NASA/TP- 20210023086

NASA Ames Mars Global Climate Model (GCM) Tutorial Legacy Version Tutorial

Melinda April Kahre
Ames Research Center

November 2021

NASA/TP- 20210023086

NASA Ames Mars Global Climate Model (GCM) Tutorial Legacy Version Tutorial

Melinda April Kahre
Ames Research Center

National Aeronautics and
Space Administration

Ames Research Center

November 2021

Table of Contents

Practical: Using the Community Analysis Pipeline (CAP)

Recap: CAP is a Python toolkit designed to simplify post-processing and plotting MGCM output. Specifically, CAP consists of five Python executibles indended to perform the following functions:

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` Visualizing the MGCM output

When learning to use CAP, it is useful to divide its functions into three categories and explore them in order:

1. Retrieving Data
2. File Manipulations
3. Plotting Routines

We will practice using CAP for all three parts. You already have experience using CAP for Retrieving Data, which was covered at the end of the CAP installation instructions (the install asked you to use `MarsPull` to retrieve several `fort.11` files before the tutorial). Here, you will have a chance to practice using all five Python routines in CAP.

Activate CAP

As always with CAP, you must activate the `amesGCM3` virtual environment to access the executibles (you can revisit the installation instructions as a refresher).

```
(local)>$ source ~/amesGCM3/bin/activate      # bash
(local)>$ source ~/amesGCM3/bin/activate.csh  # csh/tcsh
```

As a reminder, each Mars executable has a `--help` argument (`-h` for short) that can show you information about an executible, for example:

```
(amesGCM3)>$ MarsPull.py -h
```

1. Retrieving Data

Using `MarsPull.py` to download MGCM output

`MarsPull` is a utility for accessing MGCM output files hosted on the MCMC Data portal. During the installation, you were asked to use `MarsPull` to download several `fort.11` files into your `INERTCLDS/` and `ACTIVECLDS/` directories. You should have already downloaded the necessary `fort.11` files for this tutorial. If you haven't, you can do so now by following along with the instructions below.

We asked that you create a `CAP_Tutorial` directory containing two subdirectories, `INERTCLDS/` and `ACTIVECLDS/` , in `amesGCM3/` :

```
(amesGCM3)>$ cd ~/amesGCM3
(amesGCM3)>$ mkdir CAP_Tutorial
(amesGCM3)>$ cd CAP_Tutorial
(amesGCM3)>$ mkdir ACTIVECLDS INERTCLDS
```

Navigate to `INERTCLDS/` and use `MarsPull` to retrieve the files. Specify the simulation identifier (`INERTCLDS/`) and the range of Solar Longitudes (255 285) corresponding to the desired file(s):

```
(amesGCM3)>$ MarsPull.py -id INERTCLDS -ls 255 285
```

Then, do the same for the `ACTIVECLDS/` case.

There should now be 5 `fort.11` files in each directory, `INERTCLDS/` and `ACTIVECLDS/` :

```
> fort.11_0719 fort.11_0720 fort.11_0721 fort.11_0722 fort.11_0723
```

> If you have any `fort.11` files *other than the ones listed above* in *either* directory, please delete them. It will be make it easier to follow the tutorial if you work with the specific subset of files listed above.

2. File Manipulations

After retrieving output from the data portal or using output from a simulation you ran yourself, you will likely need to process the data to create the files you need for your analysis. Post-processing includes interpolating and regridding data to different vertical coordinate systems, adding derived variables to the files, and converting between filetypes, just to name a few examples.

The following exercises are designed to demonstrate how CAP can be used for post-processing MGCM output. You should follow along in the directories you created containing the `fort.11` files you downloaded during the installation process. After post-processing these files, **we will use them to make plots with MarsPlot**. Don't delete anything!

Start with the radiatively inert clouds simulation (RIC), `INERTCLDS/`, and complete exercises 2.1-2.8 below. Then we will give you specific instructions regarding which exercises to repeat for the radiatively active clouds (RAC) simulation `ACTIVECLDS/`. We access files from both simulations to make plots in Section 3.

2.1 Convert the `fort.11` files into `netCDF` files for compatibility with CAP.

To do this, go to your `INERTCLDS/` directory, and type:

```
(amesGCM3)>$ MarsFiles.py fort.11_* -fv3 fixed average daily diurn
```

This created several `netCDF` files:

```
(amesGCM3)>$ ls
> 07180.atmos_average.nc  07190.atmos_average.nc  07200.atmos_average.nc  07210.atmos_av
> 07180.atmos_daily.nc    07190.atmos_daily.nc    07200.atmos_daily.nc    07210.atmos_da
> 07180.atmos_diurn.nc    07190.atmos_diurn.nc    07200.atmos_diurn.nc    07210.atmos_di
> 07180.fixed.nc          07190.fixed.nc          07200.fixed.nc          07210.fixed.n
```

> Note the five-digit sol numbers at the begining of each netcdf file, which corresponds to the time at the begining of each fort.11 output. Because the simulation is issued from a 10 year run (10 x ~668 sols/year), this particular series of outputs start at 06690, not 00000.

The `netCDF` filetypes are:

Type	Description
`*atmos_fixed.nc`	static variables that **do not change over time**

Type	Description
`*atmos_average.nc`	**5-day averages** of MGCM output
`*atmos_diurn.nc`	files contain **hourly** MGCM output averaged over 5 days
`*atmos_daily.nc`	**continuous time series** of the MGCM output

For easier post-processing and plotting, we can combine like files along the time axis. This creates one of each filetype:

```
(amesGCM3)>$ MarsFiles.py *fixed.nc -c
(amesGCM3)>$ MarsFiles.py *average.nc -c
(amesGCM3)>$ MarsFiles.py *diurn.nc -c
(amesGCM3)>$ MarsFiles.py *daily.nc -c
```

This merge created the following four files:

```
> 07180.atmos_fixed.nc 07180.atmos_average.nc 07180.atmos_diurn.nc 07180.atmos_daily.nc
```

2.2 Interpolate `atmos_average` to standard pressure coordinates.

This requires using `MarsInterp`. As a reminder, you can display documentation for MarsInterp using:

```
(amesGCM3)>$ MarsInterp.py -h
```

Convert to standard pressure coordinates by entering the following:

```
(amesGCM3)>$ MarsInterp.py 07180.atmos_average.nc -t pstd
```

which creates:

```
> 07180.atmos_average_pstd.nc
```

2.3 Add density (`rho`) and mid-point altitude (`zfull`) to `atmos_average` , then interpolate the file to standard altitude (`zstd`)

Adding or removing variables from files can be done with `MarsVars` :

```
(amesGCM3)>$ MarsVars.py -h # display documentation
(amesGCM3)>$ MarsVars.py 07180.atmos_average.nc -add rho zfull
```

This updates the original file to include the new variables. In this case, the density `rho` was derived from the pressure and temperature (which are already present in the file) and the mid-point altitude `zfull` was obtained through hydrostatic integration.

> **NOTE: if you want `rho` in an interpolated file, you need to add it before performing the interpolation because. In this case, we want `rho` in an altitude-interpolated file so we've added `rho` to the original file (`atmos_average.nc`) and we will perform the interpolation next .**

```
(amesGCM3)>$ MarsInterp.py 07180.atmos_average.nc -t zstd   # standard altitude
```

Now our directory contains three `atmos_average` files:

```
> 07180.atmos_average.nc 07180.atmos_average_pstd.nc 07180.atmos_average_zstd.nc
```

To see the variables in each file, use the `--inspect` function from `MarsPlot` :

```
(amesGCM3)>$ MarsPlot.py -i 07180.atmos_average.nc          # the original file, note tl
(amesGCM3)>$ MarsPlot.py -i 07180.atmos_average_zstd.nc     # the pressure interpolated
(amesGCM3)>$ MarsPlot.py -i 07180.atmos_average_pstd.nc     # the altitude interpolated
```

2.4 Add mass stream function (`msf`) to `atmos_average_pstd` .

In this case, we add the variable after the interpolation because the mass stream function needs to be computed on a standard pressure grid.

```
(amesGCM3)>$ MarsVars.py 07180.atmos_average_pstd.nc -add msf
```

2.5 Use `MarsFiles` to time-shift the diurn file, then pressure-interpolate the file.

The variables in `07180.atmos_diurn.nc` are organized by time-of-day in universal time at the prime martian meridian, but you can time-shift the fields to uniform local time using `MarsFiles` . You might use this function to allow plotting global variables at 3 AM and 3 PM, for example. We will only retain the surface pressure `ps` , surface temperature `ts` and atmospheric temperature `temp` using `--include` to minimize the size of the file and processing time.

```
(amesGCM3)>$ MarsFiles.py 07180.atmos_diurn.nc -t --include ts ps temp
```

This function can only be performed on `diurn` files, since only `diurn` files contain hourly output. This function creates a new, time-shifted file, `07180.atmos_diurn_T.nc` . Next, pressure interpolate the file using `MarsInterp` (like we did for `atmos_average`).

```
(amesGCM3)>$ MarsInterp.py 07180.atmos_diurn_T.nc -t pstd
```

This should take just over a minute. Note that pressure interpolating large files can take a long time which is why we only included `ps` , `ts` , and `temp` in this file. We now have three diurn filetypes:

```
> 07180.atmos_diurn.nc 07180.atmos_diurn_T.nc 07180.atmos_diurn_T_pstd.nc
```

> **Note:** We will *not* do this here, but you can specify a vertical grid to interpolate to with CAP. See the documentation for `MarsInterp.py` to learn how.

2.6 Apply a low-pass filter (`-lpf`) to the surface pressure (`ps`) and temperature (`ts`) in the `atmos_daily` with a 10 sols cut-off frequency (set `sol_max` > 10) to isolate synoptic-scale feature.

This will filter-out the pressure and save the variable in a new file:

```
(amesGCM3)>$ MarsFiles.py 07180.atmos_daily.nc -lpf 10 -include ps ts
```

2.7 Estimate the magnitude of the wind shear using CAP. Add dU/dZ and dV/dZ to `07180.atmos_average_zstd.nc` .

In addition of adding new variables, `MarsVars` can apply certain operations such as column integration or vertical differentiation to existing variables. Vertical differentiation can be done as follows:

```
(amesGCM3)>$ MarsVars.py 07180.atmos_average_zstd.nc -zdiff ucomp vcomp
```

You can use `--inspect` (`-i`) to find the names of the derived variables dU/dZ and dV/dZ:

```
(amesGCM3)>$ MarsPlot.py -i 07180.atmos_average_zstd.nc
> ===================DIMENSIONS==========================
> ['lat', 'lon', 'phalf', 'time', 'zstd']
> (etc)
> ====================CONTENT==========================
> (etc)
> d_dz_ucomp     : ('time', 'zstd', 'lat', 'lon')= (10, 45, 36, 60), vertical gradient
> d_dz_vcomp     : ('time', 'zstd', 'lat', 'lon')= (10, 45, 36, 60), vertical gradient
> (etc)
>
> Ls ranging from 255.42 to 284.19: 45.00 days
>                (MY 01)   (MY 01)
> =====================================================
```

> **The `--inspect` function works on any netCDF file, not just the ones created here!**

2.8 Display the minimum, mean, and maximum near-surface temperature .

We can display values in an array by calling `--dump` with `MarsPlot -i` (analogue of the NCL command `ncdump`). For example, the content for the reference pressure (`pfull` variable in the file) is:

```
(amesGCM3)>$ MarsPlot.py -i 07180.atmos_average.nc -dump pfull
> pfull=
> [8.7662227e-02 2.5499690e-01 5.4266089e-01 1.0518962e+00 1.9545468e+00
> 3.5580616e+00 6.2466631e+00 1.0509957e+01 1.7400265e+01 2.8756382e+01
> 4.7480076e+01 7.8348366e+01 1.2924281e+02 2.0770235e+02 3.0938846e+02
> 4.1609518e+02 5.1308148e+02 5.9254102e+02 6.4705731e+02 6.7754218e+02
> 6.9152936e+02 6.9731799e+02 6.9994830e+02 7.0082477e+02]
> ______________________________________________________________________
```

We can also index specific values using quotes and square brackets `'[ ]'` . For example, we can display the reference pressure in the first layer above the surface (we use `-1` to refer to the last array element per Python convention):

```
(amesGCM3)>$ MarsPlot.py -i 07180.atmos_average.nc -dump 'pfull[-1]'
> pfull[-1]=
> 700.8247680664062
> ______________________________________________________________________
```

`-stat` display the min, mean, and max values of a variable, which is better suited to display statistics over a large array or for specific data-slices. For example, to display the min, mean, and max air temperature for all timesteps, all latitudes, all longitudes, and near the surface (`[time,pfull,lat,lon]=[:,-1,:,:]`), we use:

```
(amesGCM3)>$ MarsPlot.py -i 07180.atmos_average.nc -stat 'temp[:,-1,:,:]'
_________________________________________________________________________
             VAR               |      MIN       |      MEAN      |      MAX       |
_______________________________|________________|________________|________________|
                  temp[:,-1,:,:]|        149.016|        202.508|         251.05|
_______________________________|________________|________________|________________|
```

> **Note:** quotes '' are necessary when browsing dimensions.

Remember to repeat this post-processing on the `ACTIVECLDS/` simulation as well!

Break!

Let's take a 15 minute break from the tutorial. You can use this time to catch up if you haven't completed parts 1 and 2 already, but we highly encourage you to step away from your machine for these 15 minutes.

3. Plotting Routines

The last part of this tutorial covers the plotting capabilities within CAP. CAP can create several kinds of plots:

Type of plot	MarsPlot designation
Longitude v Latitude	Plot 2D lon X lat
Longitude v Time	Plot 2D lon X time
Longitude v Level	Plot 2D lon X lev
Latitude v Level	Plot 2D lat X lev
Time v Latitude	Plot 2D time X lat
Time v level	Plot 2D time X lev
Any 1-dimensional line plot	Plot 1D

and CAP can display each plot on its own page or place multiple plots on the same page.

Plotting with CAP requires passing a template to `MarsPlot`. A blank template is created in the directory in which the following command is executed, so change to the `INERTCLDS/` directory and type:

```
(amesGCM3)>$ MarsPlot.py -template
```

The blank template is called `Custom.in`. Pass `Custom.in` back to `MarsPlot` using the following command:

```
(amesGCM3)>$ MarsPlot.py Custom.in
```

This will have created `Diagnostics.pdf`, a single-page PDF with a topographical plot and a cross-section of the zonal mean wind. Open the pdf to see the plots.

> You can rename `Custom.in` and still pass it to `MarsPlot` successfully:

```
(amesGCM3)>$ mv Custom.in myplots.in
(amesGCM3)>$ MarsPlot.py myplots.in
```

If the template is named anything other than `Custom.in`, `MarsPlot` will produce a PDF named after the renamed template, i.e. `myplots.pdf`.

Those are the basics of plotting with CAP. We'll try creating several plot types in exercises 3.8--3.8 below.

3.1 Plot a global map of surface albedo (`alb`) with topography (`zsurf`) contoured on top.

For this first plot, we'll edit `Custom.in` together. Open the template in your preferred text editor and make the following changes:

- Change the second default template `Plot 2D lat X lev` to `False` so that `MarsPlot` does not draw it (we will use it later)
- Set the `Title` of the first default template `Plot 2D lon X lat` to reflect the variable being plotted.
- Set `Main Variable` to albedo (`alb`, located in the `fixed` file), this will be plotted as shaded contours

- Set `2nd Variable` to topography (`zsurf` , located in the `fixed` file), this will be plotted as solid contours

Here is what your template should look like:

```
<<<<<<<<<<<<<<| Plot 2D lon X lat = True |>>>>>>>>>>>>>
Title             = 3.1: Albedo w/Topography Overplotted
Main Variable  = fixed.alb
Cmin, Cmax     = None
Ls 0-360       = None
Level [Pa/m]   = None
2nd Variable   = fixed.zsurf
Contours Var 2 = None
Axis Options  : lon = [None,None] | lat = [None,None] | cmap = binary | scale = lin | p
```

Save the template in your text editor and pass it back to `MarsPlot` :

```
(amesGCM3)>$ MarsPlot.py Custom.in
```

Open `Diagnostics.pdf` and check to make sure it contains a global map of surface albedo and topography.

> Depending on the settings for your specific pdf viewer, you may have to close and open the file.

3.2 Next, plot a cross-section of the zonal mean zonal wind at Ls=270° using altitude as the vertical coordinate.

No need to create a new template, just add this plot to `Custom.in` . Use the zonal wind stored in the `atmos_average_zstd` file. Remember to set the plot template to `True` . Edit the title accordingly.

Save `Custom.in` and pass it to `MarsPlot` .

3.3 Create the same plot for the radiatively active cloud case, and put both zonal mean zonal wind plots on their own page.

> Tip: Add to your existing template. Copy and paste the `lat x lev` plot three times. Set the plots to `True` so that `MarsPlot` recognizes them as input.

Edit the `<<<<<<< Simulations <<<<<<<` section so that `2>` points to the `/ACTIVECLDS` directory:

```
<<<<<<<<<<<<<<<<<<<<<< Simulations >>>>>>>>>>>>>>>>>>>>>
ref> None
2> ../ACTIVECLDS
```

Then, copy and paste the plot created in 3.2 and edit `Main Variable` to point to the correct directory using the `@N` syntax:

```
Main Variable  = atmos_average@2.ucomp
```

> Tip: Make use of `HOLD ON` and `HOLD OFF` for these, and Copy/Paste plot types to create multiple of the same plot.

Save `Custom.in` and pass it to `MarsPlot`.

3.4 Add temperature as solid contours overtop of the zonal wind plot.

Add `temp` as a second variable on the plots you created in 3.2 and 3.3:

```
> 2nd Variable     = atmos_average_zstd.temp
```

Save `Custom.in` and pass it to `MarsPlot`.

3.5 Plot the following four global maps (`lon x lat`) on a new page:

> Tip: Use `HOLD ON` and `HOLD OFF`. You can use this syntax multiple times in the same template.

All of the following variables come from `07180.atmos_daily.nc` and should be plotted at Ls=270.

- Surface CO2 ice content (`snow`) *north of 50 latitude*
- Surface temperature (`ts`) *For this plot, set the colorscale (* `Cmin, Cmax` *) to range from 150 K to 300 K.*
- Surface Wind Speed (`(u^2 + v^2)/2`) (this requires the use of square brackets **and** two variables)
- Diabatic Heating Rate (`dheat`) at 50 Pa (index dimension `lev` =50).

The general format will be:

```
HOLD ON

<<<<<<| Plot 2D lon X lat = True |>>>>>>
Title    = Surface CO2 Ice (g/m2)
(etc)

<<<<<<| Plot 2D lon X lat = True |>>>>>>
Title    = Surface Temperature (K)
(etc)

<<<<<<| Plot 2D lon X lat = True |>>>>>>
Title    = Surface Wind Speed (m/s)
(etc)

<<<<<<| Plot 2D lon X lat = True |>>>>>>
Title    = Diabatic Heating Rate (K/sol)
(etc)

HOLD OFF
```

> *Note:* convert kg -> g using square brackets:
>
> ```
> Main Variable  = [atmos_daily.snow]*1000
> ```
>
> and multiply two variables together like so:
>
> ```
> Main Variable  = ([atmos_daily.ucomp]**2+[atmos_daily.vcomp]**2)**0.5
> ```

Name the plots accordingly. Save `Custom.in` and pass it to `MarsPlot`.

3.6 Plot the following two cross-sections (`lat x lev`) on the same page:

- Mass Streamfunction (`msf`) at Ls=270. Change the colormap from `jet` to `bwr` and force symmetrical contouring by setting the colorbar's minimum and maximum values to -50 and 50. Adjust the y axis limits to 1,000 Pa and 1 Pa. Finally, add solid contours for `msf` =-10 and `msf` =10 on top. *Hint: set both `Main Variable` and `2nd Variable` to `msf`*
- Zonal mean temperature (`temp`) at Ls=270 from the same (pressure-interpolated) file. Overplot the zonal wind (`ucomp`).

Don't forget to use `HOLD ON` and `HOLD OFF` and to name your plots accordingly. Save `Custom.in` and pass it to `MarsPlot`.

3.7 Plot the zonal mean temperature at Ls=270 from the average file for the inert cloud case and the active cloud case. Also create a difference plot for them.

Use `HOLD ON` and `HOLD OFF`. Copy and paste a `lat x lev` plot three times. For the difference plot, you'll need to use `@N` to point to the `ACTIVECLDS/` directory and square brackets to subtract one variable from the other:

```
Main Variable  = [atmos_average_pstd.temp]-[atmos_average_pstd@2.temp]
```

Set the colormap to `RdBu` for the difference plot and set the vertical range to 1,000-1 Pa.

Save `Custom.in` and pass it to `MarsPlot`.

3.8 Generate a 1D temperature profile (`temp`) at `50°N, 150°E` at Ls=270 at both 3 AM and 3 PM from the radiatively inert case. Plot these on the same plot.

CAP can overplot 1D data on the same graph by concatenating two 1D templates together with `ADD LINE`:

```
<<<<<<| Plot 1D = True |>>>>>>
Main Variable    = var1
(etc)

ADD LINE

<<<<<<| Plot 1D = True |>>>>>>
Main Variable    = var2
(etc)
```

> You do not need to use `HOLD ON` or `HOLD OFF` with 1D plots.

You'll need to call `temp` from the `diurn_T_pstd` file, which is the time-shifted and pressure-interpolated version of the hourly file. 3 AM is index=3, 3 PM is index=15. You will have to specify `Level [Pa/m]` as the y axis:

```
Level [Pa/m]   = AXIS
```

Save `Custom.in` and pass it to `MarsPlot`.

3.9 Plot the filtered and un-filtered surface pressure over a 20 sol period.

Some hints:

- Both are 1D plots. Use `ADD LINE` to plot on the same axes
- Use `ps` from the `07180.atmos_daily.nc` and `07180.atmos_daily_lpf.nc` files
- Index noon `{tod=12}`
- Set `Latitude = 50` and `Lon +/-180 = 150`
- Under `Axis Options`, set the x axis range (time) to 260--280 (`sols = [260, 280]`)
- Under `Axis Options`, set the y axis range (pressure) to 850Pa--1000Pa(`var = [850, 1000]`)

Save `Custom.in` and pass it to `MarsPlot`.

That's a Wrap!

This concludes the practical exercise portion of the CAP tutorial. Please keep these exercises as a reference for the future!

This document was completed in October 2021. Written by Alex Kling, Courtney Batterson, and Victoria Hartwick

Please submit feedback to Alex Kling: alexandre.m.kling@nasa.gov

Installing the Community Analysis Pipeline (CAP)

Welcome!

