TIEGCM Documentation

Release 2.0

NCAR High Altitude Observatory

March 21, 2016

CONTENTS

1 Introduction **3**
1.1 Related TIEGCM Documents and Links . . . 3

2 QuickStart Procedure **5**
2.1 Downloading the model source code and required data files . . . 5
2.2 Making a Default 5-deg model run . . . 5
2.3 Switching to 2.5-degree Model Resolution . . . 6
2.4 Making a Continuation (Restart) Run . . . 6
2.5 Moving to "Production" Mode . . . 6
2.6 Notes for Users on the HAO network . . . 7
2.7 Notes for Users on the NCAR /glade disk (yellowstone) . . . 7

3 Directory Structure **9**
3.1 Working Directory (created by the user) . . . 9
3.2 Model Directory (model source code and supporting utilities) . . . 10
3.3 Data Directory (startup and data files) . . . 11
3.4 Execution Directory (build and execute) . . . 13

4 Grid Structure and Resolution **15**
4.1 Geographic 3d spatial coordinates at 5-degree resolution . . . 15
4.2 Geographic 3d spatial coordinates at 2.5-degree resolution . . . 16
4.3 Geomagnetic 3d spatial coordinates . . . 16
4.4 Altitude Coordinates in the NCAR TIE-GCM and TIME-GCM . . . 16

5 Run Control Parameters: The Namelist Input File **19**
5.1 Example Namelist Input Files . . . 19
5.2 Explanation of Valid Namelist Parameters . . . 19

6 Using the job scripts to set up and submit a model run **39**
6.1 The Linux desktop job script (tiegcm-linux.job) . . . 39
6.2 The yellowstone supercomputer job script (tiegcm-ys.job) . . . 41

7 Model Output History Files **43**
7.1 NetCDF History Output Files . . . 43
7.2 Primary and Secondary History Files . . . 43
7.3 Table of Primary History Fields . . . 43

8 Saving Diagnostic Fields **45**
8.1 Table of Available Diagnostics . . . 45
8.2 Saving Fields/Arrays from the Source Code . . . 46
8.3 Details of Diagnostic Field Calculations . . . 47

9 Model Source Code **109**
9.1 The Academic License Agreement . . . 109
9.2 Source Code Flow Diagram . . . 109
9.3 Modifying the Source Code . . . 109

10 The Make/build process: compile and link **111**
10.1 Compilers . . . 111
10.2 Required External Libraries . . . 112
10.3 Build for Debugging . . . 112

11 Benchmark Runs **115**
11.1 Making Benchmark Runs . . . 116
11.2 Model Output History Files of the tiegcm2.0 Benchmark runs . . . 118

12 Post-Processing and Visualization **119**
12.1 tgcmproc_f90 . . . 119
12.2 tgcmproc_idl . . . 119
12.3 utproc . . . 120

13 Contact Information **121**

14 Glossary **123**

15 Indices and tables **127**

Index **129**

Note: This document is up to date for TIEGCM version 2.0

Contents:

CHAPTER

ONE

INTRODUCTION

The NCAR/HAO Thermosphere Ionosphere Electrodynamics General Circulation Model, or TIEGCM, is a three-dimensional, time-dependent, numeric simulation model of the Earth's upper atmosphere, including the upper Stratosphere, Mesosphere and Thermosphere.

This document is intended to assist members of the upper atmosphere research community in building, executing, and visualizing the TIEGCM as an integral part of their research. We also welcome development efforts on the model itself, subject to the Academic Research License Agreement `tiegcmlicense.txt`.

1.1 Related TIEGCM Documents and Links

- Main TGCM website
- TIEGCM Release Documentation for version 2.0.
- TIEGCM Benchmark Results for version 2.0.
- Model Description
- tgcmgroup email list

CHAPTER

TWO

QUICKSTART PROCEDURE

This chapter is intended to provide a tutorial-like procedure for downloading source code and start-up data files, building the model, and executing a short default run on a 64-bit Linux system, or a supercomputer (linux cluster) like the NCAR yellowstone machine.

2.1 Downloading the model source code and required data files

The model source code and related input data files may be downloaded from the tiegcm2.0 download page (you will need to provide an email address, but login and password are NOT required).

TIEGCM download page http://www.hao.ucar.edu/modeling/tgcm/download.php

The following tar files are available:

- Instructions (readme.download)
- Source Code tar file (30 MB) (tiegcm2.0.tar)
- Startup and data tar file for 5.0-deg model (tiegcm_res5.0_data.tar) (1.5 GB)
- Startup and data tar file for 2.5-deg model (tiegcm_res2.5_data.tar) (4 GB)

Download the source code tar file and the data tar file for the 5.0-deg model to a large scratch disk on either your Linux desktop, or the NCAR supercomputer yellowstone. To download all files and make default runs at both resolutions you will need at least 5.5 GB of disk space. Extracting these tarballs will result in directories with the same names. When extracting the source code, you will also get these default namelist input files and job scripts (or you can download them here):

- `tiegcm-linux.job`: Default csh job script for 64-bit Linux desktop
- `tiegcm-ys.job`: Default csh job script for the NCAR supercomputer yellowstone
- `tiegcm_res5.0_default.inp`: Default namelist input for 5.0-degree resolution model
- `tiegcm_res2.5_default.inp`: Default namelist input for 2.5-degree resolution model

2.2 Making a Default 5-deg model run

The job scripts are set up to make a short (1-day) 5-degree model run (March Equinox Solar Minimum conditions). At this point, you should be able to simply type the job script name appropriate for the current machine/system on the command line. The job script will create an execution directory (tiegcm.exec), and build and execute the model there. If successful, the stdout log will be tiegcm_res5.0.out, and model output netCDF history files will be in the execution directory.

Note: A warning for user's of previous revisions of TIEGCM: Do not use old namelist input files or job scripts from previous revisions. Copy the default files from the *scripts/* directory, and modify them for your own runs. Also, for initial runs, do not build/execute the model in an old *execdir*. Instead, allow the job script to make a new execdir for you.

2.3 Switching to 2.5-degree Model Resolution

To make a default run of the 2.5-deg model, edit the job script and reset 4 shell variables as follows:

- set tgcmdata = tiegcm_res2.5_data
- set input = tiegcm_res2.5.inp
- set output = tiegcm_res2.5.out
- set modelres = 2.5

If you are on the NCAR supercomputer yellowstone, you should also make the following changes to tiegcm-ys.job, to use 64 cores:

- #BSUB -n 64
- #BSUB -R "span[ptile=16]"

Then execute the job script for the default 2.5-deg model run.

2.4 Making a Continuation (Restart) Run

A model run can be continued (restarted) from the last primary history written by the previous run. To do this, you must modify the namelist input file as follows (refer to *Explanation of Valid Namelist Parameters* for more information):

1. Comment or remove SOURCE and SOURCE_START if the previous run was an initial run.
2. Reset START_DAY, START and STOP as necessary
3. Make sure one of the files in the OUTPUT list contains the new START history
4. Increment the starting volume number of SECOUT (pre-existing secondary output history files will be overwritten).

2.5 Moving to "Production" Mode

When you are ready to make longer model runs, or especially if you are planning to modify the source code, its best to move your working directory (with the model source directory, and any job scripts or namelist input files) to a disk space that is regularly backed up, e.g., under your home. You can leave the data (*tgcmdata*) and execution directories (*execdir*) on the large scratch disk, but you must then set the tgcmdata and execdir to absolute paths in the job script.

As you proceed, you can create new working directories, and corresponding execdirs as needed. If you modify the source code, the job script will call gmake, and dependent source files will be recompiled as necessary. If you switch between resolutions using the same execdir, the entire code will be recompiled for the new resolution.

2.6 Notes for Users on the HAO network

- Startup and data files for tiegcm2.0 are in /hao/aim/tgcm/data/tiegcm2.0
- The /hao/aim disk can be slow so its best (and probably fastest) to run the model on the local Linux desktop disk (e.g., something like: set execdir = /export/data1/$user/tiegcm.exec)
- Its usually best to run with 4 or 8 processors on the Linux box (set nproc = 4).
- Although the model has been built with PGI and gfortran at hao, the model will run fastest if built with the Intel compiler (set make = Make.intel_hao64)

2.7 Notes for Users on the NCAR /glade disk (yellowstone)

- For more information on using yellowstone, see NCAR CISL documentation
- Startup and data files for tiegcm2.0 are in /glade/p/hao/tgcm/data/tiegcm2.0

The yellowstone system uses the Load Sharing Facility (LSF) as a batch job management system:

- See LSF Introduction Guide for a brief overview.
- Also see CISL Platform LSF job script examples
- LSF resources are specified using LSF #BSUB commands. The default yellowstone job script tiegcm-ys.job uses the following settings:

```
#BSUB -J tiegcm                 [arbitrary job name]
#BSUB -P P28100036              [your authorized NCAR project number (this one is used at hao)]
#BSUB -q premium                [queue (can be regular, premium, standby, etc)]
#BSUB -o tiegcm.%J.out          [specify stdout file (different from model stdout)]
#BSUB -e tiegcm.%J.out          [specify stderr]
#BSUB -N                        [not sure what this is for]
#BSUB -u $LOGNAME@ucar.edu      [send email to this address after job has completed]
#BSUB -W 0:30                   [wallclock limit hh:mm (max 12 hours at NCAR)]
#BSUB -n 16                     [use 16 processors (64 for 2.5-deg tiegcm)]
```

- To calculate wallclock time for a 5-deg run with 16 cores, use .07 secs/timestep. For example, a 1-day simulation with a 60 sec timestep: ((24*3600) / 60 * .07) / 60 = 1.68 minutes
- To calculate wallclock time for a 2.5-deg run with 64 cores, use .15 secs/timestep. For example, a 1-day simulation with a 30 sec timestep: ((24*3600) / 30 * .15) / 60 = 7.2 minutes

CHAPTER

THREE

DIRECTORY STRUCTURE

To make a model run, some directory paths must be defined:

- *workdir*: User created directory from which model runs are submitted.
- *modeldir*: Directory containing the model source code (may be an svn working copy)
- *tgcmdata*: Location of start-up history and input data files (see *Data Directory (startup and data files)* below)
- *execdir*: Directory in which the model will be built and executed

These paths are set by the user in the job script, as in these examples: (these can be absolute paths or relative to the working directory):

```
set modeldir = tiegcm_trunk
set tgcmdata = /my/prettybig/data/tiegcm.data
set execdir  = /my/big/exec/disk/tiegcm.exec
```

See this `example job script`

3.1 Working Directory (created by the user)

To get started, the user will typically create a working directory from which model runs are submitted. The user's working directory will typically look something like this:

```
                    workdir
                       |
-----------------------------------------------
          |                        |
       *.inp                  modeldir/
       *.job
       *.out
```

Here, *.inp are *namelist input* files, *.job are *job script*'s, and *.out are stdout *output* files from model and are validated by the input module (input.F).

The job script in your working directory contains a shell variable specifying the path to the *modeldir*, so it knows where to find the source code and supporting scripts for the build process. The namelist input file also refers to the *datadir* path for start-up and other data input files (e.g., *SOURCE*, *GPI_NCFILE*, *IMF_NCFILE*, etc). These namelist parameters can use the environment variable *TGCMDATA* to specify the *datadir* (see section on *namelist input files*). The job script shell variable *tgcmdata*, if set, will override the TGCMDATA env var.

3.2 Model Directory (model source code and supporting utilities)

The model root directory is what you get when you *download* the model source code tar file, or check out the code from the svn repository. The model directory contains subdirectories with the model source code, supporting scripts, documentation, and a python code to make test and benchmark runs:

```
                              modeldir
                                 |
-------------------------------------------------------------------------
   |               |                |                |             |
 src/          scripts/           doc/            tgcmrun/     benchmarks/
  |               |                |                |             |
 *.F90          Make.*         userguide/          *.py      run_climatology
 *.F           linux.job       description/        run_*       run_seasons
 *.h            ibm.job          release/         tgcmrun       run_storms
             default.inp       diags.table                    archive_hpss
            tgcm_contents      perf.table                     make_listings
             tgcm_ncdump    README.download                     postproc/
                etc
```

src/ directory contents:

- Fortran source code *.F, *.F90, *.h. The source code is f90 standard compliant, and most source files are in fixed-format fortran. There is a single header file, defs.h, which contains grid definitions and dimensions for different *resolution* s.

scripts/ directory contents:

- **Make.***: Makefiles containing platform-dependent compiler flags, Make variables, and library locations. These files be copied, renamed, and customized for the user's platform/machine environment.
- **Make.*_hao64**: Three compilers are supported on the linux desktop platform: intel, pgi, and gfortran.
- **Make.intel_ys**: Makefile for intel compiler on the NCAR supercomputer yellowstone.
- **Makefile**: The main makefile. The Make.xxx file currently in use is included in the Makefile at build time.
- **tiegcm-linux.job**: Default model *job script* for Linux desktop systems.
- **tiegcm-ys.job**: Default model *job script* for the NCAR yellowstone supercomputer.
- **tiegcm_res5.0_default.inp**: Default namelist input file for 5.0-degree resolution.
- **tiegcm_res2.5_default.inp**: Default namelist input file for 2.5-degree resolution.
- **download**: Directory in which to make source and data tar files for *download* from the TGCM website

There are several additional utilities in the scripts directory that are used by the build system or by the user to perform various tasks or to obtain information (see `README in scripts directory` for more information). directory for more information.

doc/ directory contents:

- **userguide/**: Directory containing Python Sphinx source files for the User's Guide (this document)
- **description/**: Directory containing source files for the Model Description
- **release/**: Directory containing source files for the Release Documentation
- **diags.table**: `Table of diagnostic fields` that can be saved on secondary history files.
- **perf.table**: `Table of performance statistics` for both models (tiegcm and timegcm) at both *resolution*.

- **README.download**: `Instructions` for how to make a quick-start default build and execution of the model after downloading the source code and data.

tgcmrun/ directory contents:

- Python code to make benchmark runs interactively or from shell scripts. Type 'tgcmrun' on the command line for interactive, or execute the run_* scripts to make benchmark series runs.
- For more information on benchmark runs made for the current release, please see Release Documentation

benchmarks/ directory contents:

- Shell scripts that call *tgcmrun/* to make benchmark runs:
  - run_climatology
  - run_seasons
  - run_storms
  - run_perf
- Script archive_hpss to archive benchmark runs on the hpss (see HPSS)
- Script make_listings for making lists of files related to benchmark runs
- Subdirectory postproc/ contains scripts that call the tgcmproc utility to post-process benchmark runs.

3.3 Data Directory (startup and data files)

The public TIEGCM data directory is what you get when you *download* the data tar file. This directory is typically referred to with the environment variable *TGCMDATA*, but can be set with the *tgcmdata* shell variable in the job script (the shell variable, if set, will take precedence over the environment variable):

```
                datadir for tiegcmx.x
                        |
-------------------------------------------
                        |
             tiegcmx.x_res5.0_*.nc
             tiegcmx.x_res2.5_*.nc
                     gpi*.nc
                   gswm*5.0d*.nc
                   gswm*2.5d*.nc
                   imf_OMNI_*.nc
                      etc
```

These are netCDF history startup and data files for running the current version of the model (tiegcm2.0) They are specified in the namelist input file (see *namelist input files* for more information). These files are available for download, see *Downloading the model source code and required data files*.

- **tiegcmx.x_res5.0_*.nc**: History start-up files for the 5.0-degree resolution model. These files contain a single history with initial conditions for starting the model at a specific date and time. These are typically the first history for a benchmark run (seasonal, storm simulations, and climatologies). Namelist input parameter: *SOURCE*. Here are the files for tiegcm2.0:

  tiegcm_res5.0_climatology_smax_prim.nc
  tiegcm_res5.0_climatology_smin_prim.nc
  tiegcm_res5.0_dec2006_heelis_gpi_prim.nc
  tiegcm_res5.0_dec2006_weimer_imf_prim.nc

tiegcm_res5.0_decsol_smax_prim.nc
tiegcm_res5.0_decsol_smin_prim.nc
tiegcm_res5.0_junsol_smax_prim.nc
tiegcm_res5.0_junsol_smin_prim.nc
tiegcm_res5.0_mareqx_smax_prim.nc
tiegcm_res5.0_mareqx_smin_prim.nc
tiegcm_res5.0_nov2003_heelis_gpi_prim.nc
tiegcm_res5.0_nov2003_weimer_imf_prim.nc
tiegcm_res5.0_sepeqx_smax_prim.nc
tiegcm_res5.0_sepeqx_smin_prim.nc
tiegcm_res5.0_whi2008_heelis_gpi_prim.nc
tiegcm_res5.0_whi2008_weimer_imf_prim.nc

- **tiegcmx.x_res2.5_*.nc**: History start-up files for the 2.5-degree resolution model. These files contain a single history with initial conditions for starting the model at a specific model date and time. These are typically the first history for a benchmark run (seasonal, storm simulations, and climatologies). Namelist input parameter: *SOURCE* Here are the files for tiegcm2.0:

tiegcm_res2.5_bgndlbc_hwm_msis.nc
tiegcm_res2.5_bgndlbc_saber_hrdi.nc
tiegcm_res2.5_climatology_smax_prim.nc
tiegcm_res2.5_climatology_smin_prim.nc
tiegcm_res2.5_dec2006_heelis_gpi_prim.nc
tiegcm_res2.5_dec2006_weimer_imf_prim.nc
tiegcm_res2.5_decsol_smax_prim.nc
tiegcm_res2.5_decsol_smin_prim.nc
tiegcm_res2.5_junsol_smax_prim.nc
tiegcm_res2.5_junsol_smin_prim.nc
tiegcm_res2.5_mareqx_smax_prim.nc
tiegcm_res2.5_mareqx_smin_prim.nc
tiegcm_res2.5_nov2003_heelis_gpi_prim.nc
tiegcm_res2.5_nov2003_weimer_imf_prim.nc
tiegcm_res2.5_sepeqx_smax_prim.nc
tiegcm_res2.5_sepeqx_smin_prim.nc
tiegcm_res2.5_whi2008_heelis_gpi_prim.nc
tiegcm_res2.5_whi2008_weimer_imf_prim.nc

- **gpi*.nc** GeoPhysical Indices data files (3-hourly Kp and F10.7 cm solar flux). Namelist Input parameter: *GPI_NCFILE*
- **gswm*5.0d*.nc** Global Scale Wave Model data files, used to specify tidal perturbations for the lower boundary of the TIEGCM for the 5-degree resolution. There are 4 separate files for diurnal, semi-diurnal, migrating and non-migrating tides. Namelist Input parameter: *GSWM*.
- **gswm*2.5d*.nc** Global Scale Wave Model data files, used to specify tidal perturbations for the lower boundary of the TIEGCM for the 2.5-degree resolution. There are 4 separate files for diurnal, semi-diurnal, migrating and non-migrating tides. Namelist Input parameter: *GSWM*.

- **imf_OMNI_*.nc** Interplanetary Magnetic Field OMNI data files. Namelist read parameter is *IMF_NCFILE*. These files contain data for the BX,BY,BZ components of the IMF, solar wind velocity and solar wind density. See HAO public ftp page to download imf data files for years not included on the tiegcm2.0 data download.

3.4 Execution Directory (build and execute)

The model is built and executed in the execution directory (*execdir*). The path to the execution directory is specified by the execdir shell variable in the *job script*. The job script will create the execdir for you if it does not already exist. The following file types are typically found in the execution directory:

Note: When making your first run, its best to let the job script create the execdir for you. It is not wise to use an execdir used for revisions prior to tiegcm2.0. Also, if you have build or execution problems, it will sometimes help to remove the execdir and let the job script start over.

- ***.o**: Object files produced by the compiler.
- ***.mod**: Module files produced by the compiler.
- ***PET*LogFile**: ESMF log files.
- **tiegcm*.nc**: Model output netCDF history files.
- **M***: Makefiles.

The model executable also resides in the execution directory.

CHAPTER

FOUR

GRID STRUCTURE AND RESOLUTION

The TIEGCM can be configured for two spatial/temporal resolutions (use the *modelres* shell variable in the *job script* to set the model resolution):

- 5 degrees lat x lon, 2 grid points per scale height
  (default time step = 60 secs)
- 2.5 degrees lat x lon, 4 grid points per scale height
  (default time step = 30 secs)
- The vertical coordinate `lev`, or *Zp*, is a log-pressure scale `ln(p0/p)`, where p is pressure and p0 is a reference pressure. Fields are calculated at either "interface" levels (`ilev`), or at "midpoint" levels (`lev`) (see lev and ilev coordinate definitions below).
  - At 5.0 degree horizontal, Zp at interfaces = -7 to +7 by 0.5
  - At 2.5 degree horizontal, Zp at interfaces = -7 to +7 by 0.25

Note: To interpolate model fields to constant height surfaces, you should use geometric height, which is available on the 3d model grid as "ZG" on secondary histories. See the section below on *Altitude Coordinates the NCAR TIEGCM* for a detailed explanation of the relationship between Zp and Altitude.

4.1 Geographic 3d spatial coordinates at 5-degree resolution

Following are spatial coordinates for the 5x5-degree latxlon horizontal grid, with two grid points per scale height in the vertical (delta *Zp* = 0.5):

- Dimensions:
  - lon = 72 ;
  - lat = 36 ;
  - lev = 29 ;
  - ilev = 29 ;
- Coordinates:
  - lon = -180W to +180E by 5 degrees
  - lat = -87.5S to +87.5N by 5 degrees
  - lev = -6.75 to +7.25 by 0.5
  - ilev = -7 to +7 by 0.5

4.2 Geographic 3d spatial coordinates at 2.5-degree resolution

Following are spatial coordinates for the 2.5x2.5-degree latxlon horizontal grid, with four grid points per scale height in the vertical (delta *Zp* = 0.25):

- Dimensions:
  - lon = 144 ;
  - lat = 72 ;
  - lev = 57 ;
  - ilev = 57 ;
- Coordinates:
  - lon = -180W to 177.5E by 2.5 degrees
  - lat = -88.75S to 88.75N by 2.5 degrees
  - lev = -6.875 to 7.125 by 0.25
  - ilev = -7 to +7 by 0.25

4.3 Geomagnetic 3d spatial coordinates

The longitude geomagnetic coordinate is from -180 to +180 by 4.5 degrees. The latitude coordinate is non-regular, with resolution increasing toward the magnetic equator. The vertical *Zp* (ln(p0/p)) interface coordinate is from -8.5 to 7 by 0.25:

- Dimensions:
  - mlon = 81 ;
  - mlat = 97 ;
  - mlev = 63 ;
  - imlev = 63 ;
- Coordinates:
  - mlon = -180W to 180E by 4.5 degrees
  - mlat = -90S to 90N: irregular, increasing resolution equatorward
  - mlev = -8.25 to 7.25 by 0.25
  - imlev = -8.5 to 7.0 by 0.25

4.4 Altitude Coordinates in the NCAR TIE-GCM and TIME-GCM

Author: Stan Solomon Date: April, 2016

The purpose of this document is to define the altitude coordinate systems used in the NCAR Thermosphere-Ionosphere-Electrodynamics General Circulation Model (TIE-GCM) and Thermosphere-Ionosphere-Mesosphere-Electrodynamics General Circulation Model (TIME-GCM), especially to inform model users as to how to register model output in the vertical dimension.

The TIE-GCM and TIME-GCM use a log-pressure coordinate system, with each pressure level defined as ln(P0/P), where P0 = 5x10-4 dynes/cm2 = 5x10-5 Pascal = 5x10-7 hPa = 5x10-7 mb. (Native units in these models are cgs, i.e., dynes/cm2.) This pressure occurs at ~200 km altitude, depending on conditions.

The TIE-GCM vertical coordinate extends from -7 to +7 (~97 km to ~600 km) and the TIME-GCM vertical coordinate extends from -17 to +7 (~30 km to ~600 km). Each integer interval in pressure level is one scale height apart, so the low-resolution (5°x5°xH/2) versions are spaced at half-integer intervals and the high-resolution (2.5°x2.5°xH/4) versions of the models are spaced at quarter-integer intervals:

Model/Resolution	Num Levels	Level Spacing	Bottom Level	Top Level	Min Alt	Max Alt
Low-Res TIE-GCM	29	0.5	-7	+7	97 km	600 km
High-Res TIE-GCM	57	0.25	-7	+7	97 km	600 km
Low-Res TIME-GCM	49	0.5	-17	+7	30 km	600 km
High-Res TIME-GCM	97	0.25	-17	+7	30 km	600 km

The height of the pressure surface is defined at each grid point in arrays provided in output history files (in cm). Unfortunately, there are four different possibilities for altitude definition, all slightly different.

First, we define the geopotential height z. Geopotential height is the height that the pressure surface would be, assuming that the acceleration due to gravity g is constant at the value used in the model calculations (870 cm/s2 for the TIE-GCM and 950 cm/s2 for the TIME-GCM). It is registered to the altitude of the model lower boundary, which can vary horizontally due to the tidal and climatological lower boundary specification. This is the native coordinate system for the models, and so z is included in all history files. However, it is not the appropriate altitude coordinate for comparison with real-world data. Also note that this definition of geopotential height is not the same as what is used in, e.g., tropospheric meteorology, because it is referenced to value of g that is different from the value of g at the surface (~980 cm/s2).

We can correct the geopotential height z to obtain geometric height zg. This is performed inside the models by subroutine calczg (addiag.F), using an empirical formulation of the variation of g over the globe (including centripetal force), and vertical integration, to account for the variation with altitude. It can also be done, using the same subroutine, in the Fortran model processers, and is also available in various IDL processing routines. Geometric height ZG is now forced onto secondary histories (i.e., it is output whether you request it or not) but not on primary histories (because primary histories contain only what is necessary to re-start the model). However, some older secondary history files may not include ZG which necessitates that it be calculated in the post-processing if needed for data comparison.

Now we come to the final complication, which is the distinction between model interfaces and model mid-points. The interfaces are the native coordinate system of the model grid, as defined in the table above, i.e., at -7.0, -6.5, -6.0, etc.; z and zg are defined on these interfaces. However, most model output quantities are actually reported at the midpoints, half-way between interfaces in pressure, i.e., at -6.75, -6.25, -5.75, etc. Each midpoint is a half-interval above the corresponding interface. All temperatures, winds, neutral densities, etc., are defined at these midpoints. However, electron density and electric potential are defined at the interfaces:

Field	Z	Zg	Zm	Tn	Un	Vn	O2	O	N2	NO	N	N2D	He	Ne	Te	Ti	OM	Pot
Specified at	I	I	M	M	M	M	M	M	M	M	M	M	M	I	M	M	I	I

In order to register midpoint quantities in altitude, it is therefore necessary to interpolate from the midpoints to the interfaces. Alternatively, it may be simpler to interpolate zg from the interfaces to the midpoints. For TIE-GCM 2.0, a new output variable has been added, ZGMID, which is geometric height that has been interpolated to the mid points. However, older history files do not include ZGMID. As with ZG, it is available on secondary histories but not on primary histories.

In output histories, quantities specified at interfaces are defined by the ilev coordinate variable and quantities specified at midpoints are defined by the lev coordinate variable. These quantities are generally numerically identical, but their definitions in the files can serve as a reminder of what is defined where.

Height-related Variables on TIEGCM Secondary Histories:

Note: Variables Z, ZG, and ZMAG are forced onto secondary histories. To save ZGMID to secondary histories, add ZGMID to the fields list in the namelist input file: SECFLDS='ZGMID'

Variable Name	Description
Z	Geopotential Height (cm)
ZG	Geometric Height (cm)
ZGMID	Geometric Height at Midpoints (cm)
ZMAG	Geopotential Height on Geomagnetic Grid (km)

CHAPTER

FIVE

RUN CONTROL PARAMETERS: THE NAMELIST INPUT FILE

The *namelist input* file specifies user-provided run control parameters for a model run. These parameters include the model startup file, start and stop times, solar inputs, lower boundary files, and several other flags and data files as necessary. This document describes each valid namelist parameter, their valid combinations, and provides several example input files for running the model in different modes and resolutions.

5.1 Example Namelist Input Files

Please refer to the following examples of namelist input files:

Note: Any part of a line in the namelist file following a exclamation mark '!' will be treated as a comment (see example files).

Example Namelist Input Files:

- The default input files:
  - `5-degree resolution namelist file`
  - `2.5-degree resolution namelist file`
- A continuation of the default 5-degree run:
  - `continuation.inp`
- Saving diagnostics on secondary history files:
  - `diags.inp`

5.2 Explanation of Valid Namelist Parameters

Following is a table of valid TIEGCM 2.0 namelist parameters, and their descriptions. Follow the parameter name links to explanations below.

Parameter Name	Data Type and Default	Description
AMIENH,AMIESH	string: [none]	Optional AMIE data files
AMIE_IBKG	integer: 0	Flag for how to read AMIE data
AURORA	integer: 1	0/1 flag for auroral parameterization
BGRDDATA_NCFILE	string: [none]	Data file for background lower boundary
BXIMF or BXIMF_TIME	real or real array	X-component of the IMF
BYIMF or BYIMF_TIME	real or real array	Y-component of the IMF

Continued on next page

Table 5.1 – continued from previous page

Parameter Name	Data Type and Default	Description
BZIMF or BZIMF_TIME	real or real array	Z-component of the IMF
CALENDAR_ADVANCE	real: 1	0/1 switch to advance calendar time
COLFAC	real: 1.5	O-O+ collision factor
CTMT_NCFILE	string: [none]	Lower boundary data file T,U,V,Z
CTPOTEN	real:	Cross-Tail Potential
CTPOTEN_TIME	real: [none]	Time-dependent Cross-Tail Potential
CURRENT_KQ	integer: 0	Height-integrated Current Density
CURRENT_PG	integer: 1	Current due to Plasma Pressure Gradient
CALC_HELIUM	integer: 1	0/1 switch for calculation of Helium
DYNAMO	integer: 1	0/1 switch for electro-dynamo
EDDY_DIF	integer: 0	0/1 switch for DOY-dependent or constant eddy diffusion
ENFORCE_OPFLOOR	integer: 1	Gaussian shaped floor for O+
F107 or F107_TIME	real or real array	Daily F10.7 cm solar flux
F107A or F107A_TIME	real or real array	81-day average F10.7 cm solar flux
GPI_NCFILE	string: [none]	Geophysical Indices (Kp) data file
GSWM data files	string: [none]	GSWM Model tidal lbc data files
HIST	integer(3)	Primary history write frequency
IMF_NCFILE	string: [none]	IMF OMNI data files
JOULEFAC	real: 1.5	Joule Heating Factor
KP or KP_TIME	real or real array	Kp for calc of hpower and ctpoten
LABEL	string:	Arbitrary string identifying the run
MXHIST_PRIM	integer: 10	Max histories on primary file
MXHIST_SECH	integer: 24	Max histories on secondary file
OPDIFFCAP	real: 0 [no cap]	Impose maximum O+ diffusion
OUTPUT	string array	Primary history output file(s)
POTENTIAL_MODEL	string: [HEELIS]	High-latitude Potential Model
POWER or POWER_TIME	real or real array	Hemispheric Power (GW)
SABER_NCFILE	string: [none]	SABER data file (T,Z lower boundary condition)
SECSTART	integer(3)	Secondary history start time (day,hour,minute)
SECSTOP	integer(3)	Secondary history stop time (day,hour,minute)
SECHIST	integer(3)	Secondary history write frequency (day,hour,minute)
SECFLDS	string array	Fields to be stored on secondary histories
SECOUT	string array	Secondary history output file(s)
SOURCE	string: [none]	Primary SOURCE (start-up) file
SOURCE_START	integer(3)	Model time to start on SOURCE file
START	integer(3)	Model start time (day,hour,minute)
START_YEAR	integer: 2002	Starting year
START_DAY	integer: 80	Starting day of year
STEP	integer: [none]	Model time step (seconds)
STOP	integer(3)	Model stop time (day,hour,minute)
SWDEN or SWDEN_TIME	real or real array	Solar Wind Density
SWVEL or SWVEL_TIME	real or real array	Solar Wind Velocity
TIDE	real(10)	Amplitudes and phases of semi-diurnal tide (rarely used)
TIDE2	real(2)	Amplitudes and phases of diurnal tide (rarely used)
TIDI_NCFILE	string: [none]	TIDI data file (U,V lower boundary condition)

AMIENH, AMIESH

Data files containing output from the AMIE model, to be imported to the tiegcm. AMIENH contains northern hemisphere data, AMIESH contains southern hemisphere data. Contact Gang Lu (ganglu@ucar.edu) for more information.

Data type: string
Default: [none]

Example:

- AMIENH = '$TGCMDATA/amie_apr01_10_2010_nh_ssusi.nc'
- AMIESH = '$TGCMDATA/amie_apr01_10_2010_sh_ssusi.nc'

See also:

- *AMIE_IBKG*

Back to top

AMIE_IBKG

Integer flag 0, 1, or 2 for reading real, first, or 24-hour averaged AMIE data

Data type: scalar integer
Default: 0

See also:

- *AMIE*

Back to top

AURORA

If AURORA > 0 then the auroral parameterization (aurora.F) is called by dynamics (dynamics.F), otherwise it is not called.

Data type: scalar integer
Default: 1

Back to top

BGRDDATA_NCFILE

Data file providing zonal mean climatology of T, U, V using MSIS and HWM empirical models, or UARS data. If no input file is specified, a flat lower boundary (u=v=0, Tn=181 K, z=96.4 km) is employed by default. Other zonal mean climatologies can be used by generating and specifying a different file.

Data type: string Default: [none]

Example:

- BGRDDATA_NCFILE = '$TGCMDATA/bgndlbc_hwm_msis.nc'
- BGRDDATA_NCFILE = '$TGCMDATA/bgndlbc_saber_hrdi.nc'

Back to top

BXIMF or BXIMF_TIME

X-component of the IMF. Can be specified as either a constant (BXIMF), or series of time-dependent values (BXIMF_TIME). If IMF_NCFILE is set and BXIMF is not provided, then BXIMF will be taken from the IMF data file.

Data type: real or real array

Examples:

- BXIMF = 0. ; constant for entire run
- BXIMF_TIME = 80,0,0,40., 80,1,0,30., 80,5,0,20. ; time series

See also:

- *BYIMF or BYIMF_TIME*
- *BZIMF or BZIMF_TIME*
- *IMF_NCFILE*

Back to top

BYIMF or BYIMF_TIME

Y-component of the IMF. Can be specified as either a constant (BYIMF), or series of time-dependent values (BYIMF_TIME). If IMF_NCFILE is set and BYIMF is not provided, then BYIMF will be taken from the IMF data file.

Data type: real or real array

Examples:

- BYIMF = 0. ; constant for entire run
- BYIMF_TIME = 80,0,0,40., 80,1,0,30., 80,5,0,20. ; time series

See also:

- *BXIMF or BYIMF_TIME*
- *BZIMF or BZIMF_TIME*
- *IMF_NCFILE*

Back to top

BZIMF or BZIMF_TIME

Z-component of the IMF. Can be specified as either a constant (BZIMF), or series of time-dependent values (BZIMF_TIME). If IMF_NCFILE is set and BZIMF is not provided, then BZIMF will be taken from the IMF data file.

Data type: real or real array

Examples:

- BZIMF = 0. ; constant for entire run
- BZIMF_TIME = 80,0,0,40., 80,1,0,30., 80,5,0,20. ; time series

See also:

- *BXIMF or BXIMF_TIME*
- *BYIMF or BYIMF_TIME*
- *IMF_NCFILE*

Back to top

CALENDAR_ADVANCE

Set CALENDAR_ADVANCE=1 to advance calendar time from START_DAY, otherwise calendar time is not advanced. If advancing calendar time, iday (init_module) is incremented every 24 hours, and the sun's declination and longitude is recalculated (see sub advance_day in advance.F and sub sunloc in magfield.F), thereby allowing seasonal change to take place. The earth's orbital eccentricity "sfeps" is also updated as a 6% variation in solar output over a year.

A run with CALENDAR_ADVANCE=0 is referred to as a "steady-state" run. This is often used to advance the model to a "steady-state" for a given date, prior to a seasonal run with CALENDAR_ADVANCE=1.

Back to top

COLFAC

O-O+ Collision Frequency, alias the "Burnside Factor". Default is 1.5, but there have been recommendations for 1.3. COLFAC is used in lamdas.F and oplus.F.

Data type: real
Default: 1.5

Back to top

CTMT_NCFILE

Tidal perturbations for lower boundary of Z, T, U, V. See this reference:

```
Ref. Oberheide, J., J. M. Forbes, X. Zhang, and S. L. Bruinsma
  Climatology of upward propagating diurnal and semidiurnal tides in the thermosphere
  J. Geophys. Res., 116, A11306, doi:10.1029/2011JA016784, 2011.
```

Note: This is mutually incompatible with GSWM_NCFILE

Data type: string
Default: [none]

Examples:

- CTMT_NCFILE = '$TGCMDATA/ctmt_tides.nc'

See also:

- *GSWM*

Back to top

CTPOTEN or CTPOTEN_TIME

Cross-tail (or cross-cap) potential. This is used in the auroral precipitation parameterization. It can be provided either as a single constant (CTPOTEN), or several time-dependent values (CTPOTEN_TIME). If GPI_NCFILE is set and CTPOTEN is not provided, it will be calculated from 3-hourly Kp data read from GPI_NCFILE.

The time-dependent example below specifies increasing CTPOTEN from model times 80,0,0 to 80,1,0, and 80,5,0. Interpolated values will be used between these specified model times.

Note that if POTENTIAL_MODEL='WEIMER' or 'WEIMER05', then the user is not allowed to provide CTPOTEN because it will be calculated from the Weimer electric potential.

Data type: real or real array

Examples:

- CTPOTEN = 60.
- CTPOTEN_TIME = 80,0,0,60., 80,1,0,65., 80,5,0,100.

See also:

- *POWER or POWER_TIME*
- *KP or KP_TIME*
- *GPI_NCFILE*

Back to top

CURRENT_KQ

If CURRENT_KQ=1, then height-integrated current density of current sheet, and upward current density at the top of the ionosphere is calculated (default=0) (ignored if DYNAMO=0) (see current.F90 to save JQR, JE13D, JE23D, KQPHI, KQLAM)

Data type: integer

Default: 0

Back to top

EDDY_DIF

If EDDY_DIF=1, then day-of-year dependent eddy diffusion will be calculated, otherwise eddy diffusion will be set to pressure-dependent constants. See cons.F.

Data type: integer

Default: 0

Back to top

ENFORCE_OPFLOOR

A double-Gaussian shaped floor (in latitude and altitude) is applied to O+ at low-to-mid latitudes in the F-region in order to keep the model stable when the ionosphere gets very low in density.

If set to 1 (the default), the floor is implemented in oplus.F as follows:

```
  do k=lev0,lev1-1
    opfloor = opmin*exp(-(glat(lat)/90.0)**2/0.3)*
|             exp(-((zpmid(k)-4.25)/zpmid(nlevp1))**2/0.1)
    do i=lon0,lon1
      if (opout(k,i,lat) < opfloor) opout(k,i,lat) = opfloor
    enddo
  enddo
```

Data type: integer

Default: 1

Back to top

CURRENT_PG

If CURRENT_PG=1, current due to plasma pressure gradient and gravity is calculated and included as a forcing term in the dynamo equation (default=1) (ignored if DYNAMO=0)

Data type: integer
Default: 1

Back to top

CALC_HELIUM

If calc_helium=1, helium is calculated as a major composition species. If calc_helium=0, helium is zeroed out. If calc_helium=1 and the source history does not have helium, then helium will be initialized globally to 0.1154E-5.

Data type: integer
Default: 1

Back to top

DYNAMO

Integer switch for electro-dynamo. If DYNAMO=0, then dynamo (pdynamo.F) will not be called, and ion drift velocities will be zero. If DYNAMO=1, then dynamo will be called, and ion drift velocities will be calculated.

Data type: integer
Default: 1

Back to top

F107 or F107_TIME

Daily F10.7 cm solar flux. This can be provided either as a single constant (F107), or several time-dependent values (F107_TIME). If GPI_NCFILE is set and F107 is not set, then F107 will be set from the data. The below example of F107_TIME increases the f10.7 flux from 120 to 150 in the first hour of model time, then to 200 by the fifth hour. Values are linearly interpolated at each time-step.

Data type: real or real array

Examples:

- F107 = 120.
- F107_TIME = 80,0,0,120., 80,1,0,150., 80,5,0,200.

See also:

- *F107A*
- *POTENTIAL_MODEL*
- *GPI_NCFILE*
- *IMF_NCFILE*

Back to top

F107A or F107A_TIME

81-day average F10.7 cm solar flux. This can be provided either as a single constant (F107A), or several time-dependent values (F107A_TIME). If GPI_NCFILE is set and F107A is not set, then F107A will be set from the data. The below example of F107A_TIME increases the f10.7a flux from 120 to 130 in 12 hours of model time.

Data type: real or real array

Examples:

- F107A = 120.
- F107A_TIME = 80,0,0,120., 80,6,0,125., 80,12,0,130.

See also:

- *F107*
- *POTENTIAL_MODEL*
- *GPI_NCFILE*
- *IMF_NCFILE*

Back to top

GPI_NCFILE

Specifies a netCDF data file containing 3-hourly Kp and daily F10.7 data to drive high-latitude convection and the auroral precipitation oval. If GPI_NCFILE is specified, and POTENTIAL_MODEL='HEELIS', then at least one of CTPOTEN,POWER,F107,F107A must **not** be specified. If CTPOTEN or POWER are not specified, they are calculated from the Kp data using empirical relationships (see source file gpi.F). If F107 or F107A are not specified, the data will be used.

If GPI_NCFILE is specified when POTENTIAL_MODEL='WEIMER' and IMF_NCFILE is specified, then the Weimer model and aurora will be driven by the IMF data, and only F107 and F107A will be read from the GPI data file (F107 is not available on IMF data files).

If the current model time is not available on the GPI data file, the model will print an error message to stdout, and stop.

Data Source: Ascii data is obtained from NOAA/NGDC, and an equivalent netCDF data file is written for import to the TGCM models (see code in hao:$TGCMROOT/mkgpi).

Datatype: string

Example:

- GPI_NCFILE = '$TGCMDATA/gpi_2000001-2009031.nc'

See also:

- *CTPOTEN or CTPOTEN_TIME*
- *POWER or POWER_TIME*
- *F107 or F107_TIME*
- *IMF_NCFILE*

Back to top

GSWM model data files for lbc

Paths to netCDF data files containing tidal perturbations from the Global Scale Wave Model. If provided, the files will be read and the perturbations will be added to the lower boundary conditions of T,U,V, and Z. If provided, then TIDE and TIDE2 must be zeroed out.

Warning: As of version 2.0, the model is not tuned to use the non-migrating GSWM tidal components. The default namelist input file specifies migrating diurnal and semi-diurnal tides, but not the non-migrating components. In later releases, non-migrating tides may be supported at the 2.5-deg resolution.

GSWM files must contain data compatable with the lower boundary of the model (99 km), and the horizontal resolution of the model being run (either 5 or 2.5 degrees). See examples below.

Datatype: string

Examples:

- GSWM files for the 5-degree TIEGCM:

```
GSWM_MI_DI_NCFILE   = '$TGCMDATA/gswm_diurn_5.0d_99km.nc'
GSWM_MI_SDI_NCFILE  = '$TGCMDATA/gswm_semi_5.0d_99km.nc'
GSWM_NMI_DI_NCFILE  = '$TGCMDATA/gswm_nonmig_diurn_5.0d_99km.nc'
GSWM_NMI_SDI_NCFILE = '$TGCMDATA/gswm_nonmig_semi_5.0d_99km.nc'
```

- GSWM files for 2.5-degree TIEGCM:

```
GSWM_MI_DI_NCFILE   = '$TGCMDATA/gswm_diurn_2.5d_99km.nc'
GSWM_MI_SDI_NCFILE  = '$TGCMDATA/gswm_semi_2.5d_99km.nc'
GSWM_NMI_DI_NCFILE  = '$TGCMDATA/gswm_nonmig_diurn_2.5d_99km.nc'
GSWM_NMI_SDI_NCFILE = '$TGCMDATA/gswm_nonmig_semi_2.5d_99km.nc'
```

See also:

- *TIDE*
- *TIDE2*

Back to top

HIST

Primary history write frequency, specified as a model time (day,hour,minute). HIST time must divide evenly into STOP minus START times.

Examples:

- HIST = 1,0,0 ;request daily histories
- HIST = 0,1,0 ;request hourly histories
- HIST = 0,0,12 ;request 12-minute histories

See also:

- *SECHIST*

Back to top

POWER or POWER_TIME

Hemispheric Power (GW). This is used in the auroral precipitation parameterization. It can be provided either as a single constant (POWER), or several time-dependent values (POWER_TIME). If GPI_NCFILE is set and POWER is not provided, it will be calculated from 3-hourly Kp data read from GPI_NCFILE.

