ESCI 445 – Numerical Modeling of the Atmosphere and Oceans
Programming Exercise #4

In these exercises you will write iPython notebooks to solve the advection equation,

$$\frac{\partial u}{\partial t}+c\frac{\partial u}{\partial x}=0,$$

the diffusion equation

$$\frac{\partial u}{\partial t}-K\frac{\partial^2 u}{\partial x^2}=0,$$

and the combined advection-diffusion equation,

$$\frac{\partial u}{\partial t}+c\frac{\partial u}{\partial x}-K\frac{\partial^2 u}{\partial x^2}=0.$$

Each program will use a different finite-difference scheme that we have discussed. In each of your programs you will use a grid of 601 grid points with a grid spacing of 1000 meters. You will use a time step of 30 seconds. Each simulation will run out to 250 minutes (500 time steps). The initial data will be created from the formula

$$u_j^0=\begin{Bmatrix} 0: & 0\le j\le 99 \\ 1: & 100\le j\le (100+w) \\ 0: & (101+w)\le j\le 600 \end{Bmatrix}$$

where *w* is the width of the signal.

I have created a shell notebook for the first exercise (‘exercise4-1-SHELL.ipynb’) that you can modify. You can then copy this and edit it for the remaining exercises.

For the advection equation: Allow the user to enter the speed of the signal. The programs you will write are:

1. Forward-in-time, backward-in-space
2. Forward-in-time, forward-in-space
3. Centered-in-time, centered-in-space (no filter) – Use an upwind scheme for the first time step.
4. Centered-in-time, centered-in-space with Asselin-Roberts filter.
5. 3rd-order Adams-Bashforth. – Use an upwind scheme for the first time step, and a leapfrog scheme for the second time step.

For the diffusion equation: Allow the user to enter the diffusion coefficient. The program you will write is:

6. Forward-in-time.

For the combined advection-diffusion equation: Allow the user to enter both the speed and diffusion coefficient. The program you will write is:

7. Centered-in-time, centered-in-space for advection, and forward-in-time for diffusion.

Output: The notebook automatically displays the initial and final values. It also saves the output to a .npy file. Using the Python program ‘View-1D.py’ you can load the saved data and view an animation of the results.