming to create a 2-dimensional simulation of an ideal gas consisting of particles interacting in a container. It is quickly observed that as a result, macroscopic thermodynamic properties quickly arise from the simulation. The simulation behaves and obeys the ideal gas law, as well as other conservation laws, a Maxwell-Boltzmann distribution is also quickly obtained as the system reaches equilibrium. Though functional, the simulation could have been better optimised to run experiments more efficiently with a greater number of balls.**

## I. Introduction

THE kinetic theory of gases describes the study of the physical properties of gases at the microscopic level. By modelling them as a large collection of small and identical particles, we are able to study their dynamics using our knowledge of classical mechanics and relate them to the macroscopic "thermodynamic" properties that emerge (Chang, 2018). This theory was catapulted into recognition when in the 19th century, Maxwell and Boltzmann each generalised and contributed to aspects of Statistical Physics (Britannica, n.d.), which in turn, established this theory as an important concept in thermodynamics.

This same concept is then applied here in a computational experiment, in which a 2-dimensional simulation of an ideal gas is created, composed of a large number of particles interacting in a container. By varying the initial conditions and various other factors involved, this simulation was then investigated to observe how macroscopic quantities such as pressure and temperature arise and react to such changes.

The behaviour exhibited by said particles bouncing about a container is therefore no different to say, a dynamical system of billiards balls rebounding off a snooker table, as the basic mechanics revolving around their collisions are one and the same.

## II. Theory

### A. Description of Ideal gases

An ideal gas is a theoretical model of gases in which its behaviour is predicted based off a number of assumptions (Shapley, 2011). All particles of the gas are said to be in constant motion and their collisions with the container are the sources of pressure that one would measure. The particles are also negligible in size in comparison to the volume occupied by the gas; the particles also do not interact, as it is assumed that no other forces are acting on or exist between them. Their properties are described by the ideal gas law

$$pV = Nk_BT, \tag{1}$$

Where $k_B$ is the Boltzmann constant and the pressure, volume, temperature and number of molecules in the gas are defined by $p$, $V$, $T$ and $N$ respectively. In this case however, an area would be used instead of a volume, as this is a 2-dimeensional simulation.

An example of inter-particle interaction that one would otherwise observe in a real gas would be the interatomic forces due to the Lennard-Jones potential (Naeem, 2019). Given these assumptions, a computational model of an ideal gas becomes more tangible in comparison to any attempt at modelling a real gas. Despite this, the macroscopic relations obtained from studying an ideal gas gives a surprisingly accurate description of reality.

### B. Collisions between particles

By interpreting the molecules of a gas as hard spheres (or in this case, 2-dimensional balls) that collide elastically between themselves and a circular container, we can describe the time taken to each collision, $\delta t$, between any two particles 1 and 2 by (Colling & Paterson, 2013)

$$(\boldsymbol{r}_1 + \boldsymbol{v}_1\delta t - \boldsymbol{r}_2 - \boldsymbol{v}_2\delta t)^2 = (R_1 \pm R_2)^2. \tag{2}$$

Where $\boldsymbol{r}$, $\boldsymbol{v}$ and $R$ denote the position, velocity and radius of the particles respectively. A negative solution for $\delta t$ would imply that there was a collision in the past whilst an imaginary solution signifies that two particles never collide at all. Note that a '+' on the right-hand-side of equation (2) distinguishes a ball-to-ball collision whilst a '−' would be for a ball-to-container collision. By solving this equation for $\delta t$ repeatedly for every possible pair of collisions, we are able to form the backbone of our simulation.

As all collisions are also elastic, the total kinetic energy of the system

$$E = \sum_i \frac{1}{2} m_i |\boldsymbol{v}_i|^2 , \tag{3}$$

must be conserved, where $m_i$ denotes the mass of ball $i$. This, along with the conservation of momentum allows us to calculate the resulting velocities of the balls after every collision in terms of their masses and initial velocities

$$\begin{cases} v_1'\hat{\boldsymbol{r}}_{12} = \left[\left(\dfrac{m_1 - m_2}{m_1 + m_2}\right)v_1 + \left(\dfrac{2m_2}{m_1 + m_2}\right)v_2\right]\hat{\boldsymbol{r}}_{12} \\ v_1'\hat{\boldsymbol{r}}_{12} = \left[\left(\dfrac{2m_1}{m_1 + m_2}\right)v_1 + \left(\dfrac{m_2 - m_1}{m_1 + m_2}\right)v_2\right]\hat{\boldsymbol{r}}_{12} \end{cases}, \tag{4}$$

where $v_1$ and $v_2$ are the magnitudes of their velocities along the vector between the two centres of the balls, $\hat{\boldsymbol{r}}_{12}$, and their primes denote the resultant velocities.