The time-dependent example below specifies increasing POWER from model times 80,0,0 to 80,1,0, and 80,5,0. Interpolated values will be used between these specified model times.

Data type: real or real array

Examples:

- POWER = 16.
- POWER_TIME = 80,0,0,16., 80,1,0,20., 80,5,0,70.

See also:

- *CTPOTEN or CTPOTEN_TIME*
- *KP or KP_TIME*
- *GPI_NCFILE*

Back to top

SABER_NCFILE

SABER data file for lower boundary conditions of T and Z (neutral temperature and geopotential height).

Data type: string
Default: [none]

See also:

- *TIDI_NCFILE*

Back to top

TIDI_NCFILE

TIDI data file for lower boundary conditions of U and V (zonal and meridional neutral wind).

Data type: string
Default: [none]

See also:

- *SABER_NCFILE*

Back to top

KP or KP_TIME

Geomagnetic Activity index. If KP is specified and POWER and/or CTPOTEN are commented, then the given KP will be used with empirical formulas to calculate POWER and/or CTPOTEN, which are used in the Auroral parameterization.

KP can be provided as a scalar constant (KP), or as a series of time-dependent values (KP_TIME), as in the below examples. KP cannot be set if GPI_NCFILE data file is specified.

Empirical formula used to calculate POWER from KP (see function hp_from_kp in util.F):

```
if (kp <=7.) hp_from_kp = 16.82*exp(0.32*kp)-4.86
if (kp > 7.) hp_from_kp = 153.13 + (kp-7.)/(9.-7.)*(300.-153.13)
```

Empirical formula used to calculate CTPOTEN from KP (see function ctpoten_from_kp in util.F):

```
ctpoten_from_kp = 15.+15.*kp + 0.8*kp**2
```

Examples:

- KP = 4.0
- KP_TIME = 80,0,0,4., 80,6,0,4.5, 80,12,0,5.0

See also:

- *CTPOTEN*
- *POWER*
- *GPI_NCFILE*

Back to top

IMF_NCFILE

Specifies a netCDF data file containing hourly IMF parameters BX,BY,BZ,SWVEL, and SWDEN. This can be set only when POTENTIAL_MODEL='WEIMER'. The data will be used to drive the Weimer 2005 potential model. When set, the user must **not** provide at least one of the above IMF parameters. Data will be used for IMF parameters not provided by the user. Values (scalar or time-dependent) that are provided by the user will take precedence over the data file.

If the current model time is not available on the IMF data file, the model will print an error message to stdout and stop.

Notes on creation of IMF OMNI data files:

- IMF data is derived from 1-minute OMNI satellite data available on CDAweb CDAweb. Our derivation is a multi-step process:
- Data gaps in the raw 1-minute OMNI data are linearly interpolated. If a gap happens to occur at the beginning or end of the time interval, it is set to the next known good data point.
- Gap-filled data is used to compute a 15 minute trailing average lagged by 5 minutes.
- Time averaged data is sampled at 5 minutes
- A data quality flag is calculated for every 5-minute sample point. The data quality flag is a boolean value set to "1" for all sample points derived from valid (not gap-filled) data. The data quality flag is set to "0" for any sample point that is derived from gap-filled data anywhere in the 15 minute trailing average lagged by 5 minutes.
- The data quality flag is stored in the NetCDF-formatted IMF input file. For any variable (ie. "swvel" - solar wind velocity), there exists a mask (ie. "velMask"). Find a complete list of IMF variables with command "ncdump -h [imf-file.nc]".
- Note: You should verify the IMF data quality before doing storm simulations. Known periods of invalid IMF data include approximately days 301 to 304 of 2003 (during the "Halloween Storm").

Example:

- IMF_NCFILE = '$TGCMDATA/imf_OMNI_2002001-2002365.nc'

Back to top

JOULEFAC

Joule heating factor. This factor is multiplied by the joule heating calculation (see subroutine qjoule_tn in qjoule.F).

Data type: real

Default: 1.5

Back to top

LABEL

LABEL may be any string up to 80 characters long, used to identify a run. The LABEL is written to output history files as a global file attribute. This parameter is purely a user convenience, and does not effect the model run in any way.

Data type: string

Default: 'tiegcm res=5'

Back to top

MXHIST_PRIM

Maximum number of histories to be written to primary OUTPUT files. When this many histories have been written to the current OUTPUT file, the next OUTPUT file is created and it receives subsequent histories. This parameter can be adjusted to control the size of primary OUTPUT files.

Data type: integer
Default: 10

Examples:

- MXHIST_PRIM = 15 ; allow maximum of 15 histories per primary output file

See also:

- *OUTPUT*

Back to top

MXHIST_SECH

Maximum number of histories to be written to secondary output files (SECOUT). When this many histories have been written to the current SECOUT file, the next SECOUT file is created and it receives subsequent histories. This parameter can be adjusted to control the size of secondary OUTPUT files.

Data type: integer
Default: 24

Examples:

- MXHIST_SECH = 24 ; allow 1 day of hourly histories per file
- MXHIST_SECH = 48 ; allow 2 days of hourly histories per file

See also:

- *SECOUT*

Back to top

OPDIFFCAP

Optional cap on ambipolar diffusion coefficient for O+. This can improve model stability in the topside F-region, but it is only recommended as a last resort since it will change model results. This is a new namelist parameter for tiegcm2.0. The default is 0., i.e., no cap. If this is non-zero (provided by the user), then it is implemented in subroutine rrk of src/oplus.F.

Data type: integer
Default: 0

Examples:

```
Tests have been made with these values:

* OPDIFFCAP = 1.5e8
* OPDIFFCAP = 3.0e8
* OPDIFFCAP = 6.0e8
* OPDIFFCAP = 8.0e8
```

Back to top

OUTPUT

List of primary history output files. Each file may be an absolute path, or relative to the execution directory. If an initial run (SOURCE is specified), then pre-existing OUTPUT files will be overwritten. If a continuation run (SOURCE is *not* specified), then the first OUTPUT file should contain the source history at START time. In this case, subsequent output histories will be appended to the first OUTPUT file until it is full. As each OUTPUT file is filled (see MXHIST_PRIM), the next OUTPUT file is created and histories are written until it is full, and so on.

OUTPUT files are usually specified with increasing integers imbedded in the names. See examples below. As a convenience, large sequences of files may be specified in a "short-form", see example 3 below specifying 20 files. By convention, primary history output files may use the letter "p" to indicate primary file series (see all 3 examples below, and contrast with SECOUT).

Examples:

```
OUTPUT = 'p_myoutput_001.nc'
OUTPUT = 'myoutput.p001.nc','myoutput.p002.nc','myoutput.p003.nc'
OUTPUT = 'myoutput_p001.nc','to','myoutput_p020.nc','by','1'
```

See also:

- *SECOUT*
- *SOURCE*
- *MXHIST_PRIM*

Back to top

POTENTIAL_MODEL

The high-latitude potential model used to calculate electric potential above a specified latitude. This string can have one of two values:

POTENTIAL_MODEL = 'HEELIS'
POTENTIAL_MODEL = 'WEIMER'

'HEELIS' is the Rod Heelis model (heelis.F). 'WEIMER' is the Dan Weimer 2005 model (wei05sc.F).

Note: The Weimer model of high-latitude potential is the intellectual property of Daniel Weimer and may not be extracted, distributed, or used for any purpose other than as implemented in the TIE-GCM. For further information concerning this model, please contact Dan Weimer (dweimer@vt.edu).

Data type: string
Default: 'HEELIS'

Back to top

SECFLDS

List of fields to be saved to secondary histories. These may be either fields that are also saved on primary histories (so-called "prognostic" fields), fields that have been requested via addfld calls in the source code, or fields available via the *diagnostics module* (see example below).

Note the final size of secondary output files is affected by the number of fields specified as well as the number of histories on the file. The file size can be controlled by setting the number of histories allowed on a secondary file *MXHIST_SECH*.

Data type: one or more character strings

Examples:

```
!
! Example list of fields to be written to secondary histories:
!
  SECFLDS = 'TN' 'UN' 'VN' 'O2' 'O1' 'N2' 'NO' 'N4S' 'HE' 'NE' 'TE' 'TI'
            'TEC' 'O2P' 'OP' 'OMEGA' 'POTEN' 'UI_ExB' 'VI_ExB' 'WI_ExB'
            'DEN' 'QJOULE' 'Z' 'ZG'
!
! This example lists all diagnostic fields available via the diags module
! (it is not necessary to call addfld in the code to obtain these fields)
!
  SECFLDS = 'CO2_COOL','NO_COOL','DEN','HEATING','QJOULE','QJOULE_INTEG',
      'SIGMA_PED','SIGMA_HAL','TEC','UI_ExB','VI_ExB','WI_ExB',
      'LAMDA_PED','LAMDA_HAL','HMF2','NMF2','SCHT','MU_M','O_N2','WN',
      'BX','BY','BZ','BMAG','EX','EY','EZ','ED1','ED2','PHIM2D','N2',
      'CUSP','DRIZZLE','ALFA','NFLUX','EFLUX'
```

See also: *MXHIST_SECH*

Back to top

SECSTART

Secondary history start time, specified as a model time (day,hour,minute).

Data type: 3 integers (day,hour,minute)
Valid range: 0-365 for day, 0-23 for hour, 0-59 for minute.

SECSTART time must follow these rules:

- It must be a multiple of timestep STEP and less than SECSTOP time.
- It must be greater than or equal to START time, and less than or equal to STOP time.
- In the case of an initial run (SOURCE history provided), SECSTART must not be equal to START time. This is to avoid zero valued secondary history fields.

Examples:

- SECSTART = 80,1,0 ; Start saving secondary histories at model time 80,1,0

See also:

- *SECSTOP*
- *SECHIST*

Back to top

SECSTOP

Secondary history stop time, specified as a model time (day,hour,minute).

Data type: 3 integers (day,hour,minute)
Valid range: 0-365 for day, 0-23 for hour, 0-59 for minute.

SECSTOP time must follow these rules:

- It must be a multiple of timestep STEP and greater than SECSTART time.
- It must be greater than or equal to START time, and less than or equal to STOP time.

Examples:

- SECSTOP = 81,0,0 ; Start saving secondary histories at model time 80,1,0

See also:

- *SECSTART*
- *SECHIST*

Back to top

SECHIST

Secondary history write frequency, specified as a model time (day,hour,minute). SECHIST time must divide evenly into SECSTOP minus SECSTART times.

Data type: 3 integers (day,hour,minute)
Valid range: 0-365 for day, 0-23 for hour, 0-59 for minute.

Examples:

- SECHIST = 0,1,0 ;request hourly histories
- SECHIST = 0,0,12 ;request 12-minute histories

See also:

- *HIST*

Back to top

SECOUT

List of secondary history output files. Secondary histories store diagnostic fields, usually at a higher temporal resolution than primary files. Each file may be an absolute path, or relative to the execution directory. Beware that SECOUT will overwrite any pre-existing files with the same names. As each SECOUT file is filled (see MXHIST_SECH), the next SECOUT file is created and histories are written until it is full, and so on.

SECOUT files are usually specified with increasing integers imbedded in the names. See examples below. As a convenience, large sequences of files may be specified in a "short-form", see example 3 below specifying 20 files. By convention, secondary history output files may use the letter "s" to indicate secondary file series (see all 3 examples below).

Examples:

```
SECOUT = 's_myoutput_001.nc'
SECOUT = 'myoutput.s001.nc','myoutput.s002.nc','myoutput.s003.nc'
SECOUT = 'myoutput_s001.nc','to','myoutput_s020.nc','by','1'
```

See also:

- *OUTPUT*
- *SOURCE*
- *MXHIST_SECH*

Back to top

SOURCE

SOURCE is the start-up history file for an initial run. SOURCE may be a full path or relative to the execution directory. It must be a TIEGCM history with the same grid resolution as the model being run. It does not need to be from the same model version as that being run.

If SOURCE is specified, then SOURCE_START, the model time of the history to read on the SOURCE file, must also be specified. The code will search the SOURCE file for the SOURCE_START history. If SOURCE is *not* specified, then the run is a continuation run, and the source history is provided in the first OUTPUT file at START time.

The SOURCE file must be on the local disk. The model will not look for the SOURCE history on any archive file system.

Examples:

- SOURCE = '$TGCMDATA/tiegcm_res2.5_decsol_smax_prim.nc'

See also:

- *SOURCE_START*

Back to top

SOURCE_START

This is the model time of the history to read from the SOURCE file. Model time is specified as a 3-integer triplet: day,hour,minute. If SOURCE is specified, then SOURCE_START must also be specified. If the SOURCE_START history is not found on the SOURCE file, the model will stop and print an appropriate error message to stdout.

Data type: 3 integers
Valid range: 0-365 for day, 0-23 for hour, 0-59 for minute.

Example:

- SOURCE_START = 80,0,0

See also:

- *SOURCE*

Back to top

START

Model time for start of the run. Model time is a 3-integer triplet: day,hour, minute. If CALENDAR_ADVANCE=0, then START day can be any number between 0 and 365. If CALENDAR_ADVANCE=1,

then START day must be the same as START_DAY. If an initial run, START time does not have to be the same as SOURCE_START.

Data type: 3 integers Valid range: 0-365 for day, 0-23 for hour, 0-59 for minute.

Examples:

- START = 80,0,0

See also:

- *SOURCE_START*

Back to top

START_DAY
Calendar starting day.

Data type: integer
Default: 80
Valid range: 1 to 365

Back to top

START_YEAR
Starting year for the run.

Data type: integer
Default: 2002

Back to top

STEP
Model time-step in seconds. Default value is 60 seconds for 5-degree resolution, 30 seconds for 2.5-degree resolution. During periods of quiet solar activity, the model can often be run at twice these times. During periods of intense solar activity (e.g., F10.7 > 200, or high magnitude BZ southward), the model may become numerically unstable. In this case, reducing the timestep to as low as 10 seconds may be necessary for the model get through the rough period.

Data type: integer
Default for 5.0-degree resolution: STEP=60
Default for 2.5-degree resolution: STEP=30

Back to top

STOP
Model stop time for the run. Model time is specified as a 3-integer triplet: day,hour,minute.

Data type: 3 integers
Valid range: 0-365 for day, 0-23 for hour, 0-59 for minute.

Example:

- STOP = 81,0,0

Back to top

SWDEN or SWDEN_TIME

Solar Wind Density (#/cm-3). Can be specified as either a constant (SWDEN), or series of time-dependent values (SWDEN_TIME). If IMF_NCFILE is set and SWDEN is not provided, then it SWDEN will be taken from the IMF data file.

Data type: real or real array

Examples:

- SWDEN = 4.0 ; constant for entire run
- SWDEN_TIME = 80,0,0,2., 80,1,0,3., 80,2,0,4. ; time series

See also:

- *IMF_NCFILE*

Back to top

SWVEL or SWVEL_TIME

Solar Wind Velocity (Km/s). Can be specified as either a constant (SWVEL), or series of time-dependent values (SWVEL_TIME). If IMF_NCFILE is set and SWVEL is not provided, then it SWVEL will be taken from the IMF data file.

Data type: real or real array

Examples:

- SWVEL = 400. ; constant for entire run
- SWVEL_TIME = 80,0,0,100., 80,1,0,200., 80,2,0,300. ; time series

See also:

- *IMF_NCFILE*

Back to top

TIDE

Hough mode amplitudes and phases of the semi-diurnal tide. If GSWM tidal perturbations are specified, TIDE should be set to 0.

Note: TIDE and TIDE2 should be specified only for experiments where amplitude and phases of the tides must be used. Normally, GSWM tides are specified instead of TIDE,TIDE2.

Data type: 10 reals

Example:

```
TIDE= 1.9300E+04, 1.5000E+04, 2.3100E+04, 0.7700E+04, 0.1660E+04,
      -2.600E+00,  0.000E+00, -3.300E+00, 4.2000E+00, 5.0000E+00
```

See also:

- *GSWM_MI_SDI_NCFILE*
- *GSWM_NM_SDI_NCFILE*

Back to top

TIDE2

Hough mode amplitudes and phases of the diurnal tide. If GSWM tidal perturbations are specified, TIDE2 should be set to 0.

Note: TIDE and TIDE2 should be specified only for experiments where amplitude and phases of the tides must be used. Normally, GSWM tides are specified instead of TIDE,TIDE2.

Data type: 2 floats

Example:

```
TIDE2 = 4.1E+4, -3.7
```

See also:

- *GSWM_MI_DI_NCFILE*
- *GSWM_NM_DI_NCFILE*

Back to top

CHAPTER

SIX

USING THE JOB SCRIPTS TO SET UP AND SUBMIT A MODEL RUN

6.1 The Linux desktop job script (tiegcm-linux.job)

Take a look at the default Linux job script `tiegcm-linux.job`. Near the top are several shell variables, with their default settings, which configure the job script (variables and values may vary between model versions):

```
set modeldir = tiegcm_trunk
set execdir  = /hao/aim/$user/tiegcm_trunk/tiegcm.exec
set tgcmdata = /hao/aim/tgcm/data/tiegcm2.0
set input    = $modeldir/scripts/tiegcm_res5.0_default.inp
set output   = tiegcm.out
set make     = Make.intel_hao64
set modelres = 5.0
set mpi      = TRUE  # must be TRUE for tiegcm2.0 and later
set nproc    = 4
set debug    = FALSE
set exec     = TRUE
set utildir  = $modeldir/scripts
```

Following are brief explanations of the job script shell variables:

Note: Absolute or relative paths are acceptable when specifying directories. Relative paths should be relative to the *working directory* (*workdir*). In practice, modeldir is usually relative to the working directory, and execdir and tgcmdata are usually absolute paths.

---

modeldir
: The model root directory (*modeldir* from the source code download). The example above assumes the user has checked out the trunk revision as "tiegcm_trunk". This directory contains subdirectories *src/* , *scripts/* , *doc/* , *tgcmrun/*, and *benchmarks/*.

execdir
: This is the execution directory (*execdir*), in which the model will be built and executed. It will be created if it does not already exist. It is typically on a large temporary disk. This directory will also contain the model output *netCDF* history files (see also *NetCDF History Output Files*)

tgcmdata
: Directory containing startup history files and data files for model input. It is normally on a large temporary disk. These files are available from the *data download tar file* (separate downloads for each model resolution). Note that setting *tgcmdata* in the job script is optional: if it is specified, it will override any setting of the *TGCMDATA* environment variable. If it is not specified, the job script will use the TGCMDATA environment variable. If neither are set, tgcmdata will default to the current working directory.

input
: The *namelist input file*. The default namelist file is in the scripts directory under the model root with file

name `tiegcm_res5.0_default.inp` (for 5-degree resolution), or `tiegcm_res2.5_default.inp` (for 2.5-degree resolution). The default input file can be copied to the working directory, modified, and renamed for your own runs. In that case, be sure to reset the input file in the job script.

make

Make file containing platform-specific compiler flags, library locations, etc. If not otherwise specified with a path, the job script will look for this file in the *scripts/* directory. This file is included in the main Makefile (scripts/Makefile). The user can either make necessary adjustments to an existing make file, or write their own for a different platform/compiler system.

There are three such makefiles available in the *scripts/* directory for the Linux desktop platform:

- `Make.intel_hao64` (for Intel compiler)
- `Make.pgi_hao64` (for PGI compiler)
- `Make.gfort_hao64` (for gfortran compiler)

You will need to set the paths to your local netCDF and *ESMF* libraries in these makefiles.

output

Name of the file to receive stdout *output* from the model. If this pre-exists, it will be overwritten when the model is executed. Here is an example stdout file from the root mpi task of a 4-processor run (5-degree resolution) on a Linux desktop machine: `tiegcm_task0000.out`

mpi

Logical flag indicating whether or not to link the MPI library for a multi-processor parallel run.

> **Warning:** For tiegcm versions 2.0 and later, non-MPI runs (mpi=FALSE) are NOT supported. However, mpi runs (mpi=TRUE) with a single processor (nproc=1) ARE supported.

nproc

Number of processors to use in parallel execution. This will be the number of MPI tasks made available for the domain decomposition. On linux desktops, this is typically 4. For tiegcm on linux supercomputer clusters (e.g., the NCAR yellowstone system, where there are 16 processors per node), the recommended number is 16 for 5.0-degree resolution, or 64 for 2.5-degree resolution. For debug purposes, nproc=1 is supported. The models have been tested with the following processor counts: 1,4,8,12,16,24,32,48,64,72,80. See `performance table` for performance estimates at recommended processor counts and timesteps.

modelres

Model resolution. Two resolutions are supported:

- modelres = 5.0 sets 5-degree lat x lon horizontal, and dz=0.50 vertical
- modelres = 2.5 sets 2.5-degree lat x lon horizontal, and dz=0.25 vertical

If the resolution is changed, the model should be recompiled before re-executing the job script (type "*gmake clean*" in the *execdir*.

For more information, see *Grid Structure and Resolution*.

debug

If debug = TRUE, the job script will compile the build with debug flags set. Debug flags specific to the compiler are set in the make file. If debug is changed, the code should be recompiled (type "gmake clean" in the *execdir* before re-executing the job script).

exec

If exec = TRUE, the job script will execute the model after compilation, otherwise, the job script will stop after compilation without execution.

utildir
: The utility directory containing supporting scripts. This is normally the *scripts/* subdirectory in the model root directory *modeldir*

You are now ready to build and execute a default run. To do this, simply execute the job script as follows:

```
$ tiegcm-linux.job &
```

The compilation output will be displayed. If the build is successful (and exec=TRUE), the model will be executed, and stdout will go to the specified *output* file. If the job is successful, you can edit and rename the namelist input file, reset *namelist input file* in the job script, and re-execute the job script. If there has been no change to the source code, it will not need to recompile, and will use the pre-existing executable.

6.2 The yellowstone supercomputer job script (tiegcm-ys.job)

Note: This section contains information that is specific to user's of the NCAR Linux Supercomputer yellowstone:

Linux yslogin3 2.6.32-358.el6.x86_64 #1 SMP
Tue Jan 29 11:47:41 EST 2013 x86_64 x86_64 x86_64 GNU/Linux

For more information about the NCAR yellowstone system, see https://www2.cisl.ucar.edu/resources/yellowstone

The model can be built and executed on yellowstone using the Intel compiler and the intelmpi implementation. To do this, copy and modify the job script tiegcm-ys.job from the scripts directory.

The yellowstone job script `tiegcm-ys.job` has the same user-settable shell variables as the Linux job script, but the default settings are slightly different:

```
set modeldir = tiegcm_trunk
set execdir  = /glade/scratch/$user/tiegcm_trunk/tiegcm.exec
set tgcmdata = /glade/p/hao/tgcm/data/tiegcm2.0
set input    = $modeldir/scripts/tiegcm_res5.0_default.inp
set output   = tiegcm.out
set modelres = 5.0
set make     = Make.intel_ys
set mpi       = TRUE   # must be TRUE for tiegcm2.0 and later
set debug     = FALSE
set exec      = TRUE
set utildir   = $modeldir/scripts
set runscript = run.lsf
```

In this example, it is assumed the user has checked-out the trunk revision as "tiegcm_trunk". Note the *execdir* name, and the `make` file `Make.intel_ys`. The model *resolution* in this case is 5.0 degrees.

Also note the special "#BSUB" directives near the top of the yellowstone job script (descriptions in the right-hand column are for this document only, and are not in the script itself):

```
#BSUB -J tiegcm                # job name
#BSUB -P P28100036             # authorized project number
#BSUB -q premium               # premium queue
#BSUB -o tiegcm.%J.out         # stdout file
#BSUB -e tiegcm.%J.out         # stderr file
#BSUB -N
#BSUB -u $LOGNAME@ucar.edu     # email notification address
```

```
#BSUB -W 1:00                  # wallclock limit hours:minutes
#BSUB -n 16                    # number of processors (mpi tasks)
#BSUB -R "span[ptile=16]"      # use 16 processors per node
```

These are resource settings for the Load Sharing Facility (LSF), the batch queuing system sold by Platform Computing. The LSF is used for scheduling jobs on the yellowstone system at NCAR. This job will be submitted to the premium queue command, requesting 16 processors with a wallclock limit of 1 hour.

To submit the yellowstone job, simply execute the job script on the command line. It will build the model on the interactive node, and if successful, the runscript (run.lsf by default) will be created and submitted to the LSF via the bsub command.

Watch the progress of your LSF job with the command:

```
$ bjobs
```

You can kill a LSF job with this command:

```
$ bkill job_ID
```

Where `job_ID` is the job identifier given in the `bjobs` command.

For more information about the LSF, see the Wikipedia site:

http://en.wikipedia.org/wiki/Platform_LSF

or the Platform Computing site:

http://www.platform.com/workload-management/high-performance-computing/lp

CHAPTER

SEVEN

MODEL OUTPUT HISTORY FILES

7.1 NetCDF History Output Files

TIEGCM history files are output in netCDF, a self-describing platform-independent data format written and maintained by the UCAR Unidata program.

Each netCDF file contains one or more *histories*, i.e., the state of the model output fields at a discrete instant in *model time*. Here is an example of the metadata content of a sample primary history file: `primary.ncd`. This example file contains six daily histories (days 355 to 360 of 2002). This metadata is obtained via the "ncdump" command in the netCDF utilities. This example ncdump file contains data values for scalar and singly-dimensioned vectors only. You can also use the tgcm_ncdump script in the *scripts/* directory.

TIEGCM history files are "CF compliant", i.e., they conform to the NetCDF Climate and Forecast (CF) Metadata Convention.

7.2 Primary and Secondary History Files

"Primary" history files contain the "prognostic" fields necessary to restart the model. They can be specified in namelist input as the *SOURCE* file for starting the model in an *initial run* (a *continuation run* does not specify a SOURCE file, and is continued from the START time found on one of the files in the OUTPUT file list). Typically, daily histories are stored on primary history files.

Fields on primary histories necessary for start-up of the TIEGCM are as follows: TN, UN, VN, O2, O1, N4S, NO, HE, AR, OP, N2D, TI, TE, NE, O2P, OMEGA, Z, POTEN (see the below table). There are also several fields with the suffix "_NM" that contain data from the previous timestep, necessary for the job-stepping scheme for prognostic variables. Lower boundary conditions for neutral temperature and winds are also on the primary history (TLBC, ULBC and VLBC).

"Secondary" history files contain diagnostic fields and/or primary history fields. Fields to be saved on the secondary history files are listed by the namelist input parameter *SECFLDS*. Diagnostics can be saved by calling subroutine addfld in the code (addfld.F), or by including one or more of the "standard" diagnostic fields available via the *diagnostics* module (diags.F). Secondary histories are often saved at a higher temporal resolution than primary histories, typically hourly. Here is an ncdump of an `example secondary history file`

7.3 Table of Primary History Fields

This is a table of prognostic fields that are saved on primary histories. These fields are necessary for restarting the model. They can also be saved on secondary histories by adding the short name to the *SECFLDS* field list in the namelist input file. Here is an ncdump of the example primary file: `primary.ncd`.

Short Name	Long Name	Units
TN	NEUTRAL TEMPERATURE	deg K
UN	NEUTRAL ZONAL WIND (+EAST)	cm/s
VN	NEUTRAL MERIDIONAL WIND (+NORTH)	cm/s
O2	MOLECULAR OXYGEN	mmr
O1	ATOMIC OXYGEN	mmr
N4S	N4S	mmr
NO	NITRIC OXIDE	mmr
HE	HELIUM	mmr
AR	ARGON	mmr
OP	O+ ION	mmr
N2D	N2D	mmr
TI	ION TEMPERATURE	deg K
TE	ELECTRON TEMPERATURE	deg K
NE	ELECTRON DENSITY	cm-3
O2P	O2+ ION	cm-3
OMEGA	VERTICAL MOTION	s-1
Z	GEOPOTENTIAL HEIGHT	cm
POTEN	ELECTRIC POTENTIAL	volts

As mentioned above, several of these fields are also required at the previous model timestep, for the leap-frog time-stepping scheme. These fields have the suffix '_NM', added to their names, e.g., TN_NM, UN_NM, VN_NM, etc.

Aurora Characteristic Energy' .. index:: diagnostic fields

CHAPTER

EIGHT

SAVING DIAGNOSTIC FIELDS

The diagnostics module (`diags.F`) in the TIEGCM will calculate and save *diagnostic fields* to the secondary histories. The user can add any subset of these fields to the *SECFLDS* parameter list in the namelist input file. See the `diagnostics namelist example`.

8.1 Table of Available Diagnostics

Following is a table of diagnostic fields that can be saved on secondary histories by including the short names in the *SECFLDS* namelist input parameter. Click on the short name to obtain detailed information about the calculation and saving of a diagnostic field.

On the history files, "Short Name" will be the variable name, and "Long Name" and "Units" will be attributes of the variable. "Grid" refers to the number of dimensions (2d lat-lon, or 3d lat-lon-level), and whether the field is on the geographic or geomagnetic grid.

A `text version of the table` is also available, and is printed to stdout by a model run (ordering of the fields in the text table may be different than in the below table).

Also note that prognostic variables that are saved on the primary histories can also be saved on secondary histories by adding the short name to SECFLDS. See *Primary History Fields* for a table of the prognostic fields.

Short Name	Long Name	Units	Grid
CO2_COOL	CO2 Cooling	erg/g/s	3d geo
NO_COOL	NO Cooling	erg/g/s	3d geo
DEN	Total Neutral Density	g/cm3	3d geo
HEATING	Total Heating	erg/g/s	3d geo
SCHT	Pressure Scale Height	km	3d geo
SIGMA_HAL	Hall Conductivity	S/m	3d geo
SIGMA_PED	Pedersen Conductivity	S/m	3d geo
LAMDA_HAL	Hall Ion Drag Coefficient	1/s	3d geo
LAMDA_PED	Pedersen Ion Drag Coefficient	1/s	3d geo
UI_ExB	Zonal Ion Drift	cm/s	3d geo
VI_ExB	Meridional Ion Drift	cm/s	3d geo
WI_ExB	Vertical Ion Drift	cm/s	3d geo
MU_M	Molecular Viscosity Coefficient	g/cm/s	3d geo
WN	Neutral Vertical Wind	cm/s	3d geo
O_N2	O/N2 Ratio	[none]	3d geo
QJOULE	Joule Heating	erg/g/s	3d geo
QJOULE_INTEG	Height-integrated Joule Heating	erg/cm2/s	2d geo
HMF2	Height of the F2 Layer	km	2d geo
NMF2	Peak Density of the F2 Layer	1/cm3	2d geo

Continued on next page

Table 8.1 – continued from previous page

Short Name	Long Name	Units	Grid
FOF2	Critical Frequency of F2 Layer	MHz	2d geo
TEC	Total Electron Content	1/cm2	2d geo
JE13D	Eastward current density (3d)	A/m2	3d mag
JE23D	Downward current density (3d)	A/m2	3d mag
JQR	Upward current density (2d)	A/m2	2d mag
KQLAM	Height-integ current density (+north)	A/m	2d mag
KQPHI	Height-integ current density (+east)	A/m	2d mag
BX	BX/BMAG eastward electric field	[none]	2d mag
BY	BY/BMAG northward electric field	[none]	2d mag
BZ	BZ/BMAG upward electric field	[none]	2d mag
BMAG	Magnetic field magnitude	Gauss	2d mag
EX	Zonal component of electric field	V/m	3d geo
EY	Meridional component of electric field	V/m	3d geo
EZ	Vertical component of electric field	V/m	3d geo
ED1	Mag eastward component electric field	V/m	3d mag
ED2	Mag downward component electric field	V/m	3d mag
PHIM2D	2d Electric Potential on magnetic grid	V	2d mag
N2	Molecular Nitrogen	mmr	3d geo
ZGMID	Geometric Height at midpoints	cm	3d geo
CUSP	Cusp Low Energy Electron Flux	erg/cm2/s	2d geo
DRIZZLE	Drizzle Low Energy Electron Flux	erg/cm2/s	2d geo
ALFA	Aurora Characteristic Energy	keV	2d geo
NFLUX	Aurora Number Flux	#/cm2/s	2d geo
EFLUX	Aurora Energy Flux	erg/cm2/s	2d geo

8.2 Saving Fields/Arrays from the Source Code

In addition to the "sanctioned" diagnostics above, arbitrary 2d and 3d arrays can be saved from the model to secondary histories by inserting a call to subroutine *addfld* (`addfld.F`) in the source code. (See the chapter on *Modifying Source Code* in this document for information about modifying the source code.) There are hundreds of examples of this in the source code, some commented, others uncommented. To put these on secondary histories, uncomment the addfld call if necessary, and add the short name (first argument) to the secondary history field list (SECFLDS) in the namelist input file. For more information about how to make calls to addfld, please see comments in the addfld.F source file.

Here are a couple of examples of addfld calls from near the end of subroutine qrj (qrj.F). These calls are inside a latitude loop, where the loop variable index is "lat". Normally, in parallel code, subdomains of the field are passed, e.g., lon0:lon1 and lat0:lat1:

```
 call addfld('QO2P'  ,' ',' ',  qo2p(lev0:lev1,lon0:lon1,lat),
|  'lev',lev0,lev1,'lon',lon0,lon1,lat)
 call addfld('QN2P'  ,' ',' ',  qn2p(lev0:lev1,lon0:lon1,lat),
|  'lev',lev0,lev1,'lon',lon0,lon1,lat)
 call addfld('QNP'   ,' ',' ',   qnp(lev0:lev1,lon0:lon1,lat),
|  'lev',lev0,lev1,'lon',lon0,lon1,lat)
```

The calling sequence for subroutine addfld is explained in comments at the top of source file `addfld.F`.

8.3 Details of Diagnostic Field Calculations

CO2_COOL

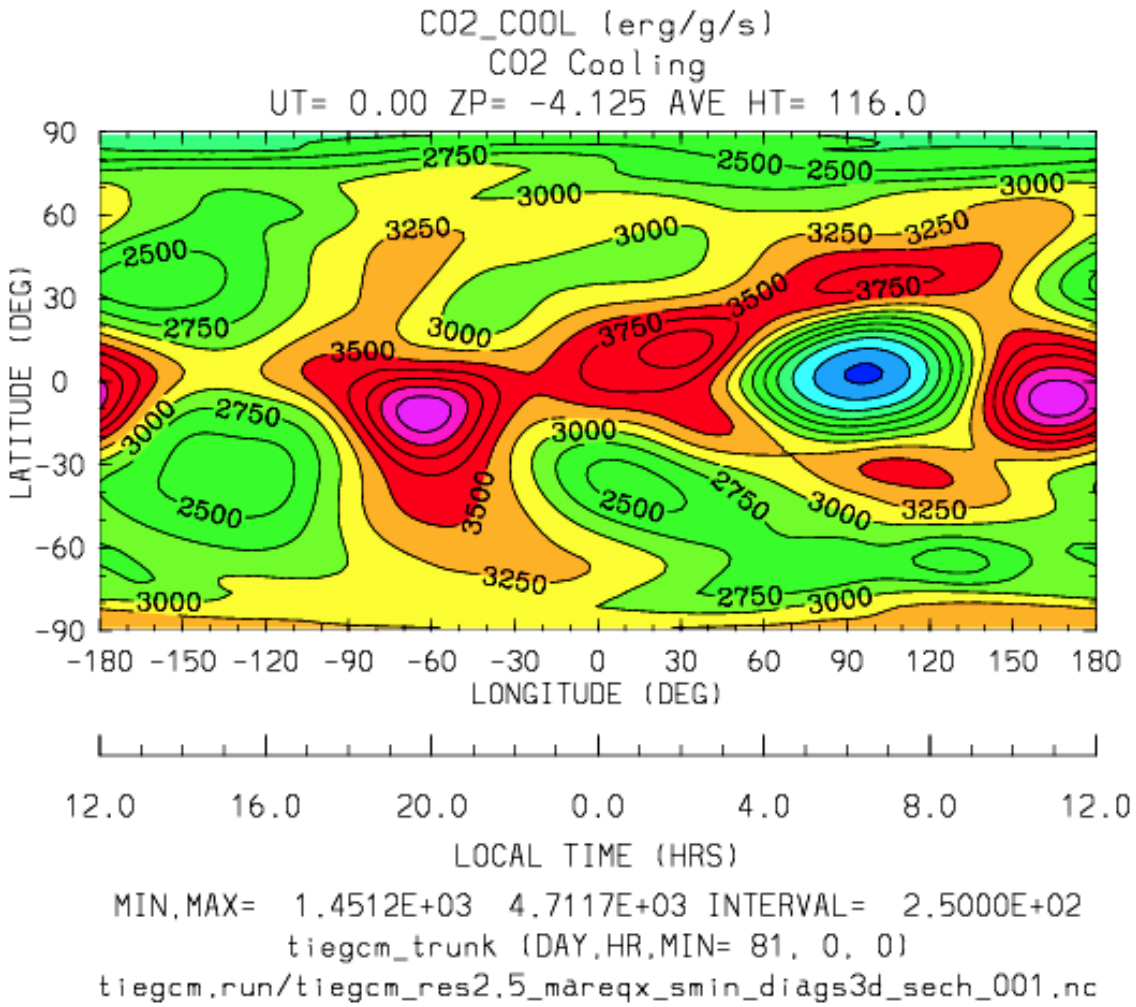
Diagnostic field: CO2 Cooling (erg/g/s):

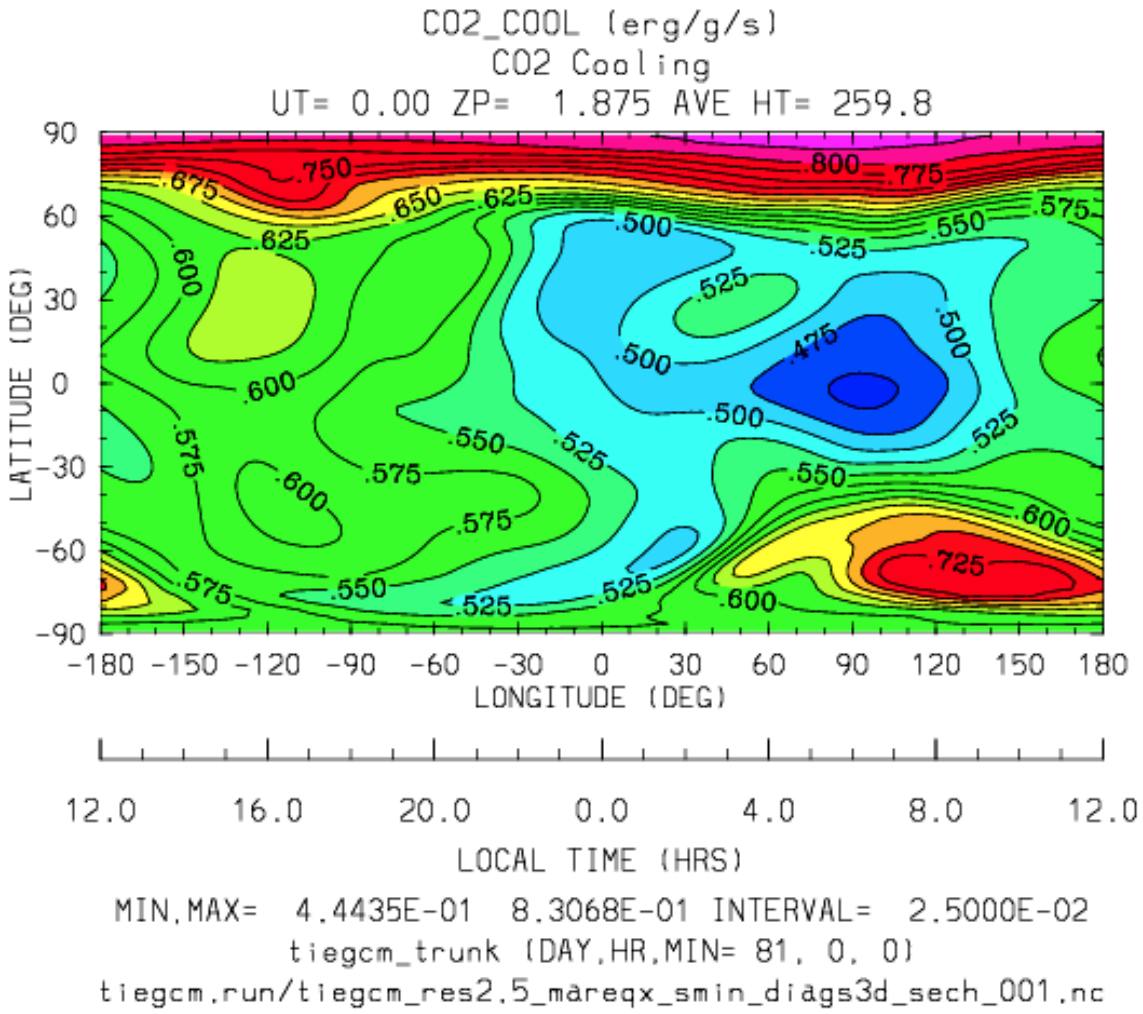
```
diags(n)%short_name = 'CO2_COOL'
diags(n)%long_name  = 'CO2 Cooling'
diags(n)%units      = 'erg/g/s'
diags(n)%levels     = 'lev'
diags(n)%caller     = 'newton.F'
```

This field is calculated in `newton.F` and passed to mkdiag_CO2COOL (`diags.F`), where it is saved to the secondary history. The calculation of CO2 cooling in newton.F is as follows:

```
      co2_cool(k,i) = 2.65e-13*nco2(k,i)*exp(-960./tn(k,i))*
|       avo*((o2(k,i)*rmassinv_o2+(1.-o2(k,i)-o1(k,i))*rmassinv_n2)*
|       aco2(k,i)+o1(k,i)*rmassinv_o1*bco2(k,i))
```

Sample images: CO2_COOL Global maps at Zp -4, +2:





Back to diagnostics table

NO_COOL

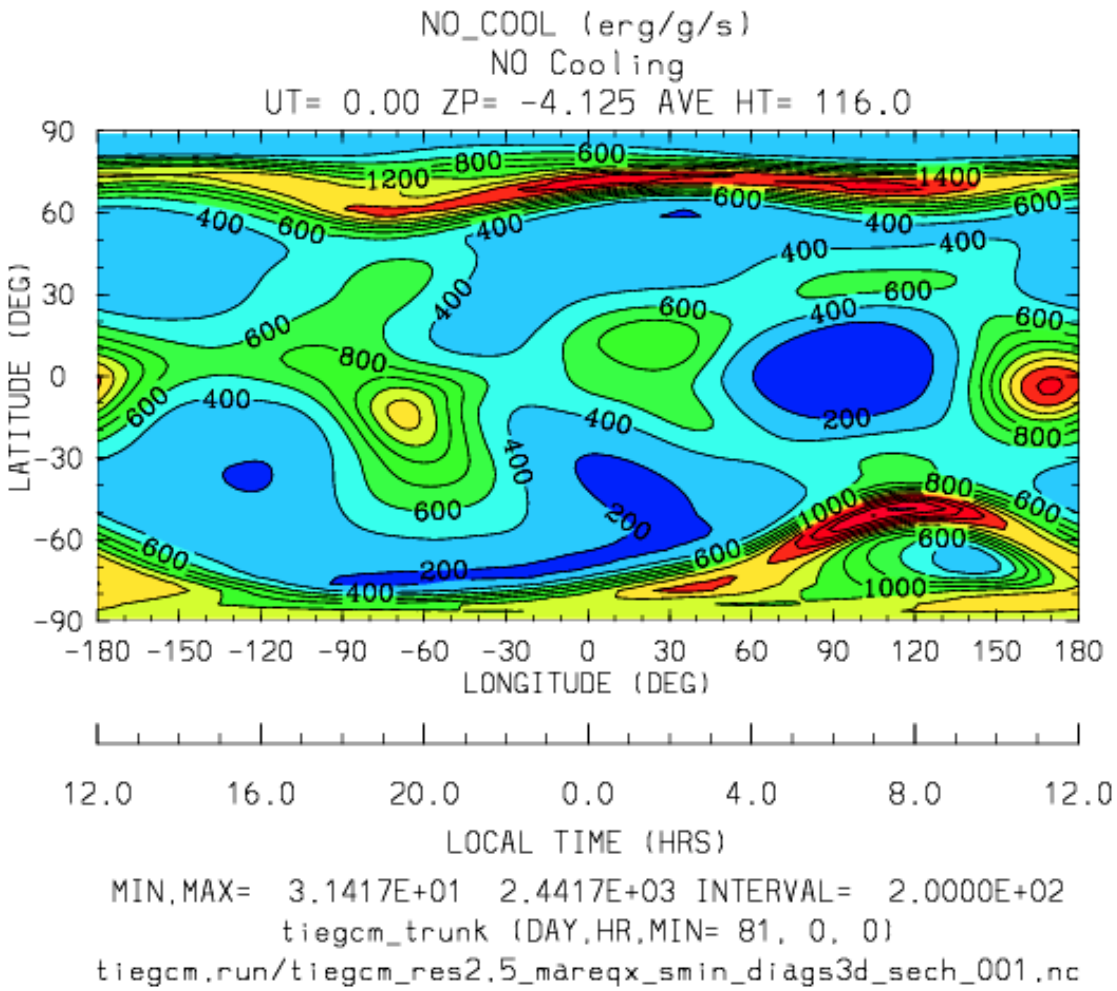
Diagnostic field: NO Cooling (erg/g/s):

