Problem Set 9

Exercises for the Lecture Fundamentals of Simulation Methods (WS24/25)
Cornelis Dullemond and Michela Mapelli (Lecture Tuesday and Thursday 14h - 16h)
Lucia Haerer (Tutor Group T1 on Wednesday 11h - 13h)
Stefano Rinaldi (Tutor Group T2 on Thursday 11h - 13h)
Nicholas Storm (Tutor Group T3 on Friday 11h - 13h)
Submit the solution in electronic by the **Friday 6pm**, **December 20, 2024**.

9. Hyperbolic PDEs and advection algorithms

9.1. Advection: Instability of symmetric schemes (*14 points*)

We consider the following 1D advection equation:

$$\frac{\partial\rho(x,t)}{\partial t} + u\frac{\partial\rho(x,t)}{\partial x} = 0 \tag{1}$$

with $u > 0$ *constant in space and time*. We will set $u = 1$ in this exercise. The initial condition is

$$\rho(x,0) = \begin{cases} 1 & \text{for} \quad x \leq 1 \\ 0.25 & \text{for} \quad x > 1 \end{cases} \tag{2}$$

We consider the problem on a domain of $0 \leq x \leq 4$, with N gridpoints evenly spaced between 0 and 4. As boundary conditions we set:

$$\begin{aligned} \rho(0,t) &= 1 & (3)\\ \rho(4,t) &= 0.25 & (4) \end{aligned}$$

which we impose with a single "ghost grid point" (more commonly called "ghost cell") on each side of the domain. For gridpoints $i = 1 \cdots N-2$ (i.e., all but the two "ghost grid points") we can discretize the above advection equation as

$$\frac{\rho_i^{n+1} - \rho_i^n}{t_{n+1} - t_n} + u\,\frac{\rho_{i+1}^n - \rho_{i-1}^n}{x_{i+1} - x_{i-1}} = 0 \tag{5}$$

We will see that this discretization is numerically unstable. Let us experiment...

1. For $N = 100$ and $u = 1$, compute the maximum allowed time step Δt according to the Courant-Friedrichs-Lewy (CFL) condition.

2. Show that if you do 200 time steps between $t = 0$ and $t = 3$, you are well within this CFL limit.

3. Write Eq. (5) as an explicit expression for the future ρ_i^{n+1}, given the current values ρ_i^n, for a given time step $\Delta t = t_{n+1} - t_n$.

4. Write a function `advect_symmetric()` that returns an array for ρ_i^{n+1}, given the current values ρ_i^n and Δt, whereby the boundary conditions on gridpoints $i = 0$ (left) and $i = N - 1$ (right) are also accounted for.

5. Start with the above stated initial condition, and carry out 200 time steps between $t = 0$ and $t = 3$. Store the ρ_i^n each step along the way, so that you can see (plot) how the result develops over time. Tipp 1: You can use the `viewarr.py` tool provided to you to animate the solution. Tipp 2: You will note that the solution is numerically unstable. Plot the final result (at $t = 3$) to show this.

6. Try to "solve" this by taking a 10 times smaller time step (and thus 10 times more time steps!). Does the instability go away?

A stable advection algorithm would be, e.g., the "upwind algorithm". For $u > 0$ each gridpoints only "looks to the left", i.e., the direction where the information comes from. And for $u < 0$ of course to the right. The discretiation is then:

$$\frac{\rho_i^{n+1} - \rho_i^n}{t_{n+1} - t_n} + u\frac{\rho_i^n - \rho_{i-1}^n}{x_i - x_{i-1}} = 0 \tag{6}$$

for $u > 0$, and

$$\frac{\rho_i^{n+1} - \rho_i^n}{t_{n+1} - t_n} + u\frac{\rho_{i+1}^n - \rho_i^n}{x_{i+1} - x_i} = 0 \tag{7}$$

for $u < 0$.

7. Write a function `advect_upwind()` (as a replacement of `advect_symmetric()`) that implements this "upwind algorithm".

8. Apply this method to the above advection problem, and show that it shifts the function from left to right without instability.

9. You will, however, notice that the "shifted" solution is not quite what one would hope for. Describe this problem, and explain what causes it (see lecture).

9.2. Advection: Conservation equation (*10 points*)

In any practical problem, the velocity u will *not* be constant in space and time. In that case we must be sure whether we are dealing with a *conservation equation*

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial \rho(x,t)u(x,t)}{\partial x} = 0 \tag{8}$$

or a non-conservative advection equation (let's call it a *translation equation*)

$$\frac{\partial \rho(x,t)}{\partial t} + u(x,t)\frac{\partial \rho(x,t)}{\partial x} = 0 \tag{9}$$

In the first case we can write the equation as

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial F(x,t)}{\partial x} = 0 \tag{10}$$

where $F(x,t) = \rho(x,t)u(x,t)$ is the *flux*. In the second case we can write the equation as

$$\frac{D\rho(x,t)}{Dt} = 0 \tag{11}$$

where D/Dt is the *Lagrangian time derivative* (often called *comoving time derivative*) defined as

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} \tag{12}$$

Let us start from the same setup as in the previous exercise, but now define the velocity $u(x,t) = u(x)$ as

$$u(x) = 1 - \tanh(2(x-2)) \tag{13}$$

Note: This is a tangens-hyperbolicus! So this is a smooth function going from $u \simeq 2$ at the left to $u \simeq 0$ to the right.
We provide two Python functions: `advect_conservative_donorcell()` and `advect_translation_upwind()` in the file `exercise_2_tools.py`. These are the functions similar to the ones you created in the first exercise, but now for the conservative and translational versions of the advection equation, capable of dealing with non-constant $u(x)$. Note that `advect_conservative_donorcell()` expects the x and u values at the $N+1$ cell walls, whereas `advect_translation_upwind()` expects the x and u values at the N cell centers.

1. Implement the above non-constant form of $u(x)$ into your program.
2. Apply `advect_conservative_donorcell()` for the same 100 spatial cells between $x = 0$ and $x = 4$, and 200 time steps between $t = 0$ and $t = 3$ as before, and show that the density "piles up". Explain why. Tipp: Come up with a useful definition of the location of the cell walls and cell centers.
3. Now apply `advect_translation_upwind()` to that problem, and show that the density does, in this case, not pile up. Explain why.