Polar Wind Outflow Model User Manual

Alex Glocer

Center for Space Environment Modeling
The University of Michigan

October 11, 2024

Contents

1 Introduction **5**
1.1 Acknowledgments . . . 5
1.2 The PWOM in a Few Paragraphs . . . 5
1.3 System Requirements . . . 6

2 Quick Start **7**
2.1 A Brief Description of the PWOM Distribution . . . 7
2.2 General Hints . . . 7
2.3 Installing the Code . . . 8
2.4 Creating Documentation . . . 9
2.5 Building and Running an Executable . . . 9
2.6 Restarting a Run . . . 10
2.7 What next? . . . 10

3 The Basics **11**
3.1 Configuration of PWOM . . . 11
3.1.1 Setting compiler flags in Makefile.conf and with Config.pl . . . 11
3.1.2 Planet Configuration . . . 12
3.2 PARAM.in . . . 12
3.2.1 Commands, Parameters, and Comments . . . 12
3.2.2 The Order of Commands . . . 12
3.2.3 Iterations, Time Steps and Time . . . 13

4 Using the PWOM as the PW component of the SWMF **15**
4.1 Configuring the SWMF to use PWOM . . . 15
4.2 Coupling the PW component with other components . . . 15

5 Example Runs **17**
5.1 Earth Examples . . . 17
5.1.1 Configuration and Compilation for the Earth Examples . . . 17
5.1.2 Example 1: Single field line simulation and visualization . . . 17
5.1.3 Example 2: Multi field line simulation and visualization . . . 18
5.2 Saturn Example . . . 18
5.2.1 Example 3: Single field line simulation . . . 19

6 Complete List of Input Commands **21**
6.1 Input Commands for the PWOM: PW Component . . . 21
6.1.1 Numerical scheme . . . 21
6.1.2 Input/output . . . 23
6.1.3 Control . . . 25

6.1.4 Physics 26

Index of Input Commands **28**

Chapter 1

Introduction

This document descibes the installation, configuration, execution and usage of the Polar Wind Outflow Model (PWOM).

1.1 Acknowledgments

The PWOM was developed at the Center for Space Environment Modeling (CSEM) of the University of Michigan. The core software designers and code developers: Alex Glocer, Gábor Tóth. The origional version of the model was developed by Tamas Gombosi.

1.2 The PWOM in a Few Paragraphs

The Polar Wind Outflow Model (PWOM), solves the gyrotropic continuity, momentum and energy equations that describe the supersonic ion outflow along open magnetic field lines in the polar region. The PWOM can simulate the polar winds of Earth and Saturn. At Earth the behavior of three ion species, O^+, H^+, and He^+ are considered, while at Saturn only two species, H^+ and H_3^+, are considered. The model assumes a stationary neutral atmosphere. Ranging in altitude from 250 km to 8,000 km for the Earth version, or 1,400 to 61,000 km for the Saturn version, the PWOM has its lower boundary set in a reservoir at chemical and thermal equilibrium, while the top boundary is at considerably lower pressure, thus creating a transonic outflow to a low pressure external medium. The considerable altitude range covers two different regimes; the collision and chemistry dominated low altitude, and the expansion dominated high altitude. Furthermore, the ambipolar electric field is calculated at every time step, and is a major contributor to ion outflow. Other physical effects that are included in the PWOM, are topside electron heating, photo-ionization, and the expanding cross-sectional area of the magnetic flux tube.

Obtaining a single field line solution yields the vertical solution at one point. The field line convects as determined from the polar cap potential. By solving several field lines distributed throughout the polar cap, the full solution can be obtained. Since each field line solution is independent, this problem is embarassingly parallel. Accordingly, we use the Message Passing Interface (MPI) to split the computational work evenly among processors.

The PWOM can be compiled into standalone or coupled mode. In the standalone mode, the polar cap potential is read from a file or calculated from the Weimer empirical model, and the polar wind solution is obtained independantly of other models.In the coupled mode, the PWOM interacts with the SWMF via a coupling interface known as a wrapper. In this case the polar cap potential is obtained from the IE component, and the fluxes at the top of the PWOM are passed to the GM component. Most of this manual is devoted to the standalone operation of the PWOM, but chapter 4 discusses some details on how to utilize the coupled model.

