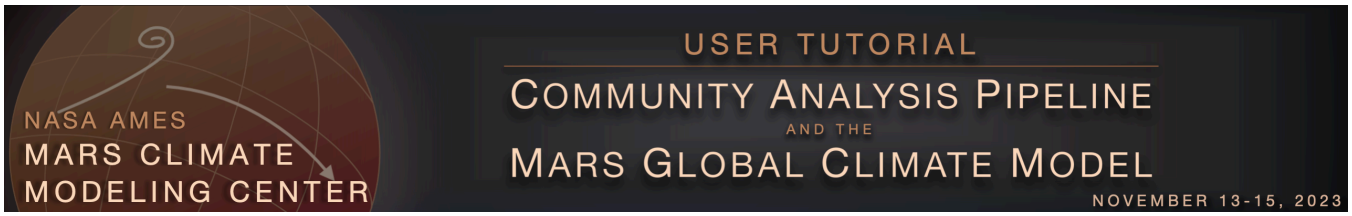
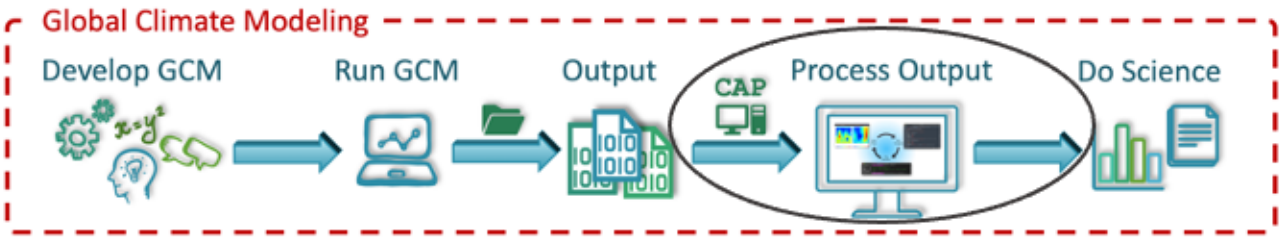


Practice Exercises - Community Analysis Pipeline (CAP)



CAP is a Python toolkit designed to simplify post-processing and plotting MGCM output. CAP consists of five Python executables:

1. `MarsPull.py` → accessing MGCM output
2. `MarsFiles.py` → reducing the files
3. `MarsVars.py` → performing variable operations
4. `MarsInterp.py` → interpolating the vertical grid
5. `MarsPlot.py` → plotting MGCM output

The following exercises are organized into two parts by function. We will go through **Part I on Monday Nov. 13** and **Part II on Tuesday Nov. 14**.

Part I: File Manipulations → `MarsFiles.py` , `MarsVars.py` , & `MarsInterp.py`

Part II: Plotting with CAP → `MarsPlot.py`

> We will not be going over `MarsPull.py` because it is specifically for retrieving MGCM data from the online MGCM data repository. Instructions for `MarsPull.py` was covered in the 2021 Legacy Version Tutorial.

Table of Contents

Activating CAP

Activate the `amesCAP` virtual environment to use CAP:

```
(cloud)~$ source ~/amesCAP/bin/activate
(amesCAP)~$
```

Confirm that CAP's executables are accessible by typing:

```
(amesCAP)~$ MarsVars.py -h
```

The `--help [-h]` argument prints documentation for the executable to the terminal. Now that we know CAP is configured, make a copy of the file `amescap_profile` in your home directory, and make it a hidden file:

```
(amesCAP)~$ cp ~/amesCAP/mars_templates/amescap_profile ~/.amescap_profile
```

CAP stores useful settings in `amescap_profile`. Copying it to our home directory ensures it is not overwritten if CAP is updated or reinstalled.

Following Along with the Tutorial

Part I covers file manipulations. Some exercises build off of previous exercises so *it is important to complete them in order*. If you make a mistake or get behind in the process, you can go back and catch up during a break or use the provided answer key before continuing on to Part II.

Part II demonstrates CAP's plotting routine. There is more flexibility in this part of the exercise.

We will perform every exercise together.

> *Feel free to **put questions in the chat** throughout the tutorial. Another MGCM member can help you as we go.*

Return to Top

Part I: File Manipulations

CAP has dozens of post-processing capabilities. We will go over a few of the most commonly used functions in this tutorial. We will cover:

- **Interpolating** data to different vertical coordinate systems (`MarsInterp.py`)

- **Adding derived variables** to the files (`MarsVars.py`)
- **Time-shifting** data to target local times (`MarsFiles.py`)
- **Trimming** a file to reduce its size (`MarsFiles.py`).

The required MGCM output files are already loaded in the cloud environment under `tutorial_files/cap_exercises/` . Change to that directory and look at the contents:

```
(amesCAP)~$ cd tutorial_files/cap_exercises
(amesCAP)~$ ls
03340.atmos_average.nc  03340.backup.zip  part_1_key.sh
03340.atmos_diurn.nc    03340.fixed.nc    part_2_plots.in
```

The three MGCM output files have a 5-digit sol number appended to the front of the file name. The sol number indicates the day that a file's record begins. These contain output from the sixth year of a simulation. The zipped file is an archive of these three output files in case you need it.

The other two files, `part_1_key.sh` and `part_2_plots.sh` are discussed later. We can ignore them for now.

> *The output files we manipulate in Part I will be used to generating plots in Part II so do **not** delete any file you create!*

Let's begin the tutorial.

1. MarsPlot's `--inspect` Function

The inspect function is part of `MarsPlot.py` and it prints netCDF file contents to the screen.

To use it on the `average` file, `03340.atmos_average.nc` , type the following in the terminal:

```
(amesCAP)~$ MarsPlot.py -i 03340.atmos_average.nc
```

> This is a good time to remind you that if you are unsure how to use a function, invoke the `--help [-h]` argument with any executable to see its documentation (e.g., `MarsPlot.py -h`).

Return to Part I

---

2. Editing Variable Names and Attributes

In the previous exercise, `--inspect [-i]` revealed a variable called `opac` in `03340.atmos_average.nc`. `opac` is dust opacity per pascal and it is similar to another variable in the file, `dustref`, which is opacity per (model) level. Let's rename `opac` to `dustref_per_pa` to better indicate the relationship between these variables.

We can modify variable names, units, longnames, and even scale variables using the `-edit` function in `MarsVars.py`. The syntax for editing the variable name is:

```
(amesCAP)~$ MarsVars.py 03340.atmos_average.nc -edit opac -rename
dustref_per_pa
03340.atmos_average_tmp.nc was created
03340.atmos_average.nc was updated
```

We can use `--inspect [-i]` again to confirm that `opac` was renamed `dustref_per_pa`:

```
(amesCAP)~$ MarsPlot.py -i 03340.atmos_average.nc
```

The `--inspect [-i]` function can also **print a summary of the values** of a variable to the screen. For example:

```
(amesCAP)~$ MarsPlot.py -i 03340.atmos_average.nc -stat dustref_per_pa
_____________________________________________________________
  VAR           |   MIN     |    MEAN      |    MAX       |
________________|___________|______________|______________|
  dustref_per_pa|          0|  0.000384902|    0.0017573|
________________|___________|______________|______________|
```

Finally, `--inspect [-i]` can **print the values** of a variable to the screen. For example:

```
(amesCAP)~$ MarsPlot.py -i 03340.atmos_average.nc -dump lat
lat=
[-89. -87. -85. -83. -81. -79. -77. -75. -73. -71. -69. -67. -65. -63.
 -61. -59. -57. -55. -53. -51. -49. -47. -45. -43. -41. -39. -37. -35.
 -33. -31. -29. -27. -25. -23. -21. -19. -17. -15. -13. -11.  -9.  -7.
  -5.  -3.  -1.   1.   3.   5.   7.   9.  11.  13.  15.  17.  19.  21.
   1.  25.  27.  29.  31.  33.  35.  37.  39.  41.  43.  45.  47.  49.
   2.  53.  55.  57.  59.  61.  63.  65.  67.  69.  71.  73.  75.  77.
   3.  81.  83.  85.  87.  89.]
```

Return to Part I

---

3. Splitting Files in Time

Next we're going to trim the `diurn` and `average` files by L_s. We'll create files that only contain data around southern summer solstice, L_s=270. This greatly reduces the file size to make our next post-processing steps more efficient.