### C. Pressure in container

As the balls continuously collide with each other, every collision a ball makes with the container imparts an impulse

$$\Delta\rho = 2mv \tag{5}$$

on the container, with $v$ being the velocity perpendicular to the wall of the container. We can then find the force exerted by the balls on the container by

$$F = \frac{d\rho}{dt}. \quad (6)$$

This in turn allows us to determine the overall pressure the balls exert on the container

$$P = \frac{d\rho}{Cdt}, \quad (7)$$

where $C$ denotes the circumference of the container. The pressure in this case is defined as the 'Force applied per unit length' as opposed to 'per unit area' as we are considering a 2-dimensional system.

This then gives us an initial idea of how macroscopic properties arise from such a system and how we make the extension to the thermodynamics our 'gas' is explored in section IV.

## III. Method

### A. Ball and Container Classes

Using object-oriented programming (OOP), the simulation was designed with the main focus of data and objects in mind rather than functions and logic (Rouse, 2020). In typical OOPs, objects can be defined to be a set of data which have unique attributes and methods.

Starting with the ball class, it was defined to have scalar attributes of mass and radius, whilst attributes of position and velocity were given to be 2-dimensional arrays. An extra 'patch' attribute was also defined. This allowed for an illustration to be assigned to each ball for the animation, and a '*get patch*' method was defined so that the objects could be animated for every collision. Other methods were also defined which acted on the various ball attributes.

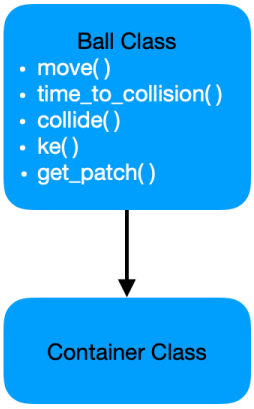


Fig. 1. A Ball class with its defined methods and the Container class, which inherits all attributes and methods from the Ball class.

As shown in Fig. 1, a Container Class was also defined which inherited from the Ball Class, the only exception being that the container has a very large mass ($9.9 \times 10^{10}$ in this case) as it should not move. It is also given a radius of 15 which is meant to be much larger than the radius of the balls.

The most basic methods implemented were the '*move*' and '*ke*' methods. The '*move*' method involved moving the position of the ball along its velocity over a given time $\delta t$

$$\boldsymbol{r}_f = \boldsymbol{r}_i + \boldsymbol{v}\delta t, \quad (8)$$

with subscripts $i$ and $f$ denoting the initial and final positions $\boldsymbol{r}$ of each ball. The '*ke*' method on the other hand, returned the kinetic energy of the ball, which is the result of equation (3) for $i = 1$.

### B. 'Time to collision' method

The '*time_to_collision*' method was then defined by implementing the solved forms for $\delta t$ of equation (2), which was a quadratic equation. As this method was to be used to determine the time taken until the next collision between any two objects, a number of 'if' loops were implemented to check for conditions in order to obtain the correct solution for $\delta t$.

The first condition to be checked for was the discriminant of the solution. If the discriminant was a negative value, the balls do not collide at all. Hence, the 'if' loop would return an infinite value (where the reasoning for this is described in III *D*). Otherwise, the equation would yield 2 solutions. The desired solution however, would yet again be decided under another 'if' loop. As described in II *A*, a negative solution implies that a collision has happened in the past, so the correct conditions would return the smallest, positive value for $\delta t$, signifying the next possible collision in the future.

Another condition to check for was if they were travelling at the same velocity parallel to each other. This would yet again be checked for in a loop and an infinite value would be returned if it were 'True'. A ball-to-ball collision and ball-to-container collision also needed to be checked for. Again, in equation (2), the '+' sign was used for summing the two radiuses for a collision between two balls whilst the '−' was used for a ball to container collision.

### C. 'Collide' method

Defining the '*collide*' method involved splitting the velocities of the two balls into components parallel and perpendicular to the vector along the centres of the two balls. Equation (4) was then applied to the parallel components of each ball and recombined with the unchanged perpendicular components to update the balls with their new velocities post-collision.

An extra check was implemented in order to ensure a collision was possible. This was done by calculating the total kinetic energy of both balls before and after the collision and checking if they were equal. If the check returned a 'False', a 'TypeError' would be raised.

