Computer Problem 2
2D Advection, Rotational Flow

Due: in class, Tuesday, Feb. 12.
Turn in: your code (submitted on Moodle), 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	$s_j^{n+1} = s_j^n - \frac{\nu}{2}\left(s_{j+1}^n - s_{j-1}^n\right) + \frac{\nu^2}{2}\left(s_{j+1}^n - 2s_j^n + s_{j-1}^n\right)$
2. Takacs (1985) *(Note: his formulation is for $\nu > 0$; I have modified it so it can also be used with negative courant numbers.)*	$\nu \ge 0:$ $\begin{cases} s_j^{n+1} = s_j^n - \frac{\nu}{2}\left(s_{j+1}^n - s_{j-1}^n\right) + \frac{\nu^2}{2}\left(s_{j+1}^n - 2s_j^n + s_{j-1}^n\right) \\ \qquad -\left(\frac{1+\nu}{6}\right)\nu(\nu-1)\left(s_{j+1}^n - 3s_j^n + 3s_{j-1}^n - s_{j-2}^n\right) \end{cases}$ $\nu < 0:$ $\begin{cases} s_j^{n+1} = s_j^n - \frac{\nu}{2}\left(s_{j+1}^n - s_{j-1}^n\right) + \frac{\nu^2}{2}\left(s_{j+1}^n - 2s_j^n + s_{j-1}^n\right) \\ \qquad -\left(\frac{1+\lvert\nu\rvert}{6}\right)\nu(\nu+1)\left(s_{j-1}^n - 3s_j^n + 3s_{j+1}^n - s_{j+2}^n\right) \end{cases}$
3. Crowley 6th-order *extra credit: 10%*	*See Tremback p. 542, ORD=6 (advective form)*

Utilize *two* ghost points to accommodate Takacs' method; *three* if also doing Crowley.

Domain: The computational domain is a *two-dimensional staggered C-grid*, with the scalar field (hereafter called ***s***) in a 121x121 domain, with Δx=Δy=*1.0/real(nx-1)*. The physical coordinates *for* ***s*** range from -0.5 to +0.5 in each direction. The ***u*** and ***v*** wind field components vary in space but are *time*-invariant and thus have no ghost points. In C-grid staggering, the physical location for ***u(i,j)*** is ½Δx to the left of ***s(i,j)***; ***v(i,j)*** is located ½Δy below ***s***. Due to staggering, u is dimensioned *(nx+1,ny),* and v is dimensioned *(nx,ny+1)* ... again, without ghost points for the *u, v* velocity variables.

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. 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.

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

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

Settings

- Initial condition, time step: cone radius r = **0.120**, center x_0,y_0=(0.0, 0.30); take 600 steps (one cycle); $\Delta t=(\pi/600)$. 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:

$\rho = \dfrac{\sum(s_d-\overline{s}_d)(s_T-\overline{s}_T)}{\sqrt{\sum(s_d-\overline{s}_d)^2\sum(s_T-\overline{s}_T)^2}}$	s_d and s_T here refer to the finite difference and true solutions for the scalar field "s"

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

Advection schemes: you are using *Lax-Wendroff*, *Takacs* and (perhaps) 6th-order Crowley methods, which are all 2-time-level and 1-D. Both use directional splitting, X followed by Y-advection. Note that Takacs needs 2 ghost points, and Crowley needs 3. Apply them in 2-D by first doing advection in x (for all rows), and then y-advection for all columns (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.

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

- You need 2 two-dimensional scalar arrays of size *(nx,ny)* and named *s1* and *s2* for the scalar being advected. Include ***2 ghost points*** *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 velocity components.
- *Confirm that your initial conditions are OK **first*** before proceeding further!

Coding requirements for this problem:

- Do *not* simply add Takacs code to your advection routine! Instead …
- Copy your 1D advection code file to a new "advect1d" file; add *1D Takacs* code inside the advect1d routine. "advection" calls *advect1d* to do the work!
- **Do not** (in C) assume point 0 as single ghost point, etc … use I1, I2 notation!!
- Pass *staggered* u & v data to *advect1d*. Averaging of u, v is done *inside* advect1d.
- 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:

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