This document contains the instructions for installing the NASA Ames MCMC's Community Analysis Pipeline (CAP). **We ask that you come to the MGCM Tutorial on November 2-4 with CAP installed on your machine** so that we can jump right into using it! On the second day of the tutorial (November 3rd), we will be using CAP to analyze MGCM output.

Installing CAP is fairly straightforward. We will create a Python virtual environment, download CAP, and then install CAP in the virtual environment. That's it!

A quick overview of what is covered in this installation document:

1. Creating the Virtual Environment

We begin by creating a virtual environment in which to install CAP. The virtual environment is an isolated Python environment cloned from an existing Python distribution. The virtual environment consists of the same directory trees as the original environment, but it includes activation and deactivation scripts that are used to move in and out of the virtual environment. Here's an illustration of how the two Python environments might differ:

```
    anaconda3                    virtual_env3/
    ├── bin                      ├── bin
    │   ├── pip       (copy)     │    ├── pip
    │   └── python3    >>>>      │    ├── python3
    └── lib                      │    ├── activate
                                 │    ├── activate.csh
                                 │    └── deactivate
                                 └── lib

 ORIGINAL ENVIRONMENT           VIRTUAL ENVIRONMENT
     (untouched)            (vanishes when deactivated)
```

We can install and upgrade packages in the virtual environment without breaking the main Python environment. In fact, it is safe to change or even completely delete the virtual environment without breaking the main distribution. This allows us to experiment freely in the virtual environment, making it the perfect location for installing and testing CAP.

Step 1: Identify Your Preferred Python Distribution

If you are already comfortable with Python's package management system, you are welcome to install the pipeline on top any python**3** distribution already present on your computer. Jump to Step #2 and resolve any missing package dependency.

For all other users, we highly recommend using the latest version of the Anaconda Python distribution. It ships with pre-compiled math and plotting packages such as `numpy` and `matplotlib` as well as pre-compiled libraries like `hdf5` headers for reading `netCDF` files (the preferred filetype for analysing MGCM output).

You can install the Anaconda Python distribution via the command-line or using a graphical interface (scroll to the very bottom of the page for all download options). You can install Anaconda at either the `System/` level or the `User/` level (the later does not require admin-priviledges). The instructions below are for the **command-line installation** and installs Anaconda in your **home directory**, which is the recommended location. Open a terminal and type the following:

```
(local)>$ chmod +x Anaconda3-2021.05-MacOSX-x86_64.sh   # make the .sh file executable (actual name may diffe
(local)>$ ./Anaconda3-2021.05MacOSX-x86_64.sh           # runs the executable
```

Which will return:

```
> Welcome to Anaconda3 2021.05
>
> In order to continue the installation process, please review the license agreement.
> Please, press ENTER to continue
> >>>
```

Read (`ENTER`) and accept (`yes`) the terms, choose your installation location, and initialize Anaconda3:

```
(local)>$ [ENTER]
> Do you accept the license terms? [yes|no]
> >>>
(local)>$ yes
> Anaconda3 will now be installed into this location:
> /Users/username/anaconda3
>
>  - Press ENTER to confirm the location
>  - Press CTRL-C to abort the installation
>  - Or specify a different location below
>
> [/Users/username/anaconda3] >>>
(local)>$ [ENTER]
> PREFIX=/Users/username/anaconda3
> Unpacking payload ...
> Collecting package metadata (current_repodata.json):
>   done
> Solving environment: done
>
> ## Package Plan ##
> ...
> Preparing transaction: done
> Executing transaction: -
> done
> installation finished.
> Do you wish the installer to initialize Anaconda3 by running conda init? [yes|no]
> [yes] >>>
(local)>$ yes
```

> For Windows users, we recommend installing the pipeline in a Linux-type environment using Cygwin. This will enable the use of CAP command line tools. Simply download the Windows version of Anaconda on the Anaconda website and follow the instructions from the installation GUI. When asked about the installation location, make sure you install Python under your emulated-Linux home directory (`/home/username`) and ***not*** in the default location (`/cygdrive/c/Users/username/anaconda3`). From the installation GUI, the path you want to select is something like:

> `C:/Program Files/cygwin64/home/username/anaconda3`. Also be sure to check **YES** when prompted to "Add Anaconda to my `PATH` environment variable."

Confirm that your path to the Anaconda Python distribution is fully actualized by closing out of the current terminal, opening a new terminal, and typing:

```
(local)>$ python[TAB]
```

If this returns multiple options (e.g. `python`, `python2`, `python 3.7`, `python.exe`), then you have more than one version of Python sitting on your system (an old `python2` executable located in `/usr/local/bin/python`, for example). You can see what these versions are by typing:

```
(local)>$ python3 --version     # Linux/MacOS
(local)>$ python.exe --version  # Cygwin/Windows
```

Check your version of `pip` the same way, then find and set your `$PATH` environment variable to point to the Anaconda Python *and* Anaconda pip distributions. If you are planning to use Python for other projects, you can update these paths like so:

```
# with bash:
(local)>$ echo 'export PATH=/Users/username/anaconda3/bin:$PATH' >> ~/.bash_profile
# with csh/tsch:
(local)>$ echo 'setenv PATH $PATH\:/Users/username/anaconda3/bin\:$HOME/bin\:.'  >> ~/.cshrc
```

Confirm these settings using the `which` command:

```
(local)>$ which python3         # Linux/MacOS
(local)>$ which python.exe      # Cygwin/Windows
```

which hopefully returns a Python executable that looks like **it was installed with Anaconda**, such as:

```
> /username/anaconda3/bin/python3     # Linux/MacOS
> /username/anaconda3/python.exe      # Cygwin/Windows
```

If `which` points to either of those locations, you are good to go and you can proceed from here using the shorthand path to your Anaconda Python distribution:

```
(local)>$ python3     # Linux/MacOS
(local)>$ python.exe  # Cygwin/Windows
```

If, however, `which` points to some other location, such as `/usr/local/bin/python`, or more than one location, proceed from here using the **full** path to the Anaconda Python distribution:

```
(local)>$ /username/anaconda3/bin/python3 # Linux/MacOS
(local)>$ /username/anaconda3/python.exe  # Cygwin/Windows
```

Step 2: Set Up the Virtual Environment:

Python virtual environments are created from the command line. Create an environment called `amesGCM3` by typing:

```
(local)>$ python3 -m venv --system-site-packages amesGCM3    # Linux/MacOS Use FULL PATH to python if needed
(local)>$ python.exe -m venv --system-site-packages amesGCM3  # Cygwin/Windows Use FULL PATH to python if nee
```

First, find out if your terminal is using *bash* or a variation of *C-shell* (*.csh*, *.tsch*…) by typing:

```
(local)>$ echo $0
> -bash
```

Depending on the answer, you can now activate the virtual environment with one of the options below:

```
(local)>$ source amesGCM3/bin/activate          # bash
(local)>$ source amesGCM3/bin/activate.csh      # csh/tcsh
(local)>$ source amesGCM3/Scripts/activate.csh  # Cygwin/Windows
```

> In Cygwin/Windows, the `/bin` directory may be named `/Scripts`.

You will notice that after sourcing `amesGCM3`, your prompt changed indicate that you are now *inside* the virtual environment (i.e. `(local)>$` changed to `(amesGCM3)>$`).

We can verify that `which python` and `which pip` unambiguously point to `amesGCM3/bin/python3` and `amesGCM3/bin/pip`, respectively, by calling `which` within the virtual environment:

```
(amesGCM3)>$ which python3         # in bash, csh
> amesGCM3/bin/python3
(amesGCM3)>$ which pip
> amesGCM3/bin/pip

(amesGCM3)>$ which python.exe      # in Cygwin/Windows
> amesGCM3/Scripts/python.exe
(amesGCM3)>$ which pip
> amesGCM3/Scripts/pip
```

There is therefore no need to reference the full paths while **inside** the virtual environment.

---

2. Installing CAP

Now we can download and install CAP in `amesGCM3`. CAP was provided to you in the tarfile `amesgcm-master.zip` that was sent along with these instructions. Download `amesgcm-master.zip` and leave it in `Downloads/`.

Using `pip`

Open a terminal window, activate the virtual environment, and untar the file:

```
(local)>$ source ~/amesGCM3/bin/activate          # bash
(local)>$ source ~/amesGCM3/bin/activate.csh      # cshr/tsch
(local)>$ source ~/amesGCM3/Scripts/activate.csh  #  Cygwin/Windows
(amesGCM3)>$
(amesGCM3)>$ tar -xf amesgcm-master.zip
(amesGCM3)>$ cd amesgcm-master
(amesGCM3)>$ pip install .
```

> Please follow the instructions to upgrade pip if recommended during that steps.

That's it! CAP is installed in `amesGCM3` and you can see the `MarsXXXX.py` executables stored in `~/amesGCM3/bin/` :

```
(local)>$ ls ~/amesGCM3/bin/
> Activate.ps1      MarsPull.py      activate.csh              nc4tonc3        pip3
> MarsFiles.py      MarsVars.py      activate.fish             ncinfo          pip3.8
> MarsInterp.py     MarsViewer.py    easy_install              normalizer      python
> MarsPlot.py       activate         easy_install-3.8          pip             python3
```

> Shall you need to modify any code, note that when you access the `Mars` tools above, those are **not** executed from the `amesgcm-master/` folder in your `/Downloads` directory, but instead from the `amesGCM3` virtual environment where they were installed by pip. You can safely move amesgcm-master.zip and the amesgcm-master directory to a different location on your system.

Double check that the paths to the executables are correctly set in your terminal by exiting the virtual environment:

```
(amesGCM3)>$ deactivate
```

then reactivating the virtual environment:

```
(local)>$ source ~/amesGCM3/bin/activate     # bash
(local)>$ source ~/amesGCM3/bin/activate.csh # csh/tsch
(local)>$ source ~/amesGCM3/Scripts/activate.csh
```

and checking the documentation for any CAP executable using the `--help` option:

```
(amesGCM3)>$ MarsPlot.py --help
(amesGCM3)>$ MarsPlot.py -h
```

or using **full** paths:

```
(amesGCM3)>$ ~/amesGCM3/bin/MarsPlot.py -h      # Linux/MacOS
(amesGCM3)>$ ~/amesGCM3/Scripts/MarsPlot.py -h # Cygwin/Windows
```

If the pipeline is installed correctly, `--help` will display documentation and command-line arguments for `MarsPlot` in the terminal.

This completes the one-time installation of CAP in your virtual environment, `amesGCM3`, which now looks like:

```
amesGCM3/
├── bin
│   ├── MarsFiles.py
│   ├── MarsInterp.py
│   ├── MarsPlot.py
│   ├── MarsPull.py
│   ├── MarsVars.py
│   ├── activate
│   ├── activate.csh
│   ├── deactivate
│   ├── pip
│   └── python3
├── lib
│   └── python3.7
│       └── site-packages
│           ├── netCDF4
│           └── amesgcm
│               ├── FV3_utils.py
│               ├── Ncdf_wrapper.py
│               └── Script_utils.py
├── mars_data
│   └── Legacy.fixed.nc
└── mars_templates
    ├──amesgcm_profile
    └── legacy.in
```

Using `conda`

If you prefer using the `conda` package manager for setting up your virtual environment instead of `pip`, you may use the following commands to install CAP.

First, verify (using `conda info` or `which conda`) that you are using the intented `conda` executable (two or more versions of `conda` might be present if both Python2 and Python3 are installed on your system). Then, create the virtual environment with:

```
(local)>$ conda create -n amesGCM3
```

Activate the virtual environment, then install CAP:

```
(local)>$ conda activate amesGCM3
(amesGCM3)>$ conda install pip
(amesGCM3)>$ cd ~/Downloads
(amesGCM3)>$ tar -xf CAP_tarball.zip
(amesGCM3)>$ cd amesgcm-master
(amesGCM3)>$ pip install .
```

The source code will be installed in:

```
/path/to/anaconda3/envs/amesGCM3/
```

and the virtual environment may be activated and deactivated with `conda`:

```
(local)>$ conda activate amesGCM3
(amesGCM3)>$ conda deactivate
(local)>$
```

> **Note:** CAP requires the following Python packages, which were automatically installed with CAP:
>
> ```
> matplotlib        # the MatPlotLib plotting library
> numpy             # math library
> scipy             # math library and input/output for fortran binaries
> netCDF4 Python    # handling netCDF files
> requests          # downloading GCM output from the MCMC Data Portal
> ```

Removing CAP

To permanently remove CAP, activate the virtual environment and run the `uninstall` command:

```
(local)>$ source amesGCM3/bin/activate          # bash
(local)>$ source amesGCM3/bin/activate.csh      # csh/tcsh
(local)>$ source amesGCM3/Scripts/activate.csh  # Cygwin/Windows
(amesGCM3)>$ pip uninstall amesgcm
```

You may also delete the `amesGCM3` virtual environment directory at any time. This will uninstall CAP, remove the virtual environment from your machine, and will not affect your main Python distribution.

3. Testing & Using CAP

Whenever you want to use CAP, simply activate the virtual environment and all of CAP's executables will be accessible from the command line:

```
(local)>$ source amesGCM3/bin/activate          #   bash
(local)>$ source amesGCM3/bin/activate.csh      #   csh/tcsh
(local)>$ source amesGCM3/Scripts/activate.csh  #   Cygwin/Windows
```

You can check that the tools are installed properly by typing `Mars` and then pressing the **TAB** key. No matter where you are on your system, you should see the following pop up:

```
(amesGCM3)>$ Mars[TAB]
> MarsFiles.py   MarsInterp.py  MarsPlot.py    MarsPull.py    MarsVars.py
```

If no executables show up then the paths have not been properly set in the virtual environment. You can either use the full paths to the executables:

```
(amesGCM3)>$ ~/amesGCM3/bin/MarsPlot.py
```

Or set up aliases in your `./bashrc` or `.cshrc` :

```
# with bash:
(local)>$ echo alias MarsPlot='/Users/username/amesGCM3/bin/MarsPlot.py' >> ~/.bashrc
(local)>$ source ~/.bashrc

# with csh/tsch
(local)>$ echo alias MarsPlot /username/amesGCM3/bin/MarsPlot >> ~/.cshrc
(local)>$ source ~/.cshrc
```

4. Practical Tips for Later Use During the Tutorial

Install `ghostscript` to Create Multiple-Page PDFs When Using `MarsPlot`

Installing `ghostscript` on your local machine allows CAP to generate a multiple-page PDF file instead of several individual PNGs when creating several plots. Without `ghostscript`, CAP defaults to generating multiple `.png` files instead of a single PDF file, and we therefore strongly recommend installing `ghostscript` to streamline the plotting process.

First, check whether you already have `ghostscript` on your machine. Open a terminal and type:

```
(local)>$ gs -version
> GPL Ghostscript 9.54.0 (2021-03-30)
> Copyright (C) 2021 Artifex Software, Inc.  All rights reserved.
```

If `ghostscript` is not installed, follow the directions on the `ghostscript` website to install it.

Enable Syntax Highlighting for the Plot Template

The `MarsPlot` executable requires an input template with the `.in` file extension. We recommend using a text editor that provides language-specific (Python) syntax highlighting to make keywords more readable. A few options include: Atom and vim (compatible with MacOS, Windows, Linux), notepad++ (compatible with Windows), or gedit (compatible with Linux).

The most commonly used text editor is vim. Enabling proper syntax-highlighting for Python in **vim** can be done by adding the following lines to `~/.vimrc`:

```
syntax on
colorscheme default
au BufReadPost *.in  set syntax=python
```

5. Do This Before Attending the Tutorial

In order to follow along with the practical part of the MGCM Tutorial, we ask that you **download several MGCM output files beforehand**. You should save these on the machine you'll be using during the tutorial.

We'll use CAP to retrieve these files from the MGCM Data Portal. To begin, activate the virtual environment:

```
(local)>$ source amesGCM3/bin/activate      # bash
(local)>$ source amesGCM3/bin/activate.csh  # csh/tcsh
```

Choose a directory in which to store these MGCM output files on your machine. We will also create two sub- directories, one for an MGCM simulation with radiatively inert clouds (RIC) and one for an MGCM simulation with radiatively active clouds (RAC):

```
(amesGCM3)>$ mkdir CAP_tutorial
(amesGCM3)>$ cd CAP_tutorial
(amesGCM3)>$ mkdir INERTCLDS ACTIVECLDS
```

Then, download the corresponding data in each directory:

```
(amesGCM3)>$ cd INERTCLDS
(amesGCM3)>$ MarsPull.py -id INERTCLDS -ls 255 285
(amesGCM3)>$ cd ../ACTIVECLDS
(amesGCM3)>$ MarsPull.py -id ACTIVECLDS -ls 255 285
```

That's it! `CAP_tutorial` now holds the necessary `fort.11` files from the radiatively active and inert MGCM simulations:

```
CAP_tutorial/
├── INERTCLDS/
│   └── fort.11_0719  fort.11_0720  fort.11_0721  fort.11_0722  fort.11_0723
└── ACTIVECLDS/
    └── fort.11_0719  fort.11_0720  fort.11_0721  fort.11_0722  fort.11_0723
```

You can now deactivate the virtual environment:

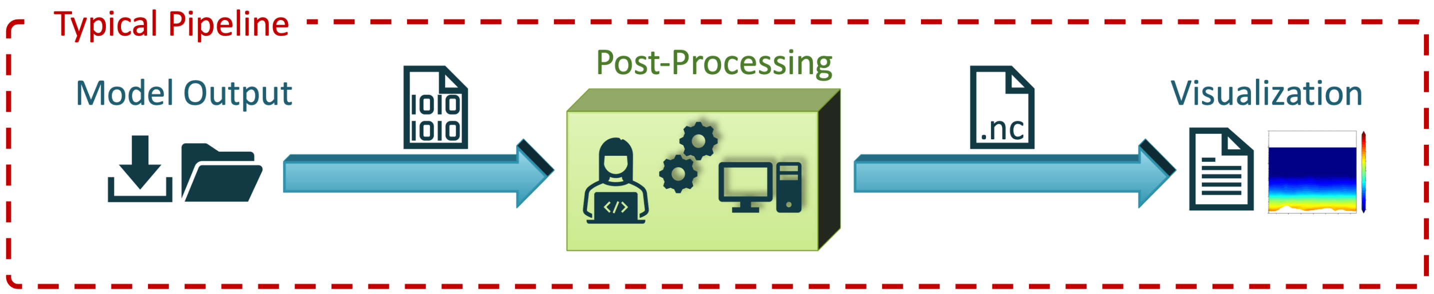
```
(amesGCM3)>$ deactivate
```

> If you encounter an issue during the download process, please verify the files availability on the MCMC Data Portal and try again later. You may also download the 10 files listed above manually.

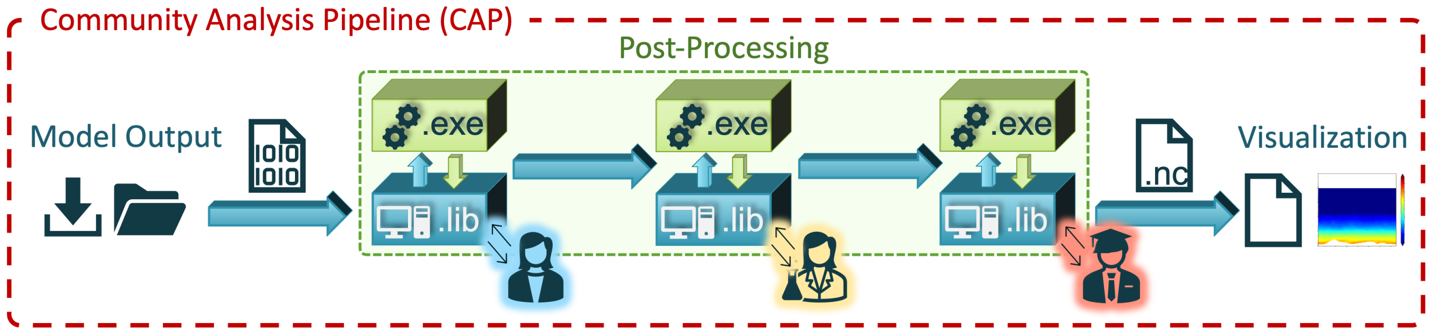
and we'll see you **November 2, 2021** for the tutorial!

Introducing the Community Analysis Pipeline (CAP)

CAP is a toolkit designed to simplify the post-processing of MGCM output. CAP is written in Python and works with existing Python libraries, allowing any Python user to install and use CAP easily and free of charge. Without CAP, plotting MGCM output requires that a user provide their own scripts for post-processing, including code for interpolating the vertical grid, computing and adding derived variables to files, converting between file types, and creating diagnostic plots. In other words, a user would be responsible for the entire post-processing effort as illustrated in Figure 1.



Such a process requires that users be familiar with Fortran files and be able to write (or provide) script(s) to perform file manipulations and create plots. At best, this effort is cumbersome. At worst, it excludes users who lack access to (or knowledge of how to write) post-processing scripts and/or Fortran code. CAP standardizes the post-processing effort by providing executibles that can perform file manipulations and create diagnostic plots from the command line. This enables users of almost any skill level to post-process and plot MGCM data (Figure 2).



Specifically, CAP consists of five executables:

1. `MarsPull.py` Access MGCM output
2. `MarsFiles.py` Reduce the files
3. `MarsVars.py` Perform variable operations
4. `MarsInterp.py` Interpolate the vertical grid
5. `MarsPlot.py` Visualize the MGCM output

and

These executables and their commonly-used functions are illustrated in the cheat sheet below in the order in which they are most often used. You should feel free to reference during and after the tutorial.

NASA AMES
MARS CLIMATE MODELING CENTER

Quick Start

Create, Source the Environment:
```
% /path/to/python3 -m venv -system-site-packages amesGCM3
% source ~/amesGCM3/bin/activate
```
Install CAP:
```
% pip install git+https://github.com/alex-kling/amesgcm.git
```
Update CAP:
```
% pip install git+https://github.com/alex-kling/amesgcm.git --upgrade
```
Uninstall CAP:
```
% pip uninstall amesgcm
```
Deactivate the Environment:
```
% deactivate
```

Frequently Used Commands

MarsPull.py
MarsFiles.py
MarsVars.py
MarsInterp.py
MarsPlot.py

Access MGCM Output
```
% MarsPull.py -id INERTCLDS -ls 255 285
% MarsPull.py -id INERTCLDS -f fort.11_0670 fort.11_0671
```

File Manipulations
```
% MarsFiles.py LegacyGCM_*.nc -fv3
% MarsFiles.py *atmos_average.nc -combine
% MarsFiles.py *atmos_diurn.nc -tshift
```

Variable Operations
```
% MarsVars.py *atmos_average.nc -col vap_mass
% MarsVars.py *atmos_average.nc -add rho
```

Interpolations
```
% MarsInterp.py *atmos_average.nc -t pstd
% MarsInterp.py *atmos_average.nc -t pstd -l phalf_mb
```

Visualizations and File Contents
```
% MarsPlot.py -template
% MarsPlot.py Custom.in
% MarsPlot.py -i 00000.atmos_average.nc
```

CAP is designed to be modular. For example, a user could post-process and plot MGCM output exclusively with CAP or a user could employ their own post-processing routine and then use CAP to plot the data. Users are free to selectively integrate CAP into their own analysis routine to the extent they see fit.

