Computer modeling of physical phenomena

19-20.03.2024

Lab IV: Some (hopefully) helpful stuff...

Arrays

In this lab we need multiple arrays of different shapes. I suggest you use the following ones:

$\mathbf{e}$: 9×2	array of direction vectors, consisting of 9 directions, each having (x, y) components
$\rho(x, y)$: $N_x \times N_y$	density, calculated in each point
$\mathbf{u}(x, y)$: $N_x \times N_y \times 2$	velocity vector, calculated in each point and consisting of (x, y) components
$\mathbf{f}(x, y)$: $9 \times N_x \times N_y$	density population function, consisting of 9 directions and calculated in each point

Note that density is very easily calculated, just by doing a sum over the first index of population function array.

$$\rho(x, y) = \sum_i f_i(x, y)$$

It’s simillar with velocity – we just sum the direction vectors weighted by the population functions.

$$\mathbf{u}(x, y) = \frac{1}{\rho} \sum_i f_i(x, y)\, \mathbf{e}_i$$

When calculating the new density population function $\mathbf{f}(x, y)$ using arrays defined this way, you should be able to cope with all the scalar products using np.dot(u[:,:], e). The part with $|\mathbf{u}|^2$ may not work that easily with numpy, but there you can just use a sum $u_x^2 + u_y^2$. If you want something more fancy (and maybe `cleaner`), you may experiment with np.einsum or np.newaxis.

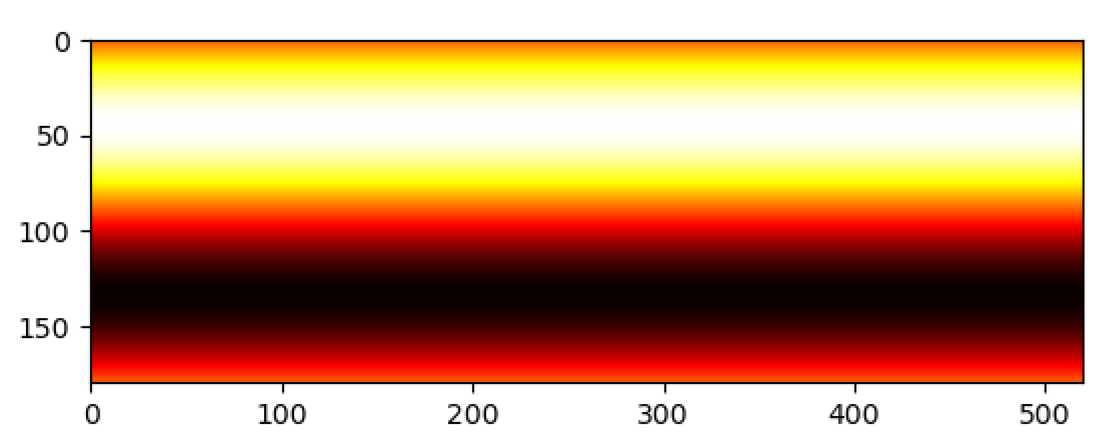
Initialization

At the beginning we are given initial velocity <u>everywhere</u>. It has only y dependence, for every x it is the same. It is also aligned along x direction, y component is equal zero.

$$u_x^0[:,:] = u_{in}[1 + \epsilon \sin(\frac{2\,\pi\,y}{N_y - 1})]$$

$$u_y^0[:,:] = 0$$

$N_x \qquad N_y$



Initial velocity field. Note that we ignore the obstacle during initialization. Also the magnitude of velocity is nearly the same everywhere, what we see on the plot is just that small y-axis perturbation.

Initialization

To start the simulation, we need to calculate the initial population function f <u>everywhere</u>. To do it, we assume the initial density is uniform.

$$\rho_0[:,:] = 1$$

$$N_x \qquad N_y$$

We assume that initial population is at the beginning equal to its equilibrium value everywhere, so we use the initial velocity and unit density to calculate it. Note that although velocity points in the x direction, all nine components of **f** are a priori non-zero.

$$f_i^0[:,:] = W_i \rho_0[:,:]\left[1 + 3(\mathbf{u}[:,:] \cdot \mathbf{e}_i) + \frac{9}{2}(\mathbf{u}[:,:] \cdot \mathbf{e}_i)^2 - \frac{3}{2}\mathrm{u}[:,:]^2\right]$$

$=1$ $\qquad\qquad$ $u_x^2 + u_y^2$

Now we use $\mathbf{f}^{\mathbf{0}}$ to start our algorithm.

