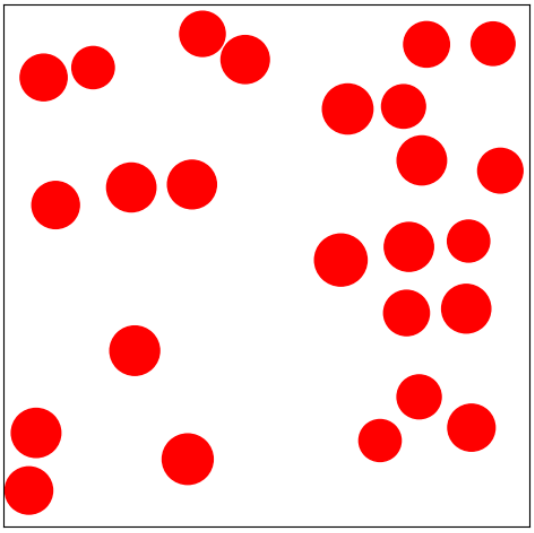
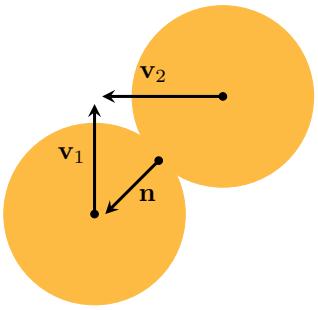
LMECA2300: Homework 1
An event-driven method for a granular gas

In this first homework, we are going to model a two-dimensional granular gas in a square box with the help of an event-driven method.
Granular gases correspond to an agitated state of granular materials. They mainly differ from classical gases by the fact that the interactions between the grains are dissipative, while the interactions between molecules are usually elastic. Because of this, the gaseous state of granular materials is unstable, and requires a constant supply of energy to be sustained, for example in the form of vibrations. These dissipative properties can be of great interest in damping devices, as they are not dependent on temperature and suffer less from aging compared to classical viscous fluid dampers.



As the event-driven method is a Lagrangian method, the positions $\mathbf{x}$ and velocities $\mathbf{v}$ of the grains are tracked in time and the contacts between the grains are solved. As explained in the lecture, the time step is not fixed, but changes during the simulation. The basic principle of the method is to find the minimum time to a contact t_c given a set of grains and walls. The simulation then jumps to the moment of the contact, updating the positions of the grains. The corresponding contact is solved based on the conservation of momentum and on a coefficient of restitution r describing the amount of dissipated kinetic energy. The velocities of the grains implied in the contact are then updated and the simulation proceeds as before.

Contact between two frictionless grains



The contact between two grains can be solved by applying the principle of conservation of momentum and by using the coefficient of restitution to relate the velocities of the grains before and after the contact.
As the grains are considered frictionless here, the two-dimensional problem can be reduced to a one dimensional problem in the normal direction. A normal vector associated to the contact can be defined as $\mathbf{n} = (\mathbf{x}_1 - \mathbf{x}_2)/|\mathbf{x}_1 - \mathbf{x}_2|$. The equations for the contact in the normal direction are as follows:

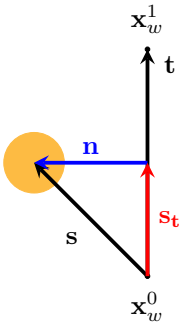
$$(m_1\mathbf{v}_1 + m_2\mathbf{v}_2) \cdot \mathbf{n} = (m_1\mathbf{v_1}' + m_2\mathbf{v}_2') \cdot \mathbf{n}$$
$$(\mathbf{v}_1' - \mathbf{v}_2') \cdot \mathbf{n} = -r(\mathbf{v}_1 - \mathbf{v}_2) \cdot \mathbf{n}.$$

Solving these equations, the velocities after the contact $\mathbf{v}_1'$ and $\mathbf{v}_2'$ can be expressed as:

$$\mathbf{v}_1' = \mathbf{v}_1 - \Delta v_n \mathbf{n}$$
$$\mathbf{v}_2' = \mathbf{v}_2 + \Delta v_n \mathbf{n},$$

with Δv_n the normal velocity correction due to the contact.

Contact with a boundary wall



In a two-dimensional configuration, a wall can be represented as a line segment, defined by two points $\mathbf{x}_w^0$ and $\mathbf{x}_w^1$. In order to compute the time of a contact between a grain and the wall and solve it, it is necessary to compute a normal vector $\mathbf{n}$ to the wall. To do so, the vectors $\mathbf{t} = \mathbf{x}_w^1 - \mathbf{x}_w^0$ and $\mathbf{s} = \mathbf{x}_g - \mathbf{x}_w^0$ are first constructed. The projection of $\mathbf{s}$ on $\mathbf{t}$ is then computed as $\mathbf{s}_t = (\mathbf{s}\cdot\mathbf{t})\mathbf{t}/|\mathbf{t}|$. Finally, the normal vector is obtained as $\mathbf{n} = \mathbf{s} - \mathbf{s}_t$. The norm $n = |\mathbf{n}|$ of $\mathbf{n}$ gives the distance between the grain and the wall. The contact time t_c between the wall and the grain, if it exists, is the solution of the following equation:

$$n^2 + (\mathbf{v}_g \cdot \mathbf{n})t_c = nR_g,$$

with $\mathbf{v}_g$ the velocity of the grain and R_g its radius.

