Phys 607 - Computational Physics, Ideal Gas Particles in a 2D Box Simulation

Ben Maves, Haoyang Zhou

October 2024

We are interested in a simulation of non-interacting ideal gas particles in a confined 2D box with a user-specified force field that applies acceleration in one direction.

1 Classes

There are two classes in this simulation. This simulation always starts with initializing a system of particles. The system itself is a class, which contains many particles. Particle is also a class, where each particle is individually initialized when a system is built.

1.1 Particle Class

Particles are initialized with random mass, random initial x and y positions, and random initial velocities in both x and y directions. The class also records a list of its x and y position, and its energy at each iteration of the simulation, for visualization purposes. The velocities of the particle has a built-in maximum of one fifth of the box length, which means at maximum it takes a particle five iterations to go across one side to the other. This prevents particles going too fast, which improves visibility of animations made in the end.

1.2 System Class

A system initializes with several input parameters that can be set by user, which includes:

1. Number of Particles
2. Boundary Length
3. Force Field Strength (in +x direction)
4. Maximum Particle Mass

5. Number of Iterations

6. Time Step Size

The force field boundary is set parallel and centered around the x axis, simulating to a beam that comes from the left side of the box.

2 Iteration Algorithm

Particles have their velocities, and positions updated through their sets of differential equation. Particles preserve their constant velocities in both x and y direction under no external field influence. Their differential equations become $\frac{dv}{dt} = 0$ and $\frac{dr}{dt} = v$. Under external field, field strength is applied as acceleration vector in positive x direction. The box we are designing is not permeable on its boundary, which means particles cannot escape. The particles have their positions updated through scipy.solveivp() method. This method uses RK45 algorithm as default, and has local error on the order of fifth power of step size.

2.1 Boundary Check

Before solveivp() is called each time, our method under integration.py checks if the current particle is exceeding any boundary. Firstly, integration method checks x-boundary. Since our box is situated in the first quadrant with its left bottom corner at origin, having current x-value smaller than zero or greater than the dimension would count as going over the boundary. In the case of (x¡0), the particle would have its x-velocity and x-coordinate negated. In the case of (x¿x-dimension), x-velocity is negated, while x-coordinate would shift left of the boundary by the amount it exceeded. The same logic applies to y-velocity and y-coordinate. Integration method checks x and y separately since a particle could exceed x and y boundary at the same time.

3 Overall Trajectories

3.1 Expectation

Particles which do not interact with the beam or with the infinite potential wall should move linearly with constant velocities. When interacting with the wall, trajectories and velocities should reverse with conserved momentum. Particles interacting with the beam should experience an acceleration in the positive x-direction. Velocities and trajectories should follow the differential equations

$$\frac{d\vec{r}}{dt} = \vec{v}, \quad \frac{d\vec{v}}{dt} = \vec{B} \tag{1}$$

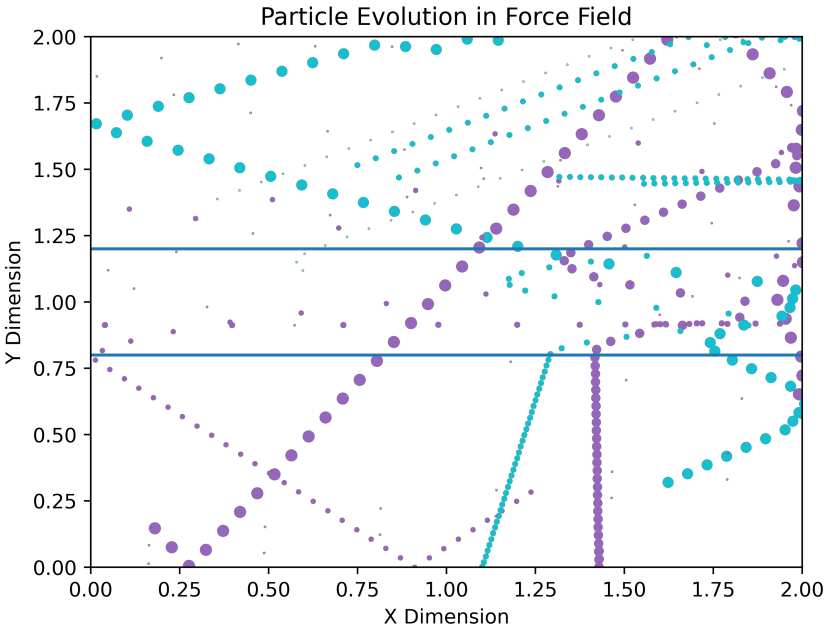

Figure 1: Trajectories for N particles in the box with beam turned on at strength 1. Inside beam particles experience a constant non-zero acceleration in the positive x-direction while outside there is no acceleration

3.2 Result

Particles follow expected trajectories inside and outside of the beam. We plot trajectories for N particles over many timesteps.

4 Energy Conservation

4.1 Expectation

Because our system is driven and non-dissipative we expect no conservation of total energy between time steps. The change in energy between steps can increase (particles interact with beam to increase speed), decrease (particles interact with beam to decrease speed) or remain the same (no average interactions with beam). Over a sufficiently large number of iterations though we expect the average energy of the system to increase as more beam interactions will lead to a higher probability of an increase in alignment of trajectories with the beam thus providing an increase in single particle energy.

