

Interpreting and analyzing model output (A *very* cursory introduction)

Here will talk briefly about using “ncview” and “matlab” to analyze output from your model. The model output is in netcdf format. The standard runs of the dynamical core often write out two files: `atmos_daily.nc` and `atmos_average.nc`. The former contains output written once each day, the later gives the mean values over the entire integration length. I use `ncview` to get a rough idea if an integration has run properly, and get a feel for the output. I use `matlab` to measure, plot, compute, etc..

Recall the the model output is stored in:

```
/scratch/nyuID/model_output/[simulation_ID]_d[final_date]
```

Simulation ID is the name you gave your run, say, `t42l20e_hs`, and final date is the final date of the run.

ncview

This application allows you to quickly view the data in netcdf files. It's not so useful for quantitative analysis, but gives you a quick overview of what you have. To start:

```
ncview [your file name].nc &
```

or, if you have copied over my aliases, you can just type:

```
nv [your file name].nc
```

This brings up a window that's somewhat self explanatory - click on the variable you'd like to see, and use the play buttons on the top to scroll around in tim. It's really just a way to browse the data and watch things evolve. Handy tips:

- 1) To view different axes, click “Axes.” The default is generally lat-lon, but you can request pfull - lat to see meridional cross sections.
- 2) Click “Inv P” to get pressure pointing upwards (or more generally, to flip the y-axis).
- 3) The color range will be optimized to fit the whole file, so for many plots, you’ll see nothing. To optimize the colorbar for a given view, left click range.
- 4) The << and >> buttons at the top let you animate along the time axis. To scroll through other axes (say pressure level, if you’re looking at lat-lon plots), right/left click on the boxes under “Current” in the bottom rows. (The value in these boxes tell you the level + date shown; the date is also shown at the top.)

matlab

I hope you have some experience with matlab. There are handy tutorials available online. The basic strategy for figuring something out is:

- 1) Try `> help [what you want]`. Sometimes this works!
- 2) Ask someone next to you.
- 3) Try googling it.
- 4) Work you way through the mathworks matlab site.

Now some specific advice for analyzing the GCM output. Recent versions (matlab 7 and higher) have built in commands to handle netcdf files, but they are a bit rudimentary. I’ve written a handy script that should do everything you need: `ncget.m`. It’s available from the models webpage, where you found this (and you may have already copied over this file when you first set up your model). Put this script in your `~/matlab/` directory. (If you don’t have a `~/matlab` directory, make one.) When you open any session of matlab, any script in this directory will automatically

be accessible. (If you're following my earlier directions for setting up the core, you may have this already.)

Try help `ncget` for advice on how this works. Here's what you'll see:

This function reads variables from a netcdf file.

```
usage: variable_out = ncget('input_file','variable_name',r1,r2,r3,r4)
```

See below for ways to get information about the netcdf file!

Inputs

required:

`input_file` = the name of the file to open; can include path
(e.g. `'/home/gerber/sample.nc'` or `'sample.nc'`)
`variable_name` = the variable to read (e.g. `'lat'`)

optional:

`r1,r2,r3,r4` the range of the variable to get. If no range is given, you'll get the entire variable. Make sure to ask only for dimensions available; otherwise you'll get an error. Specify the range like any matlab array. Some examples.

Suppose `'ps'` is 128 x 64 x 100 (lon, lat, time) and

1) You'd like just half of it (one hemisphere)

```
> ps_nh=ncget(input_file,'ps',:,:33:64,:);
```

2) You'd like just date 27

```
> ps27=ncget(input_file,'ps',:,:27);
```

3) You'd like every other longitude and latitude

```
> ps_course=ncget(infile,'ps',1:2:127,1:2:63,:);
```

4) There are many ways to ask for the entire variable `ps`:

```
> ps=ncget(input_file,'ps');
```

```
> ps=ncget(input_file,'ps',:,:,:);
```

```
> ps=ncget(input_file,'ps',1:128,1:64,1:100); etc.
```

Unfortunately this script can only handle up to 4 dimensional arrays.