Table of Contents

The big question... How do I do this? > Ask for help!

Use the `--help` (`-h` for short) option on any executable to display documentation and examples.

```
(amesGCM3)>$ MarsPlot.py -h
> usage: MarsPlot.py [-h] [-i INSPECT_FILE] [-d DATE [DATE ...]] [--template]
>                    [-do DO] [-sy] [-o {pdf,eps,png}] [-vert] [-dir DIRECTORY]
>                    [--debug]
>                    [custom_file]
```

1. `MarsPull.py` - Downloading Raw MGCM Output

`MarsPull` is a utility for accessing MGCM output files hosted on the MCMC Data portal. MGCM data is archived in 1.5 hour intervals (16x/day, '*ntod*') and packaged in files containing 10 sols ('*time*') of data. The file naming convention is:

```
LegacyGCM_LsXXX_LsYYY.nc
```

Where XXX and YYY are three-digit Solar Longitude (L_s) values. The files can be retrieved from the command line using CAP by providing `MarsPull` with either a range of Solar Longitudes from which to pull data or a specific filename.

2. `MarsFiles.py` - Reducing the Files

`MarsFiles` provides several tools for file manipulations, including code designed to create binned, averaged, and time-shifted files from MGCM output. These are the file formats that `MarsFiles` can create from the fort.11 MGCM output files:

File name	description
* **fixed**	contains such as surface albedo and topography
* **average**	contains 5-sol averages of all variables
* **daily**	contains a continuous time series of data
* **diurn**	contains 5-day averaged data binned by time of day
* **_T**	contains time-shifted data (same time of day at all longitudes)
* **_lpf,_hpf,_bpf**	low, high and band pass filtered
* **_tidal**	tidally-decomposed files into harmonics
* **_to_average _to_diurn**	custom re-binning of daily files
* **_regrid**	4N-dimensional interpolation (lon,lat,time,altitude) to a different grid

`MarsFiles` can concatenate like-files together on the time dimension. `MarsFiles` can also be used to perform basic tidal analyses (temporal and spatial filtering, diurnal tides and their harmonics).

CAP is capable of applying high-, low-, and band-pass filters to netCDF files using the syntax:

```
(amesGCM3)>$ MarsFiles.py file.nc -hpf --high_pass_filter sol_min
(amesGCM3)>$ MarsFiles.py file.nc -lpf --low_pass_filter  sol_max
(amesGCM3)>$ MarsFiles.py file.nc -bpf --band_pass_filter sol_min sol max
```

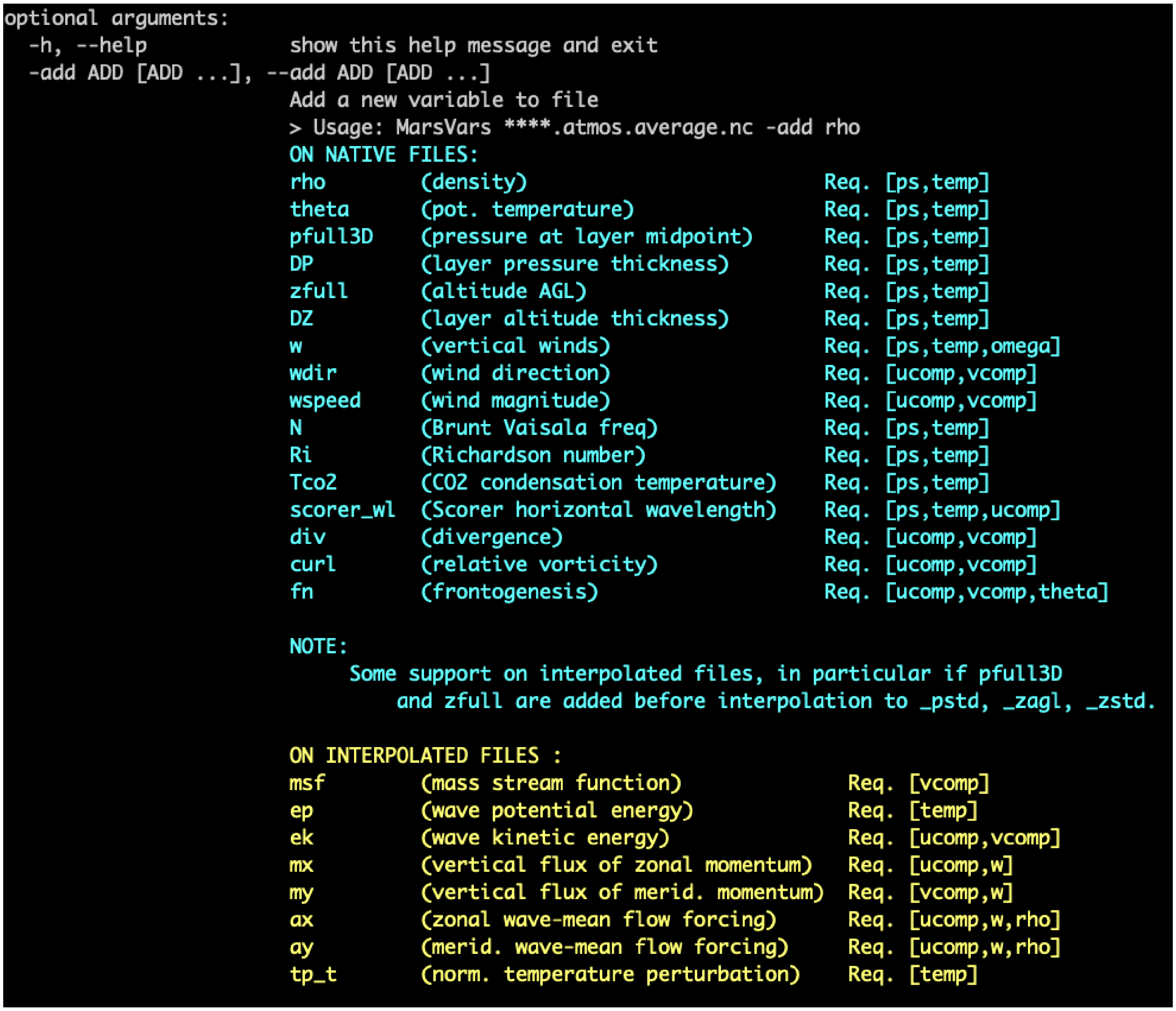
Where `sol_min` and `sol_max` are the minimum and maximum number of days in a filtering period, respectively.

3. `MarsVars.py` - Performing Variable Operations

`MarsVars` provides several tools relating to variable operations such as adding and removing variables and performing column integrations. With no other arguments, passing a file to `MarsVars` displays file content much like `ncdump` :

```
(amesGCM3)>$ MarsVars.py 00000.atmos_average.nc
>
> ===================DIMENSIONS==========================
> ['bnds', 'time', 'lat', 'lon', 'pfull', 'scalar_axis', 'phalf']
> (etc)
> ====================CONTENT==========================
> pfull          : ('pfull',)= (30,), ref full pressure level  [Pa]
> ps             : ('time', 'lat', 'lon')= (4, 180, 360), surface pressure  [Pa]
> temp           : ('time', 'pfull', 'lat', 'lon')= (4, 30, 180, 360), temperature  [K]
> omega          : ('time', 'pfull', 'lat', 'lon')= (4, 30, 180, 360), omega  [Pa/s]
> (etc)
```

This file contains several variables including `ps` , `temp` , and `omega` . Since this is a native file (i.e. the vertical grid is `pfull` indicating the file has not been interpolated), we can calculate the vertical wind (`w`) using `ps` , `temp` , and `omega` and add it to the file:


```
optional arguments:
  -h, --help            show this help message and exit
  -add ADD [ADD ...], --add ADD [ADD ...]
                        Add a new variable to file
                        > Usage: MarsVars ****.atmos.average.nc -add rho
                        ON NATIVE FILES:
                        rho         (density)                       Req. [ps,temp]
                        theta       (pot. temperature)              Req. [ps,temp]
                        pfull3D     (pressure at layer midpoint)    Req. [ps,temp]
                        DP          (layer pressure thickness)      Req. [ps,temp]
                        zfull       (altitude AGL)                  Req. [ps,temp]
                        DZ          (layer altitude thickness)      Req. [ps,temp]
                        w           (vertical winds)                Req. [ps,temp,omega]
                        wdir        (wind direction)                Req. [ucomp,vcomp]
                        wspeed      (wind magnitude)                Req. [ucomp,vcomp]
                        N           (Brunt Vaisala freq)            Req. [ps,temp]
                        Ri          (Richardson number)             Req. [ps,temp]
                        Tco2        (CO2 condensation temperature)  Req. [ps,temp]
                        scorer_wl   (Scorer horizontal wavelength)  Req. [ps,temp,ucomp]
                        div         (divergence)                    Req. [ucomp,vcomp]
                        curl        (relative vorticity)            Req. [ucomp,vcomp]
                        fn          (frontogenesis)                 Req. [ucomp,vcomp,theta]

                        NOTE:
                             Some support on interpolated files, in particular if pfull3D
                                 and zfull are added before interpolation to _pstd, _zagl, _zstd.

                        ON INTERPOLATED FILES :
                        msf         (mass stream function)            Req. [vcomp]
                        ep          (wave potential energy)           Req. [temp]
                        ek          (wave kinetic energy)             Req. [ucomp,vcomp]
                        mx          (vertical flux of zonal momentum) Req. [ucomp,w]
                        my          (vertical flux of merid. momentum) Req. [vcomp,w]
                        ax          (zonal wave-mean flow forcing)    Req. [ucomp,w,rho]
                        ay          (merid. wave-mean flow forcing)   Req. [ucomp,w,rho]
                        tp_t        (norm. temperature perturbation)  Req. [temp]
```

```
(amesGCM3)>$ MarsVars.py 00000.atmos_average.nc -add w
```

We can see that `w` was added by calling `MarsVars` with no argument as before:

```
(amesGCM3)>$ MarsVars.py 00000.atmos_average.nc
>
> ===================DIMENSIONS==========================
> ['bnds', 'time', 'lat', 'lon', 'pfull', 'scalar_axis', 'phalf']
> (etc)
> ====================CONTENT==========================
> pfull          : ('pfull',)= (30,), ref full pressure level  [Pa]
> ps             : ('time', 'lat', 'lon')= (4, 180, 360), surface pressure  [Pa]
> temp           : ('time', 'pfull', 'lat', 'lon')= (4, 30, 180, 360), temperature  [K]
> omega          : ('time', 'pfull', 'lat', 'lon')= (4, 30, 180, 360), omega  [Pa/s]
> w              : ('time', 'pfull', 'lat', 'lon')= (4, 30, 180, 360), vertical wind (a
> (etc)
```

`MarsVars` can also remove variables from files which is particularly useful for reducing file sizes:

```
(amesGCM3)>$ MarsVars.py 00000.atmos_average.nc
>
> ===================DIMENSIONS==========================
> ['bnds', 'time', 'lat', 'lon', 'pfull', 'scalar_axis', 'phalf']
> (etc)
> ====================CONTENT==========================
> pfull          : ('pfull',)= (30,), ref full pressure level  [Pa]
> temp           : ('time', 'pfull', 'lat', 'lon')= (4, 30, 180, 360), temperature  [K]
> omega          : ('time', 'pfull', 'lat', 'lon')= (4, 30, 180, 360), omega  [Pa/s]
> w              : ('time', 'pfull', 'lat', 'lon')= (4, 30, 180, 360), vertical wind (a
> (etc)
```

`MarsVars` is useful when performing column integrations because the function preserves the original variable and creates a new variable ending in `_col` that contains the column integrated values:

```
(amesGCM3)>$ MarsVars.py 00000.atmos_average.nc -col temp
> Performing colum integration: temp...
> temp: Done
```

You can see the added variable in the file:

```
(amesGCM3)>$ MarsVars.py 00000.atmos_average.nc
>
> ===================DIMENSIONS==========================
> ['bnds', 'time', 'lat', 'lon', 'pfull', 'scalar_axis', 'phalf']
> (etc)
> ====================CONTENT==========================
> pfull          : ('pfull',)= (30,), ref full pressure level  [Pa]
> temp           : ('time', 'pfull', 'lat', 'lon')= (4, 30, 180, 360), temperature  [K]
> omega          : ('time', 'pfull', 'lat', 'lon')= (4, 30, 180, 360), omega  [Pa/s]
> w              : ('time', 'pfull', 'lat', 'lon')= (4, 30, 180, 360), vertical wind (ac
> temp_col       : ('time', 'lat', 'lon')= (4, 180, 360), column integration of temp  [I
```

4. `MarsInterp.py` - Interpolating the Vertical Grid

Native MGCM output files use pressure as the vertical coordinate (`pfull`), which means the geometric height and pressure level of an atmospheric layer varies based on location. Climate data is usually analyzed on a standardized grid, however, and it is often necessary to interpolate the files to standard pressure coordinates. The `-type` (`-t`) argument in `MarsInterp` can interpolate files for you:

```
(amesGCM3)>$ MarsInterp.py  00000.atmos_average.nc -t pstd
```

An inspection of the file shows that the pressure level axis which was `pfull` (30 layers) has been replaced by a standard pressure coordinate `pstd` (36 layers), and all 3- and 4-dimensional variables reflect the new shape:

```
(amesGCM3)>$ MarsInterp.py  00000.atmos_average.nc -t pstd
(amesGCM3)>$ MarsVars.py 00000.atmos_average_pstd.nc
>
> ===================DIMENSIONS==========================
> ['bnds', 'time', 'lat', 'lon', 'scalar_axis', 'phalf', 'pstd']
> ====================CONTENT==========================
> pstd           : ('pstd',)= (36,), pressure  [Pa]
> temp           : ('time', 'pstd', 'lat', 'lon')= (4, 36, 180, 360), temperature  [K]
> omega          : ('time', 'pstd', 'lat', 'lon')= (4, 36, 180, 360), omega  [Pa/s]
> w              : ('time', 'pstd', 'lat', 'lon')= (4, 36, 180, 360), vertical wind (ad
> temp_col       : ('time', 'lat', 'lon')= (4, 180, 360), column integration of temp  [I
```

You can also interpolate files to a standard height `zstd` or height above ground level `zagl`. The specific pressure and altitude definitions that `pstd`, `zstd`, and `zagl` correspond to can be found in `/amesGCM3/bin/MarsInterp.py`.

Other grids you can interpolate to can be found in `/amesGCM3/mars_templates/amesgcm_profile`. You can use these by calling `MarsInterp` with the `-level` (`-l`) argument followed by the name of the array you want to use in `amesgcm_profile`.

You can even add your own vertical coordinate array to `amesgcm_profile` so that `MarsInterp` can interpolate MGCM files to your preferred vertical coordinate system.

5. `MarsPlot.py` - Plotting the Results

Overview

The last component of CAP is the plotting routine, `MarsPlot`, which accepts a modifiable template (`Custom.in`) containing a list of plots to create. `MarsPlot` is useful for creating plots from MGCM output quickly, and it is designed specifically for use with the `netCDF` output files (`daily`, `diurn`, `average`, `fixed`) generated by `MarsFiles`.

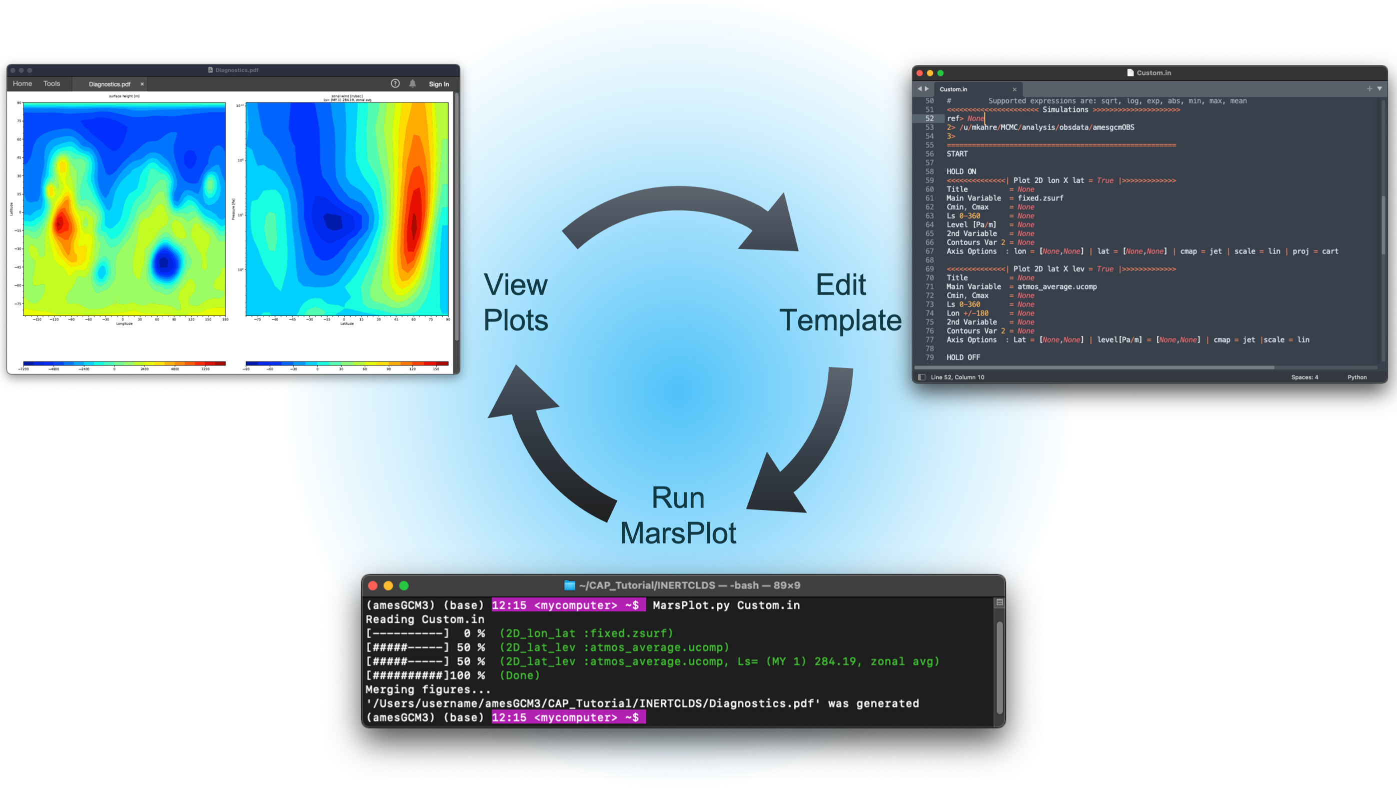
The default template, Custom.in, can be created by passing the `-template` argument to `MarsPlot`. Custom.in is pre-populated to draw two plots on one page: a topographical plot from the fixed file and a cross-section of the zonal wind from the average file. Creating the template and passing it into `MarsPlot` creates a PDF containing the plots:

```
(amesGCM3)>$ MarsPlot.py -template
> /path/to/simulation/run_name/history/Custom.in was created
(amesGCM3)>$
(amesGCM3)>$ MarsPlot.py Custom.in
> Reading Custom.in
> [----------]  0 % (2D_lon_lat :fixed.zsurf)
> [#####-----] 50 % (2D_lat_lev :atmos_average.ucomp, Ls= (MY 2) 252.30, zonal avg)
> [##########]100 % (Done)
> Merging figures...
> /path/to/simulation/run_name/history/Diagnostics.pdf was generated
```

The following figure shows the three components of MarsPlot:

- *MarsPlot.py*, opened in **a terminal** to inspect the netcdf files and ingest the Custom.in template

- *Custom.in* , a template opened in **a text editor**
- *Diagnostics.pdf*, refreshed in a **pdf viewer**



```
#         Supported expressions are: sqrt, log, exp, abs, min, max, mean
<<<<<<<<<<<<<<<<<<<<<< Simulations >>>>>>>>>>>>>>>>>>>>>
ref> None
2> /u/mkahre/MCMC/analysis/obsdata/amesgcmOBS
3>
=======================================================
START

HOLD ON
<<<<<<<<<<<<<<| Plot 2D lon X lat = True |>>>>>>>>>>>>>
Title          = None
Main Variable  = fixed.zsurf
Cmin, Cmax     = None
Ls 0-360       = None
Level [Pa/m]   = None
2nd Variable   = None
Contours Var 2 = None
Axis Options  : lon = [None,None] | lat = [None,None] | cmap = jet | scale = lin | proj = cart

<<<<<<<<<<<<<<| Plot 2D lat X lev = True |>>>>>>>>>>>>>
Title          = None
Main Variable  = atmos_average.ucomp
Cmin, Cmax     = None
Ls 0-360       = None
Lon +/-180     = None
2nd Variable   = None
Contours Var 2 = None
Axis Options  : Lat = [None,None] | level[Pa/m] = [None,None] | cmap = jet |scale = lin

HOLD OFF
```

Line 52, Column 10   Spaces: 4   Python

~/CAP_Tutorial/INERTCLDS — -bash — 89×9

```
(amesGCM3) (base) 12:15 <mycomputer> ~$  MarsPlot.py Custom.in
Reading Custom.in
[----------]  0 %  (2D_lon_lat :fixed.zsurf)
[#####-----] 50 %  (2D_lat_lev :atmos_average.ucomp)
[#####-----] 50 %  (2D_lat_lev :atmos_average.ucomp, Ls= (MY 1) 284.19, zonal avg)
[##########]100 %  (Done)
Merging figures...
'/Users/username/amesGCM3/CAP_Tutorial/INERTCLDS/Diagnostics.pdf' was generated
(amesGCM3) (base) 12:15 <mycomputer> ~$
```

