Computing Project B:

Thermodynamics Snookered

I Kit Cheng
941640

Submitted on 04/11/16

Abstract

The motion of gas particles inside a 2-dimensional container was simulated alongside with an animation using object-oriented programming in Python to illustrate the links between modern descriptions of gases and classical thermodynamics. The motion of the particles was modelled by solving a quadratic equation relating the time to collision and the position and velocity vectors of the particles. It was found that the model was consistent with the ideal gas law, and obeyed the laws of conservation of energy and conservation of angular momentum. However, the slope of the PT diagram gave a value of k_B that was 55% larger than the true value, which shows that it is not perfect.

1. Introduction

An animation is an effective way to show the evolution of a physical system. In problems where complexity is high, it is advantages to use object-orientated programing because the problem can be separated into small blocks called objects which hides the internal complexity, giving a clean end-user interface.

The purpose of this project is to use object-oriented programming in Python to build a 2-dimensional simulation of gas particles in a container and produce an animation to illustrate and investigate the relationship between the thermodynamic state variables and their relation to modern descriptions of gases, i.e. how kinetic energy is linked to temperature and how the force of particles bouncing off the container produces pressure. By repeatedly solving a quadratic equation relating the time to collision with the position and velocity vectors for each particle, a minimum time to collision is returned and tells the simulation to move forward in time until the next collision where changes to the velocities of the particles involved are made, and the cycle repeats.

2. Theory

In order to simplify the problem, the Lennard-Jones potential which dictate the inter-atomic interactions is not considered. Instead, the following assumptions are made: the particles are hard spheres and only interact when they collide, the collisions are perfectly elastic which means energy and momentum are conserved after collisions, and that the particles are confined in a circular container. These assumptions are very much analogous to the interaction of billiard balls in a game of snooker. The dynamics of their motion is given by

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

where $\vec{r}$, $\vec{v}$ and R are the position, velocity and radius of the particle, or the container, respectively, and δt is the time until next collision [1]. Solving for δt gives different scenarios: when Eq. 1 has no real solutions for δt; it means the particles cannot be in this configuration, when Eq. 1 only has negative solutions; it means the particles will not collide again, when Eq. 1 has both a positive and a negative solution; the positive value corresponds to next collision and the negative one corresponds to a collision that already happened, when Eq. 1 has both positive solutions; the smaller value is

the correct time to next collision whereas the bigger value refers to a collision after the particles have gone through each other. The significance of $\pm$ on right-hand side of Eq. 1 signifies a collision with the container (see Lab Book pg.90).

To solve for the velocity after an elastic collision, the relative velocity $\overrightarrow{v_1} - \overrightarrow{v_2}$, is projected onto the vectors which are normal and parallel to the line of centres, $\overrightarrow{r_1} - \overrightarrow{r_2}$, using dot product (see Lab Book pg.91 for diagram). The normal component is unchanged by the collision because there is no impulse perpendicular to the line of centres. The parallel component is simplified to a 1-D collision which can be solved using the laws of conservation of linear momentum and kinetic energy. The final velocity of the particles are then given by

$$\overrightarrow{v_1}' = \overrightarrow{v_1} - \frac{2\,m_2}{m_1 + m_2}\langle \overrightarrow{dv_1}, \widehat{dr_1} \rangle\, \widehat{dr_1} \quad (2)$$

$$\overrightarrow{v_2}' = \overrightarrow{v_2} - \frac{2\,m_1}{m_1 + m_2}\langle -\overrightarrow{dv_1}, -\widehat{dr_1} \rangle(-\widehat{dr_1}) \quad (3)$$

where $\overrightarrow{dv_1}$ and $\overrightarrow{dr_1}$ are the relative velocity and position vectors [2]. The collision with the container is similar to that with a ball, but instead with a mass m_2 tending to infinity, hence giving a slightly different equation

$$\overrightarrow{v_1}' = \overrightarrow{v_1} - 2\langle \overrightarrow{dv_1}, \hat{r_1} \rangle\, \hat{r_1} \quad (4)$$

After determining the new velocity of the balls, it is used to solve Eq. 1 for the next time to collision and the process repeats.

The accuracy of the model can be tested against well-established laws in physics. From kinetic theory, pressure arises due to the force of collisions between particles and the walls of the container. Hence, it is expected that the pressure depends on the speed of gas particles. From the ideal gas law,

$$PV = Nk_BT, \quad (5)$$

It can be deduced that temperature also depends on speed and hence depends on the kinetic energy (KE) of all the particles. The relationship and can be written as,

$$T = \frac{2\,(KE)}{N_d k_B T}, \quad (6)$$

where N_d is the degrees of freedom (in 2-D, N_d = 2)[3]. The conservation of energy must hold for the model to be accurate, meaning the internal energy (kinetic energy only as potential energy is not considered) of the system should be constant. Angular momentum should also be conserved whilst linear momentum is not because the container is assumed to be stationary, so the impulse from the walls on the ball constitutes an external force.