### D. Simulation Class

Having defined the classes of all our objects, we were then able to define our Simulation class which initialises both classes as attributes. Attributes such as the number of balls in the simulation and the time passed were defined. Extra attributes such as the total kinetic energy of the system, an empty list of impulses and a list of velocities were also added which aid in the investigation of the system in section IV. To account for multiple balls, the simulation also initialises the ball attribute as a list of ball objects.

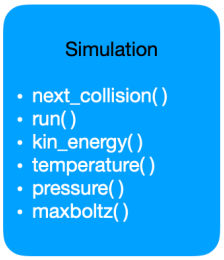


Fig. 2. Simulation class with its defined methods listed.

The core function of the simulation relies heavily on the '*next_collision*' method. This first involves identifying the time taken for the next collision to occur, then moving that system up to that given point in time and performing a collision, changing the velocities of the colliding objects. This repeatedly happens for a given number of frames which is specified in the '*run*' method.

In order to perform the former, an $n \times (n+1)$ matrix of zeroes is first formed, with $n$ being the number of balls. Each row and column in the matrix denote the $n$'th ball in the simulation and the corresponding elements signify the $\delta t$s for the collisions between ball $n$ of the row and ball $n$ of the column

$$\begin{pmatrix} 0 & \delta t_{12} & \dots & \dots & \delta t_{1n} & \delta t_{1c} \\ 0 & 0 & \ddots & \ddots & \ddots & \delta t_{2c} \\ 0 & 0 & 0 & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \delta t_{nc} \end{pmatrix}. \tag{9}$$

Excluding the final column, it is apparent that the $\delta t$s along the upper and lower triangular matrix would be the same. As a result, only values of $\delta t$ above the diagonal of the matrix were calculated. This in turn saves computational time and makes our simulation more efficient. The extra final column of the matrix, which we denote by $c$, will be the values of $\delta t$ for each ball's collision with the wall of the container.

The next step involved determining the actual time taken for the next collision. This was done by retrieving the smallest non-zero element of matrix (9) as well as its index. As mentioned in III *B*, unwanted or imaginary solutions in the '*time_to_collision*' method returned infinite values, this was so that the returned value would never be retrieved by the '*next_collision*' method.

The index of the smallest element also indicates which balls would collide. Hence, the system is evolved by performing the '*move*' method on every ball over the given $\delta t$ and finally colliding the two balls (or the ball with a container).

At the end of this method, the final velocities of all balls, the total kinetic energy of the system, and any impulses imparted onto the container from the balls are appended to their respective attribute lists. The time attribute is also updated with the $\delta t$ of the recent collision.

The '*run*' method was then added which performs the '*next_collision*' for any desired number of frames, whilst having the option to animate the simulation using the '*get_patch*' function of the objects.

### *E. Initialising the system*

Before running the simulation, an efficient way of initialising the system with multiple balls had to be considered. Initially, an array of 2-dimensional elements was created which pre-specified evenly spaced positions within a $5 \times 5$ grid. Though simple, it only allowed for up to a maximum of 25 balls to be initialised and any attempt at using more balls would have required remaking a new grid.

Another method which was eventually used involved rooting the desired number of balls and using the 'linspace' function to create an evenly spaced square grid of positions and giving each ball random velocities. Despite this method only working best for square numbers, it was arguably more elegant.

### *F. Ball sticking glitch*

When running the simulation, some balls randomly overlapped at times, causing them to stick and repeatedly collide with each other. This was fixed by moving the system by a small time-margin at the start of the '*next_collision*' method.

## IV. Results

### *A. Kinetic Energy and Temperature*

Treating our simulation now as a gas, we were able to use the thermodynamic relation of temperature to the energy of each particle

$$T = \frac{2E}{Nk_B n}. \tag{10}$$

$E$ represents the energy of each particle, which in this case is purely kinetic, $N$ is the total number of particles and $n$ is the number of degrees of freedom of each particle, which is 2 in this simulation. Using this, we are able to calculate the Temperature of our 'gas'.

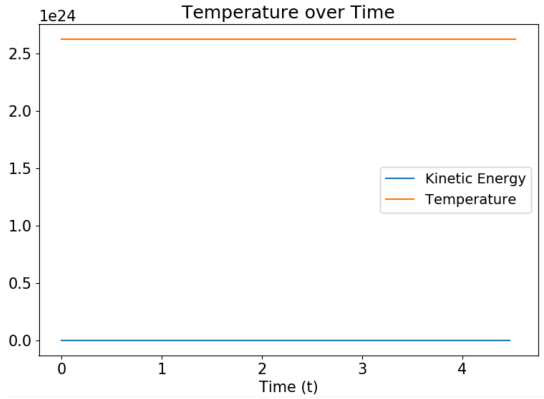


Fig. 4. Plot of the temperature of the system over time (orange line), along with the total kinetic energy (blue line), in which both properties are constant throughout.

