GITM User Manual

Version 2.1

A.J. Ridley, A.G. Burrell

April 1, 2013

Contents

1 Introduction to GITM **5**
- 1.1 Source Terms . . . 5
- 1.2 Ghost Cells . . . 5
- 1.3 Code Outline . . . 6

2 Getting Started **15**
- 2.1 Extracting the code from a tar file . . . 15
- 2.2 Checking out the code with CVS . . . 15
- 2.3 Configuring and Making GITM . . . 16
- 2.4 Running the Code . . . 16
- 2.5 Post Processing . . . 17
- 2.6 The Code Won't Compile!! . . . 18

3 Inputs **21**
- 3.1 Principle Input . . . 21
  - 3.1.1 Required Inputs . . . 21
  - 3.1.2 Temporal Boundaries . . . 22
  - 3.1.3 Defining Outputs . . . 23
  - 3.1.4 Restart Input . . . 24
  - 3.1.5 Numerical Options . . . 25
  - 3.1.6 Data Assimilation . . . 25
  - 3.1.7 Solar and Geomagnetic Indices . . . 26
  - 3.1.8 Physical Processes . . . 27
  - 3.1.9 Geographic Boundaries . . . 33
- 3.2 Setting the Grid . . . 34
  - 3.2.1 Running 3D Over the Whole Globe . . . 34
  - 3.2.2 Running 3D Over the Part of the Globe . . . 35
  - 3.2.3 Running in 1D . . . 35
- 3.3 Auxiliary Input Files . . . 36
  - 3.3.1 IMF and Solar Wind . . . 36
  - 3.3.2 Hemispheric Power . . . 37
  - 3.3.3 Solar Irradiance . . . 39
  - 3.3.4 Satellites . . . 39

4 Outputs **41**
- 4.1 IDL . . . 43
- 4.2 Python . . . 45

Bibliography **50**

Chapter 1

Introduction to GITM

The Global Ionosphere Thermosphere Model (GITM) is a 3D model of the upper atmosphere. It runs for Earth, Mars and Titan. A version is being worked on for Jupiter. GITM solves for the coupled continuity, momentum and energy equations of the neutrals and ions. For the ions, the time rate of change of the velocity is ignored, so the steady-state ion flow velocity is solved for. The ion temperature is a mixture of the electron and neutral temperature.

The neutrals are solved for using the Navier Stokes Equations. The continuity equation is solved for for each major species. One of the problems with GITM that needs to be rectified is that there are no real tracer species, so a species is either solved for completely or is not at all. These species can still be included in the chemistry calculation. There is only one horizontal velocity that is computed, while there are vertical velocities for each of the major species. A bulk vertical velocity is calculated as a mass weighted average. The temperature is a bulk temperature.

1.1 Source Terms

Chemistry is the only real source term for the continuity equation. Typically, diffusion is added in the continuity equation to allow for eddy diffusion, but this is not the case in GITM. What happens is that the vertical velocities are solved for, then a friction term is applied to that the velocities stay very close together in the eddy diffusion part of the code. This way, the velocities can't differ too much from each other. Diffusion is not needed, then.

For the horizontal momentum equation, there are the following sources: (1) ion drag; (2) viscosity; and (3) gravity wave acceleration. For the vertical velocity, the source terms are ion drag and friction between the different neutral species.

For the neutral temperature, the following source terms are included: (1) radiative cooling; (2) EUV heating; (3) auroral heating; (4) Joule heating; (5) conduction; and (6) chemical heating. The biggest pain for the temperature equation is the use of a normalized temperature. This means that the `temperature` variable in GITM does not contain the actual temperature, it contains the temperature multiplies by Boltzmann's Constant divided by the mean mass. This turns out to be a factor that is very similar to the specific heat, or roughly or order 1000. In order to get the actual temperature, the variable has to be multiplied by `temp_unit`.

1.2 Ghost Cells

GITM is a parallel code. It uses a 2D domain decomposition, with the altitude domain being the only thing that is not broken up. Blocks of latitude and longitude are used. These blocks are then distributed among different processors. In order to communicate between the processors, ghostcells are used. These are cells that essentially overlap with the neighboring block. MPI (message passing interface) is then used to move information from one block to another, filling in the ghostcells. The code then loops from 1-N, where the flux is calculated at the boundaries from the 0-1 boundary to the N-N+1 boundary. A second order scheme is used to calculate the fluxes, along with a flux limiter. Therefore, two ghost cells are needed.

In the vertical direction, ghost cells are also used to set boundary conditions. The values in these cells are used to calculate the fluxes, just as described above. Different types of boundary conditions (constant values, constant fluxes, constant gradients, floating, zero fluxes, etc) can be set by carefully choosing the right values in the ghost cells.

1.3 Code Outline

This is an outline of GITM. To produce this file, go into the `src` directory, and type:

```
cd ../src
./calling_sequence.pl > outline.tex
mv outline.tex ../srcDoc
cd ../srcDoc
```

`main program` in **main.f90**

- `init_mpi` in file **init_mpi.f90**
  - `MPI_INIT`
  - `MPI_COMM_RANK`
  - `MPI_COMM_SIZE`
- `start_timing` in file **timing.f90**
- `delete_stop` in file **stop_file.f90**
  - `report` in file library.f90
- `init_planet` in file **ModEarth.f90**
  - `time_int_to_real` in file time_routines.f90
- `set_defaults` in file **ModInputs.f90**
  - `set_strings` in file ModInputs.f90
  - `time_int_to_real` in file time_routines.f90
  - `set_planet_defaults` in file Earth.f90
- `read_inputs` in file **read_inputs.f90**
  - `report` in file library.f90
  - `stop_gitm` in file library.f90
  - `stop_gitm` in file library.f90
  - `stop_gitm` in file library.f90
  - `MPI_Bcast`
  - `stop_gitm` in file library.f90
  - `MPI_Bcast`
  - `stop_gitm` in file library.f90
- `set_inputs` in file **set_inputs.f90**
  - `report` in file library.f90
  - `read_in_int` in file set_inputs.f90

- time_int_to_real in file time_routines.f90
- time_real_to_int in file time_routines.f90
- fix_vernal_time in file set_inputs.f90
- read_in_int in file set_inputs.f90
- time_int_to_real in file time_routines.f90
- read_in_time in file set_inputs.f90
- read_in_int in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- time_real_to_int in file time_routines.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- IO_set_f107_single
- IO_set_f107a_single
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_real in file set_inputs.f90
- IO_set_hpi_single
- read_in_real in file set_inputs.f90
- IO_set_kp_single
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- IO_set_imf_by_single

- IO_set_imf_bz_single
- IO_set_sw_v_single
- read_in_string in file set_inputs.f90
- IO_set_inputs
- read_MHDIMF_Indices
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_string in file set_inputs.f90
- read_in_string in file set_inputs.f90
- read_in_string in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_int in file set_inputs.f90
- read_in_int in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90

- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_int in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_string in file set_inputs.f90
- read_in_string in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_logical in file set_inputs.f90
- read_in_int in file set_inputs.f90
- read_in_int in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_real in file set_inputs.f90
- read_in_logical in file set_inputs.f90

  - read_in_logical in file set_inputs.f90
  - read_in_time in file set_inputs.f90
  - read_in_time in file set_inputs.f90
  - read_in_real in file set_inputs.f90
  - read_in_logical in file set_inputs.f90
  - read_in_real in file set_inputs.f90
  - read_in_int in file set_inputs.f90
  - read_in_string in file set_inputs.f90
  - read_in_real in file set_inputs.f90
  - read_in_int in file set_inputs.f90
  - read_in_string in file set_inputs.f90
  - read_in_real in file set_inputs.f90
  - read_satellites in file satellites.f90
  - read_in_real in file set_inputs.f90
  - read_in_real in file set_inputs.f90
  - read_in_real in file set_inputs.f90
  - read_in_logical in file set_inputs.f90
  - read_in_string in file set_inputs.f90
  - read_in_string in file set_inputs.f90
  - read_in_real in file set_inputs.f90
  - read_in_logical in file set_inputs.f90
  - read_in_string in file set_inputs.f90
  - Set_Euv in file calc_euv.f90
  - read_in_logical in file set_inputs.f90
  - read_in_real in file set_inputs.f90
  - read_in_string in file set_inputs.f90
  - IO_set_inputs
  - read_NGDC_Indices
  - read_in_string in file set_inputs.f90
  - read_in_string in file set_inputs.f90
  - IO_set_inputs
  - read_SWPC_Indices
  - read_in_string in file set_inputs.f90
  - IO_set_inputs
  - read_NOAAHPI_Indices
  - stop_gitm in file library.f90
  - time_int_to_real in file time_routines.f90

- initialize_gitm in file **initialize.f90**
  - init_radcooling in file ModEarth.f90

- start_timing in file timing.f90
- report in file library.f90
- time_real_to_int in file time_routines.f90
- fix_vernal_time in file set_inputs.f90
- get_f107
- get_f107a
- init_grid in file init_grid.f90
- read_inputs in file read_inputs.f90
- set_inputs in file set_inputs.f90
- read_restart in file restart.f90
- init_msis in file init_msis.Earth.f90
- set_RrTempInd in file ModRates.Earth.f90
- init_euv in file calc_euv.f90
- init_altitude in file init_altitude.f90
- UAM_gradient
- init_heating_efficiency in file Earth.f90
- init_magheat in file ModEarth.f90
- init_isochem in file Mars.f90
- init_aerosol in file ModEarth.f90
- init_isochem in file Mars.f90
- init_msis in file init_msis.Earth.f90
- get_temperature in file init_altitude.f90
- init_iri in file init_iri.Earth.f90
- read_waccm_tides in file tides.f90
- update_waccm_tides in file tides.f90
- read_tides in file tides.f90
- update_tides in file tides.f90
- init_b0 in file init_b0.f90
- init_energy_deposition in file init_energy_deposition.f90
- report in file library.f90
- SUBSOLR
- exchange_messages_sphere in file exchange_messages_sphere.f90
- calc_pressure in file calc_pressure.f90
- UA_calc_electrodynamics in file calc_electrodynamics.f90
- calc_eddy_diffusion_coefficient in file Earth.f90
- calc_rates in file calc_rates.Earth.f90
- calc_viscosity in file calc_rates.Earth.f90
- calc_rates in file calc_rates.Earth.f90
- end_timing in file timing.f90