`Custom.in` can be modified using your preferred text editor and renamed to your liking. The above plots are created from the first two blocks of code in `Custom.in` which are set to `= True`. These code blocks are named after the type of plots they create:

```
<<<<<<<<<<<<<<| Plot 2D lon X lat = True |>>>>>>>>>>>>>
Title          = None
Main Variable  = fixed.zsurf
Cmin, Cmax     = None
Ls 0-360       = None
Level [Pa/m]   = None
2nd Variable   = None
Contours Var 2 = None
Axis Options  : lon = [None,None] | lat = [None,None] | cmap = jet | scale = lin | proj
```

How to

Disable or add a new plot

Code blocks is set to `= True` instruct `MarsPlot` to draw those plots. Other templates in `Custom.in` are set to `= False` by default, which instructs `MarsPlot` to skip those plots. In total, `MarsPlot` is equipped to create seven plot types:

```
<<<<<| Plot 2D lon X lat  = True |>>>>>
<<<<<| Plot 2D lon X time = True |>>>>>
<<<<<| Plot 2D lon X lev  = True |>>>>>
<<<<<| Plot 2D lat X lev  = True |>>>>>
<<<<<| Plot 2D time X lat = True |>>>>>
<<<<<| Plot 2D time X lev = True |>>>>>
<<<<<| Plot 1D            = True |>>>>> # Any 1D Plot Type (Dimension x Variable)
```

The settings for each plot may vary but every plot requires at least the following inputs:

```
Title          = Temperature            # Plot title
Main Variable  = atmos_average.temp     # filename.variable *REQUIRED
Cmin, Cmax     = 240,290                # Colorbar limits (minimum, maximum)
2nd Variable   = atmos_average.ucomp    # Overplot U winds
Contours Var 2 = 0,100,200              # List of contours for 2nd Variable
```

Some plots require these inputs:

```
# dimensions that might be required:
Ls 0-360       = 180      # The time at which to plot the variable
Level [Pa/m]   = 50       # The level at which to plot the variable
Lon +/-180     = -90      # The Longitude at which to plot the variable
Latitude       = 50       # The Latitude at which to plot the variable
```

Customize Plots

`Axis Options` specify the axes limits, colormap, linestyle and color for 1D-plots, projection for certain plots :

```
# Axis Options for 2D plots may include:
Lat         = [0,90]        # Latitude range for axes limits
Level[Pa/m] = [600,10]      # Level range for axes limits
sols        = [None,None]   # Sol range for axes limits
Lon +/-180  = [-180,180]    # Longitude range for axes limits
cmap        = jet           # Python colormap to use
scale       = lin           # Color map style ([lin]ear, [log]arithmic)
proj        = cart          # Projection ([cart]esian, [robin]son, [moll]weide, [Npole]
# Axis Options for 1D plots may include:

 lat,lon+/-180,[Pa/m],sols = [None,None] # range for X or Y axes limit
 var = [None,None]                       # range for displayed variables
 linestyle = -                           # Line style following matplotlib convention'-
 axlabel = None                          # Change the default name for the axis
```

Make a 1D-plot

The 1D plot template is different from the others in a few key ways:

- Instead of `Title` , the template requires a `Legend` . When overploting several 1D variables on top of one another, the legend option will label them insetad of changing the plot title.
- There is an additional `linestyle` axis option for the 1D plot.
- There is also a `Diurnal` option. The `Diurnal` input can only be `None` or `AXIS` , since there is syntax for selecting a specific time of day. The `AXIS` label tells `MarsPlot` which dimension serves as the X axis. `Main Variable` will dictate the Y axis.

```
<<<<<<<<<<<<<<| Plot 1D = True |>>>>>>>>>>>>>
Legend          = None                  # Legend instead of Title
Main Variable   = atmos_average.temp
Ls 0-360        = AXIS                  #       Any of these can be selected
Latitude        = None                  #       as the X axis dimension, and
Lon +/-180      = None                  #       the free dimensions can accept
Level [Pa/m]    = None                  #       values as before. However,
Diurnal  [hr]   = None                  #   ** Diurnal can ONLY be AXIS or None **
```

There are several other plot customizations you can use:

- When two or more blocks are sandwiched between a `HOLD ON` and `HOLD OFF` , `MarsPlot` will draw the plots on the same page.

- Plots are created on a standard page (8.5 x 11 inches) in landscape mode, but can be drawn in portrait mode as well.
- Plots can be saved as images instead of PDFs by specifying your preferred filetype (PNG, EPS, etc.) when passing the `--output` (`-o`) argument to `MarsPlot`.
- When creating 1D plots of data spanning multiple years, you can overplot consecutive years by calling `--stack_year` (`-sy`) when submitting the template to `MarsPlot`.
- Specify which MGCM output file to use when plotting by passing the `--date` (`-d`) argument to `MarsPlot` followed by the 5-digit file prefix corresponding to the file you want to use. Alternatively, add the prefix to the filename in the template (e.g. `Main Variable = 00000.fixed.thin`).

Access simulation in a different directory

The final plot-related functionality in `MarsPlot` is the simulation list, which allows you to point `MarsPlot` to different directories containing the MGCM output:

```
<<<<<<<<<<<<<<<<<<<<<< Simulations >>>>>>>>>>>>>>>>>>>>>
ref> None
2> /path/to/another/sim                            # another simulation
3>
=======================================================
```

To access a variable from a file in another directory, just point to the correct simulation when calling `Main Variable`:

```
Main Variable  = XXXXX.filename@N.variable`
```

Where `N` is the number in `<<< Simulations >>>` corresponding the the correct path.

Element-wise operations

The `Main Variable` input also accepts variable operations and time-of-day selections like so:

```
Main Variable  = [filename.variable]*1000  # multiply all values by 1000
Main Variable  = filename.variable{tod = 20}  # select the 20th hour of the day
```

At minimum, `Main Variable` requires `filename.variable` for input, but the above syntax can be combined in several ways allowing for greater plot customization. For example, to plot dust mixing ratio from the diurnal file in simulation #3 at 3 PM local time, the input is:

```
Main Variable  = [atmos_diurn_plevs_T@2.dst_mass_micro{tod = 15}]*1.e6 # dust ppm
#                [filename@N.variable{dimension = X}]*Y
```

Debugging

`MarsPlot` is designed to make plotting MGCM output easier and faster so it handles missing data for you. For example, when dimensions are omitted with `None`, `MarsPlot` makes educated guesses for data selection and will tell you exactly how the data is being processed both in the title for the figures (if `Title = None`), and in the terminal output. Specifics about this behavior are detailed in the instructions at the top of `Custom.in`.

> `MarsPlot` handles many errors by itself. It reports errors both in the terminal and in the generated figures. To by-pass this behavior (when debugging), use the `--debug` option with `MarsPlot` which will raise standard Python errors.

GCM Overview: Lecture

Introduction

Welcome to the overview portion of the Mars Climate Modeling Center (MCMC) Legacy Mars Global Climate Model (GCM) tutorial. By the end of this section of the tutorial, you will have a basic understanding of the main components and structure of the GCM.

The GCM presented here is extensively documented in:

- Haberle et al. 2019. Documentation of the NASA/Ames Legacy Mars Global Climate Model: Simulations of the present seasonal water cycle.

Outline: GCM Overview

---

1. What is a GCM?

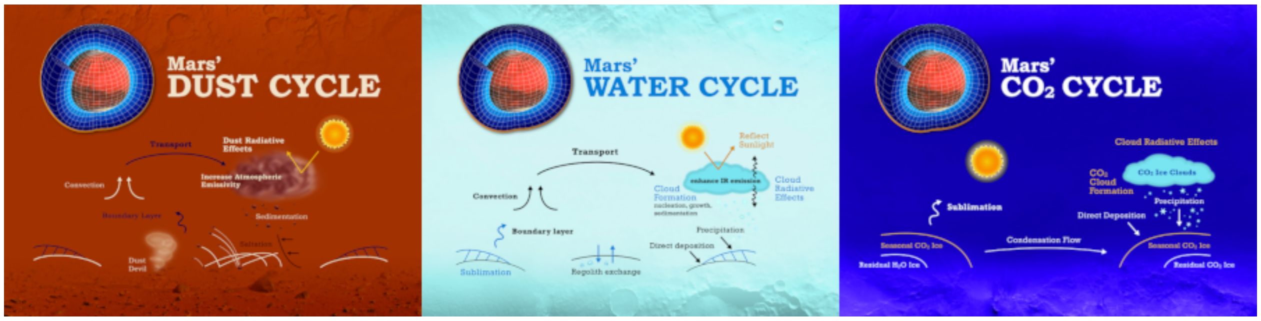
A GCM is a discretized numerical model of a planet's atmosphere that advances through time by solving a set of equations to conserve momentum, mass, and energy. GCMs can generally be divided into two parts based on Newton's second law:

$$F = ma$$

1. The model geophysical fluid dynamics, which represent accelerations (ma).
   - Adiabatic processes, computed in the dynamical core

1. The model physics, which provide the forcing functions for the circulation (F).

   - Diabatic processes, computed in the physics routines
   - For Mars, it is critical to realistically represent the radiative effects of atmospheric dust and clouds



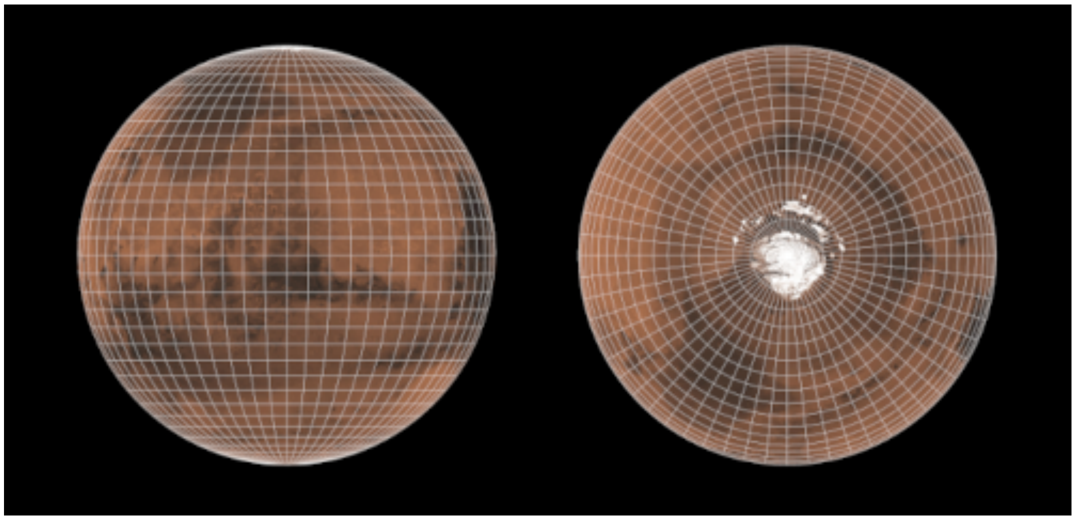
---

2. Dynamical Core (DYCORE)

Called every dynamical timestep (~2 min).

Overview:

- The Legacy GCM utilizes the NASA GSFC ARIES/GEOS dynamical core
- Tracer transport is based on the Van Leer I scheme

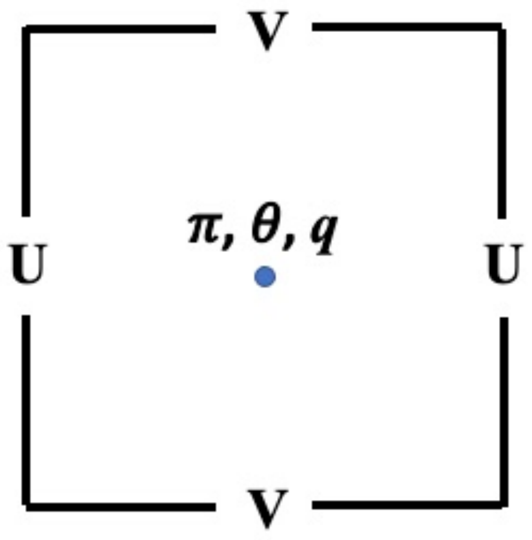


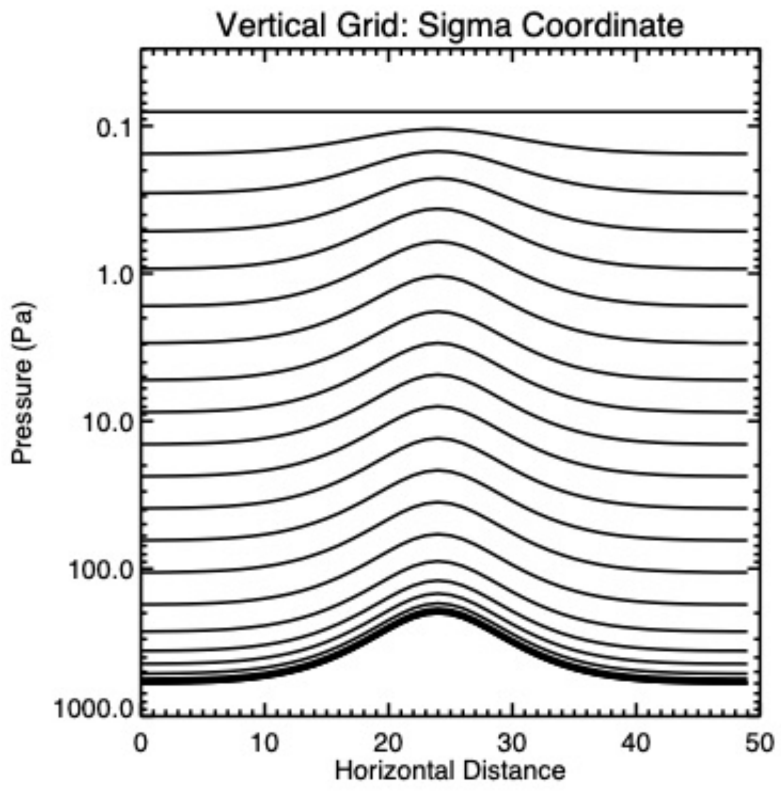
Purpose:

- Computes pressure, wind, potential temperature, and tracer tendencies.

Methodology:

- Solves the primitive equations of meteorology in spherical coordinates using finite differences:
    - Momentum equations in U and V
    - Continuity equation
    - Hydrostatic equation
    - Thermodynamic energy equation
- Tracer transport is based on the flux form of the continuity and advection equations
    - Estimate distribution of tracer mixing ratio in each grid-box with a slope
    - Allows for transport across more than one grid in one timestep in the zonal direction only
- Grid Structure
    - Horizontal: Arakawa C-Grid (Staggered U and V winds)
    - Vertical: Sigma (terrain-following) coordinate





Notes:

- Designed to conserve energy and enstrophy
- Second-order accuracy for all terms, except fourth-order accuracy for vorticity advection
- Dry dynamics only

References:

- Suarez and Takacs, 1995
- Hourdin and Armengaud, 1998

---

3. Physics: Summary of Processes

No	Process	Primary Subroutine(s)
1	Surface CO_2, Surface and Sub-Surface Temperatures	TEMPGR
2	Radiative Heating and Cooling	OPTCV, OPTCI, SFLUXV, SFLUXI
3	Atmospheric CO_2 Condensation	COLDAIR
4	Planetary Boundary Layer	NEWPBL
5	Atmospheric Dust Distribution	FILLTAUCUM, MICROPHYS
6	Dust and Cloud Microphysics	MICROPHYS
7	Convective Adjustment to Ensure Stability	CONVECT
8	Rayleigh Friction	computed in COMP3

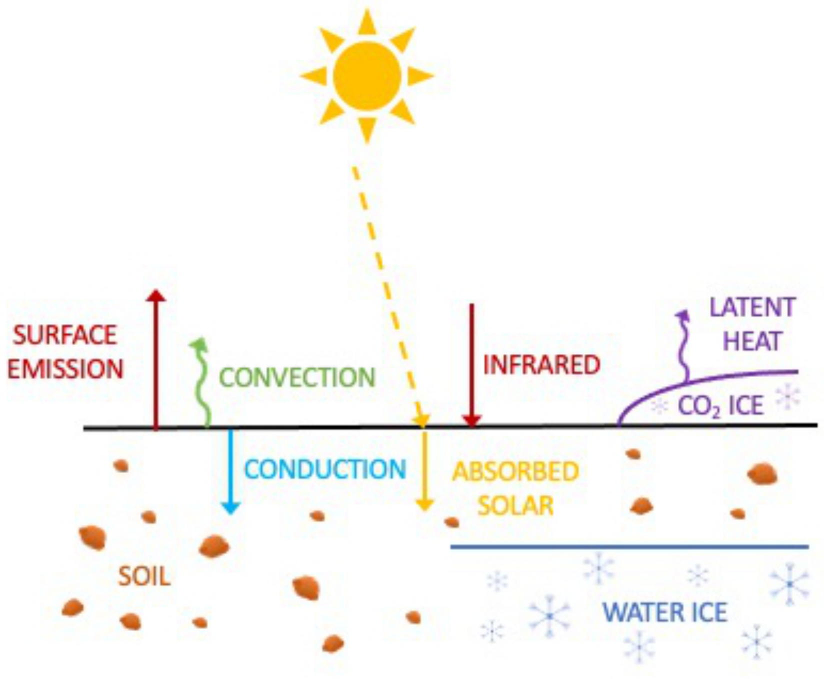
- Subroutines are called from COMP3

3.1 Physics: TEMPGR

Called every dynamical timestep (~2 min).

Purpose:

- Solve surface energy balance equation to calculate surface temperature
- Compute rate of CO_2 condensation at the surface
- Compute subsurface temperatures



Surface Temperature (Energy Balance):

$$\epsilon_G \sigma T_G^4 - F_{IR}^{\downarrow} - (1 - A)\, F_s + F_{conv} + F_{cond} = 0.$$

- Where:

Parameter	Meaning	Units
T_G	Ground Temperature	K
ϵ_G	Surface Emissivity	None
$F_{IR}^{\downarrow}$	Downward IR Flux at the Surface	W m^{-2}
$(1 - A)\, F_s$	Absorbed Solar Flux at the Surface	W m^{-2}
F_{conv}	Upward Heat Exchange with the Atmosphere	W m^{-2}

- Solved using the Newton-Raphson method

Surface CO_2 Condensation:

- Compute CO_2 condensation temperature, T_{CO_2}:

$$T_{\mathrm{CO_2}} = \frac{3192.48}{23.349 - \ln(p_s)}$$

- Hold T_G at $T_{\mathrm{CO_2}}$, and use surface energy balance and latent heat of condensation of CO$_2$, L, to compute rate of CO$_2$ condensation/sublimation:

$$\frac{\partial M_{\mathrm{CO_2}}}{\partial t} = \frac{\epsilon_G \sigma T_{\mathrm{CO_2}}^4 - F_{IR}^{\downarrow} - (1-A)\,F_s + F_{conv} + F_{cond}}{L}$$

- Where:

Parameter	Meaning	Units
L	CO$_2$ Latent Heat of Condensation	J kg^{-1}

Subsurface Temperatures (Diffusion Equation):

$$\frac{\partial T_s}{\partial t} = \frac{\partial}{\partial z}\left(\frac{J}{\rho_s c_s}\right) = \frac{\partial^2}{\partial z^2}\left(\frac{T\lambda_s}{\rho_s c_s}\right)$$

- Where:

Parameter	Meaning	Units
T_s	Soil Temperature	K
J	Conductive Heat Flux	W m^{-2}
ρ_s	Soil/Ice Density	kg m^{-3}
c_s	Soil/Ice Specific Heat	J kg^{-1} K^{-1}
λ_s	Soil Conductivity	W m^{-1} K^{-1}

- Simple two-component soil model (soil over ice)
- Solved explicitly

References

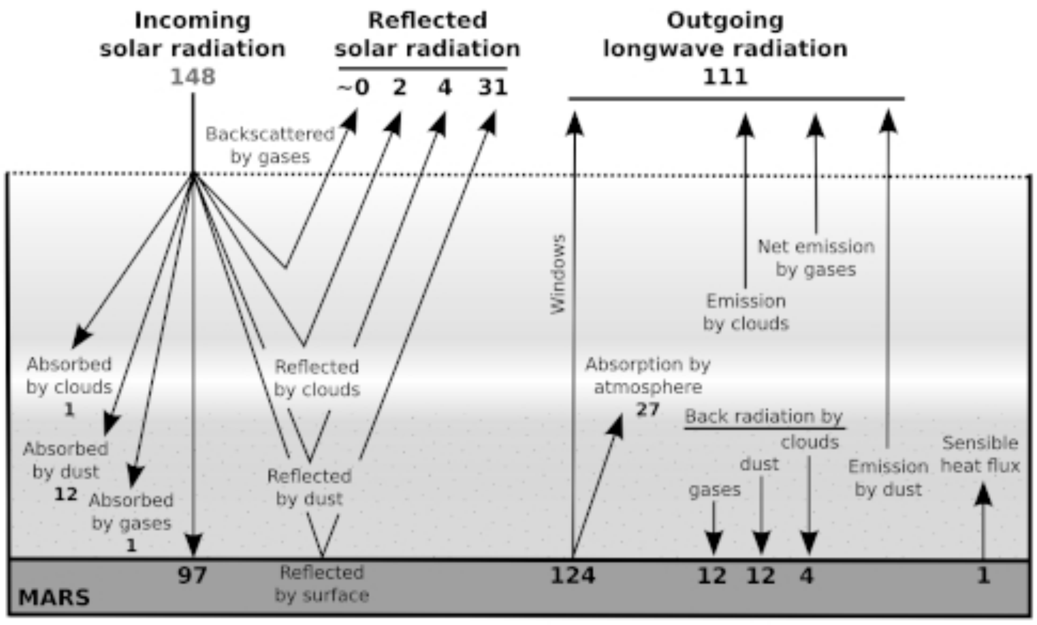
- Haberle and Jakosky, 1991
- Haberle et al. 1999

3.2 Physics: Radiation Code

Called every physical timestep (~16 min).

Purpose:

- Compute solar and infrared heating rates (K per second)



From Wolff et al. 2017

Method:

- Compute heating rates from flux divergences
- Compute fluxes from 2-stream code (needs opacities and scattering properties)

Radiatively active species: CO_2, H_2O, aerosols (dust and ice)

Opacities:

- Correlated k's for gases (CO_2/H_2O)
- Extinction efficiencies for dust and ice **Scattering properties:**

- Rayleigh scattering for CO_2

- Aerosol scattering properties are functions of size and amount of ice.
  - We use a core/mantle Mie code to generate a lookup table.
  - Refractive indices from Wolff (2009)2169-9100.CRISM1) for dust and Warren (1984)

for ice.