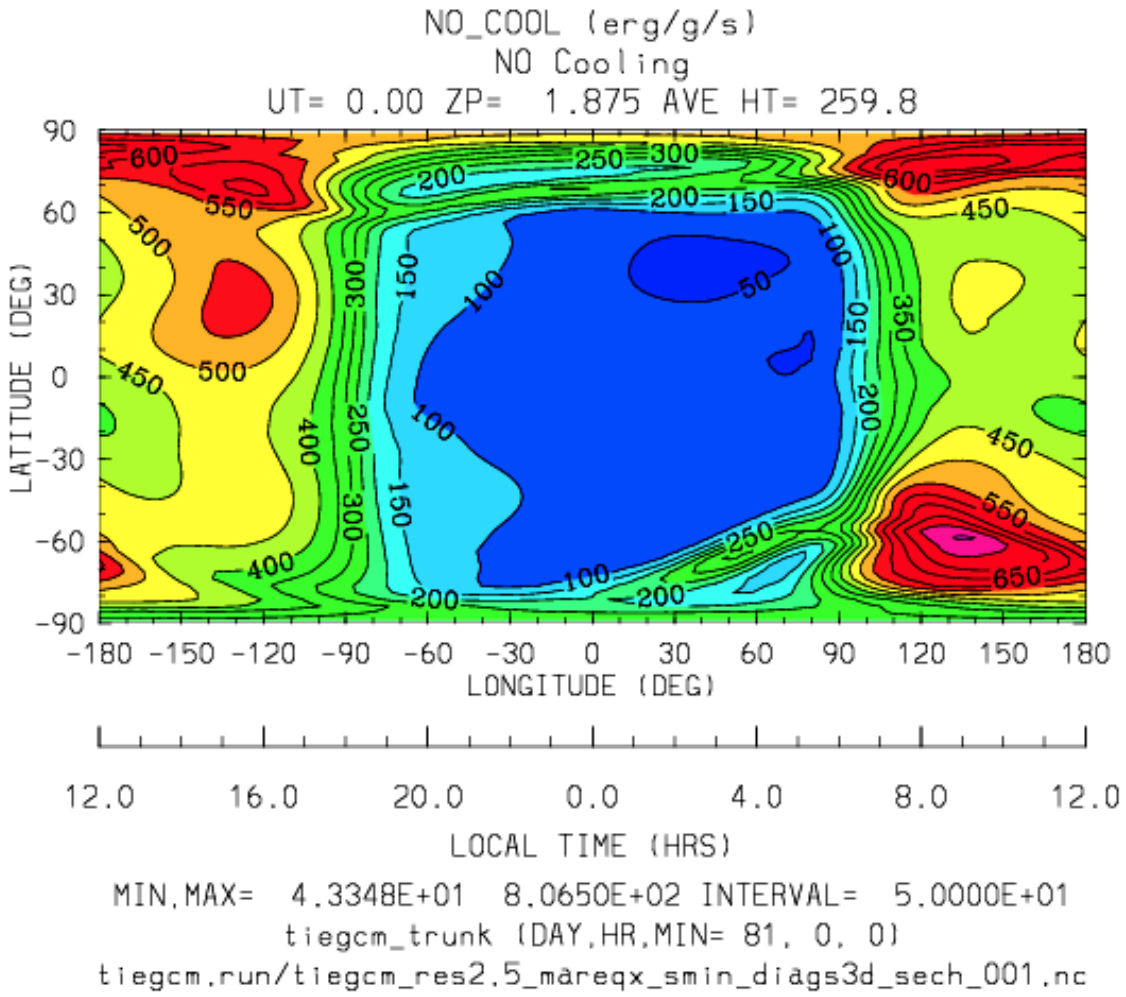
```
diags(n)%short_name = 'NO_COOL'
diags(n)%long_name  = 'NO Cooling'
diags(n)%units      = 'erg/g/s'
diags(n)%levels     = 'lev'
diags(n)%caller     = 'newton.F'
```

This field is calculated in `newton.F` and passed to mkdiag_NOCOOL (`diags.F`), where it is saved to the secondary history. The calculation of NO cooling in newton.F is as follows:

```
      no_cool(k,i) = 4.956e-12*(avo*no(k,i)*rmassinv_no)*
|       (ano(k,i)/(ano(k,i)+13.3))*exp(-2700./tn(k,i))
```

Sample images: NO_COOL Global maps at Zp -4, +2:


NO_COOL (erg/g/s)
NO Cooling
UT= 0.00 ZP= -4.125 AVE HT= 116.0
LATITUDE (DEG)
LONGITUDE (DEG)
LOCAL TIME (HRS)
MIN,MAX= 3.1417E+01 2.4417E+03 INTERVAL= 2.0000E+02
tiegcm_trunk (DAY,HR,MIN= 81, 0, 0)
tiegcm.run/tiegcm_res2.5_mareqx_smin_diags3d_sech_001.nc



Back to diagnostics table

DEN

Diagnostic field: Total Density (g/cm3):

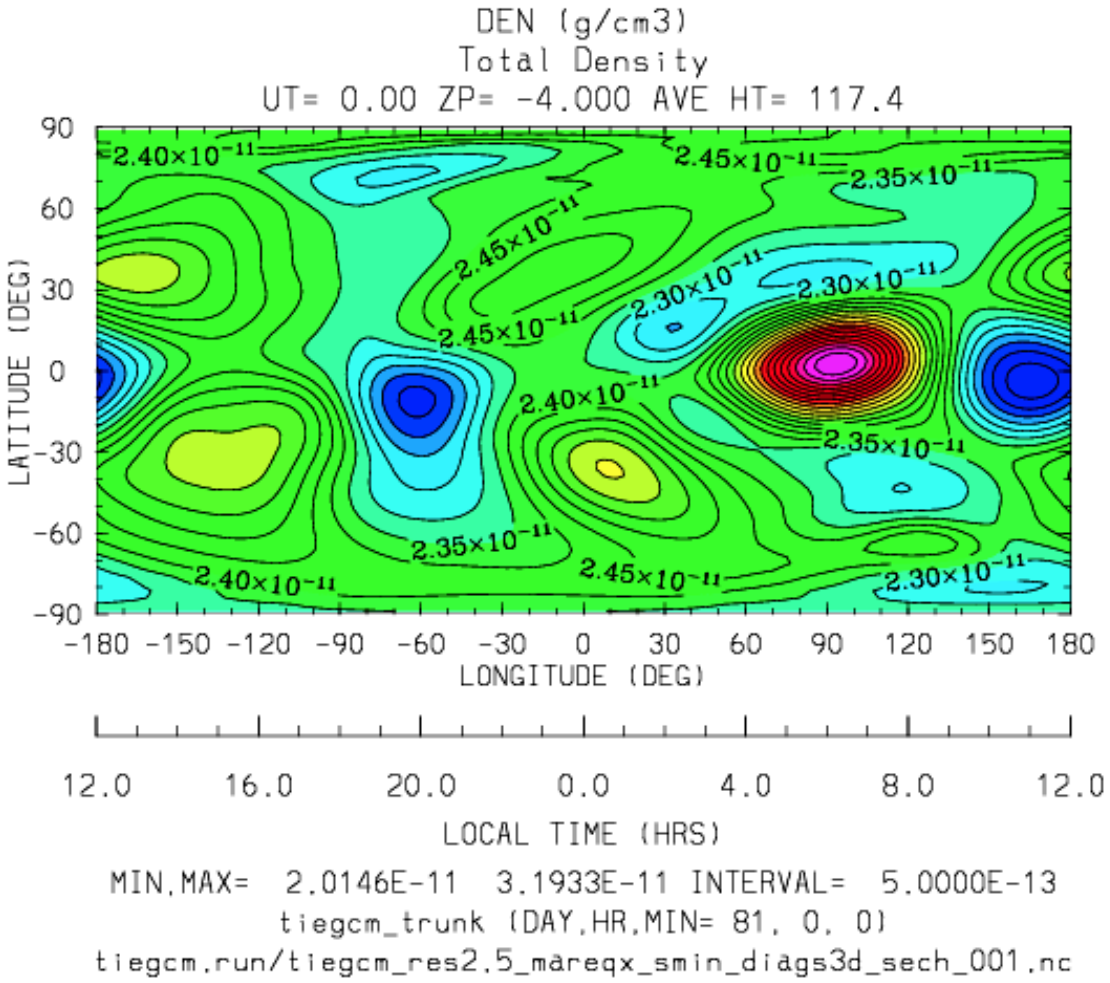
```
diags(n)%short_name = 'DEN'
diags(n)%long_name  = 'Total Density'
diags(n)%units      = 'g/cm3'
diags(n)%levels     = 'ilev'
diags(n)%caller     = 'dt.F'
```

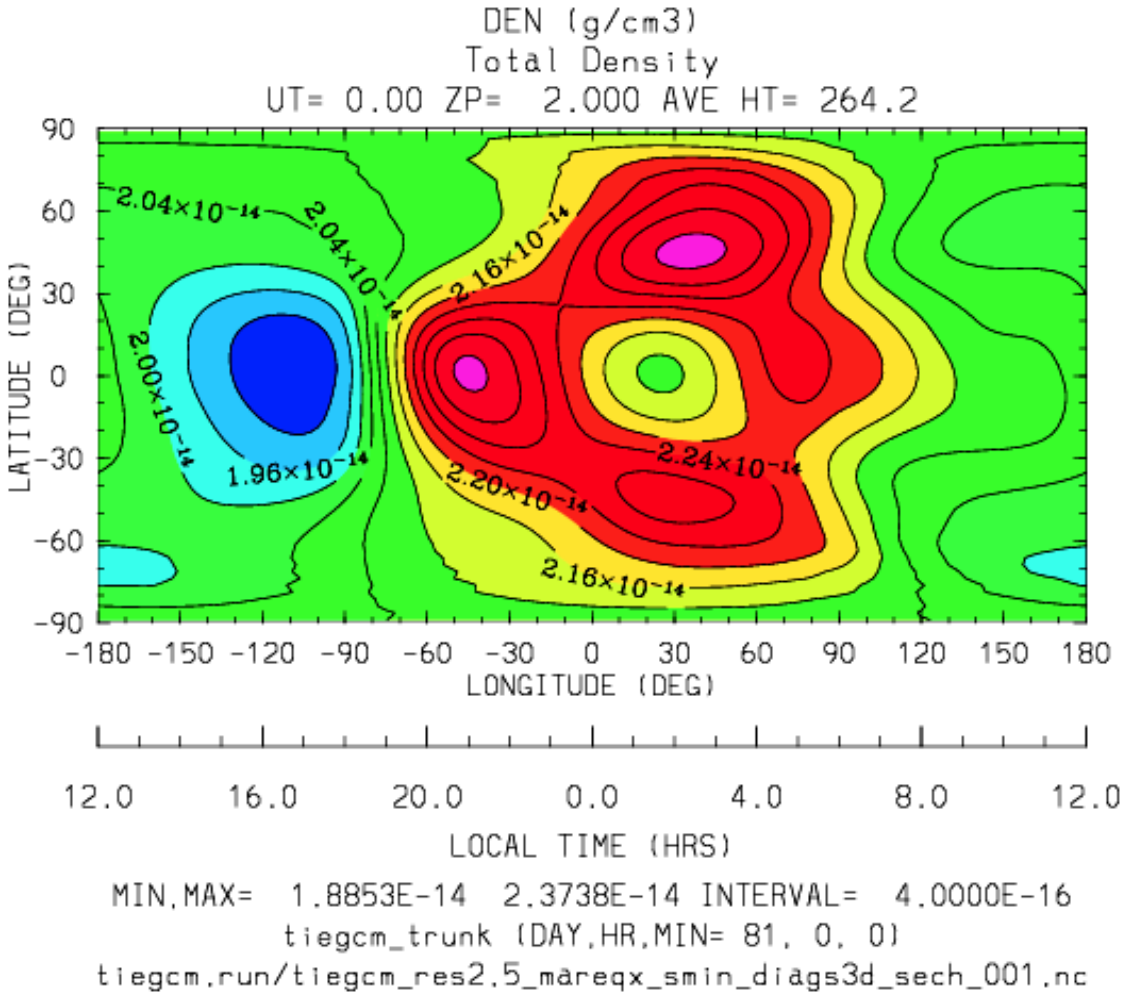
This field is calculated in `dt.F` and passed to mkdiag_DEN (`diags.F`), where it is saved to the secondary history. The calculation of DEN (rho) in dt.F is as follows:

```
do i=lon0,lon1
  do k=lev0+1,lev1-1
    tni(k,i) = .5*(tn(k-1,i,lat)+tn(k,i,lat))
    h(k,i) = gask*tni(k,i)/barm(k,i,lat)
    rho(k,i) = p0*expzmid_inv*expz(k)/h(k,i)
  enddo ! k=lev0+1,lev1-1
```

```
  rho(lev0,i) = p0*expzmid_inv*expz(lev0)/h(lev0,i)
  rho(lev1,i) = p0*expzmid*expz(lev1-1)/h(lev1,i)
enddo ! i=lon0,lon1
```

Sample images: DEN Global maps at Zp -4, +2:





Back to diagnostics table

HEATING

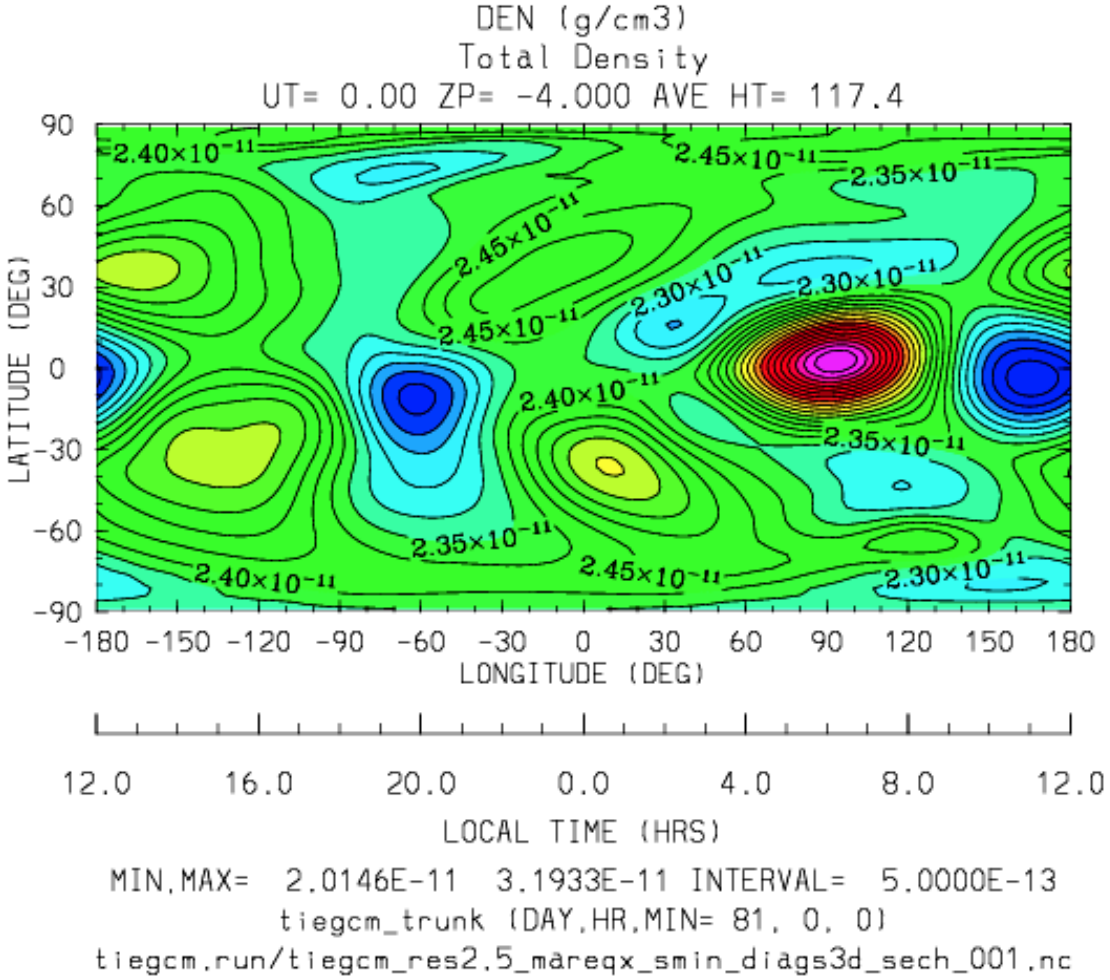
Diagnostic field: Total Heating (erg/g/s):

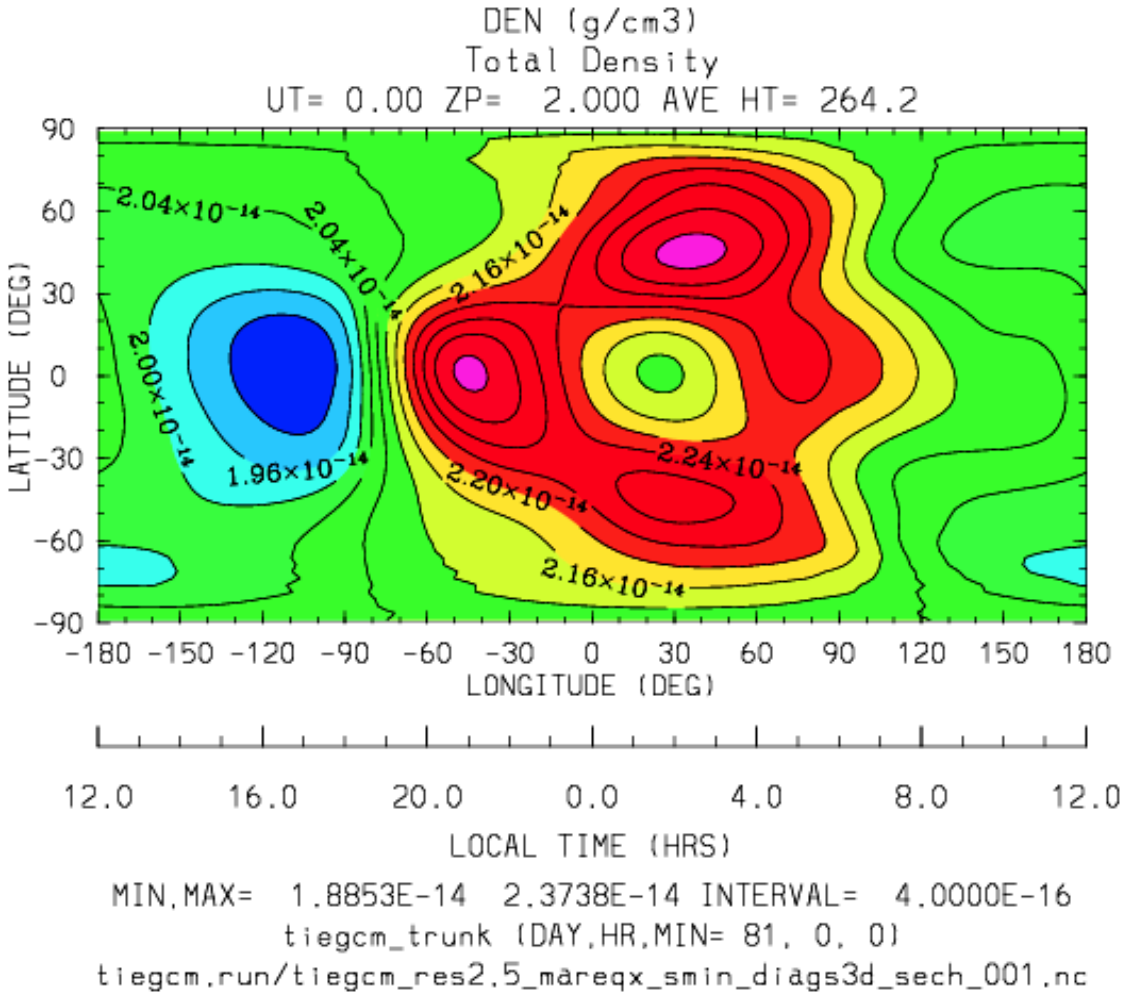
```
diags(n)%short_name = 'HEATING'
diags(n)%long_name  = 'Total Heating'
diags(n)%units      = 'erg/g/s'
diags(n)%levels     = 'lev'
diags(n)%caller     = 'dt.F'
```

This field is calculated in `dt.F` and passed to mkdiag_HEAT (`diags.F`), where it is saved to the secondary history. The calculation of HEATING (rho) in dt.F sums the following heat sources:

- Total solar heating (see *qtotal* in `qrj.F`)
- Heating from 4th order horizontal diffusion
- Heating due to atomic oxygen recombination
- Ion Joule heating
- Heating due to molecular diffusion

Sample images: HEATING Global maps at Zp -4, +2:





Back to diagnostics table

HMF2

Diagnostic field (2d lat x lon): Height of the F2 Layer (km):

```
diags(n)%short_name = 'HMF2'
diags(n)%long_name  = 'Height of the F2 Layer'
diags(n)%units      = 'km'
diags(n)%levels     = 'none' ! hmf2 is 2d lon x lat
diags(n)%caller     = 'elden.F'
```

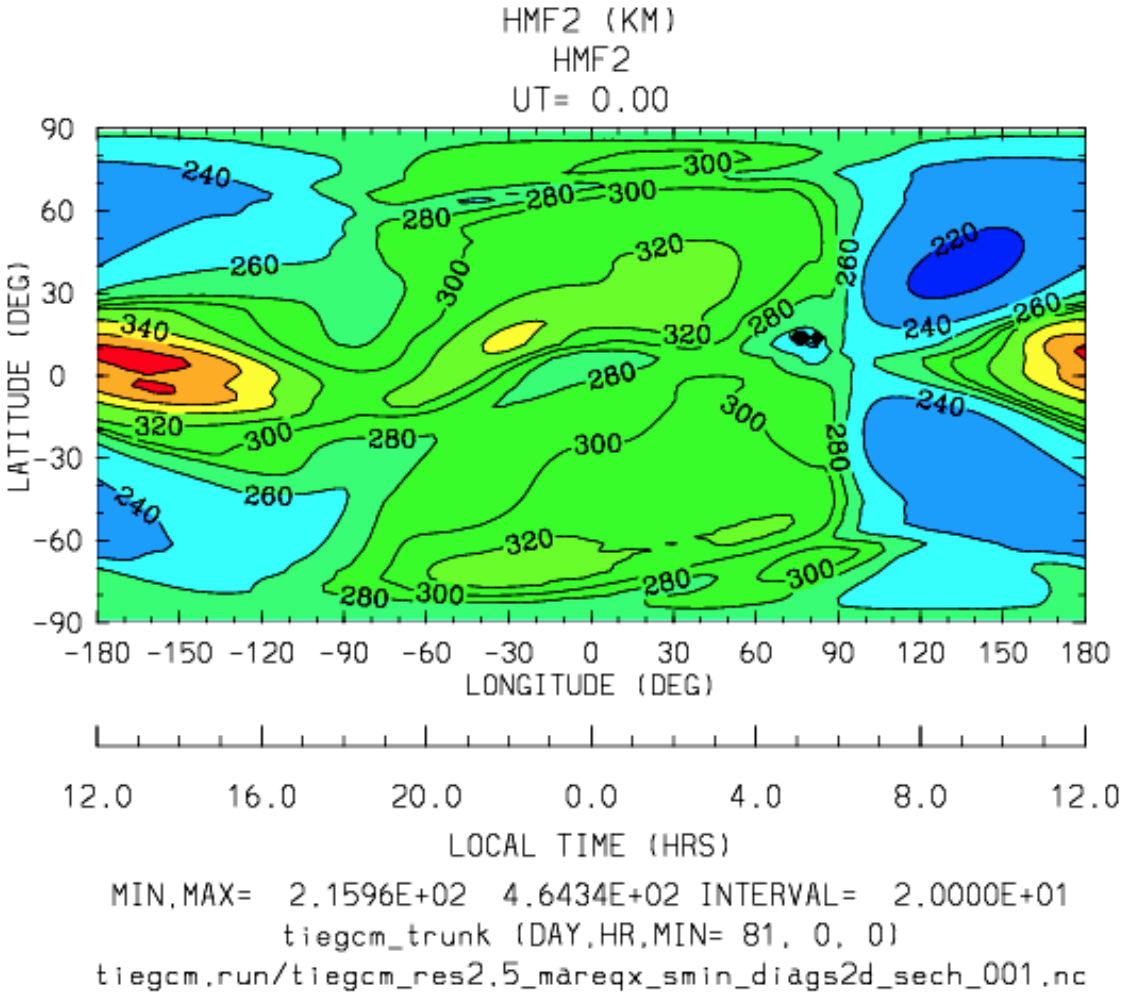
The height of the F2 layer is calculated and saved by subroutines *mkdiag_HNMF2* and *hnmf2* in source file `diags.F`.

Sub *mkdiag_HNMF2* is called by subroutine *elden* in source file `elden.F`, as follows:

call mkdiag_HNMF2('HMF2',z,electrons,lev0,lev1,lon0,lon1,lat)

Note: Occaisionally this algorithm will return the peak electron density in the E-region, instead of the F-region, in small areas of the global domain, usually at high latitide. This can result in pockets of anonymously low values for HMF2, e.g., around 125 km.

Sample images: HMF2 Global map:



Back to diagnostics table

NMF2

Diagnostic field (2d lat x lon): Peak Density of the F2 Layer (1/cm3):

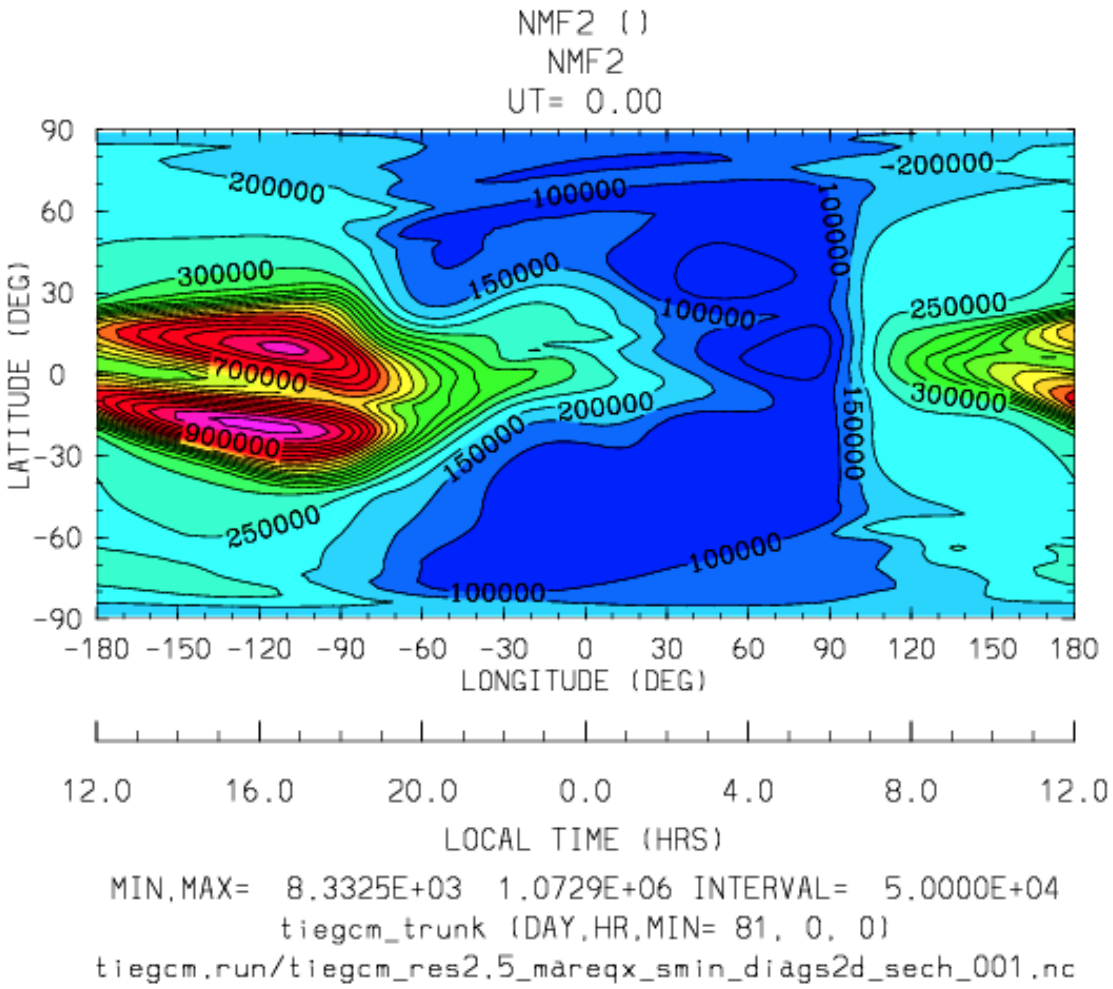
```
diags(n)%short_name = 'NMF2'
diags(n)%long_name  = 'Peak Density of the F2 Layer'
diags(n)%units      = '1/cm3'
diags(n)%levels     = 'none' ! nmf2 is 2d lon x lat
diags(n)%caller     = 'elden.F'
```

The peak density of the the F2 layer is calculated and saved by subroutines *mkdiag_HNMF2* and *hnmf2* in source file `diags.F`.

Sub *mkdiag_HNMF2* is called by subroutine *elden* in source file `elden.F`, as follows:

call mkdiag_HNMF2('NMF2',z,electrons,lev0,lev1,lon0,lon1,lat)

Sample images: NMF2 Global map:



Back to diagnostics table

FOF2

Diagnostic field (2d lat x lon): Peak Density of the F2 Layer (1/cm3):

```
diags(n)%short_name = 'FOF2'
diags(n)%long_name  = 'Critical Frequency of the F2 Layer'
diags(n)%units      = 'Mhz'
diags(n)%levels     = 'none' ! fof2 is 2d lon x lat
diags(n)%caller     = 'elden.F'
```

The critical frequency of the the F2 layer is calculated and saved by subroutines *mkdiag_HNMF2* and *hnmf2* in source file `diags.F`.

Sub *mkdiag_HNMF2* is called by subroutine *elden* in source file `elden.F`, as follows:

```
call mkdiag_HNMF2('FOF2',z,electrons,lev0,lev1,lon0,lon1,lat)
```

Back to diagnostics table

TEC

Diagnostic field (2d lat x lon): Total Electron Content (1/cm2):

```
diags(n)%short_name = 'TEC'
diags(n)%long_name  = 'Total Electron Content'
diags(n)%units      = '1/cm2'
diags(n)%levels     = 'none' ! 2d lon x lat
diags(n)%caller     = 'elden.F'
```

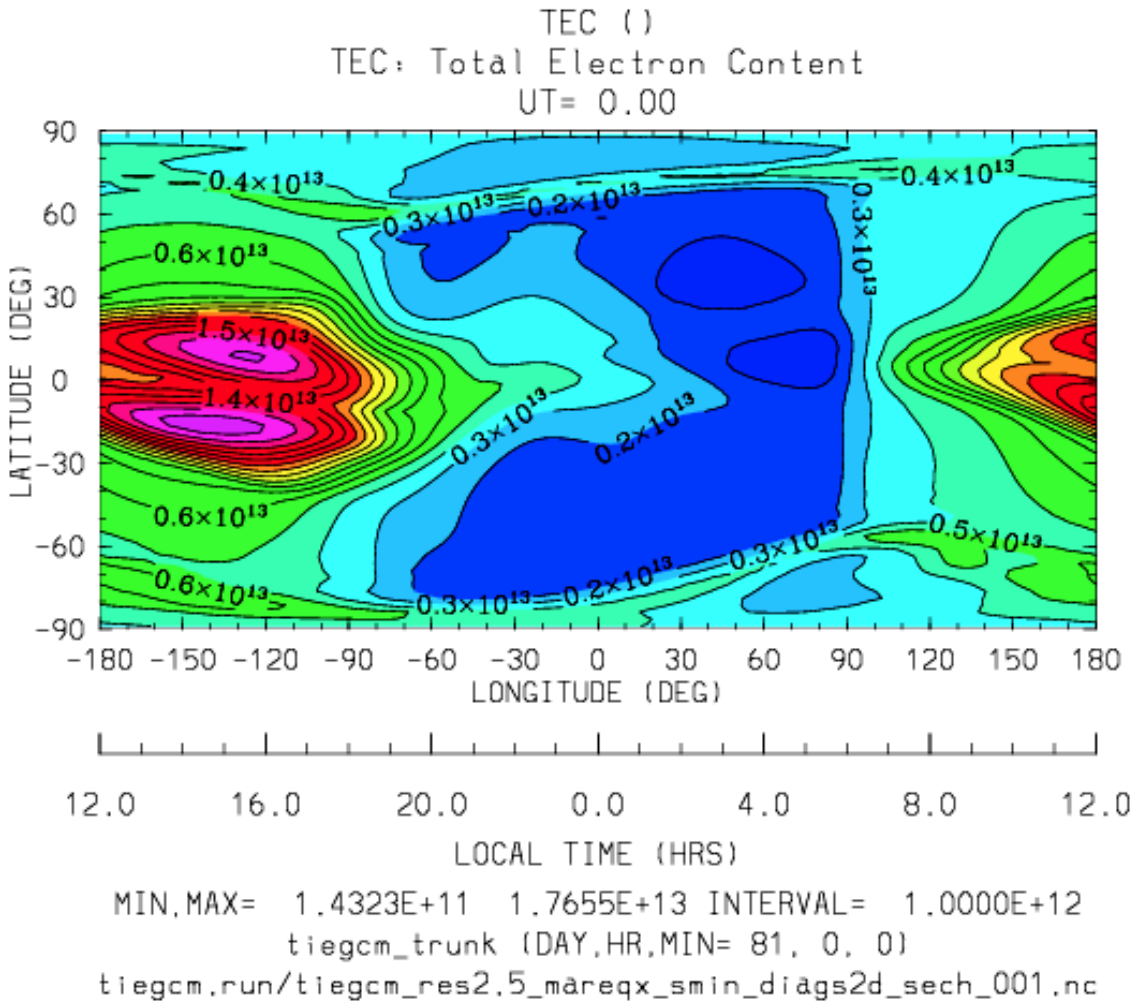
Total Electron Content is calculated by subroutine *mkdiag_TEC* in source file `diags.F`, as follows:

```
!
! Integrate electron content in height at current latitude:
      tec(:) = 0.
      do i=lon0,lon1
        do k=lev0,lev1-1
          tec(i) = tec(i)+(z(k+1,i)-z(k,i))*electrons(k,i)
        enddo
      enddo
```

Subroutine *mkdiags_TEC* is called by subroutine *elden* in source file `elden.F` as follows:

```
call mkdiag_TEC('TEC',tec,z,electrons,lev0,lev1,lon0,lon1,lat)
```

Sample images: TEC Global map



Back to diagnostics table

SCHT

Diagnostic field: Pressure Scale Height (km):

```
diags(n)%short_name = 'SCHT'
diags(n)%long_name  = 'Pressure Scale Height'
diags(n)%units      = 'km'
diags(n)%levels     = 'lev'
diags(n)%caller     = 'addiag.F'
```

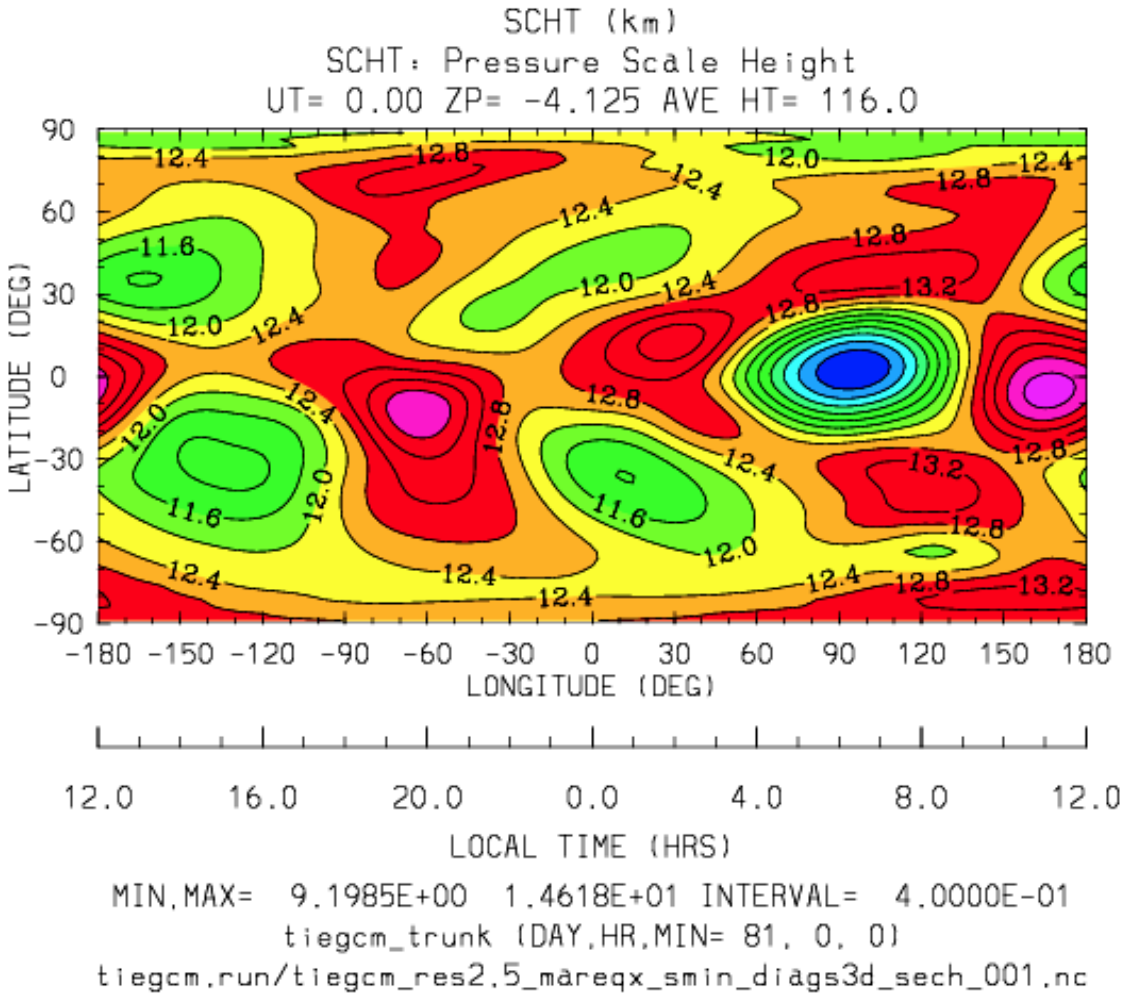
The Pressure Scale Height is calculated from the geopotential and saved by subroutine *mkdiag_SCHT* in source file `diags.F`. This code summarizes the calculation:

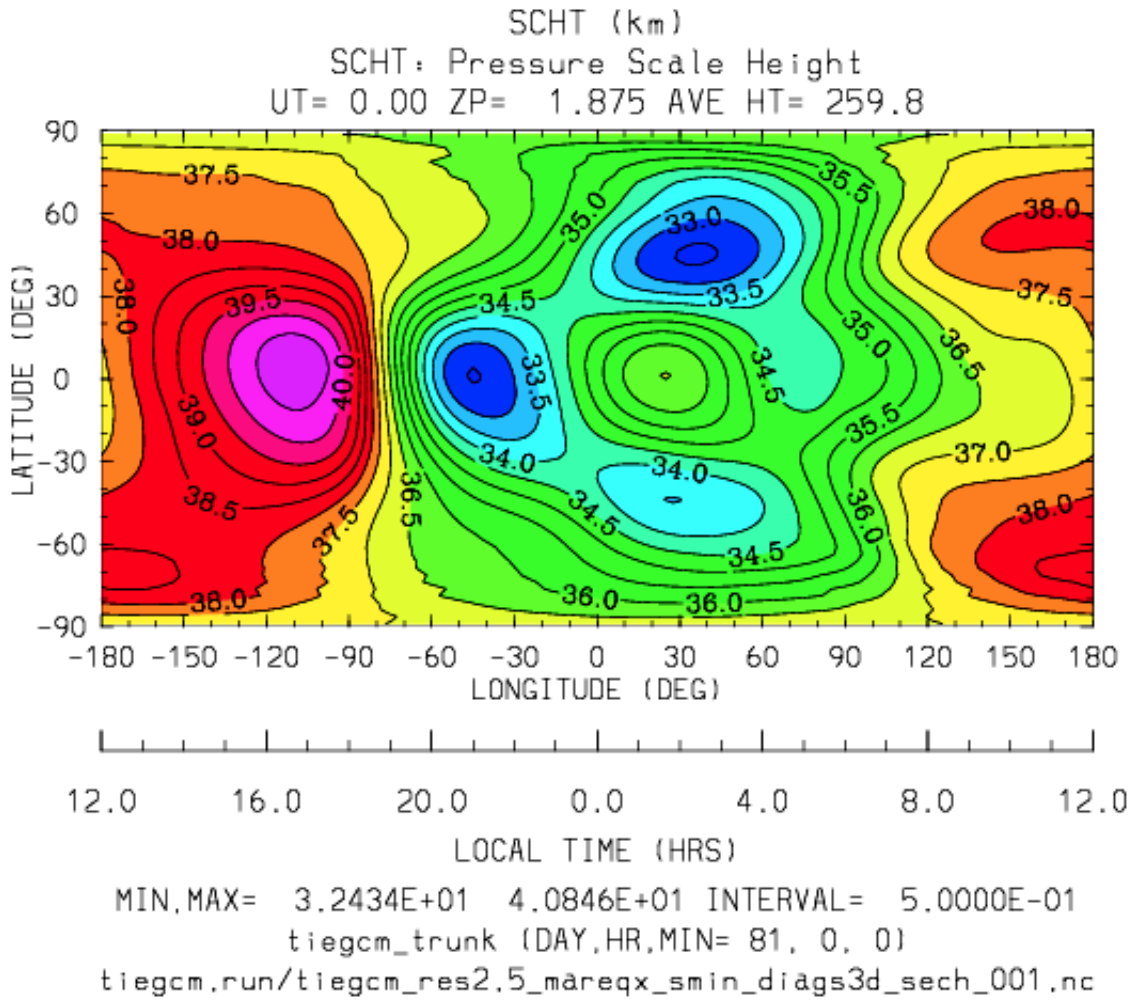
```
!
! Take delta Z:
      do j=lat0,lat1
        do i=lon0,lon1
          do k=lev0,lev1-1
            pzps(k,i) = zcm(k+1,i,j)-zcm(k,i,j)
          enddo
```

```
          pzps(lev1,i) = pzps(lev1-1,i)
!
! Generic for dlev 0.5 or 0.25 resolution:
          pzps(:,i) = pzps(:,i)/dlev
        enddo ! i=lon0,lon1
        pzps = pzps*1.e-5 ! cm to km
      enddo ! j=lat0,lat1
```

Subroutine mkdiag_SCHT is called from subroutine *addiag* (source file `addiag.F`).

