PyRsw Documentation

Release 0.1

PyRsw Team

November 17, 2015

Contents:

CHAPTER

ONE

PLAN

List or problems to solve

1. solve 1d 1-layer SW (periodic/boundaries)
2. solve 1d n-layer SW
3. solve 2d 1-layer SW
4. solve 2d n-layer SW
5. Linear Stability Calculations

Equations and latex to include:

1. derivation of SW
2. governing equations for different models (see above)
3. different numerical methods:
   (a) FD Sardouney
   (b) spectral
   (c) WENO
   (d) f2py
   (e) openmp
   (f) openmp + mpi

CHAPTER

TWO

ONE-LAYER ROTATING SHALLOW WATER MODEL

2.1 Classical Form

The one-layer, two-dimensional rotating shallow water model on a rotating f-plane can be written as,

$$\begin{aligned}\frac{\partial u}{\partial t} + (\mathbf{u}\cdot\nabla)\, u - fv &= -g\frac{\partial h}{\partial x},\\ \frac{\partial v}{\partial t} + (\mathbf{u}\cdot\nabla)\, v + fu &= -g\frac{\partial h}{\partial y},\\ \frac{\partial h}{\partial t} + \nabla\cdot(h\mathbf{u}) &= 0.\end{aligned}$$

This describes the motion of a pancake like fluid in that it is thin in the vertical and much longer in the horizontal. It contains pressure forces due to the free surface and a Coriolis pseudo-force because of the rotating frame of reference.

The fluid moves as columns that can be translated in the horizontal and stretched/contracted in the vertical. If the height of a column changes then the voriticty must change, as can be reflected in the fact that, in the absence of forcing and dissipation, Potential Vorticity is conserved following the motion,

$$\frac{D}{Dt}\left(\frac{\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} + f}{h}\right) = 0.$$

2.2 Conservation Form

There are a variety of forms, these equations can be written. One of them is conservation where the three fields that have time derivatives are $U = hu$, $V = hv$ and h.

$$\begin{aligned}\frac{\partial U}{\partial t} + \frac{\partial}{\partial x}\left(\frac{U^2}{h} + \frac{gh^2}{2}\right) + \frac{\partial}{\partial y}\left(\frac{UV}{h}\right) - fV &= 0,\\ \frac{\partial V}{\partial t} + \frac{\partial}{\partial x}\left(\frac{UV}{h}\right) + \frac{\partial}{\partial y}\left(\frac{V^2}{h} + \frac{gh^2}{2}\right) + fU &= 0,\\ \frac{\partial h}{\partial t} + \nabla\cdot(h\mathbf{u}) &= 0.\end{aligned}$$

Conservation form is attractive because it can help to ensure that, using a clever numerical scheme, some quantitites are conserved. Note that if you have topography then there are source terms that appear on the right-hand side and this causes some problems.

2.3 Vorticity-Bernoulli Form

A third form arises from rewriting the nonlinear acceleration terms as a gradient and a cross product term. Using a vector identity, it can be shown that the above system is mathematically equivalent to the following,

$$\begin{aligned}\frac{\partial u}{\partial t} - qhv &= -\frac{\partial B}{\partial x},\\ \frac{\partial v}{\partial t} + qhu &= -\frac{\partial B}{\partial y},\\ \frac{\partial h}{\partial t} + \nabla\left(h\mathbf{u}\right) &= 0.\end{aligned}$$

Note that above we have defined the vorticity and the Bernoulli function,

$$\begin{aligned}q &= \frac{\zeta + f}{h} = \frac{\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} + f}{h},\\ B &= gh + \frac{1}{2}\left(u^2 + v^2\right).\end{aligned}$$

Before we look at solving this complicated set of equations we consider the one and a half dimensional limit.

2.4 $1\frac{1}{2}$ Dimensional Limit

We assume that none of the variables depend on one horizontal direction, say y. But, it is very important to realize that the velocity in that direction is not necessarily zero. Indeed, if you have flow in the x-direction, the Coriolis force will deflect it to the right, which will then generate a flow that is perpendicular. This will continue and often give rise to inerital oscialltions in the horizontal. So we can have motion in either direction but the motion only changes with respect to x.

If we simply the governing equations we get

$$\begin{aligned}\frac{\partial u}{\partial t} - qhv &= -\frac{\partial B}{\partial x},\\ \frac{\partial v}{\partial t} + qhu &= 0,\\ \frac{\partial h}{\partial t} + \frac{\partial}{\partial x}\left(hu\right) &= 0.\end{aligned}$$

where the vorticity and the Bernoulli function simplify to,

$$\begin{aligned}q &= \frac{\zeta + f}{h} = \frac{\frac{\partial v}{\partial x} + f}{h},\\ B &= gh^2 + \frac{1}{2}\left(u^2 + v^2\right).\end{aligned}$$

2.5 Conserved Quantities

In the purely conservative or nondissipative limit, there are three quantities that are exactly conserved.

1. Mass:

$$M = \int_D h\, dA$$

2. Total Energy: sum of potential and kinetic energies

$$E = \frac{1}{2}\int_D \left(gh^2 + h(u^2 + v^2)\right)\, dA$$

3. Potenal Enstrophy:

$$Q = \frac{1}{2}\int_D hq^2\, dA$$

In a numerical model we cannot expect these to be conserved but we would like them to be close to be conserved. If they are very badly conserved than this could reflect that the numerical scheme is behaving badly. However, just because these are conserved that does not guarantee that the solution is correct. But they are usually good indicators as to how our method is doing in the conservative limit.

