Computer Problem 4
2D Automatic Nesting

Due: 2:00 p.m. Wed., March 27 – *after Spring Break*

Turn in on Moodle (*no* paper plots): a MS Word or OpenOffice document with all plots and statistics in it (import plot images into it); also your code (an *archive file*).

Problem being solved: Linear advection, 2-D, *using automatic grid nesting*.

Initial conditions: Scalar: Cone. Flow field: time-invariant rotational flow from program 2. There are 2 runs: *nest-feedback* and *no-feedback*.

Physical domain: same as programs 2 and 3: 2-D, [-0.5:+0.5 for scalar *s*] in X,Y.

Method: Directionally-split Lax-Wendroff (only), with and without *nesting*.

Input: Read in; do **not** set ("hardcode") in your program:

(1) Nest refinement ratio (integer ratio used to scale Δx and Δt for nest)
(2) Radius of the cone
(3) Number of time steps for integration of the coarse grid.
The time step Δt will be set to $\pi/(number\_of\_steps)$.
(4) Nest movement interval – how often we recompute nest position and move it
(5) Feedback from nest to coarse grid: on or off
(6) Plotting interval (as number of steps e.g. every 5 steps, or every 600)

Evaluation: For each run: Compute Takacs-type errors (total, dissipation and dispersion, list to ***6*** decimal places) for the coarse grid at end of run. Also show the ratio of total error for *no-feedback* divided by total error for the *feedback*-case.

Plotting: Submit plots (images in one Word/OpenOffice document) for the following –

(1) Initial condition: contour plots of coarse grid *u* and *v* with contour interval 0.1 (staggered: *not* averaged to *s* locations), and the scalar *s* field (contours *and* surface; contour interval 0.5)
(2) Coarse grid solution: contour + surface plots *at end of run*; use cntr() routine to *show location of nest* if feedback being done
(3) Nested grid solution (only if feedback being done): contour plot when the nest is first placed, and at the end of the run
(4) Each case: Compute and plot (contours *and* surface) the 2-D "max *s*" field (max at each grid point for all time steps) at end of each run

Parameter	Standard (i.e. not extra credit or test) settings
Grid, time steps	109x109; take 500 steps, with Δt = π/500
Scalar specification	Initial center (x,y) 0.0, 0.3; cone radius 0.075
Nesting	**3:1** ratio, update (move nest) every 5 time steps
Boundary conditions	**Coarse** grid: 0-gradient as before; **Nest**: from coarse grid.

Checking your results: I have provided a solution and error computations for a case somewhat different than the above, so you may check your code.

Code layout: For full credit **you *must* follow the programming guidelines** mentioned here and earlier:

- Your advection routine **must** be separate from the main program and **must** prepare 1-d arrays for use by a separate 1-d advection routine as part of the integration. You will have to pass a variable telling the advection routine whether it is handling the coarse grid or nest, because you compute s^{n+1} for *1…nx* on the coarse grid, but only from *2…nx-1* for the nest. *Pass* the coarse or nested grid arrays to advection. Do (always) x-advection followed by y-advection, as before.
- Use a boundary condition (BC) routine to handle nested *and* coarse grids. We are *not* doing time interpolation for nested grid boundaries in this problem.
- You will implement a separate time step loop for the nest, including: calls to (a) the boundary condition routine and (2) the advection routine and (3) the update step within this nest time loop.

Note: set boundary conditions *before* calling advection for coarse or nested grids.

Nest feedback: For feedback cases, nest values are interpolated back to the coarse grid for all but boundary (j=1 and j=nx) points, as discussed in class. I have *provided the routine for the interpolation* (to create or move the nest), nest boundary conditions, and feedback (to update the coarse grid) on the class web site at ~tg457444/502/Pgm4. Use the *nestwind* routine (also provided) for interpolated nested grid wind fields for advection on the inner grid.

Nest placement, relocation: The initial nest is placed at the start of the first time step and is re-evaluated at the start of every 5 time steps beginning with n=5. Nest placement is based on truncation error from the modified equation for 1-D Lax-Wendroff. Do truncation error computations in a separate routine, using the following:

$$Error_{trunc} = \left| \frac{\partial^3 s}{\partial x^3} (\mu^2 - 1) \frac{u(\Delta x)^2}{3!} \right| \text{ and } \frac{\partial^3 s}{\partial x^3} = \frac{s_{j+2} - 2s_{j+1} + 2s_{j-1} - s_{j-2}}{2(\Delta x)^3}$$

The procedure is:

1. Pass the u- and v-wind components averaged to *s* locations, and the scalar *s* field, to your new truncation error function/subroutine.
2. Compute the *(absolute value of the)* truncation error at all interior grid points, for *i=3,…,nx-2* and *j=3,…,ny-2 (set 0 elsewhere)*. Store the max of [truncation error for X- and Y- directions] at each point, and determine the peak for entire domain.
3. Scan through your data to determine I and J limits encompassing columns and rows where the maximum truncation error is ≥ 50% of the overall peak error.
4. Average these I and J bounds to determine the new *center* of your nested domain.
5. Determine the nest left, right, top and bottom locations by subtracting or adding ½ of the nest size (in coarse grid coordinates) to the new nested grid center location. If your nest would extend off any edge of the coarse grid, set the edge at the coarse grid boundary and the remainder of the nest extending from there.
6. When you are determining the new position for the nest, remember to first store the prior nest position in separate variables so you may then pass the old *and* new nest positions to the interpolation routine, so the nest may be moved correctly.