Sample images: SCHT Global maps at Zp -4, +2:





Back to diagnostics table

SIGMA_HAL

Diagnostic field: Hall Conductivity (S/m):

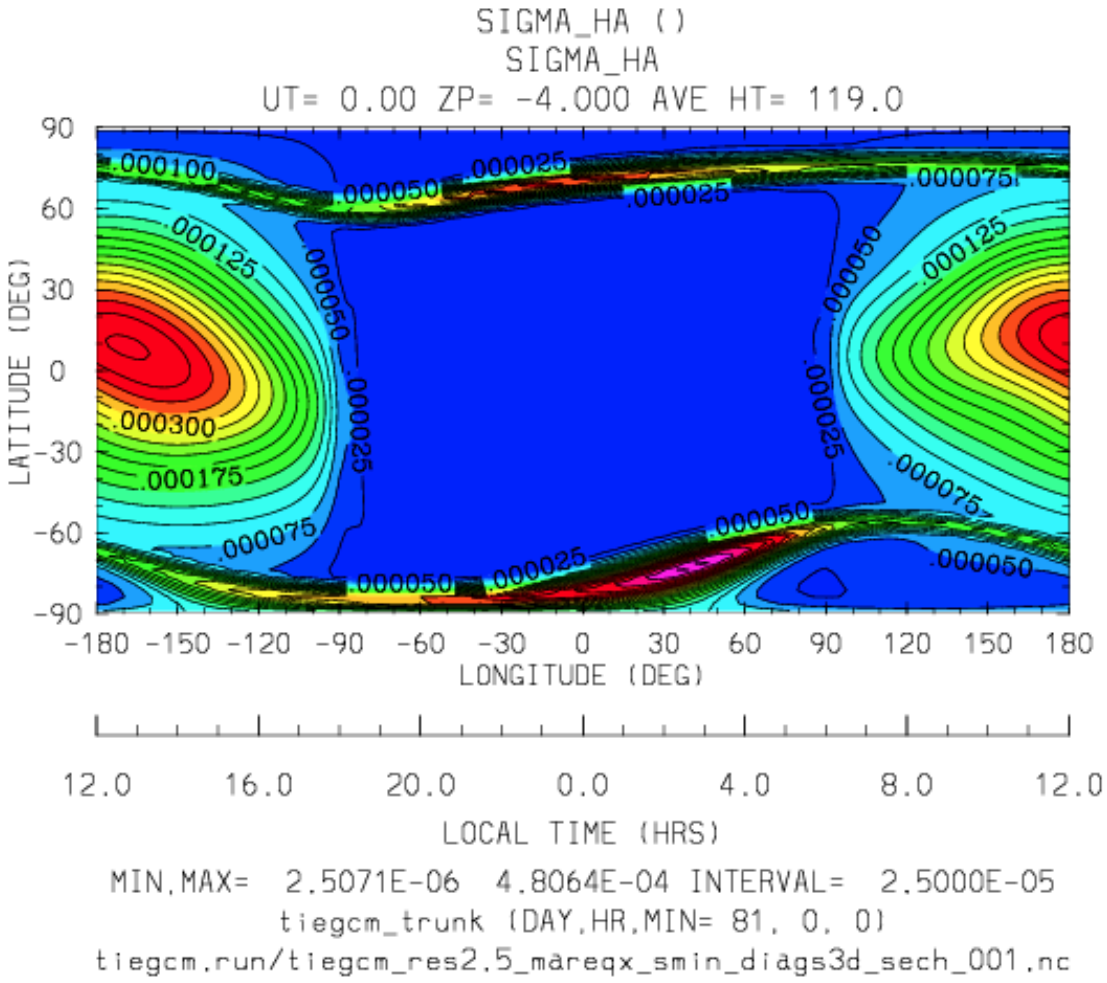
```
diags(n)%short_name = 'SIGMA_HAL'
diags(n)%long_name  = 'Hall Conductivity'
diags(n)%units      = 'S/m'
diags(n)%levels     = 'lev'
diags(n)%caller     = 'lamdas.F'
```

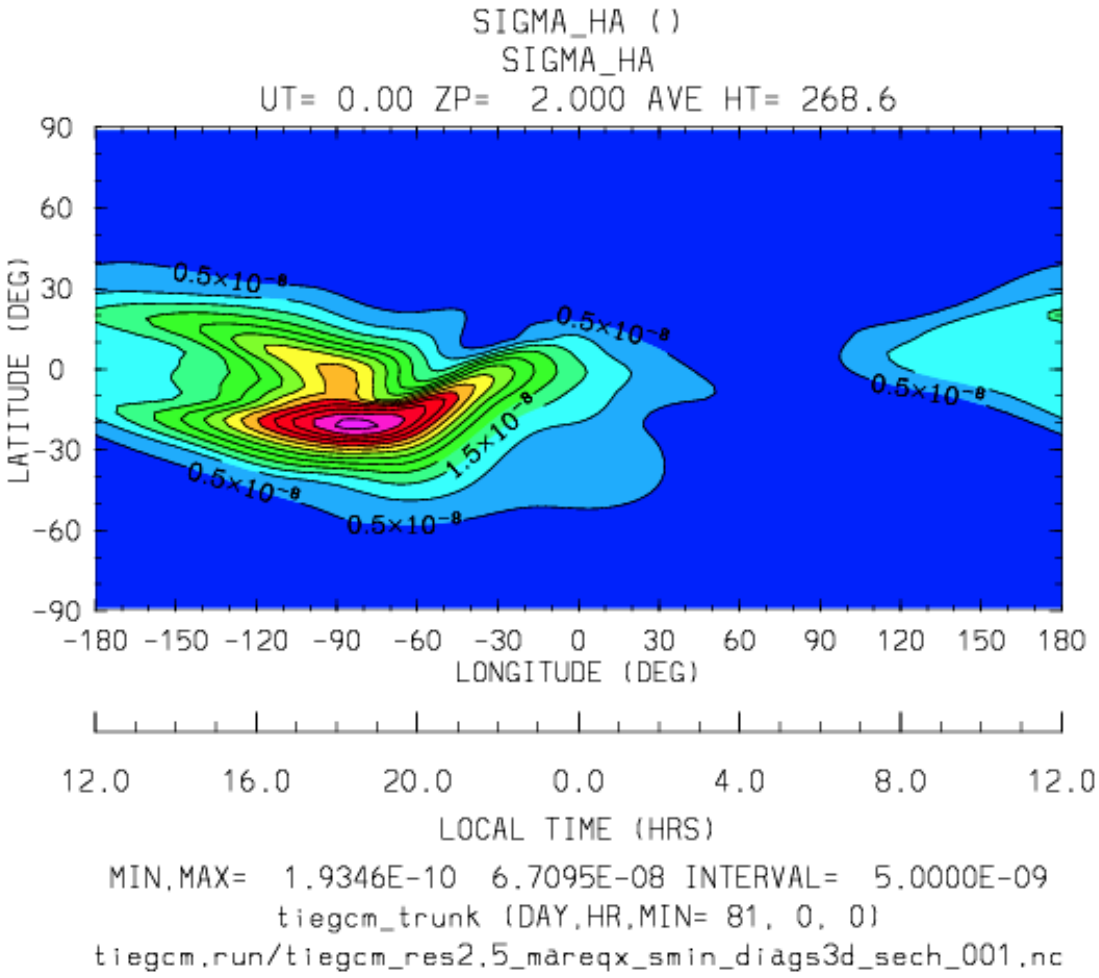
The Hall Conductivity is calculated by subroutine *lamdas* (source file `lamdas.F`), and passed to sub *mkdiag_SIGMAHAL* (`diags.F`), where it is saved to secondary histories. The calculation in lamdas.F is summarized as follows:

```
! Pedersen and Hall conductivities (siemens/m):
! Qe_fac includes conversion from CGS to SI units
!  -> e/B [C/T 10^6 m^3/cm^3], see above.
! number densities [1/cm^3]
!
      do i=lon0,lon1
```

```
        do k=lev0,lev1-1
!
! ne = electron density assuming charge equilibrium [1/cm3]:
          ne(k,i) = op(k,i)+o2p(k,i)+nop(k,i)
!
! Hall conductivity [S/m] (half level):
          sigma_hall(k,i) = qe_fac(i)*
|           (ne (k,i)/(1.+rnu_ne (k,i)**2)-
|            op (k,i)/(1.+rnu_op (k,i)**2)-
|            o2p(k,i)/(1.+rnu_o2p(k,i)**2)-
|            nop(k,i)/(1.+rnu_nop(k,i)**2))
        enddo ! k=lev0,lev1-1
      enddo ! i=lon0,lon1
```

Sample images: SIGMA_HAL Global maps at Zp -4, +2:





Back to diagnostics table

SIGMA_PED

Diagnostic field: Pedersen Conductivity (S/m):

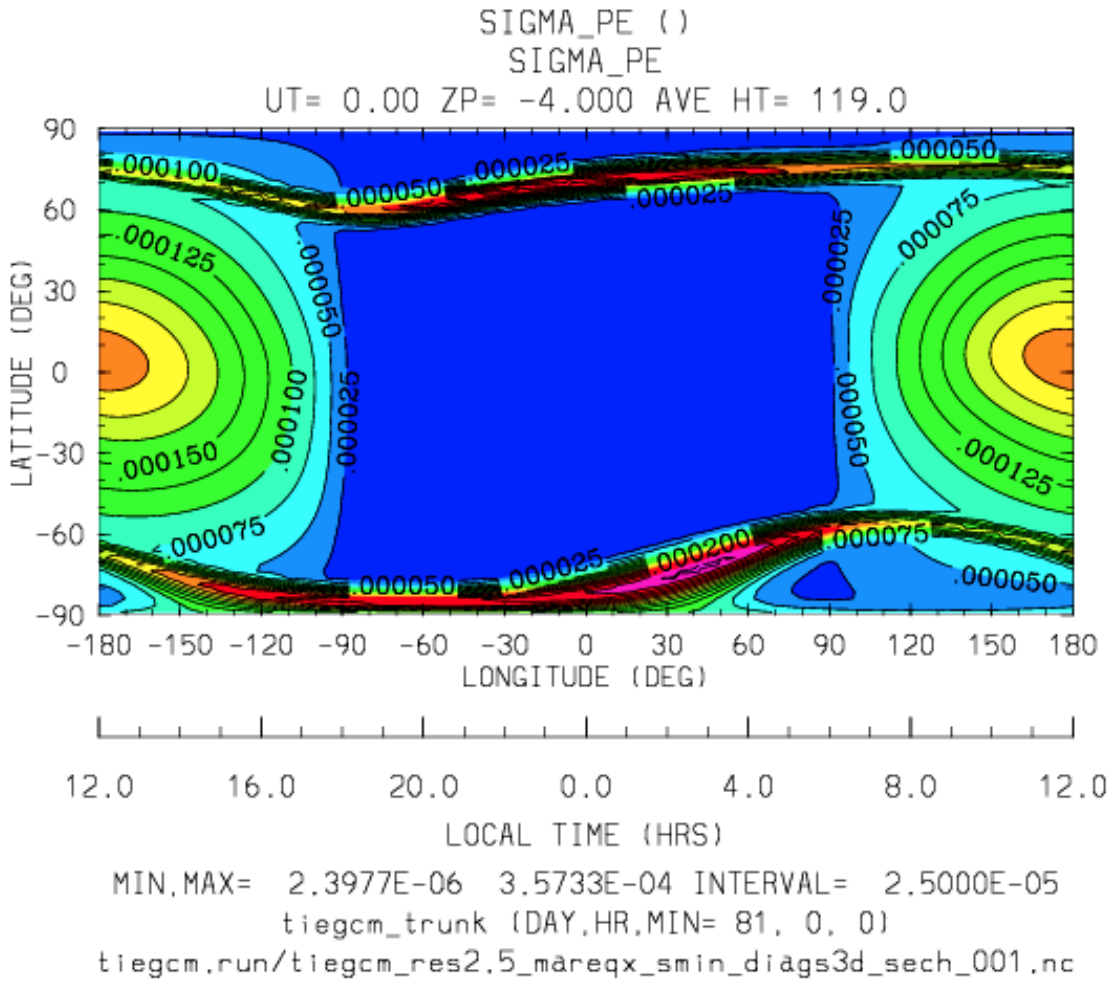
```
diags(n)%short_name = 'SIGMA_PED'
diags(n)%long_name  = 'Pedersen Conductivity'
diags(n)%units      = 'S/m'
diags(n)%levels     = 'lev'
diags(n)%caller     = 'lamdas.F'
```

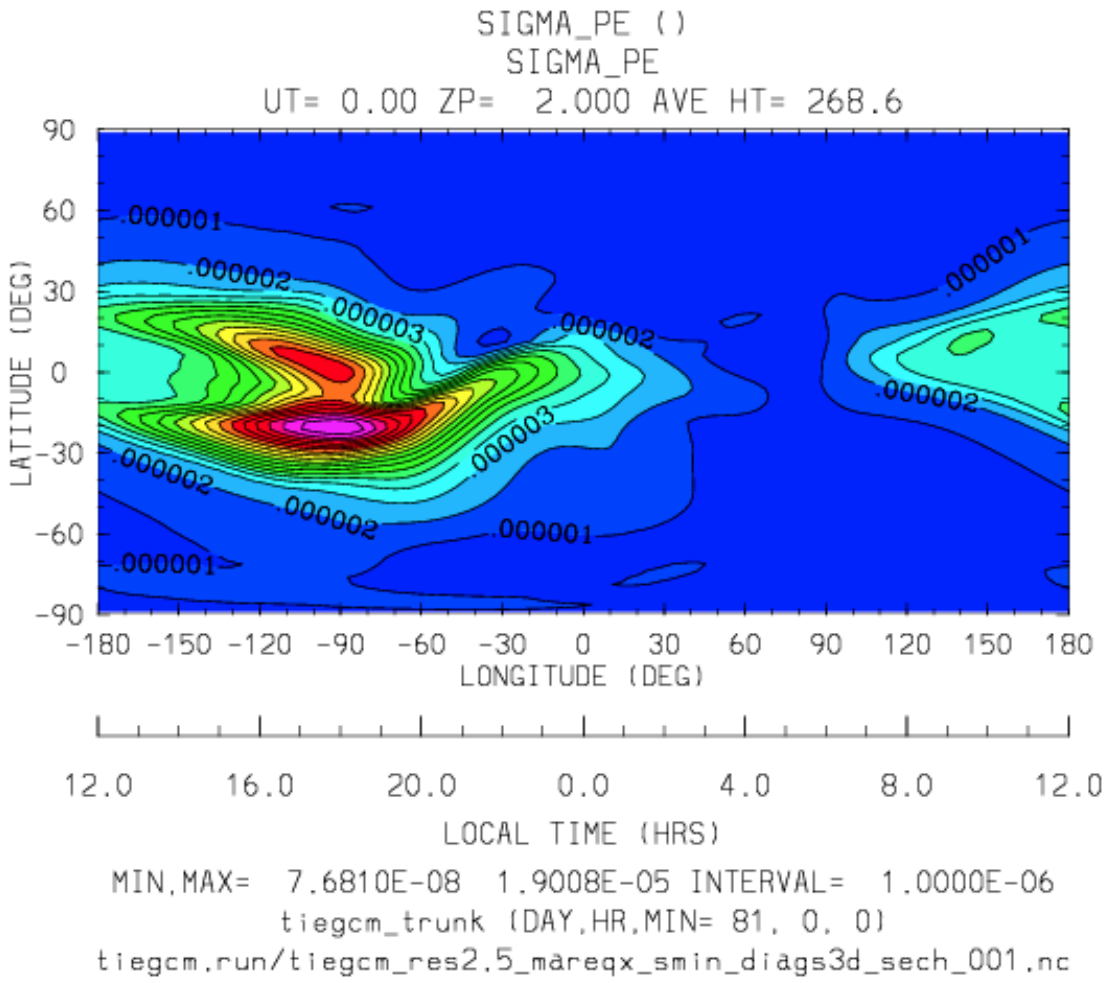
The Pedersen Conductivity is calculated by subroutine *lamdas* (source file `lamdas.F`), and passed to sub *mkdiag_SIGMAPED* (`diags.F`), where it is saved to secondary histories. The calculation in lamdas.F is summarized as follows:

```
! Pedersen and Hall conductivities (siemens/m):
! Qe_fac includes conversion from CGS to SI units
!  -> e/B [C/T 10^6 m^3/cm^3], see above.
! number densities [1/cm^3]
!
      do i=lon0,lon1
        do k=lev0,lev1-1
```

```
!
! ne = electron density assuming charge equilibrium [1/cm3]:
          ne(k,i) = op(k,i)+o2p(k,i)+nop(k,i)
!
! Pedersen conductivity [S/m] (half level):
          sigma_ped(k,i) = qe_fac(i)*
     |       ((op (k,i)*rnu_op (k,i)/(1.+rnu_op (k,i)**2))+
     |        (o2p(k,i)*rnu_o2p(k,i)/(1.+rnu_o2p(k,i)**2))+
     |        (nop(k,i)*rnu_nop(k,i)/(1.+rnu_nop(k,i)**2))+
     |        (ne (k,i)*rnu_ne (k,i)/(1.+rnu_ne (k,i)**2)))
        enddo ! k=lev0,lev1-1
      enddo ! i=lon0,lon1
```

Sample images: SIGMA_PED Global maps at Zp -4, +2:





Back to diagnostics table

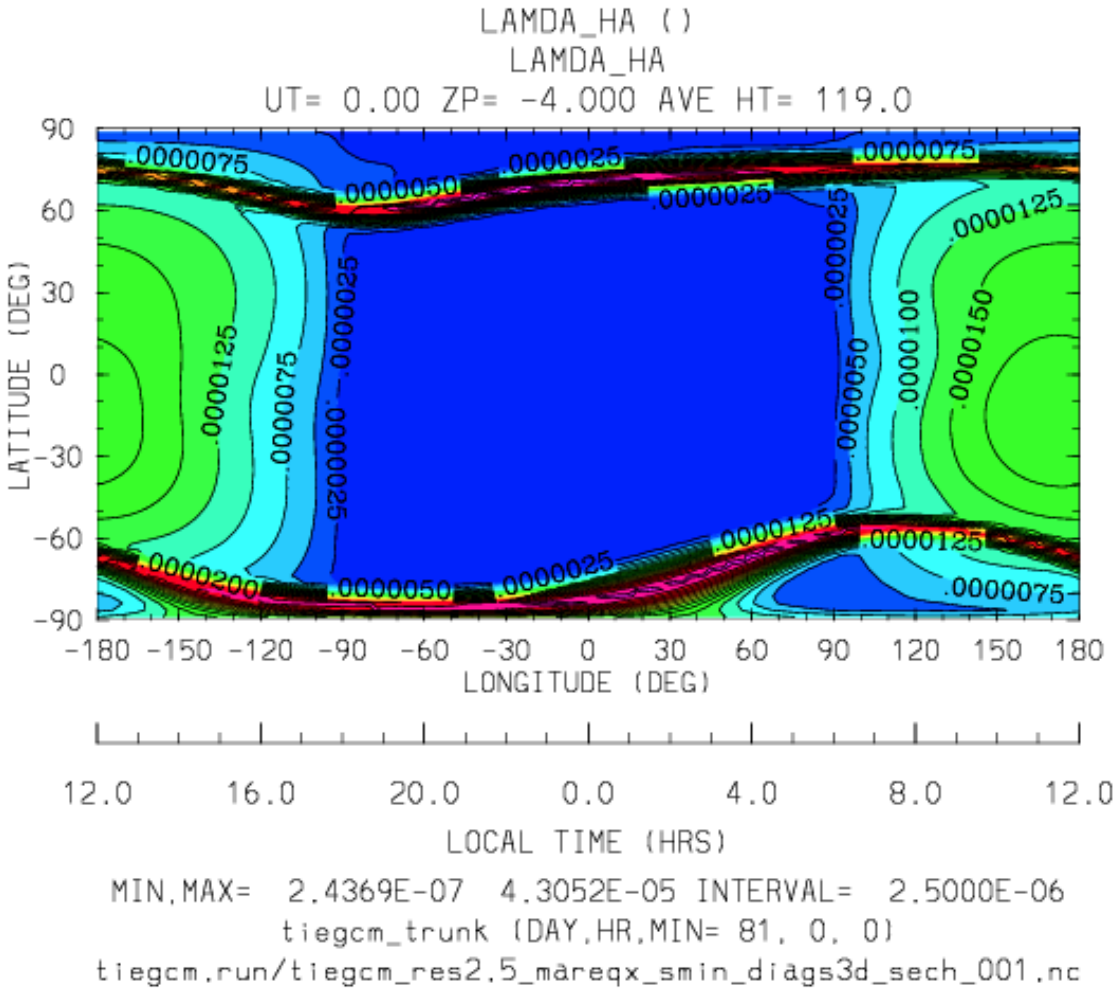
LAMDA_HAL

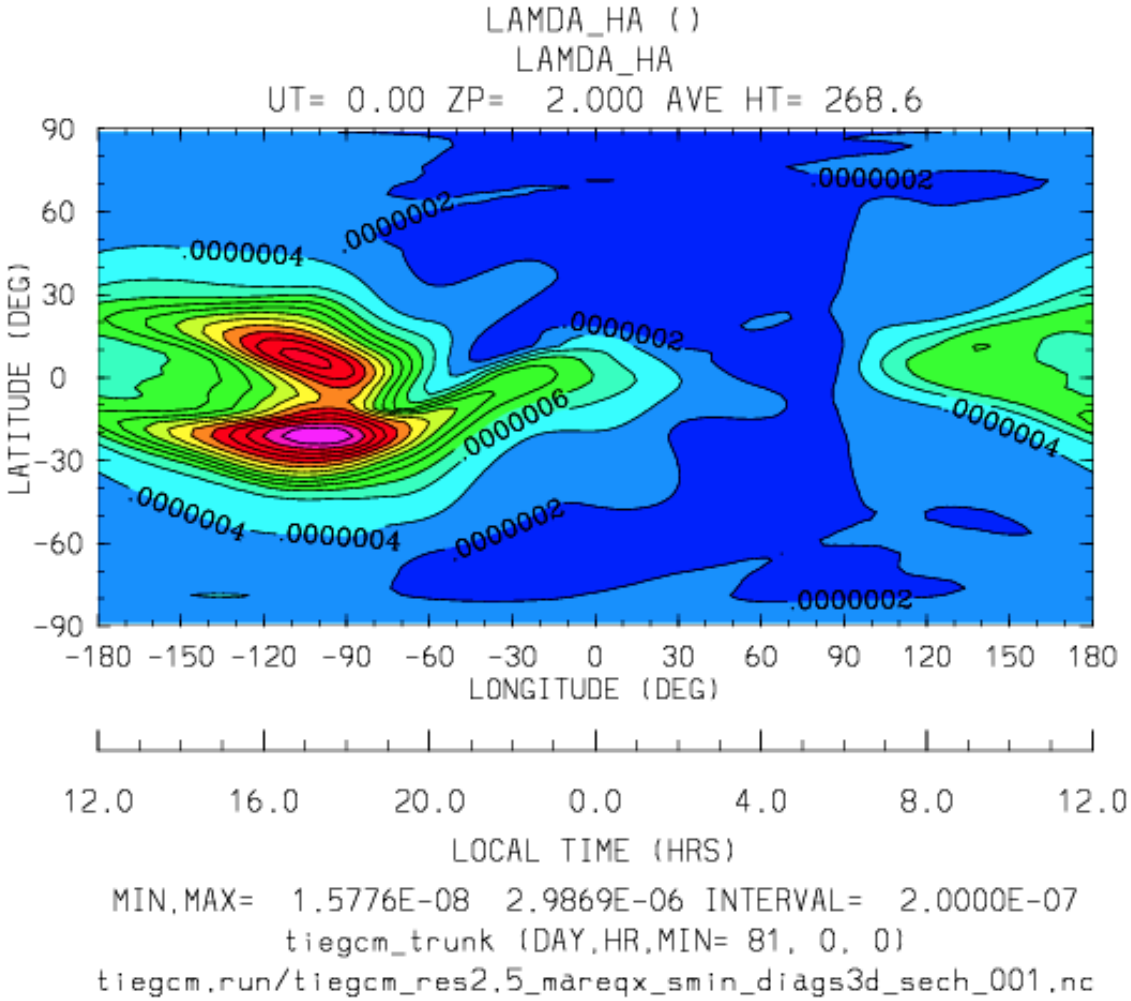
Diagnostic field: Hall Ion Drag Coefficient (1/s):

```
diags(n)%short_name = 'LAMDA_HAL'
diags(n)%long_name  = 'Hall Ion Drag Coefficient'
diags(n)%units      = '1/s'
diags(n)%levels     = 'lev'
diags(n)%caller     = 'lamdas.F'
```

The Hall Ion Drag Coefficient is calculated in subroutine *lamdas* (source file `lamdas.F`), and saved to seconday histories by subroutine *mkdiag_LAMDAHAL* (`diags.F`).

Sample images: LAMDA_HAL Global maps at Zp -4, +2:


LAMDA_HA ()
LAMDA_HA
UT= 0.00 ZP= -4.000 AVE HT= 119.0
LATITUDE (DEG)
90
60
30
0
-30
-60
-90
-180 -150 -120 -90 -60 -30 0 30 60 90 120 150 180
LONGITUDE (DEG)
12.0 16.0 20.0 0.0 4.0 8.0 12.0
LOCAL TIME (HRS)
MIN,MAX= 2.4369E-07 4.3052E-05 INTERVAL= 2.5000E-06
tiegcm_trunk (DAY,HR,MIN= 81, 0, 0)
tiegcm.run/tiegcm_res2.5_mareqx_smin_diags3d_sech_001.nc



Back to diagnostics table

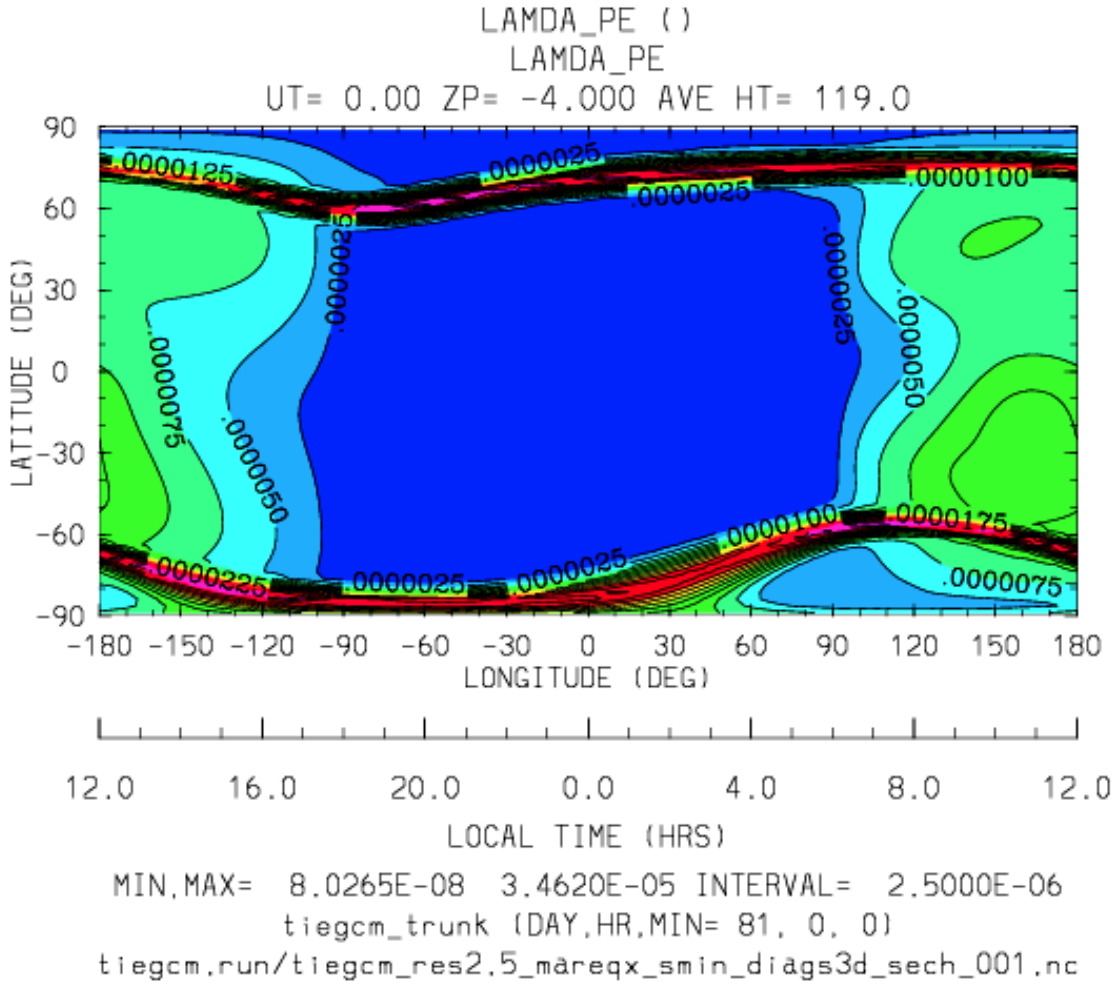
LAMDA_PED

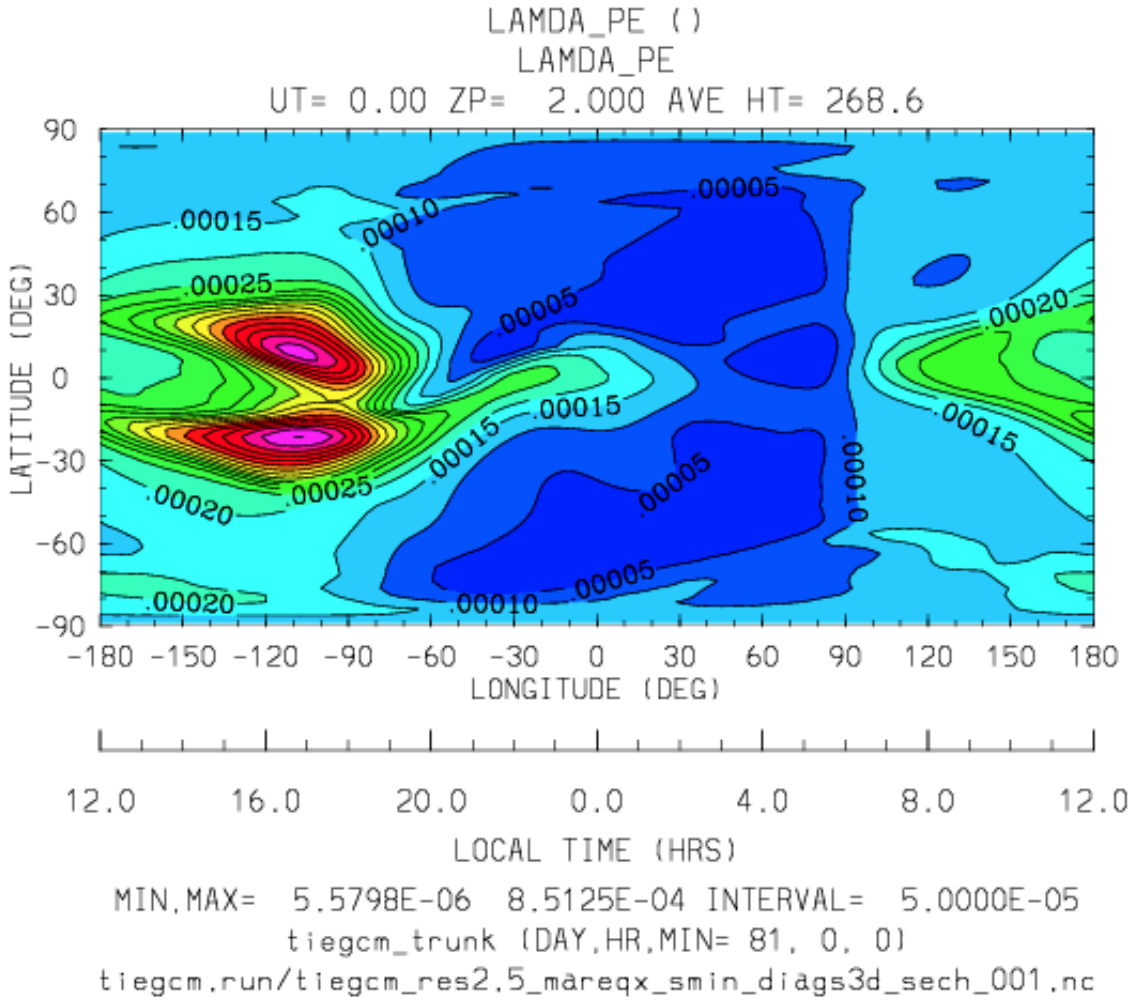
Diagnostic field: Hall Ion Drag Coefficient (1/s):

```
diags(n)%short_name = 'LAMDA_PED'
diags(n)%long_name  = 'Pedersen Ion Drag Coefficient'
diags(n)%units      = '1/s'
diags(n)%levels     = 'lev'
diags(n)%caller     = 'lamdas.F'
```

The Pedersen Ion Drag Coefficient is calculated in subroutine *lamdas* (source file `lamdas.F`), and saved to secondary histories by subroutine *mkdiag_LAMDAPED* (`diags.F`).

Sample images: LAMDA_PED Global maps at Zp -4, +2:


LAMDA_PE ()
LAMDA_PE
UT= 0.00 ZP= -4.000 AVE HT= 119.0
90
60
30
0
-30
-60
-90
LATITUDE (DEG)
.0000125
.0000025
.0000100
.0000225
.0000075
.0000050
.0000175
-180 -150 -120 -90 -60 -30 0 30 60 90 120 150 180
LONGITUDE (DEG)
12.0 16.0 20.0 0.0 4.0 8.0 12.0
LOCAL TIME (HRS)
MIN,MAX= 8.0265E-08 3.4620E-05 INTERVAL= 2.5000E-06
tiegcm_trunk (DAY,HR,MIN= 81, 0, 0)
tiegcm.run/tiegcm_res2.5_mareqx_smin_diags3d_sech_001.nc



Back to diagnostics table

UI_ExB

Diagnostic field: Zonal Ion Drift (ExB) (cm/s):

```
diags(n)%short_name = 'UI_ExB'
diags(n)%long_name  = 'Zonal Ion Drift (ExB)'
diags(n)%units      = 'cm/s'
diags(n)%levels     = 'ilev'
diags(n)%caller     = 'ionvel.F'
```

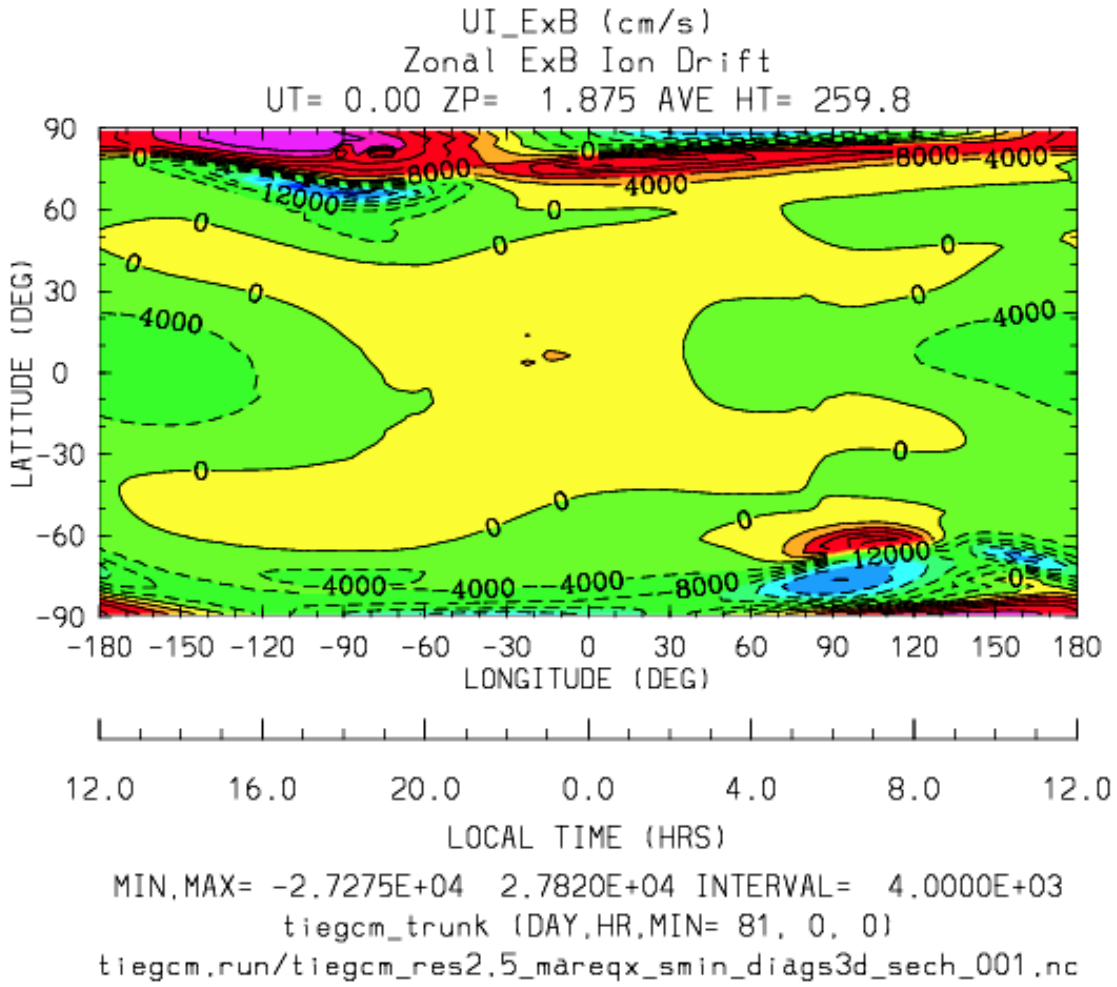
Calculated by subroutine *ionvel* (`ionvel.F`):

```
!
! ion velocities = (e x b/b**2)
! ui = zonal, vi = meridional, wi = vertical
        do k=lev0,lev1
          do i=lonbeg,lonend
            ui(k,i,lat) = -(eey(k,i)*zb(i-2,lat)+eez(k,i)*xb(i-2,lat))*
|             1.e6/bmod(i-2,lat)**2
            vi(k,i,lat) =  (eez(k,i)*yb(i-2,lat)+eex(k,i)*zb(i-2,lat))*
```

```
|              1.e6/bmod(i-2,lat)**2
             wi(k,i,lat) =  (eex(k,i)*xb(i-2,lat)-eey(k,i)*yb(i-2,lat))*
|              1.e6/bmod(i-2,lat)**2
           enddo ! i=lon0,lon1
         enddo ! k=lev0,lev1
```

Subroutine ionvel calls subroutine *mkdiag_UI* (`diags.F`) to save the field to secondary histories. The field is converted from m/s to cm/s in ionvel before the call to mkdiag_UI.

Sample images: UI_ExB Global maps at Zp +2:



Back to diagnostics table

VI_ExB

Diagnostic field: Meridional Ion Drift (ExB) (cm/s):

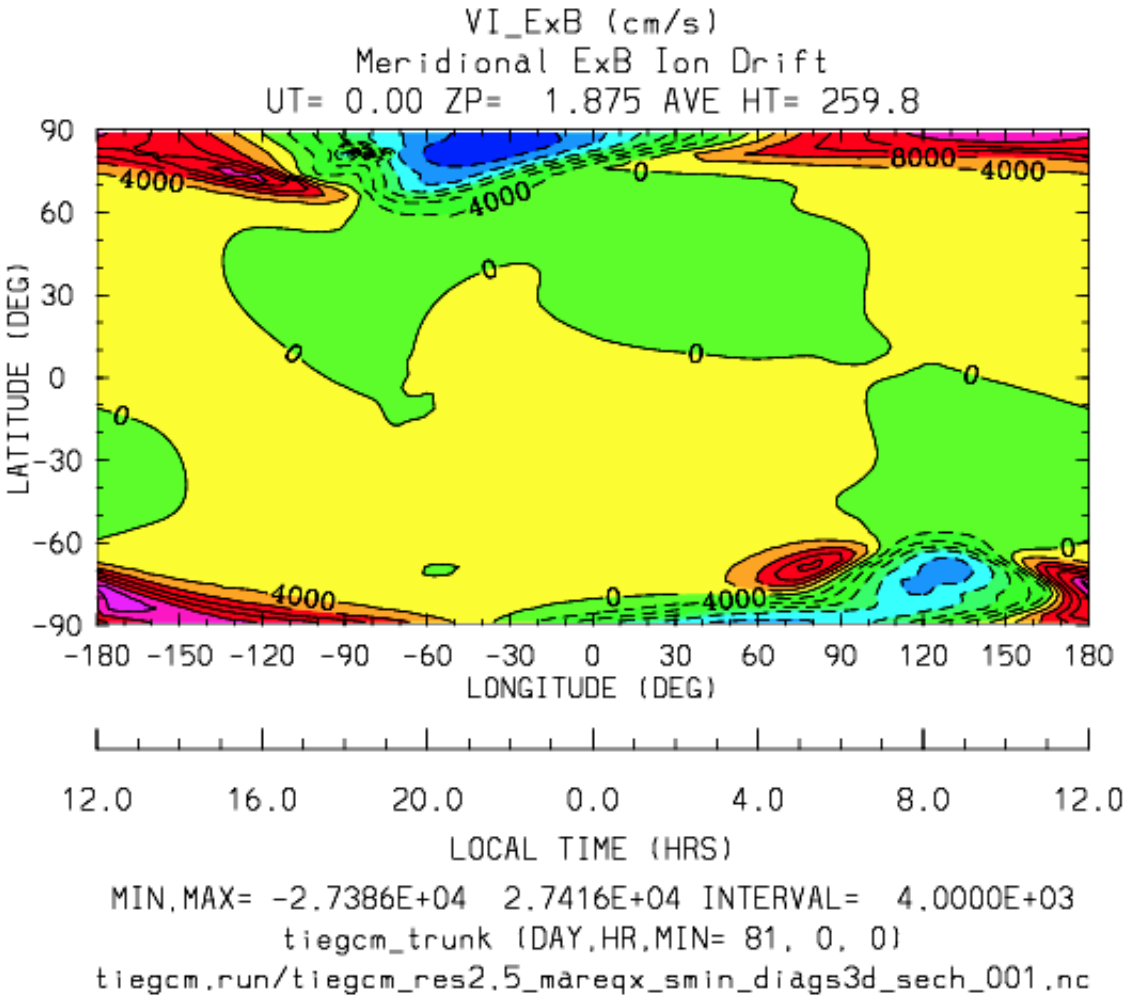
```
diags(n)%short_name = 'VI_ExB'
diags(n)%long_name  = 'Meridional Ion Drift (ExB)'
diags(n)%units      = 'cm/s'
diags(n)%levels     = 'ilev'
diags(n)%caller     = 'ionvel.F'
```

Calculated by subroutine *ionvel* (`ionvel.F`):

```
!
! ion velocities = (e x b/b**2)
! ui = zonal, vi = meridional, wi = vertical
        do k=lev0,lev1
          do i=lonbeg,lonend
            ui(k,i,lat) = -(eey(k,i)*zb(i-2,lat)+eez(k,i)*xb(i-2,lat))*
|             1.e6/bmod(i-2,lat)**2
            vi(k,i,lat) =  (eez(k,i)*yb(i-2,lat)+eex(k,i)*zb(i-2,lat))*
|             1.e6/bmod(i-2,lat)**2
            wi(k,i,lat) =  (eex(k,i)*xb(i-2,lat)-eey(k,i)*yb(i-2,lat))*
|             1.e6/bmod(i-2,lat)**2
          enddo ! i=lon0,lon1
        enddo ! k=lev0,lev1
```

Subroutine ionvel calls subroutine *mkdiag_VI* (`diags.F`) to save the field to secondary histories. The field is converted from m/s to cm/s in ionvel before the call to mkdiag_VI.

