Computer Problem 1

One-Dimensional Linear and Nonlinear Advection

Due: 4pm Friday, January 25.

Turn in:
- Your plots, printed out - handed in
- Your code, submitted via our Moodle site (required)

Problem being solved: 1-D advection (transport) equation for a variable *q*.

Linear cases: $\frac{\partial q}{\partial t} = -c\frac{\partial q}{\partial x}$	Nonlinear case: $\frac{\partial q}{\partial t} = -q\frac{\partial q}{\partial x}$

The only difference above is use of (the constant) ***c*** vs. ***q*** before $\partial q / \partial x$.

Horizontal domain: your solution domain is a 1-D mesh *(grid)* of ***nx*** points. We solve the PDEs above for each point on this mesh.

Initial condition: a single sine wave with a wavelength of $nx \bullet \Delta x$.

Boundary condition: periodic; the wave "exits" the right side of the domain and "re-enters" the left side (for speed $c > 0$).

Scheme: The scheme (numerical method) you will use to solve these PDEs is called *Lax-Wendroff*, with forward time differencing, and centered space differencing. This scheme is commonly written:

$$q_j^{n+1} = q_j^n - \frac{\nu}{2}\left(q_{j+1}^n - q_{j-1}^n\right) + \sigma\left(q_{j+1}^n - 2q_j^n + q_{j-1}^n\right)$$

where:
- ***q*** is the scalar field that is being advected by the numerical method
- ***n*** is the time level, where ***n*** is "now" and ***n+1*** is the next time step.
- ***j*** is an index representing each of the ***nx*** grid points
- ν is called the "Courant number" and $\sigma = \nu^2/2$.
- ν is set to $(c\Delta t/\Delta x)$ in the linear case, and $(q_j^n \Delta t/\Delta x)$ otherwise, i.e. the local velocity value q(point *j*, time *n*) replaces the constant "c"

Cases, and **Settings:** There are *3 cases*. Use the following settings in your code:
- Phase speed ***c*** = constant = 1.0
- Grid size ***nx*** = 75
- Grid spacing $\Delta x = 0.1$
- Time step Δt determined from ν

Case	**Advection**	**Time step Δt**	**Courant number ν**	**Run for...**	**Look for ...**
A	Linear	0.05	0.5	150 steps	A good solution.
B	Linear	0.105	1.05	*Try 150 steps..*	Instability: it blows up
C	**Non**linear	0.05	Enter "1"	150 steps	Shock; damping

Time steps: Are given above. Note that the Courant number is constant in the linear cases, but in the nonlinear case it varies locally depending on the value of the variable q.

How it works: You are simulating the movement of a 1-D sine wave using a "grid" of 75 points. To move this wave, you integrate the PDE on the previous page by taking a series of ***time steps***. During each step you will, for all points j = 1...nx, compute the future time step (n+1) values given the known values at the present time (n). After this time step is complete, you replace all (n) array values with the (n+1) results, before starting the next time step; repeat until done! You will therefore use **two data arrays**, one to hold the *current* time step values, and one for *new* (predicted) results. You will also have to enforce some ***boundary conditions*** prior to each time step. We will discuss this in class.

Required**:**

- **Prepare and submit your code** – to do so,
    1. "make archive" to create a *pgm1.tar* archive containing all your code.
    2. "Mail -a pgm1.tar your-email-address" to send yourself the archive. *after entering a subject, hit return and Then type control-D to send it.*
    3. upload *pgm1.tar* on Moodle.
- *Only* if Moodle is down: send your archive as an email attachment to me.
- **The code already makes plots.** Plot and hand in the solution *at the end* of each run, ***or*** when any value of your array is greater than or equal to +/-1.5. The code halts if the solution is blowing up; this *will* happen in case ***B***!!!
- **Plot** a time series of maximum absolute value of q versus time step.
- **Plot** your initial condition, which is the same for all cases.
- You will *hand in* a total of 7 plots. See *Supercomputing on Stampede* to make images from your plots.

Demo code: A **demonstration program** (in Fortran and C) will be placed in my home directory (named "tg457444" for historical reasons) on Stampede. To get it:

cp ~tg457444/502/Pgm1/Fortran/ .* (Fortran 90)
cp ~tg457444/502/Pgm1/C/ .* (for C code)

The code contains a "Makefile" with which to compile the code, creating a text listing, or to make an archive. *make p1* compiles it; *make listing* creates the listing file; *make archive* creates the archive file mentioned above.

This program has most of the code needed for this assignment, including plotting. The only changes you need to make:

a. Put your name at the top of the code program (pgm1.f90 or pgm1.c)
b. Change the # of grid points, ***nx***, to 75; AND insert the correct boundary code.
c. Insert the Lax-Wendroff integration code for linear and nonlinear cases.

Testing your code

1. Test results will be put online for a slightly different nx, courant number etc.
2. Cases ***A*** and ***B*** are being run one cycle (or 'revolution'), to arrive at the starting point. ***A*** will provide a nearly perfect solution – looks like the initial condition.
3. Case ***B*** will "blow up" *before* 150 time steps have passed.
4. Case ***C*** develops a sharp gradient in the middle and decays – not at all like ***A*** or ***B***.