Lab 5: Python frontal analysis

Objective: To continue learning Python and how it could be used to diagnose frontal features.

Materials: Your laptop, Enthought Python, access to the Internet, a No. 2 black pencil, eraser, and colored pencils. It will not be necessary to log into the SoM machines if you have a laptop, however if you do not have a laptop you can also complete this lab on any of the SoM machines in room 5720.

Procedure:

1) Downloads
   (a) We have already downloaded and converted a GFS analysis file to NETCDF format. All you need to do is download it from `http://weather.ou.edu/~metr4424/files/`. Click on the file named `gfs_4_20120908_0000_000.nc`.
   (b) Keeping in mind our file naming convention from the previous labs, fill in the valid time of this GFS file below:
      (i) Year:
      (ii) Month:
      (iii) Day:
      (iv) Hour:
   (c) New or updated scripts to download from the class repository (`https://github.com/metr4424/classcode`) are:
      `weather_modules.py`
      `print_netcdf_info.py`
      `plot_gfs_cross_section.py`

      You will be using the same `plot_gfs_fields_forlab.py` from Lab 3, so if you re-download all python scripts be sure to save your `plot_gfs_fields_forlab.py` somewhere else first.
2) Plots
   (a) Open up `plot_gfs_fields_forlab.py`, which you should already have from your previous lab. Examine the user options, near the top of the file, under the comment "Set user options."
   (b) Set `date_string` and `fpath` the correct date and directory of your GFS netcdf file.
   (c) Set `level_option` to -1. This option is for plotting a surface analysis.
   (d) Make sure that `map_projection = 'lcc'`, and `plot_barbs = 'true'`.
   (e) Set figname = `"gfs_surface_analysis1"`
   (f) Change your map settings so that the map focuses more on the Continental United States. Under the line `elif map_projection == 'lcc'`, change:

```
m = Basemap(llcrnrlon=-125.5,llcrnrlat=15.,urcrnrlon=-30.,urcrnrlat=50.352,\
```

      to

```
m = Basemap(llcrnrlon=-120.0,llcrnrlat=20.,urcrnrlon=-60.0,urcrnrlat=50.0,\
```

   (g) Run `plot_gfs_fields_forlab.py`. If you did everything right, you will notice that you get an error `'temperature_plot' is not defined`.

(h) You will need to add in temperatures for your surface plot. The standard for a surface map is to plot 2 meter temperatures in degrees Fahrenheit. Recall from Lab 2 how you determined the variable names and dimensions of a GFS netcdf file. There are a couple of options for you to find the correct variable here:

(i) Option 1: Even though the date of the GFS file has changed, the format and naming scheme of the NETCDF variables have not. You can obtain your text file you named `gfs_fields` from lab2, which should be in your user space on the SoM machines. You do not need to transfer the file, unless you want to (in which case it's probably easiest to put it on your web space and download it to your laptop via a web browser). You can also look at the contents of a text file by typing `more gfs_fields` in the directory your text file called `gfs_fields` is located. If you want to print the values of a certain field within the netcdf file, from an SoM machine, you will need to execute the following command from the directory where your original netcdf file is located:

```
ncdump -v '[variable_name]' [netcdf_filename]
```

For example, if you want to print out the isobaric levels, replace `'[variable_name]'` with `'lv_ISBL0'`.

(ii) Option 2: On your laptop, open the new python script that you downloaded from the class repository called `print_netcdf_info.py`. Edit the user options section so that it points to your GFS netcdf file. When you execute the script, it will print out various information about your netcdf file. You can apply the example provided in `print_netcdf_info.py` for printing information about a variable, such as dimensions or values.

(iii) All variables that have to do with temperature begin with `TMP_P0_`. For instance, `TMP_P0_L100_GLL0` has dimensions `(lv_ISBL0, lat_0, lon_0)`. If you print the values of `lv_ISBL0`, you will notice that `lv_ISBL0` are isobaric levels, which means that `TMP_P0_L100_GLL0` is temperature on each latitude and longitude grid point for every isobaric level. Hence, this is not the temperature array we are looking for, because we are looking for temperature on a constant height surface of 2 meters.

(i) Once you have determined the correct variable name, read it in under the `(level_option == -1 ) :` section as

```
temperature_plot = f.variables['TMP_P0_aaaa_bbbb'][0,::-1,:].squeeze()
```

where you will need to change `aaaa` and `bbbb` to match the correct variable name you discovered above. Note that the values of `lv_HTGL2` are `2, 80, 100`, which means index 0 corresponds to 2 meter heights, index 1 to 80 meter heights, and index 2 to 100 meter heights. `temperature_plot` is already set to read in the correct index in the example here.

(j) You will need to repeat the above for the u and v components of the wind. The standard is to plot surface winds 10 meters above ground level. Read in these variables under `temperature_plot` using the variable names `u_plot` and `v_plot`, respectively.
Hint: Look for a vertical dimension named `lv_HTGL9`.