  - calc_vtec in file calc_tec.f90
  - calc_single_vtec in file calc_tec.f90
- write_output in file **write_output.f90**
  - output in file output_common.f90
  - move_satellites in file satellites.f90
  - write_restart in file restart.f90
  - logfile in file logfile.f90
- report in file **library.f90**
- Loop Start
  - calc_pressure in file **calc_pressure.f90**
    * report in file library.f90
  - calc_timestep_vertical in file **calc_timestep.f90**
    * report in file library.f90
    * MPI_AllREDUCE
    * stop_gitm in file library.f90
  - calc_timestep_horizontal in file **calc_timestep.f90**
    * report in file library.f90
    * MPI_AllREDUCE
  - advance in file **advance.f90**
    * report in file library.f90
    * start_timing in file timing.f90
    * update_tides in file tides.f90
    * update_waccm_tides in file tides.f90
    * advance_vertical_all in file advance.f90
    * add_sources in file add_sources.f90
    * advance_horizontal_all in file advance.f90
    * time_real_to_int in file time_routines.f90
    * get_f107
    * stop_gitm in file library.f90
    * get_f107a
    * stop_gitm in file library.f90
    * init_msis in file init_msis.Earth.f90
    * init_iri in file init_iri.Earth.f90
    * init_b0 in file init_b0.f90
    * end_timing in file timing.f90
    * report in file library.f90
    * start_timing in file timing.f90
    * calc_rates in file calc_rates.Earth.f90
    * calc_viscosity in file calc_rates.Earth.f90
    * advance_vertical in file advance_vertical.f90
    * end_timing in file timing.f90

        * report in file library.f90
        * start_timing in file timing.f90
        * exchange_messages_sphere in file exchange_messages_sphere.f90
        * calc_rates in file calc_rates.Earth.f90
        * calc_physics in file calc_physics.f90
        * advance_horizontal in file advance_horizontal.f90
        * calc_physics in file calc_physics.f90
        * calc_rates in file calc_rates.Earth.f90
        * advance_horizontal in file advance_horizontal.f90
        * exchange_messages_sphere in file exchange_messages_sphere.f90
        * end_timing in file timing.f90
    - check_stop in file **stop_file.f90**
        * report in file library.f90
        * start_timing in file timing.f90
        * MPI_AllREDUCE
        * check_start in file stop_file.f90
        * end_timing in file timing.f90
    - write_output in file **write_output.f90**
        * output in file output_common.f90
        * move_satellites in file satellites.f90
        * write_restart in file restart.f90
        * logfile in file logfile.f90
- Loop End
- finalize_gitm in file **finalize.f90**
    - UA_calc_electrodynamics in file calc_electrodynamics.f90
    - output in file output_common.f90
    - write_restart in file restart.f90
    - end_timing in file timing.f90
    - report_timing in file timing.f90
    - UAM_XFER_destroy in file ModSphereInterface.f90
    - UAM_write_error in file ModSphereInterface.f90
    - stop_gitm in file library.f90
    - MPI_FINALIZE
- stop_gitm in file **library.f90**
    - CON_stop in file main.f90
    - MPI_abort

Chapter 2

Getting Started

2.1 Extracting the code from a tar file

Create a new and empty directory, and open the tar file you received, e.g.:

```
mkdir Gitm
cd Gitm
mv ../gitm.tgz .
tar -xvzf gitm.tgz
```

2.2 Checking out the code with CVS

If CVS (Concurrent Versions System) is available on your computer and you have an account on the CVS server machine herot.engin.umich.edu, you can use CVS to install the current or a particular version of the code. First of all have the following environment variables:

```
setenv CVSROOT UserName@herot.engin.umich.edu:/CVS/FRAMEWORK
setenv CVS_RSH ssh
```

where UserName is your user name on herot. Here it is assumed that you use csh or tcsh. Also put these settings into your .cshrc file so it is automatically executed at login. Alternatively, use

```
CVSROOT=UserName@herot.engin.umich.edu:/CVS/FRAMEWORK
export CVSROOT
CVS_RSH=ssh
export CVS_RSH
```

under sh, ksh and bash shells, and also put these commands into your .bashrc or .profile file so it is automatically executed at login.

Once the CVS environment variables are set, you can download the current (HEAD) version of the GITM distribution with

```
cvs checkout GITM2
```

If you want a particular version, use

```
cvs checkout -r v2_0 GITM2
```

where v2_0 is the it tag associated with the version. To download bug fixes or new features, the

```
cvs update
```

command can be used. See `man cvs` for more information.

A lot of times, you don't really want the `GITM2` directory to stay that name, since you might download a couple different version (maybe one for development and one for runs). Therefore, typically you will:

```
mv GITM2 GITM2.Development
```

2.3 Configuring and Making GITM

In order to compile GITM, you have to configure it first. The configure script is inherited from the Space Weather Modeling Framework. There are two primary reasons you need to do the configure: (1) put the right `Makefile` in the right place, specifying the compiler and the version of MPI that you will use to link the code; (2) put the right MPI header in the right place. It also does some things like hard-codes the path of the source code into the `Makefile`. Currently the configure script is not capable of detecting the system and compilers available. Some examples for commonly used set-ups are shown below. Make sure that your .cshrc or .bashrc file is set up to detect the appropriate compilers before attempting to install GITM.

Installing on Nyx:

```
./Config.pl -install -compiler=ifortmpif90 -earth
```

Installing with an Intel compiler and OpenMPI (such as Pleiades):

```
./Config.pl -install -compiler=ifort -earth
```

Installing a computer with gfortran and OpenMPI:

```
./Config.pl -install -compiler=gfortran -earth
```

Installing on a computer with gfortran and not using MPI:

```
./Config.pl -install -compiler=gfortran -earth -nompi
```

Sometimes people have a hard time with the ModUtilities.F90 file. If you have errors with this file, try (for example):

```
./Config.pl -uninstall
./Config.pl -install -compiler=gfortran -earth -noflush
```

Don't forget, after configuring the Makefiles, you must still compile the code!

```
make
make test_earth
make install
```

2.4 Running the Code

GITM requires a bunch of files to be in the right place in order to run. Therefore, it is best to use the makefile to create a run directory:

```
make rundir
mv run myrun
```

where `myrun` can be whatever you want. I will use `myrun` as an example. You can actually put this directory where ever you want. On many systems (such as nyx), there is a `nobackup` scratch disk that you are supposed to use for runs, instead of your home directory. If you need to ensure that your home directory doesn't use too much space, moving the run directory onto a disk with more free space can solve the problem:

```
make rundir
mv run /nobackup/myaccount/gitm/myrun
ln -s /nobackup/myaccount/gitm/myrun .
```

This creates a shortcut to the `myrun` directory location on `nobackup` in your GITM working directory. It allows you to treat the run directory as if it were a local directory, but it isn't! It also means that you don't have to compile and install GITM on the scratch disk, where program storage may not be allowed.

Once you have created the run directory, you can run the default simulation, by:

```
cd myrun
mpirun -np 4 GITM.exe
```

Or, if your system uses Mpiexec:

```
cd myrun
mpiexec ./GITM.exe
```

This, hopefully should run GITM for Earth for 5 minutes. If it doesn't work, then you might have mpi set up incorrectly. The default is to allow you to run 4 blocks per processor, and the default `UAM.in` file is set up for 4 blocks, so you could try just running GITM without mpi, just to see if it works at all:

```
./GITM.exe
```

If that doesn't work, then it probably didn't compile correctly. Hopefully, it just worked!

2.5 Post Processing

GITM, by default, produces one file per block per output. If you are outputting often and you are running with many blocks, you can produce a huge number of files. To post process all of these files, simply:

```
cd UA
./pGITM
```

This merges all of the files for one time period, for one file type into the same file. You can actually running this while the code is running, since GITM doesn't use old files, unless you are using the APPENDFILE option. As implied by the option's name, APPENDFILE opens an existing file and appends the most recent data to it. This feature is typically used only when running a satellite track though GITM. More information on the APPENDFILE option is located in Chapter 3 Section 3.1.3.

If you are NOT using satellites and NOT using APPENDFILE, then you are free and clear to use pGITM as often as you want during a run. To avoid deleting a file that GITM is currently writing to, it is recommended that pGITM be run as part of a script in which there is a five minute pause between executions.

Another useful script, when running GITM on another system, is given in the block below. It occasionally executes an rsync between the computer that you run GITM on and your home computer. This allows you to bring over and evaluate the output files as they become available. To do this, execute pGITM at a set cadence (60 seconds in the example) while GITM is running and then rsync the remote and home directories (excluding all unprocessed files). Finally, remove the processed and rsynced files to prevent the remote directory from filling up. Be sure to replace yourname@home.computer with your user and computer names! This is a very simple, but very useful script.

```
#!/bin/csh
rm -f stop
set LOC=$1

if (-f remoteloc) set LOC=`cat remoteloc`
```

```
while (!(-f stop))
  rsync -vrae ssh log.* UAM.* imf* yourname@home.computer:$LOC
  cd UA ; ./pGITM ; rsync --exclude '*.[bsh][0123ae][0123456789at]*' -vrae ssh d
ata yourname@home.computer:$LOC ; cd ..
  sleep 60
end
```

2.6 The Code Won't Compile!!

I'm sorry. I tried to make this work on many different platforms, but sometimes machines are very specific, and it just doesn't work out of the box. Here are some ideas on how to quickly get this thing compiling.

Can't find the right Makefile.whatever

If make does not work, then there is probably a problem with not finding the FORTRAN 90 compiler. The platform and machine specific Makefiles are in srcMake. If you type:

```
uname
ls srcMake
```

If you don't see a file named something like Makefile.uname (where uname is the output of the uname command), then you will have to build a proper general Makefile.

You will need a little a little information about your computer, like what the mpif90 compiler is called and where it is located. Take a look at srcMake/Makefile.Linux, and try to figure out what all of the flags are for your system. Then create a srcMake/Makefile.uname with the correct information in it.

The compiler doesn't recognize flag -x

You have an operating system that is recognized, but probably a different compiler. In the srcMake/Makefile.uname file (where uname is the output of the uname command), there is a line:

OSFLAGS = -w -dusty

You need to change this line to something more appropriate for your compiler. Try deleting the flags and compile. If that doesn't work, you will have to check the man pages of your compiler.

src/ModHwm.90 doesn't compile

Certain versions of gfortran (4.6 and later) may give the following error:

```
src/ModHwm.f90:168.22:

        call HWMupdate(input,last,gfs,gfl,gfm,gvbar,gwbar,gbz,gbm,gzwght,glev,u
                       1
Error: Dummy argument 'ebz' of procedure 'hwmupdate' at (1) has an attribute
 that requires an explicit interface for this procedure

src/ModHwm.f90:168.22:

        call HWMupdate(input,last,gfs,gfl,gfm,gvbar,gwbar,gbz,gbm,gzwght,glev,u
                       1
Error: Dummy argument 'ebz' of procedure 'hwmupdate' at (1) has an attribute
 that requires an explicit interface for this procedure
```

This is caused by the inputs in HWM. The latest incarnations of gfortran don't allow optional inputs that are not declared. More information about this can be found at:

`http://cosmocoffee.info/viewtopic.php?p=5136`

A solution to this problem is currently being sought.

Chapter 3

Inputs

3.1 Principle Input

`UAM.in` contains the majority of the variables that can be changed in a GITM run. Very few of these input options are required as input, all other options will default to values specified by the ModInputs.f90 subroutine, which can be found in the `src` directory. Example `UAM.in` files for various types of runs are made available in the `srcData` directory.