3. Method

The simulation was achieved using object-oriented programming in Python. A Ball class was created to give all Ball objects (i.e. particles) mass, radius, position, and velocity. A *time_to_collision* method was made to solve Eq. 1 and check each pair of balls for the shortest collision time (see code lines 62-97). It was ensured that only the smallest positive time was chosen from the solutions to Eq. 1. A *collide* method was made to solve Eq. 2, 3 and 4 and change the velocity of the balls after collision (see code lines 99-157). The container was given a negative radius such that an *if* statement could be used to check whether the ball was colliding with a container. A Gas class was created to simulate the motion of all the balls instantiated from the Ball class. It contained methods which allowed an animation to be created, and collected data to establish the relationship between the thermodynamic state variables. These included the temperature (T), pressure (P), P / T ratio, velocity of every particle, kinetic energy and angular momentum. To create the animation, the *FuncAnimation* method of the *matplotlib.animation* module was used. Running the animation alongside data collection was not efficient. Consequently, a Boolean constant

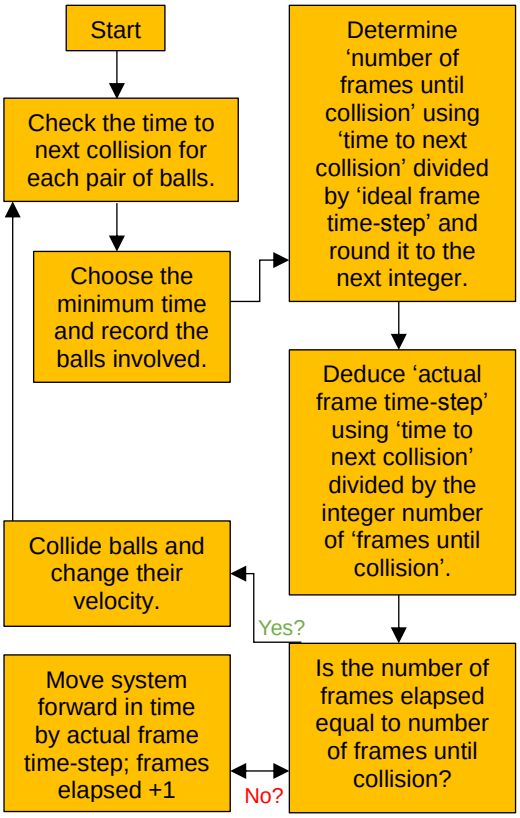

Fig.1 Flowchart showing the logic used to simulate moving particles in a 2-D container.

Animate = True/ False and a method called *simulatePhysics* (see code lines 327-401) was implemented such that large quantities of data collection was possible with less computing time.

N balls with mass of He atoms and radius of 1m were distributed uniformly in plane polar coordinates using a generator module, genpolar.py. The balls were assigned random velocities chosen from a Gaussian distribution with mean of 0, standard deviation between 100 to 1000.The logic used to model the motion of N balls in a container is shown in the flowchart (Fig.1). The rounding up of the 'number of frames until collision' variable to an integer resulted in a small error in in the animation, but it made the calculation much simpler. The time-step per frame after each collision was adjusted to ensure that there was an integer number of steps until the next collision.

4. Results & Discussion

To check whether the model was accurate, it was tested against the conservation of energy and angular momentum. The plots of each respectively are shown in Fig.2. As seen from the plots, both the kinetic energy and angular momentum were not constant over the 7 seconds elapsed. The range of the fluctuations were about 1.2×10^{-35} kgm^2s^{-1} for angular momentum and 1.0×10^{-33} J for K.E. However, given that only 10 particles were simulated in this plot, whereas in reality there would be roughly 10^{23} particles which would increase the kinetic energy to the order of 10^3 J, so the fluctuations are negligible. The reason for fluctuations in the simulation could be due to random fluctuations in floating point numbers. Therefore, the model satisfies the conservation laws to an extent. It was found that when the mean speed of the particles were doubled, both the temperature and the pressure were quadrupled. This was expected from Eq. 5 and Eq. 6, since T is proportional v^2 and P is proportional to T.