Spectral resolution: 7 bands in visible (0.4-4.5 μm), 5 bands in IR (4.5-1000 μm)

No	GCM Band	Wavelength Interval (μm)	Wavenumber Interval (cm^{-1})
1	Vis-7	0.24-0.40	41666.7 - 25000.0
2	Vis-6	0.40-0.80	25000.0 - 12500.0
3	Vis-5	0.80-1.31	12500.0 - 7633.59
4	Vis-4	1.31-1.86	7633.59 - 5376.34
5	Vis-3	1.86-2.48	5376.34 - 4032.26
6	Vis-2	2.48-3.24	4032.26 - 3086.42
7	Vis-1	3.24-4.50	3086.42 - 2222.22
8	IR-5	4.50-8.00	2222.22 - 1250.00
9	IR-4	8.00-12.0	1250.00 - 833.33
10	IR-3	12.0-24.0	833.33 - 416.67
11	IR-2	24.0-60.0	416.67 - 166.67
12	IR-1	60.0-1000	166.67 - 10.0

Main Routines Involved:

- OPTCV(I): sets optical properties
- SFLUXV(I): sums fluxes over spectral intervals
- GFLUXV(I): gets fluxes by solving a tri-diagonal matrix

Other routines involved:

- FILLPT: readies p,T fields for radiation routines
- OPT_DST & OPT_CLD: integrate over size bins to get the scattering properties

References

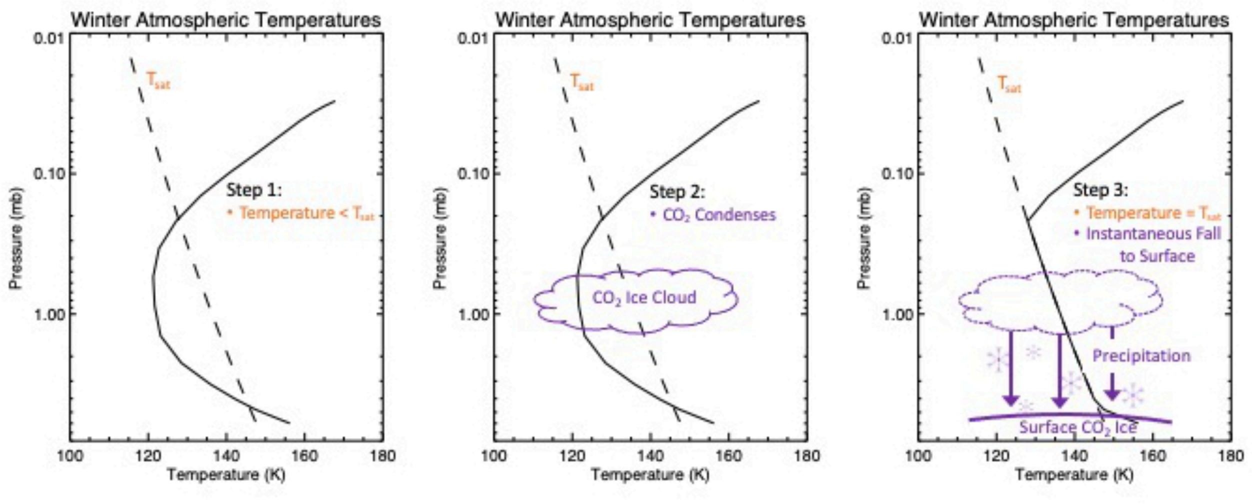
- Toon et al. 1989
- Haberle et al. 2019

3.3 Physics: COLDAIR

Called every physical timestep (~16 min).

Purpose:

- Compute atmospheric CO_2 condensation



Methodology:

- Diagnose layers that have temperatures less than CO_2 condensation temperature:

$$T_{\mathrm{CO}_2} = \frac{3192.48}{23.349 - \ln(p_l)}$$

- In these layers, compute amount of CO_2, $\delta \mathrm{M}_{\mathrm{CO}_2}$, that needs to condense to maintain T_{CO_2}:

$$\delta M_{\mathrm{CO}_2,l} = \frac{100 * C_p \left(T_{\mathrm{CO}_2} - T_l\right) \delta\sigma_l \pi}{gL}$$

- Where:

Parameter	Meaning	Units
p_l	Pressure of Layer l	mbars
C_p	Specific Heat of Air	J kg^{-1} K^{-1}
T_l	Temperature of Layer l	K
$\delta\sigma_l$	Thickness of Layer l in σ coordinates	None

π	$= p_s - p_t$	mbars
p_s	Surface Pressure	mbars
p_t	Pressure at the Top of the Dynamical Domain (Tropopause Pressure)	mbars
g	Gravity	m s^{-2}
L	CO$_2$ Latent Heat of Condensation	J kg^{-1}

- Sum $\delta M_{\mathrm{CO_2}}$ through the column and add to the surface CO$_2$ budget
- If $T_g > T_{\mathrm{CO_2}}$, calculate the amount of CO$_2$ that will remain on surface as T_g cools to $T_{\mathrm{CO_2}}$

Reference

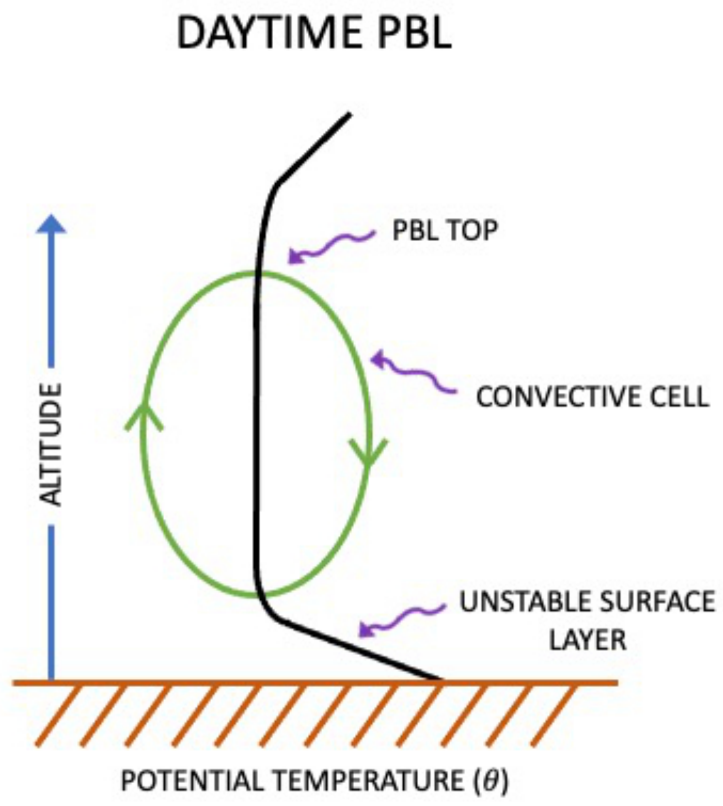
- Pollack et al. 1990

3.4 Physics: NEWPBL

Called every physical timestep (~16 min).

Purpose:

- Compute the upward surface turbulent fluxes of heat, momentum, and mass (tracers)
- Vertically mix these in the atmosphere



Basic Physics:

- Surface fluxes calculated from Monin-Obukhov theory (drag laws)
- For example, the heat flux (F_{conv}) is:

$$F_{conv} = -\rho c_p c_h u_* \left(T_a - T_g\right)$$

- Where:

Parameter	Meaning	Units
F_{conv}	Heat Flux	W m^{-2}
ρ	Near-Surface Air Density	kg m^{-3}
c_p	Air Specific Heat	J kg^{-1} K^{-1}
c_h	Heat Drag Coefficient	None
u_*	Frictional Wind Speed	m s^{-1}
T_a	Near-Surface Air Temperature	K

T_g	Ground Temperature	K

- Mixing coefficients are functions of the local Richardson number and are based the Mellor and Yamada (1984) Level 2 scheme
- The Richardson number, R_i is given by:

$$R_i = \frac{\frac{g}{\theta}\frac{\partial\theta}{\partial z}}{\left(\frac{\partial V}{\partial z}\right)^2}$$

- Where:

Parameter	**Meaning**	**Units**
g	Gravity	m s^{-2}
θ	Potential Temperature	K
V	Wind Speed	m s^{-1}
z	Altitude	m

Methodology:

- Solves a diffusion equation with an arbitrary vertical co-ordinate
- Scheme can be implicit or explicit (we always run with the implicit option)

Notes:

- Does not completely eliminate instabilities (i.e., it is not a convective adjustment scheme)
- Mixing effectively shuts off when $R_i > 0.25$
- Mass is mixed using the heat coefficients

References:

- Haberle et al. 1993
- Haberle et al. 1999

3.5 Physics: Atmospheric Dust Distribution

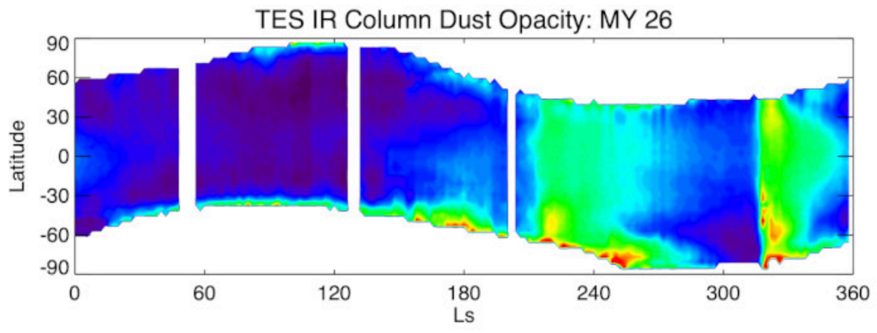
Computed every physical timestep (~16 min).

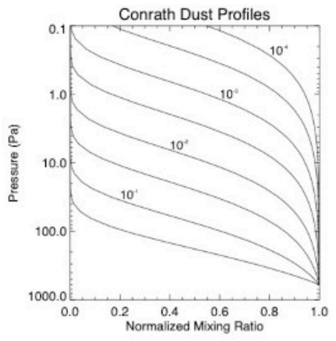
Purpose:

- Provide multiple options for determining the atmospheric dust distribution
- Hierarchy of dust treatments, from simple to complex
- Important for including the radiative forcing from dust

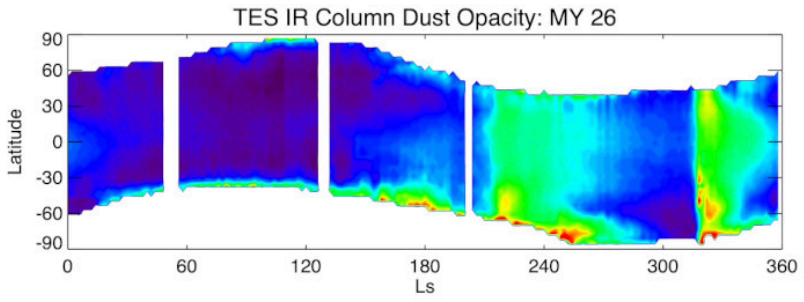
Methods

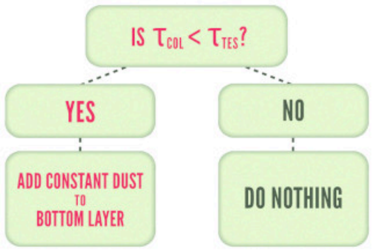
- Fully Prescribed:
  - Horizontal: Globally constant or based on an observation-based dust map
  - Vertical: Prescribed using Conrath profiles or similar (Conrath, 1975)
  - Lifting: NONE



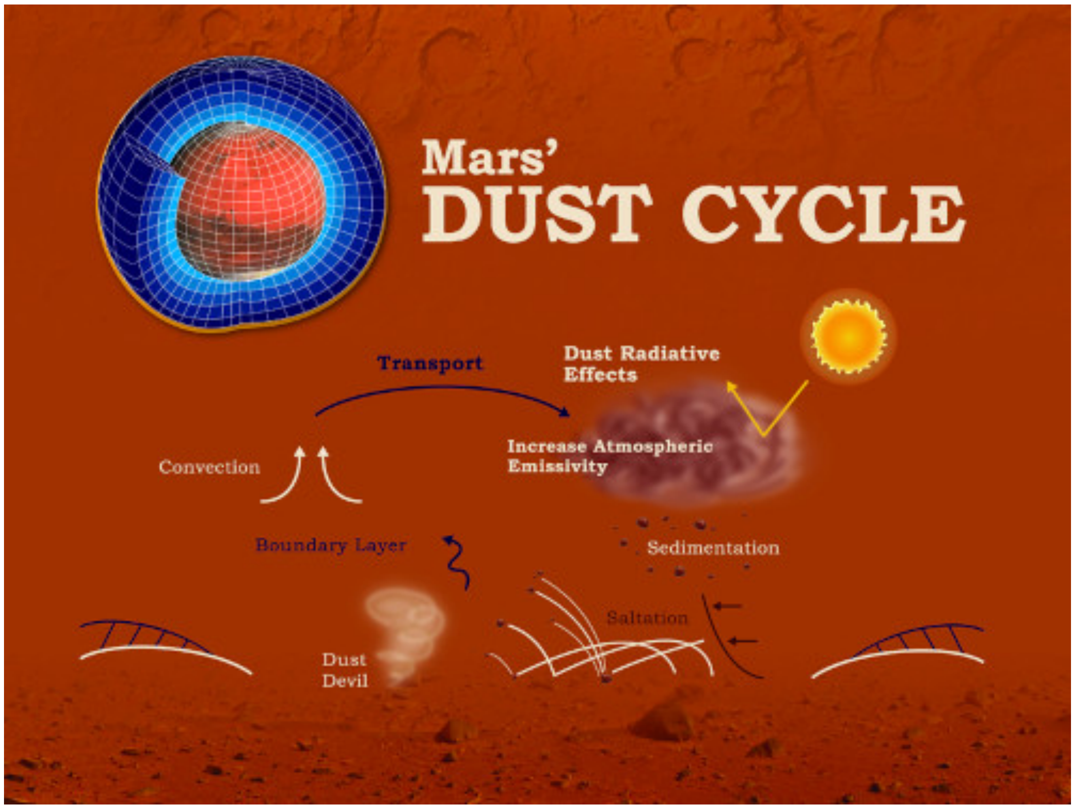


- Semi-Prescribed (Tracking):
  - Horizontal: Globally constant or based on a observation-based dust map
  - Vertical: Self-consistenly determined from transported dust tracers
  - Lifting: As needed to track desired horizontal distribution (usually a map)




IS $\tau_{COL} < \tau_{TES}$?
YES
NO
ADD CONSTANT DUST TO BOTTOM LAYER
DO NOTHING

- Fully Interactive:
  - Horizontal: Self-consistently determined by predicted lifting and transport
  - Vertical: Self-consistenly determined from transported dust tracers
  - Lifting: Based on physical dust lifting parameterizations (usually wind stress and dust devils)



References:

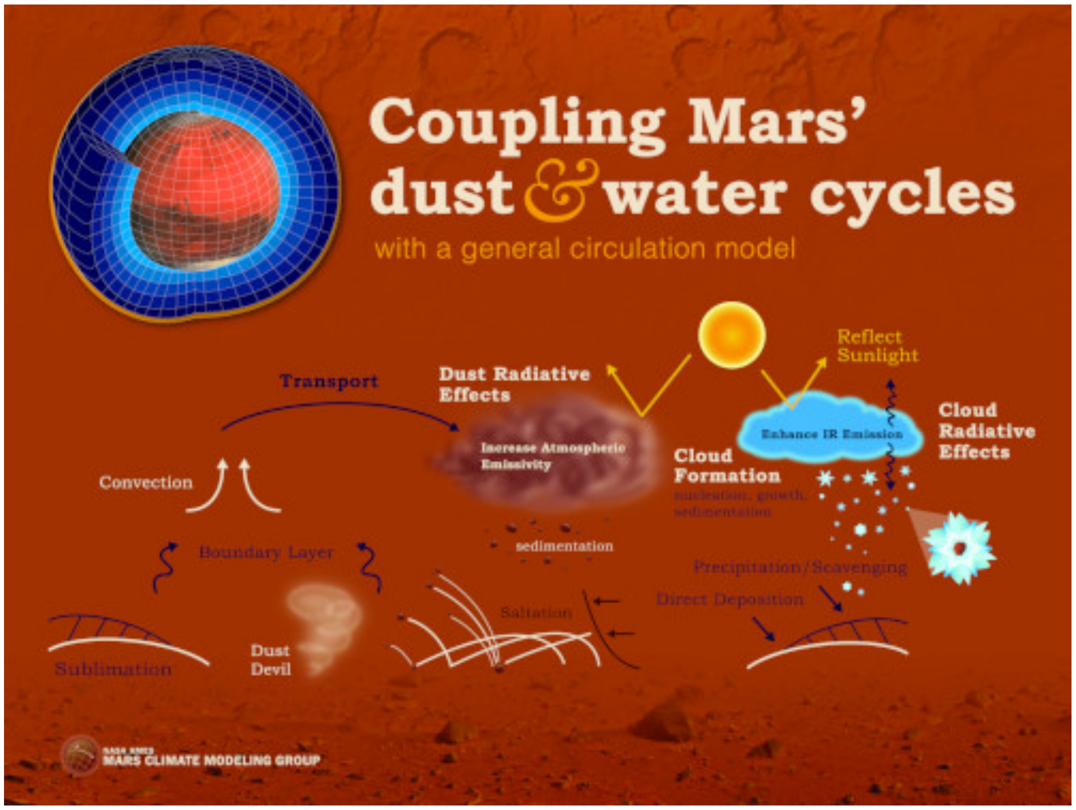
- Haberle et al. 2019
- Kahre et al. 2015

3.6 Physics: MICROPHYS

Called every physical timestep (~16 min).

Purpose:

- Compute dust injection, aerosol sedimentation, and cloud nucleation and growth



Basic Physics:

- Multiple dust injection methods, including dust tracking, dust devil lifting and wind stress lifting
- Cunningham-Stokes gravitational sedimentation
- Cloud nucleation and growth as described in Haberle et al. (2019) and Montmessin et al. (2002/2004)
- Transported dust and clouds are optionally radiatively active

Methodology:

Moment Method

- Assume particle size distribution is log-normal

$$n\left(r\right) = \frac{N_o}{r\sigma_o\sqrt{2\pi}}\exp\left[\frac{1}{2}\left(\frac{\ln(r/r_o)^2}{\sigma_o}\right)\right]$$

- It follows that the total number of particles between $r_{\min}$ and $r_{\max}$ is:

$$N=\int_{r_{min}}^{r_{max}} n(r)\,dr=\frac{N_o}{2}\left[\operatorname{erf}\left(\frac{\ln(r_{max}/r_o)}{\sqrt{2\pi\sigma_o}}\right)-\operatorname{erf}\left(\frac{\ln(r_{min}/r_o)}{\sqrt{2\pi\sigma_o}}\right)\right]$$

- Separately, M_o can be calculated from r_o:

$$M_o=\frac{4}{3}\pi\rho\int r^3 n(r)=\frac{4}{3}\pi\rho N_o r_o^3 \exp\left(\frac{9}{2}\sigma_o^2\right)$$

- Solving for r_o:

$$r_o=\left[\frac{3M_o}{4\pi\rho N_o}\right]^{1/3}\exp\left(-\frac{3}{2}\sigma_o^2\right)$$

- Thus, the distribution can be fully represented by M_o, N_o, and σ_o
- We assume σ_o is a constant, so only M_o and N_o are carried as tracers.
- Taking further advantage of the properties of log-normal distributions, many representative particle radii can be calculated

Process	Symbol	Formula
Nucleation	r_o	$\left(\frac{3M_o}{4\pi\rho N_o}\right)^{1/3}\exp\left(-1.5\sigma_o^2\right)$
Growth	r_v	$r_o\exp\left(1.5\sigma_o^2\right)$
Mass Sedimentation	$r_{sed,m}$	$r_o\exp\left(4.5\sigma_o^2\right)$
Number Sedimentation	$r_{sed,n}$	$r_o\exp\left(1.5\sigma_o^2\right)$
Opacity (total cross-section)	r_s	$r_o\exp\left(\sigma_o^2\right)$
Scattering properties	r_{eff}	$r_o\exp\left(2.5\sigma_o^2\right)$

- We use the most appropriate representative radius for each physical process

Tracers

- The GCM carries an array (`QTRACE`) with 6 atmospheric tracers:

Tracer	Units	Array Index	Index Name
Dust Mass	kg/kg	1	ima_dt
Dust Number	#/kg	2	inb_dt
Water Cloud Mass	kg/kg	3	ima_cld
Water Cloud Number	#/kg	4	inb_cld
Dust Core Mass	kg/kg	5	ima_cor
Water Vapor Mass	kg/kg	6	ima_vap

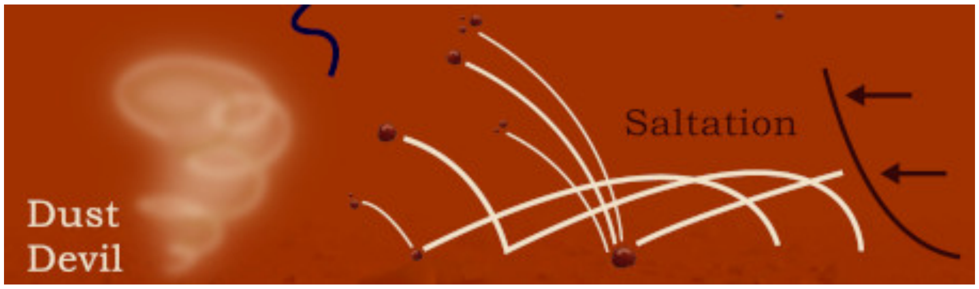
- An array of 6 surface tracers (`QCOND`) is also carried, but some indices are empty:

Tracer	Units	Array Index	Index Name
Dust Mass Deposited	kg/m^2	1	ima_dt
EMPTY	N/A	2	N/A
Water Cloud Mass Deposited	kg/m^2	3	ima_cld
EMPTY	N/A	4	N/A
Dust Core Mass Deposited	kg/m^2	5	ima_cor
Water Mass Reservoir	kg/m^2	6	ima_vap

Processes

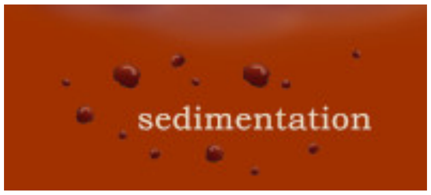
Dust Injection: **DUST_UPDATE**

- Computes flux of dust from the surface to the atmosphere
- Assumes a log-normal distribution with r_{eff} = 2.0 μm
- Adds to mass and number moments for dust



Sedimentation: **SEDIM**

- Uses mass and number weighted mean radii of dust and cloud particles
- Computes fall velocities for each type of particle
- Update column tracer arrays for all tracers (except H_2O vapor)



Cloud Nucleation and Growth: **NUCLEACOND**

- Nucleation:
    - Expands dust mass and number moments into bin
    - Computes nucleation rate for each bin
    - Sums over bins to compute total mass and number of nucleated dust particles
- Growth:
    - Compute volume mean radius for cloud particles
    - Computes growth rate and converts to cloud mass exchange
    - Updates cloud mass (and cloud number if cloud particles sublimate completely)

References:

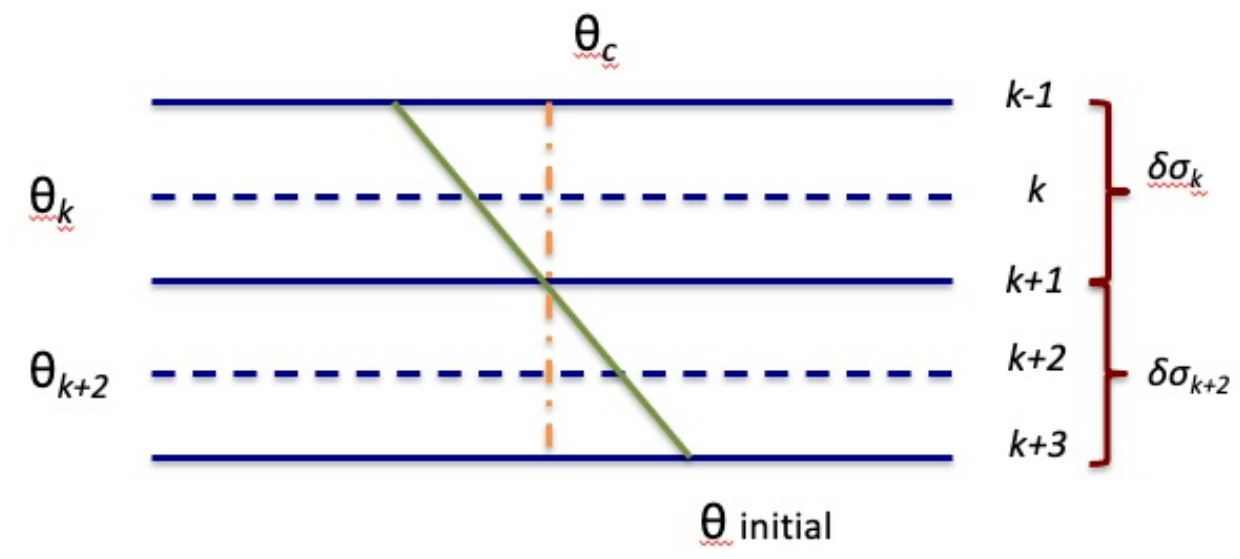
- Haberle et al. 2019
- Montmessin et al. 2004
- Montmessin et al. 2002

3.7 Physical Processes: CONVECT

Called every physical timestep (~16 min).