This input file can be altered at will without the need to recompile the code either "by hand" or using a script such as `makerun.pl`. This script, which is located on `harot.engin.umich.edu:/bigdisk1/bin`, takes command line input to create a UAM.in file and any desired auxiliary files, discussed in more detail in section 3.3, using locally stored data.

The following sections present the GITM input options grouped by type. This is not the order you will find them in a typical `UAM.in` file or the order you will find the input processed in `ModInputs.f90`. This reinforces the flexibility of the input file, which doesn't care what order the input options are specified with one exception. The starting and ending times (discussed in section 3.1.1) *must* be defined before any solar or geomagnetic indices (discussed in section 3.1.7). It is probably a good practice to begin any `UAM.in` file either with the output options (discussed in section 3.1.3) or the starting and ending times.

3.1.1 Required Inputs

The only inputs that must be specified are the starting and ending times. These times are specified using universal time blocks that specify the date (A.D.) and time of day. Remember, these two blocks must be specified before any of the solar or geomagnetic index blocks, outlined in section 3.1.7.

TIMESTART

This input block sets the starting time of the simulation. Although the input time can be defined down to the second, construction scripts will ignore temporal units smaller than a day since a lead time of a day is typically desired for any model run. Shorter runs should only used to test the model compilation.

GITM can be set to restart after pausing. This has no effect on TIMESTART, these values should always contain the real starting time, not the restart time.

```
#TIMESTART
(integer) year
(integer) month
(integer) day
(integer) hour
```

```
(integer) minute
(integer) second
```

TIMEEND

This input block sets the ending time of the simulation. As mentioned above, GITM runs typically last a minimum of two days (with one day as start-up to allow the processes to settle into equilibrium) and so the hour, minute, and second options are commonly ignored.

```
#TIMEEND
(integer) year
(integer) month
(integer) day
(integer) hour
(integer) minute
(integer) second
```

3.1.2 Temporal Boundaries

The input options described here are optional and used to set the temporal boundaries in GITM, including the processing times.

PAUSETIME

This input option sets a time for the code to pause. There is no good reason to use this option.

```
#PAUSETIME
(integer) year
(integer) month
(integer) day
(integer) hour
(integer) minute
(integer) second
```

CPUTIMEMAX

This sets the maximum CPU time that the code should run before it starts to write a restart file and end the simulation. It is very useful on systems that have a queueing system and has limited time runs. Typically, set it for a couple of minutes short of the maximum wall clock, since it needs some time to write the restart files.

```
#CPUTIMEMAX
(real) CPUTimeMax
```

CFL

The CFL option defines how large a time step GITM will take. This is set as a fraction of the maximum time step, so 1.0 is the maximum value, while 0.75 is a typical value. If instabilities form during a model run, lowering this option is a good first step to take.

```
#CFL
(real) cfl
```

3.1.3 Defining Outputs

The input option blocks described here are used to specify the level and type of verboseness.

LOGFILE

Log files are very important and the LOGFILE input option allows the user to control how much information the GITM logfile will contain. This logfile is output into the `UA/data/` directory in a file following a naming convention like `log*.dat`. Although you can output the log file at whatever frequency you would like, setting the time-step to some very small value will allow GITM to produce a log output every iteration, which is a good thing. DtLogFile specifies this time-step in seconds.

```
#LOGFILE
(real) DtLogFile
```

DEBUG

This will set the level of GITM's verboseness. Set the iDebugLevel to 0 for minimal output and to 10 to retrieve *everything*. You can also limit output to a specific processor using iDebugProc. The processors are named PE x, where PE indicates that you are tagging a processor and x is the zero offset integer indicating the processor number. So "PE 0" would indicate that output from the first processor is desired.

If you set the iDebugLevel to 0 (the default), you can set the debug report time step using DtReport. This time step is given in seconds. The final keyword in this input option, UseBarriers, forces the different nodes running the GITM code to sync up with each other frequently when employed.

```
#DEBUG
(integer) iDebugLevel
(integer) iDebugProc
(real) DtReport
(logical) UseBarriers
```

SATELLITES

To improve model-data studies, GITM can output model results along the satellite paths. Any number of satellites may be flown through a GITM simulation. Output from the satellite paths is provided at the specified universal time, geographic latitude and longitude. Instead of confining the model output to the satellite altitude, simulated output is provided over the entire GITM altitude range.

```
#SATELLITES
(integer) nSats
(string) SatFile(n)
(real) DtPlot(n)
       ⋮
```

To fly several satellites through GITM, the user must set nSats, SatFile(n), and DtPlot(n) for each satellite. nSats specifies the number of satellites, SatFile(n) gives the name of the satellite flight path file (discussed in more detail in section 3.3.4), and DtPlot(n) specifies the time-step for the GITM satellite output in seconds. The following verbatim block shows the arrangement for two satellites.

```
#SATELLITES
2 nSats
grace.sat SatFile1
1 DtPlot1
champ.sat SatFile2
1 DtPlot2
```

APPENDFILES

GITM defaults to creating many output files for satellite runs, but this option tells GITM to create just one single file per satellite. This makes GITM output significantly less files. In the future, this option may be available for other types of output as well.

```
#APPENDFILES
(logical) DoAppendFiles
```

SAVEPLOT

This input option specifies the types of files GITM will output. The most common type is 3DALL, which outputs all primary state variables. Types (specified in OutputType) include : 3DALL, 3DNEU, 3DION, 3DTHM, 3DCHM, 3DUSR, 3DGLO, 2DGEL, 2DMEL, 2DUSR, 2DTEC, 1DALL, 1DGLO, 1DTHM, 1DNEW, 1DCHM, 1DCMS, and 1DUSR. These file types specify the number of dimensions (3D, 2D, 1D) in which GITM may be run and the different primary state variables that are included in the output (NEU stands for neutrals, ION stands for ionosphere, THM stands for thermosphere, CHM stands for BLAH, USR stands for BLAH, GLO stands for day glow, TEC stands for total electron content, GEL stands for global electric BLAH, and MEL stands for BLAH). Any number of output files greater than zero may be produced, but nOutputTypes must be used to specify the number of types to be created and an OutputType and DtPlot must be provided for each output just as was done for multiple satellites. DtRestart and DtPlot specify the time-step in seconds for producing restart and output files, respectively. If no output files are specified, GITM will crash when attempting to finalize all procedures.

```
#SAVEPLOT
(real) DtRestart
(integer) nOutputTypes
(string) OutputType
(real) DtPlot
       ⋮
```

PLOTTIMECHANGE

This option allows you to change the output cadence of the files for a limited time. If you have a short event, such as a solar flare or an eclipse, this options lets you output much more often during the event than during the quiet periods preceding and succeeding the event.

```
#PLOTTIMECHANGE
(yyyy mm dd hh mm ss ms) PlotTimeChangeStart
(yyyy mm dd hh mm ss ms) PlotTimeChangeEnd
```

3.1.4 Restart Input

This section deals with options typically only specified in a restart header. These options can be used in the principle input file but have very little use if a run does not need to be restarted from a specified point in a previous run.

RESTART

Setting this input option to T (true) allows the model to restart from a previous run.

```
#RESTART
(logical) DoRestart
```

ISTEP

This input should not be specified by the user, but is used by the computer to restart after a run has been interrupted. It gives the iteration number that the model run stopped at.

```
#ISTEP
(integer) iStep
```

TSIMULATION

This options sets the offset from the starting time (specified using TIMESTART) to the current simulation time in seconds. Like ISTEP, this option should really only be used by the computer to restart after a run has been interrupted.

```
#TSIMULATION
(real) tsimulation
```

3.1.5 Numerical Options

The optional inputs in this section allow the user to modify how GITM handles computational scenarios.

LIMITER

The flux limiter specifies the balance between the model's ability to run robustly and provide realistic gradients. A realistic atmosphere will occasionally experience sharp changes in characteristics such as electron density. When GITM attempts to model these variations, the high resolution of the temporal and spatial grids allows the numerical schemes that solve the equations of state to oscillate about the actual value. One solution to the problem is to err on the side of robustness and require more gradual changes. This can be done by setting TypeLimiter to minmod or setting TypeLimiter to beta and BetaLimiter to 1.0 (both of these options are the same and the default). The other solution is to allow sharp changes to occur, realizing that the model will likely overshoot and undershoot and that if an atmospheric characteristic shoots to an unrealistic value the model may crash.

Although there are many different limiter types (see the wikipedia article for a decent introduction: `http://en.wikipedia.org/wiki/Flux_limiter`) GITM uses the Osher function [Chakravarthy and Osher, 1983], given below in equation 3.1.

$$\phi_{os}(r) = max[0, min(r, \beta)], \quad (1 \leq \beta \leq 2); \quad \lim_{r\to\infty} \phi_{os}(r) = \beta \tag{3.1}$$

In this function, β is specified by BetaLimiter, and so defaults to the minmod flux limiter function when $\beta = 1$ [Roe, 1986] and to the monotonized central (MC) flux limiter function when $\beta = 2$ [**?**].

```
#LIMITER
(string) TypeLimiter
(real) BetaLimiter
```

3.1.6 Data Assimilation

This section contains the input options that allow GITM to perform data assimilation.

RCMR

RCMR stands for Retrospective Cost Model Reference, and is a retrospective cost adaptive control (RCAC) method of data assimilation. MRAC is useful when dealing with nonlinear systems [Ali et al., 2012]. To use this system, Markov parameters need to be set. These parameters will vary based on the type and number of data sources being assimilated

and the driver being estimated. Markov parameters for the assimilation of a single satellite providing neutral density to drive the $F_{10.7}$ are currently available. Other parameters are currently being developed.

Because RCMR is used to provide an initial $F_{10.7}$ value, this input block must be included before the F107 input block. It should also be included after the SATELLITE block, so that the validity of the specified satellite index can be determined. The initial $F_{10.7}$ should not be close to the desired final value, as a perturbed location allows the assimilation to better account for model error.

```
#RCMR
(logical) Assimilate neutral density
(logical) Assimilate vertical TEC
(real) Initial F10.7
(integer) Number of satellites to include in data assimilation
(integer) Index number of satellite to assimilate
```

3.1.7 Solar and Geomagnetic Indices

This section contains the input options that specifies the level of solar and geomagnetic activity. Recall that these input options must be specified after the starting and ending times (refer to section 3.1.1) are defined. Also, note that several of these indices may be specified in different ways. If you use more than one method to define an index, the last definition will be used. It is better to just use one method in your `UAM.in` file to avoid confusion.

