Contact me:
James Penn
jp492@exeter.ac.uk
University of Exeter

Introduction

The Python code `linearshallowwater.py` models the linearised shallow water equations on the beta plane.

The equations being solved are

$$\frac{\partial u}{\partial t} - fv = -g\frac{\partial h}{\partial x} \tag{1a}$$

$$\frac{\partial v}{\partial t} + fu = -g\frac{\partial h}{\partial y} \tag{1b}$$

$$\frac{\partial h}{\partial t} + H\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0 \tag{1c}$$

where fluid height $\eta = H + h$, g is gravity, and Coriolis parameter is $f = f_0 + \beta y$.

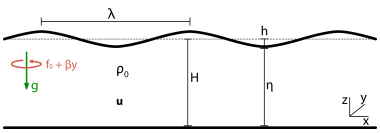

Figure 1: The Shallow Water Configuration

The partial derivatives in space are approximated by a first-order central difference method and timestepping is performed using the three-step Adams-Bashforth method.

The code can be run on either Python 2 or Python 3, the only additional requirements are the libraries `numpy` and `matplotlib`. If you have not used Python on your computer before, you are unsure if you have the libraries, or if it's just been a long time since you used Python I recommend using one of the scientific distributions that are available to download.

The two main choices are Enthought Canopy and Contiuum Anaconda. I use and recommend Anaconda; installing the latest version from the Continuum website [1] will give you the latest release of Python along with the most popular libraries [2].

A standard text editor such as Notepad is all that is required to edit the code, but not a *word processor* like Microsoft Word. While Notepad is sufficient, there are many more powerful text editors available that make writing code a much more pleasant experience. Atom [3] is a free, cross-platform option text editor with good support for Python.

[1] https://www.continuum.io/downloads

[2] Notably `numpy`, `scipy`, `pandas`, `matplotlib`, `sympy` and `ipython`. These six libraries will solve 99% of your numerical, statistical and dat-aprocessing needs. A full list of in-cluded packages can be found at http://docs.continuum.io/anaconda/pkg-docs.

[3] https://atom.io/

Experiments

Experiment 1: Geostrophic Balance in 1-D

The first experiment demonstrates geostrophic adjustment in one dimension. There is no variation in the y-direction; the system that is being modelled is

$$\frac{\partial u}{\partial t} = fv - g\frac{\partial h}{\partial x} \tag{2a}$$

$$\frac{\partial v}{\partial t} = -fu \tag{2b}$$

$$\frac{\partial h}{\partial t} = -H\frac{\partial u}{\partial x} \tag{2c}$$

Given some initial conditions and sensible values for g, H and f the system will be solved numerically. The intial conditions and constants are set

$$L_x = 2\times10^7 \qquad H = 100 \qquad f = 1\times10^{-5} \qquad g = 1$$

$$u_0 = 0 \qquad v_0 = 0 \qquad h_0 = \tanh(100\cdot x/L_x)$$

The code does not assume any units, but here the choice of constants scales the problem to be in [m] and [s].

and the code is allowed to evolve in time, taking a timestep $\Delta t = 1000$.

- Download the python script `linearshallowwater.py` from the course website [4]. **If at any point you have changed several things, something goes really wrong and you can't get the code to run anymore, just go back to the course website and download a new copy. You are encouraged to experiment and try new things!**

[4] http://empslocal.ex.ac.uk/people/staff/gv219/ecmm719/index.html

- Open a terminal (Mac/Linux) or command prompt (Windows), change directory to the location of the saved file and run it with the `python` command.

```
$ cd Downloads
$ python linearshallowwater.py
```

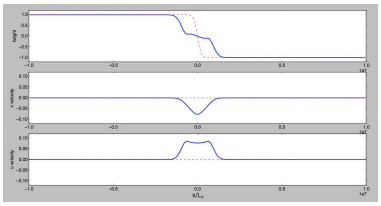

Figure 2: A stepped height field undergoing geostrophic adjustment.

- The code should begin to run and an animation window will appear (Figure 2). If you can't see the chart check:
  - That it is not hidden behind another window. Sometimes the python plotting window can be opened behind your terminal or text editor window.
  - That there are no errors on the command line. If there are and you're not sure how to fix them, let me know.
- Once you've got the code running, try changing some of the constants and see what happens.
  - Set $f \gg 1 \times 10^{-5}$ and $f = 0.0$.
    1. What is happening in each of these cases?
    2. What happens to the gravity waves that radiate outwards?
    3. What is the problem with this numerical model compared with the analytic solution? *Hint: boundary conditions.*
  - If you want to see the simulation run longer, increase `nsteps` (found at around line 414).
  - (Advanced) The initial conditions for this experiment are set around line 315. Try using a condition

$$h_0 = \cos^n(\pi \cdot x / L_x)$$

    and increase n to get a more isolated hump.

Experiment 2: Geostrophic Adjustment in 2-D

We'll now consider the two-dimensional case where derivatives $\frac{\partial}{\partial y} \neq 0$ in general. The system (1) will be solved in full.