Sample images: VI_ExB Global maps at Zp +2:



Back to diagnostics table

WI_ExB

Diagnostic field: Vertical Ion Drift (ExB) (cm/s):

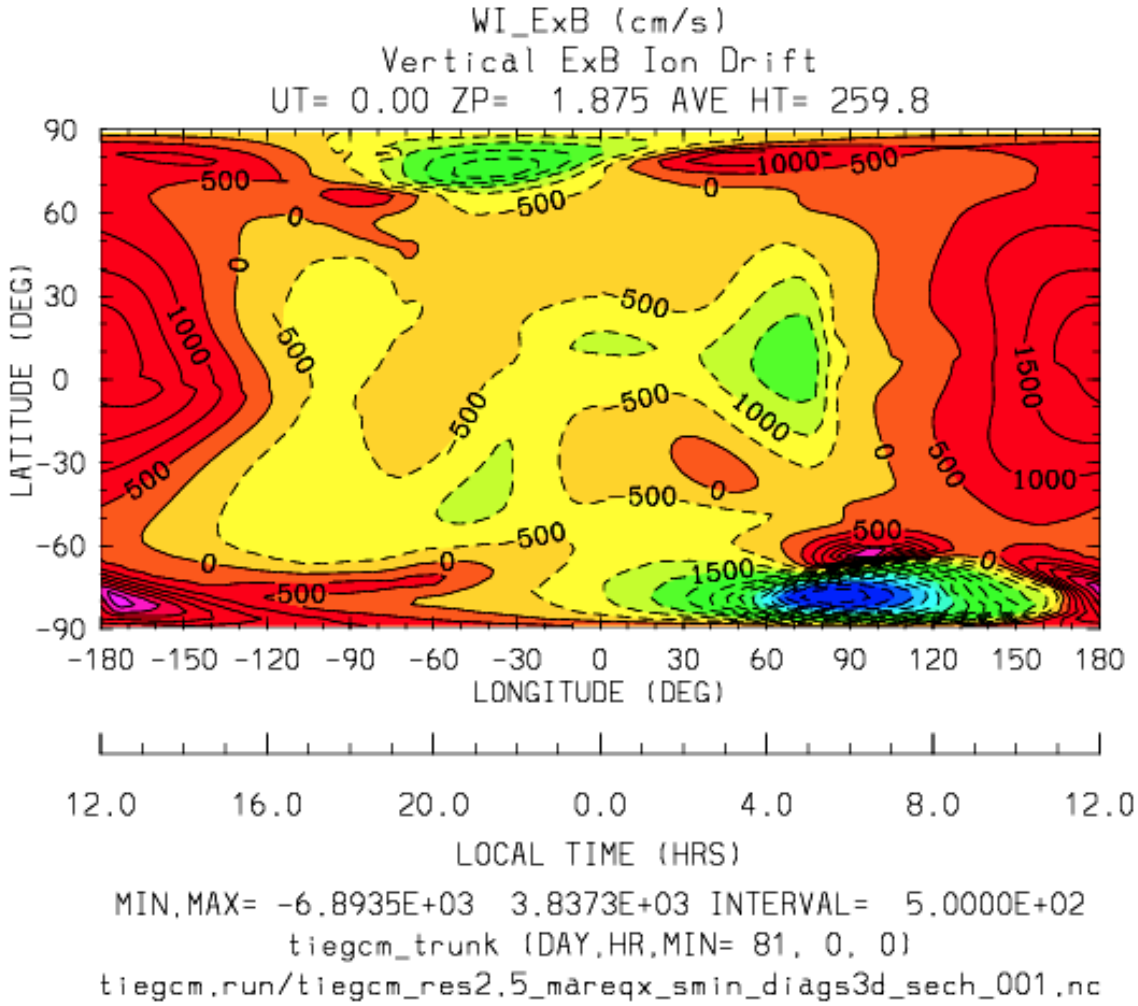
```
diags(n)%short_name = 'WI_ExB'
diags(n)%long_name  = 'Vertical Ion Drift (ExB)'
diags(n)%units      = 'cm/s'
diags(n)%levels     = 'ilev'
diags(n)%caller     = 'ionvel.F'
```

Calculated by subroutine *ionvel* (`ionvel.F`):

```
!
! ion velocities = (e x b/b**2)
! ui = zonal, vi = meridional, wi = vertical
        do k=lev0,lev1
          do i=lonbeg,lonend
            ui(k,i,lat) = -(eey(k,i)*zb(i-2,lat)+eez(k,i)*xb(i-2,lat))*
|             1.e6/bmod(i-2,lat)**2
            vi(k,i,lat) =  (eez(k,i)*yb(i-2,lat)+eex(k,i)*zb(i-2,lat))*
|             1.e6/bmod(i-2,lat)**2
            wi(k,i,lat) =  (eex(k,i)*xb(i-2,lat)-eey(k,i)*yb(i-2,lat))*
|             1.e6/bmod(i-2,lat)**2
          enddo ! i=lon0,lon1
        enddo ! k=lev0,lev1
```

Subroutine ionvel calls subroutine *mkdiag_UI* (`diags.F`) to save the field to secondary histories. The field is converted from m/s to cm/s in ionvel before the call to mkdiag_WI.

Sample images: WI_ExB Global maps at Zp +2:



Back to diagnostics table

MU_M

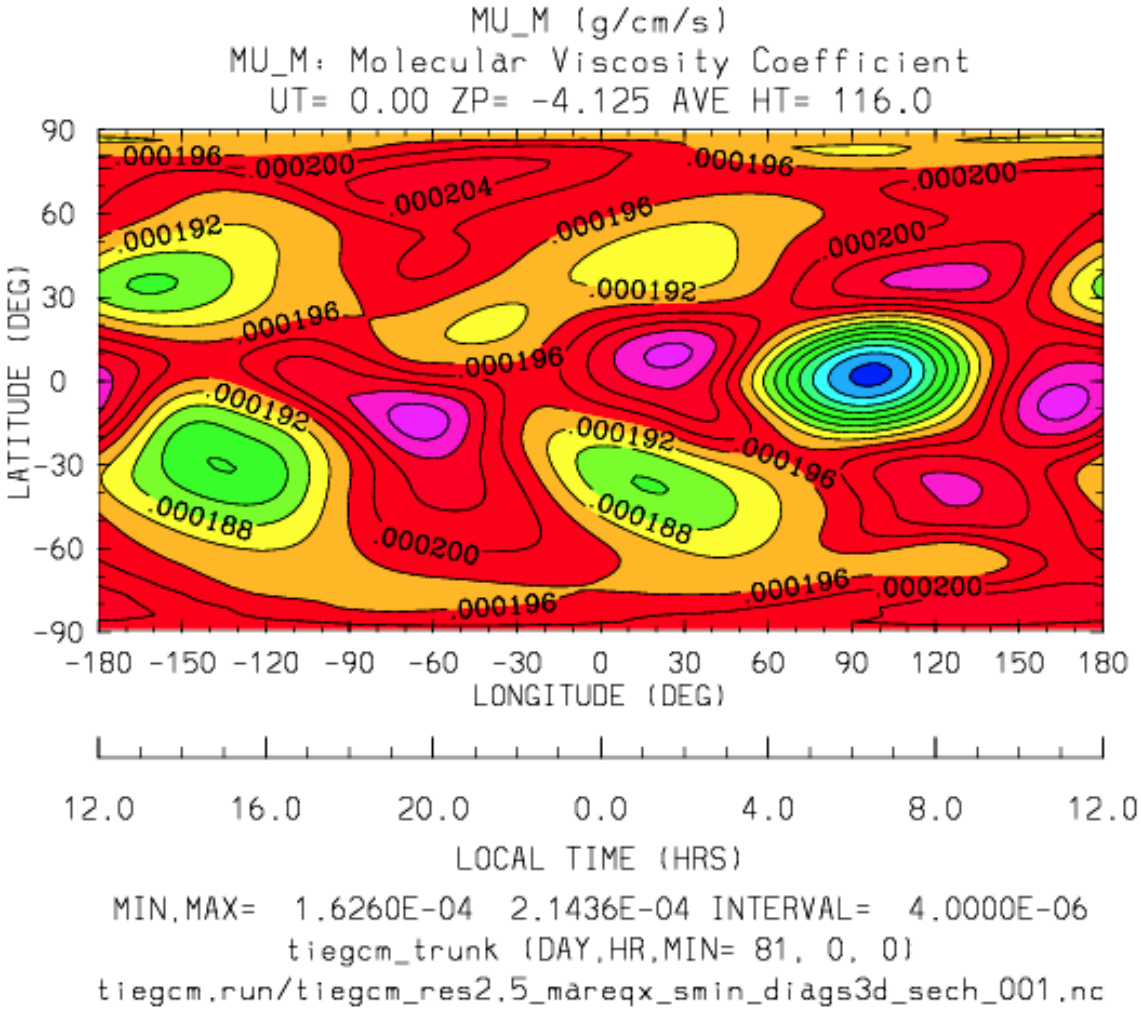
Diagnostic field: Molecular Viscosity Coefficient (g/cm/s):

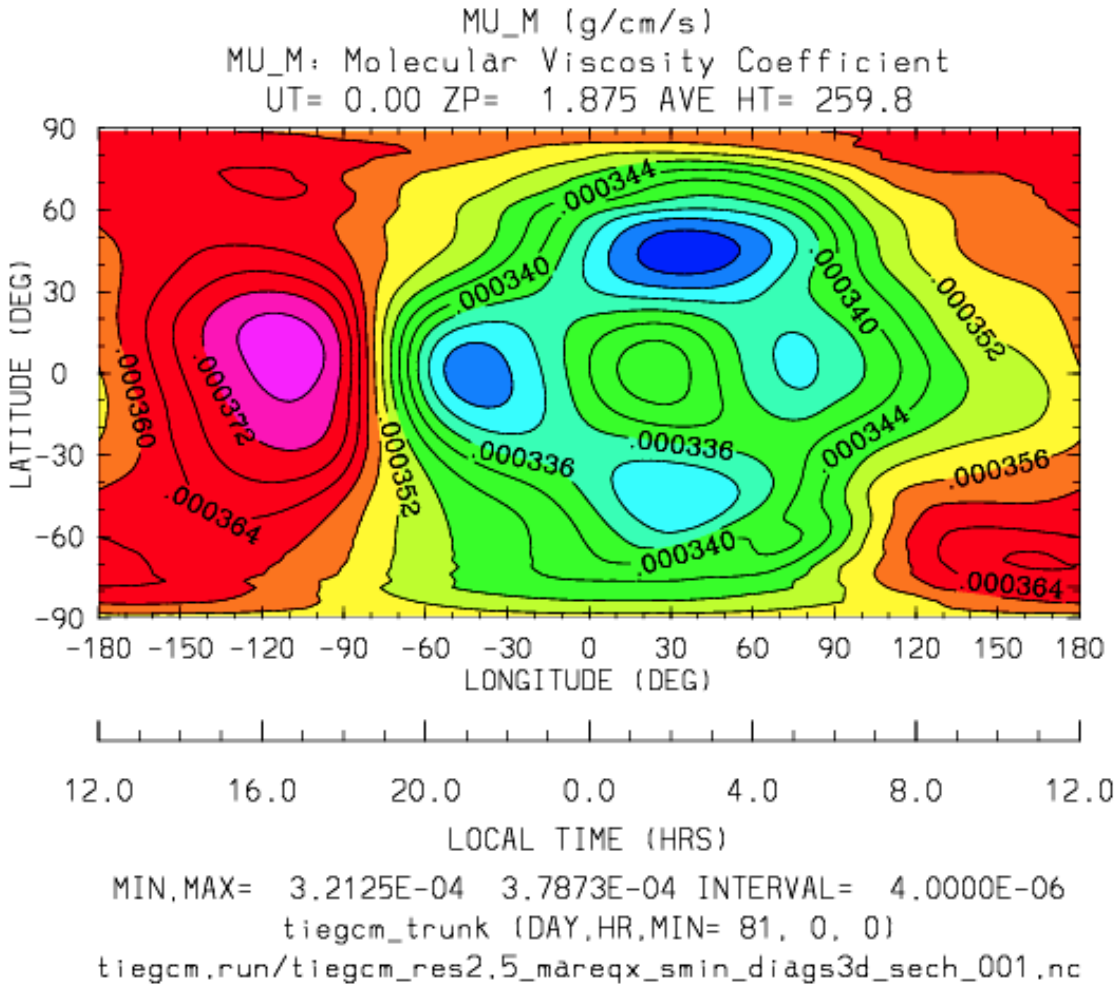
```
diags(n)%short_name = 'MU_M'
diags(n)%long_name  = 'Molecular Viscosity Coefficient'
diags(n)%units      = 'g/cm/s'
diags(n)%levels     = 'lev'
diags(n)%caller     = 'cpktkm.F'
```

The Molecular Viscosity Coefficient is calculated by subroutine *cpktkm* (source file `cpktkm.F`), and saved to secondary histories by subroutine *mkdiag_MU_M* (`diags.F`). The calculation in cpktkm is summarized as follows:

```
fkm(k,i) = po2(k,i)*4.03 + pn2(k,i)*3.42 + po1(k,i)*3.9
```

Sample images: MU_M Global maps at Zp -4, +2:


MU_M (g/cm/s)
MU_M: Molecular Viscosity Coefficient
UT= 0.00 ZP= -4.125 AVE HT= 116.0
LATITUDE (DEG)
90
60
30
0
-30
-60
-90
.000196
.000200
.000204
.000192
.000188
-180 -150 -120 -90 -60 -30 0 30 60 90 120 150 180
LONGITUDE (DEG)
12.0 16.0 20.0 0.0 4.0 8.0 12.0
LOCAL TIME (HRS)
MIN,MAX= 1.6260E-04 2.1436E-04 INTERVAL= 4.0000E-06
tiegcm_trunk (DAY,HR,MIN= 81, 0, 0)
tiegcm.run/tiegcm_res2.5_mareqx_smin_diags3d_sech_001.nc



Back to diagnostics table

WN

Diagnostic field: Neutral Vertical Wind (cm/s):

```
diags(n)%short_name = 'WN'
diags(n)%long_name  = 'NEUTRAL VERTICAL WIND (plus up)'
diags(n)%units      = 'cm/s'
diags(n)%levels     = 'ilev'
diags(n)%caller     = 'swdot.F'
```

Note: This 3d field is calculated on fixed pressure surfaces ln(p0/p), i.e., there is no interpolation to height.

Calculated from OMEGA (vertical motion) and pressure scale height by subroutine *mkdiag_WN* in source file `diags.F`:

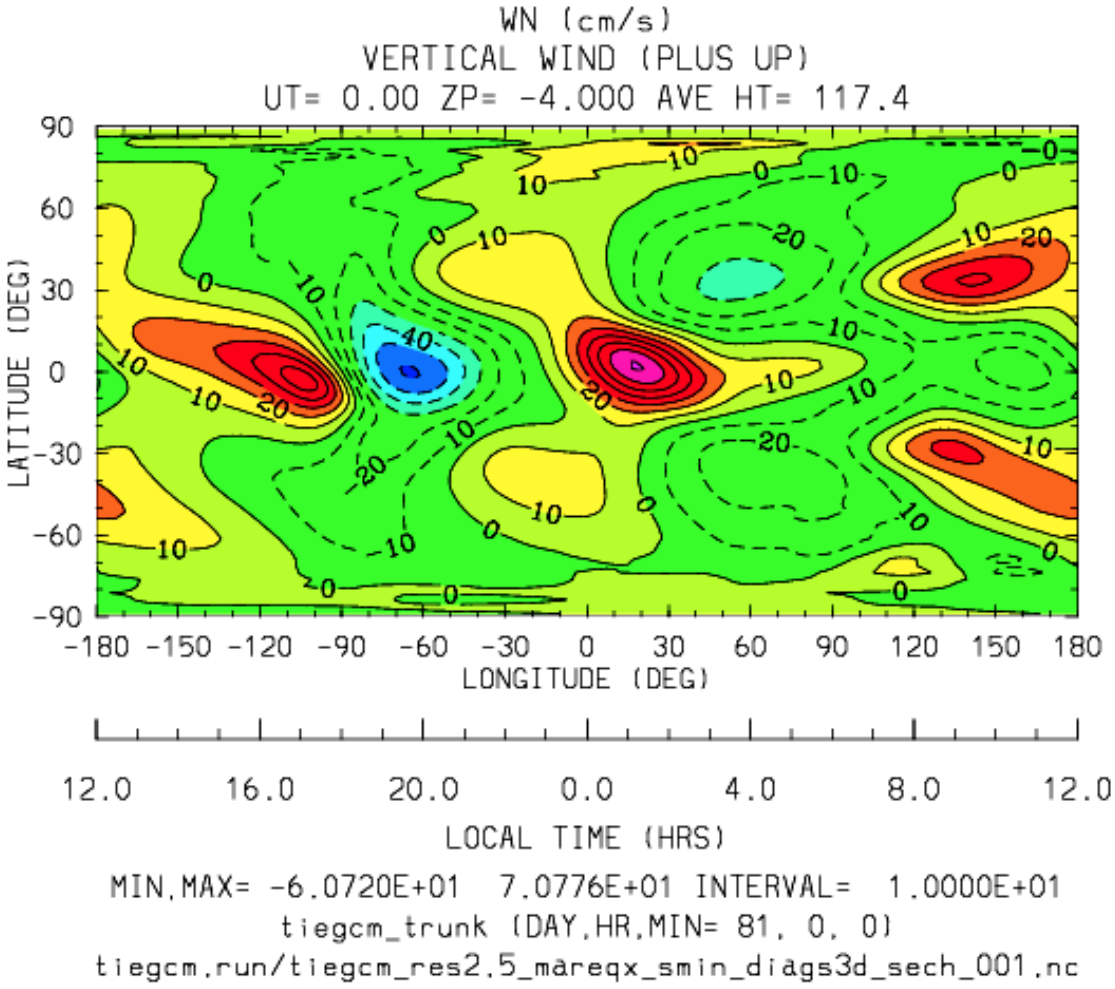
```
!-----------------------------------------------------------------------
      subroutine mkdiag_WN(name,omega,zcm,lev0,lev1,lon0,lon1,lat)
!
```

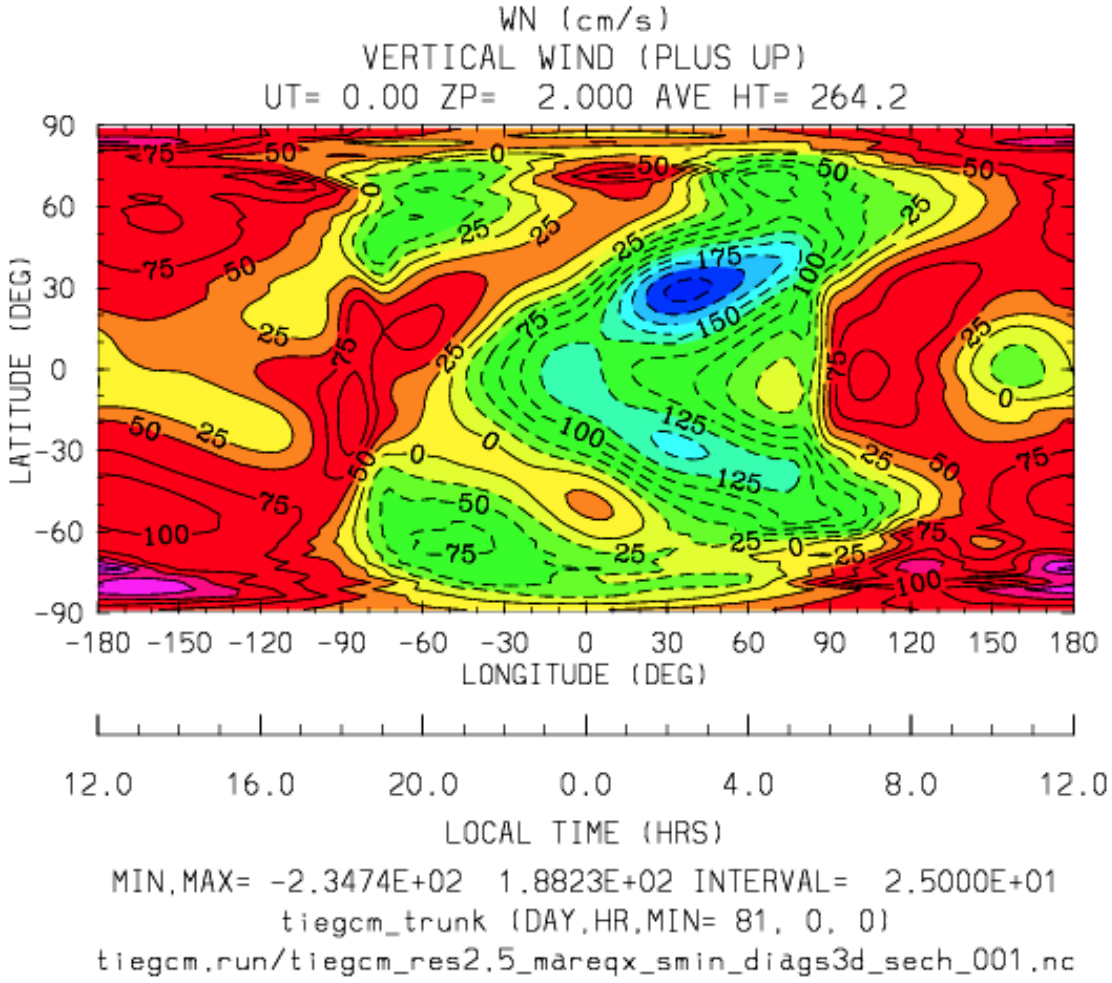
```
! Neutral Vertical Wind, from vertical motion OMEGA and scale height.
! Scale height pzps is calculated from input geopotential z (cm).
!
! Args:
      character(len=*),intent(in) :: name
      integer,intent(in) :: lev0,lev1,lon0,lon1,lat
      real,intent(in),dimension(lev0:lev1,lon0:lon1) :: omega,zcm
!
! Local:
      integer :: i,k,ix
      real,dimension(lev0:lev1,lon0:lon1) :: wn
      real,dimension(lev0:lev1) :: pzps,omega1
!
! Check that field name is a diagnostic, and was requested:
      ix = checkf(name) ; if (ix==0) return
!
! Calculate scale height pzps:
      do i=lon0,lon1
        do k=lev0+1,lev1-1
          pzps(k) = (zcm(k+1,i)-zcm(k-1,i))/(2.*dlev)
        enddo
        pzps(lev0) = (zcm(lev0+1,i)-zcm(lev0,i))/dlev
        pzps(lev1) = pzps(lev1-1)
!
        omega1(:) = omega(:,i)
        omega1(lev1) = omega1(lev1-1)
!
! Output vertical wind (cm):
        wn(:,i) = omega1(:)*pzps(:)
      enddo ! i=lon0,lon1

      call addfld(diags(ix)%short_name,diags(ix)%long_name,
     |  diags(ix)%units,wn,'lev',lev0,lev1,'lon',lon0,lon1,lat)

      end subroutine mkdiag_WN
!-----------------------------------------------------------------------
```

Called by: subroutine *swdot* in source file `swdot.F` as follows:

```
do lat=lat0,lat1
  call mkdiag_WN('WN',w(:,lon0:lon1,lat),z(:,lon0:lon1,lat),lev0,lev1,lon0,lon1,lat)
enddo
```

Sample images: WN Global maps at Zp -4, +2:


WN (cm/s)
VERTICAL WIND (PLUS UP)
UT= 0.00 ZP= -4.000 AVE HT= 117.4
LATITUDE (DEG)
90
60
30
0
-30
-60
-90
-180 -150 -120 -90 -60 -30 0 30 60 90 120 150 180
LONGITUDE (DEG)
12.0 16.0 20.0 0.0 4.0 8.0 12.0
LOCAL TIME (HRS)
MIN,MAX= -6.0720E+01 7.0776E+01 INTERVAL= 1.0000E+01
tiegcm_trunk (DAY,HR,MIN= 81, 0, 0)
tiegcm.run/tiegcm_res2.5_mareqx_smin_diags3d_sech_001.nc



Back to diagnostics table

O_N2

Diagnostic field: O/N2 RATIO:

```
diags(n)%short_name = 'O_N2'
diags(n)%long_name  = 'O/N2 RATIO'
diags(n)%units      = ' '
diags(n)%levels     = 'lev'
diags(n)%caller     = 'comp.F'
```

Note: Please note that this field is calculated at constant pressure surfaces (ln(p0/p)), and is very sensitive to fluctuations in the height of the pressure surfaces. If this field is interpolated to constant height surfaces, it will look very different than when plotted on pressure surfaces.

Note: Also note that O/N2 is a 3d field (not integrated in the vertical coordinate), and is the quotient of the mixing ratios of the species (i.e., there is no units conversion from MMR).

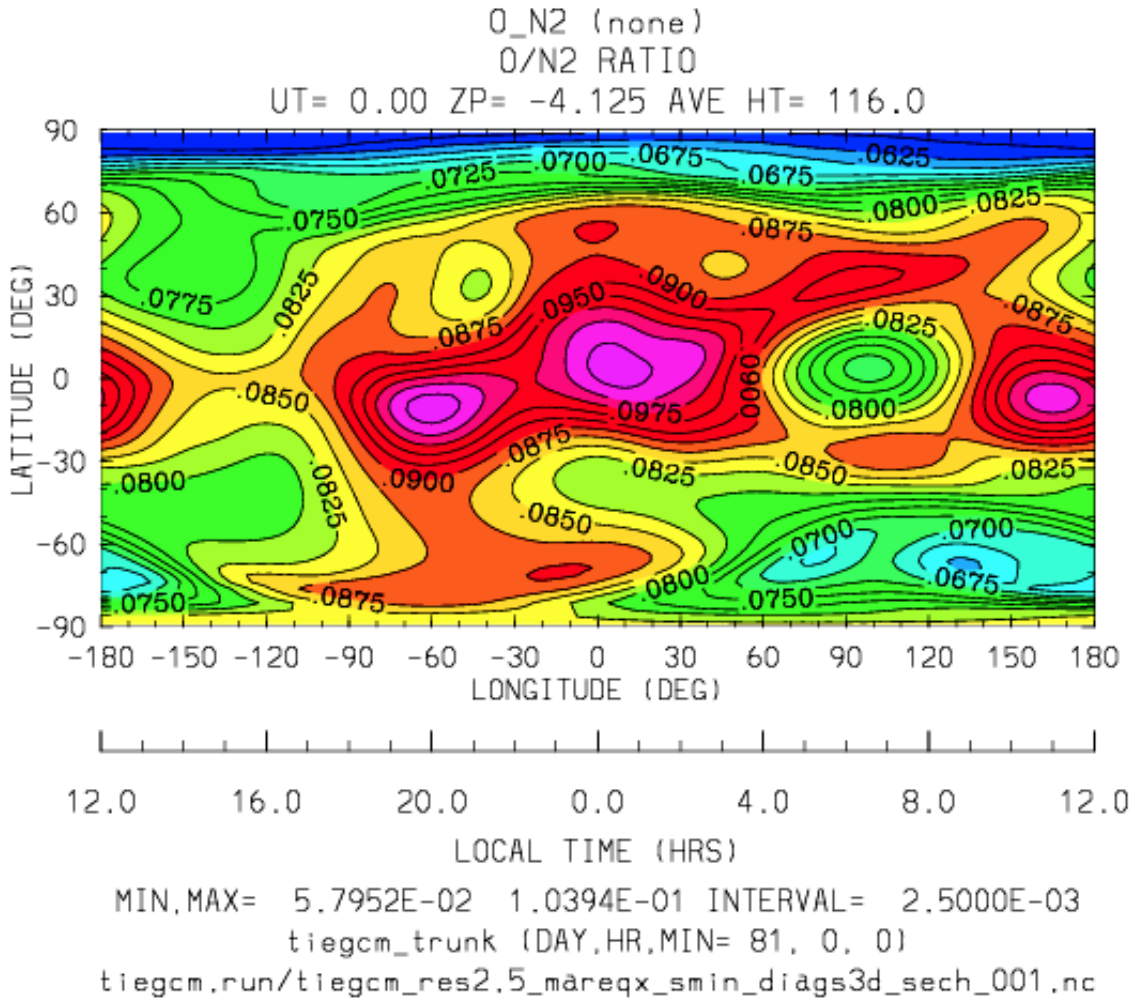
O/N2 is calculated and saved by subroutine *mkdiag_O_N2* in source file `diags.F`:

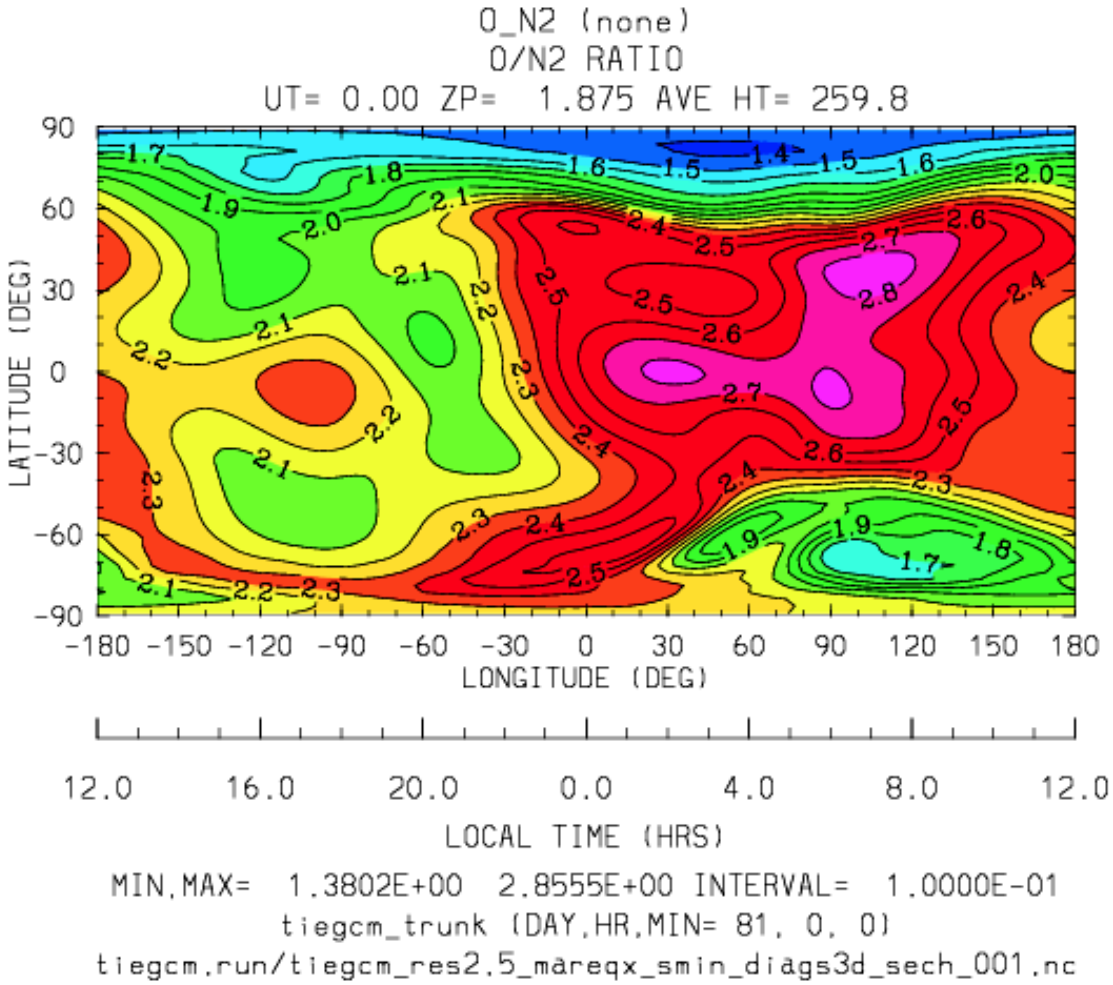
```
!-----------------------------------------------------------------------
      subroutine mkdiag_O_N2(name,o1,o2,lev0,lev1,lon0,lon1,lat)
!
! Calculate O/N2 ratio from o2 and o (mmr).
! In mass mixing ratio, this is simply o/(1-o2-o)
!
! Args:
      character(len=*),intent(in) :: name
      integer,intent(in) :: lev0,lev1,lon0,lon1,lat
      real,intent(in),dimension(lev0:lev1,lon0:lon1) :: o1,o2
!
! Local:
      integer :: ix
      real,dimension(lev0:lev1,lon0:lon1) :: n2, o_n2
!
! Check that field name is a diagnostic, and was requested:
      ix = checkf(name) ; if (ix==0) return
!
! N2 mmr:
      n2 = 1.-o2-o1
!
! O/N2 ratio:
      o_n2 = o1/n2

      call addfld(diags(ix)%short_name,diags(ix)%long_name,
     |  diags(ix)%units,o_n2,'lev',lev0,lev1,'lon',lon0,lon1,lat)

      end subroutine mkdiag_O_N2
!-----------------------------------------------------------------------
```

Called by: subroutine *comp* in source file `comp.F` as follows:

```
   call mkdiag_O_N2('O_N2',o1_upd(:,lon0:lon1,lat),
|    o2_upd(:,lon0:lon1,lat),lev0,lev1,lon0,lon1,lat)
```

Sample images: O_N2 Global maps at Zp -4, +2:


O_N2 (none)
O/N2 RATIO
UT= 0.00 ZP= -4.125 AVE HT= 116.0
LATITUDE (DEG)
90
60
30
0
-30
-60
-90
-180 -150 -120 -90 -60 -30 0 30 60 90 120 150 180
LONGITUDE (DEG)
12.0 16.0 20.0 0.0 4.0 8.0 12.0
LOCAL TIME (HRS)
MIN,MAX= 5.7952E-02 1.0394E-01 INTERVAL= 2.5000E-03
tiegcm_trunk (DAY,HR,MIN= 81, 0, 0)
tiegcm.run/tiegcm_res2.5_mareqx_smin_diags3d_sech_001.nc



Back to diagnostics table

QJOULE

Diagnostic field: Joule Heating (erg/g/s):

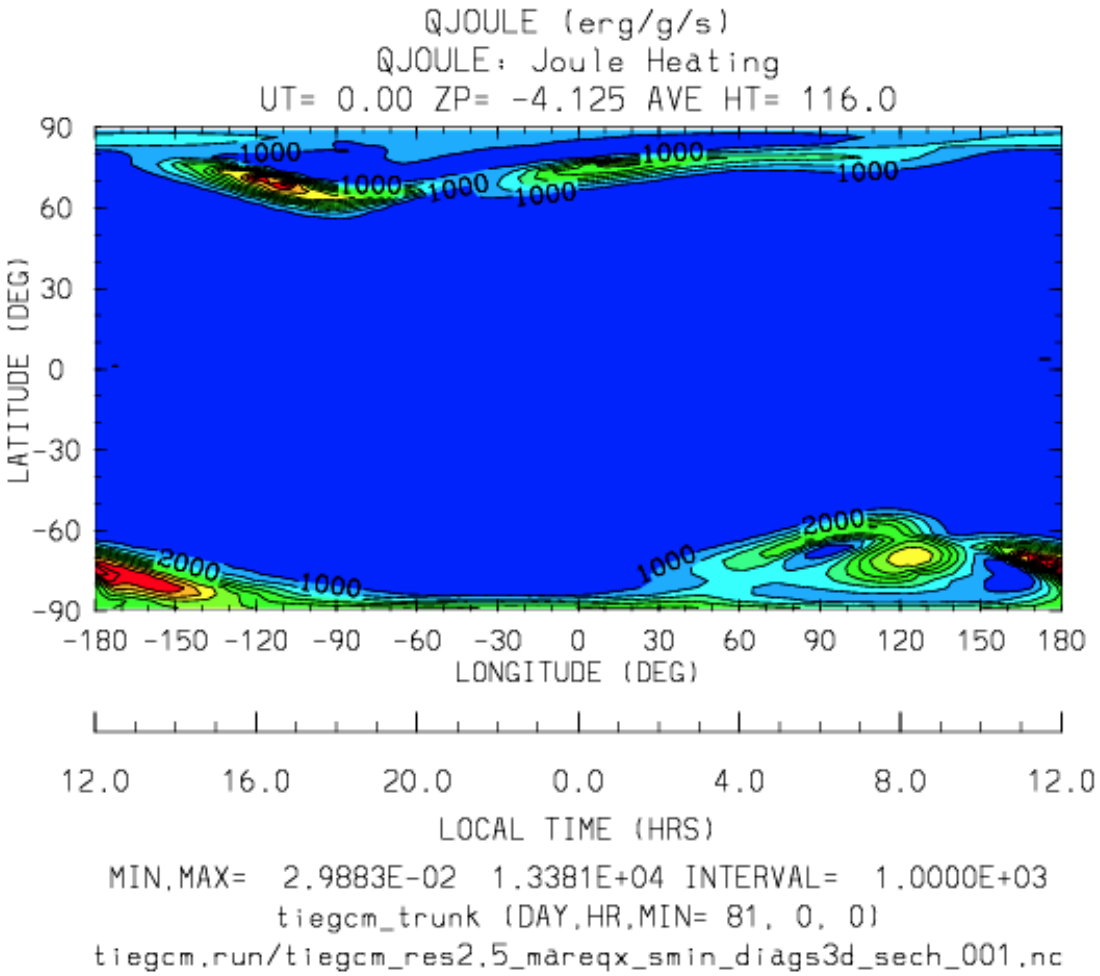
```
diags(n)%short_name = 'QJOULE'
diags(n)%long_name  = 'Joule Heating'
diags(n)%units      = 'erg/g/s'
diags(n)%levels     = 'lev'
diags(n)%caller     = 'qjoule.F'
```

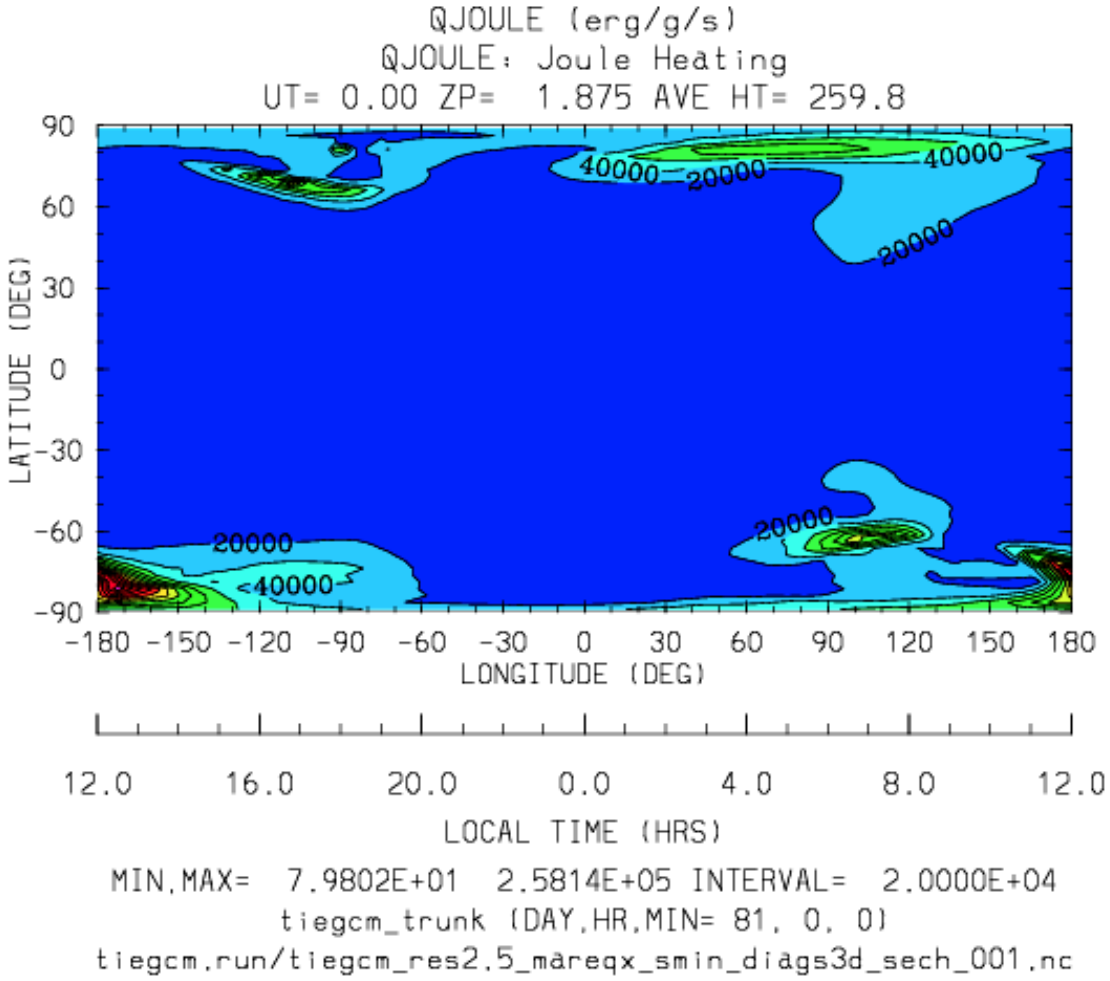
Total Joule Heating is calculated in source file `qjoule.F` as *qji_tn*, and is passed to subroutine *mkdiag_QJOULE* (`diags.F`), where it is saved to secondary histories. The following code summarizes the calculation in qjoule.F:

```
 do i=lon0,lon1
   do k=lev0,lev1-1
     scheight(k,i) = gask*tn(k,i)/
|         (.5*(barm(k,i)+barm(k+1,i))*grav)
     vel_zonal(k,i) = .5*(ui(k,i)+ui(k+1,i))-un(k,i)   ! s2
     vel_merid(k,i) = .5*(vi(k,i)+vi(k+1,i))-vn(k,i)   ! s3
     vel_vert(k,i)  = .5*(wi(k,i)+wi(k+1,i)-scheight(k,i)*
```

```
|          ( w(k,i)-w(k+1,i)) )
   enddo ! k=lev0,lev1-1
 enddo ! i=lon0,lon1
 do i=lon0,lon1
   do k=lev0,lev1-1
     qji_tn(k,i) = .5*(lam1(k,i)+lam1(k+1,i))*
|      (vel_zonal(k,i)**2 + vel_merid(k,i)**2 +
|       vel_vert(k,i)**2)
   enddo ! k=lev0,lev1-1
 enddo ! i=lon0,lon1

 call mkdiag_QJOULE('QJOULE',qji_tn,lev0,lev1,lon0,lon1,lat)
```

Sample images: QJOULE Global maps at Zp -4, +2:





Back to diagnostics table

QJOULE_INTEG

Diagnostic field: Height-integrated Joule Heating (W/m^2):

```
diags(n)%short_name = 'QJOULE_INTEG'
diags(n)%long_name  = 'Height-integrated Joule Heating'
diags(n)%units      = 'erg/cm2/s'
diags(n)%levels     = 'none'
diags(n)%caller     = 'qjoule.F'
```

Note: This field is integrated on pressure surfaces (not height), so is a 2d field. Also note it is first calculated in W/m^2, then converted to erg/g/cm2, for consistency with the model. See comment below if you would like the field to be returned in W/m^2.

