Homework 1

Kathryn Neugent

1. (a) My program is written in python. To run the program type:
> python nBody.py
Initial conditions (Kinetic energy, number of particles, box size, iteration steps, particle masses, and particle radii) can be set at the beginning of the program. Kinetic energy is measured in Joules, velocity is in meters/second, mass is in kilograms, length is in meters, and time is in seconds.
My program begins each particle with the same amount of KE which can be set by the user. It first calculates the starting velocity magnitude based off of the kinetic energy and mass of each particle. The directions of the velocities are then randomized by choosing a random value of the x velocity between 0 and the total velocity's magnitude. It is then multiplied randomly by either 1 or -1 to create negative velocities. Using the pythagorean theorem, my program then calculates the resulting velocity in the y direction. Overall, my program is not incredibly efficient. At every time step each particle is checked against every other particle. If there is no collision, the particles simply advance on based on their velocities and starting positions. If they collide with the walls they simply bounce back in a 1-d elastic collision. If they collide with one another I conserve momentum as well as kinetic energy. I was very confused on this part until I realized that there is one other set of equations I can use. I can additionally use the impulse due to the normal force in the x and y directions. The equations are as follows:

$$\Delta v \cdot \Delta r = (\Delta vx)(\Delta rx) + (\Delta vy)(\Delta ry)$$

$$J_x = \frac{J\Delta rx}{\sigma}$$

$$J_y = \frac{J\Delta ry}{\sigma}$$

$$J = \frac{2m_i m_j(\Delta v \cdot \Delta r)}{\sigma(m_i + m_j)}$$

$$vx_i' = vx_i + J_x/m_i$$

$$vy_i' = vy_i + J_y/m_i$$

$$vx_j' = vx_j + J_x/m_j$$

$$vy_j' = vy_j + J_y/m_j$$

I read about the concept of impulse online and eventually found this website with actual equations:
`https://algs4.cs.princeton.edu/61event/`

(b) Position and Velocity both Before and After

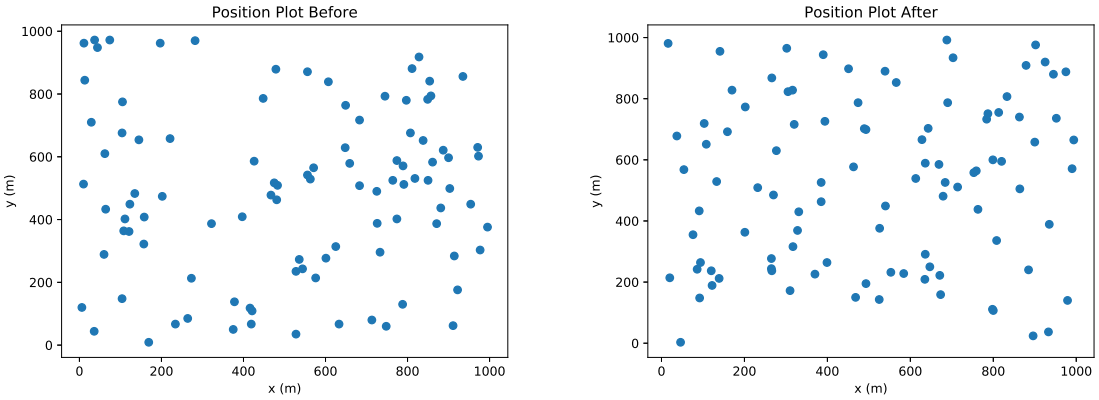

Figure 1: Position Before and After. Both appear to be randomly distributed.

