Computer Problem 2
2D Advection • Rotational flow

Due: 9 PM, Fri. Mar. 14
Turn in: your code, statistics and plots (submitted on Canvas).

Problem being solved: 2-D linear advection via fractional step (directional) splitting
Evaluation: Compute Takacs (1985) error statistics at end of run
Boundary conditions: 0-gradient (copied from grid boundary) in both directions. If you need s(j-1) or s(j+1) near a boundary, use the boundary value (for x *and* y).

Initial conditions (IC): "cone" scalar 2-D field ***s***
Flow field: one case, constant with time: rotational flow (counter-clockwise)
Settings: Cone center location is (0.0,0.3) • cone radius r = 0.120 • take 600 steps
⇒ *We are using C-grid staggering: the location (x,y) of variables **s**, **u**, and **v** differ here!*

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

Methods: There are three: Lax-Wendroff, Takacs, and 6th-order Crowley.

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 ν > 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	*See Tremback p. 542, ORD=6 (advective form)*

You need *two* ghost points to accommodate Takacs' method; ***three*** for 6th-order Crowley. So: allocate and apply **three** ghost points in all cases; Lax-W/Takacs just won't use make use of all three ghost points. In C, set BC_WIDTH to 3 and add definitions for J1, J2. For coding with Fortran, set (e.g.) real s1(-2:nx+3,-2:ny+3) .

Domain: In moving to two dimensions we are also switching to a *staggered C-grid*, with the x,y location for the scalar field ***s*** ranging from -0.5 to +0.5 and the grid spacing Δx = Δy =*1.0/real(nx-1)*. The ***u*** and ***v*** wind field components vary in space but are *time-invariant* and thus need no ghost points. In C-grid staggering, the physical location for ***u(i,j)*** is ½Δx to the left ("west") of ***s(i,j)***, and ***v(i,j)*** is located ½Δy below ("south" of) ***s***. Due to staggering, **u** is dimensioned *(nx+1,ny),* and **v** is dimensioned *(nx,ny+1)*.

If you observe asymmetry (discussed below) in your results, the *most likely cause* is a problem in the initial conditions – probably the X, Y coordinates used in creating the initial conditions. Tests such as run here are valuable for identifying any problems in the initial condition, boundary condition or advection schemes, which is why we use them.

Settings

- Read in: the numerical method to use (Lax-Wendroff, Takacs, or Crowley-6).
- I.C. details + time step: provided on web site. For our rotational flow, the exact final solution = the initial condition (IC) since we complete one rotation.
- Error analysis: *put these computations in a (sub)routine, **not** the main program.* Compute error stats for each of your rotational flow *final* solutions – *after* one full rotation - following Takacs (1985). Print total, dissipation and dispersion error **to 5 decimal places**. Compute total error with Takacs' eqn. 6.1, and dissipation and dispersion errors from eqns. 6.6 and 6.7. For *linear correlation coefficient* ρ; use:

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

Advection schemes: your numerical methods are all 2-time-level, and 1-D. All use directional splitting, X followed by Y-advection. 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). Set your boundary conditions *before* x-advection *and* after x / before y-advection. We will stick with the sequence *x-advection*, *y-advection*, *x-*, y-…

>> *Confirm that your initial conditions are OK **first*** before proceeding further!

Coding requirements for this problem:

- **Do not** (in C) use point 0 as 1 ghost point, etc … use I1, I2, J1, J2 notation!!
- Pass *staggered* u & v data to *advect1d*. Averaging of u, v is done *inside* advect1d.
- Put your Takacs statistics calculations in a separate routine, not the main program
- Move your Pgm1 1-D advection code file to a new "advect1d" code file; add *1D Takacs* and 6th-order Crowley code inside *advect1d*. "advection" calls *advect1d()*.
    - Your Makefile must be changed to incorporate this new routine (discuss)
    - C programmers must add an advect1d() function prototype in *advection.c*
    - F90 programmers must add an INTERFACE block inside *advection.f90*
- Put 2D advection in a new *advection.f90/advection.c* file, not the main program
    - The main advection routine will now handle x- vs. y-advection passes, calling your "advect1d()" routine for each pass – each row or column.

Hand in:

- Plots: contour and … *(if possible… if it works!)* 3D surface plots of the initial condition and, for **each** method, the final solution at 600 steps. Also, for each method, plot $s_{min}(t)$ and $s_{max}(t)$ to document the time series behavior.
- Print and submit the Takacs error data to **5** decimal places for your final solutions. Note: the sum of dissipation+dispersion errors should match your total error.