PARTICLE IN A BOX SIMULATION

DAVID FLEMING

1. Simulation Model

1.1. **Initial Conditions.** The particles in this code were modeled off of protons with the fundamental size and mass units being m_p = 1.6726219 x 10^{-24} g and a = 8.775 x 10^{-14} cm, respectively. Specifically, a is the charge radius which is a measure of the size of an atomic nucleus (see here for the NIST value). The initial velocity was computed by solving the following equation after specifying a temperature

$$\frac{3}{2}k_bT = \frac{1}{2}mv^2$$

This velocity was decomposed into components by computing $v_x = vcos(\theta)$ and $v_y = vsin(\theta)$ where θ was randomly sampled from $[0, 2\pi]$. The initial (x, y) were computed by randomly sampling from $[0, L]$ where L is the size of the box. As per the problem description, the size of the box, L, was set such that the particle size was equal to 0.05 of L. Finally, the time step Δt for each particle was set to $\frac{a}{3v}$ in order to ensure that not too many particles pass through each other on a given time step.

1.2. **Collision Handling.** Particles, assumed to be hard spheres, were treated as collided when the following condition was met

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} < 2a$$

where (x_i, y_i) is the center of the ith particle with radius a. When this condition held, the two particles trajectories were integrated in reverse by δt until just before they collide, that is when the distance between their centers of mass was $2a$. The time step δt is given by

$$\frac{2a - \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}{v_{rel}}$$

where v_{rel} is the relative velocity between particles i and j and (x_k, y_k) is is the center of the kth particle. Now the particles collide and their respective velocities are updated such that momentum and energy are conserved. To do this, the collision between them was modeled as an elastic forced directed along the line

Date: October 12, 2015.

connected the centers of the spheres. The new velocity for particle i after colliding with particle j is given by

$$\vec{v_i'} = \vec{v_i} - \frac{2m_j}{m_i + m_j} \frac{\langle \vec{v_i} - \vec{v_j}, \vec{x_i} - \vec{x_j} \rangle}{||\vec{x_i} - \vec{x_j}||^2} (\vec{x_i} - \vec{x_j})$$

See here for a description of the equation. After the collision, the particles were then advanced forward by δt to keep them on the same time as the rest of the particles.

1.3. **Box Boundaries.** In this simulation, the box walls were periodic. If a particle was found to be outside the box at a given time step, it would be placed on the opposition end of the box. For example, if $x > L$, x was set to $x - L$.

1.4. **Pressure.** The pressure the gas exerts on the walls was calculated each time a particle passed through the right wall. In two dimensions, pressure is given by force per length where $F = \frac{\Delta p}{\Delta t}$. The average force exerted on the right wall by a particle is $F = \frac{2mv_x}{\Delta t}$ where $\Delta t = \frac{l}{v_x}$ since for periodic boundary conditions, it takes the particle on average that much time to interact with the right wall again. To account for motion in both the x and y directions, note that $\bar{v_x^2} + \bar{v_y^2} = \bar{v^2}$ where $\bar{v_x^2} = \bar{v_y^2}$ so $\bar{v_x^2} = \frac{1}{2}\bar{v^2}$ Therefore, the pressure a particle exerts on the walls assuming a perfectly elastic collision is

$$P = \frac{m_p v^2}{l^2}$$

Each pressure event is recorded in a vector whose average is found and outputted at the end of the simulation. In general, the simulated pressure is within a factor of a few of $P = nkT$ which is a fine result given the periodic boundary conditions contributing a factor of 2, the low resolution, and the intrinsic scatter in the initial conditions.

1.5. **Algorithm.** The code, written in C++, was structured around two nested for loops. During a given time step, the first loop iterated over each particle. For each particle i, the then code looped over every other particle j to see if the two collided. If they collided according to the criterion given above, their velocities were updated while conserving energy and momentum. After the loop over every other particle ended, the code checked to see if the particle was still within the bounds of the box and applied periodic boundary conditions if it was not. At the end of each step for particle i, its position was updated by $\vec{x_i'} = \vec{x} + \Delta t \vec{v}$. Throughout the execution, the initial and final positions and velocities were stored for future output into a text file. The full code can be found at https://github.com/dflemin3/boxSim.

2. Simulation Velocity and Energy Distributions