Computer Problem 2
2D Advection, Rotational Flow

Due: 4pm, Tuesday, Feb. 9.
Turn in: your code (printed *and* submitted on Compass), and statistics & plots (on paper).

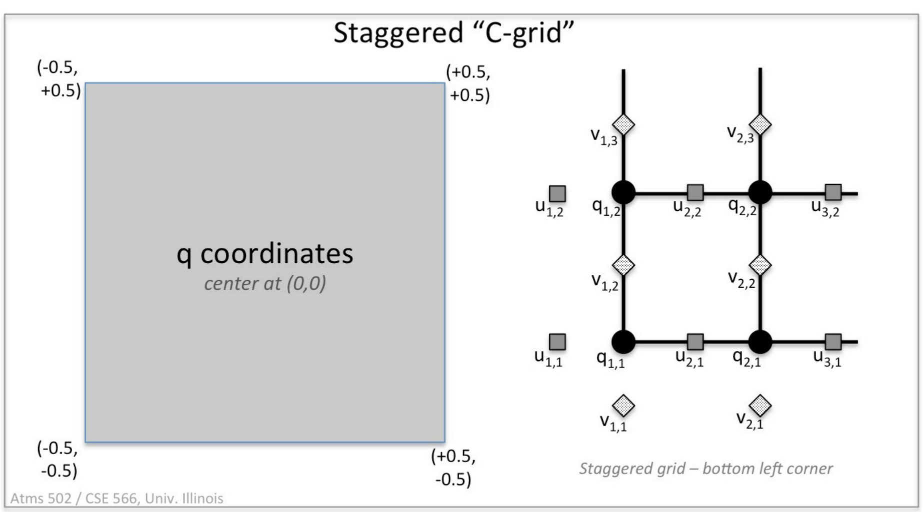
Problem being solved: 2-D linear advection via fractional step (directional) splitting
Initial conditions: Circular field (decreases as 1/radius; "cone" if plotted in 3-D)
Boundary conditions: 0-gradient (extended from grid boundary) in both directions
Flow field: rotational flow (counter-clockwise), constant w/time
Evaluation: You will compute Takacs (1985) error statistics over the 2-D domain using a known solution – the initial condition, since we will integrate over one 2-D cycle.

Methods:

1. Lax-Wendroff	$q_j^{n+1} = q_j^n - \frac{\nu}{2}\left(q_{j+1}^n - q_{j-1}^n\right) + \frac{\nu^2}{2}\left(q_{j+1}^n - 2q_j^n + q_{j-1}^n\right)$
2. Upstream	$\nu > 0: \; \left\{ q_j^{n+1} = q_j^n - \nu\left(q_j^n - q_{j-1}^n\right)\right.$ $\nu < 0: \; \left\{ q_j^{n+1} = q_j^n - \nu\left(q_{j+1}^n - q_j^n\right)\right.$

All methods here use one ghost point, as in program #1.

Domain: The computational domain is a *two-dimensional staggered C-grid*, with the scalar field (hereafter called ***q***) in a 121x121 domain, with Δx=Δy=*1.0/real(nx-1)*. The physical coordinates *for **q*** range from -0.5 to +0.5 in each direction. The ***u*** and ***v*** wind field components vary in space but are *time*-invariant. In C-grid staggering, the physical location for ***u(i,j)*** is ½Δx to the left of ***q(i,j)***; ***v(i,j)*** is located ½Δy below ***q***, as shown below. Due to staggering, u is dimensioned *(nx+1,ny),* and v is dimensioned *(nx,ny+1).*



Initial conditions:

Scalar "q" (the "cone" shape)	$q_{i,j} = \begin{cases} 0, \text{ if } d > r \\ 5[1+\cos(\pi d/r)], \text{ otherwise} \end{cases}$	*where* d=$\sqrt{(x_{i,j}-x_0)^2+(y_{i,j}-y_0)^2}$
Wind field	u(x,y) = -2y; v(x,y) = 2x *(rotational flow)*	

Settings

- Initial condition, time step: cone radius r = **0.130**, center x_0,y_0=(0.0, 0.30); take 600 steps (one cycle); Δt=(π/600). Your true final solution = the initial condition.
- Error analysis: *put these computations in a (sub)routine, **not** the main program.* Compute error stats for the final solution following Takacs (1985); print total, dissipation and dispersion error **to 5 decimal places**. Compute total error with Takacs' eqn. 6.1. The dissipation and dispersion errors are eqns. 6.6 and 6.7. In expressions (6.5-6.7), there is a *linear correlation coefficient* ρ; compute as:

$\rho = \dfrac{\sum(q_d-\bar{q}_d)(q_T-\bar{q}_T)}{\sqrt{\sum(q_d-\bar{q}_d)^2\sum(q_T-\bar{q}_T)^2}}$	q_d and q_T here refer to the finite difference and true solutions for the scalar field "q"

- Read in: scheme choice (Lax-Wendroff or Upstream) ***and*** the plotting interval.

Advection schemes: you are using the *Lax-Wendroff* and *Upstream* methods, which are both 2-time-level and 1-D. Apply them in 2-D by first doing advection in x, and then y-advection across the grid (the y-advection uses the results of the x-advection). We will stick with the sequence *x-advection*, *y-advection*, *x-*, *y-* … for all computations here.

Boundary conditions: simple "extension" of boundary values. If you need s(i-1) or s(i+1) near a boundary, use the boundary value (for x *and* y). This is "*zero-gradient*."

Use this plan for changing program 1 => program #2:

- You need 2 two-dimensional scalar arrays of size *(nx,ny)* and named *q1* and *q2* for the scalar being advected. Include *1 ghost point on each side* of your 2d *scalar* arrays. For velocity components, create arrays *u(nx+1,ny)* and *v(nx,ny+1)*. Velocity variables do not evolve; *no ghost points needed* for them.
- *Confirm that your initial conditions are OK **first*** before proceeding further.

Coding requirements for this problem:

- Do *not* simply add upstream code to your advection routine! Instead …
- Copy your 1D advection routine to a new "advect1d" routine; add *1D upstream* code in the advect1d routine. "advection" calls advect1d to do the work!
- Change main advection routine to handle x- vs. y-advection passes, each calling your "advect1d" routine each time step. It *must* pass the scheme type to advect1d.

Hand in:

- Printout of code
- 10 plots: contour *and* 3D surface plots of the initial condition and, for each method, solution at 600 steps. Also, for each method, plots of $q_{min}(t)$ and $q_{max}(t)$.
- Print and hand in Takacs error data to **5** decimal places for both final solutions.
- Also upload your code to Compass as in program 1.