Particles in a 2D box

Mathematical background

- $\frac{d\vec{x}}{dt} = \vec{v}$

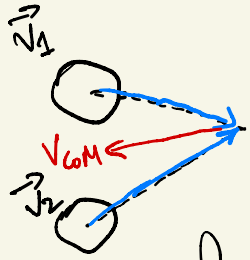
$$\int_{\vec{x}_{old}}^{\vec{x}_{new}} d\vec{x} = \int_{t_{old}}^{t_{old}+\Delta t} \vec{v}(t)\, dt = \vec{v} \int_{t_{old}}^{t_{old}+\Delta t} dt = \vec{v}\Delta t = \vec{x}_{new} - \vec{x}_{old}$$

- $\vec{x}_{new} = \vec{x}_{old} + \vec{v}\,\Delta t$

- There is no force other than collisions so $\vec{v}_{new} = \vec{v}_{old}$ unless a collision occurs

- $\vec{v}_{CoM} = \frac{d\vec{R}}{dt} = \frac{d}{dt}\left(\frac{m_1\vec{x}_1 + m_2\vec{x}_2}{m_1+m_2}\right) = \frac{m_1\vec{v}_1 + m_2\vec{v}_2}{m_1+m_2}$

- $\vec{v}_1 \to \vec{v}_1 - \vec{v}_{CoM} = \vec{v}_1'$
  $\vec{v}_2 \to \vec{v}_2 - \vec{v}_{CoM} = \vec{v}_2'$

- 

lab frame

before collision

In this frame the two bodies have exactly opposite momenta

zero momentum frame

before collision

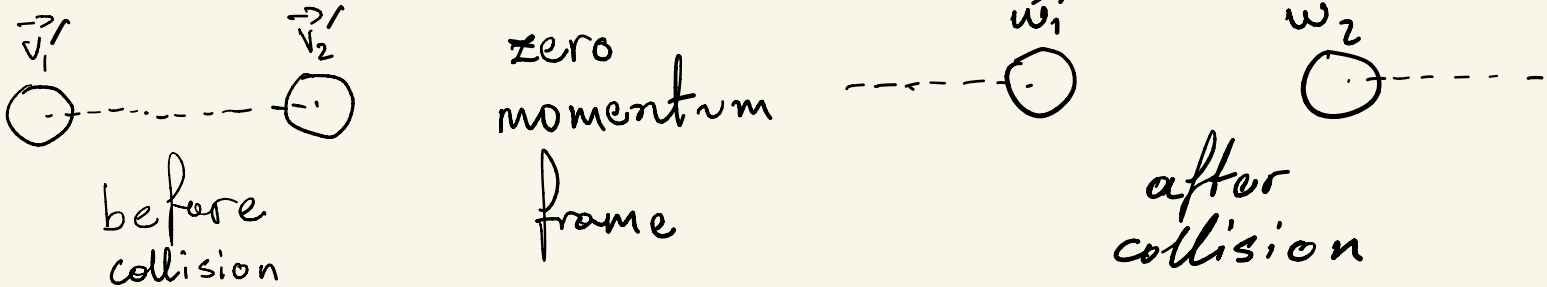
- $\vec{V}'_{CoM} = \dfrac{m_1\vec{V}'_1 + m_2\vec{V}'_2}{m_1+m_2} = \dfrac{m_1}{m_1+m_2}\left(V_1 - V_{CoM}\right) + \dfrac{m_2}{m_1+m_2}\left(V_2 - V_{CoM}\right)$

$= \dfrac{m_1}{m_1+m_2}V_1 + \dfrac{m_2}{m_1+m_2}V_2 - \left(\dfrac{m_1 V_{CoM}}{m_1+m_2} + \dfrac{m_2}{m_1+m_2}V_{CoM}\right)$

$= V_{CoM} - V_{CoM}\cancel{\left(\dfrac{m_1+m_2}{m_1+m_2}\right)^{1}} = \vec{0}$

↑ the zero vector

- $\vec{V}'_1$, $\vec{V}'_2$ } velocities in zero momentum frame $(m_1+m_2)\vec{V}'_{CoM} = \vec{P}'_{total} = \vec{0}$



- $\vec{0} = \vec{p}'_{before} = \vec{p}'_{after}$ (no net external forces $\Rightarrow$ conservation of momentum)

$m_1 \vec{v}_1' = -m_2 \vec{v}_2'$ $\quad m_1 \vec{w}_1' = -m_2 \vec{w}_2'$

$$\frac{d\vec{p}_{total}}{dt} = \frac{d\vec{p}_1}{dt} + \frac{d\vec{p}_2}{dt} = \vec{F}_{on\,1} + \vec{F}_{on\,2} = \vec{F}_{on\,1\,from\,2} + \vec{F}_{on\,2\,from\,1} + \sum \vec{F}_{on\,i\,from\,external}$$

cancel from Newton's 3rd law

assume no net external force

$$= 0 \Rightarrow \vec{p}_{total} = \text{constant with time}$$

- $\frac{1}{2} m_1 v_1'^2 + \frac{1}{2} m_2 v_2'^2 = \frac{1}{2} m_1 w_1'^2 + \frac{1}{2} m_2 w_2'^2$