F107

Sets the $F_{10.7}$ and 81 day average of $F_{10.7}$ (commonly denoted as $F_{10.7a}$ or $\overline{F_{10.7}}$). This is used to set the range of the initial altitude grid and drive the lower boundary conditions. It is also used as input into several of the models called by GITM. The $F_{10.7}$ is a measure of the solar 10.7 cm flux [Covington, 1948] and is measured in solar flux units (1 sfu = 10^{-22} W m^{-2} Hz^{-1} = 10,000 jansky). The $F_{10.7}$ is commonly used as a proxy for the solar EUV output.

```
#F107
(real) f107
(real) f107a
```

NGDC_INDICES

This input option allows GITM to use an appropriate $F_{10.7}$ for each day that the assimilation runs, instead of a fixed value. This is accomplished by providing an input file that contains the $F_{10.7}$ values for the entire GITM simulation period and the preceding 81 days. This allows GITM to compute the appropriate daily $F_{10.7a}$ as well. Typically, users simply make a file including all $F_{10.7}$ values from 1980 onward, so they never have to worry about missing any preceding days.

```
#NGDC_INDICES
(string) filename
```

KP

The Kp index is not used by GITM, but several models that GITM calls utilize it. The Kp index is a measure of mid-latitude geomagnetic disturbance and can be obtained from:

`ftp://ftp.ngdc.noaa.gov/STP/GEOMAGNETIC_DATA/INDICES/KP_AP/`

while a description of the index can be found at:

`http://www-app3.gfz-potsdam.de/kp_index/description.html`

```
#KP
(real) kp
```

HEMISPHERICPOWER

This option sets the hemispheric power of the aurora. Typical values range from $1-1000$, with 20 being a nominal, quiet time value.

```
#HPI
(real) HemisphericPower
```

SOLARWIND

This option is one way to set the driving conditions for the high-latitude electric field models. Because this option is static and will not change during the run, it is usually better to use the MHD_INDICES option instead, as MHD_INDICES provides a dynamic driving environment.

```
#SOLARWIND
(real) bx
(real) by
(real) bz
(real) vx
```

MHD_INDICES

Use this option to provide GITM with dynamic IMF and solar wind conditions. This option can either be omitted entirely; if it is desired a filename whose contents will be discussed in more detail in section 3.3.1 must be specified. This file should be located in the same directory as the `UAM.in` file.

```
#MHD_INDICES
(string) filename
```

EUV_DATA

This input option allows the user to specify the extreme ultraviolet (EUV) flux using solar spectra from any model or data source. A FISM file [Chamberlin et al., 2007, 2008] is used as an example in section 3.3.3.

```
#EUV_DATA
(logical) UseEUVData
(string) cEUVFile
```

3.1.8 Physical Processes

The following options let the user specify how GITM treats different physical boundaries and processes.

INITIAL

This option specifies the initial conditions and the lower boundary conditions. For Earth, we typically just use MSIS and IRI for initial conditions. For other planets, the vertical temperature parameters can be set here. TempMin and TempMax specify the initial temperatures at the bottom and top of the thermosphere in Kelvin, respectively. TempHeight specifies the height (in kilometers from the surface of the planet) of the midpoint of the temperature gradient. TempWidth specifies the vertical width (in kilometers) of the temperature gradient.

```
#INITIAL
(logical) UseMSIS
(logical) UseIRI
```

```
If UseMSIS is false (F) then:
(real) TempMin
(real) TempMax
(real) TempHeight
(real) TempWidth
```

STATISTICALMODELSONLY

There can be benefits to running a simpler ionosphere and thermosphere models. GITM can be used as a shell to run MSIS and IRI without any coupling. This input option bypasses GITM and only outputs results from MSIS and IRI. This option allows the user who is familiar with GITM to get MSIS and IRI output without having to install or know how to run these models independently. The UseStatisticalModelsOnly flag can be set to `T` or `F` (true or false) and the DtStatisticalModels options lets you specify the time-step (in seconds) used in MSIS and IRI.

```
#STATISTICALMODELSONLY
(logical) UseStatisticalModelsOnly
(real) DtStatisticalModels
```

TIDES

This option uses a series of logical flags to tell GITM how to use tides. The first flag (UseMSISFlat) tells GITM to use MSIS without tides. The second flag (UseMSISTides) is using MSIS with full up tides. One of these two flags should be true. The remaining flags should only be true if UseMSISTides is false, and then only one should be true. The third flag (UseGSWMTides) uses GSWM tides, while the forth flag (UseWACCMTides) sets GITM to use the WACCM tides. Essentially, only one tidal model should be used per GITM run. Information about GSWM can be found at: `http://www.hao.ucar.edu/modeling/gswm/gswm.html` and information about WACCM can be found at:
`http://www.cesm.ucar.edu/working_groups/WACCM/`.

When running with the MSIS or WACCM tides, no additional input files are needed. However, when using GSWM additional files must be made accessible to GITM in the `run/UA/DataIn` directory. These files are stored in the `Tides` directory within GITM. You can check these files out using CVS (as described in Chapter 2.2) and then create links or copy them into your run directory by using: `ln -s /GITM/Tides/*bin /GITM/run/UA/DataIn/.`

```
#TIDES
(logical) UseMSISFlat
(logical) UseMSISTides
(logical) UseGSWMTides
(logical) UseWACCMTides
```

GSWMCOMP

If you decided to use GSWM tides by selecting UseGSWMTides in the TIDES option, you can specify which diurnal and semidiurnal tidal components to include.

```
#GSWMCOMP
(logical) GSWMdiurnal(1)
(logical) GSWMdiurnal(2)
(logical) GSWMsemidiurnal(1)
(logical) GSWMsemidiurnal(2)
```

DAMPING

This option specifies whether or not to allow damping of the vertical wind oscillations that can occur in the lower atmosphere.

```
#DAMPING
(logical) UseDamping
```

GRAVITYWAVE

A switch to add heating to the atmosphere. This was primarily used to explore gravity waves on Titan and has little to no effect on the terrestrial atmosphere. This options defaults to false.

```
#GRAVITYWAVE
(logical) UseGravityWave
```

NEWELLAURORA

This input option provides several logical flags that allow GITM to use Pat Newell's aurora model, Ovation [**?**].

```
#NEWELLAURORA
(logical) UseNewellAurora
(logical) UseNewellAveraged
(logical) UseNewellMono
(logical) UseNewellWave
(logical) UseNewellRemoveSpikes
(logical) UseNewellAverage
```

AMIEFILES

This option allows the user to specify the electrodynamic state of the polar regions using the Assimilative Mapping of Ionospheric Electrodynamics (AMIE) model [Ridley and Kihn, 2004]. Input files separately specify the conditions at the northern and southern polar caps, and should be in the MIE binary format. Contact Aaron Ridley for more information about using AMIE with GITM.

```
#AMIEFILES
(string) cAMIEFileNorth
(string) cAMIEFileSouth
```

THERMO

This input option sets flags that allow the user to turn various thermospheric processes on and off. For a realistic run, all three heating flags (for solar, Joule, and auroral heating) should be set to true, along with both cooling flags (for nitrous-oxide, NO, and oxygen, O). UseConduction, UseUpdatedTurbulentCond, and UseTurbulentCond flag should also all be set to true. UseDiffusion, however, defaults to false and should not be used for a realistic run.

The only keyword that is not a flag is EddyScaling. This allows the user to control the Eddy diffusion by applying a multiplicative to the model's Eddy diffusion value.

```
#THERMO
(logical) UseSolarHeating
(logical) UseJouleHeating
(logical) UseAuroralHeating
(logical) UseNOCooling
```

```
(logical) UseOCooling
(logical) UseConduction
(logical) UseTurbulentCond
(logical) UseUpdatedTurbulentCond
(logical) UseDiffusions
(real) EddyScaling
```

THERMALDIFFUSION

This keyword sets the thermal conductivity. KappaTemp0 should be set to the desired value of the thermal conductivity in MKS units.

```
#THERMALDIFFUSION
(real) KappaTemp0
```

VERTICALSOURCES

This input option provides flags and limits for the vertical thermospheric sources. The UseEddyInSolver turns the Eddy diffusion on and off, the UseNeutralFrictionInSolver turns the vertical neutral friction on and off, and the MaximumVerticalVelocity sets the limit for vertical neutral winds in meters per second.

```
#VERTICALSOURCES
(logical) UseEddyInSolver
(logical) UseNeutralFrictionInSolver
(real) MaximumVerticalVelocity
```

EDDYVELOCITY

This input option further allows the user to specify how Eddy diffusion is treated by providing an flag to use the method presented by **?**, which was developed for the Martian atmosphere. The UseEddyCorrection flag is another option that is useful for certain extraterrestrial atmospheres.

```
#EDDYVELOCITY
(logical) UseBoquehoAndBlelly
(logical) UseEddyCorrection
```

DIFFUSION

If you use Eddy diffusion, as indicated by the UseDiffusion flag, you must specify two pressure levels in units of UNITS using EddyDiffusionPressure[0,1]. Below the first pressure level, the Eddy diffusion will be constant. Between the first and the second pressure levels, the Eddy diffusion coefficient will drop-off linearly. Thus, the first pressure must be larger than the second! The Eddy diffusion coefficient may also be specified using EddyDiffusionCoef.

```
#DIFFUSION
(logical) UseDiffusion
(real) EddyDiffusionCoef
(real) EddyDiffusionPressure0
(real) EddyDiffusionPressure1
```

WAVEDRAG

This input option allows stress heating to be applied in GITM, providing an additional source of drag in the thermosphere.

```
#WAVEDRAG
(logical) UseStressHeating
```

FORCING

This input option provides flags for physical processes that drive the thermosphere.

```
#FORCING
(logical) UsePressureGradient
(logical) UseIonDrag
(logical) UseNeutralFriction
(logical) UseViscosity
(logical) UseCoriolis
(logical) UseGravity
```

IONFORCING

This input option turns several ionospheric transport processes on and off. UseExB turns the $\mathbf{E}\times\mathbf{B}$ plasma drift on and off, UseIonGravity allows the user to determine whether or not to allow the ions to respond to gravity, UseNeutralDrag relegates the forcing from ion-neutral collisions on the ionosphere, and UseIonPressureGradient deals with the ion motions resulting from changes to the plasma pressure gradient. All of these processes should be on for a realistic simulation.

```
#IONFORCING
(logical) UseExB
(logical) UseIonPressureGradient
(logical) UseIonGravity
(logical) UseNeutralDrag
```

DYNAMO

This input option provides flags for the ionospheric dynamo. The UseDynamo flag turns on a model to calculate the ionospheric electric fields. DynamoHighLatBoundary sets the polar latitude limit (65° or 70° is a realistic limit), nItersMax sets the maximum number of iterations for the Dynamo convergence (typically set at 500), and the MaxResidual sets the maximum difference, in Volts, to allow between iterations before the value is said to have converged (typically set to 1 V). The Dynamo currently uses a static high-latitude boundary, though a dynamical one would be ideal. To get around this, input from AMIE, run using the SuperMag data network, can be used instead of the UseDynamo process. The use of AMIE input is discussed in section 3.1.8.

```
#DYNAMO
(logical) UseDynamo
(real) DynamoHighLatBoundary
(integer) nItersMax
(real) MaxResidual
```

ELECTRODYNAMICS

Sets the time-step (in seconds) for updating the high-latitude (and low-latitude) electrodynamic drivers, such as the potential and the aurora.

```
#ELECTRODYNAMICS
(real) DtPotential
(real) DtAurora
```

