NASA Ames Mars Global Climate Model (MGCM) Version 3.0 User Guide

Courtney Batterson[1,2], Richard Urata[1,2], Melinda Kahre[2], Amanda Brecht[2], Kathryn Steakley[2], John Wilson[2], Victoria Hartwick[1,2], Alex Kling[1,2], and Sonny Harman[2]

[1]Bay Area Environmental Research Institute, Moffett Field, CA
[2]NASA Ames Research Center, Moffett Field, CA

March 21, 2023

Contents

Part I

Quick Start Guide: Run MGCM 3.0 Out-of-the-Box on NAS

In this first part of the User Manual, we provide instructions for downloading MGCM 3.0, minimally modifying the runscript, and running the default simulation on the NASA Advanced Supercomputing (NAS) system. This process is parsed into three steps:

1. Download the model from GitHub
2. Compile the model
3. Run the model

> **NOTE:** These instructions assume you have all the required software listed in Chapter 3 (Section 3.1) and that you are running the model on the NAS system. Additional steps are required for running the model on non-NAS systems (see Chapters 3 and 4).

Step 1: Download the model from GitHub

MGCM 3.0 is built by downloading the NOAA-GFDL Atmospheric Model Version 4 (AM4) and installing the physics packages for Mars developed by the Mars Climate Modeling Center (MCMC) on top of it. These instructions guide the user through cloning (downloading) the AM4 model and the Mars physics package from GitHub, then patching them together.

To begin, open a terminal, `ssh` into Pleiades (`pfe`), and clone AM4 into your preferred directory:

```
user@pfe:~$ module load pkgsrc/2021Q2
 ↪  # necessary for git submodules, updates git version
user@pfe:~$ git clone --recursive --branch 2021.03
 ↪  https://github.com/NOAA-GFDL/AM4.git
```

Delete the `ice_param` module from the `AM4/src/` directory:

```
user@pfe:~$ cd AM4/src/
user@pfe:~$ rm -rf ice_param
```

Stay in the `src/` directory, create and switch to `mars_branch`, then add the Mars physics submodules to AM4:

```
user@pfe:~$ git checkout -b mars_branch
user@pfe:~$ git submodule add
 ↪  https://www.github.com/nasa/AmesGCM.git
user@pfe:~$ git submodule add
 ↪  https://github.com/NOAA-GFDL/mkmf.git
user@pfe:~$ git submodule add
 ↪  https://github.com/NOAA-GFDL/FRE.git
```

Apply the patches that stitch the Mars physics and the AM4 model together:

```
user@pfe:~$ cd AmesGCM/patches/
user@pfe:~$ ./apply_patch.sh
```

Congratulations! You have successfully built MGCM 3.0 on the NAS system.

Step 2: Compile the model

Go back three directories (from `AmesGCM/patches/`) to the `exec/` directory and compile the code:

```
user@pfe:~$ cd ../../../exec/ # from AmesGCM/patches
user@pfe:~$ ./compile.archives
```

The compilation takes $\sim$ 5 minutes. When complete, a compiled version of MGCM 3.0 will be stored in `exec.intel.mars3.0`.

Step 3: Run the model

Edit and then submit the runscript to the PBS system on NAS. Open `fms_mars_default`, located in the `exec/` directory, and modify the following PBS settings:

```
! Set the account to be billed:
PBS -W group_list=sXXXX
! Enter your email address:
PBS -M user@email.com
```

Save `fms_mars_default` and return to the terminal. Submit the run on the Pleiades supercomputing system from the command line with:

```
user@pfe:~$ qsub fms_mars_default # from the exec/ directory
```

That's it! You'll receive an email at the address you provided in the PBS settings when the simulation executes and when it completes. Barring any other changes in the runscript, the simulation will output files onto Lou (`lfe`) at:

```
/u/$USER/FV3/xanadu/am4_mars_runs/fms_mars_default/history/
```

Restart files will be ported to your `nobackup/` directory on Pleiades:

```
/nobackup/$USER/FMS_MARS_runs/am4_mars_runs/
 ↪  fms_mars_default/restart/
```

Part II

User Manual

Introduction

This document serves as a manual for obtaining and using the NASA Ames Mars Global Climate Model (MGCM) version 3.0. The MGCM simulates the Martian climate using an external finite-volume dynamical core to predict the global atmosphere given various planetary parameters and physical parameters.

> **NOTE:** This public release of the MGCM (version 3.0) includes a reduced set of physics compared to internal versions that have been used in recent publications (e.g., Bertrand et al., 2020, Haberle et al., 2019 and Kahre et al., 2022). We will release more sophisticated physics in future public releases.

A brief overview of the physics in MGCM 3.0 is included here – a more complete description will be included in later editions of the User Manual. The MGCM 3.0 dynamical core is the NOAA-GFDL Finite-Volume Dynamical Core (Xanadu version), which is publicly available on the NOAA-GFDL GitHub Repository at https://github.com/NOAA-GFDL and described in Harris et al. (2021). The dynamical core integrates the fluid mechanical primitive equations in time over the globe. It also provides the horizontal grid framework for the model, which is described in greater detail in Chapter 1.

The physics in MGCM 3.0 have been implemented and tested by the Mars Climate Modeling Center (MCMC) and are described in Haberle et al. (2019). Physical parameterizations in the model include surface properties such as thermal inertia and albedo; a ground and subsurface temperature scheme; a planetary boundary layer (PBL) scheme that provides vertical diffusive mixing in the boundary layer and vertical diffusion throughout the atmospheric column; carbon dioxide sublimation/condensation physics; an analytically prescribed dust field; and a two-stream radiative transfer code based on correlated-k's. The MGCM utilizes Local True Solar Time (LTST), which means that the length of the day is constant throughout the year and that the sun is always the highest in the sky at local noon (LTST = 12.0). A more detailed description of the physics in MGCM 3.0 is provided in Chapter 1.

All software components in MGCM 3.0 were developed and supplied by the NASA Ames MCMC. Surface maps for thermal inertia, albedo, and residual north polar cap boundary were created by team members and collaborators at Oregon State University. Dust optical properties were supplied by Michael Wolff at Space Science Institute in Boulder, CO. All other input data was supplied by NASA.

Chapter 1

Model Description

MGCM 3.0 has a finite-volume dynamical core and a cubed-sphere grid, both of which were developed by NOAA-GFDL. The vertical grid has a hybrid sigma-pressure vertical coordinate that transitions from sigma near the surface to pure pressure near the model top. The altitude at the top of the model varies between ∼100–130 km depending on which vertical grid is selected in the runscript. The vertical grid options and how to implement them are discussed in Chapter 2. Here, we provide a brief description of the horizontal and vertical grid structures in the model, and we summarize the physics included in MGCM 3.0.

1.1 The Dynamical Core

The MGCM has a mass-conserving finite-volume dynamical core that was developed at GFDL based on the transport and shallow water algorithms from Lin and Rood (1996, 1997). The Lagrangian vertical discretization scheme is as described in Lin (2004) and the pressure gradient force is computed from a modified version of Green's theorem as described in Lin (1997). The model grid has the cubed-sphere geometry described in Lin and Putman (2007) that enables high horizontal resolution simulations, higher order finite-volume numerics, and grid nesting. The dynamical core can be run in hydrostatic or non-hydrostatic mode, but this release of the MGCM (version 3.0) only supports hydrostatic mode. In hydrostatic mode, the vertical grid is defined by a hybrid sigma-pressure coordinate system, which is terrain-following near the surface and transitions to pure pressure in the upper atmosphere.

1.1.1 Horizontal Grid Structure

The cubed-sphere grid, illustrated in Figure 1.1, is comprised of six tiles: one centered over each of the poles and four around the equator. This provides near-uniform global coverage, eliminating the singularity at the poles inherent to traditional latitude-longitude grids. Relatively uniform grids such as the cubed-sphere allow for efficient computation on massively parallel machines, making high-resolution simulations practical.

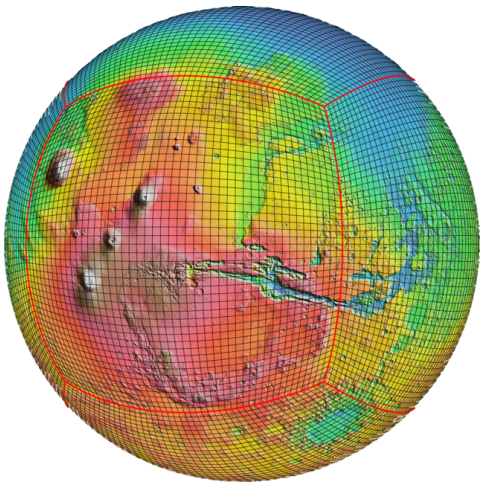

Figure 1.1: An illustration of the c48 (1.875°) cubed-sphere grid over Martian topography. Tile boundaries are shown in red.

The horizontal resolution of the model depends on the number of grid cells there are on each tile, which is determined by the grid selected at initialization. The desired grid is selected by setting the `NCX` parameter in the runscript. For example, `NCX = 24` will give a c24 simulation, which is the default setting (additional grids are described in Chapter 4). The name of the grid (e.g., c24) indicates the number of grid cells there are in the X and Y directions on each tile. The resulting horizontal resolution for `c24` is ∼3.75°, calculated by:

$$\text{resolution (degree)} = \frac{360°}{\text{NCX} * 4} \tag{1.1}$$

Or, equivalently, ∼221.5 km, calculated by:

$$\text{resolution (km)} = \frac{2\pi r}{\text{NCX} * 4} \tag{1.2}$$

where the radius of Mars (r) is 3,386 km and the number "4" refers to the four tiles surrounding the equator. The MCMC has carried out simulations using MGCM 3.0 at c12, c24, c48, c96, and c192 resolutions. Currently, the MCMC supports running MGCM 3.0 at two of those resolutions: c24, the default resolution, and c48, a higher resolution. Instructions for moving to c48 are included in Chapter 4.

NOTE: The horizontal grid resolution is not truly uniform. The cubed-sphere grid necessitates that each tile be distorted to more closely resemble a sphere than a cube.

1.1.2 Vertical Grid Structure

MGCM 3.0 utilizes a hybrid sigma-pressure vertical coordinate, transitioning from sigma near the surface to pure pressure at the top of the atmosphere (Simmons & Burridge, 1981). The sigma coordinates are terrain following, which is useful for solid surfaces with large varying terrains such

as Mars. The hybrid sigma-pressure vertical grid defining the layer interfaces is given by Equation 1.3:

$$P[k] = ak[k] + bk[k] * P_{sfc} \quad \text{for distinct } k \in \{1, \ldots, nlevs\} \tag{1.3}$$

where $nlevs = klayers + 1$, the layer index is k, the pressure component of the hybrid coordinate is ak, and the sigma component of the hybrid coordinate is bk. ak has units of pressure in Pascal, while bk is unitless and varies between 0 and 1. The top of the model corresponds to $k = 1$. The full pressure grid is constructed in `fv_eta.F90`, which references the ak and bk values for the desired vertical grid stored in:

```
AM4/src/GFDL_atmos_cubed_sphere/tools/fv_eta_mars.h
```

The default MGCM pressure grid has 56 layers. The pressure and approximate altitude (calculated using on a constant scale height of 10 km and a reference surface pressure of 705 Pa) of each layer midpoint in the 56-layer grid are listed in Table 1.1. The resolution varies from ∼5 m near the surface to ∼4 km near ∼80 km (1×10^{-1} Pa), and there are 10 pure-pressure layers that start near ∼1 Pa. The top three layers above ∼80 km have lower vertical resolution, and they are ignored for analysis due to potential model top boundary condition effects.

Other vertical grids included in MGCM 3.0 are a 30-layer grid, a 37-layer grid, and a 24-layer pure sigma grid for backward compatibility with the Legacy MGCM (Haberle et al., 2019). The pressure and altitude of the layers in the other vertical grids are illustrated in Figure 4.2 in Chapter 4, and explicit pressures and altitudes for the other vertical grids are provided in:

```
AM4/src/AmesGCM/diagnostics/
```

Table 1.1: Pressures and altitudes of the layer midpoints in the default 56-layer vertical grid. Altitude is calculated using the reference surface pressure from the model grid (705 Pa) and a 10-km scale height.

Pressure [Pa]	Altitude [km]	Pressure cont. [Pa]	Altitude cont. [km]	Pressure cont. [Pa]	Altitude cont. [km]
(top) 0.003	122.9	35.2	30.0	474.3	4.0
0.013	108.8	44.5	27.6	502.1	3.4
0.038	98.2	55.5	25.4	528.5	2.9
0.076	91.3	68.3	23.3	553.5	2.4
0.13	86.1	83.0	21.4	576.7	2.0
0.21	81.2	99.6	19.6	598.1	1.6
0.34	76.4	118.2	17.9	617.5	1.3
0.53	71.9	138.6	16.3	634.8	1.0
0.82	67.6	161.0	14.8	650.1	0.81
1.2	63.4	185.1	13.4	663.2	0.61
1.9	59.4	210.7	12.1	674.4	0.44
2.8	55.3	237.8	10.9	683.5	0.31
4.1	51.4	266.1	9.7	690.8	0.20
6.0	47.7	295.3	8.7	696.3	0.12
8.4	44.3	325.2	7.7	700.2	0.07
11.7	41.0	355.5	6.8	702.8	0.031
15.8	38.0	385.8	6.0	704.2	0.011
21.0	35.2	415.9	5.3	704.8	0.002 (bottom)
27.4	32.5	445.5	4.6		

1.2 Physics

MGCM 3.0 includes several Mars physics parameterizations and they are described here. More detailed descriptions of the model physics will be included in future versions of the User Manual.

1.2.1 Surface and Sub-Surface Properties and the CO_2 Cycle

Model topography is determined from Mars Orbiter Laser Altimeter (MOLA) retrievals, and surface thermal inertia and albedo are derived from Viking and Mars Global Surveyor (MGS) Thermal Emission Spectrometer (TES) measurements. The south polar residual CO_2 ice cap (SPRC) is not explicitly prescribed in the model, but is instead represented by regions of high surface thermal

inertia and albedo detected by TES. The depth and extent of the seasonal CO_2 ice caps are self-consistently determined by a surface energy balance that considers surface thermal inertia, surface albedo, topography, and the CO_2 ice albedo at the north and south poles. The prescribed albedos are chosen to produce a good fit to Viking data (Figure 1.2), which is a standard method for validating the annual mass variation due to the CO_2 cycle in MGCMs (Haberle et al., 2008). Surface and sub-surface temperatures are computed from a soil conduction calculation that includes a surface heat balance at the upper boundary and constant temperature at the lower boundary. Computed surface temperatures partially determine the amount of CO_2 frost on the ground. CO_2 condensation occurs as necessary to maintain the CO_2 frost point temperature.

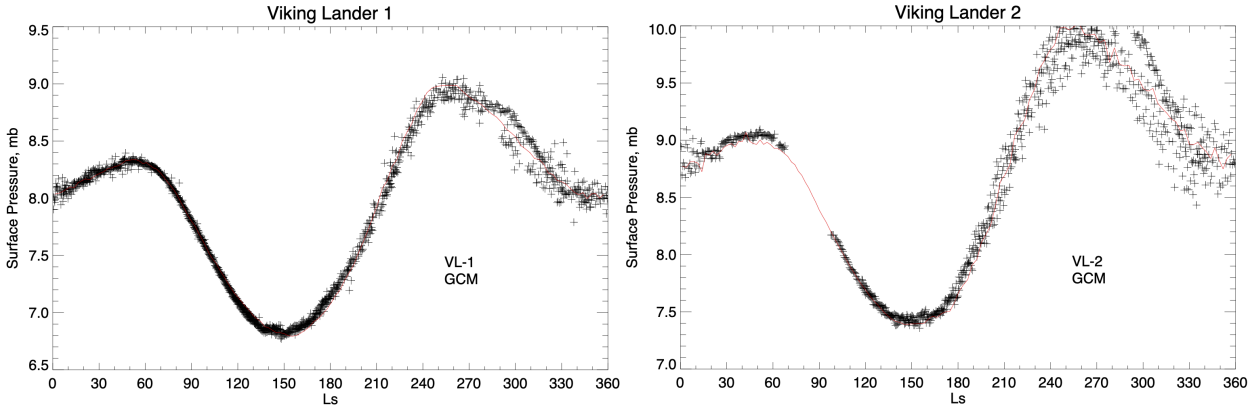

Figure 1.2: Daily mean observed (+) and simulated (red line) surface pressures at Viking Lander 1 (VL1; left) and VL2 (right). Simulated data has been interpolated and hydrostatically corrected to the locations of VL1 and VL2 for comparison.

