Computer Problem 3
2D Advection, Deformational Flow

Due: Thursday, Feb. 25.
Turn in: **your code** (printed out, and *also* emailed to me), **and plotted results** (on paper).

Problem being solved: 2-D linear advection with fractional step (directional) splitting.
Boundary condition: 0-gradient (as in program #2)

Numerical methods: Lax-Wendroff, 6th-order Crowley, and Takacs.

1. Crowley 2nd-order *(**same as** 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. Crowley 6th-order	*See Tremback p. 542, ORD=6 (advective form)* *(link to Tremback paper is on our References page)*
3. Takacs (1985) *(Note: his formulation is for $\nu > 0$; I have modified it so it can also be used with negative courant numbers.)*	$\nu \ge 0: \begin{cases} q_j^{n+1} = q_j^n - \frac{\nu}{2}\left(q_{j+1}^n - q_{j-1}^n\right) + \frac{\nu^2}{2}\left(q_{j+1}^n - 2q_j^n + q_{j-1}^n\right) \\ \qquad - \left(\frac{1+\nu}{6}\right)\nu(\nu-1)\left(q_{j+1}^n - 3q_j^n + 3q_{j-1}^n - q_{j-2}^n\right) \end{cases}$ $\nu < 0: \begin{cases} q_j^{n+1} = q_j^n - \frac{\nu}{2}\left(q_{j+1}^n - q_{j-1}^n\right) + \frac{\nu^2}{2}\left(q_{j+1}^n - 2q_j^n + q_{j-1}^n\right) \\ \qquad - \left(\frac{1+\lvert\nu\rvert}{6}\right)\nu(\nu+1)\left(q_{j-1}^n - 3q_j^n + 3q_{j+1}^n - q_{j+2}^n\right) \end{cases}$

Domain: The domain size/layout are the same as program #2. However, ***u***, ***v*** *differ from the last problem*, as does the initial position/size of the cone. Since ***u(i,j)*** (½ grid length to the left of ***q***) and ***v(i,j)*** (½ grid length below ***q***) are now functions of x *and* y, you have to determine x and y locations of q, u, and v separately when preparing initial conditions.

If you see asymmetry (discussed below) in your solutions, the #1 most likely cause is a problem in the initial conditions – probably the X and Y coordinates used in creating the initial conditions. All you need is for the cone or the U or V velocity components to be incorrectly located by dx/2 or dy/2 to result in erroneous behavior. On the *positive* side, tests with this sort of symmetry property are great at helping locate any problems in the initial condition, boundary condition or advection schemes, which is why we use them.

Advection method: Lax-Wendroff is unchanged, and 6th-order Crowley and Takacs are new. All use directional splitting, X followed by Y-advection. Note that Takacs needs 2 ghost points, and 6th-order Crowley requires 3.

Settings: nx, ny, cone center, cone radius, time step, # of steps – *see program 3 page on the ATMS 502 / CSE 566 class web site.*

Read In: the numerical method to use • number of steps to run • how often to plot

Initial conditions: Define q as before, though the cone radius and center are changed.

Wind field – deformation	$u(x,y)=\sin\left[4\pi x\right]\bullet\sin\left[4\pi y\right]$ $v(x,y)=\cos\left[4\pi x\right]\bullet\cos\left[4\pi y\right]$

Code layout requirements:

1. you **<u>must</u>** use and call *separate advection* and *advect1d routines*. Do the 1-D advection step fully in *a separate* **advect1d** routine, where 1-D methods reside. Do not combine 2D, 1D steps or embed integration code in the main program.
2. do not "hard-code" your program for any scheme! So your code must be set up for the maximum number of ghost points you could need, and to run any scheme.
3. pass the *staggered* u or v data to *advect1d*.
4. do not hard-code the program's grid dimensions or number of ghost points except at the <u>start</u> of the main program.
5. do not (in C) use point 0 as always a single ghost point, 1 as the first physical point, etc.
6. code generally! *See class content page for full code rules. Your program will not be accepted if these rules are not followed.*

These rules will save you time in the long run, and result in a clean(er) program.

Hand in:

- Contour plots of the initial u, v, and q field. And, for each method, create *sfc* and *contr* plots of the solutions at 25, 50, and 100 steps.
- Qmin(t) & Qmax(t) plots are *not* necessary, but **do** call my *contr* and *sfc* routines.