Syntax for trimming files by L_s is:

```
(amesCAP)~$ MarsFiles.py 03340.atmos_diurn.nc -split 265 275
...
/home/centos/tutorial_files/cap_exercises/03847.atmos_diurn_Ls265_275.nc was
created
```

```
(amesCAP)~$ MarsFiles.py 03340.atmos_average.nc -split 265 275
...
/home/centos/tutorial_files/cap_exercises/03847.atmos_average_Ls265_275.nc was
 created
```

The trimmed files have the appendix `_Ls265_275.nc` and the simulation day has changed from `03340` to `03847` to reflect that the first day in the file has changed.

For future steps, we need a `fixed` file with the same simulation day number as the files we just created, so make a copy of the `fixed` file and rename it:

```
(amesCAP)~$ cp 03340.fixed.nc 03847.fixed.nc
```

Return to Part I

Break

Take 15 minutes to stretch, ask questions, or let us know your thoughts on CAP so far!

---

4. Deriving Secondary Variables

The `--add` function in `MarsVars.py` derives and adds secondary variables to MGCM output files provided that the variable(s) required for the derivation are already in the file. We will add the meridional mass streamfunction (`msf`) to the trimmed `average` file. To figure out what we need in order to do this, use the `--help [-h]` function on `MarsVars.py` :

```
(amesCAP)~$ MarsVars.py -h
```

The help function shows that streamfunction (`msf`) requires two things: that the meridional wind (`vcomp`) is in the `average` file, and that the `average` file is ***pressure-interpolated***.

First, confirm that `vcomp` is in `03847.atmos_average_Ls265_275.nc` using `--inspect [-i]` :

```
(amesCAP)~$ MarsPlot.py -i 03847.atmos_average_Ls265_275.nc
...
vcomp : ('time', 'pfull', 'lat', 'lon')= (3, 56, 90, 180), meridional wind
[m/sec]
```

Second, pressure-interpolate the average file using `MarsInterp.py` . The call to `MarsInterp.py` requires:

- The interpolation type (`--type [-t]`), we will use standard pressure coorindates (`pstd`)

- The grid to interpolate to (`--level [-l]`), we will use the default pressure grid (`pstd_default`)

> All interpolation types are listed in the `--help [-h]` documentation for `MarsInterp.py` . Additional grids are listed in `~/.amescap_profile` , which accepts user-input grids as well.

We will also specify that only temperature (`temp`), winds (`ucomp` and `vcomp`), and surface pressure (`ps`) are to be included in this new file using `-include` . This will reduce the interpolated file size.

Finally, add the `--grid [-g]` flag at the end of prompt to print out the standard pressure grid levels that we are interpolating to:

```
(amesCAP)~$ MarsInterp.py 03847.atmos_average_Ls265_275.nc -t pstd -l
pstd_default -include temp ucomp vcomp ps -g
1100.0 1050.0 1000.0 950.0 900.0 850.0 800.0 750.0 700.0 650.0 600.0 550.0
500.0 450.0 400.0 350.0 300.0 250.0 200.0 150.0 100.0 70.0 50.0 30.0 20.0
10.0 7.0 5.0 3.0 2.0 1.0 0.5 0.3 0.2 0.1 0.05
```

To perform the interpolation, simply omit the `--grid [-g]` flag:

```
(amesCAP)~$ MarsInterp.py 03847.atmos_average_Ls265_275.nc -t pstd -l
pstd_default -include temp ucomp vcomp ps
...
/home/centos/tutorial_files/cap_exercises/
03847.atmos_average_Ls265_275_pstd.nc was created
```

Now we have a pressure-interpolated `average` file with `vcomp` in it. We can derive and add `msf` to it using `MarsVars.py` :

```
(amesCAP)~$ MarsVars.py 03847.atmos_average_Ls265_275_pstd.nc -add msf
Processing: msf...
msf: Done
```

Return to Part I

---

5. Time-Shifting `Diurn` Files

The `diurn` file is organized by time-of-day assuming ***universal*** time starting at the Martian prime meridian. The time-shift `--tshift [-t]` function interpolates the `diurn` file to ***uniform local*** time. This is especially useful when comparing MGCM output to satellite observations in fixed local time orbit.

Time-shifting can only be done on files with a local time dimension (`time_of_day_24` , i.e. `diurn` files). By default, `MarsFiles.py` time shifts all of the data in the file to 24 uniform local times and this generates very large files. To reduce file size and processing time, we will time-shift the data only to the local times we are interested in: 3 AM and 3 PM.

Time-shift the temperature (`temp`) and surface pressure (`ps`) in the trimmed `diurn` file to 3 AM / 3 PM local time like so:

```
(amesCAP)~$ MarsFiles.py 03847.atmos_diurn_Ls265_275.nc -t '3. 15.' -include
temp ps
...
/home/centos/tutorial_files/cap_exercises/03847.atmos_diurn_Ls265_275_T.nc was
 created
```

A new `diurn` file called `03847.atmos_diurn_Ls265_275_T.nc` is created. Use `--inspect [-i]` to confirm that only `ps` and `temp` (and their dimensions) are in the file and that the `time_of_day` dimension has a length of 2:

```
(amesCAP)~$ MarsPlot.py -i 03847.atmos_diurn_Ls265_275_T.nc
...
====================CONTENT=========================
time           : ('time',)= (3,), sol number  [days since 0000-00-00 00:00:00]
time_of_day_02 : ('time_of_day_02',)= (2,), time of day  [[hours since
0000-00-00 00:00:00]]
pfull          : ('pfull',)= (56,), ref full pressure level  [mb]
scalar_axis    : ('scalar_axis',)= (1,), none  [none]
lon            : ('lon',)= (180,), longitude  [degrees_E]
lat            : ('lat',)= (90,), latitude  [degrees_N]
areo           : ('time', 'time_of_day_02', 'scalar_axis')= (3, 2, 1), areo
[degrees]
ps             : ('time', 'time_of_day_02', 'lat', 'lon')= (3, 2, 90, 180),
surface pressure  [Pa]
temp           : ('time', 'time_of_day_02', 'pfull', 'lat', 'lon')= (3, 2,
56, 90, 180), temperature  [K]
====================================================
```

Return to Part I

---

6. Pressure-Interpolating the Vertical Axis

Now we can efficiently interpolate the `diurn` file to the standard pressure grid. Recall that interpolation is part of `MarsInterp.py` and requires:

1. Interpolation type (`--type [-t]`), and
2. Grid (`--level [-l]`)

As before, we will interpolate to standard pressure (`pstd`) using the default pressure grid in `.amesgcm_profile` (`pstd_default`):

```
(amesCAP)~$ MarsInterp.py 03847.atmos_diurn_Ls265_275_T.nc -t pstd -l
pstd_default
...
/home/centos/tutorial_files/cap_exercises/
03847.atmos_diurn_Ls265_275_T_pstd.nc was created
```

> **Note:** Interpolation could be done before or after time-shifting, the order does not matter.

