1D advection exercises:

Download 1DAdvection_template.ipynb from class class materials of March 5.

1. Complete the code for advection in x direction using the if else format (a function using the and np.absolute(u) form is provided)
2. Run the code with u = 10.0 and dt=4.0s
3. Rerun the code by setting u[:,:,:] = 5.0
4. Rerun the code by setting u[:,:,:] = 12.0
5. Change the advection code to always use i-1 and i in the advection term and rerun the code by setting u[:,:,:] = -10.0 and -5.0.

2D advection exercises:

Download 1DAdvection_template.ipynb from class materials of March 5

Add similar codes for vertical theta advection. Set periodic boundary conditions in the vertical also, following the following code for horizontal boundaries:

```
theta[t+1,:,0] = theta[t+1,:,nx-2]
theta[t+1,:,nx-1] = theta[t+1,:,1]
```

1. Run the code by setting u[:,:,:] = 0.0 and w[:,:,:] = 10.0
2. Run the code by setting u[:,:,:] = 10.0 and w[:,:,:] = 10.0. Set dt so that the Courant number is no greater than 1/sqrt(2).

2D cloud model exercises:

Download 2D_Cloud_template.ipynb, HotBubble.nc and ColdBubble.nc from class materials of March 5 and upload to jupyter.lib.ou.edu.

This template has codes to specify 5 K hot bubble or -5 K cold bubble (initial theta perturbation).

a) You need to add codes to do advection for theta and vort (vorticity).
b) You need to add codes to calculate buoyancy vorticity generation term.
c) You need to add codes to solve stream function Poisson equation.
d) You need to add codes to calculate u and w from stream function.

Precalculated solutions can be read in (the template code is set up to read in u, w and stream function) from HotBubble.nc or ColdBubble.nc that contains solutions of all fields at every time step at dt = 2.0. For 5 K or -5 K bubbles respectively, do the following:

1. Run the code without any change and examine the animation.

2. Complete step a) above and re-run – examine the solution. Is the bubble rising/sinking now?
3. Add codes to calculate u and w from stream function read in from the *.nc file, and rerun. Is the solution changed?
4. Uncomment streamf[t+1,:,:] = poisson_fft(vort[t+1,:,:], invlapl) to call provided FFT Poisson solver to solve for stream function from predicted vorticity. Rerun – how is the solution?
5. Complete function poisson2d( ) to solve the Poisson equation using SOR method to be discussed in class and re-run.

Boundary conditions for the 2D cloud model

We assume a closed model domain of 2560 x 2560 m^2 in *x* and *z* directions.
The *x* boundaries are at *x* = 0 and *x* = lenx.
The *z* boundaries are at *z* = 0 and *z* = lenz.
The number of grid points in *x* and *z* are *nx* and *nz*, respectively.

d*x* = lenx/(*nx* - 1), d*z* = lenz/(*nz* - 1).

If using the FFT version of Possion solver, preferred values of *nx* and *nz* are $2^n + 1$.
Suggested choices for *nx* and *nz* are 65, 129 and 257 for low, medium and high resolution runs, corresponding to *dx*, *dz* = 40, 20 and 10 m. Remember, arrays of shape (*nx, nz*) have indices going from 0 to *nx*-1, and 0 to *nz*-1.
The domain has rigid wall boundaries, therefore velocity components normal to the boundaries are zero:

u(*x* = 0) = *u*(*x* = lenx) = 0,
w(*z* = 0) = *w*(*z* = lenz) = 0.

We also assume the velocity components parallel to the boundaries have zero normal gradient, therefore

w(*x* = 0) = *w*(*x* = *dx*)
w(*x* = lenx) = *w*(*x* = lenx - *dx*)
u(*z* = 0) = *u*(*z* = *dz*)
u(*z* = lenz) = *u*(*z* = lenz - *dz*).

For stream function, the boundary condition is $\psi = 0$ at all boundaries.
For voriticty η and potential temperature θ', we apply zero normal gradient boundary conditions at all boundaries.