Gas simulation using hard spheres

George Su

I. Introduction

THE simulation models gas particles as hard spheres which collide perfectly elastically with each other and with the container. It is in 2D and does not account for phase changes. The aim of the simulation is to aid the investigation of gas behavior and the validity of different gas models [1].

II. Method

The simulation was split into 3 separate classes: *Ball*, *Container* and *Simulation*. The *Ball* class contains all the properties of a single particle and operations that are performed by/on the particle. The operations are colliding, moving and calculating the time to the next collision (*time_to_collision*).

The *Container* class inherits from the ball class; however, it is hard coded to have zero initial velocity and it has a mass of 10^{50} kg, which is much greater than those of the particles.

The *Simulation* class contains all the properties of the system as a whole and handles all the operations of the system. The most essential operation is handled by the *next_collision* method, which finds the next collision occurring in the system, performs this collision between the relevant particles and records the parameters that have been altered. This was achieved using 2 separate designs, the initial design was superseded by the second design as the latter had an operational time complexity of $O(n)$ rather than $O(n^2)$ [2].

Design 1

Design 1 generates a *collision matrix*, containing the *time_to_collision* between every particle in the system (Including the container), as well as the *Ball* class instances involved in each collision.

It then uses np.min() to find the collision that occurs the soonest and performs this collision.

This process is repeated every time the *next_collision* method is called. This leads to a time complexity above $O(n^2)$ as the algorithm loops through every particle in the system and compare it with every other particle. The outcome array then also has to be sorted using np.min().

Design 2

Design 2 builds upon the framework of design 1. However, the *collision matrix* only needs to be created once. After creation, the *collision matrix* is sorted in ascending order of *time_to_collision* and the first element is taken as the collision to perform each time *next_collision* is called.

Rather than having to recreate the matrix each time the method is called, only the elements containing the particles that have just collided need to be modified. (As these are the only particles that will have their trajectories modified by the collision.) This modification is done by deleting the old collision information, calculating new *time_to_collision* between the 2 particles that have their trajectory modified and the other particles in the system (this step is skipped for the container if it was involved in the collision), and then reinserting the collision information for these particles back into the *collision matrix* at the appropriate place.

The insertion is done through term-by-term comparison between the new collision elements and the unmodified elements already in *collision matrix*. The new elements are shift backwards in the *collision matrix* until the it meets a term which has a *time_to_collision* greater than itself. This method of insertion is more efficient than using np.min() or np.sort() as it takes advantage of the fact that the *collision_matrix* is already ordered.

In addition, the *combinations* function from python library *Itertools* was used when creating the initial *collision matrix* to only make comparison between every combination of particles in the system rather than every permutation. Since we are making comparison between pairs, this cuts down the number of comparisons by half.

Design Comparison

Design	**1**	**2**
Creating initial *collision matrix*	n^2	$\frac{n^2}{2}$
Regular operation (excluding sorting)	n^2	$2n$
Sorting/Inserting	n	$n <$

Tab 1. The time complexity of the components of each design

As seen in Table 1, the second design has a lower time complexity than the first in every part of the process. The difference becomes apparent for simulations with long run time and high number of particles.