(k) Run the script. If everything has been done correctly, you will get a plot of sea level pressure contours in hPa and 10-meter wind barbs in knots. Notice though that there were no temperature contours plotted. This is because the contour interval was set assuming that temperature is plotted with units of degrees Celsius in this script. The standard for plotting temperature at the surface is degrees Fahrenheit. In the python

script, you will see a comment line that says "Convert temperature to degrees Celsius." Below that line, temperature is converted from the SI unit Kelvin to degrees Celsius. Adjust your script so that it will still plot temperature in degrees Celsius for plots on constant pressure surfaces but degrees Fahrenheit for surface plots. One way to do this is to add 2 lines just below the line that currently converts to degrees Celsius following the general structure of below:

```
if ( level_option == -1 ) :
   temperature_plot = [enter your conversion to degrees Fahrenheit]
```

Hint and note: In Python, `9/5` is NOT the same as `9./5.`

(l) Now you'll need to adjust the temperature contours so that they are in the range of the temperatures in degrees Fahrenheit here. However, you only want this range for a surface plot. So under the line that contains `cflevs_temp =` , you will want to add another if statement following the general structure as below:

```
if ( level_option == -1 ) :
   cflevs_temp =  np.arange(min_temperature,max_temperature,contour_interval)
```

where `min_temperature` and `max_temperature` define the minimum and maximum limits of temperature you expect to see, and `contour_interval` defines the spacing between contours. Set `contour_interval` so that it is consistent with your hand analysis from Lab 4.

(m) Near the bottom of the script, enable it so that temperature contours are plotted using the surface option in addition to the 500 hPa option. That is, change:

```
elif (level_option == 50000 ):
```

to

```
elif (level_option == 50000 or level_option == -1):
```

(n) Run the script. If everything has been done correctly, you will have a plot of sea level pressure contours in hPa, wind barbs in knots, and temperatures in degrees Fahrenheit. Print out a copy.

3) Adding Dewpoint temperatures

(a) In the user options section, set figname = `gfs_surface_analysis2`

(b) Add the following block under the first `if ( level_option == -1 ) :` where the netcdf variables are being read in:

```
spechum = f.variables['SPFH_P0_L103_GLL0'][0,::-1,:].squeeze()
sfcpres = f.variables['PRES_P0_L1_GLL0'][::-1,:]
tdew_plot = wm.spechum_to_td(spechum, sfcpres)
tdew_plot = (tdew_plot -273.15)*(9.0/5.0)+32
tdew_plot, abc = um.addcyclic(tdew_plot, lons)
```

(c) The block above reads in specific humidity, and calls a function in `weather_modules.py` to convert to dewpoint temperature. Dewpoint temperature is then converted to degrees Fahrenheit.

(d) Adjust your script so that instead of plotting temperature, you plot dewpoint temperature with green contours. This involves commenting out the line near the bottom that reads

```
CS2 = m.contour(x, y, temperature_plot, cflevs_temp, colors='r',
```

`linestyles='dashed',linewidths=1.5)`

and replacing it with the same line, except where `temperature_plot` is changed to the array that contains your dewpoint temperature and `colors='r'` is changed to `colors='g'`.

(e) Run the script and print.

(f) You should now have 2 plots, containing pressure, temperature, wind, and moisture fields. On your printed copies, analyze the cold, warm, and occluded fronts. First use your pencil and eraser to determine the tentative frontal locations, then finalize with your colored pencils using the same colors as you used in Lab 4.

4) Cross section

(a) Open the file called `plot_gfs_cross_section.py`.

(b) Set `date_string` and `fpath` as in your other scripts. Do not change any other user option.

(c) Run `plot_gfs_cross_section.py`. If successful, you will get a plot with pressure on the y-axis and longitude on the x-axis. The plot is a cross section oriented from west (100°W) to east (80°W) along a single latitude at 37°N. Colors show geopotential height anomalies (the mean geopotential heights along each pressure surface are subtracted from the full field). Dashed red contours are isotherms, and solid black contours are isentropes.

(d) Print out the image that is saved.

(e) On your printout, draw a cold front based on *isotherms* only. Starting at the bottom of the plot (1000 hPa), with your pencil, lightly mark location of the leading edge of where you think there is a relatively strong horizontal temperature gradient with a dot or short vertical line. Move upward to the next vertical level and repeat. Continue moving upward until this pattern of horizontal temperature gradients is no longer apparent. Then, draw a line connecting these points to mark your cold front. Keep in mind the conceptual picture of a frontal zone structure and how it varies with height.

(f) Finalize your cold front with a blue colored pencil. Label the air masses by writing 'Cold' on the cold side of your front and 'Warm' on the warm side of your front.

(g) Describe your frontal structure as it varies with height. Include in your discussion the following:

(i) About what pressure does the front extends to in the vertical direction?

(ii) Where are the lowest geopotential heights with respect to the surface front?

(iii) Recalling that stronger vertical potential temperature gradients correspond to greater stability, what can you conclude about the stability in the frontal zone relative to the surrounding locations?

(h) Now look at a cross section that is oriented from south (25°N) to north (50°N) along the longitude 268° (92°W). You can do this by setting `lat_range = [25,50]` and `lon_range = [268,268]` in the user options section of your Python script. Add the following to your discussion above:

(i) At what latitude is the surface front located on the lower-most level of 1000 hPa?

(ii) Note that just like your conceptual picture of surface fronts where pressure is a minimum along the front, geopotential heights similarly reach a minimum along the front. Given this, what can you say about the tilt and strength of the **surface** front with height?

Hand in all plots for this lab with answers to any questions above. This lab is due at the beginning of class on Wednesday September 19.