We now have four different `diurn` files in our directory:

```
03340.atmos_diurn.nc                  # Original MGCM file
03847.atmos_diurn_Ls265_275.nc        # + Trimmed to Ls=265-275
03847.atmos_diurn_Ls265_275_T.nc      # + Time-shifted; `ps` and `temp` only
03847.atmos_diurn_Ls265_275_T_pstd.nc # + Pressure-interpolated
```

CAP always adds an appendix to the name of any new file it creates. This helps users keep track of what was done and in what order. The last file we created was trimmed, time-shifted, then pressure-interpolated. However, the same file could be generated by performing the three functions in any order.

Return to Part I

Optional: Use the Answer Key for Part I

This concludes Part I of the tutorial! If you messed up one of the exercises somewhere, you can run the `part_1_key.sh` script in this directory. It will delete the files you've made and performs all 6 Exercises in Part I for you. To do this, follow the steps below.

1. Source the `amesCAP` virtual environment
2. Change to the `tutorial_files/cap_exercises/` directory
3. Run the executable:

```
(amesCAP)~$ ./part_1_key.sh
```

The script will do all of Part I for you. This ensures you can follow along with the plotting

routines in Part II.

Return to Part I

CAP Practical Day 2

This part of the CAP Practical covers how to generate plots with CAP. We will take a learn-by-doing approach, creating five sets of plots that demonstrate some of CAP's most often used plotting capabilities:

1. Custom Set 1 of 4: Zonal Mean Surface Plots Over Time
2. Custom Set 2 of 4: Global Mean Column-Integrated Dust Optical Depth Over Time
3. Custom Set 3 of 4: 50 Pa Temperatures at 3 AM and 3 PM
4. Custom Set 4 of 4: Zonal Mean Circulation Cross-Sections

Plotting with CAP is done in 3 steps:

Step 1: Creating the Template (`Custom.in`)

Step 2: Editing `Custom.in`

Step 3: Generating the Plots

As in Part I, we will go through these steps together.

Part II: Plotting with CAP

CAP's plotting routine is `MarsPlot.py`. It works by generating a `Custom.in` file containing seven different plot templates that users can modify, then reading the `Custom.in` file to make the plots.

The plot templates in `Custom.in` include:

Plot Type	X, Y Dimensions	Name in `Custom.in`
Map	Longitude, Latitude	`Plot 2D lon x lat`
Time-varying	Time, Latitude	`Plot 2D time x lat`
Time-varying	Time, level	`Plot 2D time x lev`
Time-varying	Longitude, Time	`Plot 2D lon x time`
Cross-section	Longitude, Level	`Plot 2D lon x lev`
Cross-section	Latitude, Level	`Plot 2D lat x lev`
Line plot (1D)	Dimension*, Variable	`Plot 1D`

> *Dimension is user-indicated and could be time (`time`), latitude (`lat`), longitude `lon` , or level (`pfull` , `pstd` , `zstd` , `zagl`).

Additionally, `MarsPlot.py` supports:

- PDF & image format
- Landscape & portrait mode
- Multi-panel plots
- Overplotting
- Customizable axes dimensions and contour intervals
- Adjustable colormaps and map projections

and so much more. You will learn to plot with `MarsPlot.py` by following along with the demonstration. We will generate the `Custom.in` template file, customize it, and pass it back into `MarsPlot.py` to create plots.

Return to Part II

Step 1: Creating the Template (`Custom.in`)

Generate the template file, `Custom.in` :

```
(amesCAP)~$ MarsPlot.py -template
/home/centos/tutorial_files/cap_exercises/Custom.in was created
```

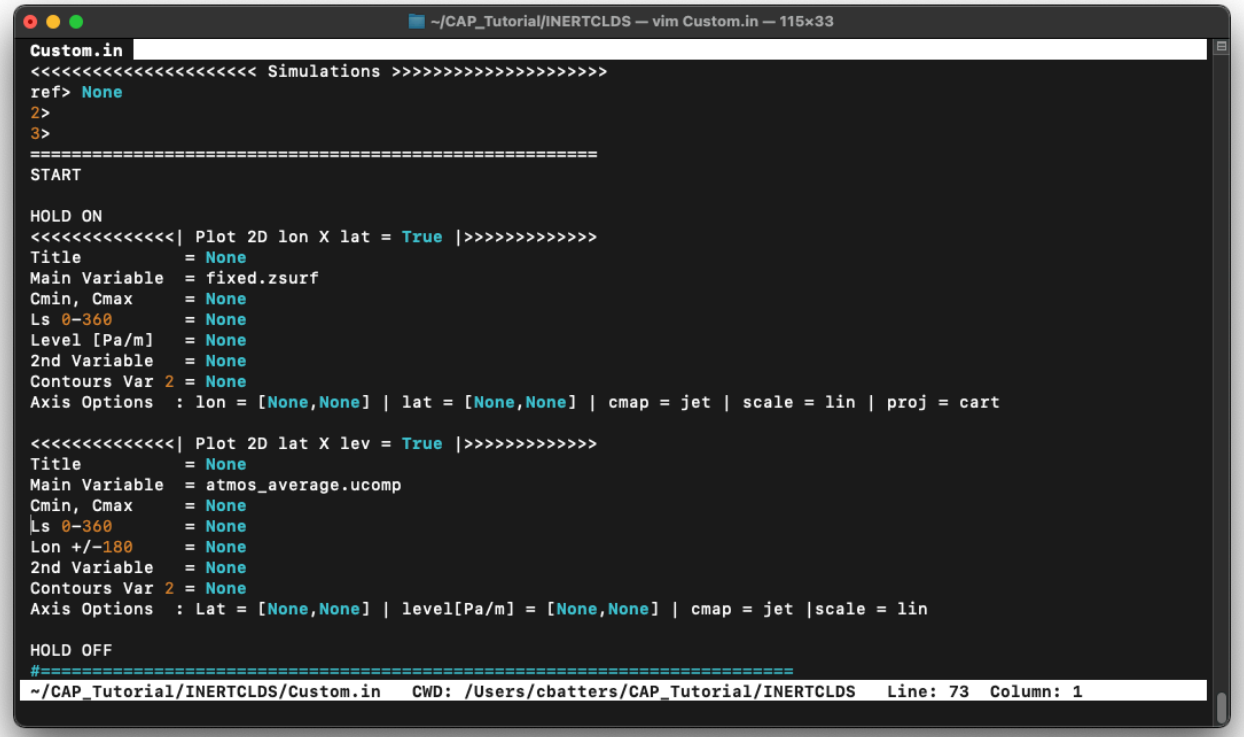
A new file called `Custom.in` is created in your current working directory.