IONPRECIPITATION

This is a highly specific input option for experienced users. Most users should set this option to false. This is only for people wishing to do very specific runs in high altitude regions. IonIonizationFilename and IonHeatingRateFilename allow the advanced user to specify the ionization and ion heating rates due to precipitation using auxiliary input files.

```
#IONPRECIPITATION
(logical) UseIonPrecipitation
(string) IonIonizationFilename
(string) IonHeatingRateFilename
```

LTERADIATION

LTERadiation allows the user to specify the time-step for the local thermodynamic equilibrium radiation process in seconds. This option is typically only used by the Mars GITM. It may be set for other versions of GITM, but each planet treats this input option differently. Versions not compiled for Mars will either modify or ignore input from LTERadiation.

```
#LTERadiation
(real) DtLTERadiation
```

DIPOLE

This input option allows the user to specify the parameters the geomagnetic field within the limits of a dipole configuration so that any configuration (centered, tilted, or off-center and tilted) may be used. x,y,zDipoleCenter specify the origin of the dipole field in ECI coordinates, MagneticPoleTilt gives the angle (in degrees) between the dipole axis and the terrestrial rotation axis, and MagneticPoleRotation gives the constant angular rate (in degrees) at which the geomagnetic dipole field rotates.

```
#DIPOLE
(real) MagneticPoleRotation
(real) MagneticPoleTilt
(real) xDipoleCenter
(real) yDipoleCenter
(real) zDipoleCenter
```

APEX

This input option indicates whether or not GITM should use the more realistic Apex geomagnetic field [Richmond, 1995]. If this option is set to false, the dipole field specified in the previous subsection will be used.

```
#APEX
(logical) UseApex
```

3.1.9 Geographic Boundaries

TOPOGRAPHY

This input option, which is defaults to false, uses topography to provide a variable vertical grid in GITM. You can also specify the minimum altitude at which the vertical grid for the thermosphere and ionosphere become consistent (AltMinUniform and AltMinIono, respectively). As with all other inputs the altitudes should be given in kilometers from the surface of the planet.

```
#TOPOGRAPHY
(logical) UseTopography
(real) AltMinUniform
(real) AltMinIono
```

ALTITUDE

The upper and lower altitude limits (in kilometers) may be specified here. These values default to 500 km and 95 km, respectively. Setting the altitude limits above or below these values may result in unstable model runs.

UseStretchedAltitude may also be turned on and off here. This flag defaults to true, as it allows the altitudes with rapidly changing pressure levels to be treated in more detail than those where the atmosphere changes slowly with altitude.

```
#ALTITUDE
(real) AltMin
(real) AltMax
(logical) UseStretchedAltitude
```

GRID

This input option sets the grid resolution in GITM. Although this process is straightforward, it deserves some thought and discussion. This is provided in section 3.2. For new GITM users, learning how to alter the grid is a good place to start moving beyond the test run shown in Chapter 1.2.

```
#GRID
(integer) nBlocksLon
(integer) nBlocksLat
(real) LatStart
(real) LatEnd
(real) LonStart
(real) LonEnd
```

STRETCH

You can stretch the grid in GITM in the latitudinal direction. It takes some practice to get the stretching just the way that you might like. To do this, you need to specify the geographic latitude (in degrees) that the stretching will center at (ConcentrationLatitude), the fraction of stretch (StretchingPercentage) where 0 means that there will be no stretching and 1 means that the latitude will be stretched by 100%, and a multiplicative factor (StretchingFactor) that helps scale the stretching. The StretchingFactor should range between 0 and 1, erring closer to 1 to provide more control.

```
#STRETCH
(real) ConcentrationLatitude
(real) StretchingPercentage
(real) StretchingFactor
```

Here is an example for stretching near the equator:

```
#STRETCH
0.0             ! Equator
0.7             ! Amount of stretching is great
0.8             ! more control
```

Here is another example for no stretching near the auroral oval:

```
#STRETCH
65.0 ! Location of minimum grid spacing
0.0   ! There is no streching here
1.0   ! For control
```

3.2 Setting the Grid

Setting the grid resolution in GITM is not very complicated, but it does involve some thought. There are a few variables that control this. In `ModSize.f90`, the following variables are defined:

```
integer, parameter :: nLons = 9
integer, parameter :: nLats = 9
integer, parameter :: nAlts = 50

integer, parameter :: nBlocksMax = 4
```

The first three variables (nLons, nLats and nAlts) define the size of a single block. In the example above, there are 9 cells in latitude, 9 cells in longitude and 50 cells in altitude. The latitude and longitude resolution in GITM is defined by the cell numbers specified here and the block numbers discussed in the following section. The size of the altitude cells are measured in scale heights instead of kilometers, as certain regions require higher resolutions than others based on the dominating chemical and dynamical processes. Each altitude cell contains $\frac{1}{3}$ of a scale height, starting at 80 km and typically reaching up to 500 km. Increasing the number of altitude blocks will increase the altitude range, but if this value is increased too much the model becomes unstable. This altitude limit makes it problematic to model the equatorial region during storms, where increased vertical plasma transport requires the model consider altitudes of 2000 km and higher.

The final variable (nBlocksMax) defines the maximum number of blocks you can have on a single processor. Most people run with one single block per processor, as this is faster. So setting this to "1" is usually preferred, and is necessary when running with the Dynamo option on. This is a necessity because the Dynamo routine does not correctly transfer the geomagnetically oriented data into the geographic coordinates used in the ghost cells. Hopefully this will be updated in future versions, as allowing multiple blocks per node can, in theory, save memory.

Don't forget, if you change any of these parameters you will need to recompile the code (run the Makefile again) so that a new executable using the newly defined variables can be produced.

3.2.1 Running 3D Over the Whole Globe

Once the number of cells is defined, then the number of blocks in latitude and longitude need to be defined. This is done in the `UAM.in` file, as shown in section 3.1. For example, the initial settings have 8 blocks in latitude and 8 in longitude:

```
#GRID
8           lons
8           lats
-90.0       minimum latitude to model
```

```
90.0        maximum latitude to model
0.0         minimum longitude to model
0.0         maximum longitude to model
```

The number of cells in the simulation domain will then be 72 in longitude, 72 in latitude, and 50 in altitude. Given that there are 360° in longitude and 180° in latitude, the resolution would be $360^\circ/72 = 5.0^\circ$ and $180^\circ/72 = 2.5^\circ$ in latitude. If one block were put on each processor, 64 processors would be required.

If one desired to run without the Dynamo turned on, the problem could also be run with multiple blocks per node. This grid fits quite nicely on either four- or eight-core processors. However, on 12-core processors this would not work very well at all. We have started to run simulations of 1° in latitude and 5° in longitude, which can fit nicely on a 12-core processor machine. For example, in `ModSize.f90`:

```
integer, parameter :: nLons = 9
integer, parameter :: nLats = 15
```

and in `UAM.in`:

```
#GRID
8 lons
12 lats
```

This can then run on 96 cores, which is nicely divisible by 12. Essentially, an infinite combination of cells per block and number of blocks can be utilized. Typically, the number of blocks in latitude and longitude are even numbers.

3.2.2 Running 3D Over the Part of the Globe

GITM can be run over part of the globe - both in latitude and in longitude. It can be run over a single polar region (by setting either the minimum or maximum latitude to be greater (or less) than $\pm 90^\circ$). If this is selected, message passing over the poles is implemented. If the pole is not selected, then boundary conditions have to be set in `set_horizontal_bcs.f90`. By default, a continuous gradient boundary condition is used on the densities and temperatures, while a continuous value is used on the velocity. This is true in both latitude and longitude. In longitude, message passing is implemented all of the time, but the values are over-written by the boundary conditions if the maximum and minimum longitude are not equal to each other.

The longitudinal resolution ($\Delta\phi$) is set by:

$$\Delta\phi = \frac{\phi_{end} - \phi_{start}}{nBlocksLon \times nCellsLon} \tag{3.2}$$

while, the latitudinal resolution ($\Delta\theta$) is set by:

$$\Delta\theta = \frac{\theta_{end} - \theta_{start}}{nBlocksLat \times nCellsLat} \tag{3.3}$$

3.2.3 Running in 1D

GITM can run in 1D mode, in which the call to advance_horizontal is not completed. This means that GITM runs exactly the same way, but ignoring all of the horizontal advection terms. You have to do two things to make GITM run in 1D. First, in `ModSize.f90`:

```
integer, parameter :: nLons = 1
integer, parameter :: nLats = 1
integer, parameter :: nAlts = 50

integer, parameter :: nBlocksMax = 1
```

This tells the code that you only want one single latitude and longitude location. To specify the exact location, in UAM.in:

```
#GRID
1           lons
1           lats
41.75       minimum latitude to model
41.75       maximum latitude to model
275.0       minimum longitude to model
275.0       maximum longitude to model
```

This is pretty close to some place in Michigan. GITM will model this exact point for as long as you specify. One thing to keep in mind with running in 1D is that the Earth still rotates, so the spot will have a strong day to night variation in temperature. In 3D, the winds decrease some of the variability between day and night, but in 1D, this doesn't happen. So, the results are going to be perfect. But, 1D is great for debugging.

3.3 Auxiliary Input Files

If you have access to the University of Michigan Atmospheric, Oceanic, and Space Science resources, the data needed for these auxiliary input files are on herot.engin.umich.edu. The following descriptions will allow you download to create your own auxiliary input files yourself, but this process is much more simple if you have access to the resources on herot. Recall that makerun.pl, previously mentioned at the beginning of section 3.1, will create many of the following input files when it builds a UAM.in file. All of these auxiliary input files should be kept in the same directory as the GITM executable and the UAM.in file.

3.3.1 IMF and Solar Wind

This file controls the high-latitude electric field and aurora when using models that depend on the solar wind and interplanetary magnetic field (IMF). It allows GITM to dynamically control these quantities. You can create either realistic IMF files or hypothetical ones.

For realistic IMF files, we typically use CDF files downloaded from the CDAWEB ftp site, located at: http://cdaweb.gsfc.nasa.gov/cdaweb_anonymousftp.htm.