Calculated and saved by subroutine *mkdiag_QJOULE_INTEG* in source file `diags.F`:

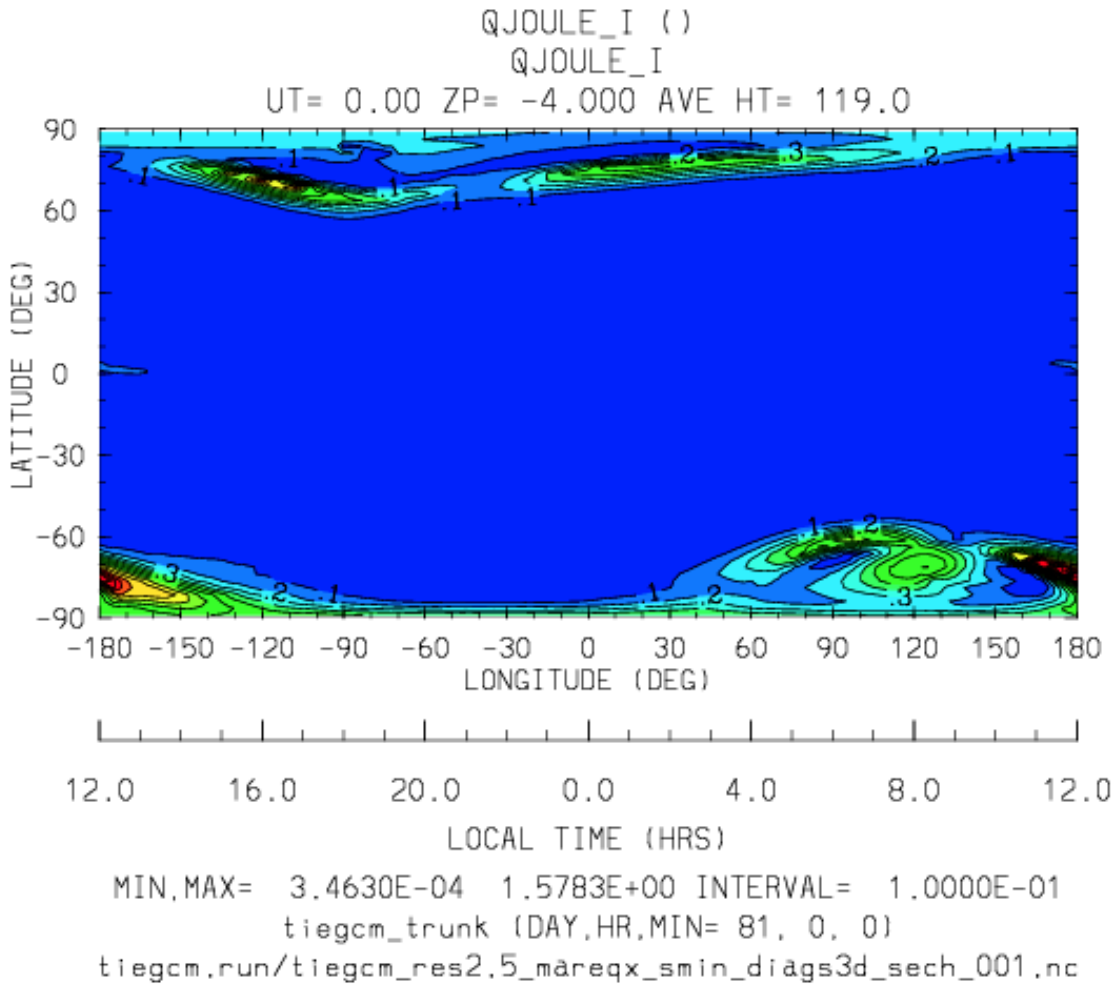
```
!-----------------------------------------------------------------------
      subroutine mkdiag_QJOULE_INTEG(name,qji_tn,lev0,lev1,lon0,lon1,
     |  lat)
```

```
      use cons_module,only: p0,grav
      use init_module,only: zpint
!
! Calculate height-integrated Joule heating (called from qjoule.F)
! This method is adapted from ncl code provided by Astrid (7/20/11)
!
! Args
      character(len=*),intent(in) :: name
      integer,intent(in) :: lev0,lev1,lon0,lon1,lat
      real,intent(in),dimension(lev0:lev1,lon0:lon1) :: qji_tn
!
! Local:
      integer :: ix,k,i
      real,dimension(lon0:lon1) :: qji_integ
      real,dimension(lev0:lev1,lon0:lon1) :: qj
      real :: myp0,mygrav
!
! Check that field name is a diagnostic, and was requested:
      ix = checkf(name) ; if (ix==0) return
!
! First integrate to get MKS units W/m^2:
! (If you want these units, comment out the below conversion to CGS)
!
      mygrav = grav*.01     ! cm/s^2 to m/s^2
      myp0 = p0*1.e-3*100.  ! to Pa
      qj = qji_tn*.0001     ! ergs/g/s to W/kg 10^(-7)*10^3

      qji_integ = 0.
      do i=lon0,lon1
        do k=lev0,lev1-1
          qji_integ(i) = qji_integ(i) + myp0/mygrav*exp(-zpint(k))*
     |      qj(k,i)*dlev
        enddo
      enddo
!
! Output in CGS units, to be consistent w/ the model:
! (note that 1 erg/cm^2/s == 1 mW/m^2)
      qji_integ = qji_integ*1000. ! W/m^2 to erg/cm^2/s
!
! Save 2d field on secondary history:
      call addfld(diags(ix)%short_name,diags(ix)%long_name,
     |  diags(ix)%units,qji_integ,'lon',lon0,lon1,'lat',lat,lat,0)

      end subroutine mkdiag_QJOULE_INTEG
!-----------------------------------------------------------------------
```

Called by: subroutine *qjoule_tn* in source file `qjoule.F` as follows:

```
 call mkdiag_QJOULE_INTEG('QJOULE_INTEG',qji_tn(:,lon0:lon1),
|  lev0,lev1,lon0,lon1,lat)
```

Sample images: QJOULE_INTEG North polar projection



Back to diagnostics table

JE13D

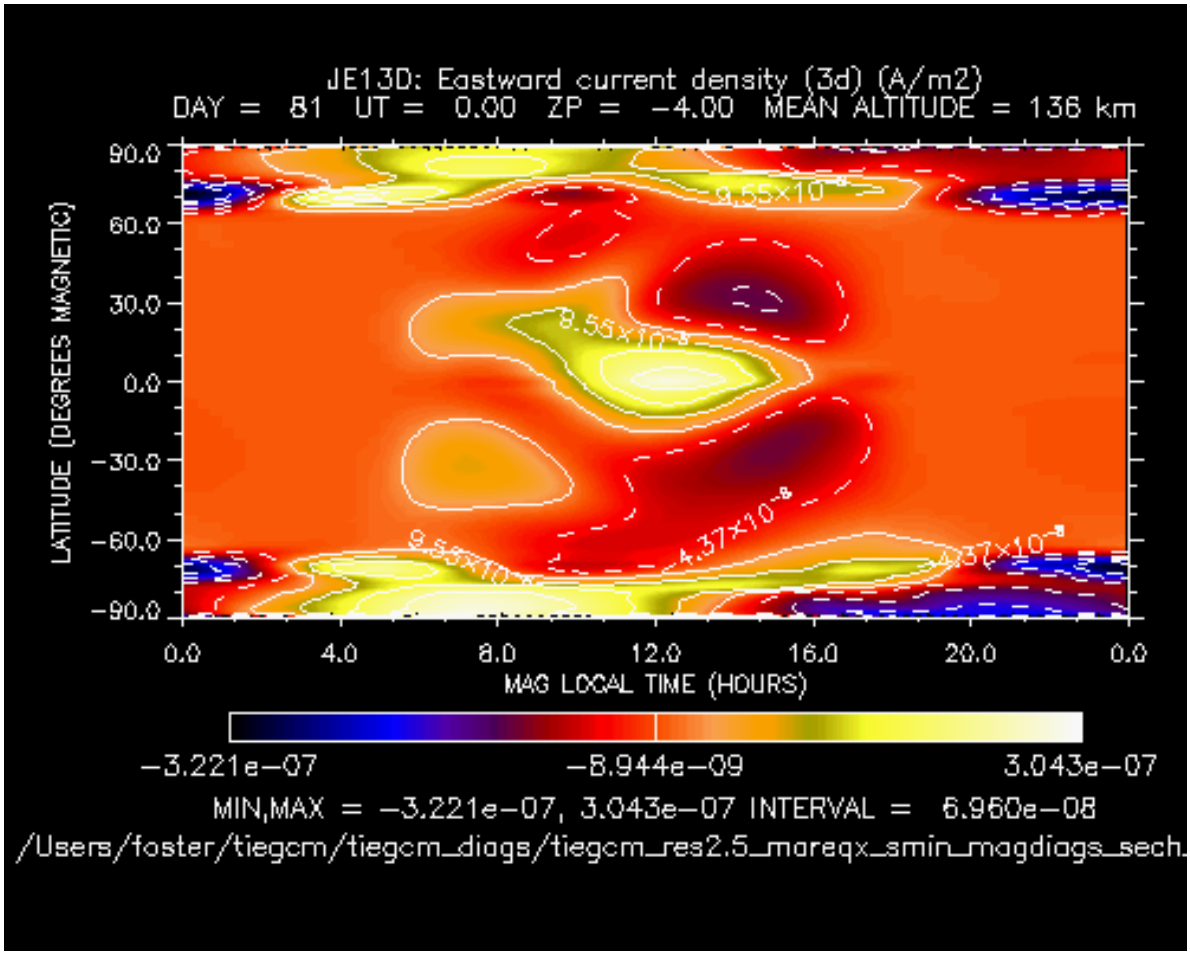
Diagnostic field: Eastward current density (A/m2) (3d on geomagnetic grid):

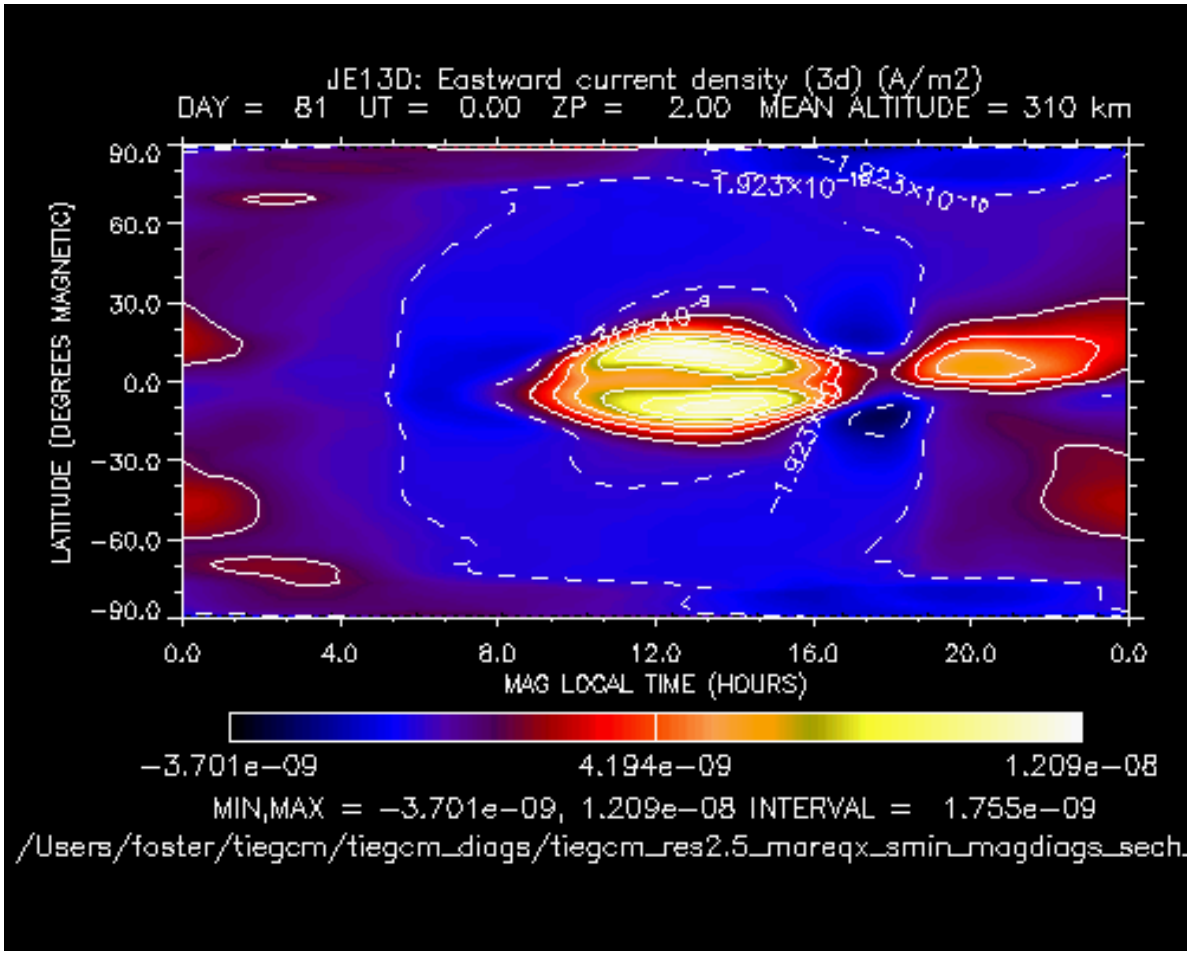
```
diags(n)%short_name = 'JE13D'
diags(n)%long_name  = 'Eastward current density (3d)'
diags(n)%units      = 'A/m2'
diags(n)%levels     = 'mlev'
diags(n)%caller     = 'current.F90'
```

Je1/D is calculated in subroutine *nosocrdens* in source file `current.F90`, and saved to secondary histories by subroutine *mkdiag_JE13D* (`diags.F`)

Note: JE13D is calculated and saved ONLY if namelist parameter CURRENT_KQ = 1 (the default is CURRENT_KQ = 0).

Sample images: JE13D North polar projection at Zp -4, +2


JE13D: Eastward current density (3d) (A/m2)
DAY = 81 UT = 0.00 ZP = -4.00 MEAN ALTITUDE = 136 km
LATITUDE (DEGREES MAGNETIC)
90.0
60.0
30.0
0.0
-30.0
-60.0
-90.0
0.0
4.0
8.0
12.0
16.0
20.0
0.0
MAG LOCAL TIME (HOURS)
-3.221e-07
-8.944e-09
3.043e-07
MIN,MAX = -3.221e-07, 3.043e-07 INTERVAL = 6.960e-08
/Users/foster/tiegcm/tiegcm_diags/tiegcm_res2.5_moreqx_smin_magdiags_sech



Back to diagnostics table

JE23D

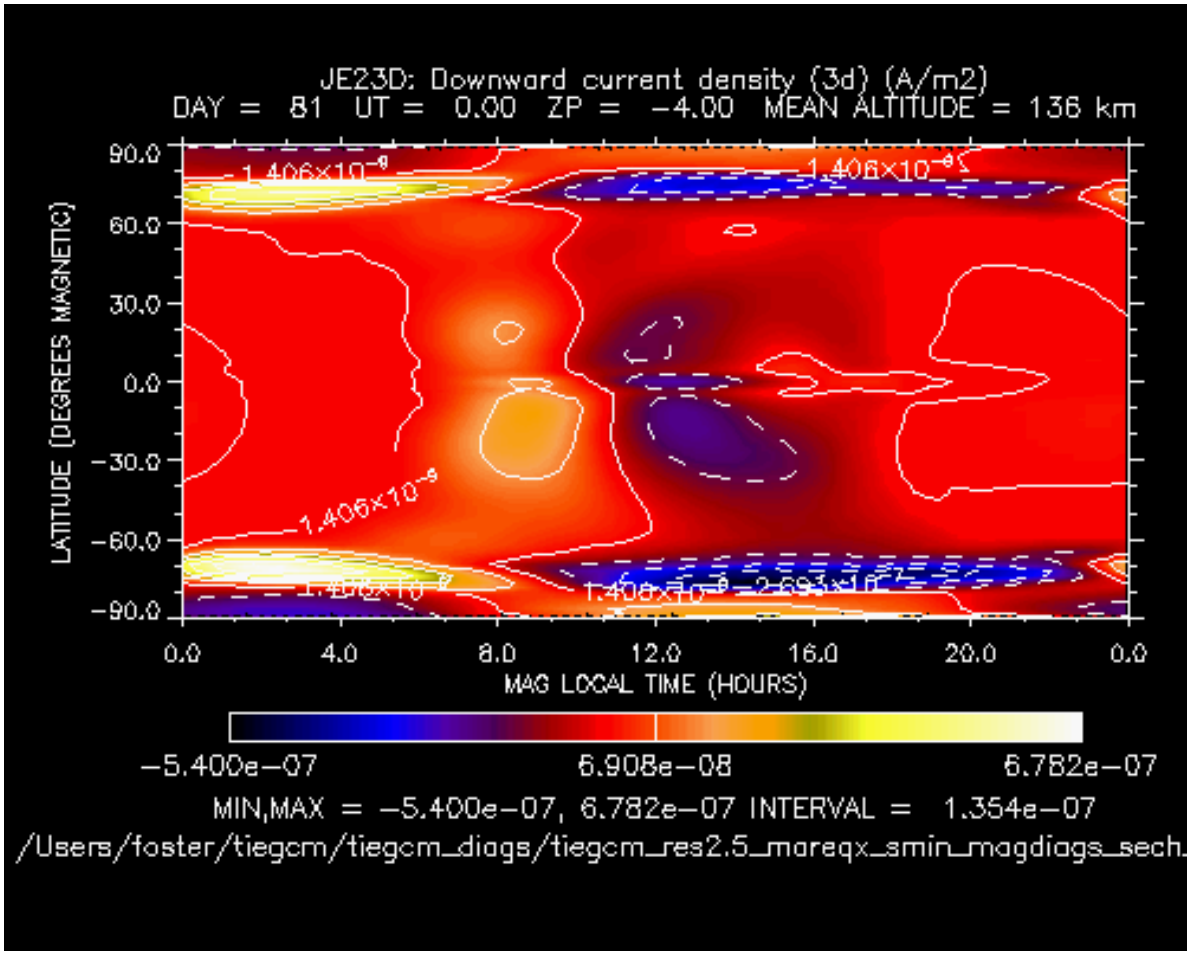
Diagnostic field: Downward current density (A/m2) (3d on geomagnetic grid):

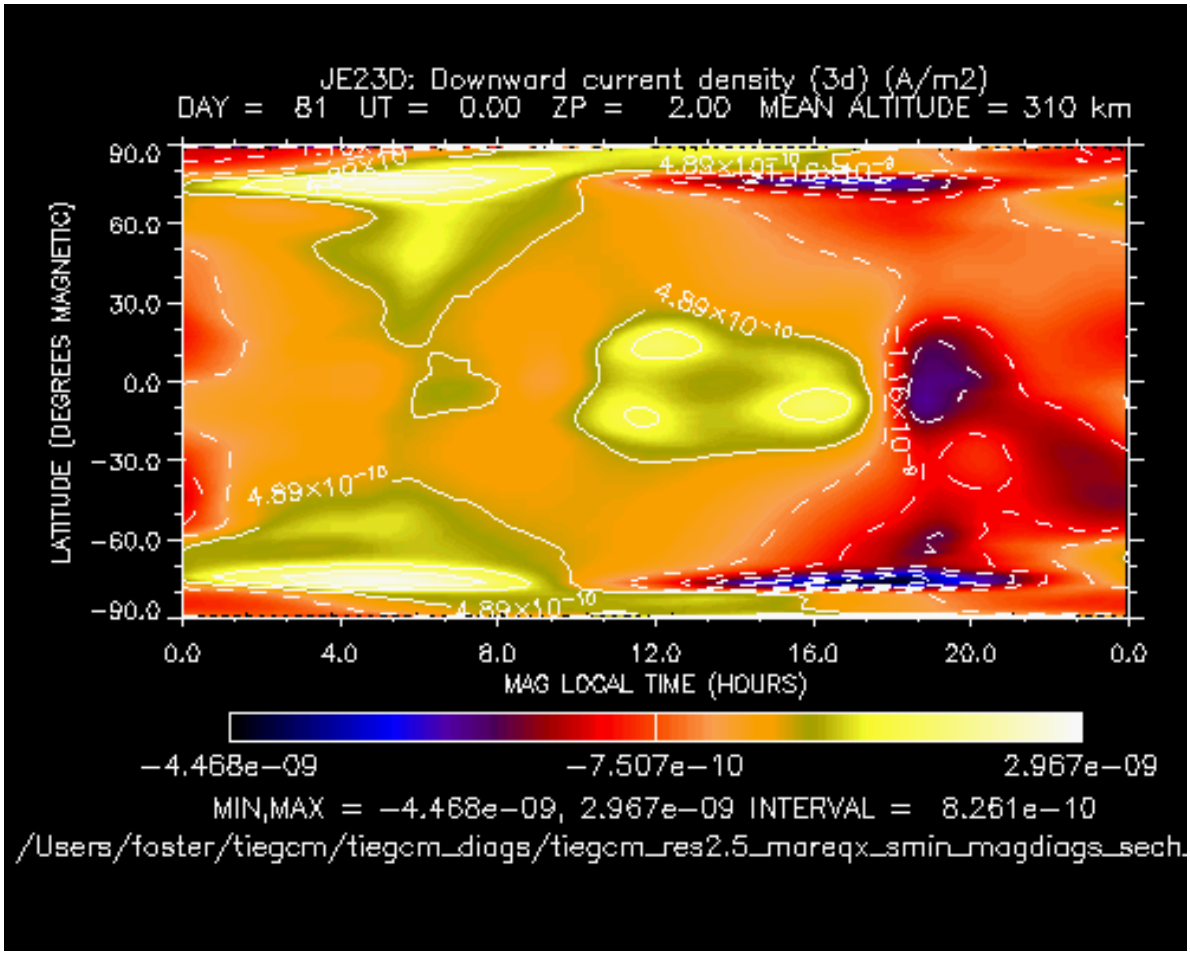
```
diags(n)%short_name = 'JE23D'
diags(n)%long_name  = 'Downward current density (3d)'
diags(n)%units      = 'A/m2'
diags(n)%levels     = 'mlev'
diags(n)%caller     = 'current.F90'
```

Je2/D is calculated in subroutine *nosocrdens* in source file `current.F90`, and saved to secondary histories by subroutine *mkdiag_JE23D* (`diags.F`)

Note: JE23D is calculated and saved ONLY if namelist parameter CURRENT_KQ = 1 (the default is CURRENT_KQ = 0).

Sample images: JE23D North polar projection at Zp -4, +2


JE23D: Downward current density (3d) (A/m2)
DAY = 81 UT = 0.00 ZP = -4.00 MEAN ALTITUDE = 136 km
LATITUDE (DEGREES MAGNETIC)
90.0
60.0
30.0
0.0
-30.0
-60.0
-90.0
0.0
4.0
8.0
12.0
16.0
20.0
0.0
MAG LOCAL TIME (HOURS)
-5.400e-07
6.908e-08
6.782e-07
MIN,MAX = -5.400e-07, 6.782e-07 INTERVAL = 1.354e-07
/Users/foster/tiegcm/tiegcm_diags/tiegcm_res2.5_moreqx_smin_magdiags_sech



Back to diagnostics table

JQR

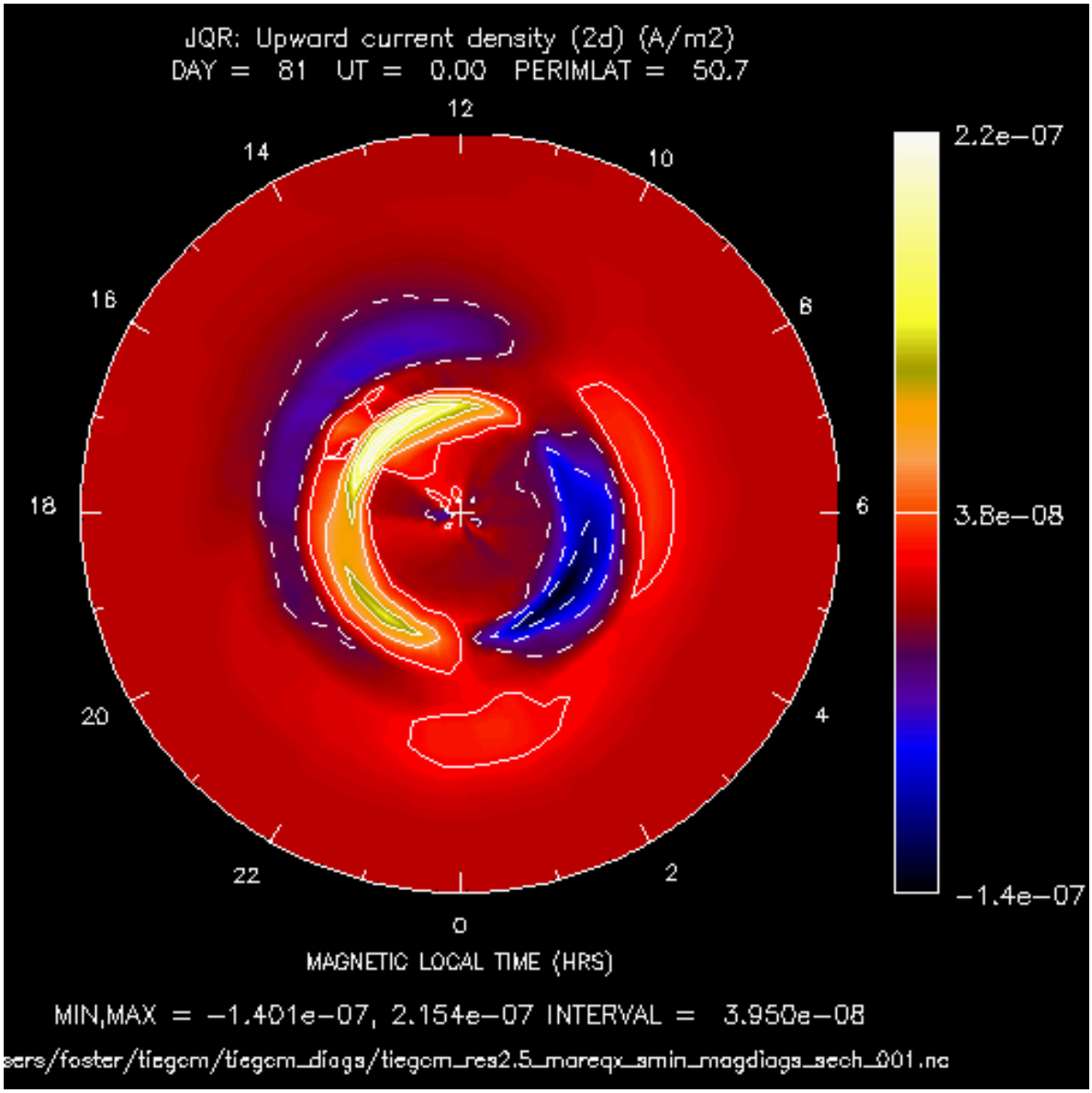
Diagnostic field: Upward current density (A/m2) (2d mlat-mlon on geomagnetic grid):

```
diags(n)%short_name = 'JQR'
diags(n)%long_name  = 'Upward current density (2d)'
diags(n)%units      = 'A/m2'
diags(n)%levels     = 'none'
diags(n)%caller     = 'current.F90'
```

Jqr is calculated in subroutine *nosocrrt* in source file `current.F90`, and saved to secondary histories by subroutine *mkdiag_JQR* (`diags.F`)

Note: JQR is calculated and saved ONLY if namelist parameter CURRENT_KQ = 1 (the default is CURRENT_KQ = 0).

Sample images: JQR North polar projection



Back to diagnostics table

KQLAM

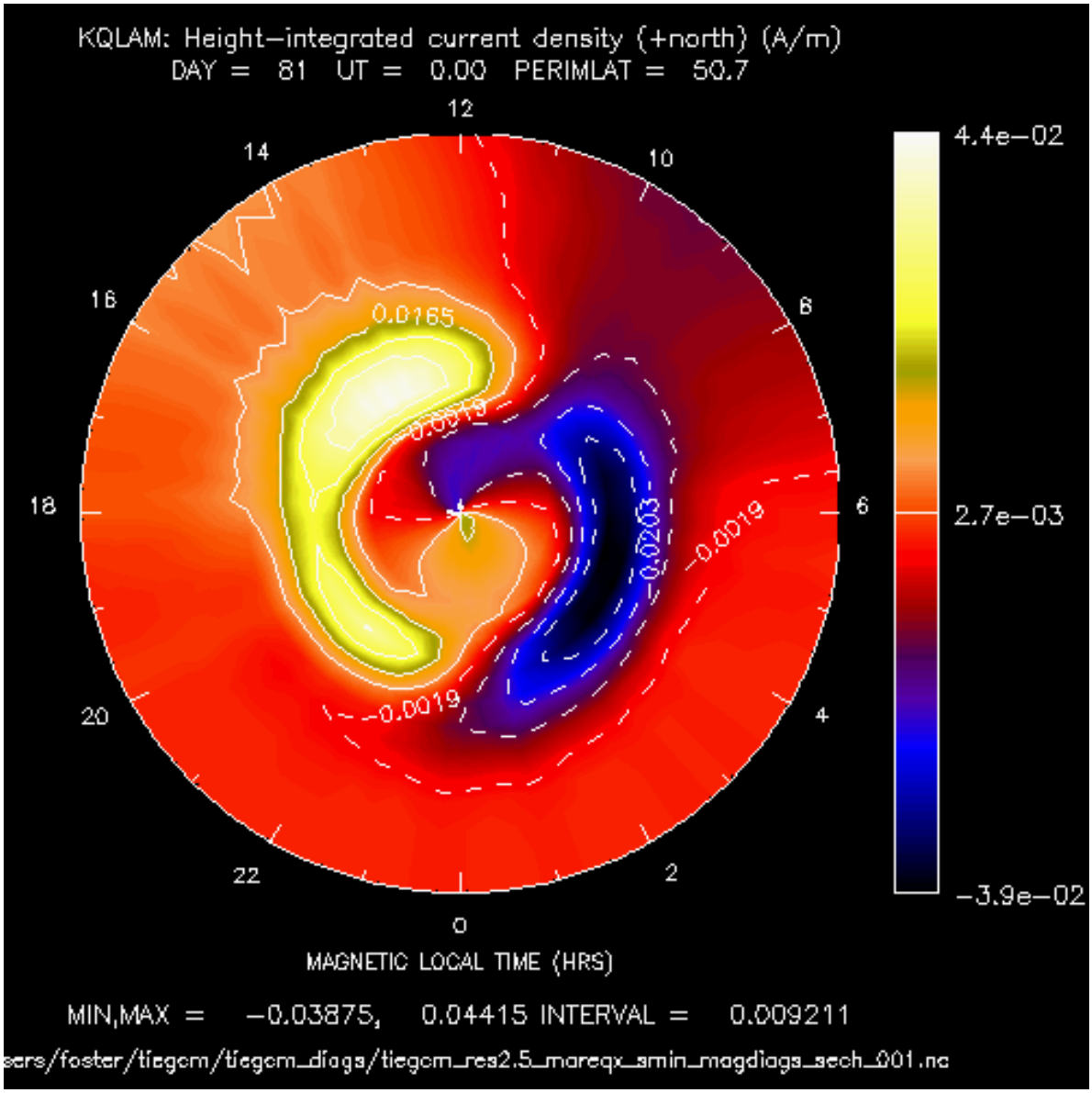
Diagnostic field: Height-integrated current density (+north) (A/m2) (2d mlat-mlon on geomagnetic grid):

```
diags(n)%short_name = 'KQLAM'
diags(n)%long_name  = 'Height-integrated current density (+north)'
diags(n)%units      = 'A/m'
diags(n)%levels     = 'none'
diags(n)%caller     = 'current.F90'
```

Kqlam is calculated in subroutine *nosocrdens* in source file `current.F90`, and saved to secondary histories by subroutine *mkdiag_KQLAM* (`diags.F`)

Note: KQLAM is calculated and saved ONLY if namelist parameter CURRENT_KQ = 1 (the default is CURRENT_KQ = 0).

Sample images: KQLAM North polar projection



Back to diagnostics table

KQPHI

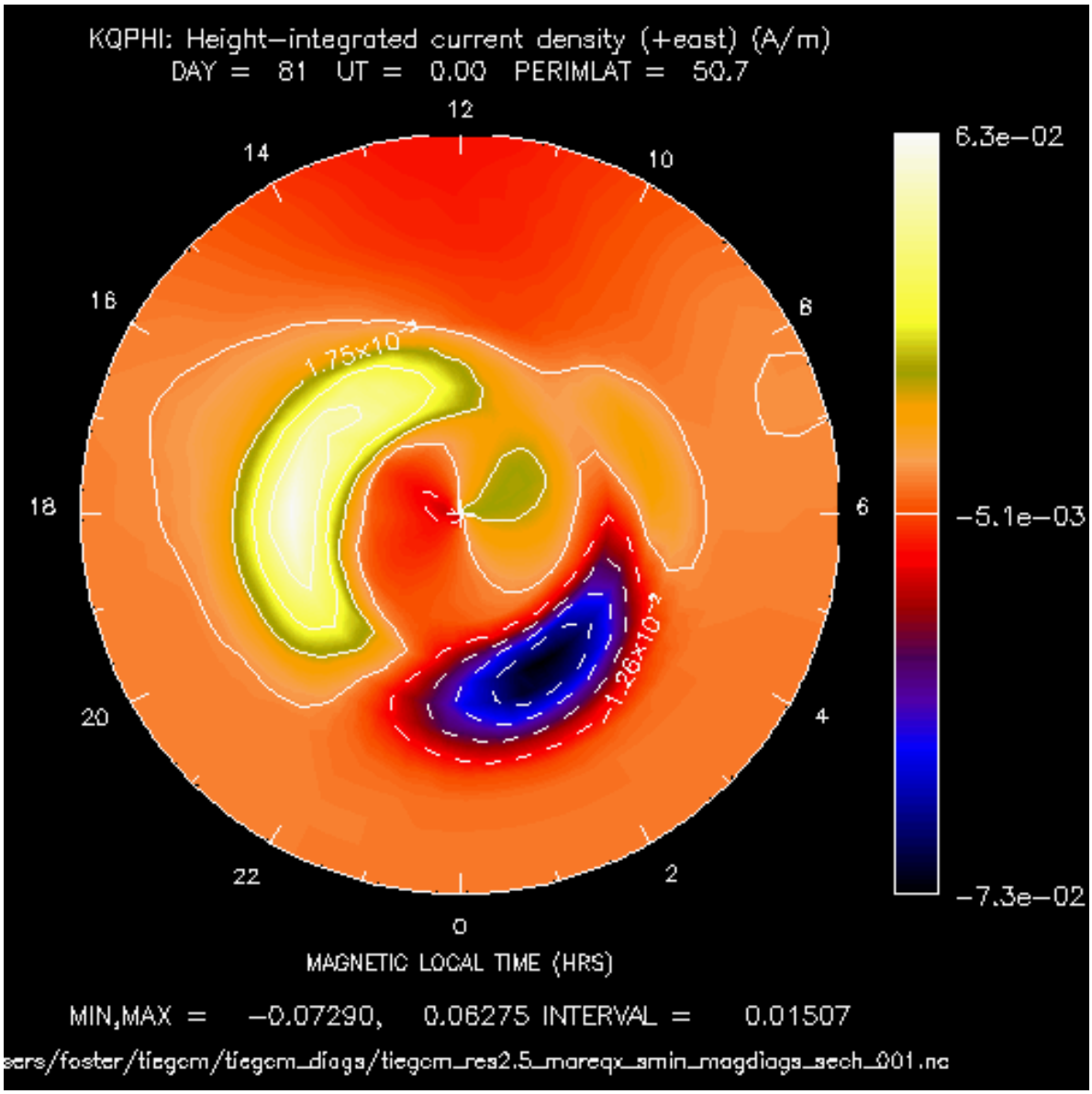
Diagnostic field: Height-integrated current density (A/m2) (2d mlat-mlon on geomagnetic grid):

```
diags(n)%short_name = 'KQPHI'
diags(n)%long_name  = 'Height-integrated current density (+east)'
diags(n)%units      = 'A/m'
diags(n)%levels     = 'none'
diags(n)%caller     = 'current.F90'
```

KQPHI is calculated in subroutine *nosocrdens* in source file `current.F90`, and saved to secondary histories by subroutine *mkdiag_KQPHI* (`diags.F`)

Note: KQLAM is calculated and saved ONLY if namelist parameter CURRENT_KQ = 1 (the default is CURRENT_KQ = 0).

Sample images: KQPHI North polar projection



Back to diagnostics table

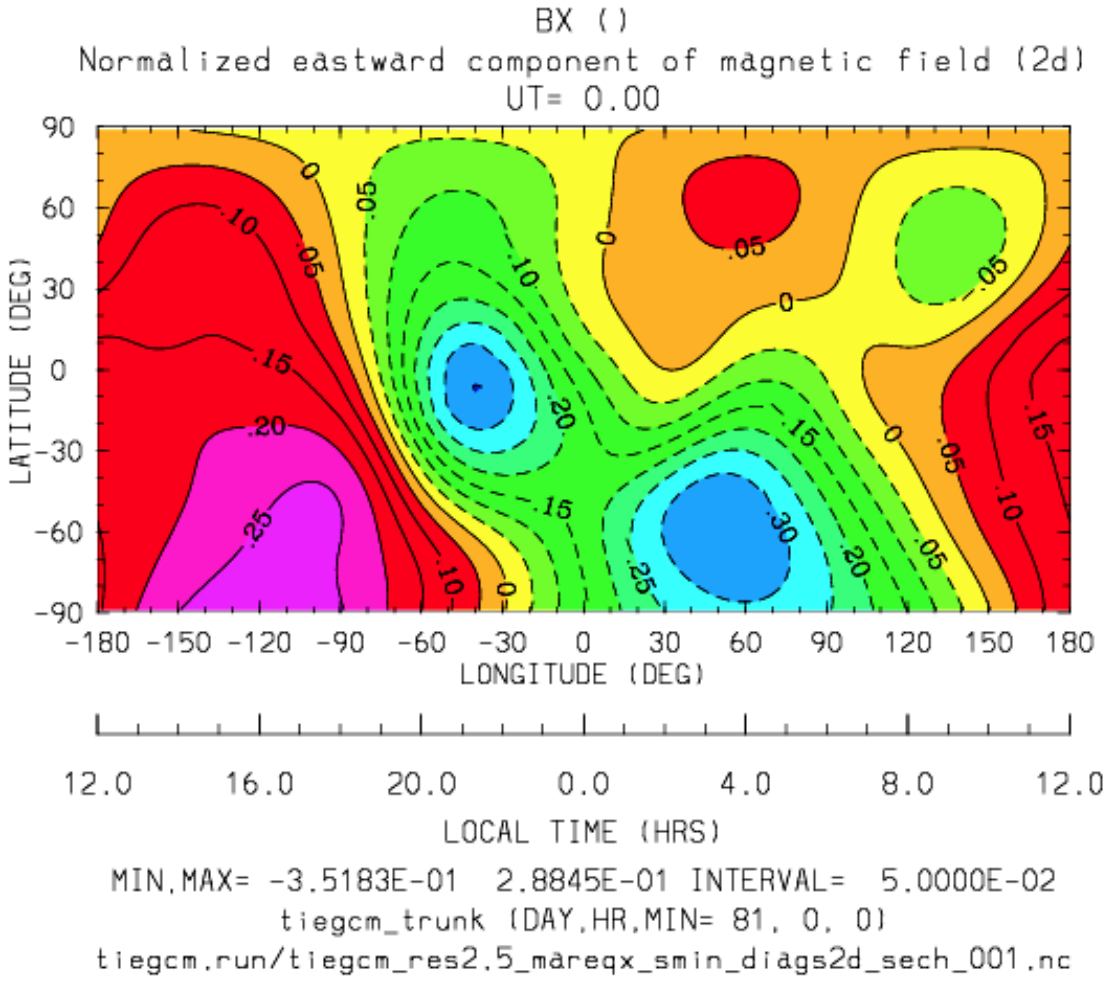
BX

Diagnostic field: Normalized eastward component of magnetic field (BX/BMAG) (2d lat-lon on geographic grid):

```
diags(n)%short_name = 'BX'
diags(n)%long_name  = 'BX/BMAG: Normalized eastward component of magnetic field'
diags(n)%units      = 'none'
```

```
diags(n)%levels      = 'none'
diags(n)%caller      = 'oplus.F'
```

Sample images: BX cylindrical equidistant projection

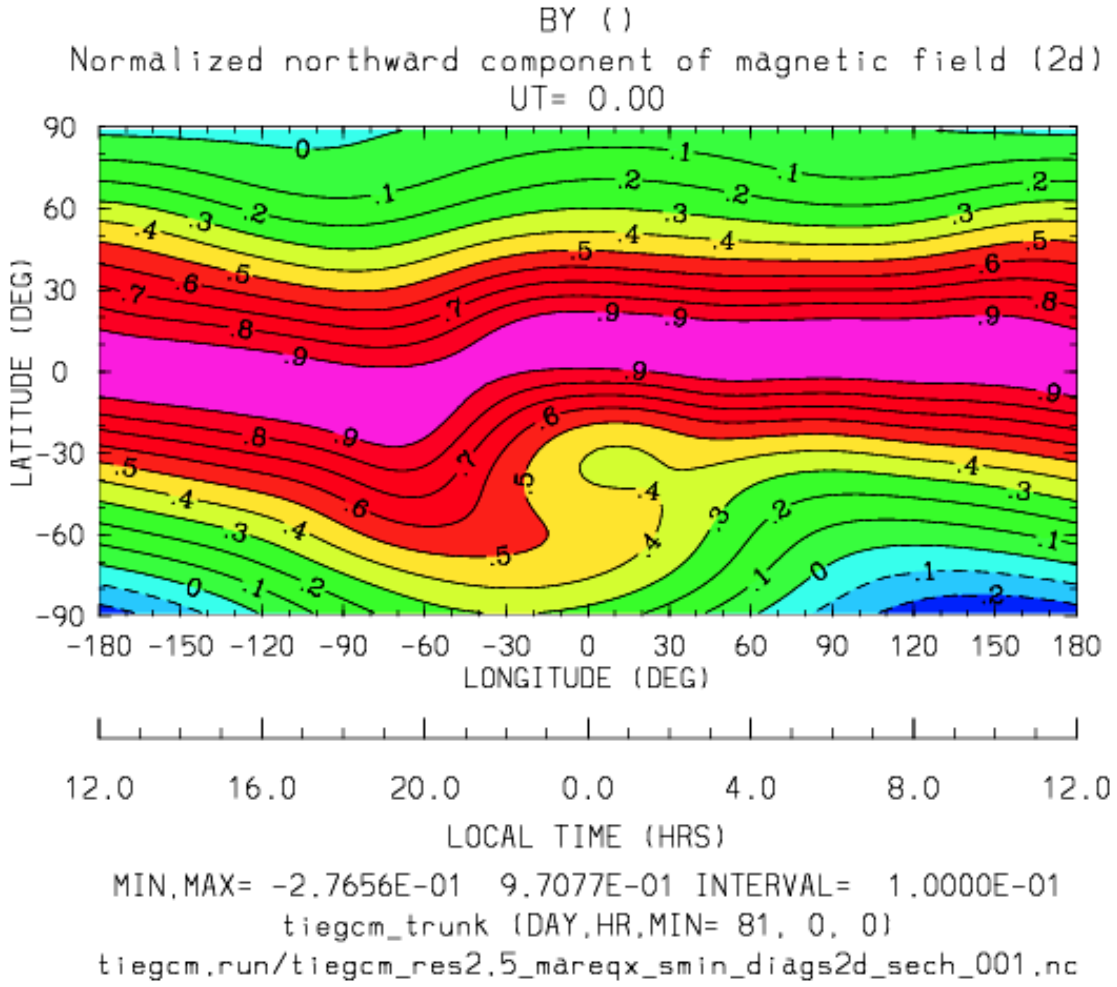


BY

Diagnostic field: Normalized eastward component of magnetic field (BY/BMAG) (2d lat-lon on geographic grid):

```
diags(n)%short_name  = 'BY'
diags(n)%long_name   = 'BY/BMAG: Normalized northward component of magnetic field'
diags(n)%units       = 'none'
diags(n)%levels      = 'none'
diags(n)%caller      = 'oplus.F'
```

Sample images: BY cylindrical equidistant projection

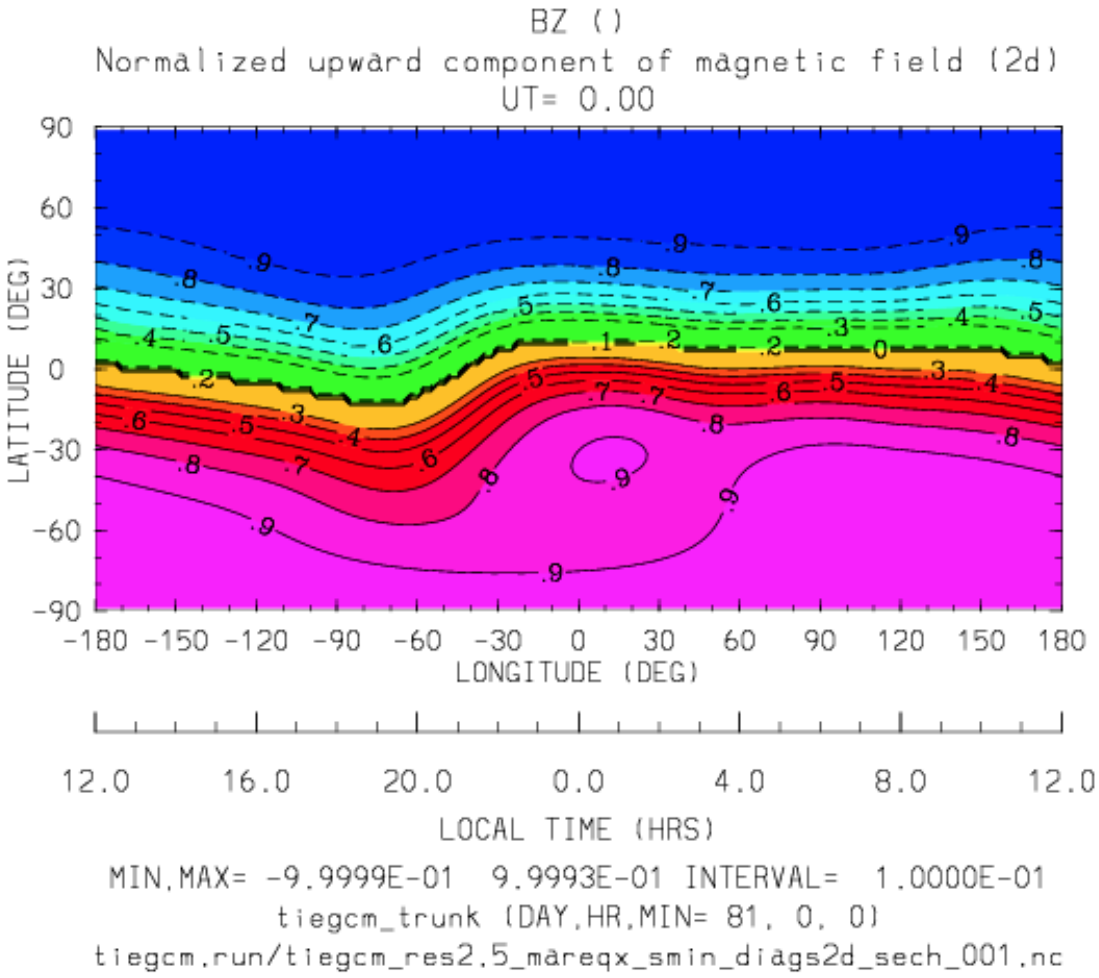


BZ

Diagnostic field: Normalized upward component of magnetic field (BZ/BMAG) (2d lat-lon on geographic grid):

```
diags(n)%short_name = 'BZ'
diags(n)%long_name  = 'BZ/BMAG: Normalized northward component of magnetic field'
diags(n)%units      = 'none'
diags(n)%levels     = 'none'
diags(n)%caller     = 'oplus.F'
```