Design	**Time taken (s)**	
	1000 frames	2000 frames
1	129	246
2	71	110

Tab 2. Result of a benchmark test on time taken to run simulation

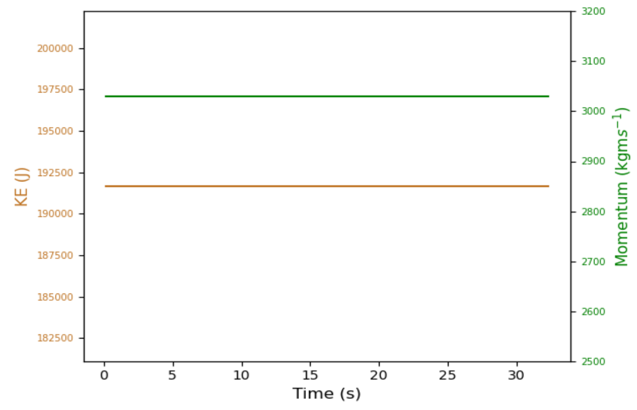
Table 2. Shows the results of a benchmark test testing the time taken to run the simulation for a large number of frames with 100 balls of radius 10, a pause-time of 0.01, inside a container of radius 1000 using the 2 separate designs. We see that for 1000 frames, the second design takes only 55% of the time of the first to complete the simulation. For 2000 frames, this decreases to 44%.

III. Results and analysis

Figure 1 shows that the simulated system conserves total kinetic energy and momentum as expected. Figure 2 shows the variation of the pressure (P) of simulated systems with temperature (T).

The data set represented with marker '●' are from simulations with particles radius (r_p) of 1 unit. While the '**x**' data set are from simulations with r_p of 200 units. For both cases, the container had a constant radius of 1500 units.

The data set with r_p of 1 (hence $r_p \ll r_{container}$) followed the relationship predicted by the ideal gas model closely at low temperatures. At high temperatures, its gradient



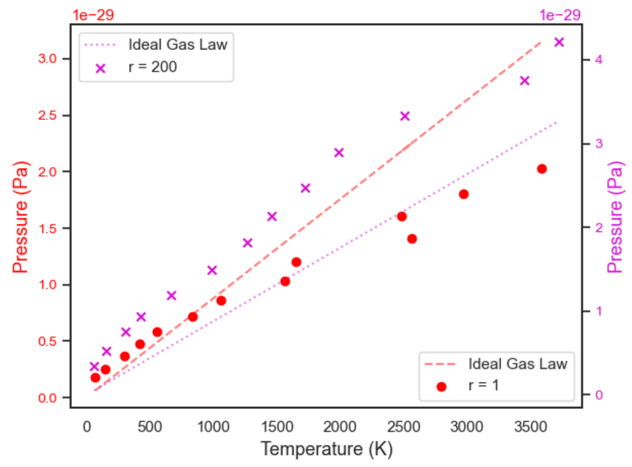


Fig. 1. Total System KE and Momentum vs Time
Fig. 2. Simulated P-T relationship compared to ideal gas equation for 2 types of particles with different radius

flattens, causing it to deviate from the predicted relationship. The data set with the larger r_p of 200 also experiences this flattening of its gradient at high temperatures, however its P values are consistently greater than the predicted values.

The limitations of the 2D simulation as a representation of 3D gases is one of the factors contributing to the discrepancies between our data and the ideal gas model. The simulated gas only has 2 degrees of freedom rather than 3+ degrees of freedom in a 3D ideal gas. Furthermore, even though the volume (V) of the container was used in the calculations for the idea gas equations, the simulation is 2D and the particles can only move within a cross-section of the volume rather than the entire volume. This also impacts the pressure calculations as the particles has less surface area to collide with the container.

Nevertheless, these limitations should affect both data sets to similar extents. Hence it unlikely for them to be the cause of the consistent gap between the obtained and predicted P values for the data set with r_p of 200. Instead, this deviation indicates the failure of the ideal gas model in accounting for the effects of different particle sizes.

Figure 3 explores this concept in deeper more detail by examining the variation of P with r_p. It is shown that at constant T and V, P increases as r_p increases. This dependence is ignored by the ideal gas equation which assumes all particles to have negligible volume in comparison to the container.