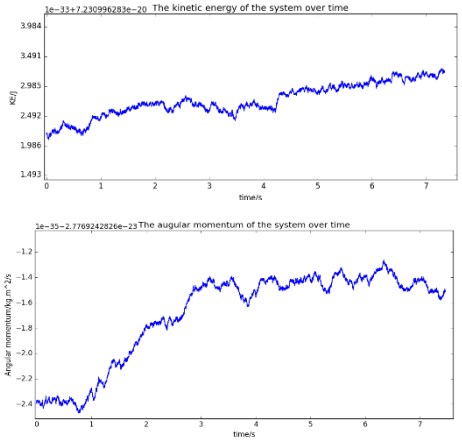

Fig.2 Graphs showing the total kinetic energy and angular momentum of the system were not conserved.

The distribution of velocities is shown in the histogram in Fig.3. The plot also contains the Maxwell-Boltzmann (M-B) distribution (red) and the Python M-B pdf (green) over it. The red curve was consistent with the actual distribution of the velocities, peaking at the most probable speed v_{mp} of 790m/s. This is consistent with the temperature of 300K measured in the simulation, using the relationship $v_{mp} = \mathrm{sqrt}(k_bT/m)$.

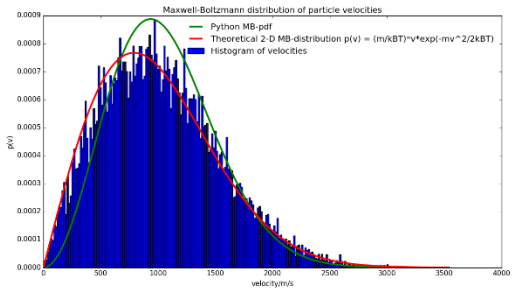

Fig.3 A plot comparing the distribution of velocities to the Maxwell-Boltzmann distribution.

From investigating how pressure and temperature varies with area of container and number of balls, it was found that pressure was inversely proportional to area (volume if 3-D) whilst temperature remained constant. This was expected from the fact that no heat was transferred and no work was done to the gas by changing the initial area, so internal energy was not changed and hence temperature remained constant. When number of balls were changed, the temperature also remained constant and the pressure was proportional to number of balls as expected from kinetic theory and Eq. 5 (see Section 2). The temperature could be explained by Eq. 6, when N is doubled, the total KE is also doubled so temperature remained unchanged. Therefore, the model showed behaviour that was consistent with the ideal gas law (Eq. 5).

The following is a PT diagram with initial condition N = 10, and V = $\pi*10^2$ (Area).

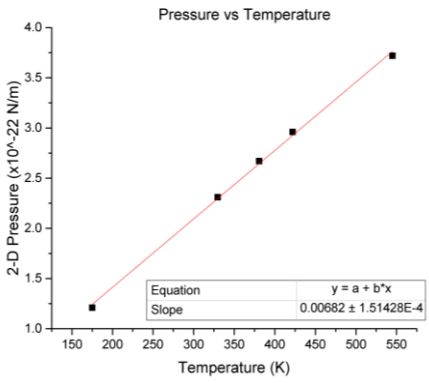

Fig.4 PT diagram.

Using the slope value from Fig.4 and Eq. 5, the Boltzmann constant k_B was calculated to be (2.14 ± 0.00) x 10^{-23} JK^{-1}, which was 55% bigger than the true value of 1.38 x 10^{-23} JK^{-1}. This suggested that the actual area was in fact smaller than $\pi*10^2$ and that the model has neglected the areas of the balls. Consider the Van Der Walls' law

$$\left(P + \frac{aN^2}{V^2}\right)(V - bN) = Nk_BT, \quad (6)$$

since the model did not simulate a potential between the balls, a Using the true value of k_B, a value was obtained as b = 11.2. This coefficient corrects for the area of the balls.

5. Conclusion

In this project, the motion of 2-D gas particles in a circular container was simulated with an animation in Python. The model was consistent with conservation law of energy and angular momentum and showed behaviour that was consistent with the ideal gas law. It also illustrated the link between kinetic energy with temperature, and force with pressure. However, it was found that the model was not perfect when the calculated k_B value was 55% larger using the ideal gas law. The Van Der Walls' law was invoked to get a more accurate representation of the physical system.

References

[1] A MacKinnon (2016) Project B: Thermodynamics Snookered. *Blackboard Imperial.* Available from: https://bb.imperial.ac.uk/bbcswebdav/pid-957933-dt-content-rid-3193661_1/courses/COURSE-PHY2_LAB-16_17/Y2Computing/Y2PyCourse/Students/Projects/html/Snooker.html [Accessed 3rd November 2016]

[2] *Elastic Collision* (2016) Available from: https://en.wikipedia.org/wiki/Elastic_collision [Accessed 3rd November 2016]

[3] D.Eakins (2015) Structure of Matter. *Blackboard Imperial* [Accessed 3rd November 2016]