Modelling an Ideal Gas

Tom Spencer

Tuesday 26$^{\text{th}}$ October, 2021

1 Introduction

We are interested in modelling a two dimensional ideal gas by modelling motions of individual particles that experience elastic collisions. We then hope to extract some interesting physics from such a simulation, such as showing speed distribution is a Maxwell-Boltzmann, computing the heat capacity and show it follows the ideal gas equation. We will first consider a model using non-rotating discs, though we may relax this later.

There are two approaches to such a simulation. A time driven approach is the most simplest, where we advance the simulation forward by a small time increment dt and then check for and perform any collisions that took place during dt. This approach is simple to understand and implement but has some drawbacks, namely:

- If we choose too large a time step, fast moving particles can skip over and avoid collisions with other particles

- This means a small step must be selected which is then computationally expensive, especially considering most particles aren't colliding

However, unlike many N-body simulation, each particle only experiences a force during a collisions, meaning they travel in straight lines in between collisions. This motivates the event-driven approach, where we have a variable time step, jumping from one collision to the next. It can be summarised as:

- Determine the next particle to experience a collision and the time t_1 it occurs, current time is t_0

- Advance all particles to the time t_1

- Perform the collision on the particle (e.g. flip perpendicular component of velocity)

- Store particle ID, time of collision, new position and velocity as an event

- Repeat

This approach has several advantages, we cannot have issues with fast moving particles skipping collisions and we don't have to perform many iterations where no collisions occurred where one larger jump is sufficient. It can also deal with simple forces acting on particles such that their trajectory between collisions is known beforehand, e.g. constant gravity. However it is more complex to implement and cannot deal with more complicated forces, e.g. Newtonian gravity.

We will implement the event-driven approach, first using an inefficient $\mathcal{O}(N^2)$ approach and then seek a more efficient solution.

2 Theory

2.1 Elastic collisions

In the following, a particle starts at position $\vec{r}_0$ and travels with velocity $\vec{v}$. Its position is therefore given by,

$$\vec{r}_p(t) = \vec{r}_0 + \vec{v}t. \tag{1}$$

2.1.1 Collisions with walls

A wall that starts at a point $\vec{A}$ and ends at $\vec{B}$ can be parameterised as,

$$\vec{r}_w(s) = \vec{A} + (\vec{B} - \vec{A})s, \tag{2}$$

with $0 \leq s \leq 1$. There are two ways a particle can collide with a wall. It can either collide with the ends at $\vec{A}$ or $\vec{B}$, or alternatively it can collide with main body of the wall, corresponding to $0 < s < 1$ in the above parametrisation. Initially, we only consider particles in a rectangular box. As such, particles cannot collide with the ends and so we initially only test for the second scenario.

When a particle of radius R collides with the wall, we note the perpendicular distance between the wall and the centre of the particle is precisely R. The collision takes place when,

$$(\vec{r}_p(t) - \vec{r}_w(s))^2 = R^2, \tag{3}$$

and occurs at,

$$s = \frac{(\vec{r}_p(t) - \vec{A}) \cdot (\vec{B} - \vec{A})}{|\vec{B} - \vec{A}|^2}. \tag{4}$$

Substituting in for s, we can rewrite Eq. 3 as,

$$(t\vec{\alpha} + \vec{\beta})^2 = R^2, \tag{5}$$

with,

$$\vec{\alpha} = \vec{v} - (\vec{v} \cdot \vec{\gamma})\vec{\gamma}, \qquad \vec{\beta} = \vec{\delta} - (\vec{\delta} \cdot \vec{\gamma})\vec{\gamma}, \qquad \vec{\gamma} = \frac{\vec{B} - \vec{A}}{|\vec{B} - \vec{A}|}, \qquad \vec{\delta} = \vec{r}_0 - \vec{A}. \tag{6}$$

This has solutions,

$$t = \frac{1}{\alpha^2}\left(-\vec{\alpha} \cdot \vec{\beta} \pm \sqrt{(\vec{\alpha} \cdot \vec{\beta})^2 + \alpha^2(R^2 - \beta^2)}\right). \tag{7}$$

We are only interested in collisions in the future and so are only interested in the negative root.

All collisions are currently considered to be elastic. When a particle hit a wall, the component of the particle's velocity perpendicular to the wall is simply negated. The particle's velocity therefore changes by,

$$\Delta\vec{v} = -2(\vec{v} - (\vec{v} \cdot \vec{\gamma})\vec{\gamma}), \tag{8}$$
$$= -2\vec{\alpha}. \tag{9}$$

We may need to be careful to ensure the magnitude of $\vec{v}$ is unchanged.