One model that does account for this dependency is Van der Waal's correction to the ideal gas equation, given as

$$P = \frac{RT}{V_m - b} - \frac{a}{{V_m}^2}, \tag{1}$$

for 1 mole of gas,

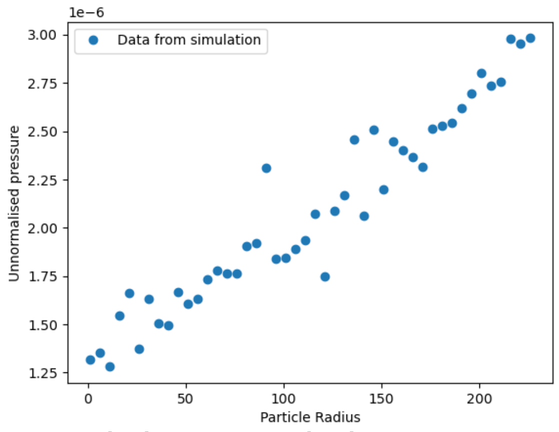


Fig. 3. Unnormalized Pressure vs Particle radius

where V_m is the molar volume of the gas, R is the universal gas constant, a is a gas specific constant relating to the intermolecular forces and b is volume of 1 mole of the gas particles, which is proportional to ${r_p}^3$. [3] Like our data, b is positively correlated with P, with a functional form that could be consistent with the curve seen in Figure 3.

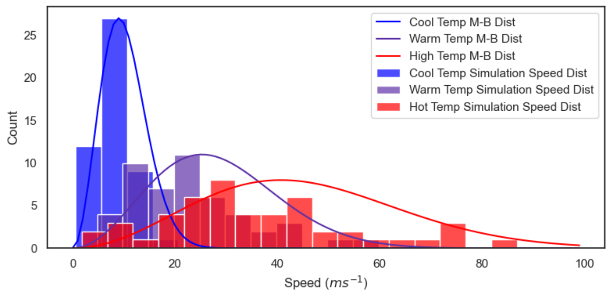


Fig. 4. Simulated speed distribution compared to Maxwell Boltzmann Distribution

Figure 4 shows simulated particle speed distributions at different temperatures compared to the theoretical distributions based off the Maxwell Boltzmann distribution. We see that our simulated data closely follows the theoretical data. This shows that particle dynamics within the system is accurate to accepted models of real gases. As the temperature increases, the spread of the distribution increases, and higher speeds are observed.

IV. Conclusion

A simulation of gas particles within a container was created. The simulated results were evaluated against theoretical models. The kinetic energy and momentum of the simulation were conserved as expected. Comparison with the ideal gas equation yielded significant discrepancies when the particle sizes became non-negligible. The pressure - particle size relationship was then compared to the Van der Waal's equation, which provided a better fit and like our data consisted of a positive correlation between the two.

The speed distributions from the simulation were evaluated against the Maxwell Boltzmann distribution and shown to closely agree.

Future investigation should aim to investigate pressure values for a greater range of particle size to see if the relationship follows Van der Waal's equation at greater radius values. Furthermore, a binary search tree could be introduced to cut down the simulation's time complexity even further.

V. References

[1] "*Project B: Thermodynamics Snookered*", Imperial College London, 2021

[2] VanderPlas, Jake. VanderPlas, Jake. *"Sorting Arrays | Python Data Science Handbook"*. Jakevdp.Github.Io, 2021, https://jakevdp.github.io/PythonDataScienceHandbook/02.08-sorting.html.

[3] "*Van Der Waals Equation Of State*". Hyperphysics.Phy-Astr.Gsu.Edu, 2021, http://hyperphysics.phyastr.gsu.edu/hbase/Kinetic/waal.html.

VI. Appendix

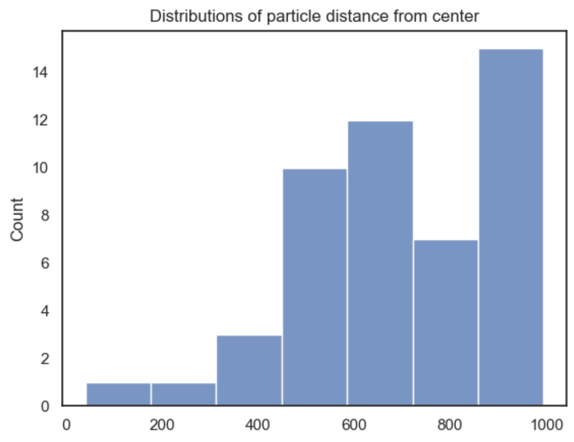


Fig. 6.1. Histogram of ball distance from container center (Container radius: 1000)