Purpose:

- Remove any remaining superadiabatic atmospheric layers after NEWPBL.
- Determine the potential temperature within the convective layer (θ_c).
- Determine the pressure of the top of the convective layer that exists above the surface (i.e., pressure at the top of the boundary layer, p_c).



Method:

- Stabilize regions where potential temperature decreases with height by conserving the total heat energy.

$$\theta_c = \frac{(p_k)^\kappa \delta\sigma_k \theta_k + (p_{k+2})^\kappa \delta\sigma_{k+2}\theta_{k+2}}{(p_k)^\kappa \delta\sigma_k + (p_{k+2})^\kappa \delta\sigma_{k+2}}$$

-Where:

Parameter	Meaning	Units
p	Pressure	mbars
$\delta\sigma$	Layer Thickness in σ Coordinates	None
θ	Potential Temperature	K
κ	$\frac{R_{gas}}{c_p}$	None
R_{gas}	Gas Constant for CO_2	J kg^{-1} K^{-1}
c_p	Specific Heat	J kg^{-1} K^{-1}

- Instantaneously mix all tracers in unstable regions.

References:

- Pollack et al. 1990
- Pollack et al. 1981

3.8 Physical Processes: Rayleigh Friction

Applied every physical timestep (~16 min).

Purpose:

- To damp waves as they approach the model top to minimize reflections from the top boundary

Method:

- Damp zonal (U) and meridional (V) winds with a rate of change that is proportional to their magnitude

$$\frac{\partial U}{\partial t} = -\frac{U}{\tau} \quad (1)$$
$$\frac{\partial V}{\partial t} = -\frac{V}{\tau} \quad (2)$$

- where τ is a damping timescale
- Convert lost kinetic energy into heat and update potential temperature of affected layers
- Applied to the top 3 model layers
- Negligible effect on the lower atmosphere

Reference:

- Pollack et al. 1990

---

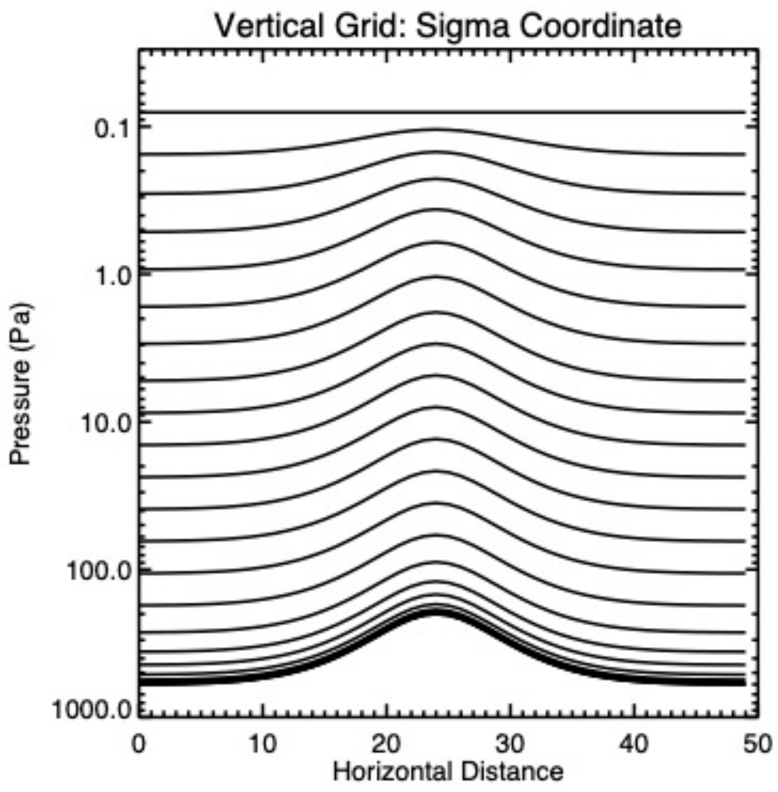
4. Grid Structure

Vertical: Sigma Coordinate (Terrain Following)

- Simplifies the handling of the lower boundary in the presence of topography.
- Defined by:

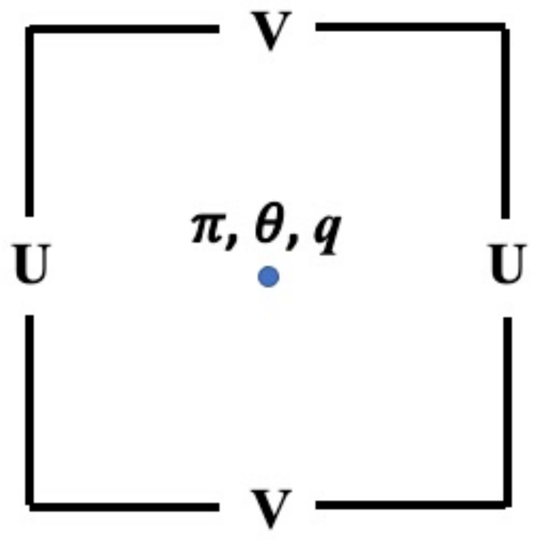
$$\sigma = \frac{(p - p_t)}{(p_s - p_t)}$$

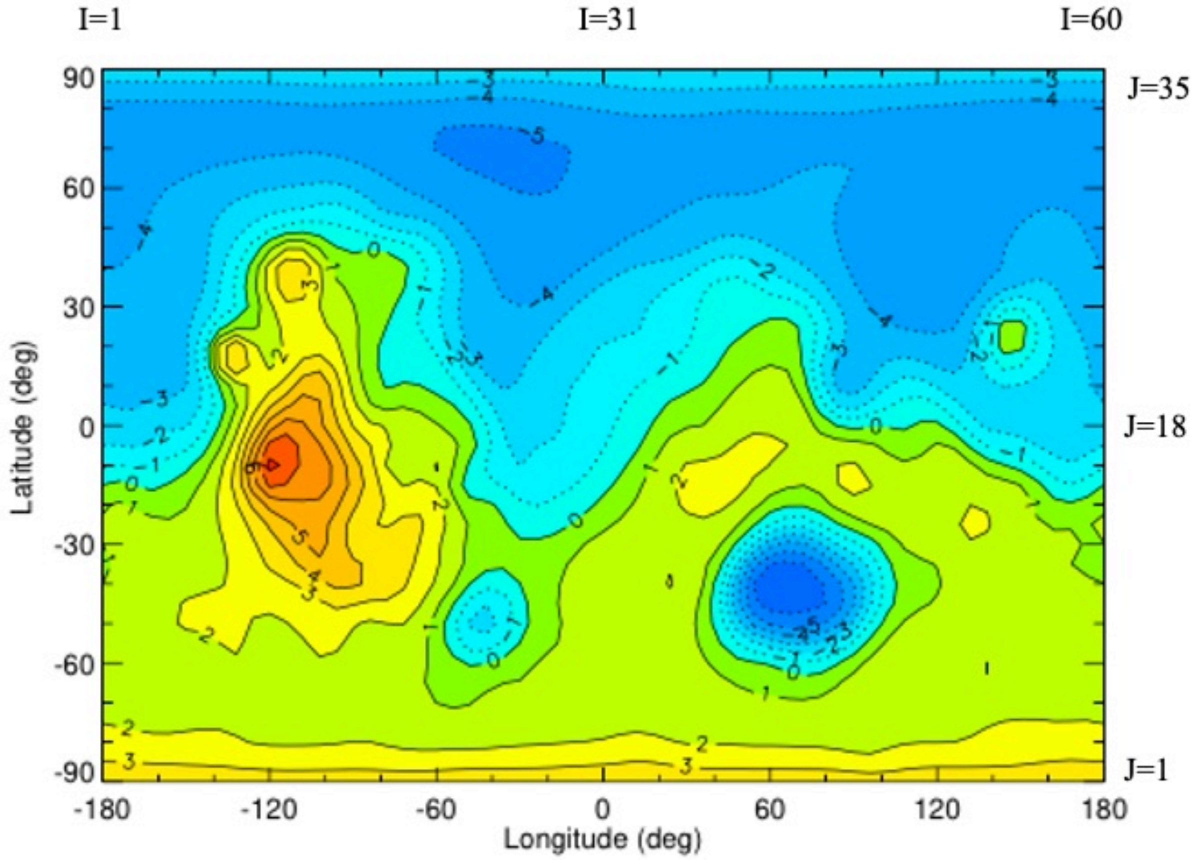
- p is pressure; p_t is pressure at the top of the dynamical domain; p_s is surface pressure
- Nominally 24 layers in the vertical
- Vertical resolution ranging from:
  - ~10 meters near the surface
  - ~10 kilometers aloft



Horizontal: Latitude-Longitude Grid

- Arakawa C-Grid Staggered Grid Structure
  - Pressure ($\pi = p_s - p_t$), potential temperature (θ), tracers (q) carried at grid mid-points
  - Winds are carried at grid boundaries
    - Zonal winds (U) staggered E/W
    - Meridional winds (V) staggered N/S
- Nominal horizontal resolution is 5^{o} latitude by 6^{o} longitude


V
U
π, θ, q
U
V


I=1
I=31
I=60
J=35
J=18
J=1
Latitude (deg)
Longitude (deg)

5. Time Stepping (NEWSTEP)

Called every dynamical timestep (~2 min).

Purpose:

- Update all atmospheric field (π, θ, U, V, QTRACE)
- Update surface tracer field (QCOND)
- Apply Robert time filter and Shapiro spatial filters (but not on QTRACE)

Method:

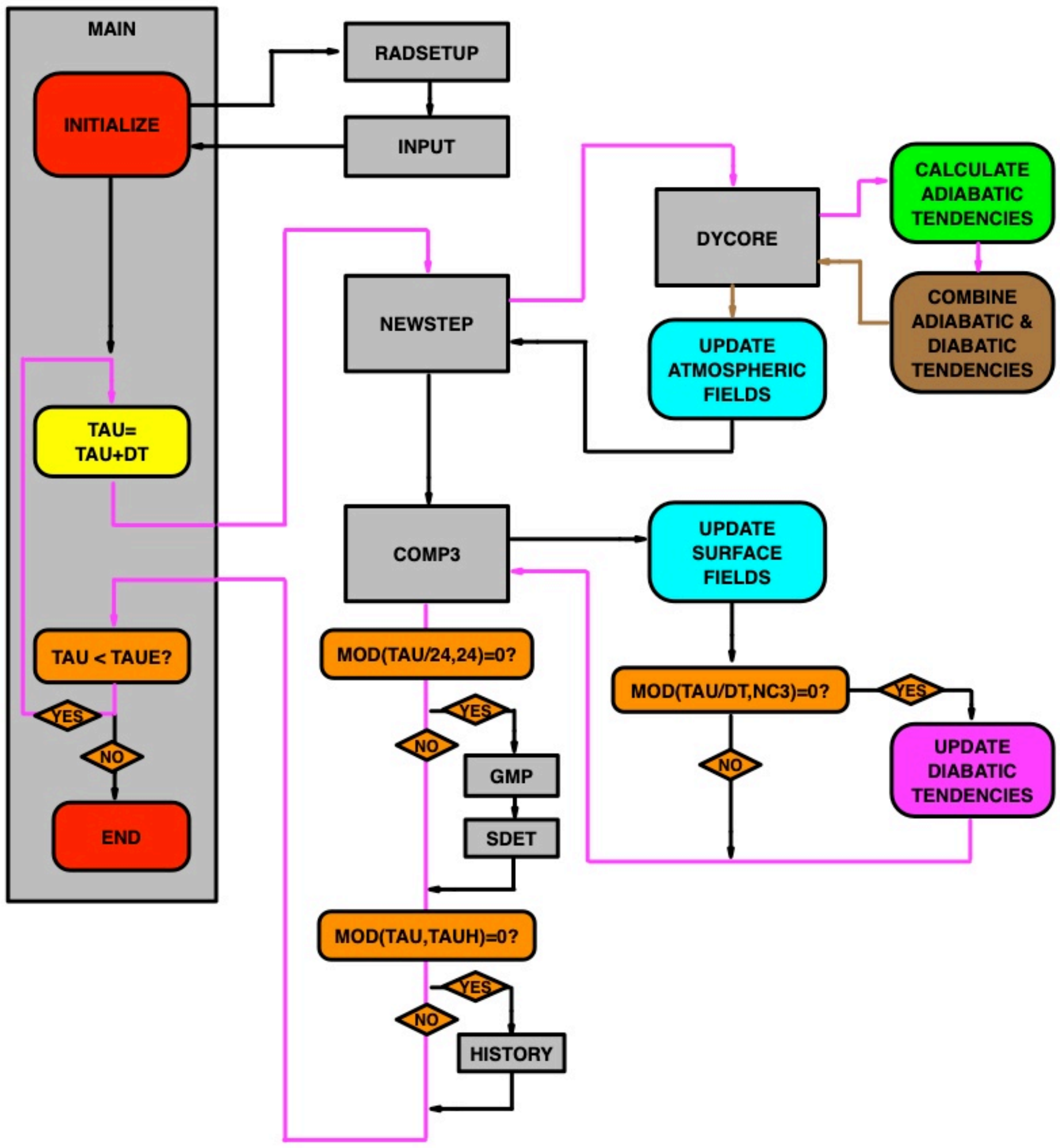
- Use leap frog scheme in combination with a Robert time filter (α = 0.05) to suppress computational mode.
- Note that potential temperature (θ) and tracers (q) are pressure-weighted, while the other fields are not.

Field	Advance Fields to Future Timestep (t+1)	Filter Current Timestep (t)
Pressure	$\pi_{t-1} + \frac{\partial \pi}{\partial t} 2\mathrm{dt}$	$\alpha \frac{\pi_{t-1} + \pi_{t+1}}{2} + (1-\alpha)\,\pi_t$
Potential Temperature	$\frac{1}{\pi_{t+1}} \left[\pi_{t-1}\theta_{t-1} + \frac{\partial \pi\theta}{\partial t} 2\mathrm{dt} \right]$	$\frac{1}{\pi_t} \left[\alpha \frac{(\pi_{t-1}\theta_{t-1} + \pi_{t+1}\theta_{t+1})}{2} + (1-\alpha)\,\pi_t\theta_t \right]$
Zonal Wind	$u_{t-1} + \frac{\partial u}{\partial t} 2\mathrm{dt}$	$\alpha \frac{(u_{t-1} + u_{t+1})}{2} + (1-\alpha)\,u_t$
Meridional Wind	$v_{t-1} + \frac{\partial v}{\partial t} 2\mathrm{dt}$	$\alpha \frac{(v_{t-1} + v_{t+1})}{2} + (1-\alpha)\,v_t$
Tracers	$\frac{1}{\pi_{t+1}} \left[\pi_{t-1}q_{t-1} + \frac{\partial \pi q}{\partial t} 2\mathrm{dt} \right]$	$\frac{1}{\pi_t} \left[\alpha \frac{(\pi_{t-1}q_{t-1} + \pi_{t+1}q_{t+1})}{2} + (1-\alpha)\,\pi_t q_t \right]$
Surface Tracers	$qc_{t-1} + \frac{\partial qc}{\partial t} 2\mathrm{dt}$	$\alpha \frac{(qc_{t-1} + qc_{t+1})}{2} + (1-\alpha)\,qc_t$

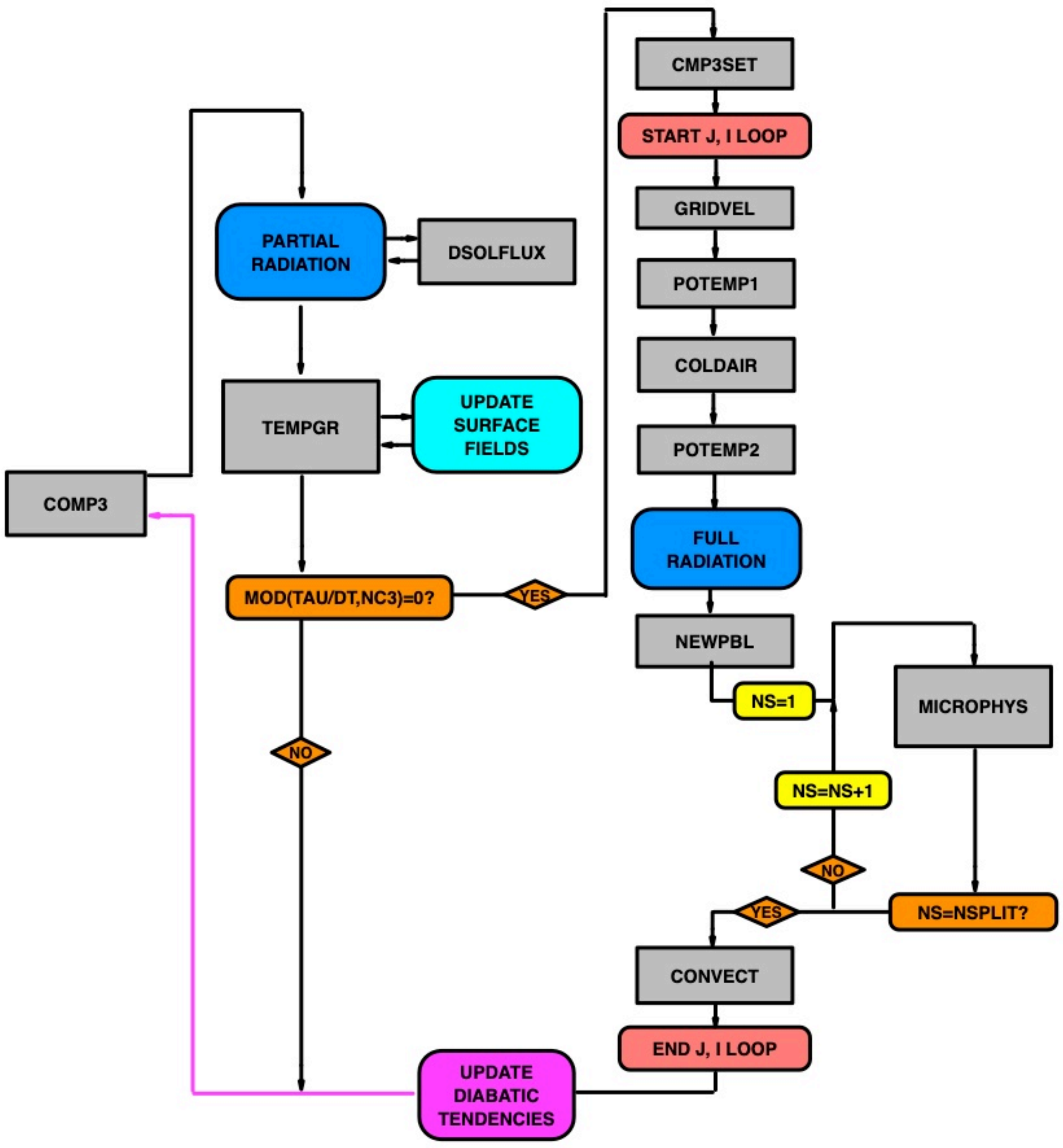
6. Code Architecture

6. Code Architecture

Full Code Architecture



Physics Code Architecture


CMP3SET
START J, I LOOP
GRIDVEL
PARTIAL RADIATION
DSOLFLUX
POTEMP1
COLDAIR
TEMPGR
UPDATE SURFACE FIELDS
POTEMP2
COMP3
FULL RADIATION
MOD(TAU/DT,NC3)=0?
YES
NEWPBL
NS=1
MICROPHYS
NO
NS=NS+1
NO
YES
NS=NSPLIT?
CONVECT
END J, I LOOP
UPDATE DIABATIC TENDENCIES

In [ ]:

GCM Practical: Lecture and Exercises

Introduction

Welcome to the practical portion of the Mars Climate Modeling Center (MCMC) Legacy Mars Global Climate Model (GCM) tutorial. By the end of this section of the tutorial, you will have the practical skills necessary to run the GCM.