In Fig. 4, it is shown that the temperature is much larger in comparison to the kinetic energy. This is due to the inverse Boltzmann constant factor in equation (10). This also validates the simulation as it obeys the law of conservation of energy.

### *B. Ideal gas relations*

Using equation (7), a method was added to the simulation to plot how the pressure of the gas varied with time, giving Fig. 5.

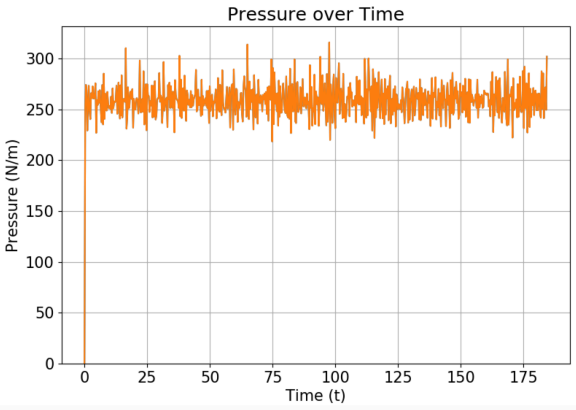


Fig. 5. Plot of pressure against time for a simulation running with 100 balls. Pressure initially starts at 0 and quickly climbs before stabilising about a particular pressure.

In Fig. 5, the pressure in the container was not completely constant from the start as it varied depending on the collisions of the balls with the container, it then began to stabilise after a given amount of time as the system began to reach thermodynamic equilibrium. As such, readings were decidedly obtained after 20 seconds of the simulation running in order to obtain more accurate results. In order to investigate the physical properties of our 'gas', the pressure was then plotted against a varying temperature.

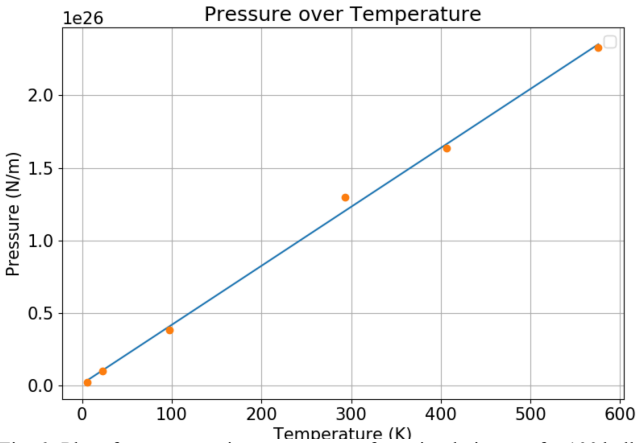


Fig. 6. Plot of pressure against temperature for a simulation run for 100 balls. A line of best fit was plotted over the points of data which travels through the origin.

Fig. 6 implies a linear relationship between the two macroscopic properties. This agrees with the ideal gas law equation (1), which shows that pressure in an ideal gas is directly proportional to the temperature and should give a straight line of gradient $Nk_B/V$.

Similarly, plots of pressure against the volume (area in this case) of the container gave an inverse relationship whilst plots of pressure against the number of balls in the simulation gave a proportional relationship. This all agrees with our assumptions as well as equation (1) which strengthens the confidence in our simulation.

### C. *Van der Waals equation*

The Van der Waals equation is an equation of state which corrects for the volume of gas particles and the attractive forces or the molecules (Colling & Paterson, 2013),

$$\left(P + a\left(\frac{N^2}{V^2}\right)\right)(V - Nb) = Nk_BT. \qquad (11)$$

The constants $a$ and $b$ each represent the magnitude of the intermolecular forces between particles and the excluded volume respectively. In this simulation, $a = 0$ as the particles were assumed to not interact with each other.

This leaves us with an equation which resembles equation (1), but with an added term which corrects for the volume of the particles. This implies that our simulation becomes a better approximation of an ideal gas as the volume of our particles tended towards 0. This was observed as systems initialised with particles of larger radii gave more inconsistent results with a larger error that agreed less with the ideal gas law as opposed to balls of smaller radii.

