-Rajeshwari D

Labels:
Particles are labeled from 0 to N-1
Label of left wall = -1
Label of right wall = -2
Label of bottom wall = -3
Label of top wall = -4

Structures defined:
Particle [x position, y position, x velocity y velocity, last updated time, no of wall collisions, no of particle collisions, label of particle]
Collision[Particles involved in collision, and time of collision]

Function wall_collision_time:
Inputs: [particle, dimensions of box, radius of particle]
Calculates the time required to collide with the nearest wall
If x_velocity is -ve collision with right wall is set to a large value, if its +ve collision with left wall is set to infinity
Similarly, collision time for bottom and top wall is changed according to the sign of y_velocity.
Return type is a particle, which contains the label as the wall label and last updated time as collision time required with nearest wall

Function particle_collision_time:
Inputs: [particles involved in collision, radius of particles]
Checks the last updated time of each particle, takes the highest of the two. Brings both particle to the same time.
Checks discriminant(D) of the equation to calculate time
Case 1:D < 0 sets the collision time to INF
Case 2: D=0 updates the collision time if the root is positive
Case 3: D>0 updates the collision time to least positive root
Return : time after the particle particle collision

Function update:
Inputs: [particles array, collision, no of particles]
Checks the labels of particles involved in collision
Update the positions of particle with those labels
If label 2 is negative, {(which means it is wall particle collision) increment the wall collision count
Change the sign of x velocity if label is -1 or -2(left or right wall collision)
Change the sign of y velocity if label is -3 or -4(bottom or top wall collision)}
If label 2 is positive, (which means it is particle particle collision) {increment the particle collision count
Interchange the velocity of the particles.}

Input: no of particles, radius, dimension of box, position velocity of particle

Calculate all possible collisions particle-particle, wall-particle and append the the array collisions

Process the collision with lowest time.
Check the condition for the particles involved in the previous collision are not same as the present

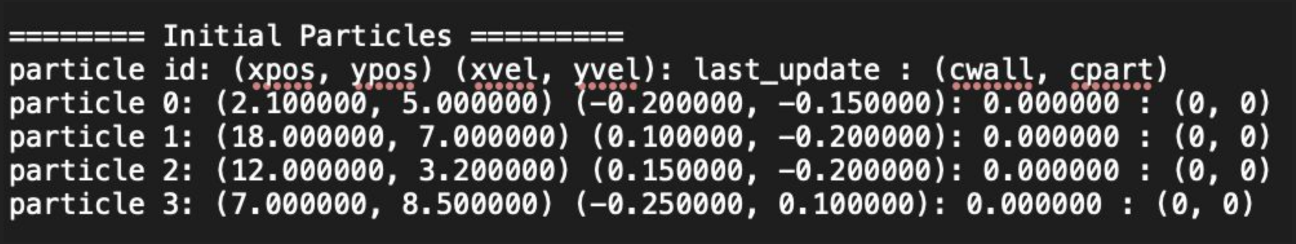
Update the position and time of the particles involved in the collision

Recalculate the collision time of each particle after processing each collision

If the next collision time is greater than the end time, stop the process.Bring all the particles to the endtime positions.

Wall-particle collision

Test case 1

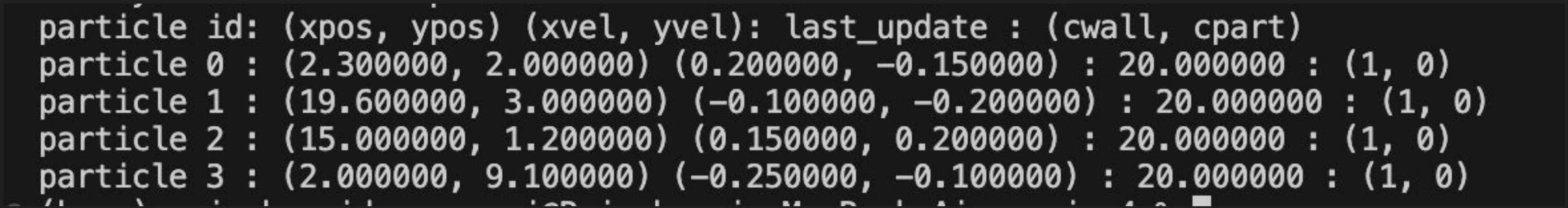


```
========= Initial Particles =========
particle id: (xpos, ypos) (xvel, yvel): last_update : (cwall, cpart)
particle 0: (2.100000, 5.000000) (-0.200000, -0.150000): 0.000000 : (0, 0)
particle 1: (18.000000, 7.000000) (0.100000, -0.200000): 0.000000 : (0, 0)
particle 2: (12.000000, 3.200000) (0.150000, -0.200000): 0.000000 : (0, 0)
particle 3: (7.000000, 8.500000) (-0.250000, 0.100000): 0.000000 : (0, 0)
```

I can see that when the particle is close to right wall and velocity is positive, it collides with right wall, and x velocity of the particle changes the sign and position of a nearest particle is updated in each step.