1.2.2 The Planetary Boundary Layer (PBL)

Interactions between the surface and the atmosphere are represented by the PBL scheme, which predicts winds and temperatures throughout approximately the lowest scale height in the atmosphere. A level-2 Mellor and Yamada (1982) turbulence closure scheme predicts near-surface heat fluxes, and surface heat and momentum fluxes are handled according to Monin-Obukhov similarity theory. In addition to mixing in the boundary layer, the PBL scheme also provides diffusive mixing throughout the atmospheric column. A convective adjustment is performed every physical time step to remove any remaining superadiabatic temperature profiles, which ultimately results in more realistic near-surface air temperatures.

1.2.3 Radiative Transfer (RT)

The radiative transfer (RT) scheme is a 2-stream code that handles the radiative effects of airborne dust and gaseous CO_2. Gaseous CO_2 opacities are calculated from correlated-k tables (tabulated off-line), and Mie theory is used to compute the extinction efficiencies and scattering properties of dust. Rayleigh scattering by CO_2 is directly calculated. The RT code predicts fluxes and flux divergences from the optical properties of dust aerosols, which are largely dependent on their effective particle size distributions. Radiative heating and cooling rates are then computed from those fluxes. More information about the RT scheme is in Chapter 2.

1.2.4 Dust Prescription

The spatially and temporally evolving dust in MGCM 3.0 is prescribed in the horizontal and the vertical through a set of user-defined options (see Chapters 2 and 4). Airborne dust is radiatively active at both visible and infrared wavelengths, but can be set to be radiatively inert.

> **NOTE:** This public release of the MGCM does not currently include a water cycle or water-ice cloud microphysical schemes, but these will be included in a future public release of the MGCM.

1.2.5 Code Architecture

Figure 1.3 shows the order in which the physics routines are called in MGCM 3.0. All of the physics paramaterizations are stored in the source code under:

```
AM4/src/AmesGCM/
```

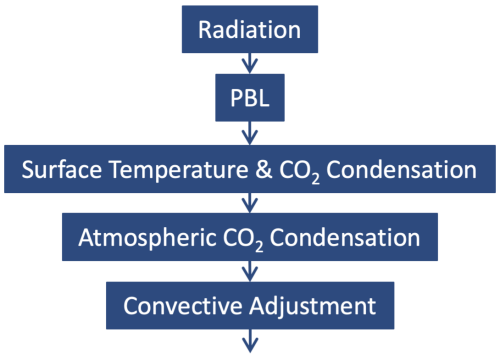

Figure 1.3: The order in which the physics modules are called in MGCM 3.0.

The MGCM 3.0 physics module begins with the radiation driver and is followed by the PBL scheme. After that, surface and sub-surface temperatures as well as surface CO_2 condensation are calculated. Next, the atmospheric CO_2 condensation module condenses CO_2 from the atmosphere to maintain air temperature at or above the CO_2 condensation temperature. Finally, after all of the physics routines have completed, a convective adjustment is performed to remove any unstable layers.

Chapter 2

Model Input Files

This chapter describes the input file structure and content in MGCM 3.0. First, we present the input files for the surface fields and the RT scheme. Then, we describe the various dust scenarios that can be used to inform the prescribed dust field in MGCM 3.0, and we provide a discussion about the "restart" files that inform "warm" starts. Finally, we describe the field and diag tables, which are created in the runscript and define tracer fields and output variables, respectively.

2.1 Surface Fields

All of the input files for MGCM 3.0 discussed in this chapter are located in the `data/` directory at:

```
AM4/src/AmesGCM/data/
```

Included in this directory are the surface topography, albedo, emissivity, and thermal inertia files:

```
mars_topo_mola16.nc            # topography
mars_TES_albedo8_new.nc        # albedo
mars_sfc_emissivity8.nc        # emissivity
mars_thermal_inertia8_2011.nc  # thermal inertia
```

Surface topography is $1/16°$ resolution MOLA data (Smith et al., 1999). Bare soil albedos and emissivities are $1/8°$ resolution MGS/TES data (Putzig & Mellon, 2007). Thermal inertias are derived by Tyler and Barnes (2014) and have been mapped onto a $1/8°$ grid for ingestion into the MGCM. All surface fields are re-gridded to the resolution of the simulation at runtime (Wilson et al., 2007). Regions of high thermal inertia represent the SPRC, as the permanent CO_2 ice cap is not explicitly represented in the MGCM. The seasonal CO_2 ice caps in both the north and south grow and recede according to a surface energy balance that considers thermal inertia, surface albedo, topography, and the CO_2 ice albedo, which is 0.65 for the northern ice and 0.43 in the south. The CO_2 ice albedos are namelist parameters that can be modified in the runscript. Their default values are chosen to produce a good fit to the Viking pressure cycle as shown in Figure 1.2. Finally, a

distribution of subsurface ice is prescribed following a spatial pattern consistent with gamma ray spectrometer (GRS) measurements (e.g. Boynton et al., 2002). The need for subsurface ice when modeling the CO_2 cycle is described in Haberle et al. (2008).

2.2 Radiation Code

MGCM 3.0 uses a two-stream correlated-k radiation code to account for the radiative effects of CO_2 and H_2O gas and atmospheric aerosols (Haberle et al., 2019; Toon et al., 1989). Dust and water ice clouds can optionally be radiatively active in the radiation code, but we note that this public release of the MGCM does not support a water cycle, so dust is the only radiatively active aerosol in MGCM 3.0. Additionally, this release of the MGCM does not support the physical transport of dust (radiatively active or inert) or surface sources/sinks. Instead, the radiation code sees prescribed dust fields (see Chapter 4 for options). The radiation code in MGCM 3.0 is based on 12 wavelength bands: 7 in the visible and 5 in the infrared (see Table 2.1).

Table 2.1: 12-Band spectral intervals for the radiation code.

Band	Wavelength Interval (μm)
Visible 1	4.50 – 3.24
Visible 2	3.24 – 2.48
Visible 3	2.48 – 1.86
Visible 4	1.86 – 1.31
Visible 5	1.31 – 0.80
Visible 6	0.80 – 0.40
Visible 7	0.40 – 0.24
Infrared 1	1000.0 – 60.0
Infrared 2	60.0 – 24.0
Infrared 3	24.0 – 12.0
Infrared 4	12.0 – 8.00
Infrared 5	8.00 – 4.50

The correlated-k and dust scattering property tables for the radiation code are listed below.

```
CO2H2O_IR_12_95_INTEL              # correlated-k visible table
CO2H2O_V_12_95_INTEL               # correlated-k infrared table
Dust_vis_wolff2010_JD_12bands.dat  # dust properties visible
Dust_ir_wolff2010_JD_12bands.dat   # dust properties infrared
```

For each of the twelve spectral intervals, correlated-k's were generated for a binary mixture of CO_2 and water vapor from a line-by-line code using the HITEMP database (Rothman et al., 2010) from

HITRAN (Gordon et al., 2022) for CO_2 and a version of the Schwenke database (Schwenke, 1998) for H_2O. The absorption coefficients were sorted, reordered in g space (cumulative probability space), and stored in a look-up table for use in the MGCM.

An offline Mie code was used to compute dust scattering properties (Q_{ext}, Q_{scat}, and g) based on the refractive indices of dust from Wolff et al. (2009). For each of the twelve spectral intervals, the Planck-weighted (215 K in the infrared; 6,000 K in the visible) properties were computed for 20 mono-disperse populations with sizes ranging from 0.1–50 μm. Finally, dust is described via a size distribution defined by an effective radius and variance (0.5 in MGCM 3.0). The scattering properties of this (fixed) dust distribution are then computed using the visible and infrared wavelengths for dust from Table 2.1.

2.3 Dust Scenario Files

The dust scenarios that inform the horizontal dust distribution are defined by the `dust_scenarios` variable in the runscript. Instructions for further modifying the dust prescription are provided in Chapter 4. We limit discussion here to describing the dust scenario files that are included in the release of MGCM 3.0.

All of the dust scenario files for MGCM 3.0 are listed below.

```
DustScenario_MY24.nc        DustScenario_MY34.nc
DustScenario_MY30.nc        DustScenario_Background.nc
DustScenario_MY31.nc        DustScenario_MGS.nc
```

The dust scenario files are automatically interpolated to the model grid upon initialization so that every scenario works for any model resolution. Among the included files are dust absorption maps for Mars Year 24 (MY24), MY30, MY31, and MY34 from the Montabone dust climatology (Montabone et al., 2015), in which the normalized dust column is specified as:

```
dustcol(lon,lat,Ls)
```

and the vertical distribution of the dust is zonally uniform and given by:

```
zmax(lon,lat,Ls)
```

There is also `DustScenario_Background.nc`, a dust scenario created to represent a typical annual dust cycle. It is a combination of the Montabone dust climatologies with additional processing that smooths out any short term dust events. It is therefore not specific to any particular Mars Year and thus represents a typical Mars dust cycle. Finally, there is a dust scenario called `DustScenario_MGS.nc` which was proposed by Montmessin et al. (2004) and has been used in multiple other studies.

2.4 Restart Files

At the end of every simulation, the model saves its current state in a series of files called "restart" files. These are tarred together and stored in the `restart/` directory on Pleiades:

```
/nobackup/$USER/FMS_MARS_runs/am4_mars_runs/
 ↪  fms_mars_default/restart/
```

At the beginning of a warm start, the requested restart file is copied to the `INPUT/` directory and untarred so that the model can reference the saved state. Restart files contain data stored on the native grid in tile-specific files. Data is saved from the last timestep of the iteration and the 5-digit code in the tarred file name indicates the sol number of the data in the file. For example, a file named `00668.restart.tar` would hold data from the last timestep of the 668^{th} day of a simulation. The variables stored in the restart files are only those required to reproduce the saved state of the simulation. Most notably, these include zonal wind (`u`), meridional wind (`v`), temperature (`T`), layer thickness (`delp`), and geopoential height (`phis`).

2.5 The Field Table

The field table is used to define additional fields, known as "tracers," in the simulation. MGCM 3.0 carries multiple (dust and water) tracers that are disconnected from any physics routines, are always zero, and are thus not supported in this release. These tracers are required for the radiation code and will be connected (with appropriate physics) and supported in a future public release of the model. Although MGCM 3.0 does not carry supported tracers, an in-line field table is present in the runscript. We therefore provide a brief description of it here.

Tracers are added to the field table by specifying their name (`'name'`), the module in which they are defined (`'module'`), their unit (`'units'`), and the name the tracer will have in the output file (`'longname'`). These parameters are formatted as follows:

```
'TRACER', 'module',     'name'
          'longname',   'user-defined-name-here'
          'units',      'user-defined-unit-here'
```

> **NOTE:** When running the model from a warm start, if the tracer `'name'` is not found in the restart files, then the tracer is initialized to zero.

2.6 The Diag Table

The diag table is where the output file types are defined and the variables to be written to those files are specified. The diag table in the runscript (the in-line diag table) lists the minimum fields that must be output by the model. There also exists an "external" diag table (`diag_table.ext`),

which is included because some users prefer to separate additional variables from those included by default in the in-line diag table.

> **NOTE:** The external table (`diag_table.ext`) omits the section defining the output filetype. It is primarily used for specifying additional variables to write out.

The following subsections provide more detail about the two parts of the in-line diag table: defining the output files and assigning variables to those files.

2.6.1 The Diag Table: Defining the Output Files

The first eight lines of the in-line `diag_table` define the output file names and the frequency at which output is stored in each of the files. MGCM 3.0 creates and stores variables in five netCDF files, but the user can add or remove files (or change the file names) if they so choose. The five files defined in the default runscript are:

- `grid_spec.nc`
- `fixed.nc`
- `atmos_daily.nc`
- `atmos_average.nc`
- `atmos_diurn.nc`

`grid_spec.nc` contains cubed-sphere geometry information needed for post-processing (e.g., conversion to a regular latitude-longitude grid; see Chapter 5). `grid_spec.nc` and `fixed.nc` are invariant and only need to be written once, but the model outputs each of these files every run iteration for consistency. The other three files (`atmos_*.nc`) store atmospheric variables that change over time. Each file type stores the variables at different output frequencies and the default settings are listed below:

```
! filename,      freq,  unit, format, time_unit, time_name,
'grid_spec',     -1,  'hours', 1,     'days',    'time',
'fixed',         -1,  'days',  1,     'days',    'time',
'atmos_daily',    6,  'hours', 1,     'days',    'time',
'atmos_average', 5,   'days',  1,     'days',    'time',
'atmos_diurn',    5,  'days',  1,     'days',    'time',
```

From left to right, the columns above define the name of the output file (`filename`), the output frequency (`freq`), the output unit (`unit`), the file format (`format`), the time unit (`time_unit`), and time name (`time_name`) for each of the five output files. Complete descriptions of each of these column parameters is listed below.

1. `'filename'` – The name of the file that will store the fields.
2. `'freq'` – The output frequency: an integer determining how often output is written to the file. If negative, output is written only once at the end of each iteration. If 0, output is written

every physical time step (`dt_atmos`).

3. `'unit'` – The time unit for the output frequency.
4. `'format'` – Should always be 1 for netCDF format.
5. `'time_unit'` – The output time axis unit. Options include: `'years'`, `'months'`, `'days'`, `'hours'`, `'minutes'`, `'seconds'`.
6. `'time_name'` – The output time axis name. This string must contain `'time'`.

Additionally, the user may add the following columns to more specifically define how often new input files are created and data are archived. The optional parameters are listed in column-order below.

7. `'new_file_freq'` – Determines how often a new file is created in `'new_file_unit'` time (integer). [optional]
8. `'new_file_unit'` – The unit for the new file frequency (e.g. `'days'`, `'hours'`). [optional]
9. `'start_time'` – A 6-integer string for writing the files. Must be in the format: "year, month, day, hour, minute, second." [optional]
10. `'file_dur'` – An integer setting the duration of the file period. [optional]
11. `'file_dur_unit'` – The string unit for the file duration. [optional]

2.6.2 The Diag Table: Assigning Variables to the Output Files

After defining the output files, the in-line diag table lists the minimum variables to be written to the output files. One variable is registered per line according to the following format:

```
'module_name', 'field_name', 'output_name', 'file_name',
 ↪  'time_sampling', 'time_method', 'spatial_opts', 'pack'
```

Descriptions of each of the above fields are provided in column order below.

- `'module_name'` – The submodule where the field is defined (e.g. `'dynamics'`, `'mars_physics'`, etc.).
- `'field_name'` – The field name as registered in the submodule above.
- `'output_name'` – The field name as it will appear in the output file.
- `'file_name'` – The output file that the field is assigned to. Options include: `'fixed'`, `'grid_spec'`, `'atmos_average'`, `'atmos_diurn'`, and `'atmos_daily'`.
- `'time_sampling'` – The sampling frequency. Must be `'all'` for every time step.
- `'time_method'` – The output frequency for the field. Options include:
  - `.true.` for an average of every timestep over the requested averaging interval [frequency]

  - `.false.` for instantaneous
  - `'min'` for minimum within an averaging period
  - `'max'` for maximum within an averaging period
  - `'diurnal24'` for hourly
  - `'diurnal12'` for every 2 hours
- `'spatial_opts'` – The location from which the field is sampled. Options include:
  - Global output (`'none'`)
  - A specific area: `'lonmin lonmax latmin latmax Pmin Pmax'` (pressure in Pa).
- `'pack'` – Determines the numeric type of the output values. Options include: 1 for double precision, 2 for float, 4 for 16-bit packed integers, and 8 for packed 1-byte.

The minimum required variables that are pre-defined in the in-line diag table are listed in Tables 5.1–5.5 in Chapter 5.

Chapter 3

Obtaining the Model

Chapters 3 and 4 provide more detailed instructions for downloading, compiling, and running MGCM 3.0 than that provided in Part I of the User Manual. Some of the settings required to run the model in non-NAS environments are included here, but MGCM 3.0 has yet to be tested on on non-NAS systems and fully documenting the necessary changes continues to be a work in progress.

3.1 Software Requirements

MGCM 3.0 is built by installing the Mars physics package developed by the MCMC on top of the AM4 model from NOAA-GFDL. The MCMC provides GitHub "patches" that integrate the Mars physics paramaterizations with the AM4 software to convert the model into a Mars simulator. The required software for downloading and running MGCM 3.0 are listed below.