### D. *Maxwell-Boltzmann Distribution*

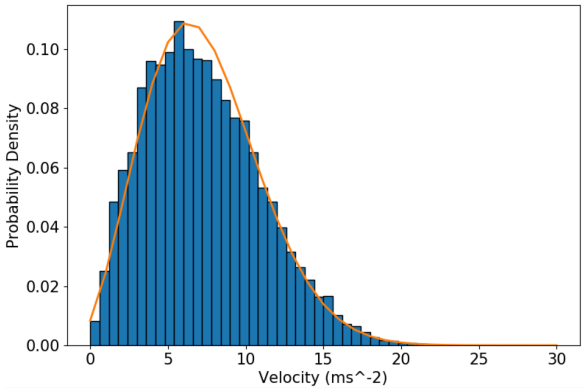


Fig. 7. Histogram of collective velocity distributions for 100 balls with max speeds of 10ms$^{-1}$ with a fit for a Maxwell-Boltzmann distribution plotted.

By plotting histograms of the velocities of the balls over 1000 collisions from being initialised with random velocities, we can see in Fig. 7 that the result obtained from our simulation agrees with the Maxwell-Boltzmann distribution. There is further agreement in Fig. 8, when the histogram obtained for higher temperatures corresponds to a shift of the graph to the right with a lower peak.

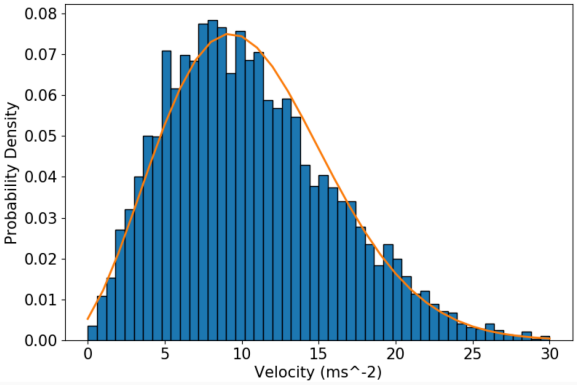


Fig. 8. Histogram of collective velocity distributions for 100 balls with max speeds of 15ms$^{-1}$

This validates the simulation as an appropriate model as its agreement with the Maxwell-Boltzmann distribution signifies that gas-like properties emerge from our simulation. Such that when it is provided with a larger amount of energy, the variance of the distribution increases as more particles have a higher kinetic energy.

## V. Conclusion

Using OOP, we aimed to model a gas as a large collection of balls and investigate the thermodynamic properties that emerge from it. As a whole, we managed to reach a working model for the simulation and were able to explicitly demonstrate concepts of kinetic theory and thermodynamics. From being initialised with random velocities, we see the system evolve into thermodynamic equilibrium as the pressure stabilises and the distribution of velocities quickly approach a Maxwell-Boltzmann distribution.

It was also shown that the simulation behaved much like a normal gas would as the measured macroscopic quantities obeyed the ideal gas law as well as its respective conservation laws. The limit being the Van der Waals equation, showing that there are corrections to be made for gas particles with a finite radius

Better measures could have been taken to improve the code by making it more efficient however. As for every collision, only the 'time to collision' for two balls would change whilst other balls remain unaffected.

## VI. References

Chang, J., 2018. *Simulating an Ideal Gas to Verify Statistical Mechanics,* Stanford: s.n.

Britannica, E. o. E., n.d. *Kinetic theory of gases.* [Online]
Available at: https://www.britannica.com/science/perfect-gas
[Accessed 12 February 2020].

Shapley, P. P., 2011. *Real Gases.* [Online]
Available at: http://butane.chem.uiuc.edu/pshapley/GenChem1/L14/2.html
[Accessed 12 February 2020].

Naeem, R., 2019. *Lennard-Jones Potential.* [Online]
Available at: https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Physical_Properties_of_Matter/Atomic_and_Molecular_Properties/Intermolecular_Forces/Specific_Interactions/Le
[Accessed 13 February 2020].

Colling, D. & Paterson, C., 2013. *Thermodynamics Snookered.* [Online]
Available at: https://bb.imperial.ac.uk/bbcswebdav/pid-1678048-dt-content-rid-5350819_1/courses/BJ201910/Y2%20Computing/Scripts/Projects/.build/html/Snooker.html#appendix-sn-a
[Accessed 10 February 2020].

Rouse, M., 2020. *object-oriented programming (OOP).* [Online]
Available at: https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP
[Accessed 11 February 2020].