- Make the following changes and run the code from the command line.

```
experiment = '2d'     # line 32
f0 = 0.0              # line 49
```

- You should see the case for a non-rotating fluid begin to run on your screen. The height field is set with an initial condition of a small gaussian added to one point in the field. As the equations evolve u and v velocities are induced and gravity waves radiate outwards from the initial disturbance.

  1. Where on the Earth is $f_0 = 0$?
  2. (Advanced) What speed do you expect the gravity waves to be propagating at in this model?

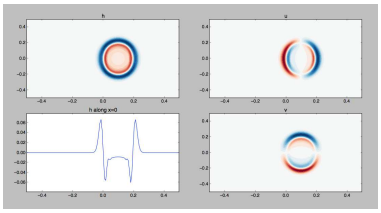

Figure 3: Gravity waves propagating away from an initial disturbance.

- Now try changing the Coriolis parameter to be something similar to that experienced in the mid-latitudes.

  1. Change the value of f_0 back to 1×10^{-5} and rerun the two-dimensional case to compare with the irrotational version.
     (a) What happens to the gravity waves?
     (b) What happens to the initial condition?
  2. Increase f_0 further. What happens to the amplitude of the gravity waves as f_0 increases?

Advanced Experiments

The code is quite versatile and capable of creating some other interesting dynamics. Here are a few more experiments you can try. [5].

[5]

- Coastal Kelvin Waves on the East Pacific coast.

```
Ly = 2.0e7                      # line 42
boundary_condition = 'walls'    # line 44
f0 = 0.0                        # line 49
beta =  2.0e-11                 # line 50
```

  You may need to increase `nsteps` to see a longer simulation.

  As the initial height field decays, an *equatorial Kelvin wave* moves Eastward. When it reaches the boundary, it becomes a *coastal Kelvin wave*.

- Equatorial Kelvin and Rossby waves.

```
boundary_condition = 'periodic'    # line 44
f0 = 0.0                           # line 49
beta =  2.0e-11                    # line 50
gr =  1.5e6                        # line 310
```

`gr` specifies the radius of the disturbance in the height field. Here it is made bigger to be of order of the *Rossby Deformation Radius* $L_d = \frac{\sqrt{gH}}{\beta}$. An equatorial Kelvin wave travels eastward along the equator, while two, slower moving, Rossby waves move westward in the tropics.

Appendix A: Numerics

This is a *very* brief overview of the numerics used to implement the model. Dale Durran's book [6] is an excellent resource for learning more about numerical methods in GFD.

[6]

If you are interested in using this code to model a specific problem, there is an up-to-date customisable version online [7]. This repository has both the linear model and a fully nonlinear model of the shallow water equations.

[7] https://github.com/jamesp/shallowwater

Spatial Derivatives

The horizontal domain is discretised into distinct nodes x_j separated by a constant distance $\Delta x = nx/L_x$ in the x-direction. Denote the value of $u_j = u(x_j)$ then the derivative at point x_j can be approximated by the *central difference* considering only the values of u at neighbouring nodes

$$\frac{\partial u_j}{\partial x} \simeq \frac{u_{j+1} - u_{j-1}}{2\Delta x} \tag{3}$$

For improved numerical stability, the locations of u, v and h values are staggered in a configuration known as the *Arakawa-C grid* (Figure 4) [8]. This distribution of value nodes means that, the *central difference* approximation to the derivative is given by, for example $\frac{\partial h}{\partial x}$:

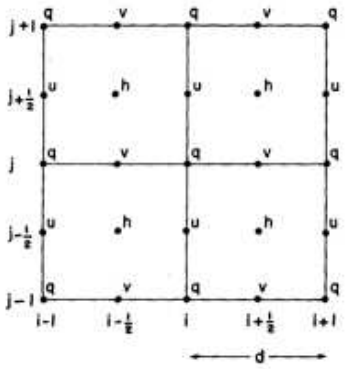

Figure 4: The Arakawa-C grid. From [?]

[8]

$$\frac{\partial h}{\partial x} \simeq \frac{h_{i+1/2} - h_{i-1/2}}{\Delta x} \tag{4}$$

and falls at the location of the u_i node. This is a desirable of the numerical scheme in the linearised shallow-water equations where $\frac{\partial h}{\partial x}$ is needed at the u_j nodes only. The same is true for the other spatial derivatives in (1).

Time Derivatives

The time stepping forward is achieved using a linear-multistep method. Once the spatial derivatives have been approximated using the central difference method, the system can be considered an initial value problem of the form $\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y)$. The three-step Adams-Bashforth multistep time integration method is given by [9]

[9]

$$y_{n+3} = y_{n+2} + \Delta t \left(\frac{23}{12} f(t_{n+2}, y_{n+2}) - \frac{4}{3} f(t_{n+1}, y_{n+1}) + \frac{5}{12} f(t_n, y_n) \right) \tag{5}$$