- **A Fortran Compiler** MGCM 3.0 is primarily written in Fortran and includes some C, C++, HTML, and Shell scripts as well. As such, a Fortran compiler is necessary for running the model. We recommend gfortran or ifort, but intel Fortran is also supported.
- **netCDF4** MGCM 3.0 input and output files are in netCDF format, which is recognized by most of the commonly-used scientific programming languages today including Python, MATLAB, IDL, GrADS, and netCDF-Fortran.
- **MPI** Installation of an MPI library is required for parallel processing.
- **Git** MGCM 3.0 is released on GitHub and we use Git to install it.
- **Mars Physics Packages** MGCM 3.0 physics are hosted on NASA's GitHub. Instructions for cloning the repository are below.
- **NOAA-GFDL AM4** The AM4 model serves as the dynamical core for MGCM 3.0. Instructions for cloning the repository are below.

The following instructions assume you are working on NAS and therefore include NAS-specific lines of code for loading the relevant modules. For non-NAS users, these instructions assume you have installed a Fortran compiler, the netCDF4 and MPI libraries, and Git already. In the following pages, instructions are provided for cloning the NOAA-GFDL AM4 public release, cloning the MGCM physics packages, and patching them together to build MGCM 3.0.

3.2 Cloning the GitHub Repository

The physics packages for MGCM 3.0 are available on the NASA GitHub at https://www.github.com/nasa/AmesGCM. The AM4 dynamical core is available on the NOAA-GFDL GitHub at https://github.com/NOAA-GFDL/AM4.git.

First, clone the AM4 model to your preferred directory:

```
user@pfe:~$ module load pkgsrc/2021Q2
 ↪  # necessary for git submodules on NAS, updates git version
user@pfe:~$ git clone --recursive --branch 2021.03
 ↪  https://github.com/NOAA-GFDL/AM4.git
```

> **NOTE:** The `module load` line is NAS-specific. It loads a module required for working with GitHub. Non-NAS systems may require a similar package to perform the installation.

Next, go into the `AM4/src/` directory and remove the `ice_param` module:

```
user@pfe:~$ cd AM4/src/
user@pfe:~$ rm -rf ice_param
```

Stay in the `src/` directory. Create and switch to `mars_branch`:

```
user@pfe:~$ git checkout -b mars_branch
```

Add the Mars physics submodules to AM4:

```
user@pfe:~$ git submodule add
 ↪  https://www.github.com/nasa/AmesGCM
user@pfe:~$ git submodule add
 ↪  https://github.com/NOAA-GFDL/mkmf.git
user@pfe:~$ git submodule add
 ↪  https://github.com/NOAA-GFDL/FRE.git
```

Apply the patches to stitch together the AM4 model and the Mars physics:

```
user@pfe:~$ cd AmesGCM/patches
user@pfe:~$ ./apply_patch.sh
```

A directory called `AM4/` is created containing the following:

```
analysis/
bin/
container/
exec/
run/
src/
README.md
```

The source code for the model is in the `src/` directory, and the Mars physics packages specifically are stored in `src/AmesGCM/` . The model is run from the `exec/` directory, which houses the default runscript (`fms_mars_default`).

Chapter 4

Building and Running the Model

4.1 Building the Model

MGCM 3.0 is compiled by linking to the appropriate libraries and executing the compile script. This creates an executable that is then used to run the model. MGCM 3.0 can be run out-of-the-box on the NAS computing system using the `fms_mars_default` runscript as described in Part I. It can also be run on other systems and the default runscript can be modified at the user's discretion. In this section, we describe the process of compiling and running the model in more detail and include instructions for modifying the runscript.

4.1.1 Compiling MGCM 3.0

MGCM 3.0 is optimized for building and running on NAS and compiling the model on NAS is straightforward. However, there are software requirements that must be met in order to run the model. These were listed and explained in Chapter 3 for your reference. This chapter assumes you have the necessary software installed already.

To build the model with the default settings, go to the `exec/` directory and compile the code using the provided script. If you are following along from Chapter 3, you are likely still in the `patches/` directory:

```
AM4/src/AmesGCM/patches/
```

and you must go back three directories to `src/`, then go into `exec/`:

```
user@pfe:~$ cd ../../../exec/
```

You can double-check that you are in `AM4/exec/` using `pwd`, which should show:

```
AM4/exec/
```

Execute the compile script from there:

```
user@pfe:~$ ./compile.archives
```

This creates a compiled version of the MGCM called `exec.intel.mars3.0`.

Compile Options

Those who are utilizing NAS to run MGCM 3.0 will likely not need to change any settings in the compile script. However, modifications to the compile script are likely necessary for users running MGCM 3.0 in non-NAS environments. Specifically, the local compiler library and the netCDF and compiler directories need to be modified in order for the MGCM to run on non-NAS platforms. The following are some of the options in the compile script that might need modification:

- `platform` – The compiler to be used (currently supported are Intel and GNU. Intel is suggested for NAS).
- `NETCDF` – The directory of the netCDF environment.
- `NETCDFPATH` – Also the directory of the netCDF environment.
- `INTEL_LICENSE_FILE` – The directory containing the platform license.
- `execdir` – The directory containing the compiled source code and the executable.
- `execname` – The name of the executable.

Additionally, there are two options listed under `CPP_defs` that may have to be changed for non-NAS environments. These options are read by the compiler before it compiles the code:

- `MARS_GCM` – Activates calls to Mars physics. [required]
- `DMARS_GDIAGS` – Activates global sum diagnostics output (i.e. global mean surface pressure, global total water vapor or dust, etc.). Using this option will slow down compilation time. [optional]

Finally, the model can be compiled with optimizations for the specific CPU types available on NAS by defining the `optim` variable:

- Use `'bridge'` for Sandy Bridge or Ivy Bridge processors.
- Use `'well'` for Haswell or Broadwell processors.

4.2 Running the Model

After compiling MGCM 3.0, the model can be run by submitting a runscript to the NAS PBS system. In the `exec/` directory, the default runscript `fms_mars_default` produces a simulation with minimal modifications. In this section, we provide instructions for modifying and submitting that runscript to the queue for processing. The following subsections align with the headers in `fms_mars_default` to make navigating the runscript as straightforward as possible.

WARNING: The MCMC does not currently support running MGCM 3.0 on non-NAS systems, meaning that the MCMC does not provide instructions for running the MGCM outside of NAS and will not have answers to troubleshooting questions specific to non-NAS environments.

4.2.1 PBS Settings

The first several lines of the runscript define the NAS PBS options for the job submission. There are a few settings that should be modified to reflect the user submitting the job:

```
-W group_list=sXXXX ! the account to be billed
-M user@email.com   ! email for job status updates
```

Other PBS options that are commonly modified in the runscript include the queue the job will be submitted to, the number of nodes and processors requested, the node type, and the amount of walltime the job needs. The default values for these options are listed below.

```
-q normal ! the queue
-l select=3:ncpus=28:model=bro
 ↪  ! 3 broadwell nodes, 28 processors/node
-l walltime=08:00:00 ! HH:MM:SS
```

NOTE: More information on choosing the right number of processors for a job are in the next section (Section 4.2.2).

Complete documentation for these and additional PBS settings can be found on the NAS website at https://www.nas.nasa.gov/hecc/support/kb/portable-batch-system-(pbs)-overview_126.html

4.2.2 Execution Variables

The next part of the runscript defines the model grid structure and associated time step (δt). It also points to the input and output directories and sets the working directory. The variables in this section are listed in table 4.1 in the order in which they appear in the runscript.

Table 4.1: Execution Variables and Descriptions

Variable	Default Value	Description
name	fms_mars_default	the output directory
scriptname	$cwd/$name	the runscript path
classdir	am4_mars_runs	the parent output directory
workdir	/nobackup/$USER/FMS_MARS_runs ↪/$classdir/$name	the model execution directory
datadir	/nobackup/$USER/FMS_MARS_data	the input directory
platform	intel	the compiling environment (intel or GNU)
TILE_LAYOUT	3,4	the layout of the processor Required CPUs is 6*TILE_LAYOUT1*TILE_LAYOUT2
model_executable	$cwd/exec.$platform.am4 ↪/FMS_MARS.x	the executable path
homedir	$cwd	the executable directory
NCX	24	horizontal resolution. Default: 24 grid cells per cube face column/row (≈4x4°)
DTA (dt_atmos)	924	atmospheric (physics) time step; decrease when running with higher resolution
NKS (k_split)	1	the number of vertical remapping iterations per DTA; increase when running with higher resolution
NNS (n_split)	4	the number of advective time steps per NKS; increase when running with higher resolution
NPZ	56	defines the vertical grid to reference
NPZ_RST	0	number of vertical layers in restart file "0" if equal to NPZ)

NOTE: The variables `TILE_LAYOUT`, `DTA`, `NKS`, and `NNS` are unix variables that set the namelist variables `layout`, `dt_atmos`, `k_split`, and `n_split`, respectively, later in the runscript. They appear together at the top of the runscript for ease of use.

We *highly* recommend that the `name` parameter in the runscript matches the name of the runscript itself. This way, the runscript and the output directory to which that simulation writes to share the same name. Likely the only time `name` should match a pre-existing output directory name is when you want to extend a run out longer (i.e. perform a *continuation* warm start, see Section 4.2.3).

WARNING: Setting `name` to an existing output directory name may overwrite any pre-existing output files in that directory.

We support two horizontal resolutions in this release: c24 and c48. The former (c24) is a $\sim$3.75° simulation, and the latter (c48) is a $\sim$1.875° simulation. The physics (atmospheric) time step, `dt_amos`, is set by `DTA` and must divide evenly into the length of the Mars day as defined in the model (88,704 seconds). Generally, `DTA` should be decreased when running with finer horizontal resolution. We have found `DTA = 924` (default) or `DTA = 462` are good choices for the lower-resolution simulations we have tested (c24 and c48).

The advective (dynamical) time step is given by:

```
advective δt = total dynamical δt = dt_atmos/(k_split*n_split)
```

where `k_split` is the number of vertical remapping iterations per physics timestep (`dt_atmos`) and `n_split` is the number of advective time steps per `k_split`. The vertical remapping conservatively re-grids the Lagrangian layers onto a set of "Eulerian" reference coordinates that many of the physics paramaterizations use (Harris et al., 2021). This remapping resolves distortions in the vertical layers that might otherwise lead to stability problems. Ideally, `n_split` should be distinctly larger than `k_split`. The advective time step is governed by the Courant Friedrichs-Le (CFL) condition and is thus dependent on model resolution.

The total number of processors required to run the model is calculated automatically later in the runscript, but it is an important number to consider so that the user can request enough computing power in the PBS settings to run the MGCM. The total number of CPUs required for a run is six times the layout of the processor (`6*TILE_LAYOUT1*TILE_LAYOUT2`). The layout of the processor is given by `TILE_LAYOUT` which accepts two numbers indicating the number of CPUs to use on each processor in the X and Y directions. By default, `TILE_LAYOUT = 3,4` which means the number of CPUs used per cube face is $3 * 4 = 12$ and the total number of CPUs used is $12 * 6 = 72$.

NOTE: One caveat: `TILE_LAYOUT` is constrained by the resolution (`NCX`) of the simulation. Specifically, both of the numbers provided in `TILE_LAYOUT` must be less than `NCX/3` .

The total number of CPUs requested in the PBS settings must be greater than or equal to the number of CPUs required by the model. PBS options vary by machine, but for NAS systems the user specifies the node type, usually the Pleiades Broadwell Nodes (`model=bro`), and the number of nodes (**`select`**`=3` for the default case) to use. Each node type has a fixed number of processors (28 for Broadwell, `ncpus=28`) so the user will likely have to request more processing power than is necessary for a run. For the default simulation, the user must request at least three Broadwell nodes in order to have enough processors to run the model:

```
! Requesting 3 broadwell nodes and 28 processors/node
! for a total of 84 processors
-l select=3:ncpus=28:model=bro
```

4.2.3 Time Set-Up

This section of the runscript defines the simulation length and number of iterations. This is also where the user initializes the model from a "cold" or a "warm" start. Table 4.2 below lists the runtime variables.

Table 4.2: Runtime variables and descriptions

Variable	Default Value	Description
`dayslist`	`668 668`	the length (Mars Days) of each model integration
`num_executions`	`$dayslist`	the number of times the model is run
`RUNTYPE`	`0`	select a cold or warm start: initialize from scratch ("cold"; 0), continue from previous run ("continuation" 1), or restart from a different run ("warm"; 2)
`restartfile`	`/nobackup/$USER` `↪/FMS_MARS_runs` `↪/am4_mars_runs` `↪/fms_mars_default` `↪/restart` `↪/00668.restart.tar`	the path to the restart file if `RUNTYPE=2` (warm start from a file)

The first variable in Table 4.2, `dayslist`, accepts a list of numbers representing the number of Mars Days of data to be written to each output file. The second variable, `num_executions`, sets the number of iterations in the simulation and therefore the number of output files that will be created. By default, `num_executions` points to `dayslist`, which means the number of entries in `dayslist` indicates the number of output files that will be created by default. In other words, `dayslist` defines the run iteration. In the default case, (`dayslist = (668 668)`) meaning the simulation is set up for two run iterations and will write two sets of output files, each with 668 days (1 MY) of data. See Chapter 5 for more information on the output file structures.

The `RUNTYPE` variable determines whether or not the model will be initialized from a "cold" or a "warm" start. A cold start initializes the model on day zero whereas a warm start initializes the model from a previous simulation. In a "continuation" warm start (`RUNTYPE=1`), the model is initialized from where it left off. This setting is useful if you want to re-submit a runscript again and have the simulation run out further. When `RUNTYPE=2`, the model is warm-started from the restart file listed under `restartfile`. This allows you to specify which simulation – and *when* during that simulation – to initialize the model.

WARNING: When warm-starting from a specified file (`RUNTYPE=2`), the untarred contents of the file (the designated restart tar file) will overwrite the files in `workdir/INPUT/`. Also, if the model dates are the same, subsequent history and restart files in `history/` and `restart/` will be overwritten as well. A guaranteed way to avoid this is to rename the runscript and, therefore, the output directory (`name` from Table 4.1).

4.2.4 Initial Conditions

The parameters in this section define the input files for the dust scenario. These parameters are listed in Table 4.3 and additional information about the dust scenario is provided below.

Table 4.3: Initial Conditions

Variable	Default Value	Description
`RESET_DATE`	`0`	logical: 1 resets the time variable for every output file, 0 does not
`dust_scenarios`	`DustScenario_Background.nc`	the dust scenarios to cycle through
`APPEND_EXTERNAL_` `↪ DIAGTABLE`	`0`	logical: 1 reads in the external diag table, 0 does not
`diagtable_ext`	`$homedir/diag_table.ext`	the external diag table to read in

The `dust_scenarios` parameter accepts one or more dust scenario files. The dust scenario files included in MGCM 3.0 are listed in Chapter 2 and stored in the source code `data/` directory:

```
AM4/src/AmesGCM/data/
```

Broadly, there are two ways to inform the dust scenario:

Option 1: Specify one file to be referenced throughout the simulation.

```
! Option 1 Example:
set dust_scenarios  = (DustScenario_Background.nc)
```

Option 2: Specify a list of files for the model to iterate through throughout the simulation.

```
! Option 2 Example:
set dust_scenarios  = (DustScenario_MY30.nc
 ↪  DustScenario_MY31.nc)
```

More specifically, listing one file after `dust_scenarios` (**Option 1**) references the specified dust scenario for every iteration of the run. **Option 1** is the default setting. Listing multiple files after `dust_scenarios` (**Option 2**) initializes the dust scenario from the first file listed, then the model then cycles through the rest of the files with each run iteration. **Option 2** is especially useful for simulating multi-year dust scenarios. For example, if the model is initialized with the following settings:

```
dayslist       = (668 668)
num_executions = $dayslist
dust_scenarios = (DustScenario_MY30.nc DustScenario_MY31.nc)
```

then the simulation will reference the MY30 dust scenario for the first year of the run and the MY31 dust scenario for the second year of the run.

WARNING: The model will move on to the next dust scenario file in the list even if it has not simulated through a full year. In the example above, if `dayslist = (50 50)` then the first 50 days would reference the MY30 scenario and the next 50 days would reference the MY31 scenario.

4.2.5 Remaining Sections

After the dust scenario is set up, the section "Set Up Directory Structure" cleans the working directory, creates (or modifies, if pre-existing) the subdirectories that will hold the input, restart, output, and ASCII files, and copies the model executable to the working directory. The two sections after that define the in-line diag and field tables. This is also where the external diag table is referenced if `APPEND_EXTERNAL_DIAGTABLE = 1`.