Step 2: Editing `Custom.in`

Open `Custom.in` using `vim`:

```
(amesCAP)~$ vim Custom.in
```

Scroll down until you see the first two templates shown in the image below:



Since all of the templates have a similar structure, we can broadly describe how `Custom.in` works by going through the templates line-by-line.

Line 1

```
# Line 1                ┌ plot type   ┌ whether to create the plot
<<<<<<<<<<<<<<| Plot 2D lon X lat = True |>>>>>>>>>>>>>>
```

Line 1 indicates the **plot type** and **whether to create the plot** when passed into `MarsPlot.py`.

Line 2

```
# Line 2          ┌ file ┌ variable
Title           = None
```

Line 2 is where we set the plot title.

Line 3

```
# Line 3          ┌ file ┌ variable
Main Variable   = fixed.zsurf           # file.variable
Main Variable   = [fixed.zsurf]/1000    # [] brackets for mathematical
operations
Main Variable   = diurn_T.temp{tod=3}   # {} brackets for dimension selection
```

Line 3 indicates the **variable** to plot and the **file** from which to pull the variable.

Additional customizations include:

- Element-wise operations (e.g., scaling by a factor)
- Dimensional selection (e.g., selecting the time of day (`tod`) at which to plot from a time-shifted diurn file)

Line 4

```
# Line 4
Cmin, Cmax      = None           # automatic, or
Cmin, Cmax      = -4,5           # contour limits, or
Cmin, Cmax      = -4,-2,0,1,3,5  # explicit contour levels
```

Line 4 line defines the **color-filled contours** for `Main Variable`. Valid inputs are:

- `None` (default) enables Python's automatic interpretation of the contours
- `min,max` specifies contour range
- `X,Y,Z,...,N` gives explicit contour levels

Lines 5 & 6

```
# Lines 5 & 6
Ls 0-360        = None # for 'time' free dimension
Level Pa/m      = None # for 'pstd' free dimension
```

Lines 5 & 6 handle the **free dimension(s)** for `Main Variable` (the dimensions that are ***not*** plot dimensions).

For example, `temperature` has four dimensions: `(time, pstd, lat, lon)`. For a `2D lon X lat` map of temperature, `lon` and `lat` provide the `x` and `y` dimensions of the plot. The free dimensions are then `pstd` (`Level Pa/m`) and `time` (`Ls 0-360`).

Lines 5 & 6 accept four input types:

1. `integer` selects the closest value
2. `min,max` averages over a range of the dimension
3. `all` averages over the entire dimension
4. `None` (default) depends on the free dimension:

```
# ┌ free dimension      ┌ default setting
Ls 0-360       = None   # most recent timestep
Level Pa/m     = None   # surface level
Lon +/-180     = None   # zonal mean over all longitudes
Latitude       = None   # equatorial values only
```

Lines 7 & 8

```
# Line 7 & 8
2nd Variable   = None           # no solid contours
2nd Variable   = fixed.zsurf    # draw solid contours
Contours Var 2  = -4,5          # contour range, or
Contours Var 2  = -4,-2,0,1,3,5 # explicit contour levels
```

Lines 7 & 8 (optional) define the **solid contours** on the plot. Contours can be drawn for `Main Variable` or a different `2nd Variable`.

- Like `Main Variable`, `2nd Variable` minimally requires `file.variable`
- Like `Cmin, Cmax`, `Contours Var 2` accepts a range (`min,max`) or list of explicit contour levels (`X,Y,Z,...,N`)

Line 9

```
# Line 9        ┌ X axes limit      ┌ Y axes limit      ┌ colormap   ┌ cmap
scale  ┌ projection
 Axis Options : lon = [None,None] | lat = [None,None] | cmap = jet | scale =
lin | proj = cart
```

Finally, Line 9 offers plot customization (e.g., axes limits, colormaps, map projections, linestyles, 1D axes labels).

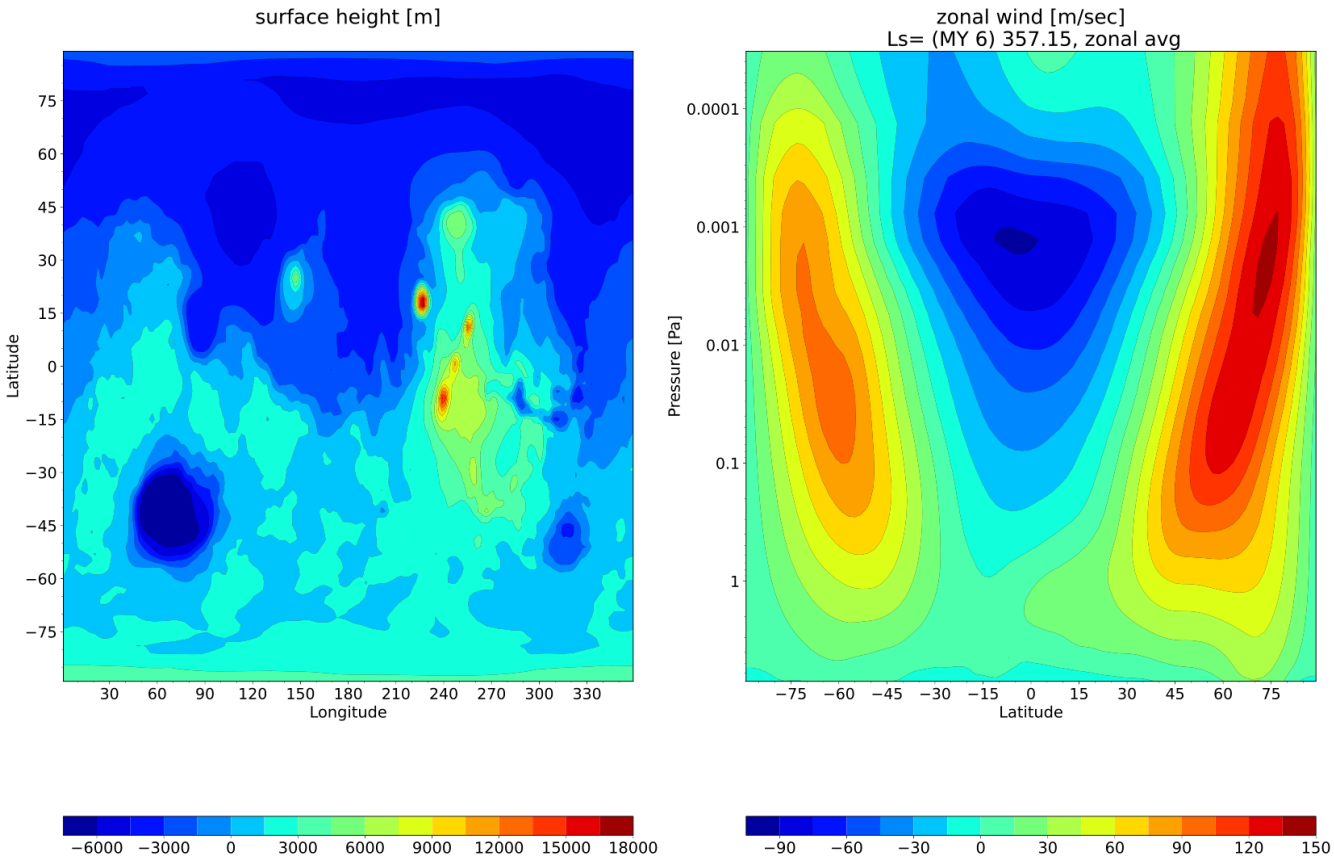
Return to Part II

Step 3: Generating the Plots

Generate the plots set to `True` in `Custom.in` by saving and quitting the editor (`:wq`) and then passing the template file to `MarsPlot.py`. The first time we do this, we'll pass the `--date [-d]` flag to specify that we want to plot from the `03340` `average` and `fixed` files:

```
(amesCAP)~$ MarsPlot.py Custom.in -d 03340
```

Plots are created and saved in a file called `Diagnostics.pdf`.