Output:

The variable (and range) you requested.

To get information about a file, just give the `input_file` name

```
> ncget(input_file)
```

or

```
> ncget(input_file,'info')
```

It will write out all the variables, giving the size of dimensions, and the dimensions of the input variables.

For information on a specific variable, try

```
> out=ncget(input_file,variable,'info')
```

It will right out the size of dimensions and the dimensions of variables, and out will contain the size of the dimension/variable.

I've tried to think of anything you'd want to do with a netcdf file, and incorporate it into this script. I work through a concrete example below, to help you get started. I'll assume you've run the core and have an output file `atmos_daily.nc`. Remember that it's best not to use the first 200 to 300 days of the integration, as it needs to spin up from a cold start.

I'll suppose that the output file is here. (note that `epg2` is my nyuID)

```
/scratch/epg2/model_output/n45l24_hs_d01200/atmos_daily.nc
```

Recall that at this point we already know what this integrations is, and the final integration. (`n45l24_hs` tells me it's a finite volume core run with 45 latitudes between the equator and pole; `d01200` means it ends at day 1200. This particular run is 200 days long.)

Start up matlab where ever you'd like to work -- it doesn't need to be the directory where the file is located.

```
> matlab -nodesktop -nosplash
```

(I'm old school, and don't use the GUI -- GUI's a fine, though.)

To find out what's in the file, use `ncget`!

```
> ncget('/scratch/epg2/model_output/n45l24_hs_d01200/atmos_daily.nc')
```

The dimensions (lengths) and variables (dimensions) available in:

```
/scratch/epg2/model_output/n45l24_hs_d01200/atmos_daily.nc
```

```
phalf (25)
```

```
time (200)
```

```
lon (144)
```

```
lonb (145)
```

```
lat (90)
```

```
latb (91)
```

```
pfull (24)
```

```
bk (25)
pk (25)
zsurf (lon,lat)
ps (lon,lat,time)
ucomp (lon,lat,pfull,time)
vcomp (lon,lat,pfull,time)
temp (lon,lat,pfull,time)
omega (lon,lat,pfull,time)
vort (lon,lat,pfull,time)
```

We now see the dimensions (with their lengths, i.e. pfull, pressure at full model levels, has 24 values) and the variables (with their dimension, i.e., ps, surface pressure, is a function of lon, lat, and time, while omega, the pressure velocity, is a function of lon, lat, pfull, and time).