Sample images: BZ cylindrical equidistant projection

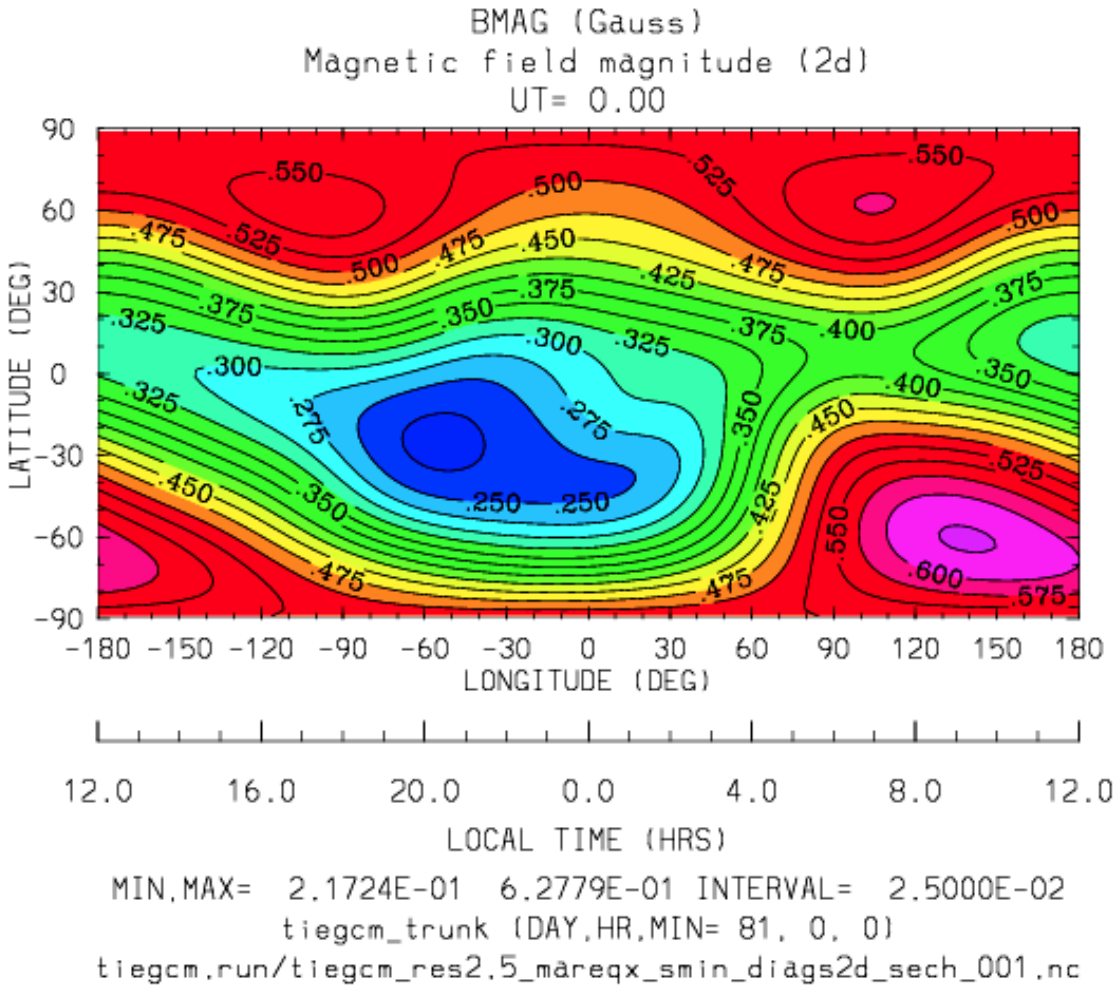


BMAG

Diagnostic field: Magnetic Field Magnitude (2d lat-lon on geographic grid):

```
diags(n)%short_name = 'BMAG'
diags(n)%long_name  = 'BMAG: Magnetic Field Magnitude'
diags(n)%units      = 'Gauss'
diags(n)%levels     = 'none'
diags(n)%caller     = 'oplus.F'
```

Sample images: BMAG cylindrical equidistant projection

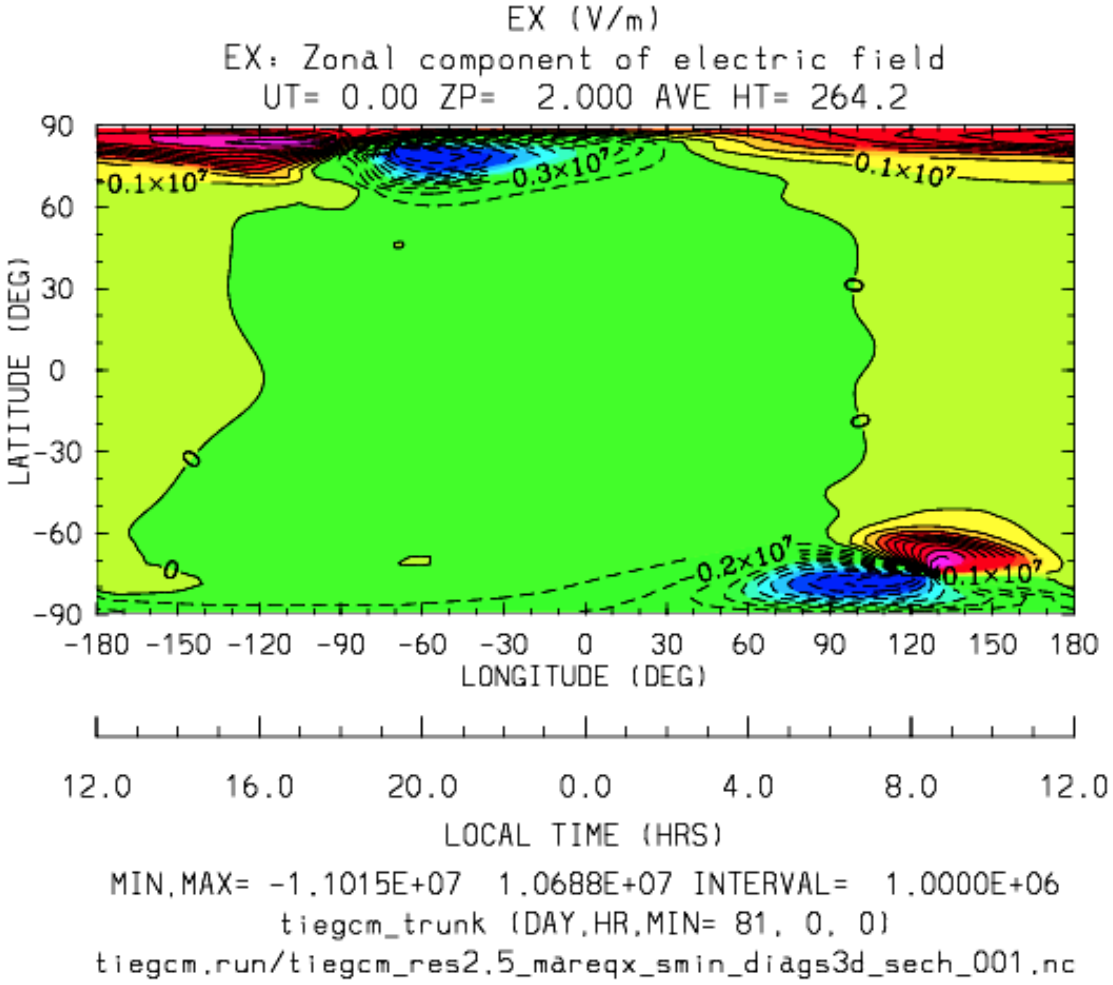


EX

Diagnostic field: Zonal Component of Electric Field (3d lat-lon on geographic grid):

```
diags(n)%short_name = 'EX'
diags(n)%long_name  = 'EX: Zonal Component of Electric Field'
diags(n)%units      = 'V/m'
diags(n)%levels     = 'ilev'
diags(n)%caller     = 'ionvel.F'
```

Sample images: EX cylindrical equidistant projection

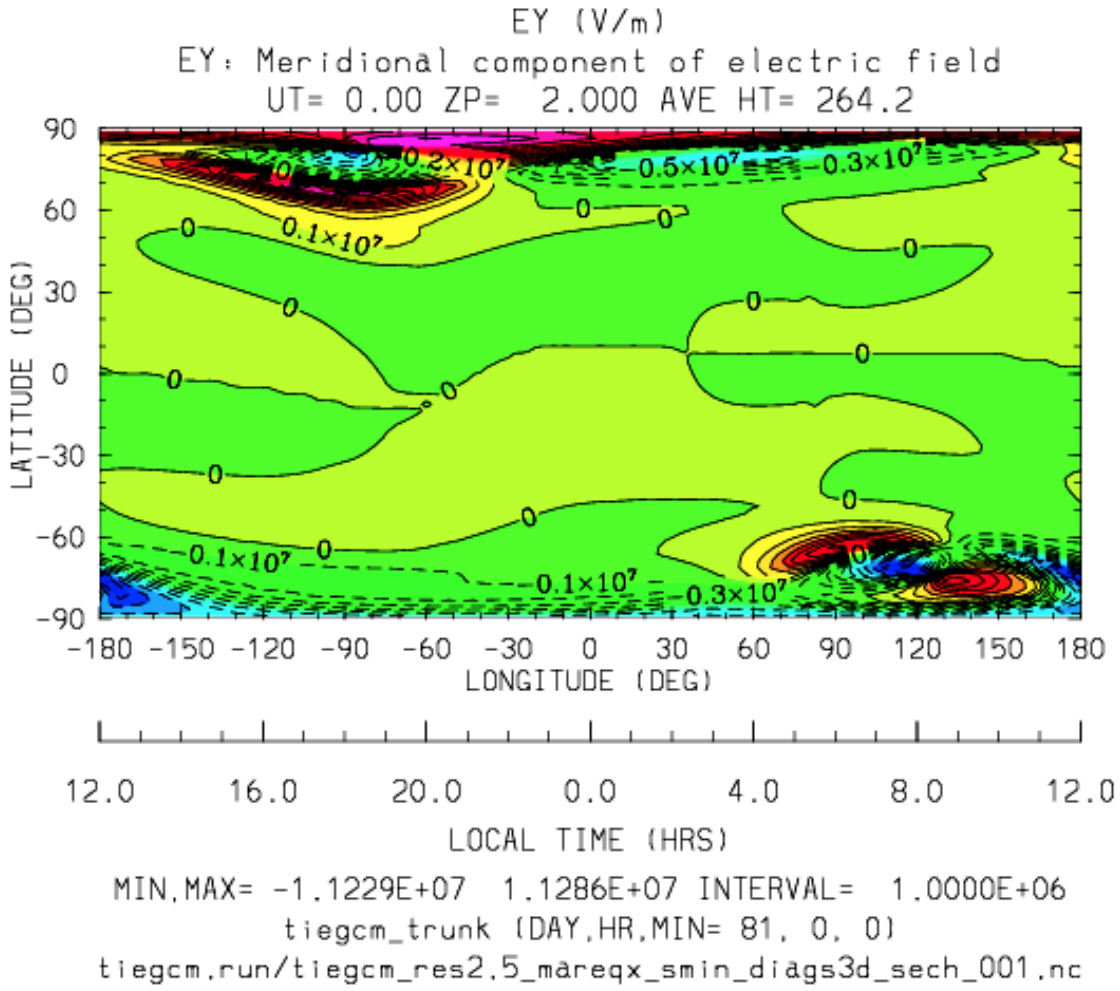


EY

Diagnostic field: Meridional Component of Electric Field (3d lat-lon on geographic grid):

```
diags(n)%short_name = 'EY'
diags(n)%long_name  = 'EY: Meridional Component of Electric Field'
diags(n)%units      = 'V/m'
diags(n)%levels     = 'ilev'
diags(n)%caller     = 'ionvel.F'
```

Sample images: EY cylindrical equidistant projection

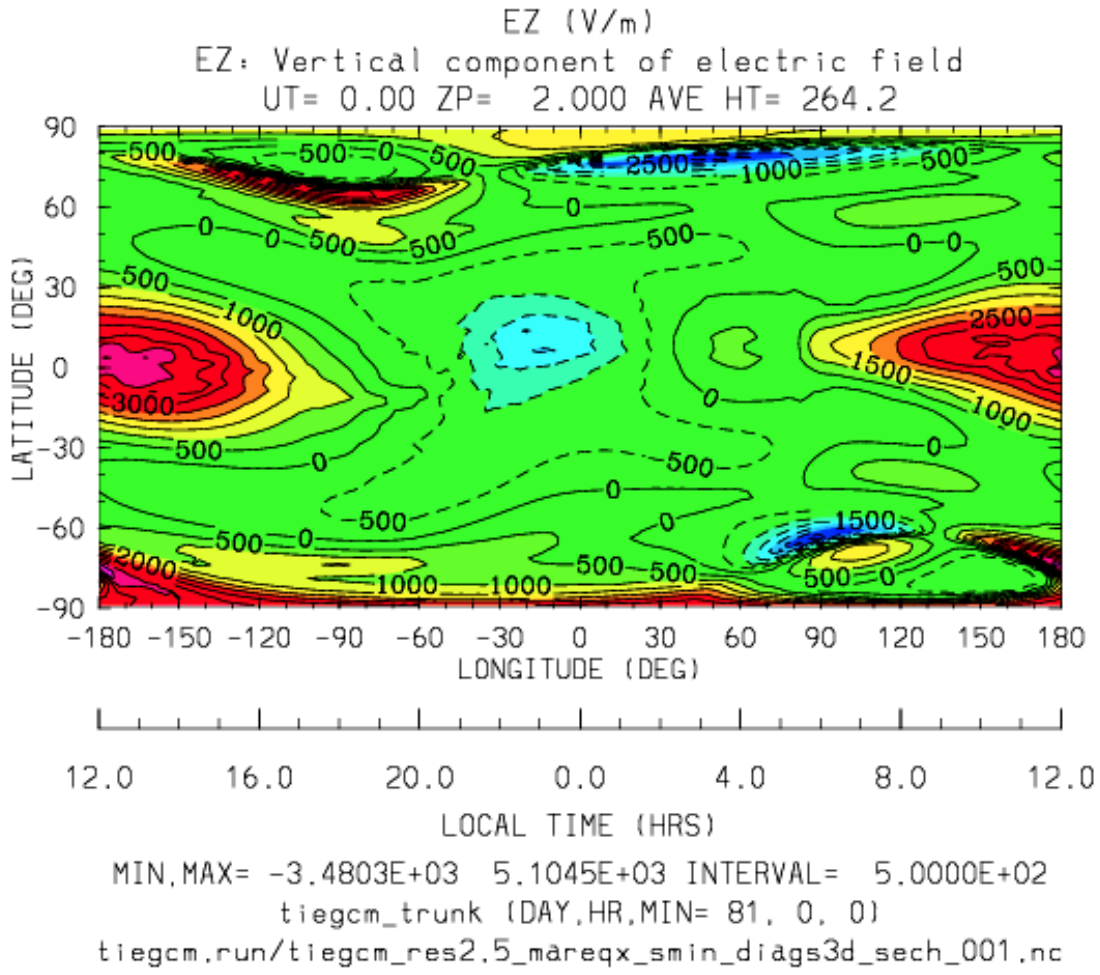


EZ

Diagnostic field: Vertical Component of Electric Field (3d lat-lon on geographic grid):

```
diags(n)%short_name = 'EZ'
diags(n)%long_name  = 'EZ: Vertical Component of Electric Field'
diags(n)%units      = 'V/m'
diags(n)%levels     = 'ilev'
diags(n)%caller     = 'ionvel.F'
```

Sample images: EZ cylindrical equidistant projection

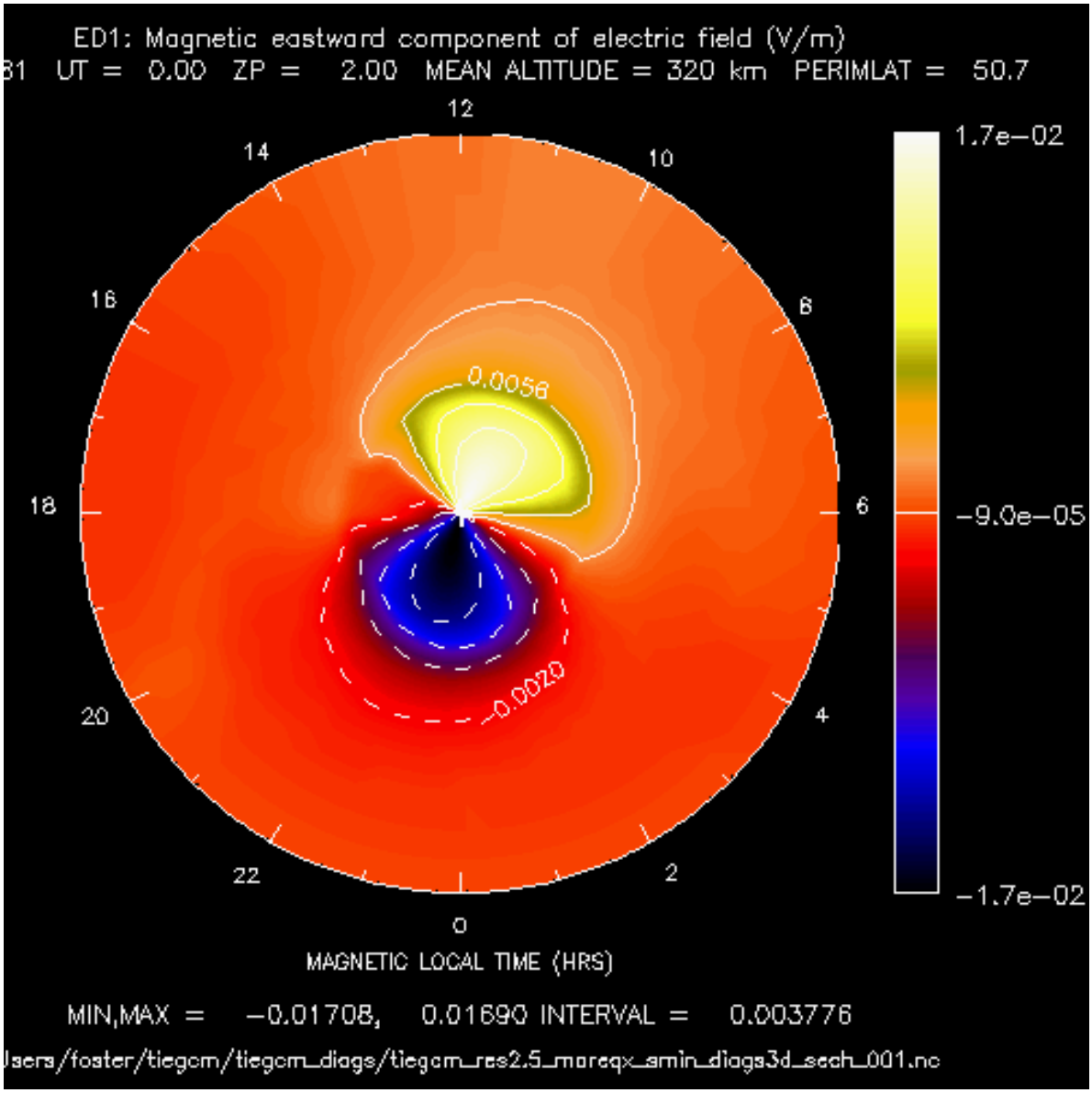


ED1

Diagnostic field: Magnetic Eastward Component of Electric Field (3d mlat-mlon on geomagnetic grid):

```
diags(n)%short_name = 'ED1'
diags(n)%long_name  = 'ED1: Magnetic Eastward Component of Electric Field'
diags(n)%units      = 'V/m'
diags(n)%levels     = 'imlev'
diags(n)%caller     = 'dynamo.F'
```

Sample images: ED1 north hemisphere polar projection

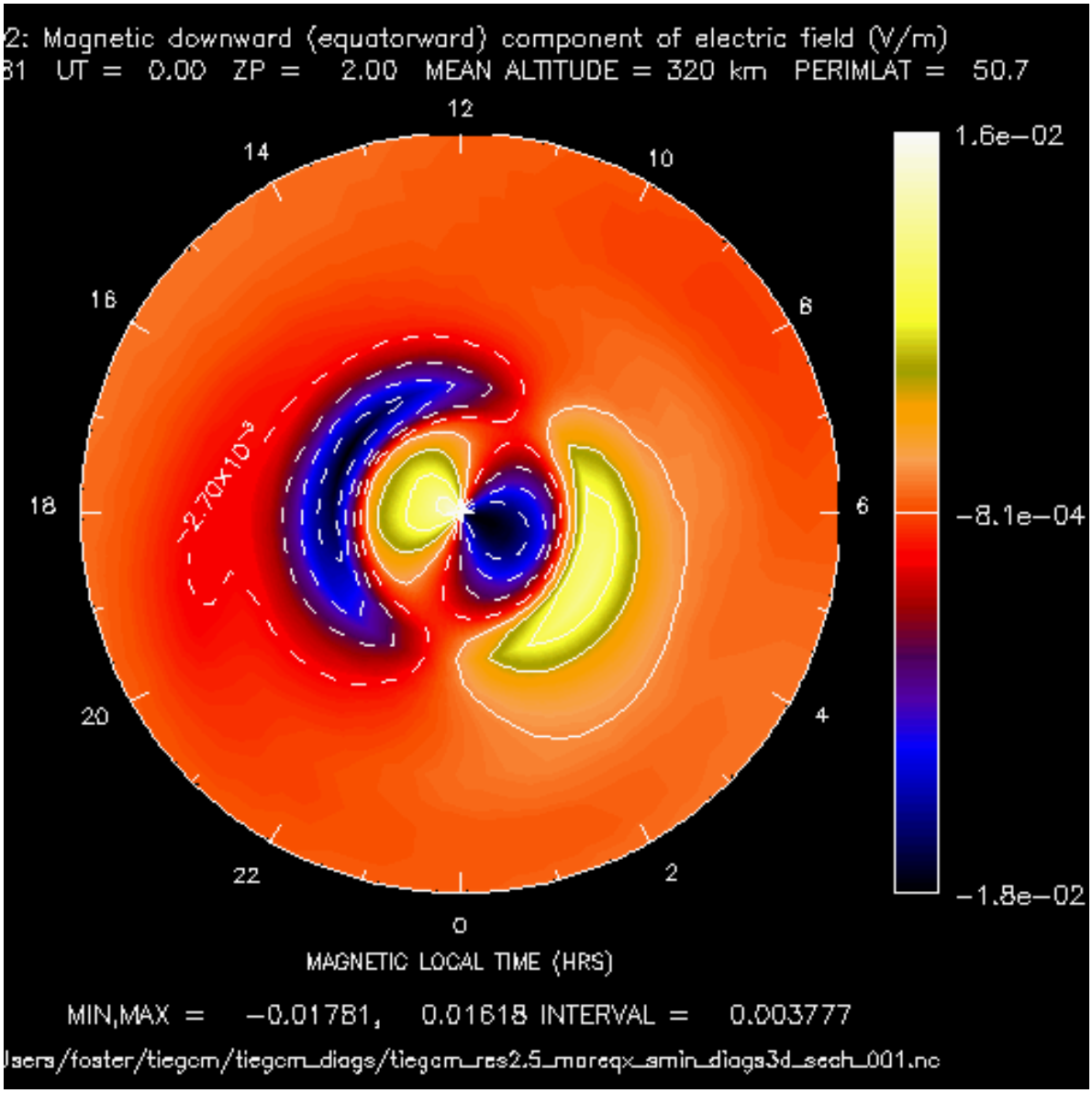


ED2

Diagnostic field: Magnetic Eastward Component of Electric Field (3d mlat-mlon on geomagnetic grid):

```
diags(n)%short_name = 'ED2'
diags(n)%long_name  = 'ED2: Magnetic Downward (Equatorward) Component of Electric Field'
diags(n)%units      = 'V/m'
diags(n)%levels     = 'imlev'
diags(n)%caller     = 'dynamo.F'
```

Sample images: ED2 north hemisphere polar projection

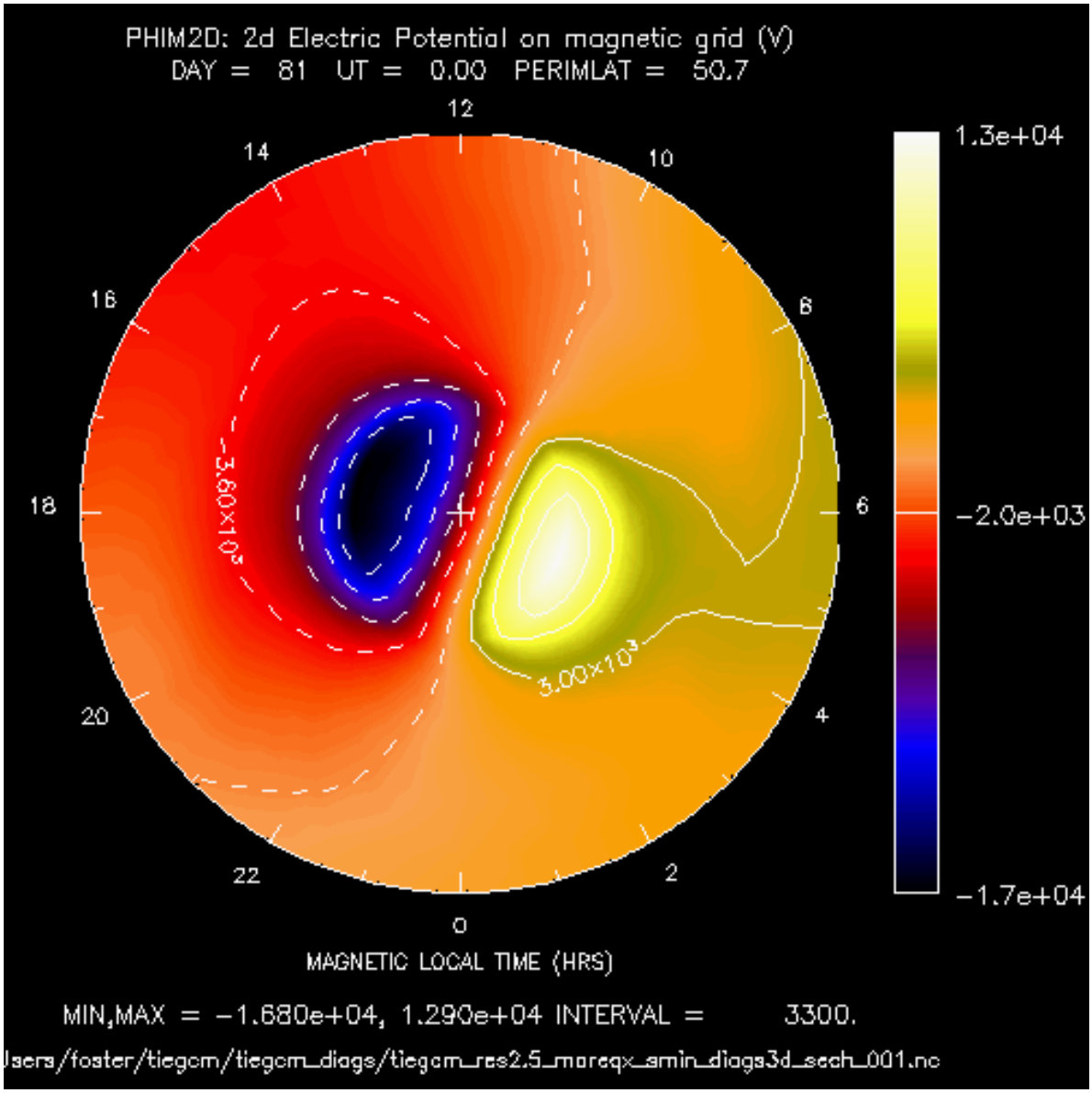


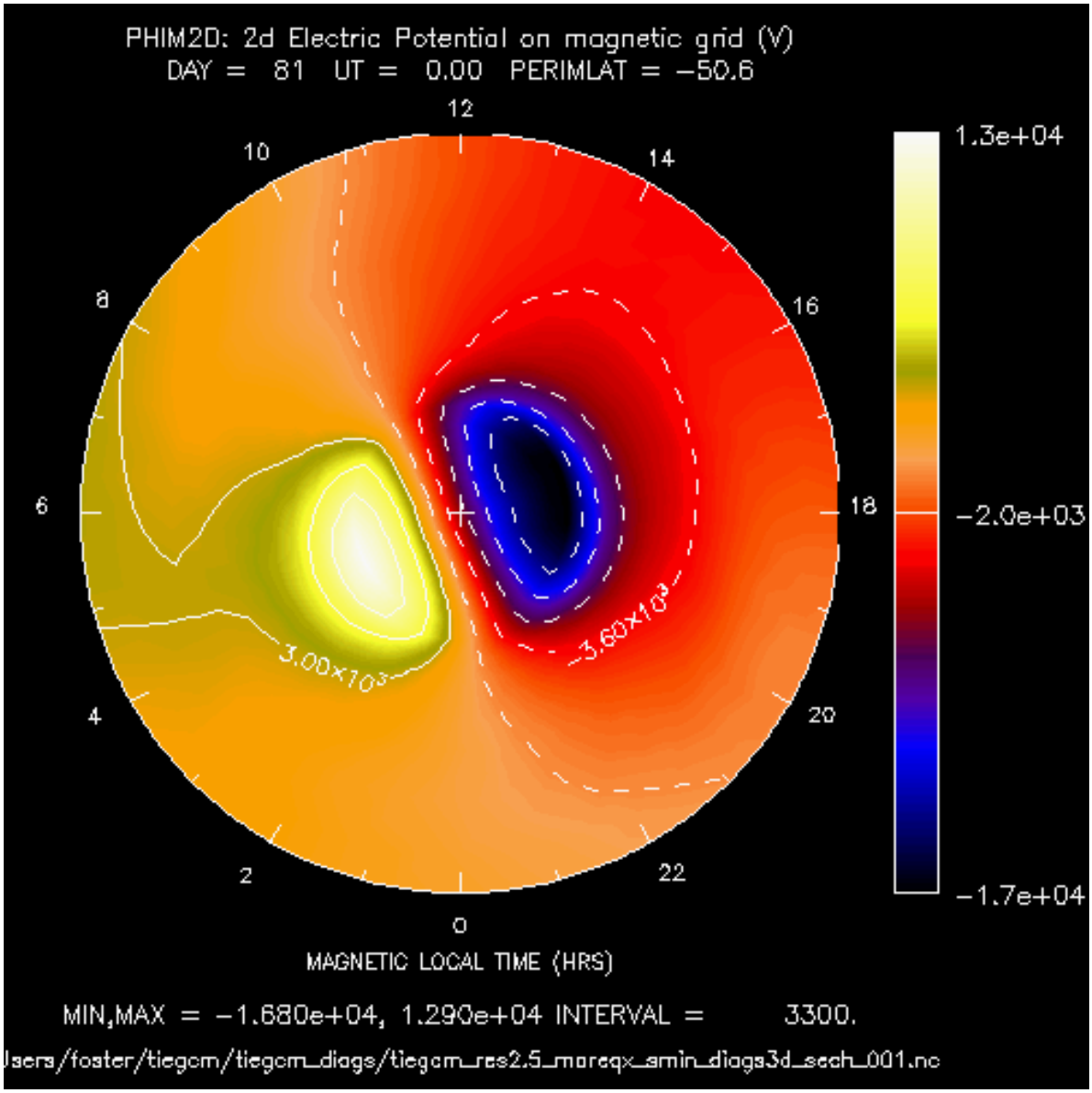
PHIM2D

Diagnostic field: 2d Electric Potential on Magnetic Grid (3d mlat-mlon on geomagnetic grid):

```
diags(n)%short_name = 'PHIM2D'
diags(n)%long_name  = 'PHIM2D: 2d Electric Potential on Magnetic Grid'
diags(n)%units      = 'V/m'
diags(n)%levels     = 'none'
diags(n)%caller     = 'dynamo.F'
```

Sample images: PHIM2D polar projection (north and south):


PHIM2D: 2d Electric Potential on magnetic grid (V)
DAY = 81 UT = 0.00 PERIMLAT = 50.7
12
14
10
16
8
18
6
20
4
22
2
0
-3.60x10³
3.00x10³
1.3e+04
-2.0e+03
-1.7e+04
MAGNETIC LOCAL TIME (HRS)
MIN,MAX = -1.680e+04, 1.290e+04 INTERVAL = 3300.
Users/foster/tiegcm/tiegcm_diags/tiegcm_res2.5_moreqx_amin_diags3d_sech_001.nc



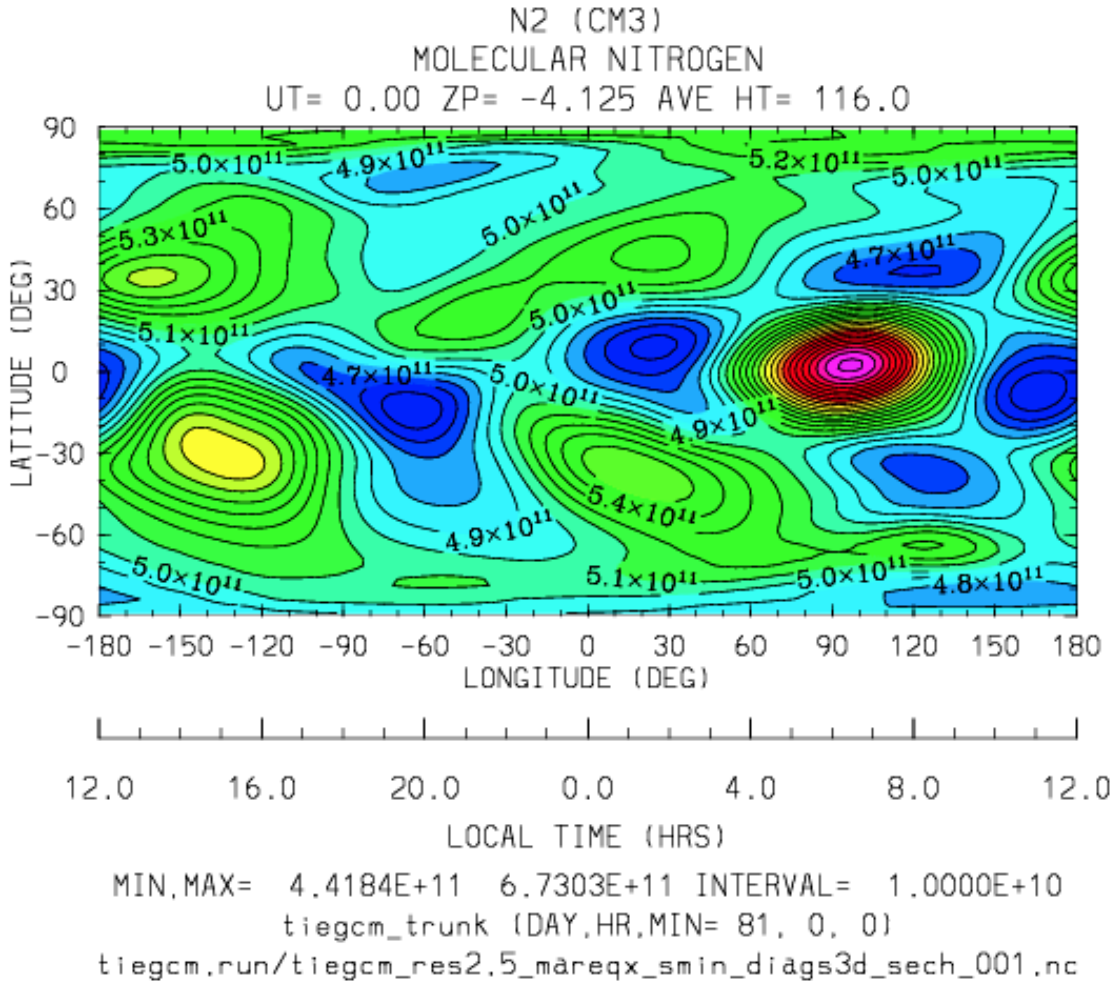
Back to diagnostics table

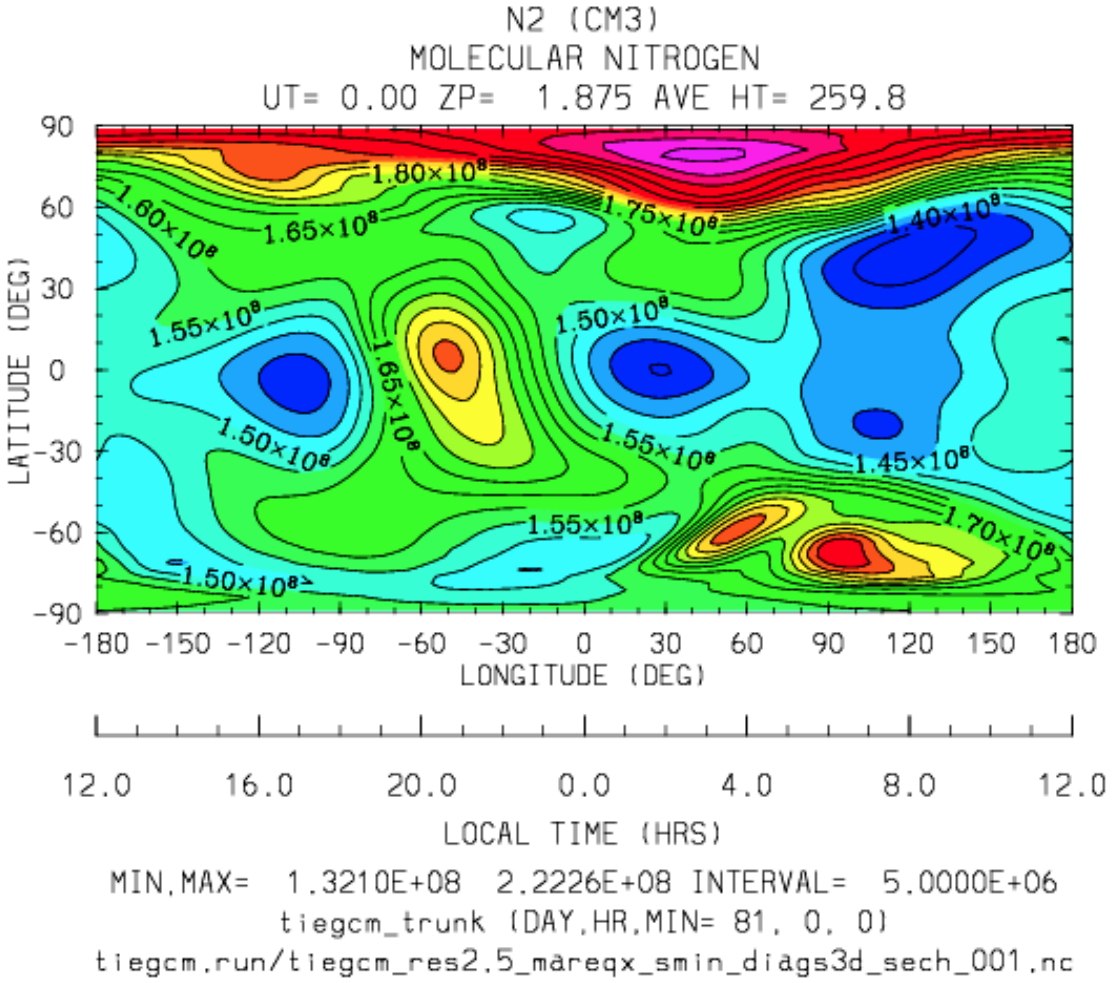
N2

Diagnostic field: Molecular Nitrogen (mmr):

```
diags(n)%short_name = 'N2'
diags(n)%long_name  = 'N2: Molecular Nitrogen'
diags(n)%units      = 'mmr'
diags(n)%levels     = 'lev'
diags(n)%caller     = 'comp.F'
```

Sample images: N2 cylindrical equidistant projection


N2 (CM3)
MOLECULAR NITROGEN
UT= 0.00 ZP= -4.125 AVE HT= 116.0
LATITUDE (DEG)
90
60
30
0
-30
-60
-90
-180 -150 -120 -90 -60 -30 0 30 60 90 120 150 180
LONGITUDE (DEG)
12.0 16.0 20.0 0.0 4.0 8.0 12.0
LOCAL TIME (HRS)
MIN,MAX= 4.4184E+11 6.7303E+11 INTERVAL= 1.0000E+10
tiegcm_trunk (DAY,HR,MIN= 81, 0, 0)
tiegcm.run/tiegcm_res2.5_mareqx_smin_diags3d_sech_001.nc



Back to diagnostics table

ZGMID

Diagnostic field: Geometric Height at Midpoints:

```
diags(n)%short_name = 'ZGMID'
diags(n)%long_name  = 'ZGMID: Geometric Height at midpoints'
diags(n)%units      = 'cm'
diags(n)%levels     = 'lev'
diags(n)%caller     = 'addiag.F'
```

Back to diagnostics table

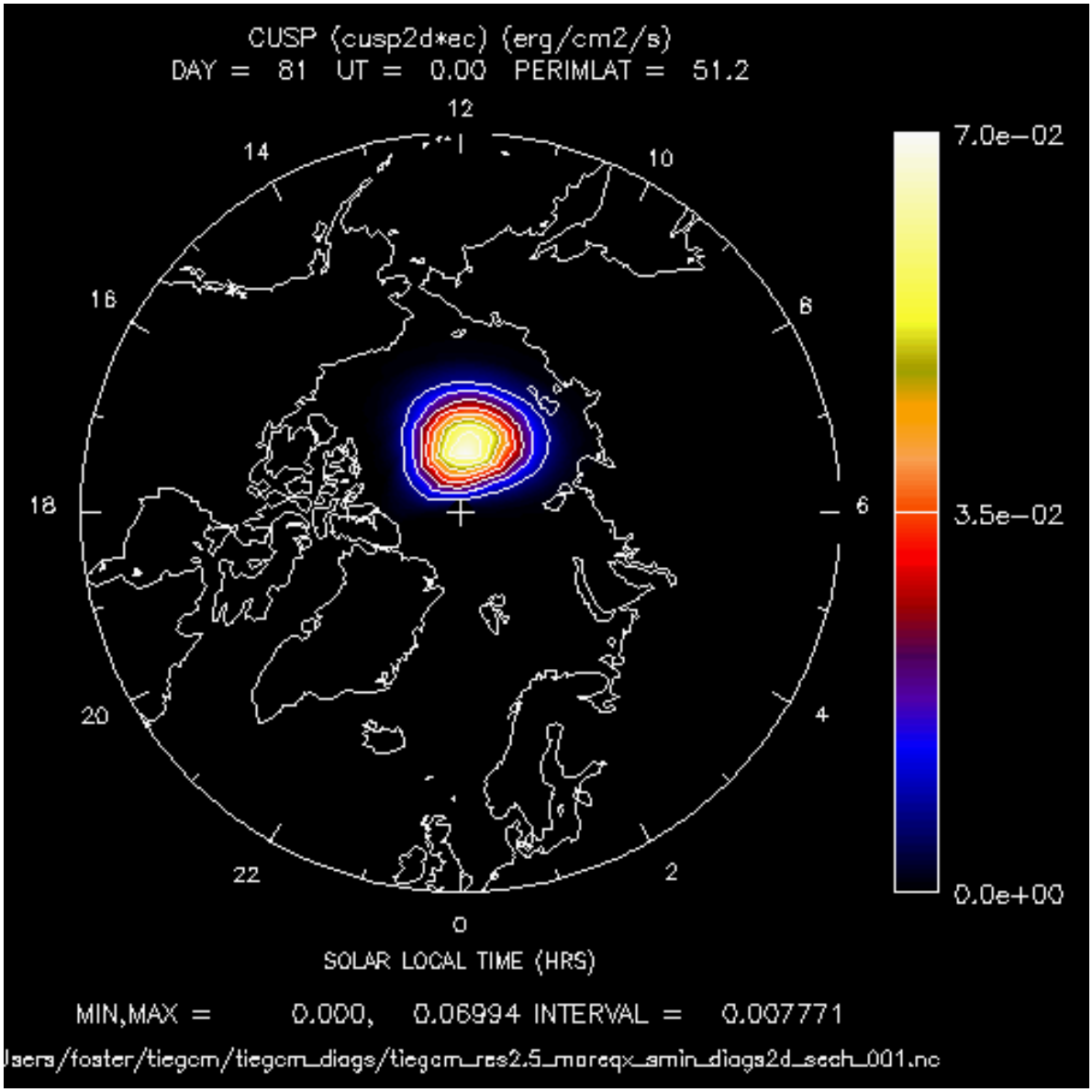
CUSP

Cusp low energy electron flux (erg/cm2/s):

```
diags(n)%short_name = 'CUSP'
diags(n)%long_name  = 'Cusp low energy electron flux'
diags(n)%units      = 'erg/cm2/s'
```

```
diags(n)%levels      = 'none'
diags(n)%caller      = 'dynamics.F'
```