Viewing Diagnostics.pdf

To open the file, we need to copy it to our local computer. We'll create an alias for the command that does this so we can easily pull `Diagnostics.pdf` from the cloud environment.

First, open a new terminal tab (`CRTL-t`).

Then, change to the directory hosting your token for this tutorial (the token is the file ending in `.pem`).

Finally, build the secure copy (`scp`) command OR use `sftp`.

Using `scp` (recommended)

To build your `scp` command:

- The name of your `.pem` file (e.g., `mars-clusterXX.pem`)
- The address you used to login to the cloud (something like `centos@YOUR_IP_ADDRESS`)
- The path to the PDF in the cloud: `tutorial_files/cap_exercises/Diagnostics.pdf`

Putting these together, the secure copy command is:

```
(local)~$ scp -i "mars-clusterXX.pem" centos@YOUR_IP_ADDRESS:tutorial_files/cap_exercises/Diagnostics.pdf .
```

To make the command into an alias named `getpdf`:

```
(local)~$ alias getpdf='scp -i "mars-clusterXX.pem" centos@YOUR_IP_ADDRESS:tutorial_files/cap_exercises/Diagnostics.pdf .'
```

Now we can pull the PDF to our local computer with one simple command:

```
(local)~$ getpdf # uses scp
```

Using `sftp`

Alternatively, if `scp` isn't working for you, you can use `sftp`. To do this, go to your new terminal tab and type:

```
(local)~$ sftp -i "mars-clusterXX.pem" centos@YOUR_IP_ADDRESS
sftp> cd tutorial_files/cap_exercises
```

Then when you want to pull the PDF to your local computer, type:

```
sftp> get Diagnostics.pdf
```

Summary

Plotting with `MarsPlot.py` is done in 3 steps:

```
(amesCAP)~$ MarsPlot.py -template # generate Custom.in
(amesCAP)~$ vim Custom.in         # edit Custom.in
(amesCAP)~$ MarsPlot.py Custom.in # pass Custom.in back to MarsPlot
```

Now we will go through some examples.

Customizing the Plots

Open `Custom.in` in the editor:

```
(amesCAP)~$ vim Custom.in
```

Copy the first two templates that are set to `True` and paste them below the line `Empty Templates (set to False)`. Then, set them to `False`. This way, we have all available templates saved at the bottom of the script.

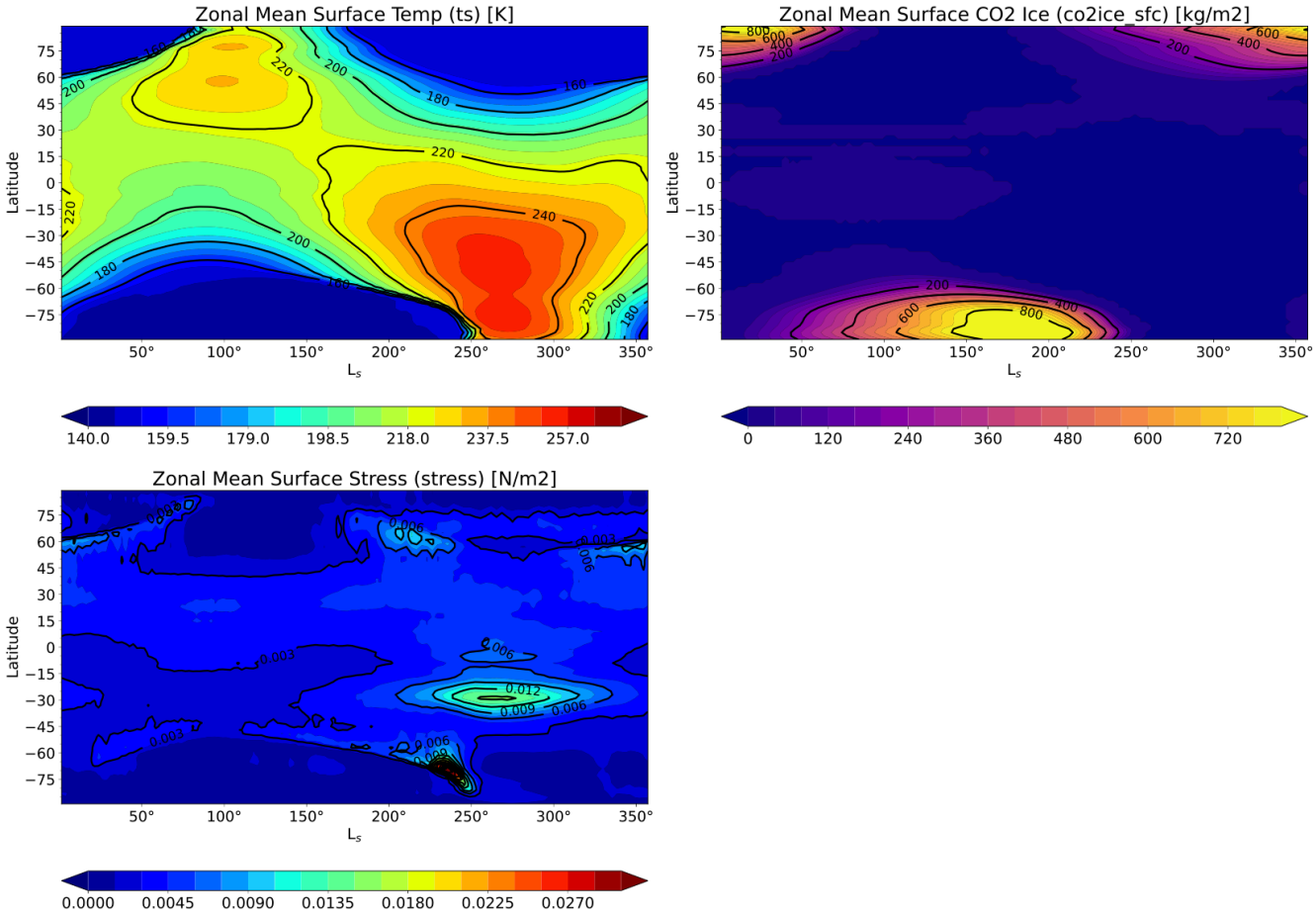
We'll preserve the first two plots, but let's define the sol number of the average and fixed files in the template itself so we don't have to pass the `--date [-d]` argument every time:

```
# for the first plot (lon X lat topography):
Main Variable  = 03340.fixed.zsurf
# for the second plot (lat X lev zonal wind):
Main Variable  = 03340.atmos_average.ucomp
```

Now we can omit the date (`--date [-d]`) when we pass `Custom.in` to `MarsPlot.py` .

Custom Set 1 of 4: Zonal Mean Surface Plots Over Time

The first set of plots we'll make are zonal mean surface fields over time: surface temperature, CO_2 ice, and wind stress.