The GCM presented here is extensively documented in:

- Haberle et al. 2019. Documentation of the NASA/Ames Legacy Mars Global Climate Model: Simulations of the present seasonal water cycle.

Outline: GCM Practical

1. Install and Compile
2. Required Input Files
3. Cold Starts
4. Exercise: TASK 1
5. Warm Starts
6. History Files
7. Changing Model Resolution
8. Exercises: TASKS 2 and 3

---

1. Installation: Clone the Repository

```
(local)>$ git clone https://github.com/nasa/legacy-mars-global-
climate-model.git
```

This will produce a directory called `legacy-mars-global-climate-model`. Navigate into that directory and list its contents:

```
(local)>$ cd legacy-mars-global-climate-model
(local)>$ ls -l
```

The following directories will be visible:

```
./documentation          # contains GCM documentation
./code                   # contains the GCM source code
./run                    # is where you will run the model
./run/data               # required input files
./analysis               # simple analysis routine
./analysis/validation    # sample plots
./tutorial               # tutorial files
```

Compile the Code

From the main model directory (`legacy-mars-global-climate-model/`), navigate to the source code (`code/`) directory:

```
<(local)>$ cd code
```

For gfortran, open the Makefile and uncomment the gfortran options and comment out the ifort options. The original lines are:

```
F90_COMP  = ifort
F_OPTS    = -c -O2
#F90_COMP = gfortran
#F_OPTS   = -c -O3 -finit-local-zero -frecord-marker=4
```

and the lines modified for gfortran will instead be:

```
#F90_COMP = ifort
#F_OPTS   = -c -O2
F90_COMP  = gfortran
F_OPTS    = -c -O3 -finit-local-zero -frecord-marker=4
```

Once the Makefile is ready, you can proceed with compiling the model. First, remove all object files (`*.o`), and module files (`*.mod`) to ensure a clean build by typing:

```
<(local)>$ make clean
```

Next, compile the code by typing:

```
<(local)>$ make
```

which creates an executable file called `gcm2.3`

NOTE: recompiling the code is required whenever there is a *source code* change.

2. Required Input Files

The GCM requires the following input files:

Description	File Name	Subroutine	Data Type
Topography[1]	topog37x60.mola_intel	input	binary
Surface Albedo[1]	osu_albedo_5x6_2011	input	binary
Thermal Inertia[1]	osu_ti_5x6_2011	input	binary
Dust Map[1]	TES_my24_dustscenario_zvary_37x60_6ls_intel	input	binary
K-coefficients (V)[2]	CO2H2O_V_12_95_INTEL	laginterp	binary
K-coefficients (IR)[2]	CO2H2O_IR_12_95_INTEL	laginterp	binary
Cloud Properties (V)[2]	waterCoated_vis_JD_12bands.dat	initcld	ascii
Cloud Properties (IR)[2]	waterCoated_ir_JD_12bands.dat	initcld	ascii
Dust Properties (V)[2]	Dust_vis_wolff2010_JD_12bands.dat	initcld	ascii
Dust Properties (IR)[2]	Dust_vis_wolff2010_JD_12bands.dat	initcld	ascii

Notes:

1. Files are horizontal resolution-specific.
   - New files will be required for different horizontal resolutions (see below).
2. Files are RT wavelength resolution-specific.

Namelist

- The namelist (called "mars" or "restart") contains runtime options for modifying the simulation.
- A sample "mars" file contains the following:

```
&inputnl
  runnumx = 2014.11
  dlat = 5.0     jm = 36    im = 60   nlay = 24
  psf  = 7.010   ptrop = 0.0008
  dtm  = 2.0     nc3  =  8
  tautot = 0.3  rptau = 6.1  conrnu = 0.03
  taue = 480.0  tauh = 1.5    tauid = 0.0  tauih = 0.0
  rsetsw = 1
  cloudon = .false.
  active_dust = .true.
  active_water = .false.
  microphysics = .true.
  co2scav = .true.
  timesplit = .false.
  albfeed = .false.
  latent_heat = .false.
  vdust = .true.
  icealb = 0.4  icethresh_depth = 5.0
  dtsplit = 30.0
  h2ocloudform = .false.
 /
```

Namelist Parameters

Key Parameters:

- #### Full List of Parameters:

Parameter	Type	Description	Units	Notes
runnumx	real	run identifier		
dlat	real	degrees between latitude grid points	degrees	
jm	integer	number of latitude grid points		
im	integer	number of longitude grid points		
nlay	integer	number of layers		
psf	real	average surface pressure	mbar	
ptrop	real	pressure at the tropopause	mbar	
dtm	real	requested time step	minutes	
tautot	real	visible dust optical depth at the reference pressure level		
rptau	real	the reference pressure level tautot uses	mbar	
taue	real	requested run time	hours	
tauh	real	history output frequency	hours	
tauid	real	starting time in days		leave 0 for now
tauih	real	starting time of run	hours	0 for cold starts; time of 1st record of a warm start file
nc3	integer	a full pass through COMP3 is done every NC3 time steps		
rsetsw	integer	cold start / warm start flag		1 for cold starts; 0 for warm starts
lday	integer	day of a Mars year corresponding to a given Ls.		

conrnu	real	dust mixing ratio scale height		not used in this version

Namelist Runtime Flags

Parameter	Type	Description	Units	Notes
cloudon	logical	radiatively active water ice clouds		
active_dust	logical	radiatively active water vapor		
microphysics	logical	call MICROPHYS		always use true
co2scav	logical	simple treatments of CO_2 cloud scavenging		
timesplit	logical	timesplitting on		dtsplit $\neq$ 1
albfeed	logical	surface water ice albedo feedback		
latent_heat	logical	water latent heat effects		surface and atmosphere
vdust	logical	read and use dust map		
icealb	real	albedo value of surface ice		when albfeed = .true.
icethresh_depth	real	depth of ice required to reset icealb	microns	when albfeed = .true.
dtsplit	real	requested timesplit DT	seconds	when timesplit = .true.
h2ocloudform	logical	h2o cloud formation		

Day of Year (LDAY)

Ls	Day of Year	Ls	Day of Year
0	173	200	578
10	193	210	594
20	213	220	610
30	234	230	626
40	256	240	641
50	277	250	657
60	300	257.4	668
70	322	257.8	0
80	344	260	3
90	366	270	19
100	388	280	34
110	410	290	50
120	431	300	66

130	451	310	83
140	471	320	100
150	490	330	117
160	509	340	135
170	527	350	154
180	545	359.9	172
190	562	0	173

3. Cold Start

There are 2 types of runs: Cold Starts and Warm Starts

1. Cold Start: initialized with an isothermal atmosphere & no winds at time $= 0$.
2. Warm Start: initialized from a previous run ("spun-up") at time $\neq 0$.

We will start with learning how to do a Cold Start.

Steps for a Cold Start are:

1. Move executable to run directory and change to that directory

```
<(local)>$ cp gcm2.3 ../run/
<(local)>$ cd ../run/
```

1. Edit `mars` file
2. Execute the code

```
<(local)>$ ./gcm2.3 <mars> m.out &
```

Standard history files are fortran binaries:

1. `fort.11` , then `fort.11_0002` , `fort.11_0003` , etc: contain bulk of information
2. `fort.45` , then `fort.45_0002` , `fort.45_0003` , etc: secondary information
3. `fort.51` , then `fort.51_0002` , `fort.51_0003` , etc: used for warm starts
4. `fort.91` , then `fort.91_0002` , `fort.91_0004` , etc: also used for warm starts

- Each file nominally contains 10 sols of output (you can modify this)
- NOTE: if these default settings are changed, changes will also be required in the analysis pipeline.

4. GCM EXERCISE

It's time to practice!

We have designed a few tasks that will require running the GCM on your system. These exercises will help reinforce the concepts we're discussing.

TASK 1: Run from a Cold Start

The first exercise focuses on running the GCM from a cold start. The first simulation we'll run is a 10 sol (240 hour) simulation that starts at L_S 90. Otherwise, we'll use the default physics options in the tutorial namelist (`mars_tutorial` file).

Steps:

1. Create a new directory to execute the model in (run directory) and populate it with `gcm2.3`, `mars_tutorial`, `htest`, and the `/data` directory (plus its contents). These files and directory are located inside subdirectories within the `legacy-mars-global-climate-model` directory that was created when you initially installed the GCM.

2. Rename the `mars_tutorial` file to `mars` with the command:

```
<(local)>$ mv mars_tutorial mars
```

>

1. In the `mars` file, change the starting L_S (`lday`) to the value appropriate for L_S 90 (`lday= 366` and the length of the simulation to 240 hours (`taue = 240.0`).

2. Execute the simulation with the command:

```
<(local)>$ ./gcm2.3 < mars > m.out&
```

>

1. After the simulation finishes, run `htest` (`<(local)>$ ./htest`) on the first record of the last `fort.11` file (`_0002`) with `J=18`, `I=1`, and `L=24`. You should see something very similar to:

```
History file name:  fort.11_0002
Record number?  1
J, I, L (Which are: Lat, Lon, Layer)  18, 1, 24

 Run number: 2014.11
```

```
  History file name:  fort.11_0002
         Run number:  2014.11
      Record number:       1
               Grid:  J = 18    I =  1    L = 24

                 Ls =       94.47
             RSDIST =      2.7285
             DECMAX =     25.2193
                TAU =      240.00
             TOFDAY =        0.00
Time at Grid Point =       12.00

             TAUTOT =      0.3000
              RPTAU =        6.10

         TOPOG(J,I) =   9688.4170  ----->  -2.6044 km
          ALSP(J,I) =      0.2795
       SURFALB(J,I) =      0.2795
        ZIN(J,I,1) =     69.3150
               GIDN =      0.0545       GIDS =      0.0805

                PSF =      7.0100
               GASP =      6.9672
GASP: Global Average Surface Pressure

              PTROP =  8.0000E-04
             P(J,I) =      8.1164
        TSTRAT(J,I) =    191.1051
           T(J,I,L) =    226.8797
           U(J,I,L) =     -3.9678
           V(J,I,L) =      3.9435

            GT(J,I) =    268.0048
      STEMP(J,I,1) =    210.7421       SDEPTH( 2) =
0.0075 m
      STEMP(J,I,5) =    210.3161       SDEPTH(10) =
0.0961 m

       CO2ICE(J,I) =  0.0000E+00
       ALICEN      =  0.6000           ALICES      =
0.5000
       EGOCO2N     =  0.8000           EGOCO2S     =
1.0000
      STRESSX(J,I) = -1.5937E-03       STRESSY(J,I) =
2.3552E-03
      TAUSURF(J,I) =     0.43410

      fuptopv(J,I) =   113.12054       fuptopir(J,I)  =
```

> ```
> 214.48886
>       fdntopv(J,I) =   449.26962
>      fupsurfv(J,I) =   115.78580        fupsurfir(J,I) =
> 291.84625
>      fdnsurfv(J,I) =   414.32309        fdnsurfir(J,I) =
> 52.13027
>
>        NPCFLAG =  F
>    Water vapor =  4.4126E-10
> ```
>
> - You will notice that the L_S has advanced to $\approx 95^\circ$.

---

5. Warm Start

The second method for starting a simulation is through a warm start, which means we will initialize a new simulation from a previous run.

Steps for a Warm Start are:

1. Move executable to run directory and change to that directory

```
<(local)>$ cp gcm2.3 ../run/
<(local)>$ cd ../run/
```

1. Rename `mars` to `restart`

```
<(local)>$ mv mars restart
```

1. Identify `fort.*_****` files required and copy them into run directory

2. As an example, assuming we want to warm start from `fort.*_0002`, rename the `fort.*` files without the extensions:

```
<(local)>$ mv fort.11_0002 fort.11
<(local)>$ mv fort.45_0002 fort.45
<(local)>$ mv fort.51_0002 fort.51
<(local)>$ mv fort.91_0002 fort.91
```

1. When warmstarting, the model will read the first record of the fort.11, fort.45, fort.51, and fort.91 files and begin the simulation from that timestamp. This timestamp also needs to be specified in the restart file as the `tauih` value. To identify the starting time, run `htest` on the `fort.11` file from which you will restart, and read the output value of time `TAU`.

```
<(local)>$ ./htest
History file name: fort.11
Record number? 1
```

```
J, I, L (Which are: Lat, Lon, Layer) 17, 1, 24
```

1. Edit `restart` file
   - set `rsetsw = 0`
   - set `tauih` to the value found in previous step

1. Execute the code

```
<(local)>$ ./gcm2.3 <restart> m.out &
```

History file sequencing will then be:

1. `fort.11`, then `fort.11_0003`, `fort.11_0004`, etc
2. `fort.45`, then `fort.45_0003`, `fort.11_0004`, etc
3. `fort.51`, then `fort.51_0003`, `fort.11_0004`, etc
4. `fort.91`, then `fort.91_0003`, `fort.11_0004`, etc

   - Note that this sequencing is based on our example of warm starting from `fort.*_0002`
   - Each file nominally contains 10 sols of output (you can modify this)
   - You may want to rename the `fort.11`, `fort.45`, `fort.51`, and `fort.91` files with the `_0002` extension before processing them. For example:

```
<(local)>$ mv fort.11 fort.11_0002
```

---

6. History Files

Characteristics

- Each file has two parts:
  - Header Record
  - Time-Dependent Records

- Each history file (`fort.11`, `fort.11_0002`, etc.) has 160 time-dependent records
  - In the nominal set-up, this covers 10 sols
    - Output every 1.5 hours; 16 outputs per sol
    - You can change this in the mars/restart file (`tauh`)
    - We recommend...

Header Record

- Written once at the beginning of each `fort.11` file from `mhisth.f`
- Code for reading header:

```
character(len=7) :: version
  integer  :: jm, im, layers, nl, ntracer
  real*4   :: runnum, dsig(layers), dxyp(jm), sdepth(nl)
  real*4   :: grav, rgas, cp, stbo, xlhtc, kapa, cmk, decmax, eccn
  real*4   :: orbinc, vinc, alicen, alices, egoco2n, egoco2s
  real*4   :: topog(jm,im), alsp(jm,im), zin(jm,im,nl)
  real*4   :: npcwik
  logical  :: npcflag(jm,im)

                     .


  read(20) runnum, jm, im, layers, nl, ntrace, version
  read(20) dsig, dxyp, grav, rgas, cp, stbo, xlhtc, kapa,
 *          cmk, decmax, eccn, orbinc, vinc, sdepth, alicen,
 *          alices, egoco2n, egoco2s, npcwikg, gidn, gids
  read(20) topog, alsp, zin, npcflag
```

- Variable Descriptions:

VARIABLE	DESCRIPTION	UNITS
runnum	The run number	
jm	Number of latitude grid points	
im	Number of longitude grid points	
layers	Number of layers in the atmosphere below the stratosphere	
nl	Number of layers in the soil model	
ntrace	Number of tracers	
version	Version number	
time	Elapsed time from the start of the run	hours
dsig(L)	$d\sigma$ - the layer thickness in σ coordinates	
sigma(K)	σ - values of σ at the model levels	
dxyp(J)	The area of each grid point at latitude J	m^2
ptrop	Pressure of the tropopause	mbar
psf	Input global surface pressure	mbar
tautot	Input (global) dust optical depth at the reference pressure	
rptau	Reference pressure for dust optical depth (TAUTOT)	mbar
nc3	Full COMP3 is done every nc3 time steps	
cp	Heat capacity of CO_2 gas	J kg^{-1} K^{-1}
grav	Acceleration due to gravity	m s^{-2}
rgas	Gas constant for Mars	J kg^-1 K^{-1}

stbo	Stefan-Boltzmann constant	J m^{-2} s^{-1} K^{-4}
xlhtc	Latent heat of CO_2	J kg^{-1}
decmax	Obliquity (maximim solar declination)	
eccn	Orbital eccentricity	
orbinc	Inclination of the orbit to the ecliptic	
vinc	VINC - 90° is the true anomaly when L_S = 0	
sdepth(NL)	Depth at the mid-point of each soil layer. (m)	
alicen	Albedo of CO_2 surface ice in the north polar cap	
alices	Albedo of CO_2 surface ice in the south polar cap	
egoco2n	Emissivity of CO_2 surface ice in the north polar cap	
egoco2s	Emissivity of CO_2 surface ice in the south polar cap	
jequator	J index of the equator	
npcwikg	Initial north polar cap water ice (kg)	
topog(J,I)	Surface topography (-geopotential)	m^2 s^{-2}
alsp(J,I)	Surface albedo	
zin(J,I,NL)	Surface thermal inertia	J m^{-2} K^{-1} $s^{-1/2}$
npcflag(J,I)	Logical flag, true if polar cap exists at this grid point	

Time-Dependent Records

- Written every `tauh` hours from `mhistv.f`
- Code for reading one record:

```
integer :: nc3, ncycle
 real*4  :: tau, ls, rsdist, tofday, psf, ptrop, tautot
 real*4  :: rptau, sind, gasp
 real*4  :: p(jm,im)
 real*4  :: t(jm,im,layers), u(jm,im,layers), v(jm,im,layers)
 real*4  :: gt(jm,im), co2ice(jm,im), tstrat(jm,im), tausurf(jm,im)
 real*4  :: ssun(jm,im), stemp(jm,im,nl)
 real*4  :: qtrace(jm,im,layers,ntrace), qcond(jm,im,ntrace)
 real*4  :: fuptopv(jm,im), fdntopv(jm,im)
 real*4  :: fupsurfv(jm,im),fdnsurfv(jm,im)
 real*4  :: fuptopir(jm,im), fupsurfir(jm,im), fdnsurfir(jm,im)
 real*4  :: surfalb(jm,im), dheat(jm,im,layers), geop(jm,im,layers)

                         .


 read(20) tau, ls, rsdist, tofday, psf, ptrop, tautot,
*         rptau, sind, gasp
```

```
read(20) nc3, ncycle

read(20) p
read(20) t
read(20) u
read(20) v
read(20) gt
read(20) co2ice
read(20) stressx
read(20) stressy
read(20) tstrat
read(20) tausurf
read(20) ssun
read(20) qtrace
read(20) qcond
read(20) stemp
read(20) fuptopv, fdntopv, fupsurfv, fdnsurfv
read(20) fuptopir, fupsurfir, fdnsurfir
read(20) surfalb
read(20) dheat
read(20) geop
```

- Variable Descriptions:

VARIABLE	DESCRIPTION	UNITS
time	Elapsed time from the start of the run	hours
tofday	Time of day at 0° longitude	hours
Ls	Seasonal date	degrees
rsdist	Square of the Sun-Mars distance	AU^2
psf	Initial global surface pressure	mbar
ptrop	Pressure at the tropopause	mbar
sind	Sine of the sub-solar latitude	
tautot	Input (global) dust optical depth at the reference pressure	
rptau	Reference pressure for dust optical depth (TAUTOT)	mbar
gasp	Global average surface pressure	mbar
nc3	Full COMP3 is done every nc3 time steps	
p(j,i)	PI (Surface pressure - P_{trop})	mbar
t(j,i,l)	Atmosphere temperature	K
u(j,i,l)	Zonal wind	m s^{-1}
v(j,i,l)	Meridional wind	m s^{-1}
tstrat(j,i)	Stratosphere temperature	K

gt(j,i)	Ground temperature	K
co2ice(j,i)	Amount of CO_2 ice on the ground	kg m^{-2}
stressx(j,i)	Surface stress - zonal component (carried at PI points)	N m^{-2}
stressy(j,i)	Surface stress - meridional component (carried at PI points)	N m^{-2}
tausurf(j,i)	Dust optical depth (in visible) at the surface	
ssun(j,i)	Solar energy absorbed by the atmosphere	W m^{-2}
qtrace(j,i,l,n)	Tracer mass mixing ratio	kg kg^{-1}
qcond(j,i,n)	Amount of tracer (n) on the ground	kg m^{-2}
stemp(j,i,nl)	Sub-surface soil temperature	K
fuptopv(j,i)	Upward visible flux at the top of the atmosphere	W m^{-2}
fdntopv(j,i)	Downward visible flux at the top of the atmosphere	W m^{-2}
fupsurfv(j,i)	Upward visible flux at the surface	W m^{-2}
fdnsurfv(j,i)	Downward visible flux at the surface	W m^{-2}
fuptopir(j,i)	Upward IR flux at the top of the atmosphere	W m^{-2}
fupsurfir(j,i)	Upward IR flux at the surface	W m^{-2}
fdndurfir(j,i)	Downward IR flux at the surface	W m^{-2}
surfalb(j,i)	Surface albedo	
dheat(j,i,l)	Total diabatic heating rate	K sol^{-1}
geop(j,i,l)	Geopotential	m^{2} s^{-2}

7. Changing Model Resolution

Horizontal Resolution

- Cold Starts only
- Steps to take:

1. In `mars` : change `DLAT` , `JM` , `IM`
2. In `modules.f90` : change `L_J` , `L_I`
3. In `input.f` : change surface fields and dust map
    - *You must have fields and a dust map at the proper resolution*
4. In `initpbl.f` : change `ZAVGTG` (initial ground temperatures) and soil/ice locations
5. In `init1.f` : change initial locations of surface H_2O ice

Vertical Resolution

- Cold Starts only
- Steps to take:

1. In `mars` : change `NLAY` and maybe `PTROP`
2. In `modules.f90` : change `L_LAYERS` and `DSIG` array
    - *The sum of the `DSIG` array always needs to be 1.0.*

> # 8. GCM EXERCISES
>
> **It's time to practice!**
>
> We have designed a few tasks that will require running the GCM on your system. These exercises will help reinforce the concepts we've discussed.
>
> ## TASK 2: Run from a Warm Start
>
> The second exercise focuses on running the GCM from a warm start. The simulation we will run now will be a continuation of our first simulation (from TASK 1).
>
> ### Steps:

> 1. Create a new run directory and populate it with `gcm2.3`, `mars`, `htest`, and the `/data` directory (plus its contents) from the TASK 1 run directory, and rename the `mars` file `restart` (`<(local)>$ mv mars restart`).
>
> 2. Copy the `fort.*_0002` files from the TASK 1 run directory into this new run directory and rename them without the `_0002` extension (change `fort.11_0002 to fort.11`, `fort.45_0002 to fort.45`, etc.).
>
> 3. Use `htest` (`<(local)>$ ./htest`) to determine the hour (`tau`) from which you will start the new simulation.
>
> 4. In the `restart` file, change the starting hour (`tauih`) to the value found in Step 3 and and toggle the warmstart flag (`rsetsw`) from 1 to 0.
>
> 5. Execute the simulation with the command:
>
> ```
> <(local)>$ ./gcm2.3 <restart> m.out&
> ```

>

> 1. After the simulation finishes, run `htest` on the first record of the last `fort.11` file (`_0003`) with `J=18`, `I=1`, and `L=24`. You should see something very similar to:
>
> ```
> History file name:  fort.11_0003
> Record number?  1
> J, I, L (Which are: Lat, Lon, Layer)  18, 1, 24
>
> Run number: 2014.11
>
>
>   History file name:  fort.11_0003
>          Run number:  2014.11
>       Record number:       1
>                Grid:  J = 18    I =  1    L = 24
>
>                 Ls =       99.04
>             RSDIST =      2.7092
>             DECMAX =     25.2193
>                TAU =      480.00
>             TOFDAY =        0.00
> Time at Grid Point =       12.00
>
>             TAUTOT =      0.3000
>              RPTAU =        6.10
>
>         TOPOG(J,I) =   9688.4170  ----->  -2.6044 km
>          ALSP(J,I) =      0.2795
> ```

```
      SURFALB(J,I) =      0.2795
        ZIN(J,I,1) =     69.3150
              GIDN =      0.0545        GIDS =      0.0805

               PSF =      7.0100
              GASP =      6.9102
GASP: Global Average Surface Pressure

             PTROP =  8.0000E-04
            P(J,I) =      8.1203
       TSTRAT(J,I) =    192.7632
          T(J,I,L) =    226.7106
          U(J,I,L) =     -4.3707
          V(J,I,L) =      5.3277

           GT(J,I) =    267.6625
      STEMP(J,I,1) =    209.8746        SDEPTH( 2) =
0.0075 m
      STEMP(J,I,5) =    209.0226        SDEPTH(10) =
0.0961 m

       CO2ICE(J,I) =  0.0000E+00
       ALICEN      =  0.6000            ALICES      =
0.5000
       EGOCO2N     =  0.8000            EGOCO2S     =
1.0000
      STRESSX(J,I) = -2.8061E-03        STRESSY(J,I) =
3.5275E-03
      TAUSURF(J,I) =     0.48165

      fuptopv(J,I) =   114.52032        fuptopir(J,I)  =
213.56181
      fdntopv(J,I) =   453.28476
     fupsurfv(J,I) =   116.20900        fupsurfir(J,I) =
290.11661
     fdnsurfv(J,I) =   415.83743        fdnsurfir(J,I) =
51.76165

       NPCFLAG =  F
   Water vapor =  8.8870E-08
```

- You will notice that time (TAU) has advanced 240 hours (10 sols) from the TASK 1 simulation.

