# Particle Simulation

## 1 Introduction

In this project we will do a particle simulation and verify the gas law $pV = nRT$ and show something called Brownian motion (*bonus project*). The particles are hard with a radius 1 and all collisions will be regarded as perfectly elastic (with the walls and other particles) and no friction is present in the box. The box will be a 2 dimensional rectangle (the collisions will be easier to handle).

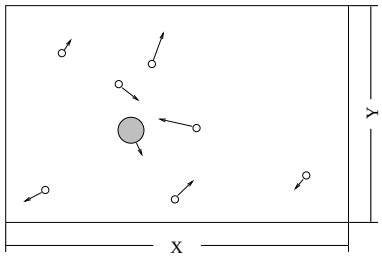


Figure 1: Gas simulation by rigid bodies.

Until a collision occur the particles will travel straight (no external forces) and if a collision occur the momentum and energy is conserved, by the elastic collision. From the following relationships the velocity after the collision can be found, $m_1, m_2$ is the mass of the particle and $\hat{v}_{(x,y)}$ is the velocity before the collision and $v_{(x,y)}$ after, see Figure 2. The law of conservation of the momentum is, after a suitable rotation of the coordinate system,

$$m_1 v_{1,x} + m_2 v_{2,x} = m_1 \hat{v}_{1,x} + m_2 \hat{v}_{2,x}$$

and the kinetic energy

$$m_1(v_{1,x}^2 + v_{1,y}^2) + m_2(v_{2,x}^2 + v_{2,y}^2) = m_1(\hat{v}_{1,x}^2 + v_{1,y}^2) + m_2(\hat{v}_{2,x}^2 + v_{2,y}^2).$$

The coordinate system is rotated so the tangent to the collision point is vertical so there is no change in the velocity in the y-direction.

When a particle hits wall the particle will bounce back with negative velocity normal to the surface.

With this simulation one can simulate the notion of pressure, the bouncing particles will exhibit a pressure on the walls each time they hit. Each time a particle hits a wall a momentum of $2mv_{x,y}$ will be absorbed by the wall. If we sum all collisions by a wall during $t$ second and divide this by the circumference of the box and $t$, we will obtain the (two dimensional) pressure in the box. You will use this pressure to verify the pressure law $pV = nRT$, $p$ pressure, $V$ volume

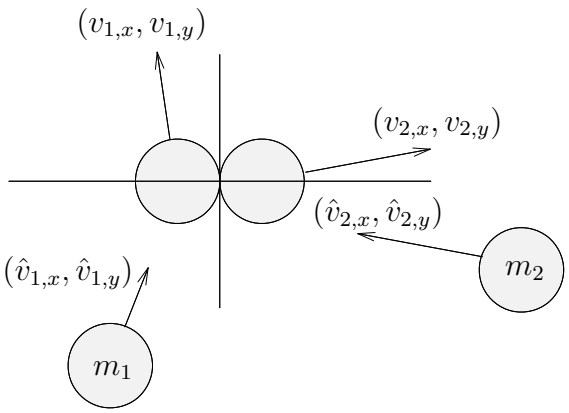


Figure 2: Interaction between two particles

(here area), $n$ number of moles (number of particles), $R$ magic constant and $T$ is the temperature, in our case the volume will be instead area.

To show the Brownian motion a particle with a greater mass and radius will be inserted in the box. The trajectory of this particle will be monitored and plotted after the simulation.

# 2 Implementation

Write the framework for the simulation of small particles in a rectangular box. Choose and motivate a good distribution of the particles between the processors, implement the communication between the processors using MPI, count the pressure and keeping track of the big particle (*bonus project, see Questions below*).

## 2.1 Datatypes

The particles is in the provided functions is represented by the following struct

```
struct part_cord {float x; float y; float vx; float vy;}
typedef struct cord cord_t;
```

$x$, $y$ is the position, $v_x$, $v_y$ the velocity. The walls can be represented by

```
struct cord {float x0; float x1; float y0; float y1;}
typedef struct cord cord_t;
```

These datatypes is defined in the file `coordinate.h`. The particles can be stored

on each processor in a fix array and the larger particle will have to be handled separate.

## 2.2 Functions provided

The interaction between the small particles is provided by the following routines in the file `physics.c`;

```
float collide (pcord_t *p1,pcord_t *p2)
interact (pcord_t *p1,pcord_t *p2,float t)
float wall_collide (pcord_t *p, cord_t wall)
feuler (pcord_t *a, float time)
```

The routine `collide` returns $-1$ if there will be no collision this time step, otherwise it will return when the collision occurs. This will then be used as one of input parameter to the routine `interact`. The routine `interact` moves two particles involved in the collision. Do **not** move these particles again. `wall_collide` checks if a particle has exceeded the boundary and returns a momentum. Use this momentum to calculate the pressure. The routine `feuler` moves a particle.

The following modified routines can be used for the interaction of the big particle in the extra assignment;

```
float collideBig (pcord_t *Big,pcord_t *p, float rBig)
interactBig (pcord_t *Big,pcord_t *p, float t, float massBig)
wall_collide_Big (pcord_t *p, cord_t wall, float rBig)
```

The big particle can interact with several small particles. When a particle hits the big one, the small particle is moved by `interactBig`, not the big particle. After all particles has been checked against the big particle, move the big particle and check for collisions between the remaining unmoved small particles.

## 2.3 Important implementation issues

The files necessary for the implementation can be found at CSCI-455's course web site: `http://cse.stfx.ca/~ltyang/csci-455/`.
Some simplifications to the model can be implemented. If the particles is small compared to the box and the time step is short, the possibility that a particle will collide with more than one other particle is statisticly very small, so if a particle hits another, we can update both and ignore them until the next time step (this is done in the procedure `interact`). Observe that this can be implemented without some sort of update flag array!. Depending on how the particles is distributed over the processors some simplifications of the communication can

be done, *motivate each simplification done!*

- Each time-step must be 1 time unit long.
- The initial velocity should be less then 50. Use the random number generator to generate the absolute velocity and a starting angle. ($r$=rand()·max_vel; $\theta$=rand()·$2\pi$; $v_x = r\cos(\theta)$; $v_y = r\sin(\theta)$)
- Typical numbers for the simulation; number of particles = 10000 × number of processors, area of the box = $10^4 \cdot 10^4$, radius of the big particle = 100 and mass of the big particle = 1000.
- The pressure can be found at the end of the simulation by dividing the total momentum from the routine `wall_collide` with the number of time-steps and the length of the circumference of the box.
- Observe that there is no need to keep the particles in any order.
- Avoid unnecessary communication by sending all particles at once.

# 3 An Example of Program Structure

- Initiate particles
- Main loop: for each time-step do
  - for all particles do
    - Check for collisions.
    - Move particles that has not collided with another.
    - Check for wall interaction and add the momentum.
  - Communicate if needed.
- Calculate pressure.

# 4 Questions

You should measure and report the following interesting quantities;

1. Explain your choice of distribution of the particles over the processors. Is there an optimal relation between the distribution and the geometry of the domain? Measure this by counting particles passed between processors in each time step.

2. What is the speedup with 1, 2, 4, 16, 32,... processors.

3. What is the scaled speedup (let the number of particles grow proportional to the number of processors).

4. Verify the gas law $pV = nRT$ by changing the number of particles ($n$) and size of the box ($V$) and then measure the pressure.

5. *Bonus Project:* Implement a big and heavy particle and plot the trajectory of it. This kind of motion is called Brownian motion and was found by a biologist studying the motion of pollen on the water surface in a bucket. This is also a example of a random walk and fractal by nature.