PHYS358: Homework 8 (due Dec 4, 9:30am)

PDEs: Advective Initial Value Problems

Note: Please read the complete homework instructions before you start. I strongly recommend Numerical Recipes, Chapter 19.1.

In our final homework, we'll discuss finite differencing schemes to solve advective initial value problems (AIVPs). We will build a suite of five AIVPs integrators, comparing their strengths and shortcomings. Note that this is just the starting point to solving hyperbolic PDEs (i.e. PDEs describing wave propagation).

At the core of any wave propagation problem lies an **advection equation** of the form

$$\frac{\partial q}{\partial t} = -c\frac{\partial q}{\partial x}. \tag{1}$$

The quantity q (e.g. a blob of green paint in water) is transported ("advected") at a velocity c. Eq. 1 is a *conservation law* for q. The advection velocity $c > 0$ shall be constant for our purposes. You can think of this homework coming in two parts: Part I discusses the (rather useless) FTCS scheme and a first attempt to fix it, namely the Lax scheme. It turns out that the fix provided by the Lax scheme is useless for all practical purposes, therefore, we will explore three more schemes (Leapfrog, Lax-Wendroff, Upwind), all with their strengths and weaknesses.

Euler Method (FTCS scheme)

This method we get by straight-forwardly discretizing eq. 1:

$$q_j^{n+1} = q_j^n - \frac{\alpha}{2}(q_{j+1}^n - q_{j-1}^n), \tag{2}$$

where $n = 0 \ldots N$ denotes the time step, and $j = 1 \ldots J$ denotes the (active) positions. As for the diffusion and potential problems, we will assume **cell-centered** quantities, therefore, the **active** spatial indices run over $j = 1 \ldots J$ inclusive. The integration factor α is given by

$$\alpha \equiv \frac{c\,\Delta t}{\Delta x}. \tag{3}$$

Note also that we wrote the spatial derivative in symmetric form, hence the name: Forward Time Centered Space. For $j = 0$ and $j = J + 1$, we will need to specify boundary conditions, and we also will need to provide initial conditions at $n = 0$.

Simple enough, no? Below, you will be asked to code this up (the integrator structure is provided in `pde_advection.py` and looks very much like that of the diffusion problem we discussed a while ago). Once you run the test however (see below), you'll be surprised: the method does not work at all. Why's that?

As for the diffusion problem, let's analyze the stability of the method by writing a solution at time n and position j as

$$q_j^n = \xi^n(k)\, e^{ijk\Delta x}, \tag{4}$$

i.e. as Fourier components.

(8a) The Stability of the FTCS scheme [5pts]: Show that

$$\xi(k) = 1 - i\alpha \sin(k\Delta x) \tag{5}$$

for the FTCS scheme, and discuss why the scheme is unconditionally unstable.

Solution: Use eq. 2 and 4. The ξ^n cancel, as well as the factors $\exp(ijk\Delta x)$, leaving

$$\xi - 1 = -\frac{\alpha}{2}(e^{ik\Delta x} - e^{-ik\Delta x}),$$

resulting in eq. 5. This is $> 1\,\forall k \neq 0$, hence $\xi > 1\,\forall k$, and the method is unconditionally unstable.

The Lax Method

The problems with the FTCS scheme arise from the fact that the center point (q_j^n) is never connected with its neighbors (just draw a sketch of the scheme to convince yourself). The remedy found by Lax is to replace q_j^n by the average of its left and right neighbors:

$$q_j^n \to \frac{1}{2}(q_{j-1}^n + q_{j+1}^n). \tag{6}$$

Thus, the *Lax Method* reads

$$q_j^{n+1} = \frac{1}{2}(q_{j-1}^n + q_{j+1}^n) - \frac{\alpha}{2}(q_{j+1}^n - q_{j-1}^n). \tag{7}$$

(8b) The Stability of the Lax Method [5pts]: Show that for the Lax Method,

$$\xi = \cos(k\Delta x) - i\alpha \sin(k\Delta x), \tag{8}$$

and prove that the Lax scheme is stable for $\alpha \leq 1$. This condition equation is the **Courant-Friedrichs-Lewy** stability criterion (also "Courant" or "CFL" condition), which simply states that within one timestep Δt, information must not travel further than by at most one resolution element Δx.

Solution: Same as above, just that the average of q_{j+1} and q_{j-1} leads to a cos-term. The condition $|\xi|^2 \leq 1$ for stability, and the fact that $\cos^2 + \sin^2 = 1$, result in the CFL condition.