On herot, an IDL code (called cdf_to_mhd.pro) merges the solar wind and IMF files to create one single file. This IDL code also propagates the solar wind and IMF from L1 to 32 Re upstream of the Earth. You can use the DELAY statement to shift the time more (e.g. in the example below, it shifts by an additional 15 minutes). cdf_to_mhd.pro requires both a solar wind file and an IMF file. For example, the IMF file would be ac_h0_mfi_20011231_v04.cdf and the solar wind file would be ac_h0_swe_20011231_v06.cdf. The code assumes that the data starts at #START, and ends when it encounters an error. This can mean that if there is an error in the data somewhere, the code will only read up to that point. To validate that the solar wind and IMF is what you think it is, it is recommended that you use the IDL code imf_plot.pro to check the output before using it to run GITM. Here is an example file:

```
This file was created by Aaron Ridley to do some wicked cool science thing.

The format is:
 Year MM DD HH Mi SS mS   Bx  By   Bz     Vx   Vy   Vz    N        T

Year=year
MM = Month
DD = Day
```

```
HH = Hour
Mi = Minute
SS = Second
mS = Millisecond
Bx = IMF Bx GSM Component (nT)
By = IMF By GSM Component (nT)
Bz = IMF Bz GSM Component (nT)
Vx = Solar Wind Vx (km/s)
Vy = Solar Wind Vy (km/s)
Vz = Solar Wind Vz (km/s)
N  = Solar Wind Density (/cm3)
T  = Solar Wind Temperature (K)

#DELAY
900.0

#START
 2000  3 20  2 53  0  0  0.0 0.0  2.0 -400.0  0.0  0.0  5.0  50000.0
 2000  3 20  2 54  0  0  0.0 0.0  2.0 -400.0  0.0  0.0  5.0  50000.0
 2000  3 20  2 55  0  0  0.0 0.0  2.0 -400.0  0.0  0.0  5.0  50000.0
 2000  3 20  2 56  0  0  0.0 0.0  2.0 -400.0  0.0  0.0  5.0  50000.0
 2000  3 20  2 57  0  0  0.0 0.0  2.0 -400.0  0.0  0.0  5.0  50000.0
 2000  3 20  2 58  0  0  0.0 0.0  2.0 -400.0  0.0  0.0  5.0  50000.0
 2000  3 20  2 59  0  0  0.0 0.0  2.0 -400.0  0.0  0.0  5.0  50000.0
 2000  3 20  3  0  0  0  0.0 0.0 -2.0 -400.0  0.0  0.0  5.0  50000.0
 2000  3 20  3  1  0  0  0.0 0.0 -2.0 -400.0  0.0  0.0  5.0  50000.0
 2000  3 20  3  2  0  0  0.0 0.0 -2.0 -400.0  0.0  0.0  5.0  50000.0
 2000  3 20  3  3  0  0  0.0 0.0 -2.0 -400.0  0.0  0.0  5.0  50000.0
 2000  3 20  3  4  0  0  0.0 0.0 -2.0 -400.0  0.0  0.0  5.0  50000.0
```

To actually read in this file, in `UAM.in`, use the input option MHD_INDICES described in section 3.1.7.

3.3.2 Hemispheric Power

The hemispheric power files describe the dynamic variation of the auroral power going into each hemisphere. Models such as Fuller-Rowell and Evans [1987] use the Hemispheric Power to determine which level of the model it should use. The Hemispheric Power is converted to a Hemispheric Power Index using the formula shown in equation 3.4.

$$HPI = 2.09 \log(HP)^{1.0475} \tag{3.4}$$

The National Oceanic and Atmospheric Administration (NOAA) provides these hemispheric power files for public use online at `http://www.swpc.noaa.gov/ftpmenu/lists/hpi.html`. There are two types of formats used for hemispheric power files (due to a change in the NOAA output format in 2007). Both file formats can be used by GITM, and are shown in the examples below.

Example file 1 for data prior to 2007:

```
# Prepared by the U.S. Dept. of Commerce, NOAA, Space Environment Center.
# Please send comments and suggestions to sec@sec.noaa.gov
#
# Source: NOAA POES (Whatever is aloft)
# Units: gigawatts
```

```
# Format:

# The first line of data contains the four-digit year of the data.
# Each following line is formatted as in this example:

# NOAA-12(S)  10031     9.0  4    .914

# Please note that if the first line of data in the file has a
# day-of-year of 365 (or 366) and a HHMM of greater than 2300,
# that polar pass started at the end of the previous year and
# ended on day-of-year 001 of the current year.

# A7    NOAA POES Satellite number
# A3    (S) or (N) - hemisphere
# I3    Day of year
# I4    UT hour and minute
# F8.1  Estimated Hemispheric Power in gigawatts
# I3    Hemispheric Power Index (activity level)
# F8.3  Normalizing factor

2000
NOAA-15(N)  10023    35.5  7    1.085
NOAA-14(S)  10044    25.3  7     .843
NOAA-15(S)  10114    29.0  7     .676
NOAA-14(N)  10135   108.7 10    1.682
NOAA-15(N)  10204    36.4  7    1.311
.
.
.
```

Example file 2 for data in and after 2007:

```
:Data_list: power_2010.txt
:Created: Sun Jan  2 10:12:58 UTC 2011


# Prepared by the U.S. Dept. of Commerce, NOAA, Space Environment Center.
# Please send comments and suggestions to sec@sec.noaa.gov
#
# Source: NOAA POES (Whatever is aloft)
# Units: gigawatts

# Format:

# Each line is formatted as in this example:

# 2006-09-05 00:54:25 NOAA-16 (S)  7  29.67   0.82

# A19   Date and UT at the center of the polar pass as YYYY-MM-DD hh:mm:ss
# 1X    (Space)
# A7    NOAA POES Satellite number
# 1X    (Space)
```

```
# A3    (S) or (N) - hemisphere
# I3    Hemispheric Power Index (activity level)
# F7.2  Estimated Hemispheric Power in gigawatts
# F7.2  Normalizing factor

2010-01-01 00:14:37 NOAA-17 (N)  1   1.45   1.16
2010-01-01 00:44:33 NOAA-19 (N)  1   1.45   1.17
.
.
.
```

This file is not specified in `UAM.in`, instead different subroutines within GITM will use it as needed.

3.3.3 Solar Irradiance

To provide GITM with realistic solar irradiance, the solar EUV must be specified. This can be done through a file containing modeled or observed solar irradiance data. An example from the FISM model is shown below.

```
#START
    2009       3      20       0       0       0   0.00389548   0.00235693
   0.00127776  0.000907677  0.000652528  0.000372993  0.000250124  0.000194781
  0.000389686  0.000118650   0.00642058   0.00618358  0.000133490  7.67560e-05
  7.80045e-05  0.000145722  5.92577e-05  5.95070e-05  0.000102437  6.48526e-05
  8.94509e-05  0.000101928  5.94333e-05  5.36012e-05  1.51744e-05  1.10265e-05
  1.26937e-05  2.16591e-05  9.57055e-06  1.82608e-05  7.07992e-05  2.55451e-05
  1.12451e-05  6.89255e-05  3.03882e-05  2.33862e-05  2.98026e-05  4.44682e-05
  1.50847e-05  3.00909e-05  8.18379e-05  3.52176e-05  0.000416491  0.000269080
  0.000269080  0.000275734  6.60872e-05  4.46671e-05  0.000220697  0.000512933
  3.85239e-05  9.30928e-05  2.71239e-05  1.23011e-05  1.05722e-05  9.30876e-06
  7.08442e-07  3.54221e-07  1.77110e-07
    2009       3      20       0       1       0   0.00389548   0.00235693
   0.00127776  0.000907677  0.000652528  0.000372993  0.000250124  0.000194781
  0.000389686  0.000118650   0.00642058   0.00618358  0.000133490  7.67560e-05
  7.80045e-05  0.000145722  5.92577e-05  5.95070e-05  0.000102437  6.48526e-05
  8.94509e-05  0.000101928  5.94333e-05  5.36012e-05  1.51744e-05  1.10265e-05
  1.26937e-05  2.16591e-05  9.57055e-06  1.82608e-05  7.07992e-05  2.55451e-05
  1.12451e-05  6.89255e-05  3.03882e-05  2.33862e-05  2.98026e-05  4.44682e-05
  1.50847e-05  3.00909e-05  8.18379e-05  3.52176e-05  0.000416491  0.000269080
  0.000269080  0.000275734  6.60872e-05  4.46671e-05  0.000220697  0.000512933
  3.85239e-05  9.30928e-05  2.71239e-05  1.23011e-05  1.05722e-05  9.30876e-06
  7.08442e-07  3.54221e-07  1.77110e-07
.
.
.
```

GITM knows to use the provided solar irradiance file through the EUV_DATA input option specified in the `UAM.in` file. More information about this input option can be found in section 3.1.7. For more information on specifying to solar irradiance, please contact Professors Ridley or Pawlowski.

3.3.4 Satellites

GITM can provide output data at a list of times and locations using the SATELLITE input option, described in more detain in section 3.1.3. Although this is designed to output data along a satellite orbit, any list of locations may be

used. There is currently no routine to create a satellite input file, but the format is simple and may be easily constructed from a satellite ASCII data file using awk. Here is a sample satellite input file:

```
year mm dd hh mm ss msec long lat alt
#START
2002 4 16 23 34 25 0 299.16 -2.21 0.00
2002 4 16 23 34 25 0 293.63 -1.21 0.00
2002 4 16 23 34 25 0 291.28 -0.75 0.00
2002 4 16 23 34 25 0 289.83 -0.45 0.00
2002 4 16 23 34 25 0 288.79 -0.21 0.00
2002 4 16 23 34 25 0 287.98 -0.01 0.00
2002 4 16 23 34 25 0 287.32  0.16 0.00
2002 4 16 23 34 25 0 286.76  0.31 0.00
2002 4 16 23 34 25 0 286.26  0.46 0.00
2002 4 16 23 34 25 0 285.81  0.60 0.00
2002 4 16 23 34 25 0 285.39  0.74 0.00
```

Note that the satellite output is not specified in this sample file. This is because altitude entry doesn't matter t this time, GITM ignores the altitude and outputs altitudinal profiles of the atmospheric characteristics at each geographic location and universal time. Although millisecond accuracy is provided, GITM should not be output at a resolution smaller than 1 second. The temporal resolution in the satellite file does not need to match the output resolution.

Chapter 4

Outputs

Now that you have managed to successfully complete a GITM run you've found yourself with a bunch of output files. All of the GITM output is in mks units and this data is contained within several files located in the `UA/data` directory, as was previously discussed in Chapter 2 Section 2.5. You will have found yourself with several `iriOut_*.dat` files, a `log*.dat` file, and many `.bin` files in whichever formats you specified in SAVEPLOT (see Chapter 3 Section 3.1.3). The `iriOut_*.dat` files are required by the IRI model and not typically used when analyzing the outcome of the GITM run.