For each of the plots, source variables from the *non*-interpolated average file, `03340.atmos_average.nc` .

For the **surface temperature** plot:

- Copy/paste the `Plot 2D time X lat` template above the `Empty Templates` line
- Set it to `True`
- Edit the title to `Zonal Mean Sfc T [K]`
- Set `Main Variable = 03340.atmos_average.ts`
- Edit the colorbar range: `Cmin, Cmax = 140,270` → *140-270 Kelvin*

- Set `2nd Variable = 03340.atmos_average.ts` → *for overplotted solid contours*
- Explicitly define the solid contours: `Contours Var 2 = 160,180,200,220,240,260`

Let's pause here and pass the `Custom.in` file to `MarsPlot.py`.

Type `ESC-:wq` to save and close the file. Then, pass it to `MarsPlot.py`:

```
(amesCAP)~$ MarsPlot.py Custom.in
```

Now, go to your **local terminal** tab and retrieve the PDF:

```
(local)~$ getpdf # uses scp
```

or

```
sftp> get Diagnostics.pdf # uses sftp
```

Now we can open it and view our plot.

Go back to the **cloud environment** tab to finish generating the other plots on this page. Open `Custom.in` in `vim`:

```
(amesCAP)~$ vim Custom.in
```

Write a set of `HOLD ON` and `HOLD OFF` arguments around the surface temperature plot. We will paste the other templates within these arguments to tell `MarsPlot.py` to put these plots on the same page.

Copy/paste the `Plot 2D time X lat` template plot twice more. Make sure to set the boolean to `True`.

For the **surface CO_2 ice** plot:

- Set the title to `Zonal Mean Sfc CO2 Ice [kg/m2]`
- Set `Main Variable = 03340.atmos_average.co2ice_sfc`
- Edit the colorbar range: `Cmin, Cmax = 0,800` → *0-800 kg/m^2*

- Set `2nd Variable = 03340.atmos_average.co2ice_sfc` → *solid contours*
- Explicitly define the solid contours: `Contours Var 2 = 200,400,600,800`
- Change the colormap on the `Axis Options` line: `cmap = plasma`

For the **surface wind stress** plot:

- Set the title to `Zonal Mean Sfc Stress [N/m2]`
- Set `Main Variable = 03340.atmos_average.stress`
- Edit the colorbar range: `Cmin, Cmax = 0,0.03` → *0-0.03 N/m^2*

Save and quit the editor (`ESC-:wq`) and pass `Custom.in` to `MarsPlot.py` :

```
(amesCAP)~$ MarsPlot.py Custom.in
```

In your **local terminal** tab, retrieve the PDF to view the plots:

```
(local)~$ getpdf # uses scp
```

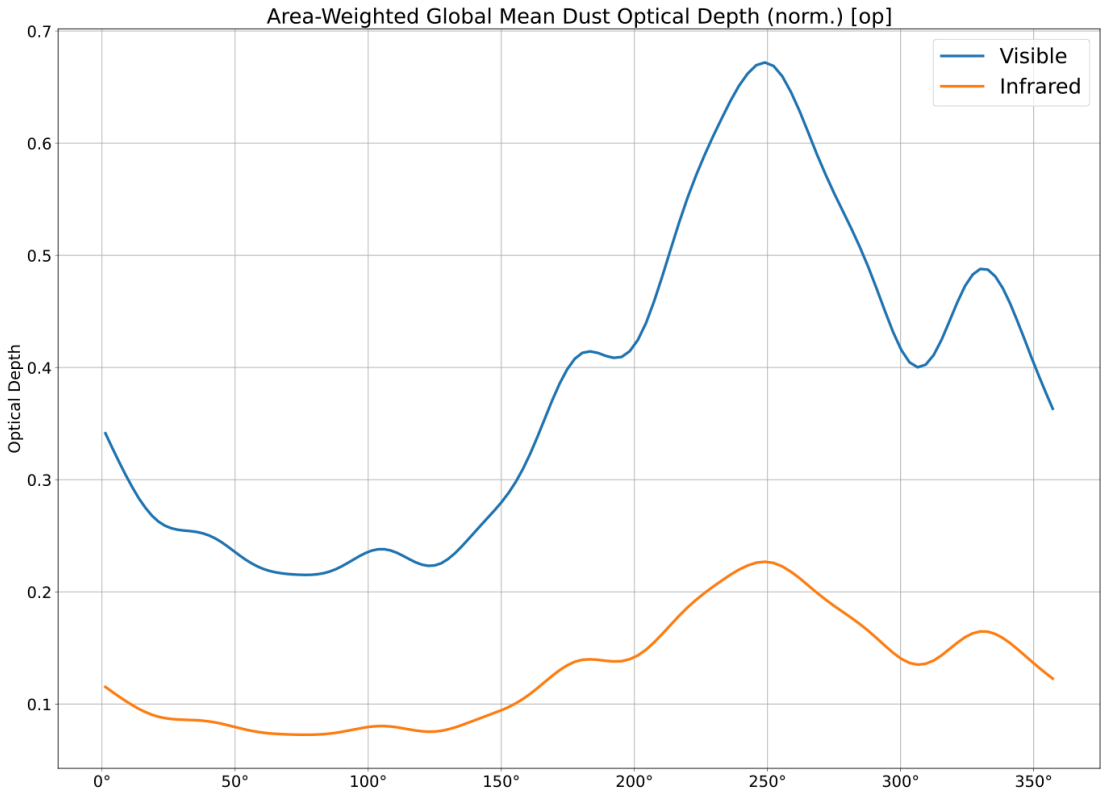
or

```
sftp> get Diagnostics.pdf # uses sftp
```

Return to Part II

Custom Set 2 of 4: Global Mean Column-Integrated Dust Optical Depth Over Time

Now we'll generate a 1D plot and practice plotting multiple lines on it.



Let's start by setting up our 1D plot template:

- Write a new set of `HOLD ON` and `HOLD OFF` arguments.
- Copy/paste the `Plot 1D` template between them.
- Set the template to `True`.

Create the **visible dust optical depth** plot first:

- Set the title: `Area-Weighted Global Mean Dust OD (norm.) [op]`
- Edit the legend: `Visible`

The input to `Main Variable` is not so straightforward this time. We want to plot the *normalized* dust optical depth, which is dervied as follows:

```
normalized_dust_OD = opacity / surface_pressure * reference_pressure
```

The MGCM outputs column-integrated visible dust opacity to the variable `taudust_VIS`,

surface pressure is saved as `ps`, and we'll use a reference pressure of 610 Pa. Recall that element-wise operations are performed when square brackets `[]` are placed around the variable in `Main Variable`. Putting all that together, `Main Variable` is:

```
# ┌ norm. OD      ┌ opacity                          ┌ surface pressure       ┌ ref. P
Main Variable  =
[03340.atmos_average.taudust_VIS]/[03340.atmos_average.ps]*610
```

To finish up this plot, tell `MarsPlot.py` what to do to the dimensions of `taudust_VIS (time, lon, lat)`:

- Leave `Ls 0-360 = AXIS` to use 'time' as the X axis dimension.
- Set `Latitude = all` → *average over all latitudes*
- Set `Lon +/-180  = all` → *average over all longitudes*
- Set the Y axis label under `Axis Options`: `axlabel = Optical Depth`

The **infrared dust optical depth** plot is identical to the visible dust OD plot except for the variable being plotted, so duplicate the **visible** plot we just created. Make sure both templates are between `HOLD ON` and `HOLD OFF` Then, change two things:

- Change `Main Variable` from `taudust_VIS` to `taudust_IR`
- Set the legend to reflect the new variable (`Legend = Infrared`)

Save and quit the editor (`ESC-:wq`). pass `Custom.in` to `MarsPlot.py`:

```
(amesCAP)~$ MarsPlot.py Custom.in
```

In your **local terminal** tab, retrieve the PDF to view the plots:

```
(local)~$ getpdf # uses scp
```

or

```
sftp> get Diagnostics.pdf # uses sftp
```

Notice we have two separate 1D plots on the same page. This is because of the `HOLD ON` and `HOLD OFF` arguments. Without those, these two plots would be on separate pages. But how do we overplot the lines on top of one another?

