5 Problem set #5: 2-D Initial Value problems

Welcome to the next dimension... let the fun begin. Just because I'm sure you need the practice, there are **two** problems for this problem set. They're a bit dull but this homework is really a preliminary step to doing an interesting convection problem which will finish everything up in problem set 6. So don't throw away your solutions to question 2. Here we go....

1. **2-D Advection schemes–a methods sampler**: Here we will play with several methods for solving the simplest advection equation in 2-D

$$\frac{\partial T}{\partial t} + \boldsymbol{\nabla} \cdot \mathbf{V} T = 0 \tag{5.1}$$

where T is the advected quantity and $\mathbf{V}$ is the transport velocity field which we will assume to be incompressible (i.e. $\boldsymbol{\nabla} \cdot \mathbf{V} = 0$). In fact this problem will investigate the two analytic incompressible flow fields in the handouts: *Rigid body rotation* about a point (x_0, y_0)

$$\mathbf{V}(x, y) = (y - y_0)\mathbf{i} - (x - x_0)\mathbf{j} \tag{5.2}$$

and a single shear cell on the unit square $0 \leq x, y \leq 1$.

$$\mathbf{V} = -\sin(\pi x)\cos(\pi y)\mathbf{i} + \cos(\pi x)\sin(\pi y)\mathbf{j} \tag{5.3}$$

   (a) derive (or simply write down) the finite difference updating schemes for (5.1) and the following approaches

      i. non-conservative staggered leapfrog
      ii. conservative staggered leapfrog
      iii. simple upwind donor-cell
      iv. semi-lagrangian schemes (just abstractly)

   (b) A key component of semi-lagrangian schemes is the ability to do polynomial interpolation on a regular grid. Here we will just work out the interpolating weights in 1-D for different orders of the polynomial (see section 4.2 on transport, Eq. 4.20). Consider a function `f(i)` tabulated on the 4 equally spaced points $i = 1, 2, 3, 4$ with a point that lies between points 2 and 3 at coordinate $i = 2.25$. If we define the interpolating function of order n as a set of weights of the functions at the nodes

$$P_n(x) = \sum_{i=m}^{m+n} w_i f(i) \tag{5.4}$$

   find the weights for

i. $P_1(x = 2.25) = w_2 f(2) + w_3 f(3)$: linear interpolation between points 2 and 3

ii. $P_3(x = 2.25) = w_1 f(1) + w_2 f(2) + w_3 f(3) + w_4 f(4)$: cubic interpolation using all four points.

(c) Digital dump time! Just for your edification, I've concocted yet another set of example programs that implement the above algorithms. These programs solve for the evolution of a gaussian initial condition in the two flow fields given above. To get going and play with these programs

i. download and unpack the code in probset5.tar.gz. Most of the code for this problems set will be in the subdirectory `f77`. (some slow example matlab code is also included though).

ii. change directory to `f77` and make all the programs using the utility script `makeall`
(see `~mm_ms/classprobs_sun5/makeall`. For debuggable but slow code use `makeall debug`. For optimized fast code use `makeall fast`. Here we will only use the programs `advectsl2` (conservative staggered-leapfrog) and `adsemi_lag01`, a 2-D semi-lagrangian solver.[4]

iii. study the main programs to see how they work (in particular, array handling in 2-D). Study the updating subroutines `upsl2.f` and `upsemlag2d00.f` to see how the algorithms are implemented.

iv. For staggered-leapfrog and semi-lagrangian schemes, reproduce the figures in the handouts for 65^2 and 129^2 grids and the two different flow fields. Record the run-times.
There are two scripts to help you run and visualize these programs. `frun-cont.m` is a matlab function that runs, plots and makes postscript contour plots. `runmovie` is a ksh script to make animated gif movies and show them using `xanim`. Both functions takes a minimum of 5 arguments `program ngrid nframes irigid frac` where program is the name of the program to run, ngrid is the number of grid-points `irigid=1` is for rigid-body rotation, 0 is shear cell and `frac` is the fraction of the courant stability limit to use. For example to run the 65^2 semi-lagrangian, shear-cell test with 10 times the courant condition with 20 frames use `runmovie adsemi_lag01 65 20 0 1`. Study the script and understand what it does. Experiment with the fractional courant condition.

2. **Mixed Advection-Diffusion: a convection warmup**: The single shear cell is actually the simplest solution for steady state thermal convection (at low Raleigh num-

[4]For completeness I have also included a non-conservative staggered-leapfrog `advectsl1`, a 2-D MPDATA scheme `advectmp2` and a more general, time dependent semi-lagrangian scheme `adsemi_lag02`.

bers and for certain types of boundary conditions). In thermal convection, temperature is controlled by an advection-diffusion equation like

$$\frac{\partial T}{\partial t} + \mathbf{V} \cdot \boldsymbol{\nabla} T = \kappa \nabla^2 T \tag{5.5}$$

where κ is the thermal diffusivity. The flow field in full convection depends on lateral temperature gradients and therefore velocity and temperature become coupled (which is what makes it an interesting problem). We will not solve the full problem yet but will lead up to it by sorting out the advection-diffusion component using a steady-state shear cell as the imposed velocity field. This problem will also give us a bit of exposure to operator splitting and boundary conditions in 2-D (like you need this, I know…).

The program `adsemi_lag02.f` solves the dimensionless problem

$$\frac{\partial T}{\partial t} + \mathbf{V} \cdot \boldsymbol{\nabla} T = 0 \tag{5.6}$$

on a more general rectangular domain with a single shear cell with maximum velocity $|\mathbf{V}| = 1$. Modify this program to solve the dimensionless advection-diffusion problem

$$\frac{\partial T}{\partial t} + \mathbf{V} \cdot \boldsymbol{\nabla} T = \frac{1}{\mathrm{Pe}} \nabla^2 T \tag{5.7}$$

using an operator splitting combination of Semi-Lagrangian advection and FTCS diffusion and the following Initial and boundary conditions.

- initial conditions on dimensionless temperature $T(x, z) = 1 - z$ Where $z = 0$ is the bottom and $z = 1$ is the top of the box.
- Dirichlet boundary conditions on the top and bottom of the box $T(x, 0) = 1$, $T(x, 1) = 0$ and reflection sides $\partial T/\partial x(0, z) = \partial T/\partial x(x_{max}, z) = 0$.

To get you through this to the groovy movie at the end do the following

(a) Rewrite `initcon2.f` to implement the initial condition.

(b) Write down the update scheme for FTCS diffusion without advection i.e. the discrete form of

$$\frac{\partial T}{\partial t} = \frac{1}{\mathrm{Pe}} \nabla^2 T \tag{5.8}$$

Allow for $dx \neq dz$.

(c) What is the maximum stable time step for this scheme? Implement a check in the main program.

(d) Implement the FTCS diffusion scheme in a subroutine called something like `updifftcs1.f`. Don't forget to fix the boundary conditions. **include the source code in your homework**

(e) Now run your brand-new program for a problem with 129×65 grid points, $\mathrm{Pe} = 1000$, `tmax=10.` and `tplot=1.`

(f) choose any graphics package you like and show me plots for $t = 0, 5, 10$. For fun, use `mergegif` to make color movies.

(g) at a Peclet number of 1000, approximately how long (in dimensionless time) does this problem need to run to steady state.

(h) Consider the same problem with $\mathrm{Pe} = 10$. How many time steps does this require? How long does it take to get to steady state? Do you think it makes sense to pair a semi-lagrangian scheme with FTCS diffusion? Why or why not?