The PWOM has been tested on the SGI Altix, on Linux Beowulf clusters with the NAG f95 compiler. We have also run the PWOM with reasonable success under Mac OS Darwin using the XLF and NAG f95 compilers, and under Linux with the PGF90 compiler.

1.3 System Requirements

In order to install and run the PWOM the following minimum system requirements apply.

- The PWOM runs only under the UNIX/Linux operating systems. This now includes Macintosh system 10.x because it is based on BSD UNIX. The PWOM does not run under any Microsoft Windows operating system.
- A FORTRAN 90 compiler must be installed.
- The Perl interpreter must be installed.
- A version of the Message Passing Interface (MPI) library must be installed.
- Small tests such as single field-line simulations can be done on a single processor
- Very large runs require many more processors.
- In order to generate the documentation you must have LaTex installed on your system. The PDF generation requires the `dvips` and `ps2pdf` utilities.

In addition to the above requirements, the PWOM output is designed to be visualized using either IDL or Tecplot. You may be able to visualize the output with other packages, but formats and scripts have been designed for only these two visualization softwares.

Chapter 2

Quick Start

2.1 A Brief Description of the PWOM Distribution

The top level directory contains the following subdirectories:

- `src` - Source code of planet independent code
- `srcEarth` - Source code of Earth specific code
- `srcSaturn` - Source code of Saturn specific code
- `Scripts` - shell and Perl scripts
- `bin` - location of executable
- `doc` - the documentation directory
- `share` - shared scripts and source code
- `util` - general utilities such as TIMING and NOMPI

and the following files

- `README` - a short instruction on installation and usage
- `Makefile` - the main makefile
- `Config.pl` - Perl script for (un)installation and configuration

2.2 General Hints

Getting help with scripts and the Makefile

Most of the Perl and shell scripts that are distributed with the PWOM provide help which can be accessed as follows using the `-h` flag. For example,

```
Config.pl -h
```

will provide a detailed listing of the options and capabilities of the `Config.pl` script. In addition, you can find all the possible targets that can be built by typing

```
make help
```

Input commands: PARAM.XML

A very useful set of files to become familiar with are the `PARAM.XML` files.

This file contains a complete list of all input commands for the component as well as the allowed ranges for each of the input parameters. Although the XML format makes the files a little hard to read, they are extremely useful. A typical usage is to cut and paste commands out of the PARAM.XML file into the PARAM.in file for a run.

2.3 Installing the Code

The first step in installing the PWOM is untarring the distribution. If the tar program knows about the -z flag, you can open the gzipped tar files with a single UNIX command:

```
tar xzf SOMETARFILE.tgz
```

If the tar program does not recognize the -z flag, two steps are needed:

```
gunzip SOMETARFILE.tgz
tar xf SOMETARFILE.tar
```

In the following descriptions the shorter form is shown, but you may need to use the two step procedure on certain platforms.

Untar the distribution using the command:

```
tar xzf PWOM.tgz
```

Change directories into the distribution:

```
cd PWOM
```

The PWOM needs to know what architecture you are running the code on and what FORTRAN compiler will be used. For most platforms and compilers, it can figure this out all by itself. To install PWOM run the command:

```
Config.pl -install
```

in the main directory. This creates `Makefile.def` with the correct absolute path to the base directory and `Makefile.conf` which contains the operating system and compiler specific part of the Makefile. If the compiler is not the default one for a given platform (e.g. not the NAG f95 compiler for a Linux platform) then the compiler must be specified explicitly with the `-compiler` flag. If the MPI header file is not the default one, it can be specified with the `-mpi` flag. For example on a Mac machine one can select the g95 compiler with the Intel version of the MPI library using

```
Config.pl -install -compiler=g95 -mpi=Intel
```

To uninstall PWOM type

```
Config.pl -uninstall
```

If the uninstallation fails (this can happen if some makefiles are missing) do reinstallation with

```
Config.pl -install
```

and then try uninstalling the code again. To get a complete description of the `Config.pl` script type

```
Config.pl -h
```

2.4 Creating Documentation

The documentation for PWOM can be generated from the distribution by the command

```
make PDF
```

which creates the user manual

```
doc/PWOM.pdf
```

In order for this to work you must have LaTex installed on your system (and dvips and ps2pdf).