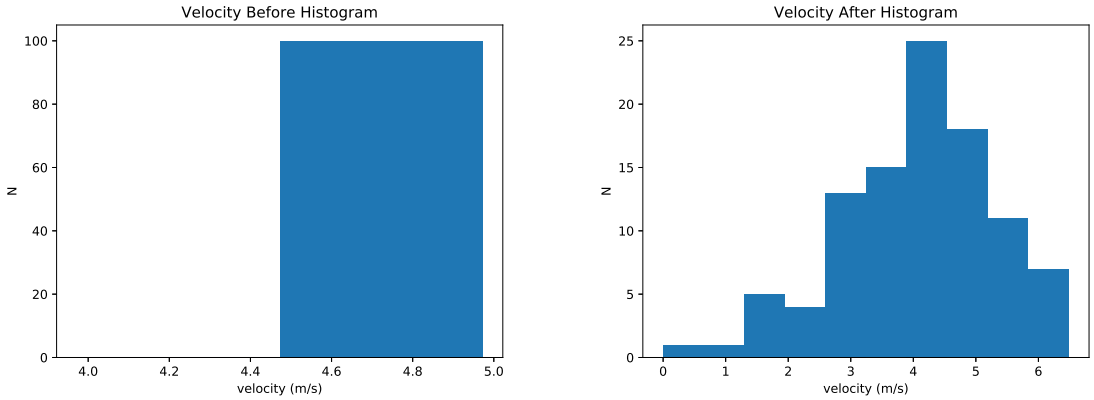

Figure 2: Velocities Before and After. Note that the velocity before is just one value. This makes sense since the KE and mass is equal for each particle. The velocity after is approaching a maxwellian.

2. (a) Derive the energy and velocity distribution for a Maxwellian.
Velocity:

$$f(q,p) \propto Ae^{-E/k_BT}$$
$$E = \frac{1}{2}mv^2$$
$$f(q,p) \propto Ae^{-mv^2/2k_BT}$$

Begin the normalization process ...

$$\int_{-\infty}^{\infty} Ae^{-x^2} dv = 1$$

$$\int_{-\infty}^{\infty} Ae^{-mv^2/2k_BT} dv = 1$$

$$x = \sqrt{\frac{m}{2kT}} v$$

$$dx = \sqrt{\frac{m}{2kT}} dv$$

$$dv = \sqrt{\frac{2kT}{m}} dx$$

$$\int_{-\infty}^{\infty} Ae^{-x^2} \sqrt{\frac{2kT}{m}} dx = 1$$

$$A\sqrt{\frac{2kT}{m}} \int_{-\infty}^{\infty} e^{-x^2} dx = 1$$

$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$$

$$A\sqrt{\frac{2kT}{m}} \sqrt{\pi} = 1$$

$$A = \sqrt{\frac{m}{2\pi kT}}$$

$$f(v_x, v_y) = \sqrt{\frac{m}{2\pi kT}} e^{-mv^2/2kT}$$

To put it into two dimensions, we can think about this as a probability that you would find a speed in each of two dimensions. So, $v = (v_x^2 + v_y^2)$. This brings out an extra "square" component and adds a v^2 leading to a final definition of:

$$f(v_x, v_y) = 2(v_x^2 + v_y^2) \frac{m}{kT} e^{-m(v_x^2+v_y^2)/2kT}$$

For the energy distribution, simply substitute $E = 1/2mv^2$.

$$f(E) = \frac{2E}{kT} e^{-E/kT}$$

(b) plot distribution over histogram

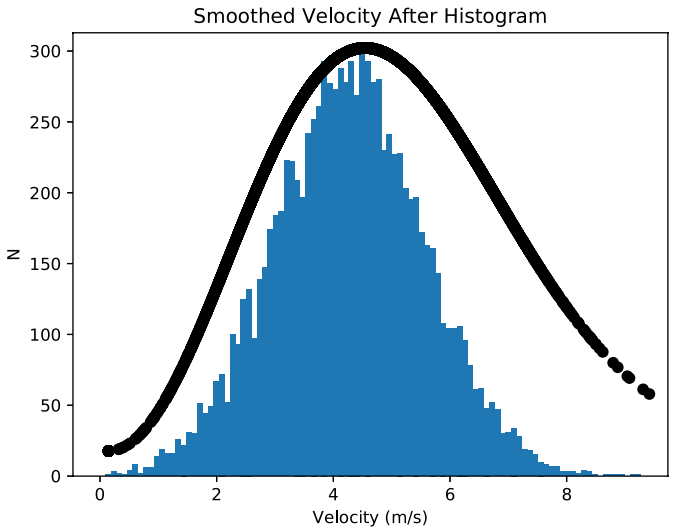

Figure 3: While it certainly isn't perfect, I believe this shows a Maxwellian distribution plotted above my velocity histogram. To get the Maxwellian to work and be on around the same scale, I needed to increase the temperature to 7×10^{24}K. This is a bit excessive.