The final modifiable section of the runscript is the namelist. The default value for each namelist variable is provided in `fms_mars_default` along with a description of the variable and its range of values. Some of the most commonly modified namelist settings are discussed in the next section (Section 4.3).

After the namelist, the rest of the runscript initializes the run. Unless the user is running MGCM 3.0 on a non-NAS environment, there are no other parts of the runscript that need modifying by the user.

WARNING: When running MGCM 3.0 on non-NAS environments, the user must take care to ensure all file paths throughout the runscript are updated for compatibility with their environment.

4.3 Customizing the Runscript

The NASA Ames MCMC supports limited customization of MGCM 3.0. This section provides instructions for modifying some of the default settings in the MGCM, including simulating the Martian atmosphere at higher resolution, changing the vertical grid, running with flat topography (a "billiard ball" run), and modifying the dust prescription.

4.3.1 Moving to Higher Resolution

There are two supported horizontal resolutions in MGCM 3.0: c24 and c48. As a reminder, the resolution is set by the `NCX` variable in the runscript. Increasing the resolution typically requires decreasing the value of `nord`, defined in the namelist under `&fv_core_nml`. `nord` sets the order of the del operator for the numerics: applied once (`nord=1`), the del operator is del-2; applied twice (`nord=2`), it is del-4; and applied thrice (`nord=3`), it is del-8. When `nord=3` (default), the runscript automatically changes `nord` if `NCX` >c24. However, this automatic setting can be overwritten if necessary by setting `nord` manually in `&fv_core_nml`. The recommended values for `nord` are listed in Table 4.4.

Table 4.4: Horizontal Grid Options

NCX	nord	Resolution (degrees)	Resolution (km)
c24	3	~4°	~221.5 x 221.5
c48	2	~2°	~110.8 x 110.8
c96	1	~1°	~55.4 x 55.4
c192	1	~0.5°	~27.7 x 27.7

Generally, increasing model resolution from c24 to c48 is straightforward and requires only the two modifications to the runscript:

1. Change the resolution: set `NCX=48`
2. Increase `k_split` and `n_split`: set `NKS=2` and `NNS=6`

The second modification increases the temporal resolution for stability. We recommend increasing the frequency of the vertical remapping loop (`k_split`) and the advection calculations (`n_split`) before decreasing the physics timestep (`dt_atmos`) or the radiative time step (`rad_calc_intv`). For the example above, we find that the default simulation at 2° horizontal resolution (c48) usually runs if `k_split=2` and `n_split=6`.

The physics time step `dt_atmos` is changed by setting `DTA`, which is nominally set to 924 seconds. This setting should be sufficient for most c24 and c48 simulations. The only constraint on `DTA` is that it must divide evenly into the length of the Mars day as it is defined in the model (88,704 seconds).

4.3.2 Changing Vertical Grids

Changing the vertical resolution of the model is simple and requires setting just one variable: `NPZ`. This variable is listed under "Execution Variables" in the runscript and is nominally set to the 56-layer vertical grid:

```
set NPZ   = 56    # number of vertical layers
```

The 56-layer grid was described in detail in Chapter 2. Equation 1.3 used to derive the vertical grids is also in Chapter 2. To change the vertical grid, simply set `NPZ` to the number of layers in the desired grid (either 56, 37, 30, or 24). The vertical grids included in the MGCM 3.0 are listed in order from highest to lowest resolution below. They are also illustrated in Figures 4.1 and 4.2.

- **56-layer grid (default)**: The uppermost layer midpoint is located at 0.003 Pa.
- **37-layer grid**: Lower resolution than L56, but more layers are defined above 0.1 Pa (see Figure 4.1). The uppermost layer midpoint is located at 0.005 Pa.
- **30-layer grid**: Lower resolution than L56 and L37. Has the lowest model top. The uppermost layer midpoint is located at 0.035 Pa.
- **24-layer grid**: Lowest resolution grid option. This grid is used in the Legacy version of the MGCM. The coordinate system is all sigma, with the top layer boundary located at $\sigma = 0$. The uppermost layer midpoint is located at 0.011 Pa.

All of the vertical grids in MGCM 3.0 have a lowest-layer midpoint between 2–5 m. Note that it is possible to change the vertical resolution of a simulation during a warm start. To do so, set `NPZ` to the desired vertical grid and `NPZ_RST` to the number of layers in the restart file (i.e. the number of layers in the previous simulation).

WARNING: Be sure to reset `NPZ_RST=0` for future runs.

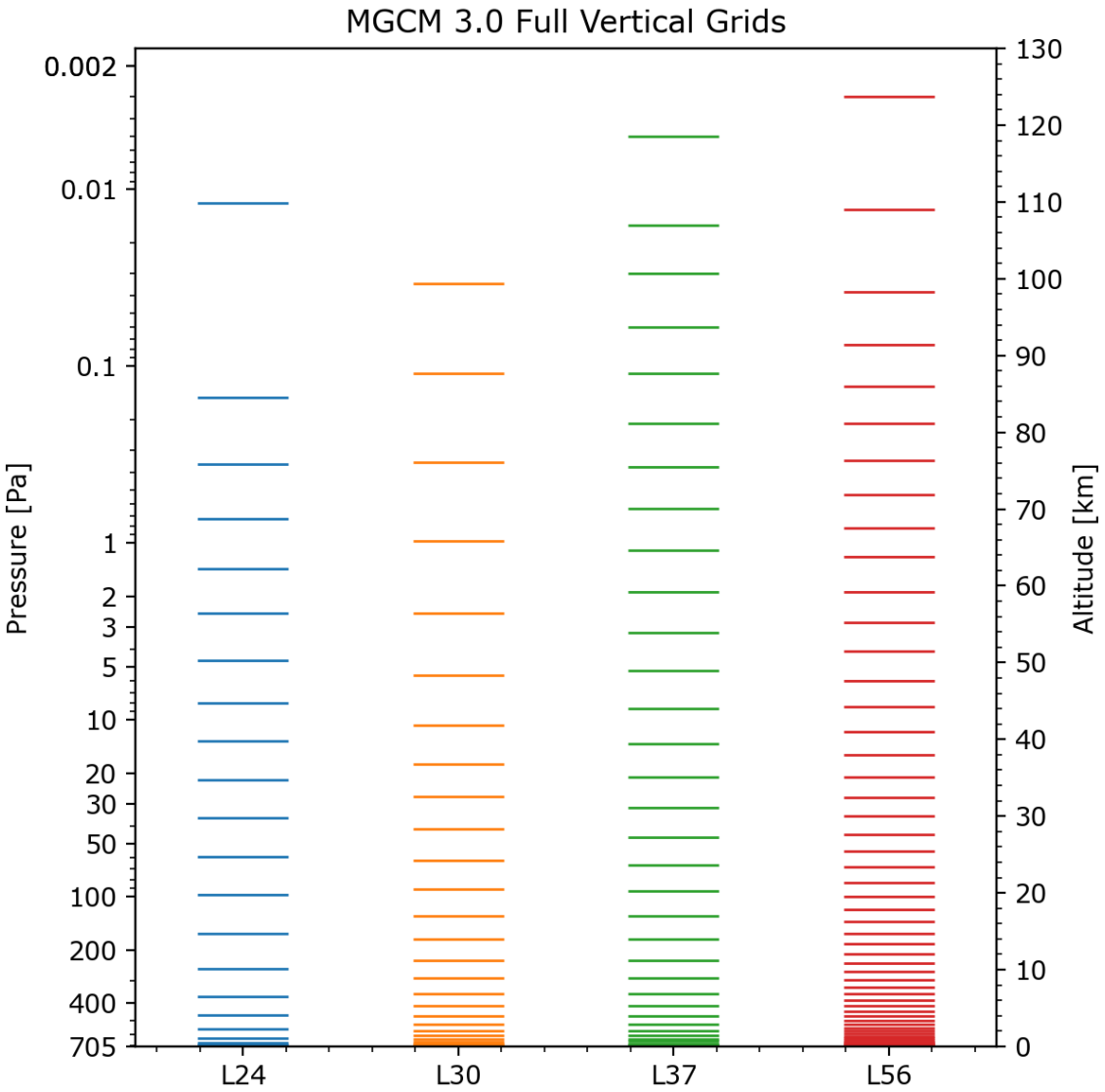

Figure 4.1: Midpoints of the vertical layers in the four vertical grids included in MGCM 3.0. From left to right, the uppermost layer midpoint is at 0.011 Pa (L24), 0.035 Pa (L30), 0.005 Pa (L37), and 0.003 Pa (L56).

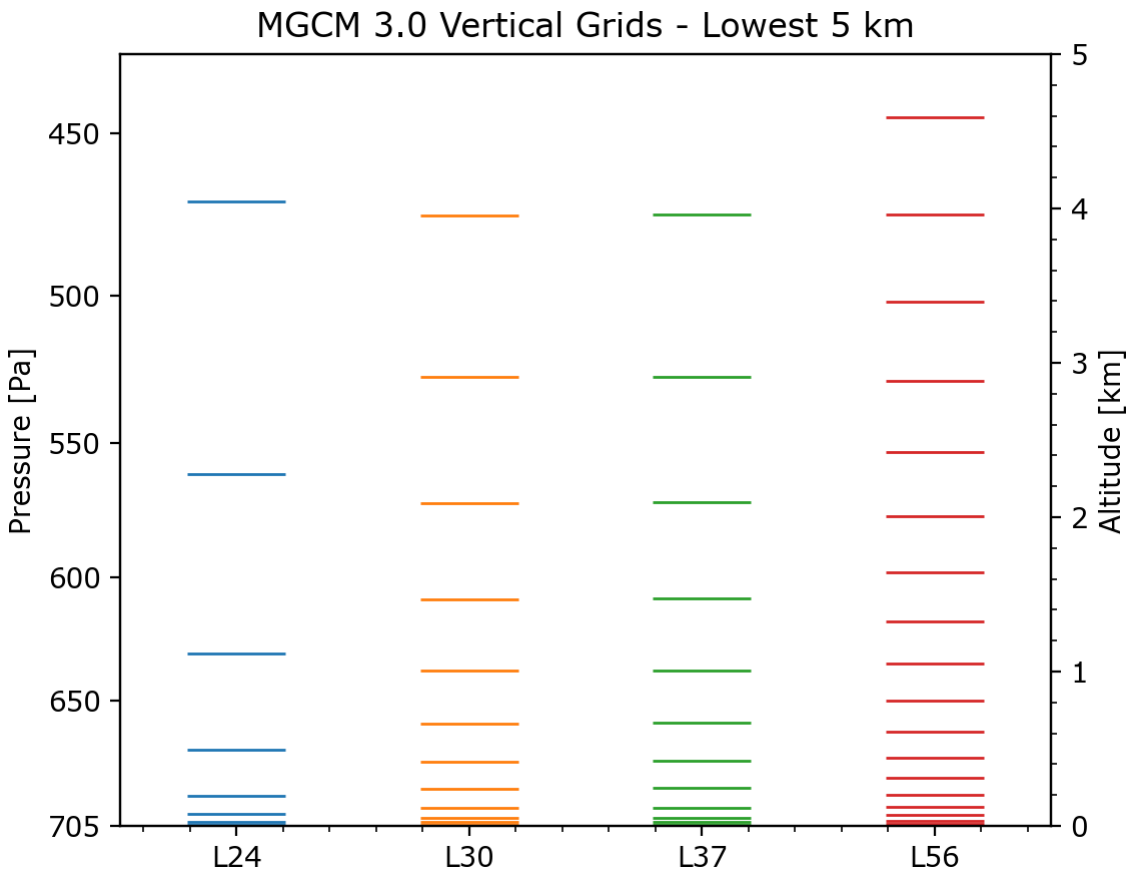

Figure 4.2: Midpoints of the vertical layers located below 5 km in the four vertical grids included in MGCM 3.0.

4.3.3 Topographical Options

By default, MGCM 3.0 is initialized with a topographic map from MOLA with 1/16° (∼3.75 km) resolution. The model can be run with a smooth surface (i.e. a "billiard ball" run) simply by changing `test_case` from "11" to "10" in the `&test_case_nml` namelist in the runscript:

```
&test_case_nml
    test_case    = 11   ! Topography type. 11= Mars; 10= flat
```

WARNING: Only `test_case=10` and `test_case=11` are supported in MGCM 3.0.

4.3.4 Options for the Dust Prescription

There are a number of customization options for the dust prescription in MGCM 3.0 in addition to the dust scenarios introduced in Chapter 4 (Section 4.2.4). This section lists a few dust prescription set-ups that are commonly used in the model.

The default dust setup references the background dust scenario file, which informs the column dust opacity. The default atmospheric dust distribution has a modified Conrath vertical profile defined in the `&aerosol_nml` namelist:

```
!****************************************
! ========= Aerosols Namelists =========
!****************************************
&aerosol_util_nml
/
&aerosol_nml
    do_inpt_dust_cycle = T,  ! Read the dust scenario from
                             ! an input file. Skipped if
                             ! optical_depth_inpt <= 0
    conrath_type   = 1,      ! Conrath vertical extent.
/
```

To increase the effective radius of the dust, add the variable `reff_fixed` to the namelist `&aerosol_util_nml` as in the example below and set its value. For example, setting a 2.5 μm effective radius would require:

```
!****************************************
! ========= Aerosols Namelists =========
!****************************************
&aerosol_util_nml
    reff_fixed     = 2.5,    ! Effective radius of the
                             ! dust for RT code
/
&aerosol_nml
    do_inpt_dust_cycle = T,  ! Read the dust scenario from
                             ! an input file. Skipped if
                             ! optical_depth_inpt <= 0
    conrath_type   = 1,      ! Conrath vertical extent.
/
```

To use a constant Conrath vertical profile instead of a modified Conrath parameter (as in the default setup), set `conrath_type` to zero, add the Conrath parameter (`conrath`) to `&aerosol_nml`, and set it to the desired value (0.003 is the default):

```
!****************************************
! ========= Aerosols Namelists =========
!****************************************
&aerosol_util_nml
/
&aerosol_nml
    do_inpt_dust_cycle = T,  ! Read the dust scenario from
                             ! an input file. Skipped if
                             ! optical_depth_inpt <= 0
    conrath_type  = 0,       ! Conrath vertical extent.
    conrath = 0.003,         ! Value of the Conrath parameter
/
```

To use a constant tau for the dust, toggle off `do_inpt_dust_cycle`, which informs the model to skip reading in the dust scenario file, add `optical_depth_inpt` to the namelist `&aerosol_nml` and set it to the desired dust opacity (5 in this example). The vertical profile of the dust is set by `conrath_type`, which must be 0 for a constant Conrath profile (of the user's choosing) when using constant dust tau.

```
!****************************************
! ========= Aerosols Namelists =========
!****************************************
&aerosol_util_nml
/
&aerosol_nml
    do_inpt_dust_cycle = F,  ! Read the dust scenario from
                             ! an input file. Skipped if
                             ! optical_depth_inpt <= 0
    conrath_type  = 0,       ! Conrath vertical extent.
    conrath = 0.003,         ! Value of the Conrath parameter
    optical_depth_inpt = 5   ! tau for the dust
/
```

WARNING: To return to the default configuration, be sure to reset `do_inpt_dust_cycle = T` and set `optical_depth_inpt = 0` (or delete the line altogether). If you set `do_inpt_dust_cycle = T` but `optical_depth_inpt` is nonzero, the MGCM will read in the dust scenario file **AND** add a constant tau on top of it.

Chapter 5

Model Output and Post-Processing

The MGCM outputs netCDF files that are stored in a compressed (`.tar`) format. NetCDF is an interface for storing and accessing geophysical data and it is coupled with a library that enables interaction with the interface. The netCDF library defines a machine-independent format for representing scientific data so that, together, the interface, library, and format support the creation, access, and sharing of scientific data. NetCDF was developed at the Unidata Program Center in Boulder, Colorado. Free resources for netCDF are available on the Unidata website at http://www.unidata.ucar.edu/software/netCDF.

> **NOTE:** netCDF files are self-descriptive. Users can look at the stored data directly from the command line using `ncdump`, which is a netCDF function that writes an ASCII representation of file contents to the screen. Documentation for `ncdump` is available at http://www.bic.mni.mcgill.ca/users/sean/Docs/netCDF/guide.txn_79.html. The MCMC-developed CAP also has a function that can be used to inspect netCDF files (see Chapter 8).