2.1.2 Collisions between discs

We consider collisions between two discs of masses m_1, m_2 and radii R_1, R_2 respectively. Initially we consider elastic collisions and suppose the discs do not rotate. Denoting the position of each particle as $\vec{r}_1(t)$ and $\vec{r}_2(t)$, a collision occurs when,

$$(\vec{r}_1(t) - \vec{r}_2(t))^2 = (R_1 + R_2)^2. \tag{10}$$

This has the same solution as the disc-wall collision case (Eq. 7) but with,

$$\vec{\alpha} = \vec{v}_1 - \vec{v}_2, \qquad\qquad \vec{\beta} = \vec{r}_{01} - \vec{r}_{02}, \qquad\qquad R = R_1 + R_2. \tag{11}$$

Here $\vec{v}_1$, $\vec{v}_2$ are the velocity and $\vec{r}_{01}$, $\vec{r}_{02}$ are the velocity and initial position of each respective disc. Again we are only interested in the negative solution and require $t > 0$ for a future collision.

To determine the result of a collision, we note the force acting on each particle is entirely parallel to $\vec{r}_1 - \vec{r}_2$ at the moment of collision. The component of the velocity of each particle perpendicular to $\vec{r}_1 - \vec{r}_2$ is therefore unchanged by the collision. Determining the components parallel to $\vec{r}_1 - \vec{r}_2$ is then reduced to a one dimensional problem.

In the one dimensional problem, particles have initial velocities u_1 and u_2. The centre of mass moves with velocity,

$$w = \frac{m_1 u_1 + m_2 u_2}{m_1 + m_2}. \tag{12}$$

Applying conservation of momentum and energy in the COM frame produces,

$$m_1 v_1 + m_2 v_2 = 0, \qquad m_1(v_1 - w)^2 + m_2(v_2 - w)^2 = m_1(u_1 - w)^2 + m_2(u_2 - w)^2, \tag{13}$$

where v_1 and v_2 are the final velocities. Solving these gives,

$$v_1 = \frac{(m_1 - m_2)u_1 + 2m_2 u_2}{m_1 + m_2}, \qquad\qquad v_2 = \frac{2m_1 u_1 + (m_2 - m_1)u_2}{m_1 + m_2}. \tag{14}$$

Alternatively, the change in velocity is,

$$\Delta v_1 = \frac{2m_2}{m_1 + m_2}(u_2 - u_1), \qquad\qquad \Delta v_2 = \frac{2m_1}{m_1 + m_2}(u_1 - u_2). \tag{15}$$

Returning to the two dimensional problem, we have the perpendicular velocity of each disc changes by Δv_1 and Δv_2 respectively. Each disc's velocity changes by,

$$\Delta\vec{v}_1 = \frac{2m_2}{m_1 + m_2} \frac{(\vec{u}_2 - \vec{u}_1) \cdot \Delta\vec{r}}{\Delta r^2}, \qquad\qquad \Delta\vec{v}_2 = -\frac{2m_1}{m_1 + m_2} \frac{(\vec{u}_2 - \vec{u}_1) \cdot \Delta\vec{r}}{\Delta r^2}. \tag{16}$$

$\Delta r = \vec{r}_2 - \vec{r}_1$ is a vector from the centre of disc 1 to disc 2 and has magnitude $R_1 + R_2$.

2.2 Insights from statistical physics

3 Program structure

Function for solving collision detection i.e. Eq. 7.

- A `Vec2D` class to represent a 2 dimensional vector.

- Include methods for magnitude, magnitude squared and dot product

- A `Disc` class to represent a disc

  - Stores position, velocity, mass and radius
  - If rotation is implemented, it will also need to include angular velocity and moment of inertia

- A `Wall` class to represent a straight wall a particle can bounce off

  - Stores start and end points, $\vec{A}$ and $\vec{B}$ of the wall
  - Stores a unit vector pointing from the start point to the wall's end point

- An `Event` class to represent either a disc-wall or disc-disc collision

  - Store index of the disc in the simulation's vector of discs, along with time of event, current position and new velocity

- A `Sim` class to represent a simulation

  - Vector to store the discs that are in the simulation, both in their current state and their initial state
  - Vector to store the walls in the simulation
  - Vector to store the events that occur during the simulation, if desired
  - Current time of the simulation
  - A method to advance the simulation by either a given number of events or duration, whichever is reached sooner, should have a member variable to set whether the events should be stored or only the current state should be stored

We also intend to produce a Cython wrapper to run and analyse the results from a simulation in Python. Initially, we'll implement a simple but inefficient algorithm for checking collisions but later we'll do something more advanced.

3.1 Initial algorithm

Initially we'll implement an inefficient $\mathcal{O}(N^2)$ algorithm as follows:

1. Store a list of the last event that occurred and any others that occurred at the same time. This could be due to a particle colliding with the corner of a box and colliding with two walls at the same time

2. Iterate over all possible disc-wall and disc-disc collisions. Find the one with the smallest positive time till collision.

3. Now we know the next collision, advance all particles (if necessary) to the time of collision

4. Perform the appropriate type of collision on the disc or discs

5. Repeat until the maximum number of collisions is reached or maximum duration is reached.

The above doesn't account for what to do if a disc undergoes two collisions at the same time, but it'll be enough to check the basic physics of checking for collisions and performing collisions are done correctly.