ME712: Mesoscale Modeling

Class mesoscale model (CMM) project, assignment 4

Due at start of class on Tuesday, 14 November.
In–class supervised work period on Thursday, 2 November.
In–class non–supervised work periods on 7 and 9 November (Dr. Parker out of town).

We will be adding computational diffusion, Asselin filtering, and a Rayleigh damping layer to CMM. When you are done, the processes should occur in this order...

variables are updated using equations implemented in task 3
(microphysics routines are called... to be done later in course)
apply Rayleigh damping
apply computational diffusion
apply Asselin filtering
apply boundary conditions

1. Implement a Rayleigh damping layer (“sponge”) in the top part of the model. The basic formulation should have these components:

! compute local damping coefficient
coef = raydmpcoef*0.5 &
   *(1.-cos(trigpi*(zu(k)-raydmpz)/(zu(nz)-raydmpz)))

! apply sponge to slowly remove perturbations
up(i,k) = up(i,k) - coef*(up(i,k)-ub(k))

where I have used Fovell’s convention for $u_p$, and $ub$ is the base state $u$–wind (we will allow it to vary in the vertical in future exercises). The damping should be applied to all the predicted variables as a part of each time step, but only for physical points that fall within the damping layer. For starters, set your damping layer to extend upward from $raydmpz=12km$, and use a value of $raydmpcoef$ equal to 1/20 (it is unitless, so this value corresponds to the complete removal of a perturbation in 20 timesteps/applications of the filter).

2. Implement computational diffusion in the model. Recall from class that the basic formulation for diffusion is:

$$up(i,k) = up(i,k) + dtlocal*( \ &
+ kmixh*(um(i+1,k)-2.0*um(i,k)+um(i-1,k))/(dx*dx) \ &
+ kmixv*(um(i,k+1)-2.0*um(i,k)+um(i,k-1))/(dz*dz) )$$

where I have used Fovell’s convention for $u_p$, and $um$. This should be applied as a part of each time step, on all physical points. You should apply it to all predicted variables except $\pi'$. The mixing coefficients should be computed in your diffusion code as follows:

$$kmixh=cmixh*dx*dx/dt$$
$$kmixv=cmixv*dz*dz/dt$$

Initially, we will set $cmixh$ and $cmixv$ to be the same value, 0.005. Based on our current model set–up, this will yield $khdif=kvdif=400 \text{ m}^2/\text{s}$. The tunable parameters are $cmixh$ and $cmixv$, which
are measures of how close we are to the limit for stability (recall that it is 0.125, so we are starting at 1/25th of that value). Multiplying by $dx^2$ (or $dz^2$) and dividing by $dt$ then gives the coefficient proper units and ensures that it can be stably applied to any grid spacing.

3. Implement an Asselin filter in the model. Recall that the formulation for Asselin filtering is given by:

$$u(i,k) = u(i,k) + \text{asscoef} \times (up(i,k) - 2 \times u(i,k) + um(i,k))$$

where I have used Fovell’s convention for $up$, $u$, and $um$. This should be applied after each time step to all variables on all physical points. For starters, use a value of $\text{asscoef}=0.1$.

4. Rerun the task 3 simulations, turning only one of the three new features on at a time. Finally, rerun the model with all three features turned on. Make a plot corresponding to $t=1200$ s (i.e. the plots we made for task 3) for each of your four runs. Then, comment briefly on the impacts of each new feature that we have added. You may wish to animate your various simulations in order to observe different transient behaviors in the model.

5. Play around with the coefficients for the filters. Also, play around with the speed of sound in the model. Rerun the task 3 simulation and report on the nature of the experiments you’ve tried and any unusual behaviors observed.

6. Using the recommended settings (above in items 1–3) for the filters, rerun the task 3 experiment on a much wider domain (i.e. 400 points or so), and with a cold bubble instead of a warm bubble. In other words change the temperature perturbation of the bubble from $+3K$ to $-3K$. Run the model for 1200 s and make a plot of $\theta'$. Also, figure out how to make a plot of vectors and show the 2D velocity vector field (the $u$ and $w$ components). In GrADS, here is the method:

```bash
set gxout vector
display u;w
```