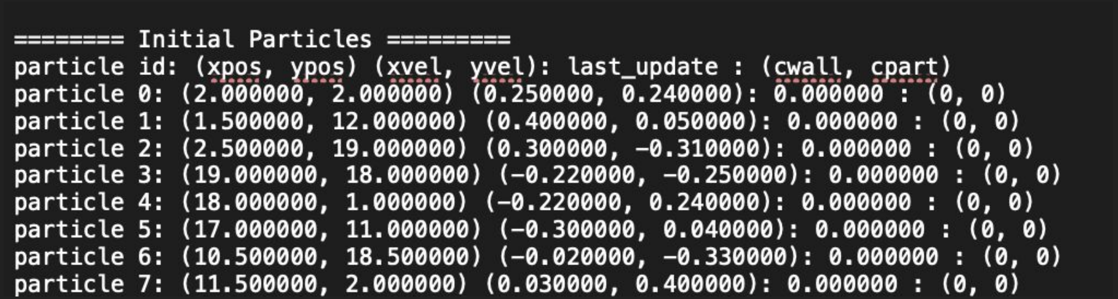
Every step the time of collision chosen by my code is same as the provided output.

The final positions of the particles given by my code matches with outputs provided



```
particle id: (xpos, ypos) (xvel, yvel): last_update : (cwall, cpart)
particle 0 : (2.300000, 2.000000) (0.200000, -0.150000) : 20.000000 : (1, 0)
particle 1 : (19.600000, 3.000000) (-0.100000, -0.200000) : 20.000000 : (1, 0)
particle 2 : (15.000000, 1.200000) (0.150000, 0.200000) : 20.000000 : (1, 0)
particle 3 : (2.000000, 9.100000) (-0.250000, -0.100000) : 20.000000 : (1, 0)
```

Particle- particle collision

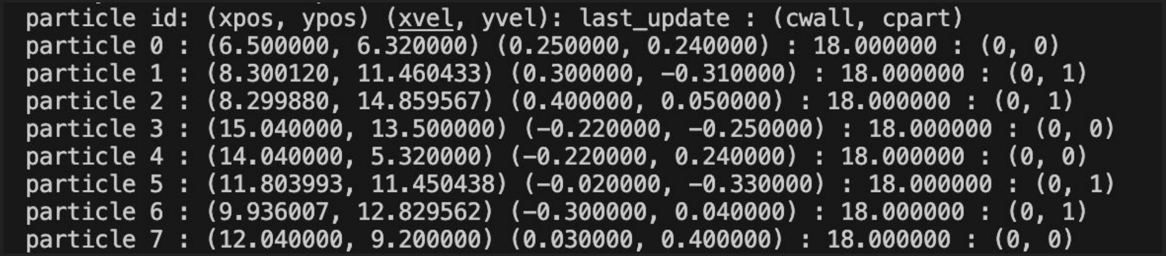


```
======== Initial Particles =========
particle id: (xpos, ypos) (xvel, yvel): last_update : (cwall, cpart)
particle 0: (2.000000, 2.000000) (0.250000, 0.240000): 0.000000 : (0, 0)
particle 1: (1.500000, 12.000000) (0.400000, 0.050000): 0.000000 : (0, 0)
particle 2: (2.500000, 19.000000) (0.300000, -0.310000): 0.000000 : (0, 0)
particle 3: (19.000000, 18.000000) (-0.220000, -0.250000): 0.000000 : (0, 0)
particle 4: (18.000000, 1.000000) (-0.220000, 0.240000): 0.000000 : (0, 0)
particle 5: (17.000000, 11.000000) (-0.300000, 0.040000): 0.000000 : (0, 0)
particle 6: (10.500000, 18.500000) (-0.020000, -0.330000): 0.000000 : (0, 0)
particle 7: (11.500000, 2.000000) (0.030000, 0.400000): 0.000000 : (0, 0)
```

At every step, the updated positions and velocity of the particles,match with the stepbystep outputs provided.

When two particles collide, the velocity of the particles are exchanged and the particle collision count is updated correctly.

The final positions and the velocity of the particles after end of the given time match with the given output.



```
particle id: (xpos, ypos) (xvel, yvel): last_update : (cwall, cpart)
particle 0 : (6.500000, 6.320000) (0.250000, 0.240000) : 18.000000 : (0, 0)
particle 1 : (8.300120, 11.460433) (0.300000, -0.310000) : 18.000000 : (0, 1)
particle 2 : (8.299880, 14.859567) (0.400000, 0.050000) : 18.000000 : (0, 1)
particle 3 : (15.040000, 13.500000) (-0.220000, -0.250000) : 18.000000 : (0, 0)
particle 4 : (14.040000, 5.320000) (-0.220000, 0.240000) : 18.000000 : (0, 0)
particle 5 : (11.803993, 11.450438) (-0.020000, -0.330000) : 18.000000 : (0, 1)
particle 6 : (9.936007, 12.829562) (-0.300000, 0.040000) : 18.000000 : (0, 1)
particle 7 : (12.040000, 9.200000) (0.030000, 0.400000) : 18.000000 : (0, 0)
```