ASTR4260: Problem Set #9

Due: Wednesday, December 6, 2023

adapted from M. Zingale, Stony Brook

Problem 1

Consider the linear advection equation

$$\frac{\partial a}{\partial t} + u\frac{\partial a}{\partial x} = 0 \tag{1}$$

In class, we saw that an explicit first-order finite-difference upwind discretization of this resulted in a stable method. Now let us consider an implicit discretization of this same upwind method. To do this, we evaluate the spatial derivative at the new time rather than the old time.

1. Perform linear stability analysis as we did in class to show that this method is stable for any choice of Courant number.
2. Solve this implicit discretization numerically with periodic boundary conditions. You may adapt the advection code discussed in class. When you write this out, you will find that you have a coupled linear system of equations that can be written in a matrix form. The matrix is almost tridiagonal, except for a single element in a corner resulting from the periodicity. You can solve this system with any matrix solver routine you wish. You may wish to investigate `scipy.sparse.linalg` for an optimized, though more complex, approach to this problem.
3. For the initial conditions, choose a Gaussian:

   $$a(x, t = 0) = e^{-(x-0.5)^2/0.1^2} \tag{2}$$

   Evolve this with periodic boundary conditions on $x = [0, 1]$. [**Graduates:** Also test a top hat function.] Compare the solutions for the Gaussian graphically after one period with 64 and 256 grid points, with C = 0.5, 1, 10.

Problem 2

The second-order, Lax-Wendroff method can be derived by Taylor expanding in time, keeping terms to $O(\Delta t^2)$, and then replacing the time-derivatives with spatial derivatives from the PDE. By using centered, second-order spatial derivatives, this results in a method that is second-order in space and time. For our advection equation, the update appears as (using the convention that subscripts are spatial indexes and superscripts are time indexes)

$$a_i^{n+1} = a_i^n - \frac{C}{2}(a_{i+1}^n - a_{i-1}^n) + \frac{C^2}{2}(a_{i+1}^n - 2a_i^n + a_{i-1}^n), \tag{3}$$

where the first term is just unstable FTCS, but the second term is effectively a positive diffusion. This method can be shown to be stable—the explicitly included diffusion counteracts the numerical diffusion that made the method unstable (as shown in class). Implement this method on a cell-centered finite-difference grid (again following the routine discussed in class if you wish) and set up the same test (or tests, for **graduates**) as for Problem 1.

We want to measure the convergence of this method. Convergence for PDEs requires that you change Δt in step with Δx—this is handled automatically if you use the same Courant number at each resolution.

To get a single error number from the spatially discretized solution, we need to define a norm for any array ϕ:

$$||\phi||_2 = \left\{ \Delta x \sum_{i=0}^{N-1} \phi_i^2 \right\}^{1/2} . \tag{4}$$

This is called the L2 norm of ϕ. We can then define our error as

$$\epsilon_{\Delta x} = ||a(x, t = T) - a(x, t = 0)||_2, \tag{5}$$

where T is one period. This is just the pointwise RMS error of the final solution compared to the initial solution, normalized by Δx.

Plot $\epsilon_{\Delta x}$ vs. the number of points N for multiple grid resolutions and estimate the convergence rate for a Gaussian. Try to reuse the code you wrote for problem 1 by generalizing it rather than repeating it. [**Graduates:** Also try a periodic function like a sine wave. How does it differ?]

Hints: depending on your choice of Δt, you may evolve past $t = T$—put a check in the evolution loop to make the final timestep smaller, if necessary, so you end exactly at $t = T$. (Note that if you reduce Δt at the end, then the Courant number C reduces by a corresponding amount). If you don't see $O(\Delta x^2)$ convergence, then you likely have a bug somewhere.