Algorithm

1. Calculate the density and equilibrium distribution on the inlet (always use initial velocity $\mathbf{u_0}$ in this step).

$$\rho(0,y) = \frac{2(f_3 + f_6 + f_7) + (f_0 + f_2 + f_4)}{1 - |\mathbf{u_0}(0,y)|}$$

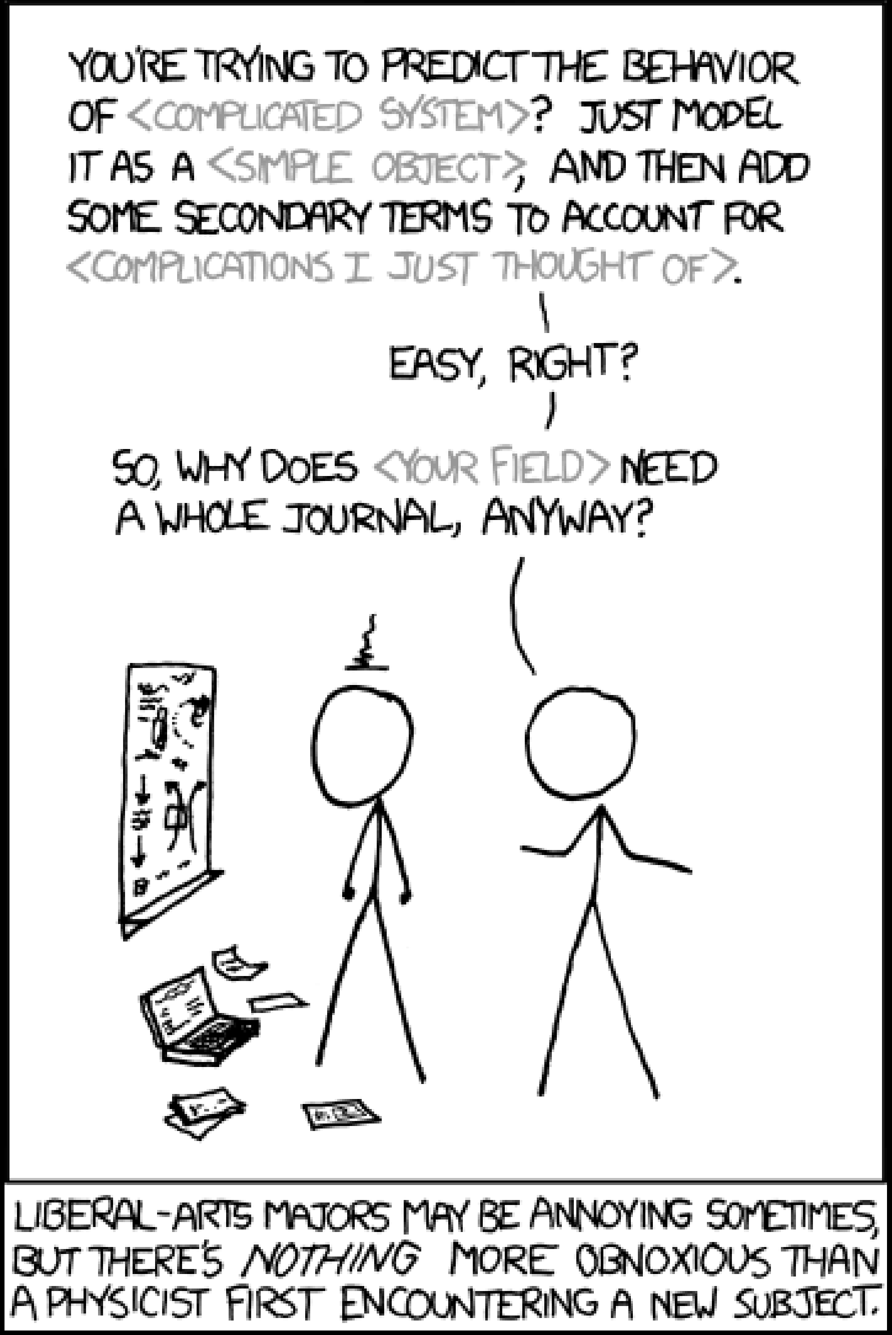
$$f_i^{eq}(0,y) = W_i\, \rho(0,y)[1 + 3(\mathbf{e}_i \cdot \mathbf{u}_0) + \frac{9}{2}(\mathbf{e}_i \cdot \mathbf{u}_0)^2 - \frac{3}{2}\mathrm{u}_0^2]$$

2. Apply the boundary conditions for the inlet and outlet.

$$f_{1,5,8}(0,y) = f_{1,5,8}^{eq}(0,y)$$

$$f_{3,6,7}(N_x,y) = f_{3,6,7}(N_x - 1, y)$$

In the first iteration, we start with $\mathbf{f} = \boldsymbol{f}^{\mathbf{0}}$. We use some of its components to calculate density and equilibrium distribution on the inlet (which are unknown to us, because they streamed from outside of the system). According to the equations, we modify components of $\mathbf{f}$ on the inlet and outlet (in the first iteration, it is still equal to $\mathbf{f}^{\mathbf{0}}$ everywhere else!). Then we proceed with the rest of the algorithm… Easy, right? ;)


YOU'RE TRYING TO PREDICT THE BEHAVIOR OF <COMPLICATED SYSTEM>? JUST MODEL IT AS A <SIMPLE OBJECT>, AND THEN ADD SOME SECONDARY TERMS TO ACCOUNT FOR <COMPLICATIONS I JUST THOUGHT OF>.
EASY, RIGHT?
SO, WHY DOES <YOUR FIELD> NEED A WHOLE JOURNAL, ANYWAY?
LIBERAL-ARTS MAJORS MAY BE ANNOYING SOMETIMES, BUT THERE'S *NOTHING* MORE OBNOXIOUS THAN A PHYSICIST FIRST ENCOUNTERING A NEW SUBJECT.