$v_1'^2 = \vec{v_1'} \cdot \vec{v_1'}$ etc

$$\frac{1}{2} m_1 v_1'^2 + \frac{1}{2} m_2 \frac{m_1^2}{m_2^2} v_1'^2 = \frac{1}{2} m_1 w_1'^2 + \frac{1}{2} m_2 \frac{m_1^2}{m_2^2} w_1'^2$$

$$v_1'^2 \left(\frac{1}{2} m_1 + \frac{1}{2} \frac{m_1^2}{m_2} \right) = w_1'^2 \left(\frac{1}{2} m_1 + \frac{1}{2} \frac{m_1^2}{m_2} \right)$$

$$|\vec{v_1'}| = |\vec{w_1'}|$$

$$|\vec{v_2'}| = |\vec{w_2'}|$$

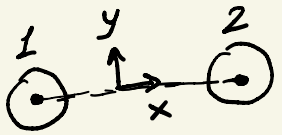
- In zero momentum frame the direction of velocity just gets <u>reversed</u> with the magnitude staying the same for each of the two velocities

↓

no impulse perpendicular to line of motion

$$\vec{w}_1' = -\vec{v}_1'$$

$$\vec{w}_2' = -\vec{v}_2'$$



$$\Delta p_{1y} = f_y \Delta t = 0 \cdot \Delta t = 0$$

- In lab frame

$$\vec{w}_1 = \vec{w}_1' + \vec{v}_{CoM} = -\vec{v}_1' + \vec{v}_{CoM} = -(\vec{v}_1 - \vec{v}_{CoM}) + \vec{v}_{CoM}$$

$$\vec{w}_2 = \vec{w}_2' + \vec{v}_{CoM} = -\vec{v}_2' + \vec{v}_{CoM} = -(\vec{v}_2 - \vec{v}_{CoM}) + \vec{v}_{CoM}$$

- Final velocities

$$\vec{w}_1 = -\vec{v}_1 + 2\vec{V}_{CoM}$$

$$\vec{w}_2 = -\vec{v}_2 + 2\vec{V}_{CoM}$$

with $$\vec{V}_{CoM} = \frac{m_1\vec{v}_1 + m_2\vec{v}_2}{m_1 + m_2}$$

Code layout

1) class Particles

attributes:

r, m, pos, vel

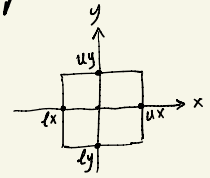
↑ radius ↑ mass ↑ np.array([x, y]) ↑ np.array([vx, vy])

2) class Box

attributes

lx, ly, ux, uy, N, particles_list

↑ define limits of Box ↑ Number of particles ↑ elements will be of type Particles



eg: lx = ly = -1

ux = uy = 1

eg:

```
for _ in range(N):
    self.particles_list.append(Particle(r, m, rdm.uniform(lx+eps, ux-eps), rdm.uniform(ly+eps, uy-eps), rdm.uniform(-speed, speed), rdm.uniform(-speed, speed)))
```

↑ Note: from numpy import random as rdm

3) Define variables

global variables:
- dx, dy, ux, uy, N
- iterations, dt, r, m
- box

eg: 5000 (iterations)

eg: 0.001 (dt)

have the same for all particles for simplicity

box of type Box

```
position_data = [[] for i in range(box.N)]
```

list of lists to hold the positions for each particle initialized to empty

So at the end of the simulation position_data will look like this for N=2

```
[[pos_0, pos_1, ...  ]
 [pos_0, pos_1, ...  ]]
```

one list for each particle with time increasing to the right, the pos_t is the position vector at time t from the attributes of each particle object

4) check_collision(idx)

```
particles_list = box.particles_list
p = particles_list[idx]
```

- If the particle has collided with the vertical walls reverse the x component of velocity
- Do the same as above for horizontal walls and y velocity
- For each other particle in particles_list if it has collided with particle p change their velocities

$$\vec{w}_1 = -\vec{v}_1 + 2\vec{v}_{CoM} \quad \text{with} \quad \vec{v}_{CoM} = \frac{m_1\vec{v}_1 + m_2\vec{v}_2}{m_1 + m_2}$$
$$\vec{w}_2 = -\vec{v}_2 + 2\vec{v}_{CoM}$$

5) simulate() function

```
particles_list = box.particles_list
for t in range(iterations):
    for i in range(len(particles_list)):
```

- check for collisions for particle i
- append the position vector for particle i to the position_data[i] list
- update position vector of particle i with

$$\vec{x}_{new} = \vec{x}_{old} + \vec{v}\,\Delta t$$

Remember: → $\vec{v}$ for particle i can be found with particles_list[i].vel

6) Call the function simulate()

At this stage we have finished with the physics and data collection and just need to do the animation

7) Create figure and axes objects:

```
fig = plt.figure()
ax = plt.axes()
```

8) Remove numbers from axes:

```
plt.xticks([])
plt.yticks([])
```

9) Create two variables x and y to hold the x and y positions of all particles at time t=0 using position_data

10) Initialize a scatter object

```
scatter_plot = ax.scatter(x, y)
```

11) animate(i) function

This function will return an updated scatter object

12) Call FuncAnimation and plt.show()

Note: from matplotlib.animation import FuncAnimation

For steps 11 and 12:

```
def animate(i): # function that updates the scatter plot using the positions of the next time step

    # this function is called by FuncAnimation below and each time its called the index i in the input is automatically incremented by 1

    if i > iterations-1: # if there is no more data to plot stop the whole file and exit
        quit()

    data = [position_data[j][i+1] for j in range(box.N)] # i starts from 0 by FuncAnimation and since we already plotted above the time 0 we
    scatter_plot.set_offsets(data) # update the scatter plot
    return scatter_plot

animation = FuncAnimation(fig, animate, interval = 0.0000000001) # function that makes the animation, interval is the microseconds between fr
plt.show() # we call this to open the window of the plot
# animation.save('boxed_particles.gif', writer = 'pillow') # a way to save the animation as a gif
```