On NAS, the MGCM outputs data to both Pleiades and the Lou mass storage system (for long-term data storage). The main output directory is on Pleiades at:

```
/nobackup/$USER/path/to/FMS_MARS_runs/am4_mars_runs/
 ↪  $runscript_name/
```

This directory includes output files ("history" files), restart files, log files (useful for diagnosing model issues), and input files. The runscript is also copied here for archiving, along with the `diag_table`, `field_table`, and model namelist (`input.nml`). An additional backup of these files is copied over to Lou and stored in the `history`, `restart`, and `ascii` directories under:

```
FV3/xanadu/am4_mars_runs/$runscript_name/
```

The `history/` directory contains files output at the end of each run iteration (recall that

the run iteration is specified by `DAYSLIST` in the runscript). The tarred file names begin with a 5-digit date indicating the first sol number of the data in the file. For example, the default run defines `dayslist = (668 688)`, so output is separated into two files: the first begins on sol 0 and ends on sol 668 (`00000.nc.tar`), and the other begins on sol 668 and ends on sol 01334 (`00668.nc.tar`).

MGCM 3.0 data is archived on the native grid and stored in the file corresponding to the tile the data came from. The location of the tiles comprising the cubed-sphere grid is shown in Figure 5.1. Tile 3 is centered over the north pole, Tile 6 over the south pole, and Tiles 1, 2, 4, and 5 wrap around the globe centered at the equator. For the default diag table settings, the model will output 30 files each run iteration: six output files (one for every tile) for each of the five netCDF output file types defined in the diag table (`atmos_average`, `atmos_daily`, `atmos_diurn`, `atmos_fixed`, `grid_spec`). The model will also output restart files as described in Chapter 2.

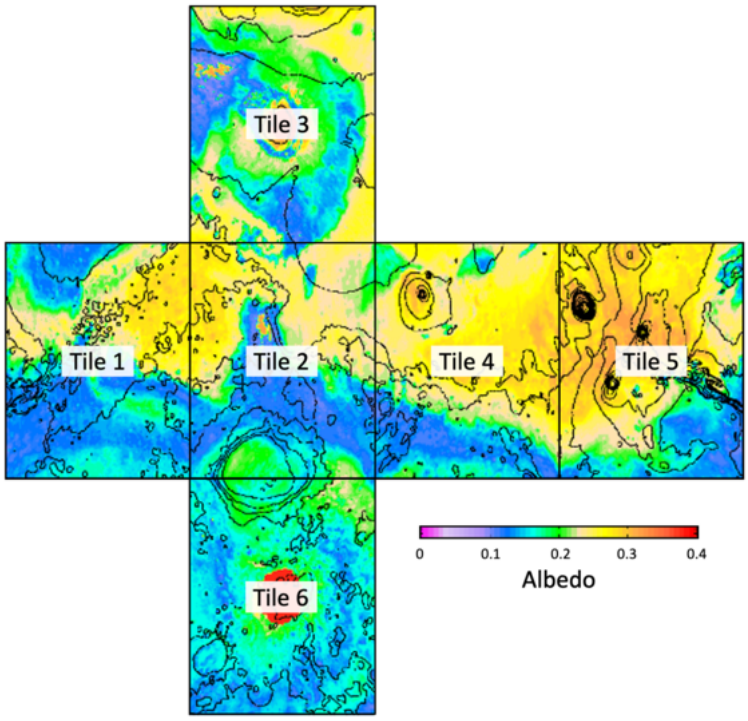

Figure 5.1: The tiles comprising the cubed-sphere grid in MGCM 3.0. Surface albedo is color-filled.

The 30 output files that are created each run iteration can be accessed by untarring one of the `XXXXX.nc.tar` files in the `history/` directory. Untarring a file reveals the following:

```
atmos_average.tile1.nc  atmos_daily.tile5.nc  fixed.tile3.nc
atmos_average.tile2.nc  atmos_daily.tile6.nc  fixed.tile4.nc
atmos_average.tile3.nc  atmos_diurn.tile1.nc  fixed.tile5.nc
atmos_average.tile4.nc  atmos_diurn.tile2.nc  fixed.tile6.nc
atmos_average.tile5.nc  atmos_diurn.tile3.nc  grid_spec.tile1.nc
```

```
atmos_average.tile6.nc  atmos_diurn.tile4.nc  grid_spec.tile2.nc
atmos_daily.tile1.nc    atmos_diurn.tile5.nc  grid_spec.tile3.nc
atmos_daily.tile2.nc    atmos_diurn.tile6.nc  grid_spec.tile4.nc
atmos_daily.tile3.nc    fixed.tile1.nc        grid_spec.tile5.nc
atmos_daily.tile4.nc    fixed.tile2.nc        grid_spec.tile6.nc
```

The variables in each of these files were defined in the diag table (Section 2.6) and are listed in Tables 5.1–5.5 below. The `grid_spec` file contains information about the latitude and longitude values for each boundary on a tile (Table 5.1). This file is used to re-grid the data to traditional latitude-longitude coordinates as described in the next section.

Table 5.1: Default Output Variables in `grid_spec`

Variable	Units	Description
`grid_lon`	degrees E	(2D) the boundary longitude of each grid cell on the tile
`grid_lat`	degrees N	(2D) the boundary latitude of each grid cell on the tile

5.1 Installing the Post-Processing Routines

Untarring and re-gridding data in the average, daily, and diurn files to global latitude-longitude coordinates can be done in one step using a modified version of NOAA-GFDL's `FRE-NCtools` toolkit. After the data is re-gridded, the files can be interpolated to a standard pressure vertical coordinate also using an MCMC-modified version of `FRE-NCtools`. In this section are instructions for downloading `FRE-NCtools` and applying MCMC-developed patches to the toolkit to modify the re-gridding and pressure-interpolating functions for compatibility with MGCM 3.0 output.

> **NOTE:** These instructions include snippets of code specific to the NAS System. However, these NAS-specific options usually refer to path names and can be modified for use on any system.

Begin by cloning `FRE-NCtools` from NOAA-GFDL's github to your preferred directory on Lou:

```
user@lfe:~$ git clone --recursive
 ↪  https://github.com/NOAA-GFDL/FRE-NCtools
```

Next, copy `srcFRE.patch` from the MGCM 3.0 source code on Pleiades to your home directory on Lou:

```
user@lfe:~$ scp USER@pfe:AM4/src/AmesGCM/patches/srcFRE.patch .
```

> **NOTE:** Change `USER` to your NAS username and modify the path to `AM4` as necessary. Refer to the NAS High-End Computing Capability website for help with moving files between Pleiades and Lou (https://www.nas.nasa.gov/hecc/support/kb/).

Next, load the following modules on Lou. These enable interfacing with GitHub and installing the NOAA-GFDL post-processing distribution, `FRE-NCtools`.

```
user@lfe:~$ module purge
user@lfe:~$ module load comp-intel/2020.4.304
user@lfe:~$ module load mpi-hpe/mpt.2.27
user@lfe:~$ module load hdf4/4.2.12
user@lfe:~$ module load hdf5/1.8.18_mpt
user@lfe:~$ module load netcdf/4.4.1.1_mpt
```

Confirm that the proper modules are loaded by typing `module list` on the command line. This should show:

```
Currently Loaded Modulefiles:
  1) comp-intel/2020.4.304    4) szip/2.1.1
  2) mpi-hpe/mpt.2.27         5) hdf5/1.8.18_mpt
  3) hdf4/4.2.12              6) netcdf/4.4.1.1_mpt
```

Set the paths to netCDF, HDF5, and the config file using the appropriate syntax for your shell. The config file will be copied over in the `nas/` subdirectory in the next step.

For `csh` or `tcsh` users:

```
user@lfe:~$ setenv netCDF_HOME /nasa/netCDF/4.4.1.1_mpt
user@lfe:~$ setenv HDF5_HOME /nasa/hdf5/1.8.18_mpt
user@lfe:~$ setenv CONFIG_SITE
 ↪  /u/USER/FRE-NCtools/site-configs/nas/config.site
```

For `bash` users:

```
user@lfe:~$ export netCDF_HOME=/nasa/netCDF/4.4.1.1_mpt
user@lfe:~$ export HDF5_HOME=/nasa/hdf5/1.8.18_mpt
user@lfe:~$ export
 ↪  CONFIG_SITE=/u/USER/FRE-NCtools/site-configs/nas/config.site
```

NOTE: Change `USER` to your NAS username and modify the path to `FRE-NCtools` as necessary.

Copy the `nas/` subdirectory containing the config file from the MGCM source code on Pleiades to `site-configs/`:

```
user@lfe:~$ scp -r USER@pfe:AM4/src/AmesGCM/diagnostics/nas
 ↪  FRE-NCtools/site-configs/.
```

WARNING: If you did not install `FRE-NCtools` in your home directory on Lou, then you have to change the path for `PREFIX` in the config file (`nas/config.site`) so that it points to your installation of `FRE-NCtools`.

Now go into `FRE-NCtools/` and apply the patch:

```
user@lfe:~$ cd FRE-NCtools
user@lfe:~$ git checkout -b mars_branch && git apply --reject
 ↪  --whitespace=fix ../srcFRE.patch
```

Reconfigure the toolkit:

```
user@lfe:~$ autoreconf -i
```

Now, build the toolkit. Create a `build/` directory in `FRE-NCtools`, disable the ocean model generator (there are no oceans on present-day Mars!), and make the installation with `make`:

```
user@lfe:~$ mkdir build && cd build
user@lfe:~$ ../configure --disable-ocean-model-grid-generator
user@lfe:~$ make && make install
```

Finally, copy the re-gridding and pressure-interpolation wrappers from the MGCM 3.0 source code on Pleiades to `FRE-NCtools/bin/`:

```
user@lfe:~$ scp
 ↪  USER@pfe:AM4/src/AmesGCM/diagnostics/runpinterp.csh
 ↪  /u/USER/FRE-NCtools/bin/.
user@lfe:~$ scp
 ↪  USER@pfe:AM4/src/AmesGCM/diagnostics/cinterp_script.csh
 ↪  /u/USER/FRE-NCtools/bin/.
```

WARNING: If you did not install `FRE-NCtools` in your home directory on Lou, then you have to modify three paths in the files you just copied over to reflect the directory `FRE-NCtools` is in:

- Modify the path to `pinterp.sh` in `runpinterp.csh`
- Modify the environment path (`setenv PATH`) in `runpinterp.csh`
- Modify `regridpath` path in `cinterp_script.csh`

Congratulations! You have installed the re-gridding and pressure-interpolation routines and modified them as appropriate for working with MGCM 3.0 output. The following two sections provide step-by-step instructions for regrinding and pressure-interpolating MGCM data.

5.2 Re-gridding Tiled Data

Begin by going into the `history/` directory for a simulation. For the default simulation (on NAS), the path is:

```
user@lfe:~$ cd
 ↪  FV3/xanadu/am4_mars_runs/fms_mars_default/history/
```

Note the resolution (`NCX`) of the data and the date code in the first part of the tarred file name. Pass these parameters to `cinterp_script.csh` as shown below. For the default simulation, `NCX=24` and the output files are `00000.nc.tar` and `00668.nc.tar`. To re-gridding output from the second year of the simulation, stay in the `history/` directory and do:

```
user@lfe:~$ ~/FRE-NCtools/bin/cinterp_script.csh -n c24 -d 00668
```

NOTE: Modify the path to `FRE-NCtools` as needed.

When `cinterp_script.csh` is finished, it will have re-gridded the average, daily, and diurn files from each of the tiles onto a latitude-longitude grid. The re-gridded files are:

- `00668.fixed.nc`
- `00668.atmos_daily.nc`
- `00668.atmos_average.nc`
- `00668.atmos_diurn.nc`

Recall that the variables defined in the diag table determine the contents of these files. The minimum variables output into each of the files are listed in Tables 5.2–5.5 below for your reference.

NOTE: Before you pressure-interpolate the data to a standard pressure grid as described here, the output files will reflect that the vertical coordinate is `pfull`. `pfull` is a time-invariant **reference** pressure grid that has the same number of layers as the model vertical grid. It is **not** the actual pressure at any specific location. It is a reference pressure grid based on a zero elevation pressure level of 705 Pa. The actual pressure at a given layer midpoint will vary depending on the local surface pressure and the thermal structure of the atmosphere.

Table 5.2: Default Output Variables in `fixed`

Variable	Units	Description
`bk`	none	the vertical coordinate sigma value
`ak`	Pa	the pressure part of hybrid coordinate
`lon`	degrees East	longitude
`phalf`	mb	the pressure of the layer interface
`lat`	degrees North	latitude
`grid_yt_bnds`	degrees North	the cell boundary latitude
`grid_xt_bnds`	degrees East	the cell boundary longitude
`zsurf`	m	surface height
`thin`	mks	surface thermal inertia
`alb`	none	surface albedo
`emis`	none	surface emissivity
`gice`	none	GRS ice

Table 5.3: Default Output Variables in `atmos_average`

Variable	Units	Description
bk	none	the vertical coordinate sigma value
ak	Pa	the pressure part of hybrid coordinate
pfull	mb	the reference pressure of the layer midpoint
time	days	number of sols since the start of the run
average_T1	days	start time for averaging period
average_T2	days	end time for averaging period
average_DT	days	length of averaging period
lon	degrees East	longitude
phalf	mb	the pressure of the layer interface
scalar_axis	none	the aggregating dimension (for netCDF)
lat	degrees North	latitude
grid_yt_bnds	degrees North	the cell boundary latitude
grid_xt_bnds	degrees East	the cell boundary longitude
time_bnds	days	time axis boundaries
areo	degrees	areocentric longitude
ps	Pa	surface pressure
ucomp	m s^{-1}	zonal wind
vcomp	m s^{-1}	meridional wind
temp	K	temperature
ukd	m s^{-1}	lowest-layer U velocity
vkd	m s^{-1}	lowest-layer V velocity
tkd	K	lowest-layer temperature
stress	N m^{-2}	surface wind stress
snow	kg m^{-2}	surface CO_2 ice
precip	kg m^{-2} dt^{-1}	amount of surface CO_2 ice originating from the atmosphere per time step
ts	K	surface temperature
t05	K	50 Pa temperature
taudust_VIS	op	visible dust opacity
dustref	optical depth per Pa	visible dust opacity per Pa

Table 5.4: Default Output Variables in `atmos_daily`

Variable	Units	Description
`bk`	none	the vertical coordinate sigma value
`ak`	Pa	the pressure part of hybrid coordinate
`areo`	degrees	areocentric longitude
`ps`	Pa	surface pressure
`ts`	K	surface temperature
`scalar_axis`	none	none
`lat`	degrees North	latitude
`lon`	degrees East	longitude
`phalf`	mb	the pressure of the layer interface
`time`	days	number of sols since the start of the run
`grid_yt_bnds`	degrees North	the cell boundary latitude
`grid_xt_bnds`	degrees East	the cell boundary longitude

Table 5.5: Default Output Variables in `atmos_diurn`

Variable	Units	Description
`bk`	none	vertical coordinate sigma value
`ak`	Pa	pressure part of hybrid coordinate
`areo`	degrees	areocentric longitude
`ps`	Pa	surface pressure
`ts`	K	surface temperature
`time`	days	number of sols since the start of the run
`average_T1`	days	start time for averaging period
`average_T2`	days	end time for averaging period
`average_DT`	days	length of averaging period
`time_of_day_24`	hours	hour of day
`scalar_axis`	none	none
`lon`	degrees East	longitude
`phalf`	mb	the pressure of the layer interface
`time_of_day_edges_24`	hours	time of day at the bin edges
`lat`	degrees North	latitude
`grid_yt_bnds`	degrees North	the cell boundary latitude
`time_bnds`	days	time axis boundaries
`grid_xt_bnds`	degrees East	the cell boundary longitude

5.3 Pressure-Interpolating Re-gridded Data

After untarring and re-gridding the output, `runpinterp.csh` can be used to pressure-interpolate each of the three `00668.atmos_*.nc` files onto a standard pressure grid. Note the names of the files to interpolate and call `runpinterp.csh` for each file as below:

```
user@lfe:~$ ~/FRE-NCtools/bin/runpinterp.csh -d 00668 -f
 ↪  atmos_average
user@lfe:~$ ~/FRE-NCtools/bin/runpinterp.csh -d 00668 -f
 ↪  atmos_daily
user@lfe:~$ ~/FRE-NCtools/bin/runpinterp.csh -d 00668 -f
 ↪  atmos_diurn
```

NOTE: Modify the path to `FRE-NCtools` as needed.