The log file provides useful information about the run, such as whether a restart was performed, which physical processes were used, and a list of the universal time, time-step, neutral temperature ranges (T), solar and geomagnetic indices, and the neutral velocity (VV) ranges for each iteration. This file can be very useful when sharing runs with other users, when revisiting an old run, or merely ensuring that GITM performed as expected. An example log file is provided below:

```
## Inputs from UAM.in
# Resart= F
# Eddy coef:   100.000 Eddy P0:     0.020 Eddy P1:     0.003 Eddy Scaling:     1.000
# Statistical Models Only:  F Apex:  T
# EUV Data:  TFile:
fismflux.dat
# AMIE: none
none
# Solar Heating:  T Joule Heating:  T Auroral Heating:  T
# NO Cooling:  T O Cooling:  T
# Conduction:  T Turbulent Conduction:  T Updated Turbulent Conduction:  T
# Pressure Grad:  T Ion Drag:  T Neutral Drag:  T
# Viscosity:  T Coriolis:  T Gravity:  T
# Ion Chemistry:  T Ion Advection:  T Neutral Chemistry:  T

#START
   iStep yyyy mm dd hh mm ss  ms      dt min(T) max(T)...
   ...mean(T) min(VV) max(VV) mean(VV) F107 F107A By Bz Vx...
   ...HP HPn HPs SubsolarLon SubsolarLat SubsolarVTEC
       2 2011  9 23  0  0  2 297  2.2979  168.75192  1062.87354...
       ...933.09984 -48.19362    524.93645  1.01910 159.3 127.9 -4.6  0.5 406.9...
       ...11.1 14.4  15.5  3.14145  -0.37655  45.73188
       .
       .
       .
```

The 3DALL output binary files can contain the following atmospheric quantities:

Altitude: Altitude from the surface of the planet (m)

Ar: Argon density (m^{-3})

Ar Mixing Ratio: Argon mixing ratio

CH4 Mixing Ratio: Methane mixing ratio

Conduction: Heat conduction

EuvHeating: EUV Heating rate

H: Hydrogen density (m^{-3})

H!U+!N: H^+ density (m^{-3})

H2 Mixing Ratio: Molecular Hydrogen mixing ratio

HCN Mixing Ratio: Hydrogen Cyanide mixing ratio

He: Helium density (m^{-3})

He!U+!N: He^+ density (m^{-3})

Heaing Efficiency: Heating efficiency

Heat Balance Total: Heat balance total

Latitude: Geographic latitude (degrees)

Longitude: Geographic longitude (degrees)

N!D2!N: N_2 density (m^{-3})

N!D2!U+!N: N_2^+ density (m^{-3})

N!U+!N: N^+ density (m^{-3})

N(!U2!ND): N(^{2}D) density (m^{-3})

N(!U2!NP): N(^{2}P) density (m^{-3})

N(!U4!NS): N(^{4}S) density (m^{-3})

N2 Mixing Ratio: Molecular nitrogen mixing ratio

NO: Nitrious Oxide density (m^{-3})

NO!U+!N: NO^+ density (m^{-3})

O!D2!N: O_2 density (m^{-3})

O!D2!U+!N: O_2^+ density (m^{-3})

O(!U1!ND): O(^{1}D) density (m^{-3})

O(!U2!ND)!U+!N: O(^{2}D) density (m^{-3})

O(!U2!NP)!U+!N: O(^{2}P) density (m^{-3})

O(!U3!NP): O(^{3}P) density (m^{-3})

O_4SP_!U+!N: O($_4$SP)$^+$ density (m^{-3})

RadCooling: Radiative Cooling rate

Rho: Neutral density (m^{-3})

Temperature: Neutral temperature (K)

V!Di!N (east): Ion velocity towards geographic East (m s^{-1})

V!Di!N (north): Ion velocity towards geographic North (m s^{-1})

V!Di!N (up): Vertical ion velocity (m s^{-1})

V!Dn!N (east): Neutral velocity towards geographic East (m s^{-1})

V!Dn!N (north): Neutral velocity towards geographic North (m s^{-1})

V!Dn!N (up): Vertical neutral velocity (m s^{-1})

V!Dn!N (up,N!D2!N): Vertical N_2 velocity (m s^{-1})

V!Dn!N (up,N(!U4!NS)): Vertical N(^{4}S) velocity (m s^{-1})

V!Dn!N (up,NO): Vertical NO velocity (m s^{-1})

V!Dn!N (up,O!D2!N): Vertical O_2 velocity (m s^{-1})

V!Dn!N (up,O(!U3!NP)): Vertical O(^{3}P) velocity (m s^{-1})

e-: electron density (m^{-3})

eTemperature: electron temperature (K)

iTemperature: ion temperature (K)

time: Universal time

There are many routines available to process and analyze the GITM binary files. The majority of these routines are written in IDL and are available in the `srcIDL` directory within the GITM model directory. Currently 50 routines have been saved in this directory and more are under development. Alternatively, python routines are currently being developed and these are located in the `srcPython` directory. Please not that when using the IDL reader the universal time is read in as epoch seconds from January 1, 1965 00:00 UT, while when using the python reader, the time is imported as a datetime object.

4.1 IDL

Here is an complete list with some description of the IDL processing and visualization routines currently available. Please feel free to update this section for other GITM users when you CVS your vetted GITM processing routines.

gitm_read_bin

This is a routine to read a GITM bin file into IDL. This is great when you want to get a handle on the data and experiment with different visualization methods.

thermo_plotsat

This is the most commonly used routine to plot the 1D GITM results. It can also be used to plot satellite files and other 1D simulations. It is relatively straight forward to use, but experimentation can be help. This is an actual program, so you have to `.run` it.

thermo_gui

This is a someone simplistic graphical user interface code for plotting 3D results. The filename has to be entered manually in the upper left. You then have to press the button for loading the file. Variables appear on the left side, and you can select which one you want to plot. You then select which of the available planes you would like to look at (lat/lon, lat/alt, or lon/alt) or scroll through the options. This interface allows you to add wind vectors, plot in polar coordinates, and plot the log of the variable.

thermo_batch_new

This code will let you look at at 3D files exactly the same way as thermo_gui, but is all scripted. There are a few features that this has that thermo_batch doesn't have:

1. You can use wildcards for the file name, so that a list of files can be read. The postscript file names created for each figure will be differentiated by appending numbers sequentially so that no figures are overwritten.
2. When plotting a lat/alt plane, you can do a zonal average.
3. You can do a global average.

thermo_plotter

All of the above plotting codes will only plot one plot per page. This code will plot many more than one plot per page. You can plot multiple variables on the same page, or multiple files with the same variable, or both.

Other IDL Routines

Please feel free to provide a description of these routines so that GITM users do not waste their time rewriting code that already exists.

- **ask**
- **c_a_to_r**
- **c_a_to_s**
- **chopr**
- **closedevice**
- **c_r_to_a**
- **c_s_to_a**
- **get_position**
- **makect**
- **mklower**
- **mm**
- **plotct**
- **plotdumb**
- **plotmlt**
- **pos_space**
- **read_thermosphere_file**
- **setdevice**
- **thermo_batch**
- **thermo_calcforce**
- **thermo_champ**
- **thermo_compare**
- **thermo_compare_time**
- **thermo_convert_champfiles**
- **thermo_guvi**
- **thermo_magequator**
- **thermo_make_summary**
- **thermo_mkguvisat**
- **thermo_mksatsave**
- **thermo_mksave**
- **thermo_mktec**
- **thermo_on2**
- **thermo_plotdist**
- **thermo_plotlog**
- **thermo_plot_new**

- **thermo_plot**
- **thermo_plotsat2**
- **thermo_plotsat_constalt_ON2**
- **thermo_plotsat_constalt**
- **thermo_plotvectors**
- **thermo_readsat**
- **thermo_sigma**
- **thermo_superposed**
- **thermo_tec**
- **thermo_temp**
- **tostr**

4.2 Python

This section provides a complete list of the vetted GITM python routines. These routines require that you use PyBats, a module included in SpacePy. This is a library developed for space physics applications by the scientists at Los Alamos and can be downloaded for free at: `http://spacepy.lanl.gov`

Another library, Basemap, is required for certain plotting routines. Basemap is a part of the Matplotlib Toolkit and can be installed using Fink, Macports, or downloaded at: `http://matplotlib.org/basemap/`

If you have questions about these routines or are at the University of Michigan and want to start using Python, Dr. Welling is the man to see. The source code behind the PyBats GITM routines are located in

gitm.py

GITM is a PyBats submodule that handles input and output from GITM. It can be helpful for those wishing to write their own GITM processing routines but doesn't contain any analysis or visualization routines.

Once you have downloaded and installed Spacepy, the gitm submodule can be accessed via `import spacepy.pybats.gitm`. This module contains the following routines:

GitmBin: A routine to load a GITM output bin file.

PbData: The base class for all PyBats data container classes. Used to hold the GITM data read in using GitmBin

dt: A shortcut for datetime.

np: A shortcut for numpy.

dmarray: A shortcut for data arrays. Used in many of the data container classes defined in PbData.

gitm_plot_rout.py

Common routines used to format and analyze GITM data.

add_colorbar: Add a color bar to a contour plot. This routine does not depend on SpacePy.

center_polar_cap: Adjust radial coordinates to produce a centered polar plot. Necessary for the northern hemisphere, where polar plots assume the radial (latitude) coordinates should be centered at zero instead of 90°. This routine does not depend on SpacePy.

find_zlimits: Find the upper and lower limits for a list of GITM data arrays.

localtime_to_glon: Find the longitude at a specified universal time and local time.

gitm_3D_global_plots.py

Routines to build and output GITM output variable contour plots over a geographic range. Several different standard plot formats are available, and routines useful for creating custom figures are also included.

plot_single_3D_image: This is a basic visualization routine that creates a filled contour plot of a single output variable from a GITM 3D at a specified altitude or 2D bin file. The output variable is plotted as a function of latitude and longitude over the entire globe, though the latitude range may be limited. The output plot may be polar or rectangular and a map of the Earth's continental boundaries may also be included in the output figure. Sample output of the electron temperature is shown in figure 4.1 (a) and (b).

plot_single_nsglobal_3D_image: A quick way to examine GITM output at both poles. This routine creates two polar contour plots centered at the geographic northern and southern poles for a single output variable from a GITM 3D at a specified altitude or 2D bin file. The equatorial and polar latitude boundaries may both be specified, though they cannot change between hemispheres. The Earth's continental boundaries may also be included in the output figure. Sample output of the electron temperature is shown in figure 4.1 (c)

plot_global_3D_snapshot: A snapshot of a single GITM output over the entire globe. This routine creates two polar contour plots centered at the geographic northern and southern poles and extending to 45° and a single rectangular plot containing the mid- and low-latitudes for a single output variable from a GITM 3D at a specified altitude or 2D bin file. The Earth's continental boundaries may also be included in the output figure. Sample output of the electron temperature is shown in figure 4.1 (d)

plot_mult_3D_slices: This routine creates a single plot containing multiple global contours of a GITM output variable from a 3D or 2D bin file at a list of specified altitudes. These plots may be either polar or rectangular, with or without continental outlines, and within a specified latitude range. Sample output of the electron temperature is shown in figure 4.2.

plot_rectangular_3D_global: This routine plots a single rectangular filled contour for a GITM output variable at a specified altitude index as a function of latitude and longitude. Title, colorbar, and continental outlines are optional. A handle to the contour plot is returned to allow the output to be further manipulated depending on what other subplots are included in the output figure.

plot_polar_3D_global: This routine plots a single polar filled contour for a GITM output variable at a specified altitude index as a function of latitude and longitude. Title, colorbar, and continental outlines are optional. A handle to the contour plot is returned to allow the output to be further manipulated depending on what other subplots are included in the output figure. The longitude at the top of the plot may also be specified, this allows one to ensure a specific local time is always located at the top of the dial using a routine like **localtime_to_glon**.

gitm_alt_plots.py

Routines to build and output GITM output variable linear and contour plots over an altitude range. Several different standard plot formats are available, and routines useful for creating custom figures are also included.

plot_single_alt_image: Creates a single linear or contour altitude plot.

plot_mult_alt_image: Creates a figure with multiple linear or contour altitude plots.

plot_alt_slices: Creates a figure with a contour plot showing the altitude dependence of a quantity as a function of latitude or longitude with several linear altitude slices at specified locations. An example is shown in figure 4.3

plot_linear_alt: Plots the the linear altitude dependence of a quantity, with altitude on the y-axis.

plot_3D_alt: Plots the altitude dependence of a quantity as the function of another spatiotemporal coordinate with the spatiotemporal coordinate on the x-axis, altitude on the y-axis, and the desired quantity as a color contour.