To clean the intermediate files type

```
cd doc/Tex
make clean
```

To remove all the created documentation type

```
cd doc/Tex
make cleanall
```

2.5 Building and Running an Executable

Compilation flags, such as the precision and optimization level are stored in `Makefile.conf`. This file is created on installation of the PWOM and has defaults which are appropriate for your system architecture. The precision of reals can be changed to single precision (for example) by typing

```
Config.pl -single
```

while the compiler flags can be modified with

```
Config.pl -debug -O0
```

to debug the code with 0 optimization level, and

```
Config.pl -nodebug -O4
```

to run the code at maximum optimization level and without the debugging flags.

To specify which planet the PWOM should be set to use

```
Config.pl -Earth or Config.pl -Saturn
```

To build the executable **bin/PWOM.exe**, type:

```
make
```

Depending on the configuration, the compiler settings and the machine that you are compiling on, this can take from 1 to up to 5 minutes.

The `PWOM.exe` executable should be run in a sub-directory, since a large number of files are created in each run. To create this directory use thecommand:

```
make rundir
```

This command creates a directory called `run`. You can either leave this directory as named, or `mv` it to a different name. It is best to leave it in the same PWOM directory, since keeping track of the code version associated with each run is quite important. The `run` directory will contain links to the codes which were created in the previous step as well as subdirectories where input and output will reside.

Here we assume that the `run` directory is still called `run`:

```
cd run
```

In order to run the PWOM you must have the input files: PARAM.in. The PARAM.in file contains the detailed commands for controlling what you want the code to do during the run. The default PARAM.in file in the run directory suitable to perform a simple test.

To run the PWOM interactively on four processors:

```
cd run
mpirun -np 4 PWOM.exe
```

To recompile the executable with different compiler settings you have to use the command

```
make clean
```

before recompiling the executables. It is possible to recompile only a component or just one subdirectory if the `make clean` command is issued in the appropriate directory.

2.6 Restarting a Run

There are several reasons for restarting a run. A run may fail due to a run time error, due to hardware failure, due to software failure (e.g. the machine crashes) or because the queue limits are exceeded. In such a case the run can be continued from the last saved state of PWOM.

The restart files are saved at the frequency determined in the PARAM.in file. Normally the restart files are saved into the PW/restartOUT directory. The files in that directory need only be copied into the PW/restartIN directory inorder to prepare a restarted run.

2.7 What next?

Hopefully this section has guided you through installing the PWOM and given you a basic knowledge of how to run it. So what next?

We suggest that you read all of chapter 3, which outlines the basic features of the PWOM as well as some things you really must know in order to use the PWOM. Once you have done this you are ready to experiment. Chapter 5 gives several examples which are intended to make you familiar with the use of the PWOM. We suggest that you try them!

Chapter 3

The Basics

3.1 Configuration of PWOM

Configuration refers to several different ways of controlling how the PWOM is compiled and run. The most obvious is the setting of compiler flags specific to the machine and version of FORTRAN compiler. The other methods refer to ways in which planet the PWOM is configured to solve.

3.1.1 Setting compiler flags in Makefile.conf and with Config.pl

The compiler flags can be modified by editing

```
Makefile.conf
```

This makefile is created during installation, and it contains the platform and compiler specific part of the makefile system. The most usual changes can be easily done with the Config.pl script. The precision of real numbers can be set with

```
Config.pl -single
```

or

```
Config.pl -double
```

If the precision is modified, the script will execute 'make clean', so that all the files are compiled with the new real precision.

The debugging flags can be switched on and off with

```
Config.pl -debug
```

and

```
Config.pl -nodebug
```

respectively. The maximum optimization level can be set to -O2 with

```
Config.pl -O2
```

The minimum level is 0, the maximum is 4 (Note that not all compilers support level 4 optimization).

3.1.2 Planet Configuration

The PWOM is set to have a very general core of code, and planet specific files that define the chemistry, neutral and ion species and other planet parameters. The planet specific codes are stored in srcEarth or srcSaturn. When the PWOM is confifugured for a specific planet, the planet specific codes are copied into the src directory. To configure the PWOM for earth simply type

```
Config.pl -Earth
```

to configure for Saturn

```
Config.pl -Saturn
```

3.2 PARAM.in

The input parameters for the PWOM are read from the `PARAM.in` file which must be located in the run directory.