This creates three pressure-interpolated files ending in `*_pstd.nc` in the directory:

- `00668.atmos_daily_pstd.nc`
- `00668.atmos_average_pstd.nc`
- `00668.atmos_diurn_pstd.nc`

The interpolation script defaults to a 48-layer grid ranging from 10^3 Pa near the surface to 10^{-5} Pa near the top. The layers are listed in order from the top of the atmosphere to the layer nearest the surface below. Given in Pa, the standard pressure layers are:

```
1.0e-5  3.0e-5  5.0e-5  1.0e-4  3.0e-4  5.0e-4  3.0e-3  5.0e-3
1.0e-2  3.0e-2  5.0e-2  0.1     0.2     0.3     0.5     1
2       3       5       7       10      20      30      50
70      1.0e+2  1.5e+2  2.0e+2  2.5e+2  3.0e+2  3.5e+2  4.0e+2
4.5e+2  5.0e+2  5.3e+2  5.5e+2  5.9e+2  6.0e+2  6.3e+2  6.5e+2
6.9e+2  7.0e+2  7.5e+2  8.0e+2  8.5e+2  9.0e+2  9.5e+2  1.0e+3
```

The pressure-interpolated files contain all the variables in the diag table(s). The vertical dimension name for every 4D variable should reflect the change from the native grid (`pfull`) to the standard pressure grid (`pstd`).

Chapter 6

Default Simulation Description

The default MGCM runscript produces a two-year long simulation of the Martian atmosphere at approximately 240 km ($\sim 4°$) horizontal resolution. This chapter provides figures and descriptions showing what the MGCM-simulated atmosphere looks like after running `fms_mars_default`. If you run the default simulation as described in Part I, re-grid and pressure-interpolate your files as described in Chapter 5, and use the Ames Community Analysis Pipeline (CAP) to produce the default set of plots as described in Chapter 8, you can create the same figures and compare them to those shown here. A copy of these plots is also available with the model release in the source code:

```
AM4/src/AmesGCM/diagnostics/Model_Release_Diagnostics.pdf
```

The first four sets of multi-panel plots (Figures 6.1–6.4) show latitude versus pressure zonal mean cross-sections of atmospheric temperature (K; top left), zonal (east-west) wind (m s^{-1}; top right), meridional (north-south) wind (m s^{-1}; bottom left), and dust visible opacity (in 10^{-3} opacity per Pa; bottom right) for the four cardinal seasons: $L_s = 0°$, 90°, 180°, and 270°. The final set of plots (Figure 6.5) shows L_s (season) versus latitude of zonal mean surface temperature (K; top left), zonal mean surface CO_2 ice (kg m^{-2}; top right), and zonal mean column-integrated visible dust optical depth (normalized to 610 Pa; bottom left).

At $L_s = 0°$ (northern hemisphere spring equinox; Figure 6.1), the atmospheric thermal structure is nearly symmetric about the equator. In the low latitudes, zonal mean temperatures maximize near the surface in the tropics just over 220 K and decrease with altitude (decreasing pressure) to less than 130 K at approximately 0.5 Pa. Warm air ($\sim$160 K) extends upwards towards the poles in both hemispheres due to compressional heating from the nearly symmetric hadley cell circulation (not shown). Cold air ($\sim$130 K) resides at $\sim$10 Pa over both poles. The wind fields are consistent with this thermal structure. Westerly zonal winds (positive contours, top right panel, Figure 6.1) exist in both hemispheres, peaking at $\sim$100 m s^{-1} in the southern hemisphere and at slightly stronger than that in the northern hemisphere. Easterlies (negative contours, top right panel, Figure 6.1) exist in the tropics and subtropics, peaking at $\sim$125 m s^{-1} aloft. The meridional winds show a strong overturning circulation that is basically symmetric about the equator, with strong southerlies (northward flow) in the northern hemisphere ($\sim$30 m s^{-1}) at $\sim$0.1 Pa, strong northerlies (south-

ward flow) in the southern hemisphere (∼25 m s^{-1}) at 0.1 Pa, and weak opposite flow near the surface. The dust field shows the highest dust opacities in the low latitudes (∼ 0.8×10^{-3} per Pa).

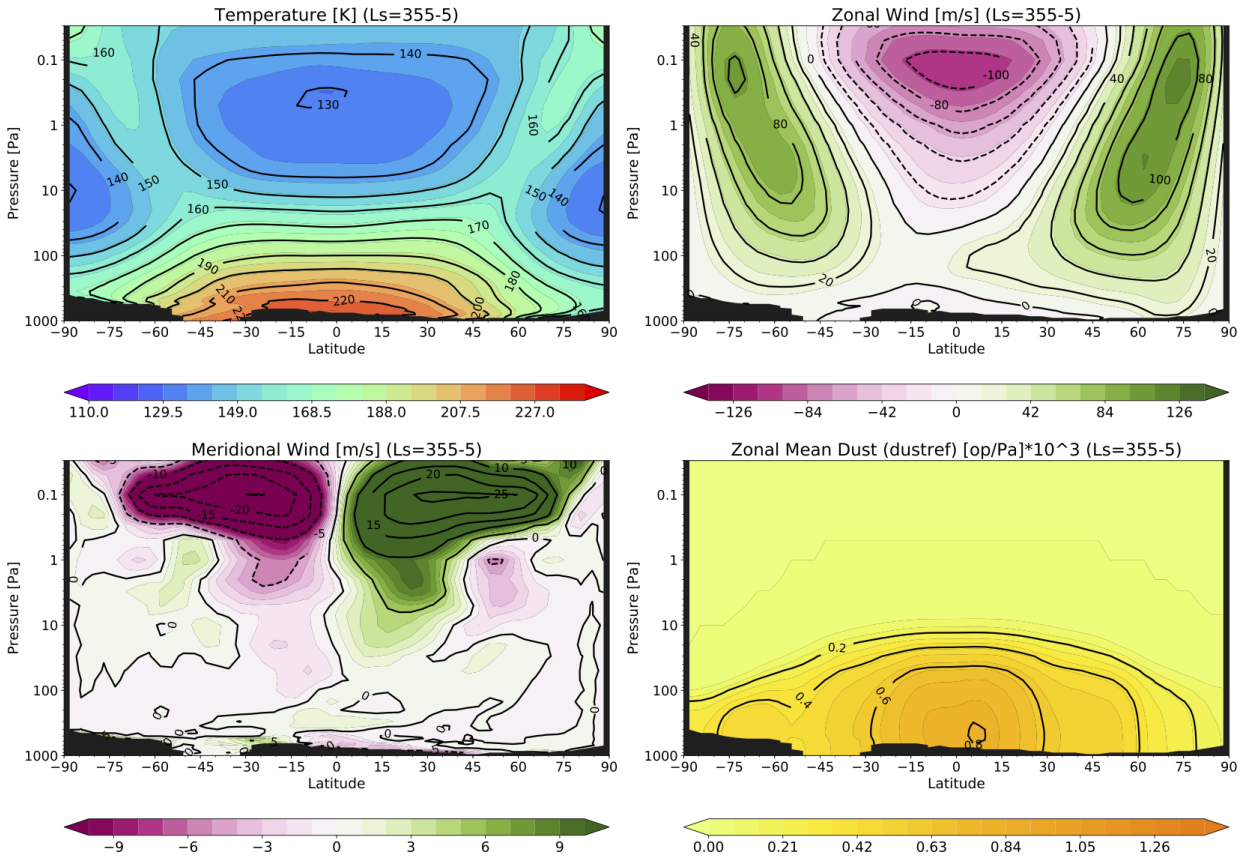

Figure 6.1: Zonal mean temperature (top left), zonal wind (top right), meridional wind (bottom left), and visible dust opacity (10^{-3} op per Pa, bottom right) at $L_s = 0°$ from the default simulation (`fms_mars_default`). Created with CAP (Chapter 8).

At $L_s = 90°$ (northern hemisphere summer solstice; Figure 6.2), peak temperatures reside in the northern hemisphere near the surface (∼230 K). The coolest temperatures are over the south (winter) pole at ∼1 Pa (∼120 K). Weak polar warming occurs due to the descending branch of the Hadley cell in the southern hemisphere. There is a layer of warm air at the top of the plot that peaks at ∼160 K in the high northern latitudes. The zonal wind field shows a westerly jet that peaks at ∼100 m s^{-1} in the southern hemisphere and easterlies throughout a large part of the northern hemisphere (peaking at ∼75 m s^{-1} at about 15° N). The meridional wind field shows multiple peaks in northerly (southward) flow: two at ∼20° S (∼10 Pa and ∼0.5 Pa, both ∼12 m s^{-1}), and one at the top of the plot at ∼25° N, peaking over 14 m s^{-1}). Southerly (northward) flow is evident very near the surface at the low latitudes. The dust field shows the highest dust opacities at the highest northern latitudes (∼ 0.6×10^{-3} per Pa).

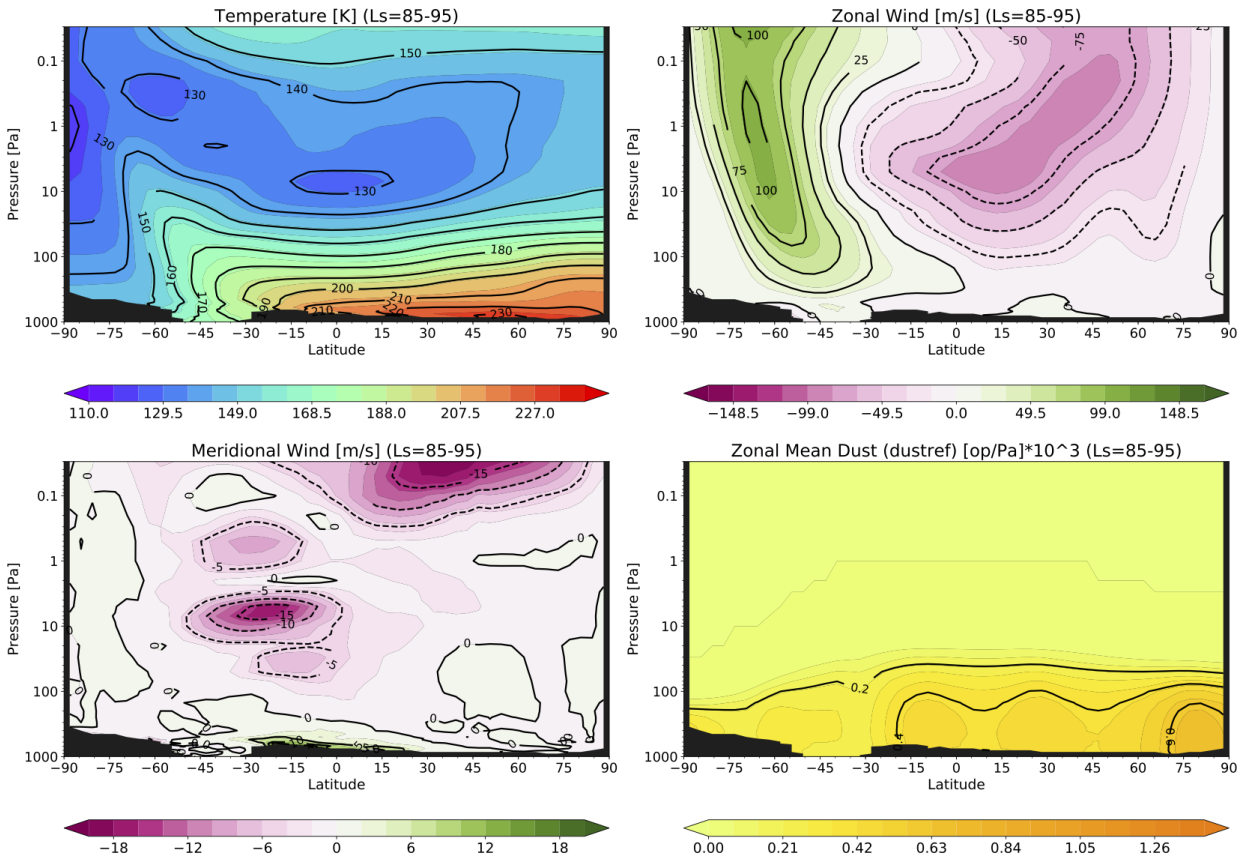

Figure 6.2: Zonal mean temperature (top left), zonal wind (top right), meridional wind (bottom left), and visible dust opacity (10^{-3} op per Pa, bottom right) at $L_s = 90°$ from the default simulation (`fms_mars_default`). Created with CAP (Chapter 8).

$L_s = 180°$ (northern hemisphere autumnal equinox; Figure 6.3), is similar in many ways to $L_s = 0°$ (Figure 6.1). In the low latitudes, zonal mean temperatures maximize near the surface in the tropics just over 230 K and decrease with altitude (decreasing pressure) to less than 130 K at approximately 0.5 Pa. Warm air ($\sim$170 K) extends upwards towards the poles in both hemispheres due to compressional heating of a nearly symmetric hadley cell circulation (not shown). Cold air ($\sim$130 K) resides at $\sim$10 Pa over both poles. Westerly zonal winds exist in both hemispheres, peaking at over 100 m s^{-1}. Easterlies (negative contours, top right panel, Figure 6.3) exist in the tropics and subtropics, peaking at $\sim$125 m s^{-1} aloft. The meridional winds show a strong overturning circulation that is basically symmetric about the equator, with strong southerlies (northward flow) in the northern hemisphere, and strong northerlies (southward flow) in the southern hemisphere (peaking at $\sim$25 m s^{-1} at 0.1 Pa), and weak opposite flow near the surface. The dust field shows the highest dust opacities in the southern middle latitudes ($\sim 1.4 \times 10^{-3}$ per Pa).

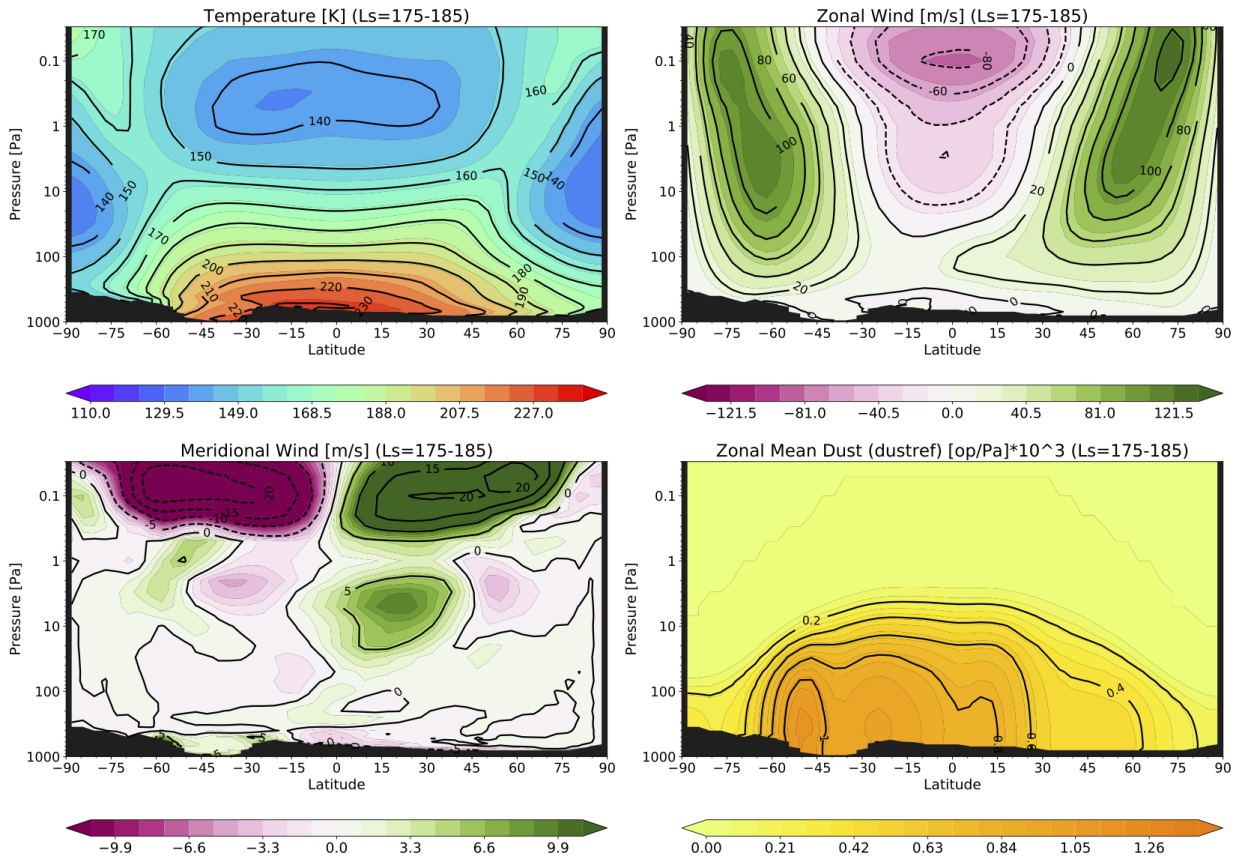

Figure 6.3: Zonal mean temperature (top left), zonal wind (top right), meridional wind (bottom left), and visible dust opacity (10^{-3} op per Pa, bottom right) at $L_s = 180°$ from the default simulation (`fms_mars_default`). Created with CAP (Chapter 8).