(c) Make half of the masses $10\times$ larger.

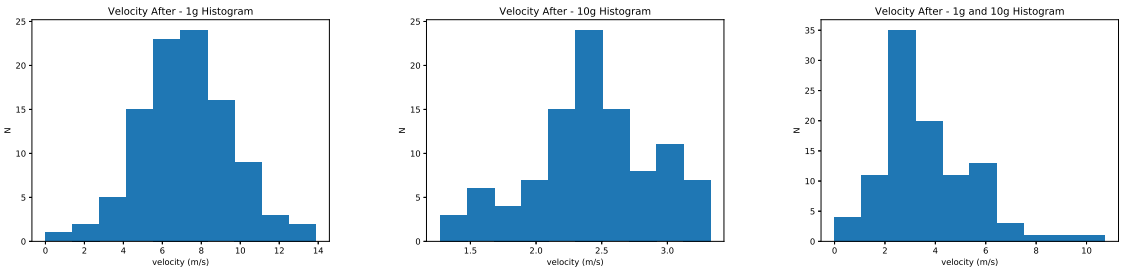

Figure 4: Velocities with 1 and 10 kg.

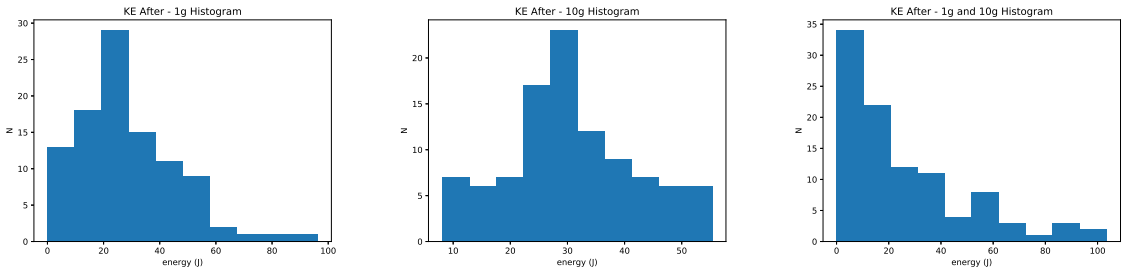

Figure 5: KE with 1 and 10 kg.

Neither the velocities or the KEs are Maxwellian after making half of the masses 10× larger (1 kg and 10 kg). Essentially there are two distributions added together, as is shown in the figures above. However, the KEs and velocities of the larger masses are more prominent making both the velocity and KE plot to be skewed to the right (lower velocities and KEs). This makes sense because the larger masses have more inertia. Once they're hit they won't speed up as much as the smaller masses. Thus, their average velocity will decrease in comparison to a simulation with a bunch of small masses

3. (a) To determine whether the system is "relaxed," I would treat it as a gaussian. I would then compare values in different "bins" on either side of the system's median value. So, if the median is at 0 m/s, the sum of the values in the 0-1 velocity range should be the same as the number of values in the -1-0 velocity range. By incrementing through these different bins (until they both have zero values in them) and testing whether the number of values is equal, it would be possible to determine if the system is maxwellian (or gaussian) or not.

   (b) The relaxation time is going to be inversely related to the initial velocity, cross section, and number density of the particles. For the initial velocity, we want the average of all of the initial velocities. This is going to be the same number when the KE and masses equal the same for all cases. Thus, an expression is:

$$t = \frac{1}{\sigma N \bar{v}}$$

   (c) I originally programmed what I outlined in Part A but I was unable to get it to work using code. Plotting the sum of the values vs. the time steps lead to a scatter plot – not what I had hoped for. I then decided to use an even simpler method. In a gaussian, the median value must equal the mean value. So, I simply determined the mean and median of each velocity at each time step, subtracted them and plotted the difference. The results were quite nice.