Additionally, we expect total system average energy to increase linearly with particle number. Any individual randomized particle has a equal probability of

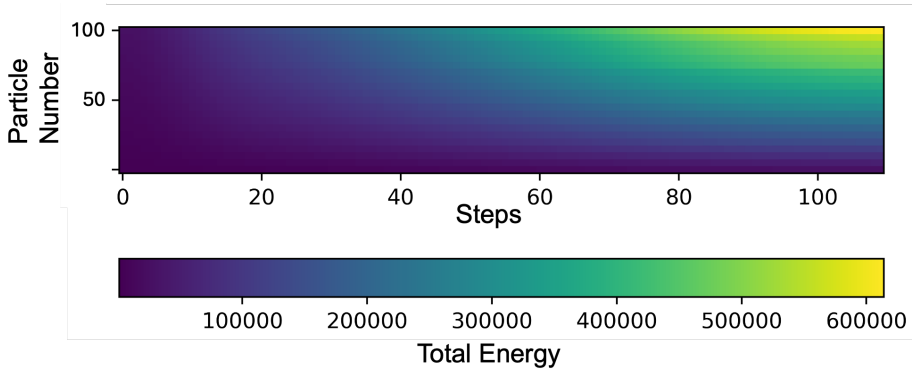

Figure 2: Total energy heat map for varying particle number and iterations

an interaction with the beam and thus with added particles there should be linearly added energy.

4.2 Result

After running $1,000$ simulations for various particle numbers, energy scales linearly with particle number and step count. Expectation value of energy at each time-step and particle is computed by taking the average of energies over N simulations. A histogram of simulation energies at an arbitrary time-point is shown in Figure 2. We also compute the energy fraction at each time-step defined as

$$E_{frac} = \frac{E_{step}}{E_0}. \tag{2}$$

This value is independent of particle number and increases linearly with step count.

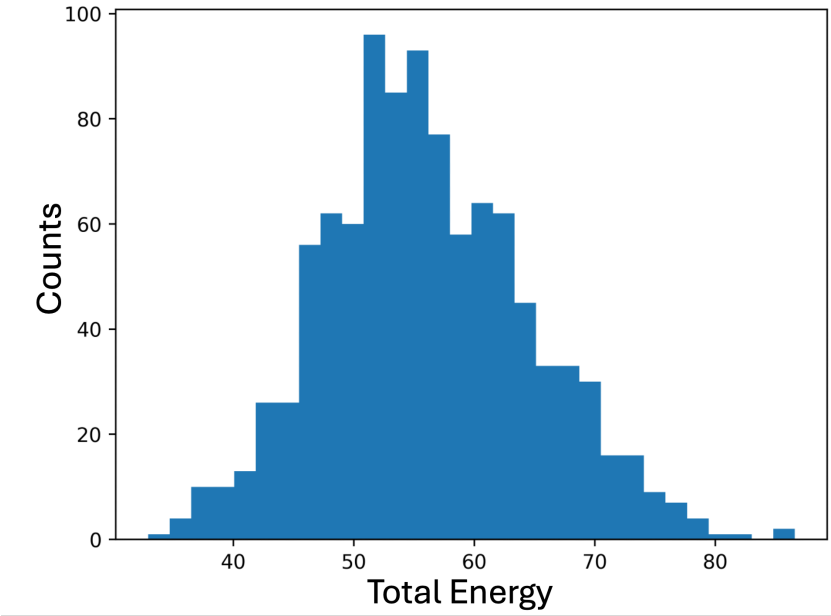

Figure 3: Histogram of total energies over 100 simulations at a given time-point

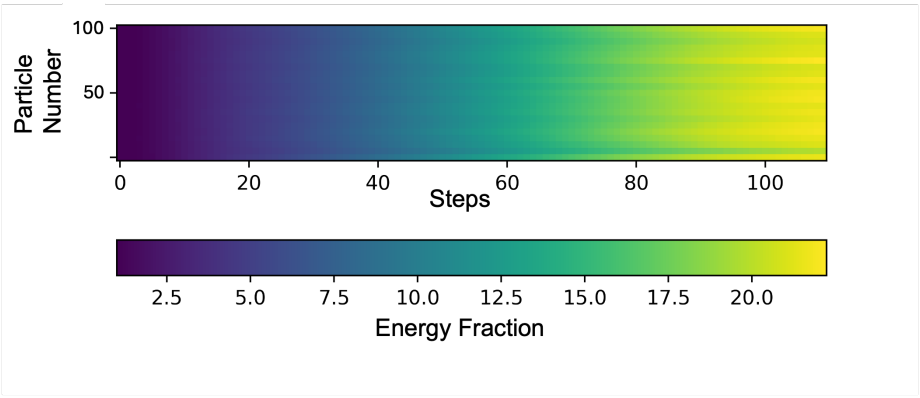

Figure 4: Heat map of energy as a fraction of initial energy. This fraction is constant for different particle numbers and increases linearly with step count.