At $L_s = 270°$ (northern hemisphere winter solstice; Figure 6.4), peak temperatures reside in the southern hemisphere near the surface (∼240 K). The coolest temperatures are over the south (winter) pole at ∼100 Pa (∼140 K) and in the low latitudes aloft (less than 135 K at ∼2 Pa). Strong polar warming occurs due to the descending branch of the Hadley cell in the northern hemisphere. The zonal wind field shows a westerly jet that peaks at ∼100 m s^{-1} in the northern hemisphere and easterlies throughout a large part of the southern hemisphere (peaking at ∼150 m s^{-1} at about 15 S). The meridional wind field shows strong (∼40 m s^{-1}) southerly (northward) flow ∼2 Pa. Northerly (southward) flow is evident very near the surface at the low latitudes. The dust field shows the highest dust opacities at the highest southern latitudes ($\sim 1.4 \times 10^{-3}$ per Pa) and throughout the tropics and subtropics ($\sim 1.2 \times 10^{-3}$ per Pa).

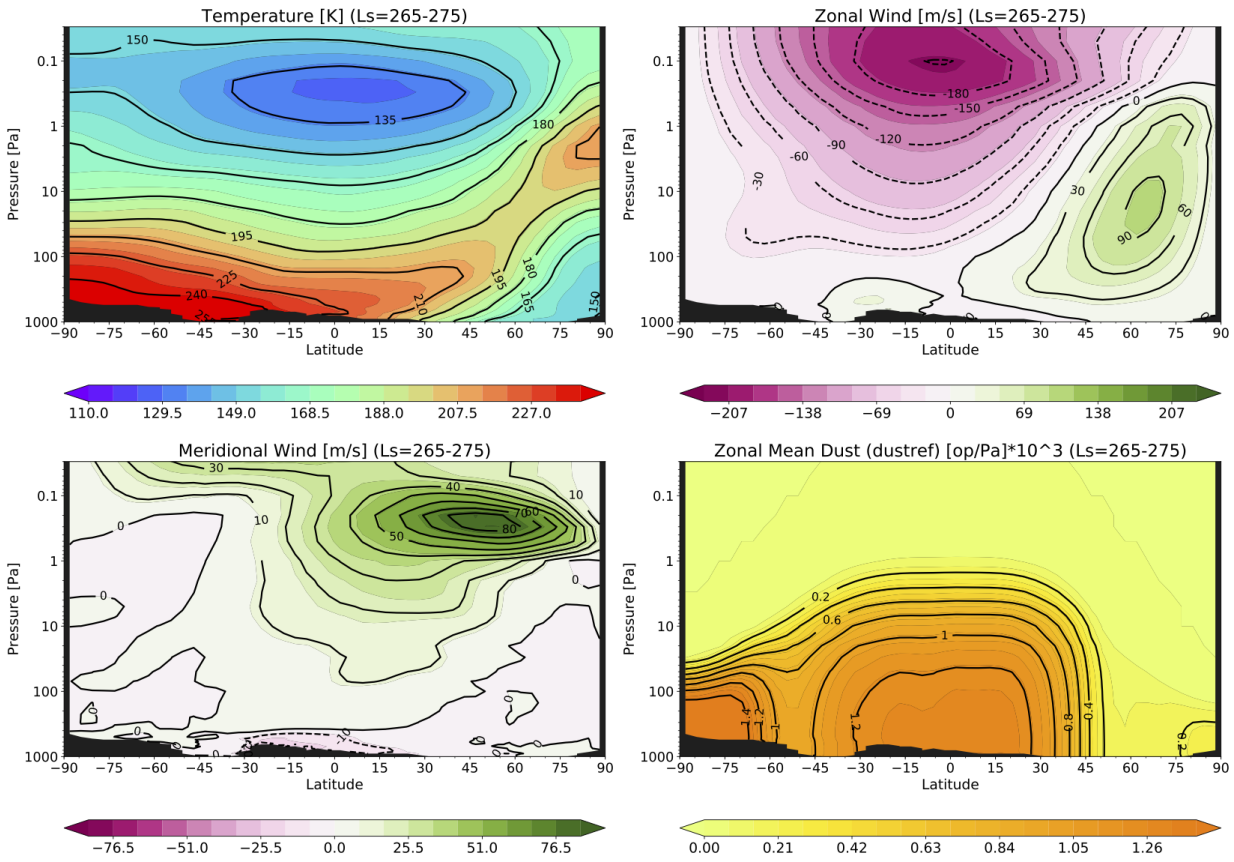

Figure 6.4: Zonal mean temperature (top left), zonal wind (top right), meridional wind (bottom left), and visible dust opacity (10^{-3} op per Pa, bottom right) at $L_s = 270°$ from the default simulation (`fms_mars_default`). Created with CAP (Chapter 8).

Finally, the annual surface/column-integrated plots in Figure 6.5 show the full annual cycle. Zonal mean surface temperatures maximize during local summer ($\sim$230 K in the north during northern summer and $\sim$250 K in the south during southern summer) and minimize during local winter ($\sim$150 K, which is controlled by the CO_2 saturation temperature). The low latitudes show a less pronounced seasonal cycle. The surface CO_2 ice field shows that CO_2 ice condenses during local autumn and sublimates during local spring, with maximum values of $\sim$600 kg m^{-2} and more than 800 kg m^{-2} in the north and south, respectively. The zonal mean visible dust optical depth field shows that the model is forced with a background dust optical depth of $\sim$0.2–0.3 during the first half of the year, with increased dust loading during the second half of the year. The pre- and post-solstitial increases in dust loading peak in the low latitudes at $\sim$0.9 and $\sim$0.7, respectively.

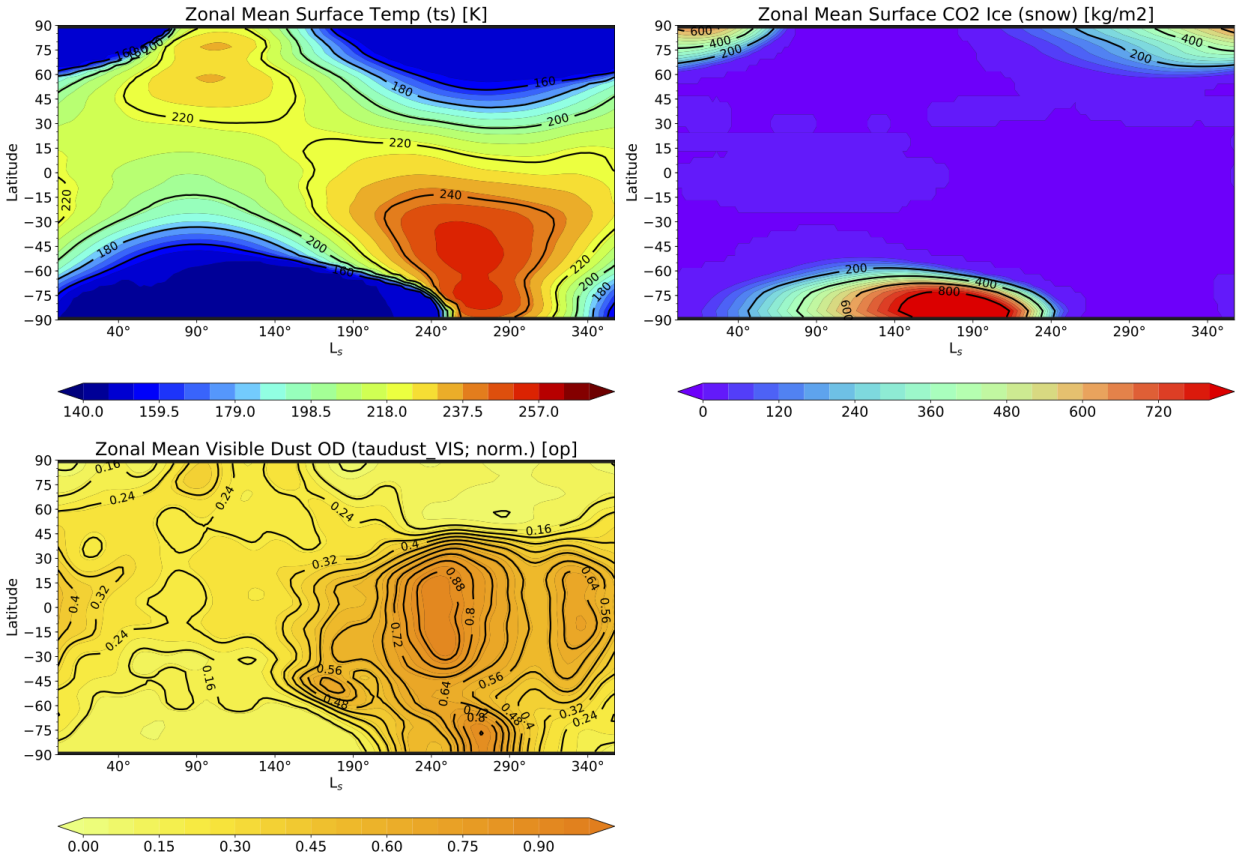

Figure 6.5: Zonal mean surface temperature (top left), surface CO_2 ice (top right), and visible dust optical depth normalized to 610 Pa (bottom left) for the second MY in the default simulation (`fms_mars_default`). Created with CAP (Chapter 8).

Chapter 7

Troubleshooting

In this chapter, we address some of the common issues that arise when running MGCM 3.0, including compilation problems, runtime errors, and model stability issues. This is *not* intended to be an exhaustive list of potential problems or solutions, but it *is* intended to be a resource for users to reference when the model is not operating as expected.

7.0.1 Compilation Errors

When the model appears to crash before even starting to run, it is possible that the model was not compiled before the runscript was submitted for execution. To confirm that this is the problem, check the `fms_mars_out.*` file in the `workdir/`, which is on Pleiades at:

```
/nobackup/$USER/FMS_MARS_runs/am4_mars_runs/$RUNSCRIPT_NAME/
```

If failing to compile the model is the issue, then the contents of `fms_mars_out.*` will appear as follows:

```
./FMS_MARS.x: No such file or directory
MPT ERROR: could not run executable.
```

There is a simple fix for this: run the compile script before re-submitting the runscript for execution. For instructions on compilation, see Chapter 4.

7.0.2 Runscript Errors

When running MGCM 3.0 on NAS, a common reason that a simulation might fail to run all the way through the requested `dayslist` is because the model ran out of walltime. To diagnose this problem, open the `*.o*` file in the `AM4/exec/` directory. The `*.o*` file is named after the runscript, so for the default simulation you will look for a file like `fms_mars_default.o*` in `AM4/exec/`. The requested walltime is listed **on line 5** and the amount of walltime used is listed at the end under `Job Resource Usage Summary`. If the requested walltime is less than the walltime used, then the simulation did not complete before running out of walltime.

If some history files were created during the simulation, warm start the simulation and adjust `dayslist` to reflect the length of the simulation remaining. You can simply perform a "continuation" warm start by setting `RUNTYPE = 1` in the runscript and the simulation will pick up from the last generated restart file. You may also warm start the model from a specific restart file by setting `RUNTYPE = 2` and specifying the path to and name of the restart file under `restartfile`. Restart files for a simulation can be found in the `restart/` subdirectory:

```
/nobackup/$USER/FMS_MARS_runs/am4_mars_runs/
 ↪  $RUNSCRIPT_NAME/restart/
```

Other errors relating to the runscript setup include incorrect file paths to the input or output directories and incorrect namelist variables (typos). Here is a list of items to check:

1. Double check for typos in the namelist variables in your runscript by comparing them to the variables listed in the default runscript `fms_mars_default`.
2. Confirm that the options selected for each namelist item are of the correct type (boolean, float, integer).
3. Confirm that the various directories defined in the runscript exist **and contain the relevant files**.
4. Ensure that all namelists have the proper syntax:
   (a) Commas appear after all but the last variable in a namelist.
   (b) A backslash appears at the end of every namelist.
   (c) All namelist names begin with an ampersand (&)

If the model cannot find an input file because either the filepath is incorrect or the file does not exist, an error will show in `fms_mars_out.*` in the `workdir` (directory path provided above). For example, if the model cannot locate the topography input file, the following fatal error appears in `fms_mars_out.*`:

```
FATAL from PE 4: surfdrv: mars_topo not found in INPUT
```

If the model cannot find or does not recognize a namelist variable, the following fatal error appears in `fms_mars_out.*`:

```
FATAL from PE   35: check_nml_error in fms_mod:  Unknown
                    namelist, or mistyped namelist variable
                    in namelist aerosol_util_nml, (IOSTAT = 19)
```

This error actually points to the namelist throwing the error, in this case, `aerosol_util_nml`. The error is often caused by one of three things:

1. A variable in the indicated namelist (`aerosol_util_nml`, in this case) is misspelled
2. The requested variable does not exist

3. The variable does exist but is listed under the wrong namelist in the runscript. The namelist listed in the error is the inappropriate namelist for the variable (`aerosol_util_nml` in the example above).

7.0.3 Model Execution Errors

Most other errors likely stem from the model execution. Model instabilities, violating the CFL condition, and calling physics modules that do not exist or are not supported in the current model release are examples of problems that can arise during model execution. While there should be no instability issues when running the model with the default runscript, if certain settings are changed, the user might inadvertently push the model into a regime where the model is unstable or CFL conditions are violated. Here we provide some solutions.

Model Stability Issues

To diagnose errors relating to model stability, check the `fms_mars_out.*` file in the `workdir` (again, listed above). Check for anomalies in the values of variables listed below:

```
Total surface pressure (Pa)     UA_top
UA                              VA
TA                              OM
dummy_tracer                    h2o_vapor
tracer1                         tracer2
tracer3                         dst_mass_micro
dst_num_micro                   ice_mass_micro
ice_num_micro                   cor_mass_micro
vap_mass_micro                  ice_cloud
Sol                             sec
ZS                              PS
```

If the vertical or horizontal resolution is increased without also increasing the model time step, the CFL condition may be violated. If this is the case, errors will typically manifest as unstable temperatures and winds that often become `NAN` just before crashing. Again, check for anomalies in the values of variables listed above, especially in the temperature and wind fields. There are two solutions to this problem:

Solution 1: Increase `n_split` and (possibly) `k_split` (by defining `NNS` and `NKS`).

Solution 2: Decrease the physics time step `dt_atmos` (by defining `DTA`).

Instructions for modifying these parameters are in Chapter 4 Section 4.3. There is no single correct answer to solving problems relating to time stepping. However, there are some implementation best-practices that we share here.

In general, we recommended attempting **Solution 1** and then, if that does not do the trick, **Solution 2**. Beyond that, how you implement the above solutions depends primarily on the resolution of your simulation. For the default horizontal (`c24`) and vertical (56-layer grid) resolution, try

increasing only `n_split` first. Increasing `k_split` is generally most useful in cases when the vertical grid has more than 56 layers.

> **NOTE:** Increasing `k_split` is a diffusive process, and it is recommended that this number be as low as possible. There is no theoretical limit to the number of horizontal advection calculations (`n_split`) the model can do, so users should feel free to increase this value until stability is achieved.

Reducing the physics time step `dt_atmos` (**Solution 2**) is only recommended in cases where diabatic heating tendencies are expected to significantly change on timescales shorter than the default setting. Sometimes, running with too large of a time step results in a segmentation fault, which appears in the `fms_mars_out.*` file like so:

```
MPT ERROR: Rank 70(g:70) received signal SIGSEGV(11).
```

When decreasing the time step, best practice is to start by cutting the time step in half. Recall that the time step must evenly multiply into the length of the day. This is why the decision was made to use 88,704 seconds to define the solar day in MGCM 3.0: it allows for a greater number of time step options.