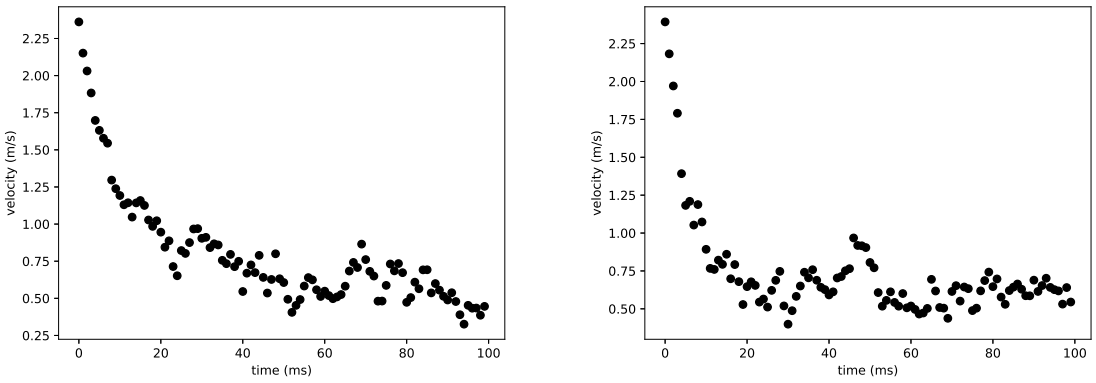

Figure 6: The left shows a simulation run with N = 100 and σ = 2m. The right shows a simulation run with N = 100 and σ = 5m. Notice that the behavior is as expected in that the relaxation time decreases as the cross section increases. They are inversely proportional.

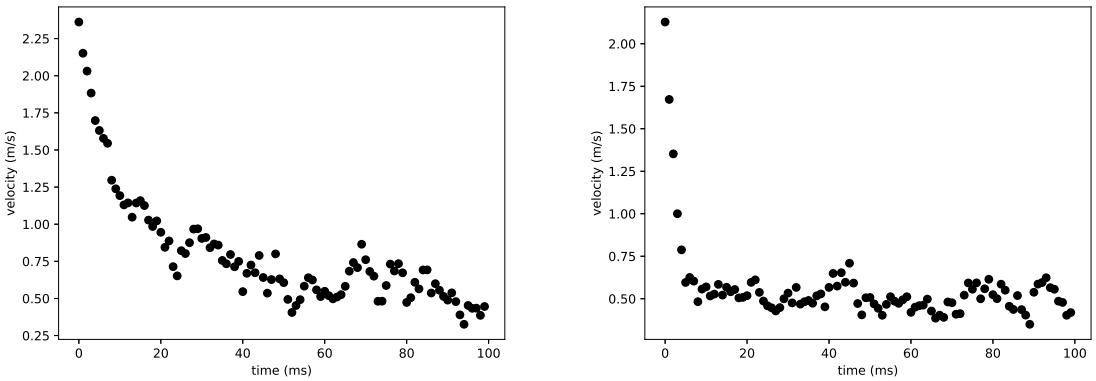

Figure 7: The left shows a simulation run with N = 100 and σ = 2m. The right shows a simulation run with N = 500 and σ = 2m. Notice that the behavior is again as expected in that the relaxation time decreases as the number density increases. They are inversely proportional.

(d) To determine the pressure, I used Equation 39.9 in Feynman. I watched each ball that hit each of the walls and saved their average masses, velocities, and number of balls. I then used the following equation to calculate the total average pressure on each wall throughout the simulation:

$$PV = N(2/3)mv^2/2$$

These were my results:

```
right N =  1828
right KE =  0.00027
right P =  0.34
```

```
left N =  2417
left KE =  0.00021
left P =  0.34

top N =  13329
top KE =  3.8-05
top P =  0.33
```

As you can see, the pressure is essentially equal for all walls ($P = nkT$). This continued when I varied the starting KEs and number of balls.