Go back to the cloud environment, open `Custom.in`, and type `ADD LINE` between the two 1D templates.

Save and quit again, pass it through `MarsPlot.py`, and retrieve the PDF locally. Now we have the overplotted lines we were looking for.

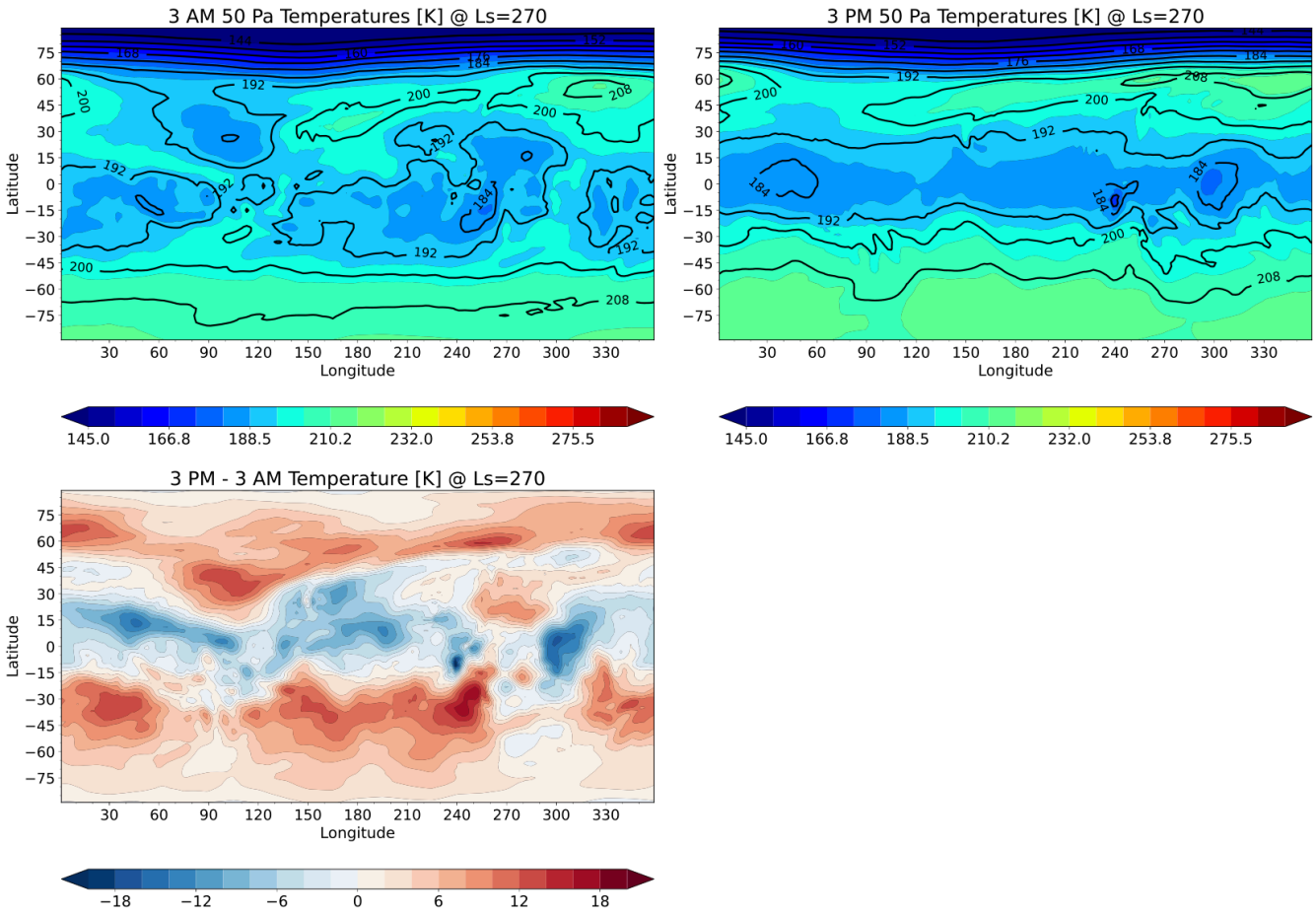
Return to Part II

Break

Take 15 minutes to stretch, ask questions, or let us know your thoughts on CAP so far!

Custom Set 3 of 4: 50 Pa Temperatures at 3 AM and 3 PM

The first two plots are 3 AM and 3 PM 50 Pa temperatures at L_s=270. Below is the 3 PM - 3 AM difference.



We'll generate all three plots before passing `Custom.in` to `MarsPlot.py`, so copy/paste the `Plot 2D lon X lat` template ***three times*** between a set of `HOLD ON` and `HOLD OFF` arguments and set them to `True`.

For the first plot,

- Title it for 3 AM temperatures: `3 AM 50 Pa Temperatures [K] @ Ls=270`
- Set `Main Variable` to `temp` and select 3 AM for the time of day using curly brackets:

```
Main Variable  = 03847.atmos_diurn_Ls265_275_T_pstd.temp{tod=3}
```

- Set the colorbar range: `Cmin, Cmax = 145,290` → *145-290 K*
- Set `Ls 0-360 = 270` → *southern summer solstice*
- Set `Level Pa/m = 50` → *selects 50 Pa temperatures*
- Set `2nd Variable` to be identical to `Main Variable`

Now, edit the second template for 3 PM temperatures the same way. The only differences are the:

- Title: edit to reflect 3 PM temperatures
- Time of day selection: for 3 PM, `{tod=15}` ***change this for `2nd Variable` too!***

For the **difference plot**, we will need to use square brackets in the input for `Main Variable` in order to subtract 3 AM temperatures from 3 PM temperatures. We'll also use a diverging colorbar to show temperature differences better.