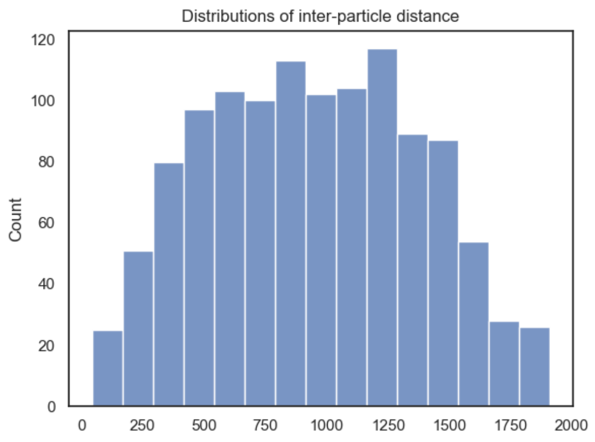


Fig. 6.2. Histogram of inter-ball separation (Container radius: 1000)

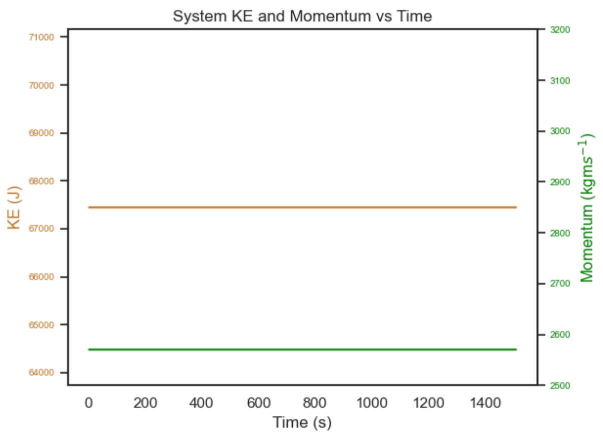


Fig 6.3 System KE and Momentum vs Time

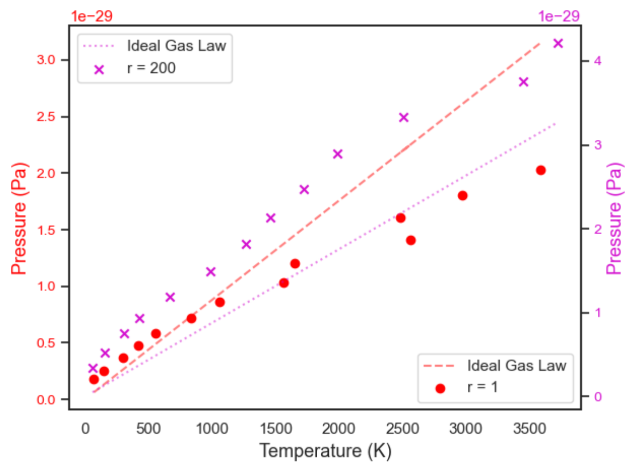


Fig 6.4 Pressure vs Temperature

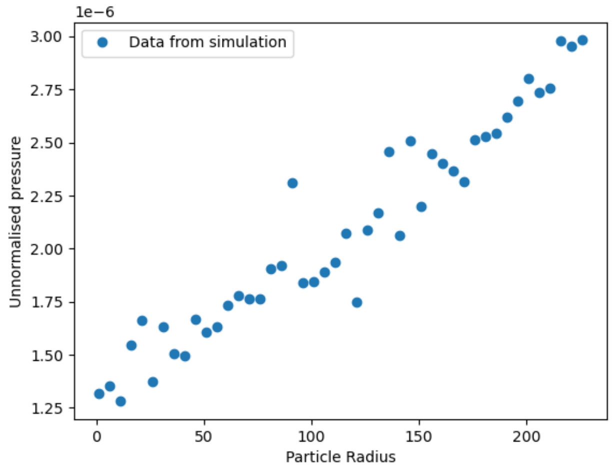


Fig 6.5 Pressure vs Particle radius