We refer to the lines starting with a # character as commands.

For example if the command string

```
#END
```

is present, it indicates the end of the run and lines following this command are ignored. If the #END command is not present, the end of the file indicates the end of the run.

There are several features of the input parameter file syntax that allow the user to easily run the code in a variety of modes while at the same time being able to keep a library of useful parameter files that can be used again and again.

The syntax and the content of the input parameter files is defined in the PARAM.XML files.

This file can be read (and edited) in a normal editor. The same files are used to produce much of this manual with the aid of the `share/Scripts/XmlToTex.pl` script. The `Scripts/TestParam.pl` script also uses these files to check the PARAM.in file. Copying small segments of the `PARAM.XML` files into `PARAM.in` can speed up the creation or modification of a parameter file.

3.2.1 Commands, Parameters, and Comments

The commands in the PARAM.in file are necessary for instucting the PWOM to carry out the simulation. The following is an example of a parameter.

```
#TIMEACCURATE
F                        DoTimeAccurate
```

In this example, the parameter is #TIMEACCURATE and the command is DoTimeAccurate.

3.2.2 The Order of Commands

In essence, the order of parameter commands within a session is arbitrary, but there are some important restrictions. We should note that the order of the parameters following the command is not arbitrary and must exactly match what the code requires.

3.2.3 Iterations, Time Steps and Time

In several commands the frequency or 'time' of some action has to be defined. This is usually done with a pair of parameters. The first defines the frequency or time in terms of the number of time steps, and the second in terms of the simulation time. A negative value for the frequency means that it should not be taken into account. For example, in time accurate mode,

```
#SAVEPLOT
-1          DtSavePlot
2000        DnSavePlot
T           SaveFirst
```

means that a plot file should be saved after every 2000th time step, while

```
#SAVEPLOT
100         DtSavePlot
-1          DnSavePlot
T           SaveFirst
```

means that it should be saved every 100 seconds in terms of physical time.

Chapter 4

Using the PWOM as the PW component of the SWMF

The polar wind outflow can have a significant impact on the solution in the magnetosphere. Likewise, changes in the field aligned currents and ionospheric convection can modify the polar wind. Because of the importance of including such connections in our model, we now turn to the issue of how to use the PWOM as a component of the SWMF.

4.1 Configuring the SWMF to use PWOM

Configuring the SWMF to use the PWOM is straight forward. Simply type:

```
Config.pl -v=PW/PWOM
```

Doing so tells the SWMF that the PWOM will represent the PW component of the SWMF. This configuration allows the user to execute the PWOM through the SWMF without any other components.

Should the user wish to include, say, a global magnetosphere or the ionosphere electrodynamics the configuration is simply:

```
Config.pl -v=PW/PWOM,GM/BATSRUS,IE/Ridley_serial
```

Coupling with these other components is described in greater detail in the next section.

4.2 Coupling the PW component with other components

The PWOM represents the PW component of the SWMF. It can be directly coupled to the Global Magnetosphere (GM) and Ionosphere Electrodynamics (IE) components by using the

```
#COUPLE1
```

or

```
#COUPLE2
```

commands (See the SWMF manual for details). The IE component is coupled in a unidirectional manner with the PW component, by which the polar cap potentials and field aligned currents are passed to PW but no information flows in the opposite direction.

The PW-GM coupling is more complicated. The GM component can be coupled unidirectionally, with the PW component setting the densities and velocities at the GM inner boundary, or it can be set bidirectionally

where the magnetospheric pressure is used to set the upper boundary of the PW model. The latter coupling is untested so we will focus on the first form in the documentation.

There are several options for how the coupling between GM and PW can proceed. They all, however, start with the same command in the PARAM file:

```
#COUPLE1
PW          COMP1
GM          COMP2
10.0        DTcoupling
```

The default method for coupling the PW and GM components is to combine the individual fluid densities and velocities at the upper boundary of PW into a single fluid density and velocity. However, the user may want to preserve the component information so as to track magnetospheric composition. In this case, the GM component can be configured to use the multi-species equations:

```
Config.pl -o=GM:e=MhdPw
```

In this case the individual species densities are preserved, but the velocities are combined into a single fluid velocity. This is because the multi-species MHD representation of BATS-R-US solves separate continuity equations, but a single momentum and energy equation.