Sample images: CUSP polar equidistant projection



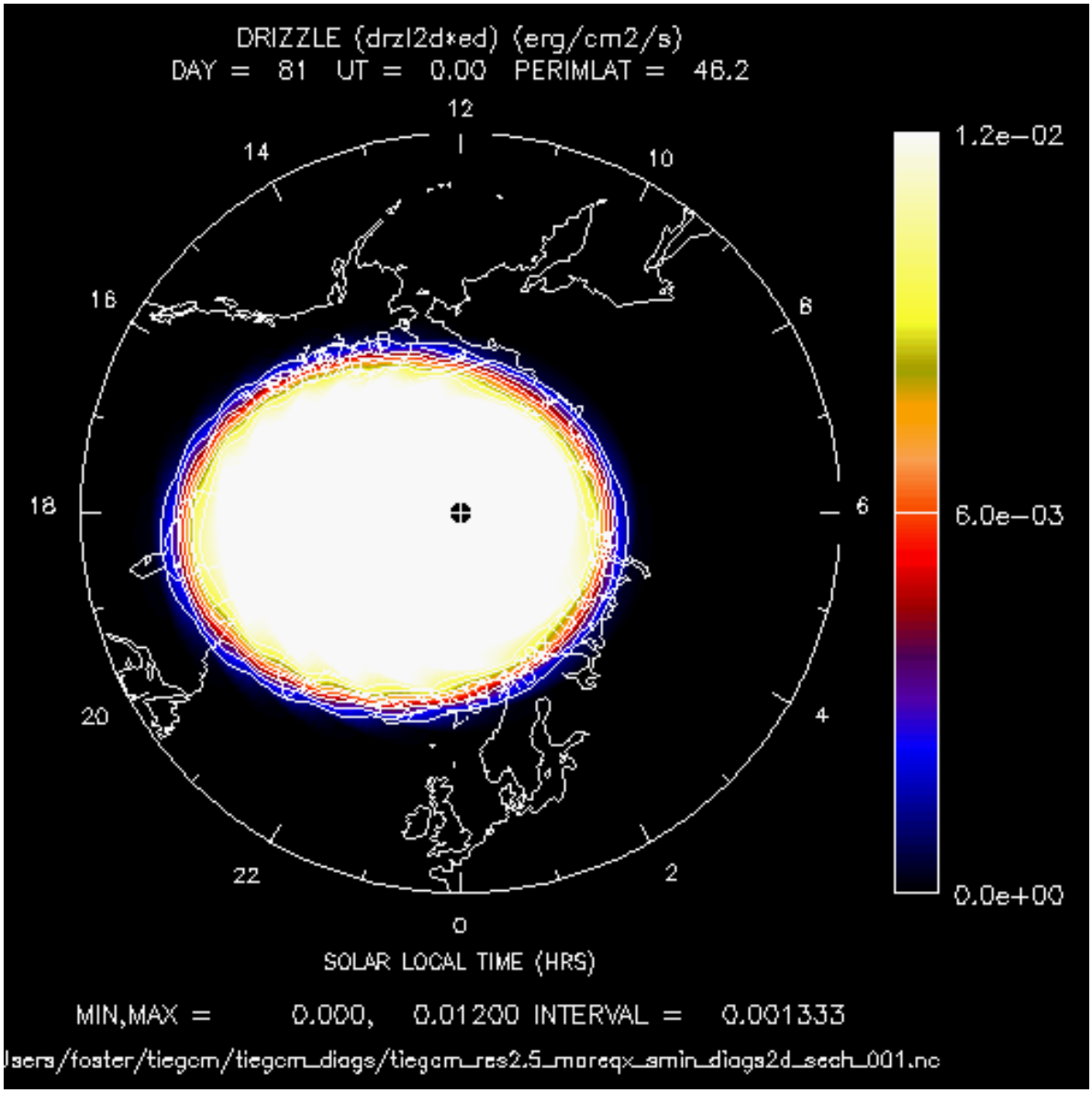
Back to diagnostics table

DRIZZLE

Drizzle low energy electron flux (erg/cm2/s):

```
diags(n)%short_name  = 'DRIZZLE'
diags(n)%long_name   = 'Drizzle low energy electron flux'
diags(n)%units       = 'erg/cm2/s'
diags(n)%levels      = 'none'
diags(n)%caller      = 'dynamics.F'
```

Sample images: DRIZZLE polar projection



Back to diagnostics table

ALFA

Aurora Characteristic Energy (keV):

```
diags(n)%short_name = 'ALFA'
diags(n)%long_name  = 'Aurora Characteristic Energy'
diags(n)%units      = 'keV'
diags(n)%levels     = 'none'
diags(n)%caller     = 'dynamics.F'
```

Sample images: ALFA polar projection

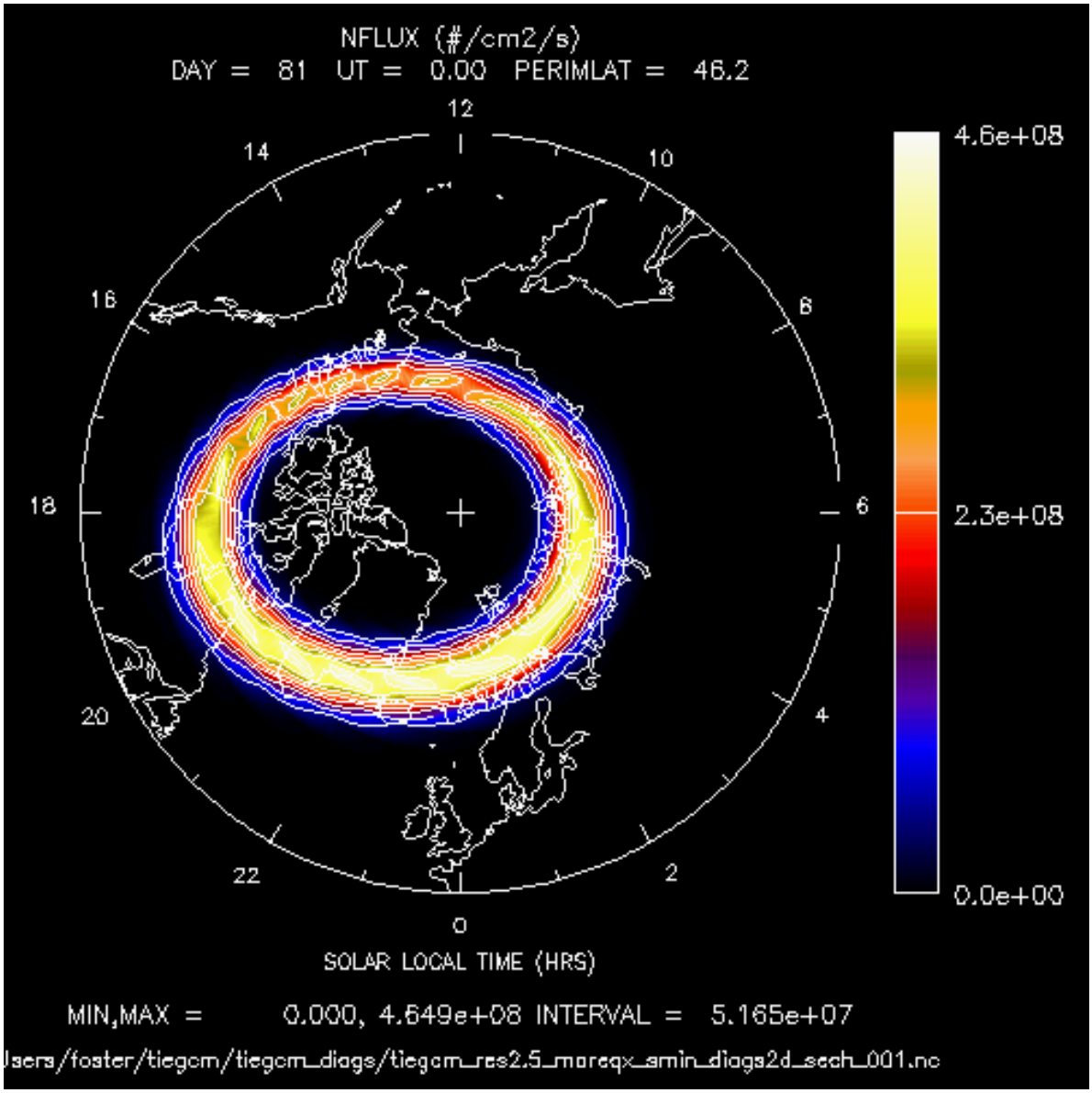
Back to diagnostics table

NFLUX

Aurora Number Flux (#/cm2/s):

```
diags(n)%short_name = 'NFLUX'
diags(n)%long_name  = 'Aurora Number Flux'
diags(n)%units      = '#/cm2/s'
diags(n)%levels     = 'none'
diags(n)%caller     = 'dynamics.F'
```

Sample images: NFLUX polar projection



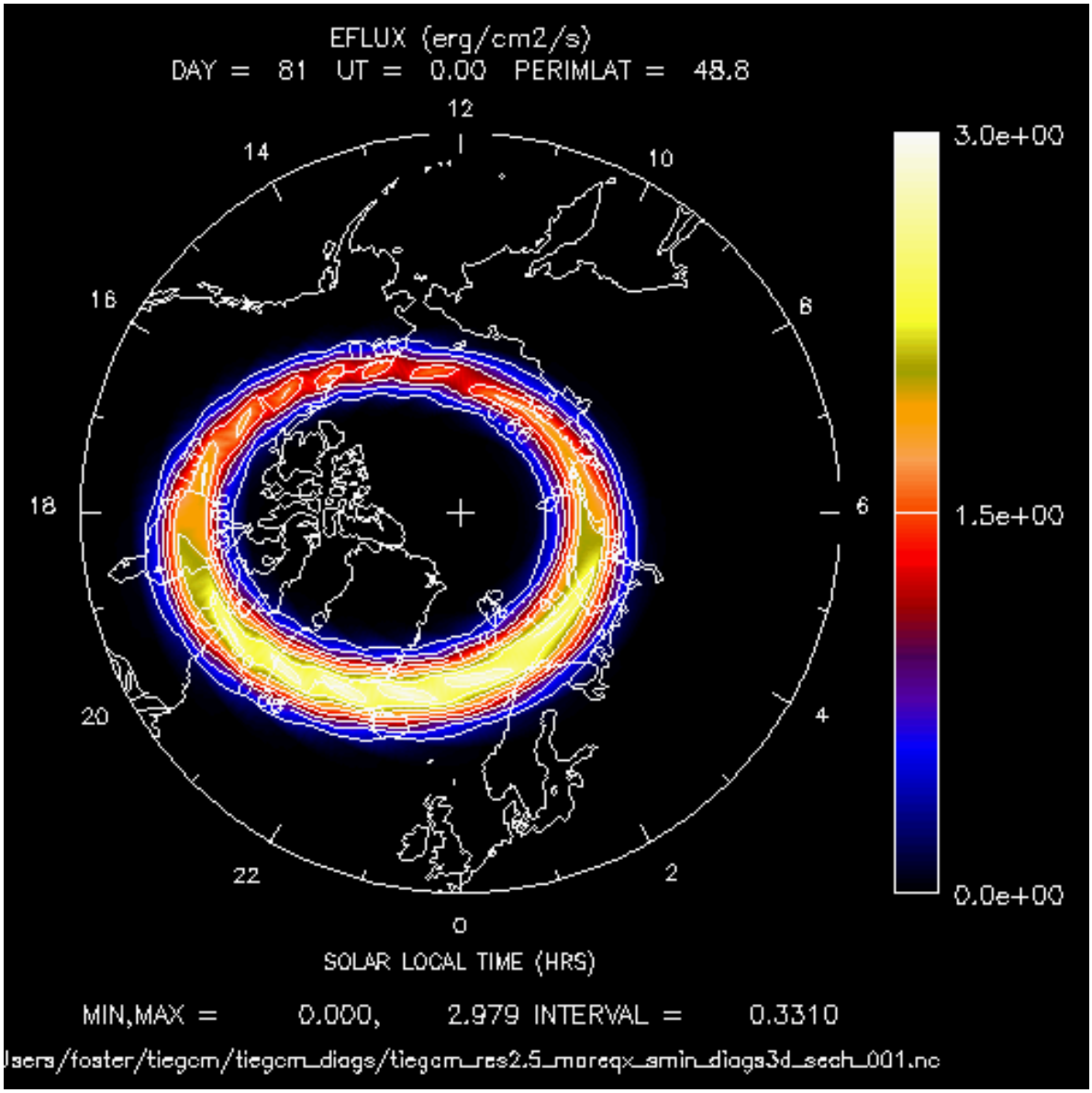
Back to diagnostics table

EFLUX

Aurora Energy Flux (#/cm2/s):

```
diags(n)%short_name = 'EFLUX'
diags(n)%long_name  = 'Aurora Energy Flux'
diags(n)%units      = 'erg/cm2/s'
diags(n)%levels     = 'none'
diags(n)%caller     = 'dynamics.F'
```

Sample images: EFLUX polar projection



Back to diagnostics table

CHAPTER

NINE

MODEL SOURCE CODE

The source code is in the *src/* subdirectory of the model root directory (*modeldir*), which is provided in the model *download* file.

9.1 The Academic License Agreement

The TIEGCM `Open Source Academic Research License Agreement` specifies the terms and restrictions under which the NCAR/UCAR grants permission to use the model, including the source code, for research, academic, and non-profit purposes.

9.2 Source Code Flow Diagram

A detailed flow diagram and calling tree of the source code structure is available in single and multi-page pdf files:

Warning: Some details of these flow charts are out of date with respect to TIEGCM version 2.0

- TIEGCM Code Structure (multi-page pdf)
- TIEGCM Code Structure (single-page pdf)

9.3 Modifying the Source Code

As a community user, student, research scientist or developer, you may need to modify the model source code. It is best to do this after building and at least making a default execution of the model (see the *QuickStart* Section). To change one or more source files, simply go to the *src/* subdirectory in the model root directory *modeldir*, and edit the files as necessary. Then return to the working directory *workdir* and re-execute the job script. It will recompile the modified files, and any other source files that depend on the modified files, and re-execute the model. Alternatively, you can enter the execution directory *execdir*, and recompile the code by typing "gmake" on the command line, then return to the working directory and re-execute the job script.

CHAPTER

TEN

THE MAKE/BUILD PROCESS: COMPILE AND LINK

The TIEGCM model is formally supported on two platform systems: 64-bit Linux Desktop, and a Linux cluster supercomputer (e.g., NCAR's yellowstone system. However, the model has been built and executed on several other platforms. The source code is f90 standard compliant, and is mostly fixed-format fortran.

10.1 Compilers

The tiegcm can be built with three compilers on 64-bit Linux Desktop systems, but only the intel compiler is used on the NCAR supercomputer yellowstone. The default is intel, since it out-performs the other compilers on both systems. As of January, 2015, these are the versions of each compiler we are using:

- Intel
  - On 64-bit desktop: intel ifort 12.0.0
  - On NCAR supercomputer yellowstone: intel ifort 12.1.5
- PGI
  - On 64-bit desktop: PGI pgf90 9.0-4
- GNU gfortran
  - On 64-bit desktop: GNU gfortran 4.4.7

Each compiler has a makefile in the scripts directory that specifies compiler-specific flags, library paths, and other parameters necessary for the build priocess. These files are provided in the model *scripts/* directory, and are included in the main Makefile at build time:

- Makefile for Intel compiler on NCAR supercomputer yellowstone:
  `Make.intel_ys` (NCAR yellowstone)
- Makefile for Intel compiler on 64-bit Linux desktop system:
  `Make.intel_hao64`
- Makefile for PGI compiler on 64-bit Linux desktop system:
  `Make.pgi_hao64`
- Makefile for GNU gfortran compiler on 64-bit Linux desktop system:
  `Make.gfort_hao64`

One of these files, or the user's own, is specified by the job script variable "make" in the *job script*. The specified file is included in the main `Makefile`. User's outside NCAR are encouraged to copy and rename one of these files, and customize it for your own operating system and compiler.

10.2 Required External Libraries

External library dependencies are netCDF, MPI, and ESMF. The MPI implementation is often bundled in with the compiler. Local paths to these libraries are specified in the compiler-specific `Make` files described above.

10.2.1 The Earth System Modeling Framework (ESMF)

The electro-dynamo code see `source file pdynamo.F`) in the TIEGCM is calculated on the *geomagnetic grid*. Since the dynamo receives inputs from the neutral atmosphere, which is on the *geographic grid*, there is a need for regridding capability between the two grid systems. The same horizontal geomagnetic coordinates are used regardless of the 5-deg or 2.5-deg resolution of the geographic grid.

The Earth System Modeling Framework (see also Modeling Infractructure for the Geoscience Community is used in the TIEGCM to perform the grid remapping in an parallel MPI environment, see `src/esmf.F`. To build the TIEGCM, the ESMF library must be included in the link step. If the ESMF library is not already on your system, you will need to download and build it, using the same compiler you are using to build the TIEGCM.

HAO is using the following ESMF libraries built on a 64-bit Linux desktop for each compiler/MPI implementation:

ESMF libraries at HAO for use on Linux desktop systems (these paths are provided in the scripts/Make.xxxx files described above). The esmf makefiles esmf.mk are included in the model's main makefile `scripts/Makefile`

- For use with the Intel compiler:
  /home/tgcm/esmf/intel/esmf_6_3_0rp1/lib/libO/Linux.intel.64.intelmpi.default
  See `Makefile esmf.mk for Intel build`
- For use with the PGI compiler:
  /home/tgcm/esmf/pgi-9.04/lib/libO/Linux.pgi.64.mpich.default
  See `Makefile esmf.mk for PGI build`
- For use with the GNU gfortran compiler:
  /home/tgcm/esmf/gfort/esmf_6_3_0rp1/lib/libO/Linux.gfortran.64.openmpi.default
  See `Makefile esmf.mk for GNU gfortran build`
- For the NCAR Linux cluster yellowstone: esmf-6.3.0r-ncdfio-mpi-O
  The ESMF library is loaded on yellowstone with the "module load" command,
  executed by the `job script tiegcm-ys.job`.

10.2.2 netCDF

The Network Common Data Form (NetCDF) is a cross-platform, self-describing metadata file format, developed by UNIDATA at UCAR. Please see NetCDF for more information. It is necessary to link the netCDF library when the model is built, since all data files imported to the model, and all model output history files are in NetCDF format. Because NetCDF is platform-independent, all history and data files can be used on either linux desktops or the NCAR yellowstone system. At HAO on linux desktops, we are using NetCDF version 4.1.1. On the NCAR yellowstone system, we are using version 4.3.2.

10.3 Build for Debugging

The model can be built with debug flags set in the compiler. To do this, simply set debug = TRUE in the *job script*, and resubmit (see also *job scripts section*).

The debug flags are set in the compiler-specific Make files described above. They can be adjusted there, of course, but usually they include floating-point exception traps, and core dumps with traceback. If debug was false in a previous run, the entire code will be rebuilt with the debug flags set, however, it doesn't hurt to go to the execdir and type "gmake clean" before resubmitting. Keep in mind that because optimization is turned off when debug flags are set, performance will be destroyed, and the model will run agonizingly slow.

Although we do not support the model with MPI turned off, it can also be useful for debugging to run the model with only a single MPI task. To do this, set nproc=1 in the linux job script, or set #BSUB -n 1 in the yellowstone job script.

CHAPTER

ELEVEN

BENCHMARK RUNS

Note: Benchmark results (pdf plot files) for version tiegcm2.0 are available here: Post-processing of TIEGCM benchmarks

A series of benchmark runs are made for each major release of the tiegcm These runs (for tiegcm2.0) were made using the python scripts in the *tgcmrun/* directory.

Netcdf files with the first history of each benchmark run are available in the *data download file*. These files can be used as start-up *SOURCE* files to reproduce the runs.

Benchmark runs are provided in three groups: full-year climatology, seasonal, and solar storm events. Here is a table of runs at 2.5-degree model resolution (there is also a set at the 5-degree resolution):

```
Full-year climatologies (solar min,max):
  tiegcm_res2.5_climatology_smax       Full-year climatology at solar maximum conditions
  tiegcm_res2.5_climatology_smin       Full-year climatology at solar minimum conditions

Seasons (Equinox, Solstice, solar min,max):
  tiegcm_res2.5_decsol_smax            December Solstice, solar maximum (days 355-360)
  tiegcm_res2.5_decsol_smin            December Solstice, solar minimum (days 355-360)
  tiegcm_res2.5_junsol_smax            June Solstice, solar maximum     (days 172-177)
  tiegcm_res2.5_junsol_smin            June Solstice, solar minimum     (days 172-177)
  tiegcm_res2.5_mareqx_smax            March Equinox, solar maximum     (days 80-85)
  tiegcm_res2.5_mareqx_smin            March Equinox, solar minimum     (days 80-85)
  tiegcm_res2.5_sepeqx_smax            September Equinox, solar maximum (days 264-269)
  tiegcm_res2.5_sepeqx_smin            September Equinox, solar minimum (days 264-269)

Solar storm events (Heelis/GPI and Weimer/IMF,GPI):
  tiegcm_res2.5_dec2006_heelis_gpi     December 2006 storm, Heelis/GPI          (days 330-360)
  tiegcm_res2.5_dec2006_weimer_imf     December 2006 storm, Weimer/IMF,GPI      (days 330-360)
  tiegcm_res2.5_jul2000_heelis_gpi     July 2000 storm,     Heelis/GPI          (days 192-202)
  tiegcm_res2.5_jul2000_weimer_imf     July 2000 storm,     Weimer/IMF,GPI      (days 192-202)
  tiegcm_res2.5_nov2003_heelis_gpi     November 2003 storm, Heelis/GPI          (days 323-328)
  tiegcm_res2.5_nov2003_weimer_imf     November 2003 storm, Weimer/IMF,GPI      (days 323-328)
  tiegcm_res2.5_whi2008_heelis_gpi     Whole Helio Interval 2008 Heelis/GPI     (days 81-106)
  tiegcm_res2.5_whi2008_weimer_imf     Whole Helio Interval 2008 Weimer/IMF,GPI (days 81-106)
```

Note:

Seasonal runs and full-year Climatologies were run with constant solar forcing, as follows:

- Solar Minimum:

  POWER = 18.
  CTPOTEN = 30.
  F107 = 70.

F107A = 70.

- Solar Maximum:

  POWER = 40.
  CTPOTEN = 60.
  F107 = 200.
  F107A = 200.

For comprehensive plots of all benchmark runs, please see the TIEGCM2.0 Release Documentation

Seasonal (equinoxes, solstices, solar min, max):

- decsol: December Solstice (days 355-360)
- junsol: June Solstice (days 172-177)
- mareqx: March Equinox (days 80-85)
- sepeqx: September Equinox (days 264-269)

Climatologies (full-year runs with daily histories):

- climatology_smax: Climatologies at Solar Maximum
- climatology_smin: Climatologies at Solar Minimum

December, 2006 "AGU" Storm Case (days 330-360):

- dec2006: Heelis potential model with GPI (Kp) data
- dec2006: Weimer potential model with IMF data (F10.7 from GPI)

November 19-24, 2003 Storm Case (days 323-328)

- nov2003: Heelis potential model with GPI (Kp) data
- nov2003: Weimer potential model with IMF data (F10.7 from GPI)

July 11-21, 2000 "Bastille Day" Storm Case (days 192-202)

- jul2000: Heelis potential model with GPI (Kp) data
- jul2000: Weimer potential model with IMF data (F10.7 from GPI)

Whole Heliosphere Interval (WHI) (March 21 to April 16, 2008)

- whi2008: Heelis potential model with GPI (Kp) data
- whi2008: Weimer potential model with IMF data (F10.7 from GPI)

Note: For more detailed information and access to history file output, and extensive post-processing of these runs, see the TIEGCM2.0 Release Documentation

---

11.1 Making Benchmark Runs

The *tgcmrun* directory under the model root directory *(modeldir)* contains Python code that semi-automates submission of selected benchmark model runs on the NCAR supercomputer system (yellowstone). The tgcmrun command can be executed interactively on the command line, or from a shell script. Type "tgcmrun -h" on the command line for a detailed usage message. Typing "tgcmrun" on the command line will cause the program to print the available benchmark runs and prompt the user as follows:

```
The following runs are available:

NUMBER NAME            DESCRIPTION
------ ----            -----------
0      default_run     Default run
1      decsol_smax     December Solstice Solar Maximum
2      decsol_smin     December Solstice Solar Minimum
3      junsol_smax     June Solstice Solar Maximum
4      junsol_smin     June Solstice Solar Minimum
5      mareqx_smax     March Equinox Solar Maximum
6      mareqx_smin     March Equinox Solar Minimum
7      sepeqx_smax     September Equinox Solar Maximum
8      sepeqx_smin     September Equinox Solar Minimum
9      nov2003_heelis_gpi      November 2003 storm case, Heelis potential model, GPI data
10     nov2003_weimer_imf      November 2003 storm case, Weimer potential model, IMF, GPI data
11     dec2006_heelis_gpi      December 2006 "AGU storm", Heelis potential model, GPI data
12     dec2006_weimer_imf      December 2006 "AGU storm", Weimer potential model, IMF and GPI data
13     whi2008_heelis_gpi      2008 "Whole Heliosphere Interval", Heelis potential model, GPI data
14     whi2008_weimer_imf      2008 "Whole Heliosphere Interval", Weimer potential model, IMF, GPI da
15     jul2000_heelis_gpi      July 2000 "Bastille Day" storm, Heelis potential model, GPI data
16     jul2000_weimer_imf      July 2000 "Bastille Day" storm, Weimer potential model, IMF, GPI data
17     climatology_smin        Climatology run with constant solar minimum conditions (Jan 1-5)
18     climatology_smax        Climatology run with constant solar maximum conditions (Jan 1-5)

Enter number of desired run (0-18) ('q' to quit, 'p' to print list, default=0):
```

At this point the user can enter an integer 0 to 18, specifying the desired run. The user will then be prompted for a few additional parameters (tiegcm or timegcm model, resolution, model root directory, etc). However, it is easiest to set a few environment variables before executing tgcmrun, to minimize the need to enter long file paths at the prompt:

11.1.1 Environment variables to set before using the tgcmrun utility:

- **TGCMTEMP**: Path to a large temporary directory where the model can be built, executed, and output stored.
- **TGCMDATA**: Path to a directory containing data files required by the model (netcdf data and start-up history files)
- **TIEGCM_ROOT**: Path to the tiegcm model root directory containing source code, scripts, tgcmrun, etc. (not necessary if making only TIMEGCM runs)
- **TIMEGCM_ROOT**: Path to the timegcm model root directory containing source code, scripts, tgcmrun, etc. (not necessary if making only TIEGCM runs)

Source history files (start-up netcdf files with a single history) to start these runs are provided on the *data download page <http://www.hao.ucar.edu/modeling/tgcm/download.php>* (there are separate data downloads available for each model resolution). These source files should be located in the *TGCMDATA* directory (or the path may be specified in the job script with the *tgcmdata* shell variable).

The tgcmrun program can also be executed from a shell script. There are several example tcsh scripts in the tgcmrun directory that make series of runs for various purposes. The scripts optionally run at one or both model resolutions. History files, stdout log files, and job scripts used, are stored in a directory tree below the working directory.

Standard 18 benchmark runs (as in the interactive tgcmrun command above):

- **run_climatology**: Start climatology runs (smin,smax). These can be extended to a full year by the user.
- **run_seasons**: Make seasonal benchmark runs (equinoxes, solstices, at smin, smax)
- **run_storms**: Make storm case benchmark runs (heelis_gpi and weimer_imf)

Additional runs for testing compilers, performance, etc.:

- **run_compilers**: Make three runs, each with a different compiler (linux desktop systems only)
- **run_perf**: Make several runs using different processor (MPI task) counts (super systems only)
- **run_scriptsonly**: This only makes the namelist input and job scripts (does not submit the jobs)

11.2 Model Output History Files of the tiegcm2.0 Benchmark runs

Model output history files are stored in CF-compliant netCDF format (see *NetCDF History Output Files*). Benchmark history files are available via Globus research data sharing service. The tiegcm benchmark history files are stored at the "NCAR Data Sharing Service" *Globus* shared endpoint (for users with an NCAR/CISL login: this endpoint is /glade/u/datashare/tgcm).

See these CISL docs for information regarding the NCAR Data Sharing Service:

- NCAR Data Sharing Service
- Globus file transfers (see especially "Transferring files with the webh interface")
- Retreiving data from a shared endpoint

Here is a summary procedure for accessing the tiegcm2.0 benchmark data:

Note: You do *NOT* have to have an NCAR user account or token to retrieve this data.

- You must have or create a Globus account. If your institution/organization has a Globus data sharing endpoint, you can use your institutional authorization to login to Globus. Otherwise, you can create a Globus personal account to transfer files to your personal laptop or desktop computer.
- Log in to your Globus account, and click on "File Transfer"
- To reach the NCAR/TIEGCM source endpoint, click in the "Endpoint" text box on the left, and type "TIEGCM v2.0". It should retrieve directory contents, and show a "benchmarks" folder.
- Next, establish your destination endpoint on the right. This is either your institutional endpoint, or the username of your personal Globus login.
- Select the locations/files you want to download from the left side, and the destination location on the right, then click the right arrow at the top to begin the transfer.

Here's a screen shot of a Globus file transfer from the TIEGCM v2.0 endpoint to my personal Macbook Pro: `Globus_screenshot.png`

In each of the 6 benchmark groups are folders for each run, with folders containing the history files (hist), post-processing (proc), and scripts and log files (stdout). Individual files or whole directories can be downloaded.

Note: Users wanting to use their NCAR authentication rather than personal GlobusID, apparently need to have a login on the NCAR RDA (Research Data Archives) to access the NCAR GLADE endpoint on Globus.

CHAPTER

TWELVE

POST-PROCESSING AND VISUALIZATION

TIEGCM netCDF history files can be read by any application with access to the netCDF library, including many freely available software packages developed for mainpulating or displaying netCDF data (see http://www.unidata.ucar.edu/software/netcdf/software.html). At HAO, we often use the netCDF Operators NCO for file manipulation (subset extracting, concatenation, hyperslabbing, metadata editing, etc). However for visualization, we typically use one of three post-processors developed at HAO:

12.1 tgcmproc_f90

- *tgcmproc_f90* is a batch-style processor written in fortran 90. This program reads a user namelist input file via stdin, and outputs multi-frame plot files (cgm and/or ps), and output data files (e.g., ascii, netCDF).
- Uses the freely available NCAR Graphics libraries for basic contouring, making maps at various projections, vector plots, etc.
- Plots 2d horizontal and vertical slices, and time-dependent plots on the model grid.
- Calculates a large number of diagnostics from fields on the histories.
- Custom contouring (setting cmin,cmax,cint)
- Can interpolate to constant height surfaces.
- Can be downloaded from the TGCM website, but the f90 code must be compiled, and NCAR Graphics libraries must be linked.

12.2 tgcmproc_idl

- *tgcmproc_idl* is an IDL application for browsing and plotting TIEGCM output, with an easy to use Graphical User Interface (GUI).
- 2d contouring of horizontal and vertical slices, including maps at various projections.
- Can save images and plots to a variety of image formats.
- Custom contouring (setting cmin,cmax,cint).
- Can interpolate to constant height surfaces.
- Can plot fields on the magnetic grid (tgcmproc_f90 does not do this).
- Can make and save png movie animations.
- Should run fine for anybody w/ IDL, but IDL is licensed, and can be expensive.

12.3 utproc

- *utproc* is an IDL/GUI application that makes time-series contours and images including ut vs zp pressure at selected grid lat x lon locations, and ut vs latitude at selected zp pressure surfaces.

These applications are available at the TGCM download page. Tgcmproc_f90 is best for generating large numbers of plots in a "batch-style" environment, whereas tgcmproc_idl is best for browsing history files in a GUI interface, and saving plots or images as desired. The utproc processor is a hybrid in the sense that a series of plots can be setup using the GUI, and then created when requested.

At HAO, we also use the NCAR Command Language (NCL) for plotting, analysis, and converting to/from various file formats (GRIB, HDF, etc). NCL scripts can be used to generate customized plots and images, as well as providing a variety of analysis and file-manipulation tools.

CHAPTER

THIRTEEN

CONTACT INFORMATION

The TIEGCM and related Thermosphere-Ionosphere models have been developed by the "Atmosphere Ionosphere Magnetosphere" (AIM) Section of the High Altitude Observatory (HAO) at NCAR (see http://www.hao.ucar.edu/modeling/tgcm).

For more information, questions or problems about the models, please subscribe to the moderated email list tgcm-group@ucar.edu.

For questions and information regarding the physics and chemistry implementation of the model, numerical algorithms, and analysis of model results, you may also contact the following HAO scientists:

- Stan Solomon (stans@ucar.edu)
- Art Richmond (richmond@ucar.edu)
- Hanli Liu (liuh@ucar.edu)
- Gang Lu (ganglu@ucar.edu)

CHAPTER

FOURTEEN

GLOSSARY

benchmark runs Selected validation runs made with each release of the model. These runs can be made using Python code in the *tgcmrun/* directory.

benchmarks/ Directory in the model root directory containing shell scripts that call *tgcmrun/* for making benchmark runs, some utility scripts, and a subdirectory postproc/ containing scripts that do post-processing on benchmark results. See *Benchmark Runs* for more information. Benchmark results (plots) for version tiegcm2.0 are available here: Benchmarks Results

continuation run A continuation run continues from the last output history of the previous run. That history (*START* time) must be on the first *OUTPUT* file provided by the namelist input file. A continuation run must not specify a *SOURCE* file or *SOURCE_START* time. See also *Continuation Run*

diagnostic fields A list of diagnostic fields are available to be saved on secondary history files. See section *Saving Diagnostic Fields*.

datadir Directory containing startup history and data files necessary for running the model. This is specified with the *tgcmdata* shell variable in the *job script*.

doc/ Subdirectory under the *modeldir* containing documentation, e.g., the User's Guide, Model Description, Release Notes, etc.

ESMF "Earth System Modeling Framework". The ESMF library is used in the electro-dynamo code (pdynamo.F in version 2.0 or later) for regridding between geographic and geomagnetic grids in an mpi environment. This is open software that can be downloaded at https://www.earthsystemcog.org/projects/esmf/download/ If you build the ESMF library, it should be built with the same compiler with which the model is built.

execdir The model execution directory. This is the directory where the model is built and executed. It should be on a large temporary disk, capable of storing model object and module code, netCDF output history files, and other data. When a job script is executed from a working directory, the execdir is created if it does not already exist. During a model run, output history files are written to the execdir. The execdir is set in the *job script*. See also *Execution Directory*

geomagnetic coordinates The electro-dynamo fields (electric potential, electric field, and ion drift velocities) are calculated on a geomagnetic grid, see *magnetic coordinates*

Globus Globus Data Sharing Service for scientific research. The tiegcm2.0 benchmark history files and post-processing are available via Globus. See *Model Output History Files of the tiegcm2.0 Benchmark runs* for more information.

history A model history records the state of the model at a discrete instant in *model time*. One or more histories are stored in netCDF history files.

initial run An initial run is started from a history on a *SOURCE* file (see also *SOURCE_START*). Subsequent *continuation runs* do not provide SOURCE or SOURCE_START, but rather search for the *START* time on the first *OUTPUT* history file provided in the namelist input, and continue the run from there.

job script A csh script in the scripts/ directory which, when executed, will build and execute the model. The user defines a few shell variables in the job script, such as the *modeldir*, and the *namelist input*. See example `job script for Linux desktops`, and `job script for Super computer` See *Using the job scripts to set up and submit a model run* for more detailed information.

model time TIEGCM model time is represented by an integer triplet: day,hour,minute, where day is the julian day of the year, and hour is the ut. The variable for model time on history files is mtime(3,ntimes). For example, a history file may contain 24 hourly histories for day 80: mtime = 80,1,0, 80,2,0, ... 81,0,0.

modeldir The model root directory. This directory typically contains subdirectories *src/* (model source code), *scripts/* (utility scripts), *doc/* (documentation), and *benchmarks/*. The modeldir is available via *download*, and is typically a subdirectory of the model working directory (*workdir*). See also *Model Directory*

namelist input The model reads user specified parameters from the *namelist input file* via f90 standard namelist read. Keyword/Value pairs are read from unit 5, and are validated by the input module (input.F). See also *job scripts*.

netCDF TIEGCM output history files are written in netCDF, a self-describing platform-independent data format written and maintained by the UCAR Unidata program.

output File to receive stdout output from the model. This file will be created if it does not exist, or overwritten if it does exist.

resolution The TIEGCM can be run in one of two resolutions:

- 5 x 5 deg lat x lon, 2 grid levels per scale height (dz = 0.50)
- 2.5 x 2.5 deg lat x lon, 4 grid levels per scale height (dz = 0.25)

The resolution is set by the "modelres" shell variable in the TIEGCM *job script*. See also the section on *Grid Structure and Resolution*.

Note: The 2.5-degree resolution model is available in version 2.0, but it is not fully validated or supported by the public release.

---

scripts/ Subdirectory under the *modeldir* containing supporting and utility scripts, including job scripts, the default namelist input file, several Make files, etc.

src/ Subdirectory under the *modeldir* containing the model source code (*.F, *.h files).

tgcmrun/ Subdirectory under the *modeldir*. The tgcmrun directory contains Python code to make *benchmark runs* for the current release. The 'tgcmrun' command may be used to interactively submit selected benchmark runs, or tgcmrun can be executed from a shell script using command-line options. There are several run_xxxxx shell scripts there demonstrating how to make benchmark runs.

tgcmdata A directory path to start-up and other input data files required for running the model. This should be on a large temporary disk. tgcmdata is a csh variable optionally specified in the *job script*. If not specified, the job script will use the *TGCMDATA* environment variable. See also *job script shell variables*.

env var TGCMDATA A linux environment variable that refers to the *tgcmdata*. This environment variable may be used when referring to data files in the namelist read file, e.g., "GPI_NCFILE = *$TGCMDATA/gpi_xxxxx.nc*". See *namelist read files*.

tgcmproc_f90 Post-processor and visualizer for TIEGCM netCDF history files. Written in f90, and available at the TIEGCM download site See *tgcmproc_f90*.

tgcmproc_idl Post-processor and visualizer for TIEGCM netCDF history files. This processor is Written in IDL with a GUI, and is available at the TIEGCM download site See *tgcmproc_idl*.

utproc Post-processor and visualizer for TIEGCM netCDF history files. This processor reads time-series history files and makes ut vs pressure and ut vs latitude contours. It is written in IDL with a GUI, and is available at the TGCM download site. See *utproc*.

workdir User-created local working directory. This will typically contain the model root directory *modeldir* and related namelist input files, job scripts, stdout files, etc. Because the model source files are critical, this should be on backed-up disk, typically under your home directory.

Zp Vertical log pressure coordinate ln(p0/p) of the TIEGCM. This is the "lev" coordinate on the history files. See the chapter on *Altitude Coordinates the NCAR TIEGCM* for a detailed explanation of the relationship between Zp and Altitude.

CHAPTER

FIFTEEN

INDICES AND TABLES

- *genindex*
- *search*

Note: This document was last updated on March 21, 2016

A

ALFA, 106
amie_ibkg, 21
amienh, 20
amiesh, 20
aurora, 21

B

benchmark runs, **123**
benchmarks/, **123**
bgrddata_ncfile, 21
BMAG, 94
BX, 91
bximf, 21
BY, 92
byimf, 22
BZ, 93
bzimf, 22

C

calc_helium, 25
calendar_advance, 22
CO2_COOL, 47
colfac, 23
continuation run, **123**
ctmt_ncfile, 23
ctpoten, 23
current_kq, 24
current_pg, 24
CUSP, 104

D

datadir, **123**
default
    namelist read file, 19
DEN, 50
diagnostic fields, **123**
    ALFA, 106
    BMAG, 94
    BX, 91
    BY, 92
    BZ, 93
    CO2_COOL, 47
    CUSP, 104
    DEN, 50
    DRIZZLE, 105
    ED1, 98
    ED2, 99
    EFLUX, 107
    EX, 95
    EY, 96
    EZ, 97
    FOF2, 56
    HEATING, 52
    HMF2, 54
    JE13D, 84
    JE23D, 86
    JQR, 88
    KQLAM, 89
    KQPHI, 90
    LAMDA_HAL, 64
    LAMDA_PED, 66
    MU_M, 72
    N2, 102
    NFLUX, 106
    NMF2, 55
    NO_COOL, 48
    O/N2, 77
    PHIM2D, 100
    QJOULE, 80
    QJOULE_INTEG, 82
    SCHT, 58
    SIGMA_HAL, 60
    SIGMA_PED, 62
    TEC, 56
    UI_ExB, 68
    VI_ExB, 69
    WI_ExB, 71
    WN, 74
    ZGMID, 104
doc/, **123**
download, 5
DRIZZLE, 105
dynamo, 25

E

ED1, 98
ED2, 99
eddy_dif, 24
EFLUX, 107
enforce_opfloor, 24
env var TGCMDATA, **124**
ESMF, **123**
EX, 95
execdir, **123**
EY, 96
EZ, 97

F

f107, 25
f107a, 25
FOF2, 56

G

geomagnetic coordinates, **123**
Globus, **123**
gpi, 26
gswm, 26

H

HEATING, 52
hist, 27
history, **123**
HMF2, 54
hpower, 27

I

imf, 28
initial run, **123**
input, 18

J

JE13D, 84
JE23D, 86
job script, **124**
joulefac, 29
JQR, 88

K

kp, 28
KQLAM, 89
KQPHI, 90

L

label, 29
LAMDA_HAL, 64
LAMDA_PED, 66

M

model time, **124**
modeldir, **124**
MU_M, 72
mxhist_prim, 30
mxhist_sech, 30

N

N2, 102
namelist input, 18, **124**
namelist input
- amie_ibkg, 21
- amienh, 20
- aurora, 21
- bgrddata_ncfile, 21
- bximf, 21
- byimf, 22
- bzimf, 22
- calc_helium, 25
- calendar_advance, 22
- colfac, 23
- ctmt_ncfile, 23
- ctpoten, 23
- current_kq, 24
- current_pg, 24
- dynamo, 25
- eddy_dif, 24
- enforce_opfloor, 24
- f107, 25
- f107a, 25
- gpi_ncfile, 26
- gswm, 26
- hist, 27
- hpower, 27
- imf_ncfile, 28
- joulefac, 29
- kp, 28
- label, 29
- mxhist_prim, 30
- mxhist_sech, 30
- opdiffcap, 30
- output, 31
- potential_model, 31
- saber_ncfile, 28
- secflds, 32
- sechist, 33
- secout, 33
- secstart, 32
- secstop, 33
- source, 34
- source_start, 34
- start, 34
- start_day, 35
- start_year, 35

step, 35
stop, 35
swden, 36
swvel, 36
tide, 36
tide2, 37
tidi_ncfile, 28
netCDF, **124**
NFLUX, 106
NMF2, 55
NO_COOL, 48

O

O/N2, 77
opdiffcap, 30
output, 31, **124**

P

perf.table, 10
PHIM2D, 100
potential_model, 31

Q

QJOULE, 80
QJOULE_INTEG, 82

R

resolution, **124**

S

saber_ncfile, 28
SCHT, 58
scripts/, **124**
secflds, 32
sechist, 33
secout, 33
secstart, 32
secstop, 33
SIGMA_HAL, 60
SIGMA_PED, 62
source, 34
source_start, 34
src/, **124**
start, 34
start_day, 35
start_year, 35
step, 35
stop, 35
swden, 36
swvel, 36

T

TEC, 56
tgcmdata, **124**
tgcmproc_f90, **124**
tgcmproc_idl, **124**
tgcmrun/, **124**
tide, 36
tide2, 37
tidi_ncfile, 28

U

UI_ExB, 68
utproc, **124**

V

VI_ExB, 69

W

WI_ExB, 71
WN, 74
workdir, **125**

Z

ZGMID, 104
Zp, **125**