(8c) Why is the Lax Method stable? [5pts]: Show that the Lax method can be written as

$$\frac{\partial q}{\partial t} = -c\frac{\partial q}{\partial x} + \frac{(\Delta x)^2}{2\Delta t}\frac{\partial^2 q}{\partial x^2}, \tag{9}$$

i.e. the Lax scheme is the FTCS scheme with a diffusion term added. When does the diffusion term *not* affect the solution?

Solution: Can be done by adding and subtracting finite-difference form of the diffusion operator. For $|\xi| \equiv 1$, the terms cancel, and the diffusion operator vanishes. This is not surprising, since for that value, the solution is just copied from j to $j+1$ (for $c > 0$).

The Test: To test the integrators, we will advect a top-hat function, with

$$q(x) = \begin{cases} 1 & \text{for } -1/4 < x < 1/4 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The function is supposed to make exactly one roundtrip, i.e. the time integration should go over exactly one crossing time

$$T \equiv \frac{x_{max} - x_{min}}{c}. \tag{11}$$

Thus, the spatial boundary conditions must be periodic. As before, the integration coefficient is given by

$$\alpha \equiv \frac{c\,\Delta t}{\Delta x}. \tag{12}$$

Here are the parameters (some of them are already set – check `pde_advection.py`):

$$c = 1 \tag{13}$$
$$J = 100\text{: number of spatial support points} \tag{14}$$
$$N = T\,div\,\Delta t\text{: number of time integration steps} \tag{15}$$
$$x_{min} = -1/2 \tag{16}$$
$$x_{max} = +1/2 \tag{17}$$
$$T = 1\text{: one period, meaning the profile does one "roundtrip".} \tag{18}$$

Make sure your top-hat function does *exactly* one round-trip.

(8d) The Integrators [10pts]:

Implement all the integrators above (see `pde_advection.py` for calling sequence etc). The program should show two plots: the initial condition and the solution after exactly one round trip, and the midpoint value `q[J//2,:]` against time. Obviously, after one round trip, the initial and final profile should be identical.

Solution: See `pde_advection.py` of the solution set. [5 pts] per integrator.

(8e) Test of the integrators [10 pts]:

1. Run `pde_advection.py 100 1.0 ftcs tophat` and confirm your results from (8a). Repeat with $\alpha = 0.1$. Anything changing?

2. Let's forget about `ftcs` and try `lax`. Set $\alpha = 1.0$. Describe how the Lax-scheme works for $\alpha = 1$ to explain the results.

3. Now try the Lax-scheme with $\alpha = 0.5$, and then 0.1. What changes? If you reduce alpha, to what (constant) value does $q(x)$ converge? Explain the result.

Solution:

1. `ftcs` should be unstable. Reducing α usually just increases values (more amplification) [1pt]

2. This should perfectly reproduce the initial conditions after exactly one crossing time [2pts], because Lax at $\alpha = 1$ just means copying q_j to q_{j+1} (for $c > 0$) [2pts]. No averages in space or time, hence no losses.

3. Now diffusion kicks in, and the amplitude decreases [2pts]. For small α, $q \to 0.5\ \forall x$ for our initial condition [2pts]. The diffusion constant is proportional to $1/\Delta t$, therefore, reducing Δt increases diffusion [1pt].

So far, we haven't been very successful. We developed one method that doesn't work at all, because it's inherently unstable, and another one which is ridiculously diffusive. Can we do better?

(8f) Leapfrog scheme [10 pts]: One option is to move to second order accuracy in time. This leads us to the leapfrog scheme

$$q_j^{n+1} = q_j^{n-1} = -\alpha(q_{j+1}^n - q_{j-1}^n). \tag{19}$$

Yes, this scheme draws on the *two previous* steps n and $n-1$.

1. Explain why this scheme is 2nd order accurate in time. *Hint: Remember the definition of* α.
2. Implement and test the leapfrog scheme. The trick is to get the scheme started by a 1st order (in time) step. The easiest way is to use eq. (19.1.27) in Numerical Recipes (1992) for $c > 0$. Repeat the above tests for `tophat` at $\alpha = 1.0, 0.5, 0.1$. What do the results look like? Finally, test `gaussian` for $\alpha = 0.5, 0.1$. Why does this work better than the `tophat`?