The resolution of the contact is similar to the case between two grains, except that the wall is considered to have an infinite mass.

A regular data writing

The fact that the time step is variable makes the event-driven method fast and efficient. However, it can be more convenient to write the data on a regular basis for a visualization purpose. A data writing time step dt can then be defined so that the relevant data will be written every dt. In between those writing operations, the event-driven method goes by as usual.

This can be done by defining a time variable δ that represents the time left until the next data writing. Its initial value is set to $\delta = dt$. The minimum contact time t_c is then found. It can be either smaller or larger than δ. If $t_c < dt$, the simulation jumps in time to t_c and the contact is solved. The value of δ is updated: $\delta \leftarrow \delta - t_c$, and the next t_c is computed. This process is repeated until $t_c > \delta$. If $t_c > \delta$, then the simulation advances of δ in time, writes the relevant data and the value of δ is reset to dt.

What you have to do

You are asked to:

1. Write a function

   ```
   tc = contact_time_grains(x1,x2,v1,v2,R1,R2)
   ```

   that finds the contact time between two grains. Their positions `x1,x2` and velocities `v1,v2` are arrays of size 2, while their radii `R1,R2` are floats.
   The function returns, if it exists, the contact time between the two grains. If this time does not exist or is negative, the function should return the maximum possible value `sys.float_info.max`.

2. Write a function

   ```
   tc = contact_time_walls(w,x1,v1,R1)
   ```

   that finds the contact time between a grain and a wall. The positions of the points that define the segment representing the wall are given in the two-dimensional array `w` of size `2×2`. The position `x1` and velocity `v1` of the grain are arrays of size 2, while its radius `R1` is a float.
   The function returns, if it exists, the contact time between the grain and the wall. If this time does not exist or is negative, the function should return the maximum possible value `sys.float_info.max`.

3. Write a function

```
tc, index = minimum_contact_time(x,v,R,w,indexOld)
```

that finds the minimum contact time for the set of `n` grains and `4` walls. The positions `x` and velocities `v` of the grains are two-dimensional arrays of size `n`×`2`, while their radii are given in the unidimensional array `R` of size `n`. The array `w` is multidimensional and of size `4`×`2`×`2` and contains, for each wall, the two points defining the corresponding segment. The array `indexOld` of size `2` contains the index of the objects implied in the previous contact.
The function returns the minimum contact time `tc` and an array `index` of size `2`. The first entry of `index` indicates the index of the first grain implied in the contact. If the other object implied in the contact is another grain, then the second entry of `index` is positive and corresponds to the index of the grain. If the other object implied in the contact is a wall, then the second entry of `index` is negative and corresponds to the index of the wall in the array `w`. For example, if the nearest contact in time implies the grain 3 and the wall 2, the value of `index` should be `[3,-2]`.
The suggested approach is the naive one in $O(n^2)$ that check every possible pair of objects. Trying a more efficient one may earn you some bonus points, but remember that you cannot make any change to the general architecture of the program. The argument `indexOld` should be used to ensure that the previous contact cannot be considered anymore, which could very well happen if you don't pay attention because of limited machine precision.

4. Write a function

```
dv1 = contact_solve_wall(wall,x1,v1,r)
```

that solves a contact between a grain and a wall with a restitution coefficient `r`. The two-dimensional array `wall` contains the coordinates of the two points that define the segment representing the wall. The position `x1` and velocity `v1` of the grain are arrays of size `2`, while the restitution coefficient `r` is a float.
The function returns the velocity increment `dv1` to be added to the previous velocity of the grain in the form of an array of size `2`.

5. Write a function

```
dv1,dv2 = contact_solve_grains(x1,x2,v1,v2,m1,m2,r)
```

that solves a contact between two grains with a restitution coefficient `r`. The positions `x1,x2` and velocities `v1,v2` of the two grains are arrays of size `2` while the masses `m1,m2` and the restitution coefficient `r` are floats.
The function returns the velocity increments `dv1,dv2` to be added to the previous velocities of the grains resulting from the contact in the form of arrays of size `2`.
To obtain the formula for the velocity corrections, you must do a little bit of algebra and solve the equations mentioned above.

You are given a first version of the function `update`, that updates the positions `x` and velocities `v` of the grains for a time interval `dt`:

```
def update(x,v,R,m,w,indexOld,r,dt,t):
  tc, index = minimum_contact_time(x,v,R,w,indexOld)
  if tc > dt:
    x += v*dt
    t += dt
  else:
```

```
x += v*tc
t += tc
if index[1] < 0:
  dv1 = contact_solve_wall(w[-(index[1]+1)],x[index[0],:],v[index[0],:],r)
else:
  dv1, dv2 = contact_solve_grains(x[index[0],:],x[index[1],:],v[index[0],:],
                                  v[index[1],:],m[index[0]],m[index[1]],r)
  v[index[1],:] += dv2
v[index[0],:] += dv1
indexOld = index
update(x,v,R,m,w,indexOld,r,dt-tc,t)
```

If you want to make your program more efficient, you are free to improve the body of this function, as long as you do not change its specifications.

You have to implement those functions in the file `granularGas.py`. You are also given the script `granularGasTest.py` so that you can test your functions. It is a good idea to start with the easy initial conditions you will find in the script. This script will produce an animation of your simulation, which can be written in a video file if you uncomment the corresponding line. You have until Thursday February 29 23:59 to submit your program `granularGas.py` on the server.