Calling Inactive Modules

The model will exit if flags are toggled to activate physics modules that have not been released. In this case, the model will communicate which physics module it attempted and failed to activate in the `fms_mars_out.*` file. For example, calling the microphysics module in MGCM 3.0 throws the following error:

```
FATAL from PE    56: dust_update_init: The null version of
                     dust_update_init should not be called.
```

BUS Error

When warm-starting the model on the NAS system, occasionally a "BUS" error will arise. This will appear in the `fms_mars_out.*` file like so:

```
MPT ERROR: Rank 23(g:23) received signal SIGBUS(7).
```

This can occasionally indicate that NAS had some internal issue unrelated to the runscript or the model execution, and the model exited as a result. As in the procedure for Model Stability Issues above, check the `fms_mars_out.*` file in the `workdir` for anomalies in the values of variables listed in that section. If no anomalies can be found, which would point to some other issue, it may be the rare case of a NAS issue. In our experience, simply re-submitting the same runscript again with no changes solves the problem.

7.0.4 Useful Metrics

When diagnosing issues with the model, it is often useful to know what season the simulated atmosphere was in when the model exited. This can be roughly determined by the Sol number when the model exited. This can be found in the last entry of `Sol = XXX` in the `fms_mars_out.*` file. Table 7.1 below shows the approximate sol number and corresponding L_s for a three year simulation.

Table 7.1: Approximate Sol number and corresponding L_s for a 3-year simulation.

L_s	Sol YR 1	Sol YR 2 (+668)	Sol YR 3 (+1336)
0°	0	668	1336
15°	29	697	1365
30°	60	728	1396
45°	93	761	1429
60°	125	793	1461
75°	159	827	1495
90°	192	860	1528
105°	225	893	1561
120°	257	925	1593
135°	287	955	1623
150°	317	985	1653
165°	345	1013	1681
180°	371	1039	1707
195°	397	1065	1733
210°	421	1089	1757
225°	445	1113	1781
240°	468	1136	1804
255°	491	1159	1827
270°	514	1182	1850
285°	538	1206	1874
300°	562	1230	1898
315°	586	1254	1922
330°	612	1280	1948
345°	639	1307	1975

NOTE: Sol can be larger than 668 because it continuously counts the number of Mars days from the beginning of the simulation.

Chapter 8

The Community Analysis Pipeline (CAP)

CAP is a user-friendly, command line data analysis and visualization tool designed to interface with output from the NASA Ames MGCM. At the time of writing, both the NASA Ames Legacy MGCM and MGCM 3.0 are supported in CAP. The purpose of this chapter is to provide an overview of some of the most useful CAP utilities for interacting with MGCM 3.0 output. A full description of CAP, including detailed installation instructions, a user tutorial, and definitions for every executable and function are available on GitHub at https://github.com/NASA-Planetary-Science/AmesCAP.

CAP is Python-based software that standardizes the post-processing effort by providing executables that perform file manipulations and create diagnostic plots from the command line. CAP enables users of almost any skill level to process and plot MGCM data. CAP is designed to be modular so that users are free to selectively integrate CAP's functions into their own analysis routines to the extent they see fit. 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. Brief descriptions of each of the CAP executables are listed below.

- `MarsFiles` provides tools for file manipulations, including code designed to create binned, averaged, and time-shifted files from MGCM output, perform tidal analysis that decompose variables into tidal harmonics, and filter data using a low or high band-pass filter.
- `MarsVars` provides tools relating to variable operations such as adding and removing variables, and performing column integrations. Variable operations include (but are not limited to) density, potential temperature, and mass stream function, to name a few. For more variable options, refer to the tutorial link provided above.
- `MarsInterp` performs the vertical interpolation from reference (`pfull`) layers to standard pressure (`pstd`), altitude (`zstd`), and altitude above the ground (`zagl`) layers. `MarsInterp` is flexible and can perform interpolations to any user-provided vertical grid.
- `MarsPlot` is a plotting routine that accepts a modifiable template (`Custom.in`) containing a list of plots to create. It is useful for creating plots from MGCM output quickly and is designed specifically for use with the MGCM output files.

The plotting routine, `MarsPlot`, has several useful functions including averaging (e.g.,

zonal, global values), data reduction (e.g., specific altitude or latitude range), comparison between simulations (difference plots), simple variable operations (such as scaling), and data selection across multiple files. The `Custom.in` file provides templates for all of the typical meteorological 1D and 2D plots (e.g., time series, longitude/latitude cross-sections, and vertical profiles). It additionally provides some degree of flexibility, such as adjusting axes, color scales, or building multi-panel plots, but it is not intended to be fully customizable.

8.1 Plotting with CAP

CAP was used to create the figures from the default simulation shown in Chapter 6. Here we provide instructions specifically for using CAP to re-create those figures using output from your own simulations. These instructions assume CAP is installed on the system hosting MGCM 3.0 output files. See the installation instructions on GitHub if necessary.

The general process for plotting with `MarsPlot` is illustrated in Figure 8.1 and can be summarized as follows:

1. Open and modify `Custom.in` using a text editor
2. Pass `Custom.in` to `MarsPlot` to generate plots
3. View the plots in `Diagnostic.pdf`

When defining new plots in `Custom.in`, we recommend copy-pasting (or deleting) the example code blocks provided in the template. Be sure set any copied code blocks to `TRUE` so that `MarsPlot` knows to include the plot in the final PDF. If the name of the `Custom.in` file is not changed, the resulting PDF will be called `Diagnostic.pdf`. If the name of `Custom.in` is changed, then the name of the PDF will match the renamed `Custom.in` file. For example, renaming the template to `MyPlots.in` creates `MyPlots.pdf`.

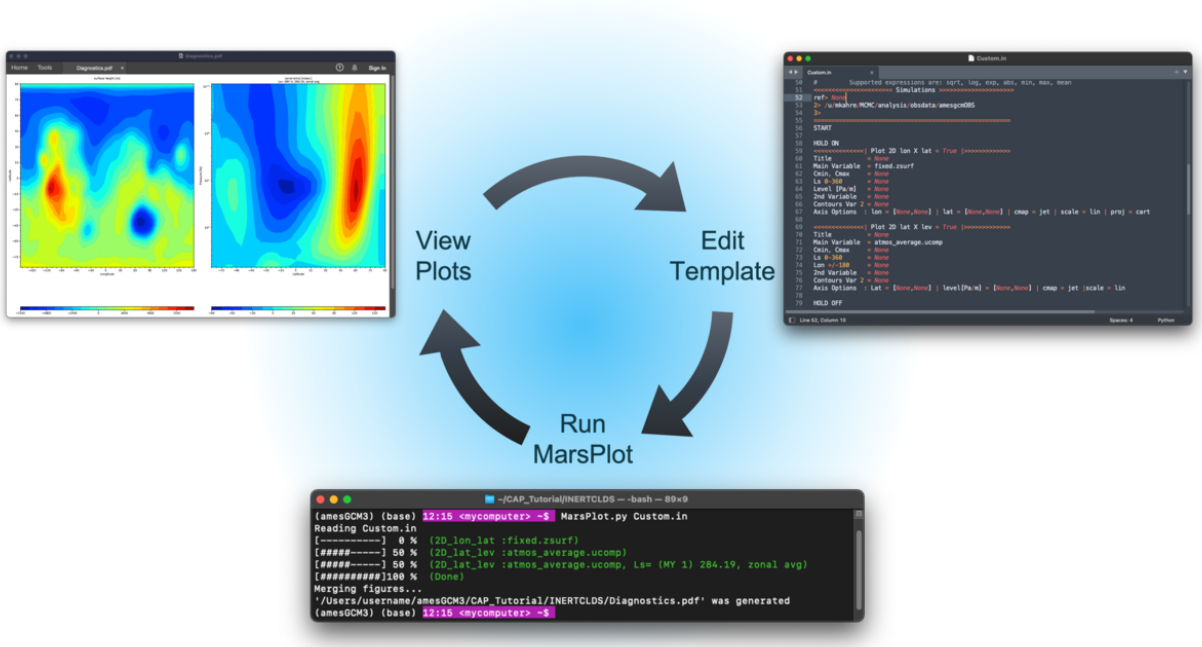

Figure 8.1: The `MarsPlot` cycle. Edit Template: Desired plots are defined in `Custom.in`; Run `MarsPlot`: `Custom.in` is passed into `MarsPlot` via the command line; View Plots: Plots are created in `Diagnostics.pdf`.

The plots shown in Chapter 6 were created using `Model_Release_Diagnostics.in`, which is included in the MGCM source code at:

```
AM4/src/AmesGCM/diagnostics/Model_Release_Diagnostics.in
```

To use `Model_Release_Diagnostics.in`, make a copy of the file and move it to your output directory:

```
user@lfe:~$ scp -r USER@pfe:AM4/src/AmesGCM/diagnostics/
 ↪  Model_Release_Diagnostics.in .
```

> **NOTE:** The secure copy command above contains NAS-specific syntax. Modify as necessary for non-NAS environments.

To make the plots in `Model_Release_Diagnostics.in` on NAS, you have to first install CAP. The NAS-specific installation is explained on the CAP GitHub at https://github.com/NASA-Planetary-Science/AmesCAP/blob/master/docs/NAS_INSTRUCTIONS.txt.

Once you have installed CAP, initiate the CAP virtual environment. The syntax for this could vary slightly depending on your package manager (`pip` or `conda`) and terminal shell (e.g. `bash`, `csh`, `tcsh`), but is typically performed with:

```
source ~/AmesCAP/bin/activate.csh
```

Then, pass `Model_Release_Diagnostics.in` to `MarsPlot` to create the figures:

```
MarsPlot.py Model_Release_Diagnostics.in
```

All of the figures will appear in a single PDF called `Model_Release_Diagnostics.pdf` in your current directory. You can use `Model_Release_Diagnostics.in` to verify that the default MGCM simulation performs as expected by comparing your plots to the ones provided in this User Guide. You may also find it useful to use `Model_Release_Diagnostics.in` for other simulations.

References

Bertrand, T., Wilson, R. J., Kahre, M. A., Urata, R., & Kling, A. (2020). Simulation of the 2018 global dust storm on Mars using the NASA Ames Mars GCM: A multi-tracer approach. *Journal of Geophysical Research*, *125*(7). doi: 10.1029/2019JE006122

Boynton, W. V., Feldman, W. C., Squyres, S. W., Prettyman, T. T., Brückner, J., Evans, L. G. ., & Shinohara, C. (2002). Distribution of hydrogen in the near surface of Mars: Evidence for subsurface ice deposits. *Science*, *297*(5578), 81-85. doi: 10.1126/science.1073722

Gordon, I., Rothman, L., Hargreaves, R., Hashemi, R., Karlovets, E., Skinner, F., & ... Yurchenko, S. (2022). The HITRAN2020 molecular spectroscopic database. *Journal of Quantitative Spectroscopy and Radiative Transfer*, *277*. doi: 10.1016/j.jqsrt.2021.107949

Haberle, R. M., Forget, F., Colaprete, A., Schaeffer, J., Boynton, W. V., Kelly, N. J., & Chamberlain, M. A. (2008). The effect of ground ice on the martian seasonal co_2 cycle. *Planetary and Space Science*, *56*, 251-255. doi: 10.1016/j.pss.2007.08.006

Haberle, R. M., Kahre, M. A., Hollingsworth, J. L., Montmessin, F., Wilson, R. J., Urata, R. A., ... Schaeffer, R., James (2019). Documentation of the NASA/Ames Legacy Mars Global Climate Model: Simulations of the present seasonal water cycle. *Icarus*, *333*, 130-164. doi: 10.1016/j.icarus.2019.03.026

Harris, L., Chen, X., Putman, W., Zhou, L., & Chen, J.-H. (2021). A scientific description of the GFDL finite-volume cubed-sphere dynamical core. *NOAA technical memorandum OAR GFDL*. doi: 10.25923/6nhs-5897

Kahre, M. A., Wilson, R. J., Haberle, R. M., Harman, C. E., Urata, R. A., Kling, A., ... Bertrand, T. (2022, June). Update and status of the Mars Climate Modeling Center at NASA Ames Research Center. In *Seventh international workshop on the mars atmosphere: Modelling and observations.* Paris, FR. doi: 2022mamo.conf.1101K

Lin, S.-J. (1997). A finite-volume integration method for computing pressure gradient force in general vertical coordinates. *Quarterly Journal of the Royal Meteorological Society*, *123*, 1749-1762. doi: 10.1002/qj.49712354214

Lin, S.-J. (2004). A "vertically lagrangian" finite-volume dynamical core for global models. *Monthly Weather Review*, *132*, 2293-2307. doi: 10.1175/1520-0493(2004)132⟨2293:AVLFDC⟩2.0.CO;2

Lin, S.-J., & Putman, W. M. (2007). Finite-volume transort on various cubed-sphere grids. *Journal of Computational Physics*, *227*, 55-78. doi: 10.1016/j.jcp.2007.07.022

Lin, S.-J., & Rood, B. R. (1996). Multidimensional flux-form semi-lagrangian transport schemes. *Monthly Weather Review*, *124*, 2046-2070. doi: 10.1175/1520-0493(1996)124⟨2046:MFFSLT⟩2.0.CO;2

Lin, S.-J., & Rood, B. R. (1997). An explicit flux-form semi-lagrangian shallow-water model on

the sphere. *Quarterly Journal of the Royal Meteorological Society*, *123*, 2477-2498. doi: 10.1002/qj.49712354416

Mellor, G. L., & Yamada, Y. (1982). Development of a turbulence closure model for geophysical fluid problems. *Reviews of Geophysics*, *20*(4), 851-875. doi: 10.1029/RG020i004p00851

Montabone, L., Forgét, F., Millour, E., Wilson, R. J., Lewis, S. R., Cantor, B., ... Wolff, M. J. (2015). Eight-year climatology of dust optical depth on Mars. *Icarus*, *251*, 65-95. Retrieved from http://www-mars.lmd.jussieu.fr/mars/dust_climatology/ doi: 10.1016/j.icarus.2014.12.034

Montmessin, F., Forget, F., Rannou, P., Cabane, M., & Haberle, R. M. (2004). Origin and role of water ice clouds in the martian water cycle as inferred from a general circulation model. *Journal of Geophysical Research: Planets*, *109*. doi: 10.1029/2004JE002284

Putzig, N. E., & Mellon, M. T. (2007). Apparent thermal inertia and the surface heterogeneity of Mars. *Icarus*, *191*, 68-94. doi: 10.1016/j.icarus.2007.05.013

Rothman, L., Gordon, I., Barber, R., Dothe, H., Gamache, R., Goldman, A., ... Tennyson, J. (2010). HITEMP, the high-temperature molecular spectroscopic database. *Journal of Quantitative Spectroscopy and Radiative Transfer*, *111*, 2139-2150. doi: 10.1016/j.jqsrt.2010.05.001

Schwenke, D. (1998). New H2O rovibrational line assignments. *Journal of Molecular Spectroscopy*, *190*. doi: 10.1006/jmsp.1998.7603

Simmons, A. J., & Burridge, D. M. (1981). An energy and angular-momentum conserving vertical finite-difference scheme and hybrid vertical coordinates. *Monthly Weather Review*, *109*, 758-766. doi: 10.1175/1520-0493(1981)109⟨0758:AEAAMC⟩2.0.CO;2

Smith, D. E., Zuber, M. T., Solomon, S. C., Phillips, R. J., Head, J. W., Garvin, J. B., ... Duxbury, T. C. (1999). the global topography of Mars and implications for surface evolution. *Science*, *284*(5419), 1495-1503. doi: 10.1126/science.284.5419.1495

Toon, O. B., Mckay, C. P., Ackerman, T. P., & Santhanam, K. (1989). Rapid calculation of radiative heating rates and photodissociation rates in inhomogenous multiple scattering atmospheres. *Journal of Geophysical Research*, *94*(D13), 16287-16301. doi: 10.1029/JD094iD13p16287

Tyler, D., & Barnes, J. R. (2014). Atmospheric mesoscale modeling of water and clouds during northern summer on Mars. *Icarus*, *237*, 388-414. doi: 10.1016/j.icarus.2014.04.020

Wilson, R. J., Neumann, G. A., & Smith, M. D. (2007). Diurnal variation and radiative influence of Martian water ice clouds. *Geophysical Research Letters*, *34*(2). doi: 10.1029/2006GL027976

Wolff, M. J., Smith, M. D., Clancy, T. R., Arvidson, R., Kahre, M. A., Seelos IV, F., ... Savijärvi, H. (2009). Wavelength dependence of dust aerosol single scattering albedo as observed by the Compact Reconnaissance Imaging Spectrometer. *Journal of Geophysical Research: Planets*, *114*. doi: 10.1029/2009JE003350