Say we'd like to plot the instantaneous vorticity in the northern hemisphere on day 1127 at 225 hPa, and the time mean, zonal mean, zonal wind. Here we make those two plots. I'll try to give some useful tips along the way. Recall that % means comments.

```
> infile='/scratch/epg2/model_output/n45124_hs_d01200/atmos_daily.nc'
> % get the axes
> lon=ncget(infile,'lon');
> lat=ncget(infile,'lat');
> pfull=ncget('infile','pfull');
> time=ncget('infile','time');
```

```
> % let's first make that vorticity plot, showing off the capabilities
> % of ncget. Find the pressure level corresponding to 225 hPa
> pfull
```

```
pfull =
    3.6503
   19.0884
   52.3402
   99.1299
  157.3810
  224.7492
  298.9444
```

```
378.0866
458.3251
533.7864
601.7172
662.5931
716.8593
764.9444
807.2677
844.2424
876.2772
903.7783
927.1495
946.7935
963.1113
976.5034
987.3693
996.1084
```

```
> pfull(6)
```

```
ans =
```

```
224.7492
```

```
> % the date we're looking for (this file has 200 days in it)
```

```
> time(127)
```

```
ans =
```

```
1127
```

```
> lat(46:90) % output not shown -- but this is the NH!
```

```
> % looks like we need the 6th pressure level and 127th time slot
```

```
> % and latitudes 46:90.
```

```
> % get some information on the vorticity variable
```

```
> ncget(infile,'vort','info')
```

```
vort (lon,lat,pfull,time)
```

```
ans =
```

```
144 90 24 200
```

```

> % The answer here is the size of the variable.

> % get just the data we need
> v225d1127=ncget(infile,'vort',:,:46:90,6,127);

> % and plot it (note that the ' means transpose)
> contour(lon,lat(46:90),v225d1127',20);
> set(gca,'fontsize',20)
> colorbar;
> xlabel('longitude');
> ylabel('latitude');

> % Now, the second plot!

> % Now, let's get that time mean, zonal mean wind.
> ucomp=ncget(infile,'ucomp')
> % when we don't specify range parameters, we get the whole variable.
> size(ucomp)

ans =

    144    90    24    200

> % take means
> u_time_ave=mean(ucomp,4);
> u_time_zonal_ave=squeeze(mean(u_time_ave,1));
> % squeeze gets rid of the empty first dimension

> size(u_time_zonal_ave)

ans =

    90    24

> contourf(lat, pfull, u_time_zonal_ave,20)
> colorbar;
> set(gca,'ydir','reverse'); % get pressure the right direction!
> % if you'd like to shift to log pressure coordinates, use this
> set(gca,'yscale','log','ytick',[10 30 100 300 900])
> set(gca,'fontsize',20)
> xlabel('latitude');
> ylabel('pressure (hPa)');

```

Some advice on plotting model output, that is, how to avoid typical “rookie” mistakes!

Caution with the colorbar!

If you want to compare output from two different simulations or time steps, make sure that you plot a fair comparison. Matlab will scale the colorbar to match the output by default, but this makes it hard to compare to plots, as the colors don’t correspond to the same. Find out the range of all the data first, and create a vector that delineates the colors you want to show. For example, say that you want to compare “data1” with “data2.”

```
> max(max(data1))
ans= 34.1
> min(min(data1))
ans = -12.1
> max(max(data2))
ans = 12
> min(min(data2))
ans = -23.1
```

Base on this, it looks like -25 to 35 would be a good range., and perhaps a 5 unit interval will delineate the structures. We create the vector “cvec” to tell the contour command which intervals we want. The caxis command after it is important, too. This forces matlab to anchor the colors to the full range.

```
> cvec=-25:5:35;
> figure(1)
> contourf(lat, pfull, data1,cvec)
> caxis([min(cvec) max(cvec)])
> colorbar;
> xlabel('x-units');ylabel('y-units');

>figure(2)
> contourf(lat, pfull, data2,cvec)
> caxis([min(cvec) max(cvec)])
> colorbar;
> xlabel('x-units');ylabel('y-units');
```


Use the correct axes!

Often it is tempting to get the data and plot it really quickly, say,
> `contourf(data1,20)`

This is fine at first, but now the axes won't give the units of (say) pressure and latitude. When you start to compare different models, the axes may change ... but you won't know if you're not being careful. I had a final presentation in a previous class where the student accidentally plotted only the first 32 latitudes (as opposed to all the latitudes) but never noticed because he wasn't plotting the axis data.

Also, label your axes. You should never show a plot where it's unclear what you're plotting. It helps you catch mistakes, and will make your plot readable in the future.

How to save a plot for a paper or presentation?

Use the print command. .eps (encapsulated postscript) files save the exact details of your figures. Assuming you are printing figure 1, this command will work. (The epsc2 is for a color .eps file, level 2.)

```
> print -f1 -depsc2 output_file_name
```

For printing figure 2, it's -f2, and so forth. You can also save figures in matlab format, which allows you to load them up later and edit them. However, these files can be included in a latex or word file.

Why is the fontsize so small?

I don't know, either. But it's a pain for saving your plots. To fix this, use this command, as I did in examples above.

```
> set(gca, 'fontsize', 20)
```