- Set the title to `3 PM - 3 AM Temperature [K] @ Ls=270`
- Build `Main Variable` by subtracting the 3 AM `Main Variable` input from the 3 PM `Main variable` input:

```
Main Variable =
[03847.atmos_diurn_Ls265_275_T_pstd.temp{tod=15}]-[03847.atmos_diurn_Ls265_275_T_pstd.temp
```

- Center the colorbar at `0` by setting `Cmin, Cmax = -20,20`
- Like the first two plots, set `Ls 0-360 = 270` → *southern summer solstice*
- Like the first two plots, set `Level Pa/m = 50` → *selects 50 Pa temperatures*
- Select a diverging colormap in `Axis Options`: `cmap = RdBu_r`

Save and quit the editor (`ESC-:wq`). pass `Custom.in` to `MarsPlot.py`, and pull it to your local computer:

```
(amesCAP)~$ MarsPlot.py Custom.in
# switch to the local terminal...
(local)~$ getpdf # uses scp
```

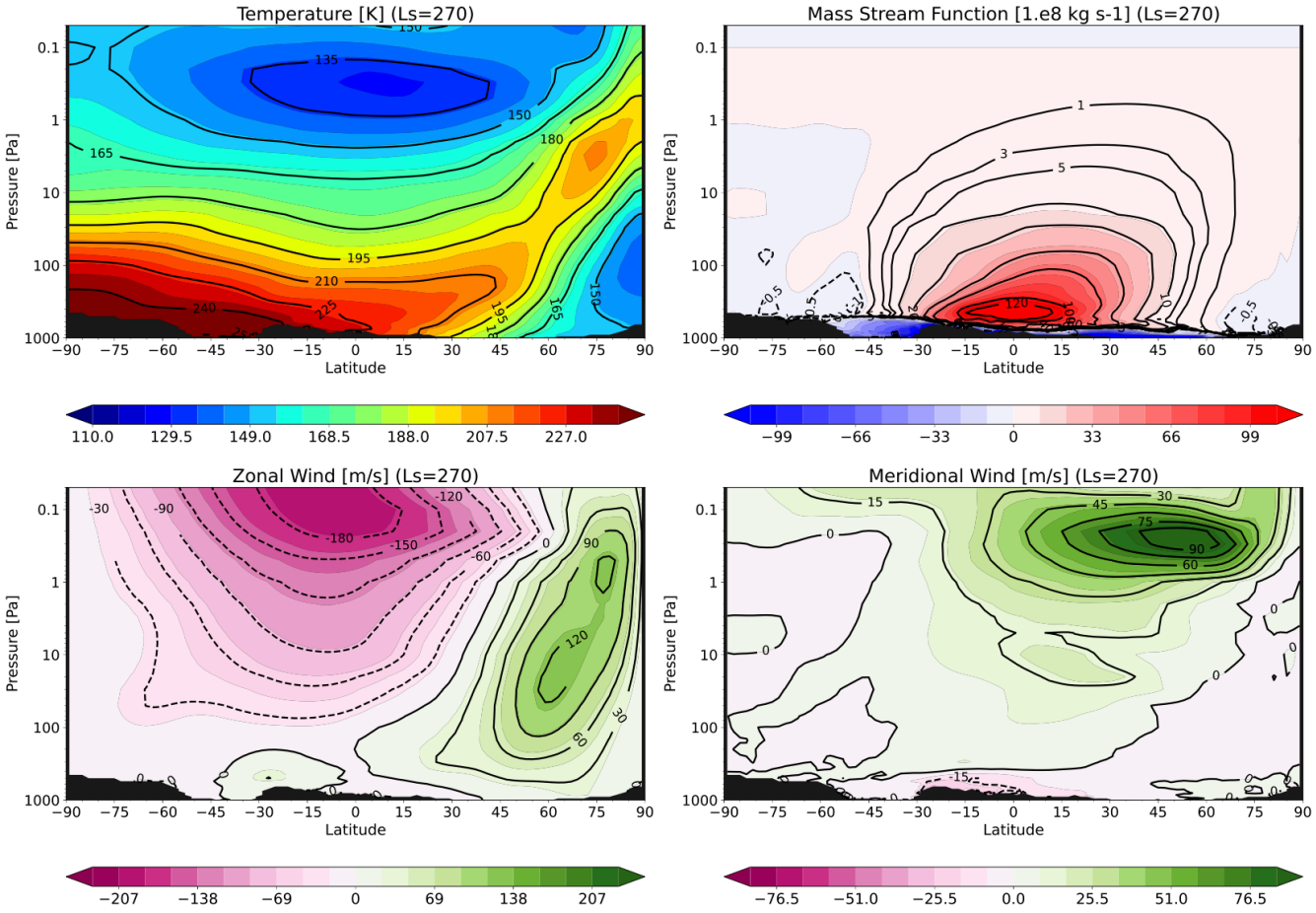
or

```
sftp> get Diagnostics.pdf # uses sftp
```

Return to Part II

Custom Set 4 of 4: Zonal Mean Circulation Cross-Sections

For our final set of plots, we will generate four cross-section plots showing temperature, zonal (U) and meridional (V) winds, and mass streamfunction at L_s=270.



Begin with the usual 3-step process:

1. Write a set of `HOLD ON` and `HOLD OFF` arguments
2. Copy-paste the `Plot 2D lat X lev` template between them
3. Set the template to `True`

Since all four plots are going to have the same X and Y axis ranges and `time` selection, let's edit this template before copying it three more times:

- Set `Ls 0-360 = 270`
- In `Axis Options`, set `Lat = [-90,90]`
- In `Axis Options`, set `level[Pa/m] = [1000,0.05]`

Now copy/paste this template three more times. Let the first plot be temperature, the second be mass streamfunction, the third be zonal wind, and the fourth be meridional wind.

For **temperature**:

```
Title          = Temperature [K] (Ls=270)
Main Variable  = 03847.atmos_average_Ls265_275_pstd.temp
Cmin, Cmax     = 110,240
...
2nd Variable   = 03847.atmos_average_Ls265_275_pstd.temp
```

For **streamfunction**, define explicit solid contours under `Contours Var 2` and set a diverging colormap.

```
Title          = Mass Stream Function [1.e8 kg s-1] (Ls=270)
Main Variable  = 03847.atmos_average_Ls265_275_pstd.msf
Cmin, Cmax     = -110,110
...
2nd Variable   = 03847.atmos_average_Ls265_275_pstd.msf
Contours Var 2 = -5,-3,-1,-0.5,1,3,5,10,20,40,60,100,120
# set cmap = bwr in Axis Options
```

For **zonal** and **meridional** wind, use the dual-toned colormap `PiYG`.

```
Title          = Zonal Wind [m/s] (Ls=270)
Main Variable  = 03847.atmos_average_Ls265_275_pstd.ucomp
Cmin, Cmax     = -230,230
...
2nd Variable   = 03847.atmos_average_Ls265_275_pstd.ucomp
# set cmap = PiYG in Axis Options
```

```
Title          = Meridional Wind [m/s] (Ls=270)
Main Variable  = 03847.atmos_average_Ls265_275_pstd.vcomp
Cmin, Cmax     = -85,85
...
2nd Variable   = 03847.atmos_average_Ls265_275_pstd.vcomp
# set cmap = PiYG in Axis Options
```

Save and quit the editor (`ESC-:wq`). pass `Custom.in` to `MarsPlot.py` , and pull it to your local computer:

```
(amesCAP)~$ MarsPlot.py Custom.in
# switch to the local terminal...
(local)~$ getpdf # uses scp
```

or

```
sftp> get Diagnostics.pdf # uses sftp
```

Return to Part II

End Credits

This concludes the practical exercise portion of the CAP tutorial. Please feel free to use these exercises as a reference when using CAP the future!

Written by Courtney Batterson, Alex Kling, and Victoria Hartwick. This document was created for the NASA Ames MGCM and CAP Tutorial held virtually November 13-15, 2023.

Questions, comments, or general feedback? Contact us.

Return to Top