Of course when nonconservative forces are introduced things will change. One might argue that since the world is non-dissipative then we don't need to worry about conserving these. However, it is desirable to know that basis of your model is well behaved and therefore why we should worry about conserved quantities.

CHAPTER

THREE

EXAMPLES

3.1 Geostrophic Adjustment: 1D and 1L

In the directory `src` you will find an example entitled `example_1D_1L_spectral.py`

First, libraries are imported. Two standard ones are numpy, for calculations, and matplotlib.pyplot for plotting. Those are standard to numpy. Then, there are four other things that are imported:

- **Steppers** This contains different time-stepping functions. At the moment we have Euler, Adams-Bashforth 2 (AB2) and Runge-Kutta 4 (RK4). PyRsw uses adaptive time stepping to try and be more efficient in how the solution is marched forward.
- **Fluxes** This contains the fluxes for the RSW model. At the moment there is only the option for a pseudo-spectral model but this will be generalized to include a Finite Volume method as well.
- **PyRsw** This is the main library and importing Simulation imports the core of the library.
- **constants** This has some useful constants, more can be added if desired.

After the libraries are imported then a simulation object is created.

```
sim = Simulation()
```

Below specifies the geometry in x and y: [Options 'periodic', 'walls']

We use AB2, a spectral method: [Options: Euler, AB2, RK4]

We solve the nonlinear dynamics (can be Linear)

Use spectral sw model (no other choices). Maybe hide this.

```
sim.geomx       = 'walls'
sim.geomy       = 'periodic'
sim.stepper     = Step.AB2
sim.method      = 'Spectral'
sim.dynamics    = 'Nonlinear'
sim.flux_method = Flux.spectral_sw
```

We specify a lot of parameters. There are some default values that are specified in .

```
sim.Lx   = 4000e3          # Domain extent                (m)
sim.Ly   = 4000e3          # Domain extent                (m)
sim.geomx = 'periodic'     # Boundary Conditions
sim.geomy = 'periodic'     # Boundary Conditions
sim.Nx   = 128             # Grid points in x
sim.Ny   = 1               # Grid points in y
sim.Nz   = 1               # Number of layers
sim.g    = 9.81            # Gravity                      (m/sec^2)
```

```
sim.f0  = 1.e-4           # Coriolis                    (1/sec)
sim.cfl = 0.05            # CFL coefficient             (m)
sim.Hs  = [100.]          # Vector of mean layer depths (m)
sim.rho = [1025.]         # Vector of layer densities   (kg/m^3)
sim.end_time = 36.*hour   # End Time                    (sec)
```

We can specify the periodicity of plotting and whether we want a life animation or make a video. More on this this later.

```
sim.output = False        # True or False
sim.savet  = 1.*hour      # Time between saves
```

Specify periodicity of diagnostics and whether to compute them. This is not tested.

```
sim.diagt    = 2.*minute  # Time for output
sim.diagnose = False      # True or False
```

Initialize the simulation.

```
sim.initialize()
```

Specify the initial conditions. There is an option whether we want the domain in x or y. At the moment there is no difference because there is no β-plane but this will be added.

```
for ii in range(sim.Nz):  # Set mean depths
    sim.soln.h[:,:,ii] = sim.Hs[ii]

# Gaussian initial conditions
x0 = 1.*sim.Lx/2.      # Centre
W  = 200.e3              # Width
amp = 1.                 # Amplitude
if sim.Ny==1:
    sim.soln.h[:,:,0] += amp*np.exp(-(sim.x-x0)**2/(W**2)).reshape((sim.Nx,1))
elif sim.Nx==1:
    sim.soln.h[:,:,0] += amp*np.exp(-(sim.y-x0)**2/(W**2)).reshape((1,sim.Ny))
```

Solve the problem.

```
sim.run()
```

Plot the Hovmöller diagram in time versus space.

```
if sim.Ny==1:
    plt.figure
    t = np.arange(0,sim.end_time+sim.plott,sim.plott)/86400.

    for L in range(sim.Nz):
        field = sim.hov_h[:,0,:].T - np.sum(sim.Hs[L:])
        plt.subplot(sim.Nz,1,L+1)
        plt.pcolormesh(sim.x/1e3,t, field,
            cmap=sim.cmap, vmin = 0, vmax = amp)
        plt.xlim([sim.x[0]/1e3, sim.x[-1]/1e3])
        plt.ylim([t[0], t[-1]])
        plt.title(r"$Hovm{\"o}ller Plot\, {of} \,\, \eta$")
        plt.xlabel(r"$distance \, \, (km)$")
        plt.ylabel(r"$Time \, \, (days)$")
        plt.colorbar()
    plt.show()
```

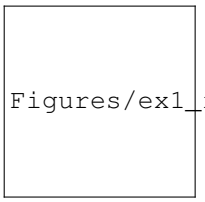

Fig. 3.1: Final solution for the test case.

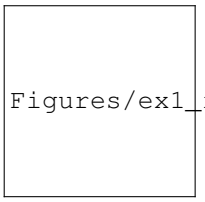

Fig. 3.2: Hovmöller plot for the test case.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search