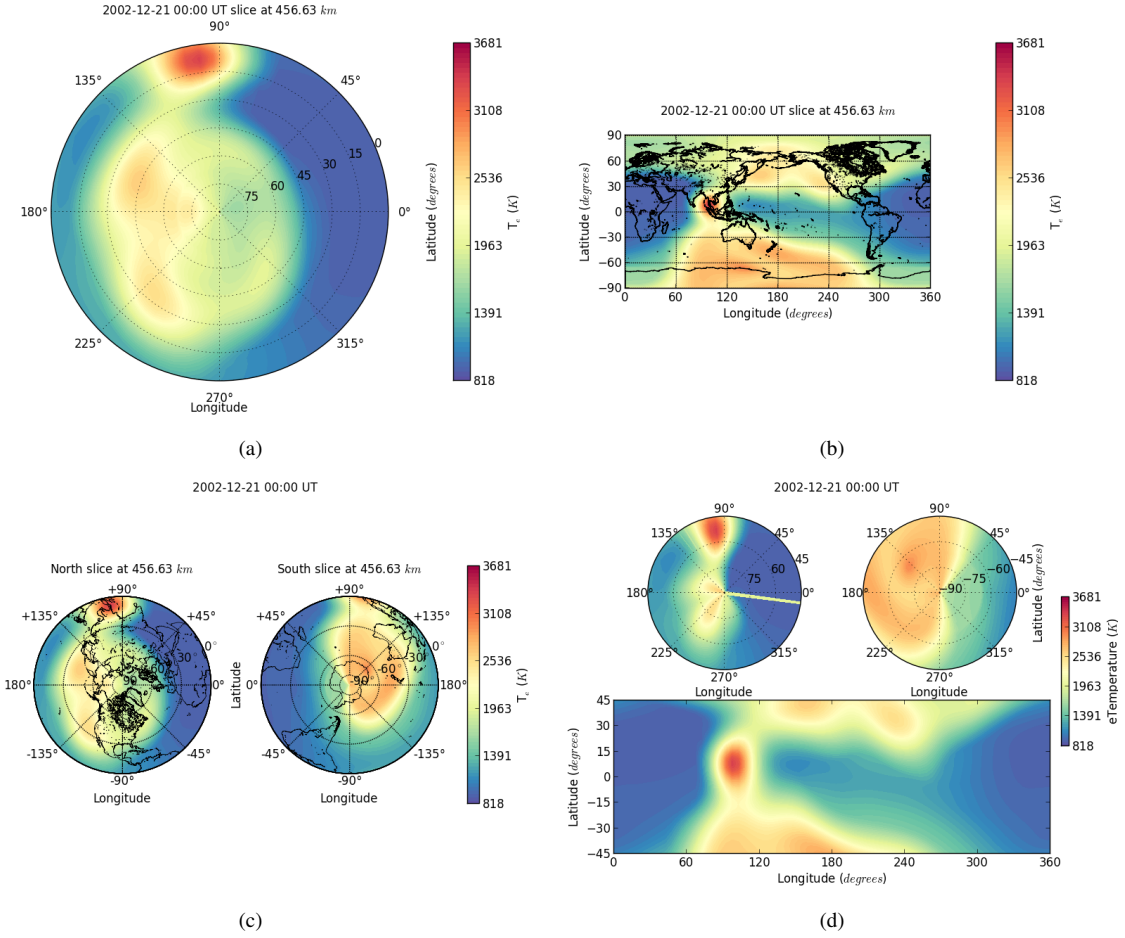
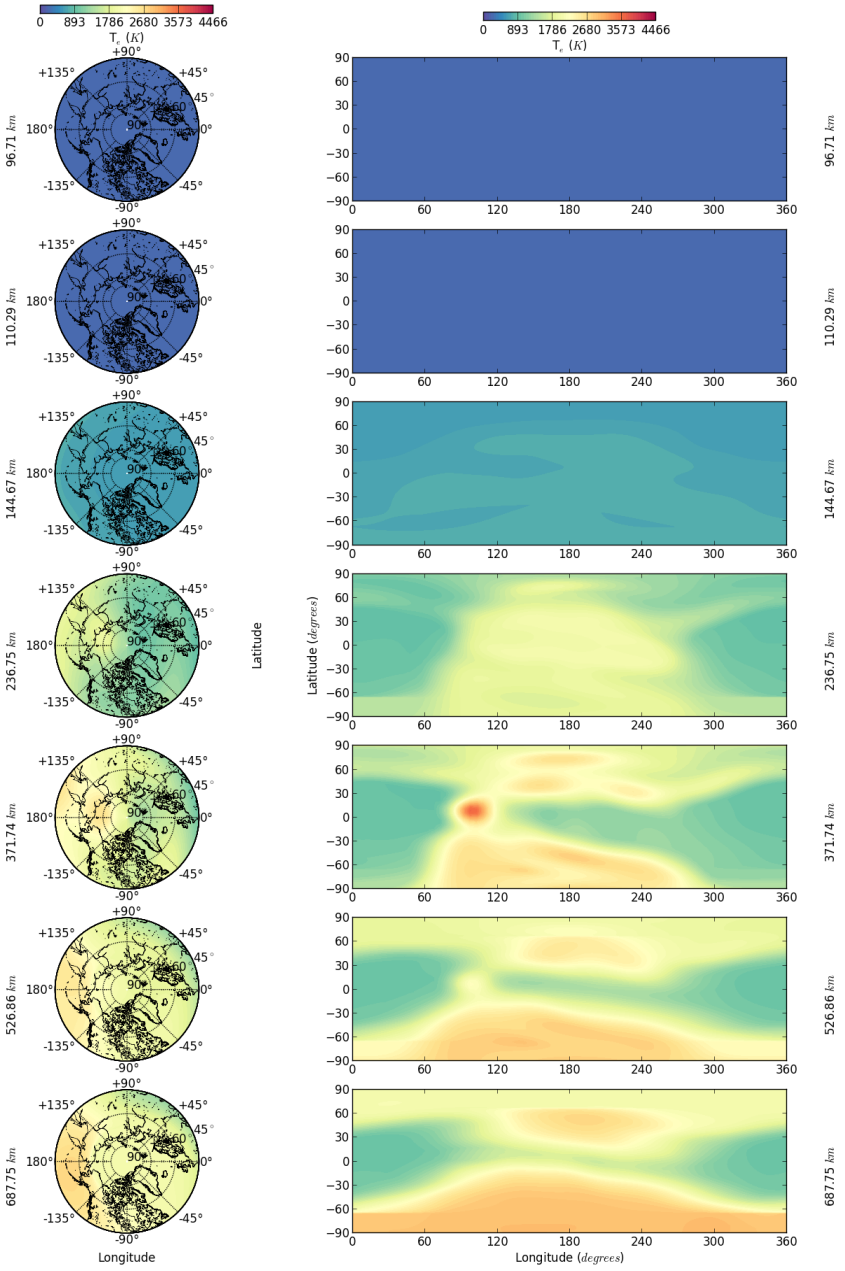


Figure 4.1: GITM electron temperature at 456.63 km altitude for: (a) northern latitudes, (b) over the entire globe, (c) over the entire globe, as viewed from the poles, and (d) as a global snapshot.



(a) (b)

Figure 4.2: GITM electron temperature at seven altitude slices for (a) northern latitudes and (b) the entire globe.

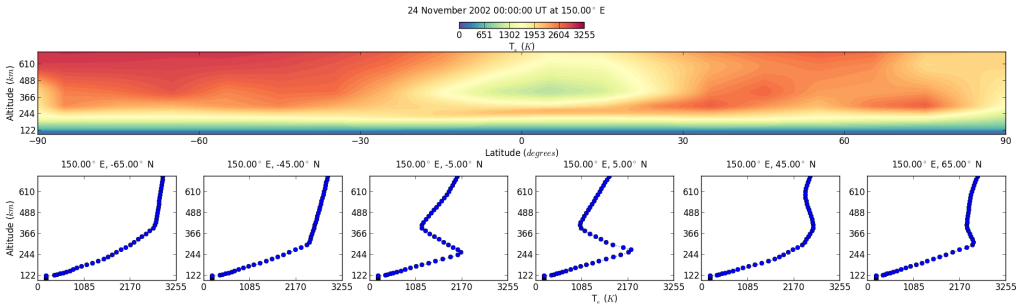


Figure 4.3: GITM electron temperature at a constant longitude with six latitude slices.

Bibliography

Ali, A. A., K. Agarwal, A. M. D'Amato, A. J. Ridley, and D. S. Bernstein (2012, August). Retrospective-Cost Subsystem Identification for the Global Ionosphere-Thermosphere Model. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Minneapolis, MN (USA), pp. 1–12. University of Michigan.

Chakravarthy, S. R. and S. Osher (1983, July). High resolution applications of the Osher upwind scheme for the Euler equations . *6th Computational Fluid Dynamics Conference*, 363–372.

Chamberlin, P. C., T. N. Woods, and F. G. Eparvier (2007). Flare irradiance spectral model (FISM): Daily component algorithms and results. *Space Weather 5*(7), S07005.

Chamberlin, P. C., T. N. Woods, and F. G. Eparvier (2008, May). Flare Irradiance Spectral Model (FISM): Flare component algorithms and results. *Space Weather 6*, S05001.

Covington, A. (1948). Solar noise observations on 10.7 centimeters. *Proceedings of the IRE 36*(4), 454–457.

Fuller-Rowell, T. J. and D. S. Evans (1987, July). Height-integrated Pedersen and Hall conductivity patterns inferred from the TIROS-NOAA satellite data. *Journal of Geophysical Research 92*(A7), 7606–7618.

Richmond, A. D. (1995). Ionospheric electrodynamics using magnetic apex coordinates. *Journal of Geomagnetism and Geoelectricity 47*(2), 191–212.

Ridley, A. J. and E. A. Kihn (2004, April). Polar cap index comparisons with AMIE cross polar cap potential, electric field, and polar cap area. *Geophysical Research Letters 31*, L07801.

Roe, P. L. (1986). Characteristic-based schemes for the Euler equations. *Annual review of fluid mechanics 18*(1), 337–365.