TASK 3: Run with Modified Runtime Physics Options

The third exercise has two parts and focuses on running the GCM from with modified options for the treatment of dust. Instead of using radiatively active transported dust (`ACTIVE_DUST = .TRUE.`), these simulations will use prescribed dust in the vertical (`ACTIVE_DUST = .FALSE.`) with globally uniform and constant (`VDUST = .FALSE.`) total column dust optical depths that the user can set to any value (`TAUTOT = VALUE`).

We will execute two simulation with `TAUTOT` values of 0.3 and 2.0, respectively, which represent low and high dust loading cases. We will test these new options by warm starting a simulation from the end of the TASK 1 simulation, which means that the warm start files used will be the same as those used in TASK 2.

TASK 3.1: Global Dust Optical Depth = 0.3

Steps:

1. Create a new run directory and populate it with `gcm2.3` , `restart` , `htest` , and the `/data` directory (plus its contents) from the TASK 2 run directory.

2. Copy the `fort.*_0002` files from the TASK 1 run directory into this new run directory and rename them without the `_0002` extension.

3. In the `restart` file, change the following flags to:

- `active_dust = .false.`
- `vdust = .false.`

1. Also in the `restart` file, verify that `tautot = 0.3`

2. Execute the simulation.

3. After the simulation finishes, run `htest` on the first record of the last `fort.11` file (`_0003`) > with `J=18` , `I=1` , and `L=24` . You should see something very similar to:

```
History file name:  fort.11_0003
Record number?  1
J, I, L (Which are: Lat, Lon, Layer)  18, 1, 24

Run number: 2014.11


  History file name:  fort.11_0003
         Run number:  2014.11
      Record number:      1
```

```
              Grid:  J = 18    I =  1    L = 24

                Ls =       99.04
            RSDIST =      2.7092
            DECMAX =     25.2193
               TAU =      480.00
            TOFDAY =        0.00
 Time at Grid Point =        12.00

            TAUTOT =      0.3000
             RPTAU =        6.10

        TOPOG(J,I) =   9688.4170  ----->  -2.6044 km
         ALSP(J,I) =      0.2795
      SURFALB(J,I) =      0.2795
        ZIN(J,I,1) =     69.3150
              GIDN =      0.0545       GIDS =      0.0805

               PSF =      7.0100
              GASP =      6.9155
 GASP: Global Average Surface Pressure

             PTROP =  8.0000E-04
            P(J,I) =      8.1335
       TSTRAT(J,I) =    191.7084
          T(J,I,L) =    226.1000
          U(J,I,L) =     -2.6611
          V(J,I,L) =      5.0604

           GT(J,I) =    267.7570
      STEMP(J,I,1) =    210.1247        SDEPTH( 2) =
0.0075 m
      STEMP(J,I,5) =    208.9654        SDEPTH(10) =
0.0961 m

       CO2ICE(J,I) =  0.0000E+00
       ALICEN      =  0.6000            ALICES       =
0.5000
       EGOCO2N     =  0.8000            EGOCO2S      =
1.0000
      STRESSX(J,I) = -3.0583E-03        STRESSY(J,I) =
4.2290E-03
      TAUSURF(J,I) =     0.41011

      fuptopv(J,I) =   116.24009        fuptopir(J,I)  =
213.04865
      fdntopv(J,I) =   453.28494
     fupsurfv(J,I) =   117.04001        fupsurfir(J,I) =
290.54834
     fdnsurfv(J,I) =   418.81110        fdnsurfir(J,I) =
51.34554

      NPCFLAG =   F
   Water vapor =   7.9585E-08
```

- You will notice that the ground temperature (`GT`) at this grid point is ≈ 267.75 K and the (near-> surface, since we chose L = 24) air temperature (T) is ≈ 226.1 K.

TASK 3.2: Global Dust Optical Depth = 2.0

Steps:

1. Create a new run directory and populate it with `gcm2.3` , `restart` , `htest` , and the `/data` directory (plus its contents) from the TASK 2 run directory.

2. Copy the `fort.*_0002` files from the TASK 1 run directory into this new run directory and rename them without the `_0002` extension.

3. In the `restart` file, change the following flag to:

- `tautot = 2.0`

1. Also in the `restart` file, verify that:

- `active_dust = .false.`
- `vdust = .false.`

1. Execute the simulation.

2. After the simulation finishes, run `htest` on the first record of the last `fort.11` file (`_0003`) with `J=18` , `I=1` , and `L=24` . You should see something very similar to:

```
History file name:  fort.11_0003
Record number?  1
J, I, L (Which are: Lat, Lon, Layer)  18, 1, 24

Run number: 2014.11


  History file name:  fort.11_0003
         Run number:  2014.11
      Record number:       1
               Grid:  J = 18    I =  1    L = 24
```

```
                Ls =       99.04
            RSDIST =      2.7092
            DECMAX =     25.2193
               TAU =      480.00
            TOFDAY =        0.00
 Time at Grid Point =        12.00

            TAUTOT =      2.0000
             RPTAU =        6.10

        TOPOG(J,I) =   9688.4170  ------>  -2.6044 km
         ALSP(J,I) =      0.2795
      SURFALB(J,I) =      0.2795
        ZIN(J,I,1) =     69.3150
              GIDN =      0.0545       GIDS =      0.0805

               PSF =      7.0100
              GASP =      6.9340
 GASP: Global Average Surface Pressure

             PTROP =  8.0000E-04
            P(J,I) =      8.0227
       TSTRAT(J,I) =    191.3996
          T(J,I,L) =    234.7205
          U(J,I,L) =      2.7318
          V(J,I,L) =      3.8033

           GT(J,I) =    260.3738
      STEMP(J,I,1) =    221.2417        SDEPTH( 2) =
0.0075 m
      STEMP(J,I,5) =    212.0935        SDEPTH(10) =
0.0961 m

       CO2ICE(J,I) =  0.0000E+00
       ALICEN      =  0.6000           ALICES      =
0.5000
       EGOCO2N     =  0.8000           EGOCO2S     =
1.0000
      STRESSX(J,I) =  1.5366E-04       STRESSY(J,I) =
4.8486E-03
      TAUSURF(J,I) =     2.69408

      fuptopv(J,I) =   114.68562       fuptopir(J,I)  =
119.80155
      fdntopv(J,I) =   453.28500
     fupsurfv(J,I) =    69.66179       fupsurfir(J,I) =
259.86200
     fdnsurfv(J,I) =   249.27484       fdnsurfir(J,I) =
120.63870
```

> ```
>        NPCFLAG =  F
>    Water vapor =  6.5969E-08
> ```
>
> - You will notice that the daytime ground temperature is cooler and the daytime near-surface air temperature is warmer than in the previous simulation (TASK 3.1). This is expected since we've significantly increased the atmospheric dust loading.

Configure Directory Structure

< Optional >

While not necessary, it may be useful to place the different directories described above in different locations on your computer.

```
/code, /documentation, /tutorial, /analysis

      - where the code and documents reside and where the
  executable will be created
      - does not require much disk space


/run

      - where the executable will be run and where the output
  files will be created
      - significant disk space required, so it's often in a
  different location (scratch, etc)


/data

      - where required input files reside
      - default location is ```/run/data```
      - you'll point to this directory in the source code if it
  isn't in default location
```

Update Paths to Input Files

- BEFORE MODIFYING SOURCE CODE: make sure you're in a location where making changes is OK
- Change paths in input.f, laginterp.f90, and initcld.f
- As an example, modying input.f for path to topography file:

```
OPEN(UNIT=9,
```

```
!     *  FILE='data/topog37x60.mola_intel',
     *  FILE='/scr1/jsmith/gcm/data/topog37x60.mola_intel',
     *  STATUS='OLD',FORM='UNFORMATTED')
      READ(9) BOUNDUM
      CLOSE(9)
```

- Make sure to change the paths to all required input files.

```
In [ ]:
```

NASA Ames Legacy Mars Global Climate Model README

Official Public Release

This software has reached end of life and will not receive further updates aside from critical bug fixes.

Installing, Compiling, and Running The Legacy GCM

Introduction

Welcome to the Mars Climate Modeling Center (MCMC) Legacy Mars Global Climate Model (GCM) installation tutorial. By the end of this tutorial, you will know how to configure, compile, and run the GCM, and how to check the initial model results for accuracy. The analysis pipeline tutorial will build on what you learn here by expanding on model analysis capabilities.

The GCM presented here is extensively documented in Haberle et al. 2019 (*Documentation of the NASA/Ames Legacy Mars Global Climate Model: Simulations of the present seasonal water cycle*).

Requirements

This tutorial has been tested on Mac, Linux, and Windows 10. The commands below are assumed to be utilized in the terminal on Mac and Linux machines, and in either the native terminal or Cygwin on Windows machines (depending on the version of the Windows). If you need Cygwin for a Windows machine, you can find instructions for download and installation here.

FORTRAN Compiler

A FORTRAN compiler is required to compile the GCM. We recommend the Intel or GNU compiler. While the Intel (ifort) FORTRAN compiler requires purchasing a license, the GNU (gfortran) FORTRAN compiler is freely available. Information on the Intel compiler can be found here:

https://www.intel.com. The GNU compiler can be installed with a package manager such as Homebrew, MacPorts or Cygwin, or directly from the source code. Once you have installed the compiler, make sure you have the path to it included in your bash or csh profile.

Installing and Compiling the GCM

The first steps are to install and compile the GCM on your machine.

Installation: Clone the repository

```
% git clone https://github.com/nasa/legacy-mars-global-climate-model.git
```

This will produce a directory called `legacy-mars-global-climate-model`. Navigate into that directory and list its contents:

```
% cd legacy-mars-global-climate-model
% ls -l
```

The following directories will be visible:

```
./documentation          # contains GCM documentation
./code                   # contains the GCM source code
./run                    # is where you will run the model
./run/data               # contains the required input files for the GCM
./analysis               # contains a simple analysis routine for checking a simulation
./analysis/validation    # contains sample plots from the default settings
./tutorial               # contains files used in this tutorial
```

> **A Note on Directory Configuration**
>
> While not necessary, it may be useful to place the different directories described above in different locations on your computer. The source code and data files are relatively small and can be placed anywhere. The GCM history files can take up large amounts of disk space (a typical 2 year run uses ~20 gigabytes) so you may want to place the `run`/ directory in a scratch directory or somewhere similar. Note that any changes you make to the default directory structure need to be taken into account in the instructions below, so if you are unsure about how to do this, it is probably best to start with the default structure.

Compilation: Makefile & Compiler Compatibility

From the main model directory (`legacy-mars-global-climate-model/`), navigate to the source code (`code/`) directory:

```
% cd code
```

The Makefile is set up to use the GNU Fortran (gfortran) compiler by default. Settings for the Intel (ifort) compiler are also available. You will need to modify the Makefile to use ifort. To do this, open the Makefile and comment the gfortran options and uncomment the ifort options. The original lines are:

```
#F90_COMP  = ifort
#F_OPTS    = -c -O2
F90_COMP = gfortran
F_OPTS   = -c -O3 -finit-local-zero -frecord-marker=4
```

and the lines modified for ifort will instead be:

```
F90_COMP = ifort
F_OPTS   = -c -O2
#F90_COMP  = gfortran
#F_OPTS    = -c -O3 -finit-local-zero -frecord-marker=4
```

Once the Makefile is ready, you can proceed with compiling the model. First, remove all object files (`*.o`), and module files (`*.mod`) to ensure a clean build by typing:

```
% make clean
```

Next, compile the code by typing:

```
% make
```

which creates an executable file called `gcm2.3` . Note that warnings that are not fatal may appear during the compile. If the executable is written, the compilation was successful. If fatal errors are produced during the compile and the compile fails, no executable will be created. In this case, check that the compiler can be found on the command line (i.e., set the `$PATH` correctly) and that the Makefile is set up properly for your compiler.

Running the Model

Before running the model, you will need to move (or copy) the executable file (`gcm2.3`) to the `run/` directory as well as the tutorial namelist file (`tutorial/mars_tutorial`). Navigate into that directory, and list its contents (note that this command assumes you're using the default directory structure):

```
% cp gcm2.3 ../run/
% cd ../run/
% ls
```

You will see the following in this directory, which are all the necessary components for running the model:

```
gcm2.3           # the executable
mars_tutorial    # the input file that includes runtime options
```

The `mars_tutorial` file has the default set-up for your initial test simulation. The parameters and options in the `mars_tutorial` file are described in more detail below, but for now, no editing of this file is needed.

To run the model, type:

```
% ./gcm2.3 <mars_tutorial> m.out &
```

In this command, `m.out` is the name of the output log file that will be produced, and the `&` will make the model run in the background.

If the model is running correctly, you will see fort.11, fort.51, and fort.91 files being created in the current directory (`run/`). The initial default simulation will simulate 20 sols (Martian days) and take approximately an hour to complete.

Analyzing GCM Output

Checking the Default Simulation

To check that the simulation ran properly, we will run a simple analysis routine called `htest.f90` . Navigate to the `tutorial/` directory:

```
% cd ../tutorial/
```

To compile `htest.f90` using gfortran, type:

```
% gfortran -c historymod.f90
% gfortran -o htest htest.f90 historymod.o
```

To compile using ifort, type:

```
% ifort -c historymod.f90
% ifort -o htest htest.f90 historymod.o
```

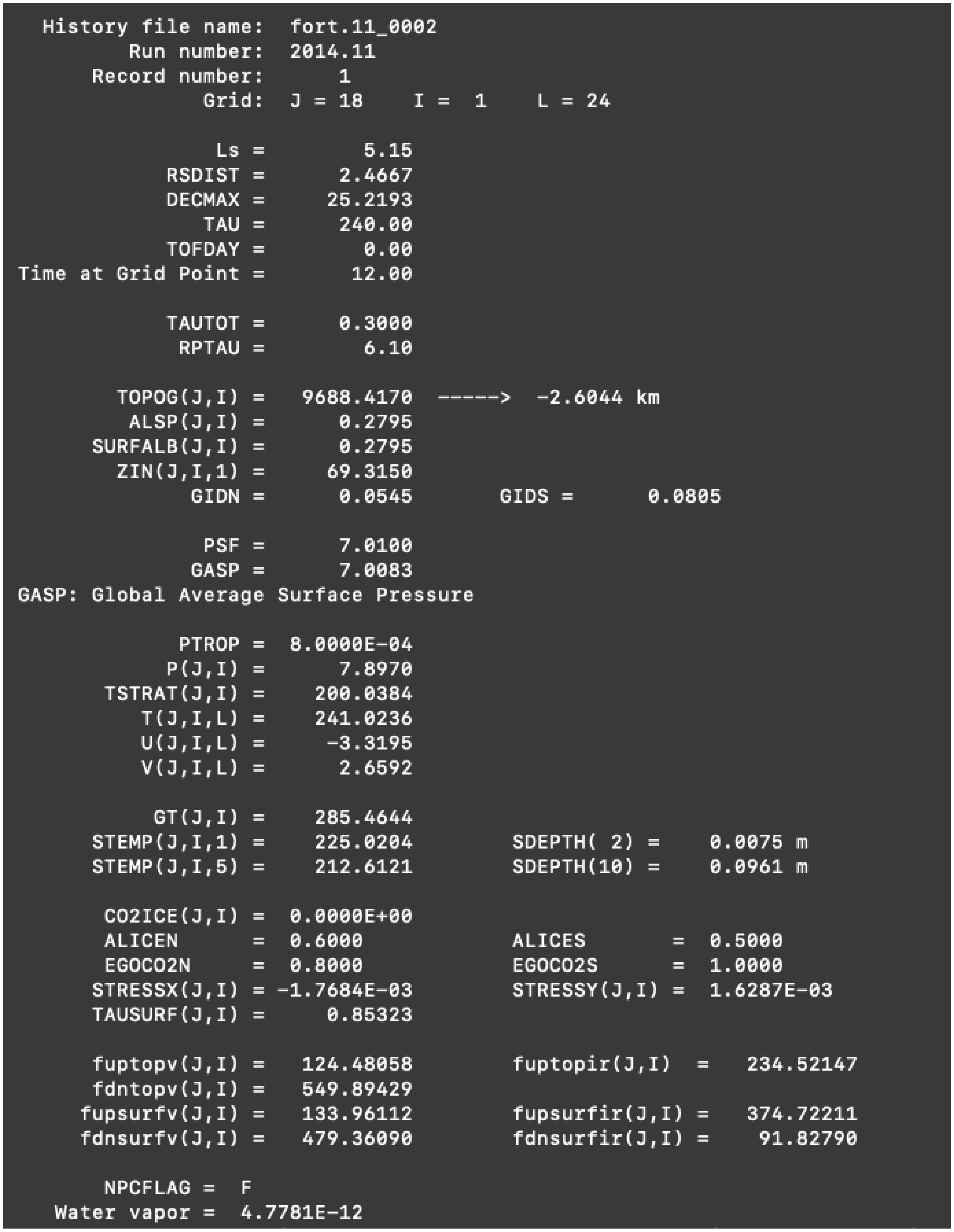
Next, copy the `htest` executable (`htest.exe` in Windows) and `htest.in` input file to the `run/` directory and navigate to it:

```
% cp htest htest.in ../run/   #  cp htest.exe htest.in ../run/ in Windows
% cd ../run/
```

`htest` is a routine that prints out information from one location (in latitude, longitude, and vertical coordinate) at one time step of the simulation. To print out information from latitude index number 18 (the equator), longitude index 1 (-180 East), and vertical coordinate index 24 (first atmospheric layer above the ground) at the first output from the 20th martian day of simulation, type:

```
% ./htest < htest.in
```

The output from `htest` will be:



```
  History file name:  fort.11_0002
         Run number:  2014.11
      Record number:       1
               Grid:  J = 18    I =  1    L = 24

                Ls =        5.15
            RSDIST =      2.4667
            DECMAX =     25.2193
               TAU =      240.00
            TOFDAY =        0.00
Time at Grid Point =       12.00

            TAUTOT =      0.3000
             RPTAU =        6.10

        TOPOG(J,I) =   9688.4170  ----->  -2.6044 km
         ALSP(J,I) =      0.2795
      SURFALB(J,I) =      0.2795
        ZIN(J,I,1) =     69.3150
              GIDN =      0.0545       GIDS =      0.0805

               PSF =      7.0100
              GASP =      7.0083
GASP: Global Average Surface Pressure

             PTROP =  8.0000E-04
            P(J,I) =      7.8970
       TSTRAT(J,I) =    200.0384
          T(J,I,L) =    241.0236
          U(J,I,L) =     -3.3195
          V(J,I,L) =      2.6592

           GT(J,I) =    285.4644
      STEMP(J,I,1) =    225.0204        SDEPTH( 2) =    0.0075 m
      STEMP(J,I,5) =    212.6121        SDEPTH(10) =    0.0961 m

       CO2ICE(J,I) =  0.0000E+00
       ALICEN      =  0.6000            ALICES       =  0.5000
       EGOCO2N     =  0.8000            EGOCO2S      =  1.0000
      STRESSX(J,I) = -1.7684E-03        STRESSY(J,I) =  1.6287E-03
      TAUSURF(J,I) =     0.85323

      fuptopv(J,I) =   124.48058        fuptopir(J,I)  =   234.52147
      fdntopv(J,I) =   549.89429
     fupsurfv(J,I) =   133.96112        fupsurfir(J,I) =   374.72211
     fdnsurfv(J,I) =   479.36090        fdnsurfir(J,I) =    91.82790

       NPCFLAG =  F
   Water vapor =  4.7781E-12
```

You should reproduce the information above quite closely. There may be variations in the fourth or fifth decimal place due to hardware and compiler differences but they should be small. If you reproduce these `htest` results, you have successfully installed and run the GCM.

Notices:

Disclaimers