TP 9 : Introduction to the Lattice Boltzmann Method

Lea Heiniger

12.05.2023

Lattice Boltzmann Method

A method for fluid simulation based on the Lattice gas automata. This method can be used to numerically solve Navier-Stokes equations.

$$\partial_t u - (u \cdot \nabla)u = -\frac{1}{\rho_0}\nabla p + \nu \nabla u$$
$$\nabla \cdot u = 0$$

with u the velocity, p the pressure and ν the viscosity

Lattice Boltzmann Method

We simulate the **populations** of particles on a lattice and therefore we work with continuous values but time and space are still discrete.

Adventages over Lattice gas automata :

- Macroscopic scale
- More particles can be simulated

D2Q9 model

Two dimensions and 9 directions (8 plus non-moving population)

The value for each direction represents the **density** of particles going in that direction at that point.

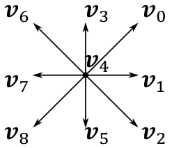
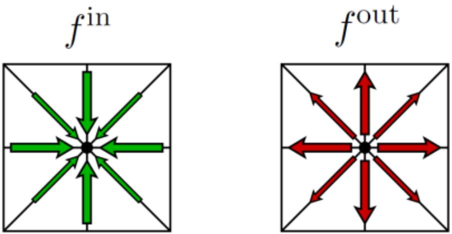


Figure: vectors of the 9 possible directions

BGK Collision model



$$f_i^{out} = f_i^{in} - \omega \cdot (f_i^{in} - E(i, \rho, u))$$

With E the local equilibrium

$$E(i, \rho, u) = \rho t_i (1 + \frac{\frac{\delta x}{\delta t} v_i \cdot u}{c_s^2} + \frac{1}{2c_s^4} (\frac{\delta x}{\delta t} v_i \cdot u)^2 - \frac{1}{2c_s^2} |u|^2)$$

Propagation

Once f_i^{out} computed we simply propagate every density to the point it is directed to in order to create f_i^{in} for time $t + \delta t$

$$f_i^{in} = f_i^{out}(x - v_i \delta t, t - \delta t)$$

Border outflows

Since we work on a limited domain we have to take into account the populations going outside of the domain.

therefore we apply outflows conditions :

```
fin[col1,0,:] = fin[col1,1,:]
fin[col3,-1,:] = fin[col3,-2,:]
fin[lin1,:,0] = fin[lin1,:,1]
fin[lin3,:,-1] = fin[lin3,:,-2]
```

Perturbation function and initial parameters

In order to simulate a Tornado we have to apply a perturbation at the center point at each time.

$$\tilde{u}(t) = u_{LB} \begin{bmatrix} cos(\omega_p t) \\ sin(\omega_p t) \end{bmatrix}$$

With u_{LB} the propagation speed and ω_p the pulse frequency

We initialise the system at equilibrium (the velocity is null everywhere) and the density to 1

macroscopic quantities

The density :

$$\rho(x,t) = \sum_{i=0}^{8} f_i^{in}(x,t)$$

The pressure :

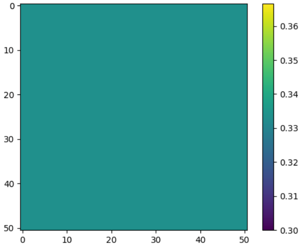
$$p = c_s^2 \rho$$

The velocity :

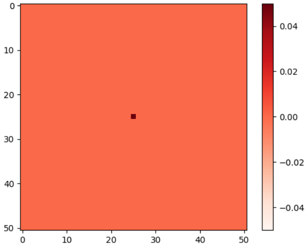
$$u(x,t) = \frac{1}{\rho(x,t)} \frac{\delta x}{\delta t} \sum_{i=0}^{8} v_i f_i^{in}(x,t)$$

Initial result

With the initial values for re, n_x, n_y, u_{LB} and $\omega_p = 0.2$

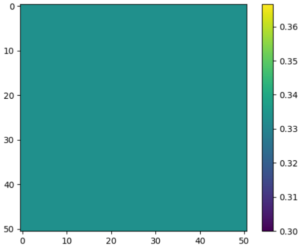


(a) Pressure

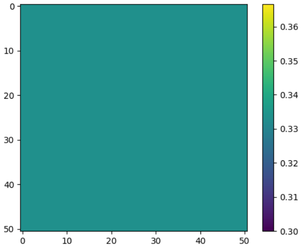


(b) velocity

the pulse frequency ω_p



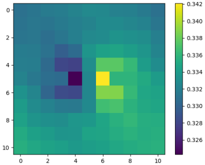
(a) $\omega_p = 0.8$



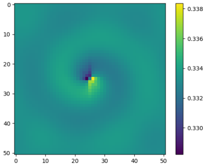
(b) $\omega_p = 0.15$

n_x, n_y

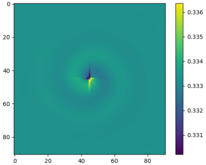
n_x and n_y define the size of the grid.