Solution: The time derivative is now centered on n, in the same way the spatial derivative is centered on j. Thus, the scheme formally is of 2nd order in time and space [2 pts]. Implementation of leapfrog, see `pde_advection.py` [3pts]. For $\alpha = 1$, the solution is just copied over. For lower α, the solution does not decay that quickly, but develops strong oscillations [2pts]. The Gaussian works better because the discontinuities in the tophat profile lead to "infinite" derivatives and thus overshoots. Note that the Gaussian is not perfect [3pts].

(8g) The Lax-Wendroff scheme [10pts]: Fourth scheme, one to go. It is 2nd order in time and space, and, not unlike the Lax scheme, it introduces a (small) diffusion to deal with oscillations. For our purposes, the scheme can be written (compare to eq. (19.1.39) of Numerical Recipes) as

$$q_j^{n+1} = q_j^n - \frac{1}{2}\alpha\left((q_{j+1}^n + q_j^n) - \alpha(q_{j+1}^n - q_j^n) - (q_j^n + q_{j-1}^n) + \alpha(q_j^n - q_{j-1}^n)\right). \tag{20}$$

Implement the scheme and repeat the tests of (8f). Does the scheme work better or worse than the Leapfrog scheme?

Solution: Implementation [5pts]. Scheme is slightly more stable, but still shows oscillations for `tophat` [2pts]. The `gaussian` profile is nearly fine, but shows some asymmetry, as in the leapfrog scheme [3pts].

(8h) The (1st order) Upwind scheme [10pts]: Let's go back to the output of e.g. the Lax-Wendroff scheme. For $c > 0$, the tophat is advected to the right. Imagine for a moment that you, the observer, are located at position $x = 0.3$, and the tophat starts moving towards the right (its right wall being at $x = 0.25$). If you were looking down the x-axis (toward negative x), you'd see a wall of q approaching, rising from $q = 0$ to $q = 1$. You'd be eyeing the wall, calculating (based

on its approach rate) the moment it reaches you. You probably would not turn around and look toward **positive** x, because you know this is all "history", i.e. material that's **downstream** from you won't affect you. It's the wall $q = 1$ **upstream** from you that keeps you worried. Now, once the wall reaches your position at $x = 0.3$, imagine you'd be (gently, but instantaneously) lifted up to $q = 1$. You'd still sit at position $x = 0.3$, but now $q = 1$. Now you turn around and face toward positive x: you'd see the precipice move away from you. (Turn around within the next $t < 0.5$, because q will drop to 0 again...).

The downfall of the Lax-Wendroff and Leapfrog methods is that they keep looking **both ways**, i.e. they calculate the next state at position j based on information from $j - 1$ **and** $j + 1$. But we've just seen that this is unphysical: material **downstream** of position j (i.e. in our case at $j+1$) should not contribute to the solution. The fact that both schemes draw on $j \pm 1$ to calculate j means that perturbations downstream from j can travel **upstream**, violating causality, and thus leading to instabilities. This is why Lax-Wendroff and Leapfrog both show oscillations **upstream** of the discontinuities.

We've just met another physics problem whose numerical implementation requires some thought about the physical principles at play (remember our attempts to solve the Kepler problem with RK integrators – the attempt was doomed to fail because the RK integrators turned out to neither conserve energy nor angular momentum, sending Mercury on its merry way, eventually). Here, the physical principle is causality: **Only upstream information should contributed to the solution of an advection problem**. This is a basic principle baked into any modern technique for modeling fluid dynamics. The technical term for the above is "upwinding", or, in our case "upwind differencing". The first-order version looks like

$$q_j^{n+1} = q_j^n - \alpha \begin{cases} (q_j^n - q_{j-1}^n), & c > 0 \\ (q_{j+1}^n - q_j^n), & c < 0 \end{cases} \tag{21}$$

1. Implement the upwind scheme in `upwind`. Remember that for our case $c > 0$.

2. Rerun the tests of (8g), i.e. run the `tophat` for $\alpha = 1.0, 0.5, 0.1$ and compare to Lax-Wendroff. Discuss the results. Specifically, do you still see oscillations with the `upwind` scheme? Does it perform better or worse than the Lax scheme regarding diffusion?

3. Given all four (stable) schemes (Lax, Leapfrog, Lax-Wendroff, Upwind), which scheme performed the task of advecting a tophat profile for $\alpha = 0.5$ best?

Solution: Implementation see `pde_advection.py` [3pts]. There shouldn't be any oscillations now [2pts]. Upwind performs better than Lax regarding diffusion [2pts]. Upwind works best of all four schemes [3pts].