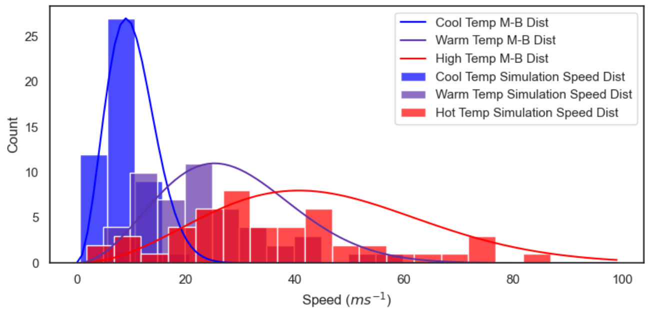
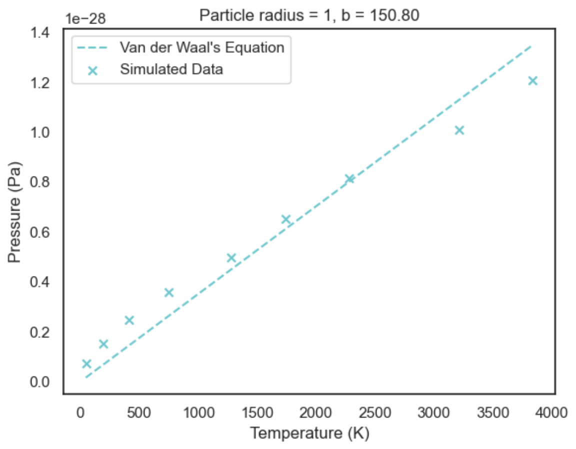
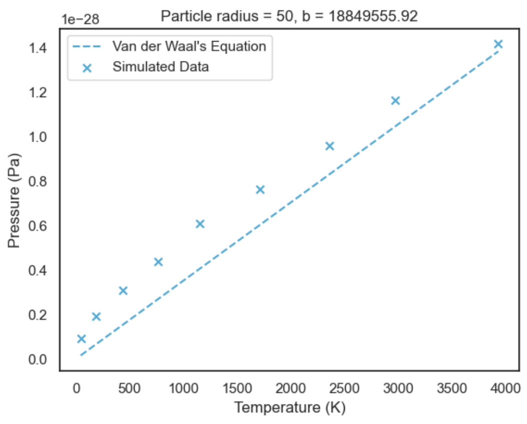


Fig 6.6 Histogram of ball speeds vs Maxwell Boltzmann Distribution (at 3 different temperatures)





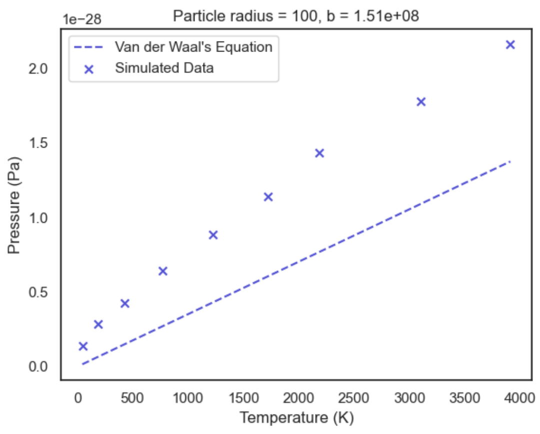


Fig 6.7-6.9 Simulated Data and Fitted Van der Waal's Law for particle radius 1,50 and 100.