(a) $n_x = n_y = 11$



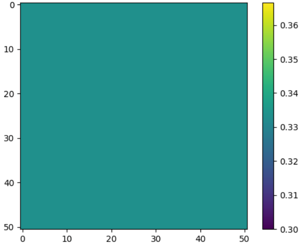
(b) $n_x = n_y = 51$



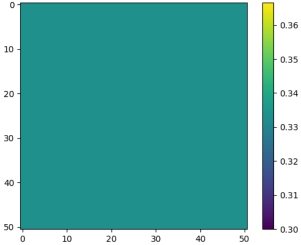
(c) $n_x = n_y = 91$

Reynolds number Re

$Re = \frac{UL}{\nu}$ is the ratio between the inertial forces and the viscous forces



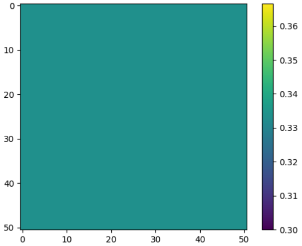
(a) $Re = 5$



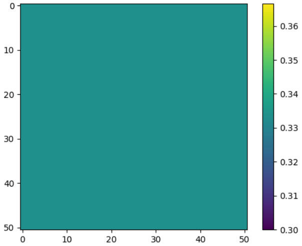
(b) $Re = 50$

u_{LB}

u_{LB} is the speed of propagation of the population.
It is related to the viscosity of the fluid.



(a) $u_{LB} = 0.1$



(b) $u_{LB} = 0.005$

Tornado center

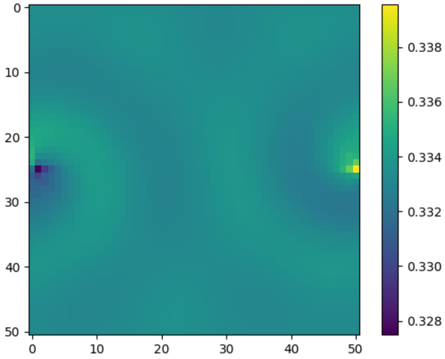
Whe introduce a new variable `center` that contains the coordinates of the tornado center on which we apply the perturbation.

```
center = array([int(nx/2), int(ny/2)])

for t in range(maxIter):
    ...
    u[:,center[0],center[1]] = perturbation(t)
    ...
```

Outflow conditions

When moving the center closer to one of the border we have to take into account the the outflow conditions conditions will have an impact.



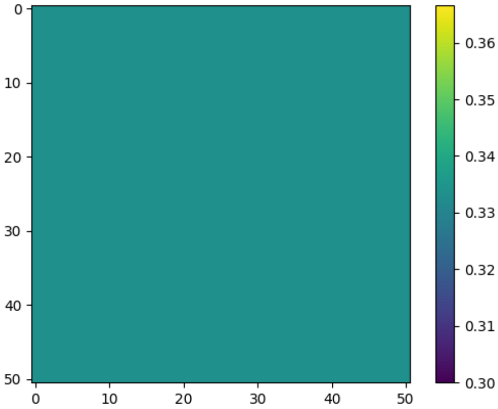
Linear trajectory

One possibility is to move the center along a straight line.

```
center = array([int(nx/4),int(ny/4)])

for t in range(maxIter):
    ...
    u[:,center[0],center[1]] = perturbation(t)
    if time%80 == 0 :
        center += 1
    ...
```

Linear trajectory



Pressure determined trajectory

A more realistic option is to move the center towards the lowest pressure at each iteration.

```
center = array([int(nx/2), int(ny/2)])

for t in range(maxIter):
    ...
    u[:,center[0],center[1]] = perturbation(t)
    c1, c2 = where(P==amin(P))
    center = array([c1[0], c2[0]])
    ...
```

Pressure determined trajectory