The user also has the option to preserve velocity information as well as density information when putting the PW output into GM. This requires the user to configure the GM component to solve the full multi-fluid equations:

```
Config.pl -o=GM:e=MultiIon
```

In this case the individual species densities and velocities are conserved, as BATS-R-US solves a full continuity, momentum, and energy equation for each species.

Chapter 5

Example Runs

In this chapter we will provide some examples of how to actually use the PWOM. After reading this chapter you should be familiar enough with the use of the model to setup your own simple runs. Three stand-alone examples are provided: Two for Earth and one for Saturn. The tests will require two separate configurations and compilations. One for each planet.

Additionally one example of using the PWOM as the PW component of the SWMF is provided. This example assumes that you already have the SWMF installed.

5.1 Earth Examples

5.1.1 Configuration and Compilation for the Earth Examples

The default configuration of the PWOM is for the planet Earth. To check the current configuration, type

```
Config.pl
```

from the PWOM directory. If the PWOM is not currently configured for Earth, then type

```
Config.pl -Earth
```

and the planet configuration will be changed to Earth. After changing the configuration to Earth, the code will need to be recompiled. This can be accomplished by simply typing

```
make
```

To carry out the examples in the following subsections, a run directory is needed. The creation of a run directory can be accomplished by typing

```
make rundir
```

This command creates a directory called 'run'. Changing into the newly created run directory, we are now prepared to carry out the example simulations

5.1.2 Example 1: Single field line simulation and visualization

In the run directory there is a PARAM.in file that is set up for 4 field lines. Changing the #FIELDLINES command from 4 to 1 allows us to simulate a single field line. The initial solution for the field line is given in

```
restartIN/restart_iline0001.dat
```

The location of the field line can be changed by editing the restart file above. The second line of said file gives the latitude and longitude of the field line foot point. Modifying these numbers will change the location of the field line. The code can then be run by typing.

```
./PWOM.exe
```

The result is given in the plots directory and the restart file is given in the restartOUT directory.

The results can be visualized using the IDL routines described in the BATS-R-US user manual. Simply change into the plots directory and you will see several files labeled as

```
plots_iline*.out
```

Where the star refers to the line number. Entering IDL you can then visualize the solution using the

```
.r getpict
.r plotfunc
```

commands as described in the BATS-R-US manual.

5.1.3 Example 2: Multi field line simulation and visualization

To carry out a run with a multiple field lines, the first thing that needs to be done, is to edit the PARAM.in file. Change the #FIELDLINES command from 1 to 125. We then need initial solutions for each field line. Type

```
cp ../Scripts/CreateRestart.pl restartIN/
```

from the run directory. Change directories into the restartIN directory and type

```
CreateRestart.pl
```

Returning to the PWOM directory, we are now prepared to carry out a multi field line simulation. Simply type

```
./PWOM.exe
```

or use mpirun to execute the code on as many processors as you desire (up to the number of field lines).

Each individual field line can be visualized as before, but sometimes a horizontal slice is desired. To visualize a 2D horizontal slice change into the plots directory. Then type

```
../Scripts/PostProcessTec.pl
../Scripts/CreateMacro.pl
```

At each command you will be prompted for the altitude slice of interest. There should now be 2D Tecplot files and a corresponding Tecplot macro file called Macro.mcr. Now just open up Tecplot and run the macro.

5.2 Saturn Example

Return to the PWOM directory. To change the planet configuration to Saturn, simply type

```
Config.pl -Saturn
```

The code must then be recompiled. To do this, type

```
make
```

5.2.1 Example 3: Single field line simulation

Move the old run directory out of the way as follows:

```
mv run run_earth
```

Then create a new run directory by typing

```
make rundir
```

Then change into the run directory. Everything should now be prepared for a Saturn simulation. Carry out the simulation as before. The visualization is then the same as before.

Chapter 6

Complete List of Input Commands

The content of this chapter is generated from the Param/PARAM.XML file. The XML file can be read with an editor and can be used for creating PARAM.in files by copying small parts from them.

The transformation of the XML format into LaTex is done with the share/Scripts/XmlToTex.pl script. This script generates index terms for all commands, which are used to create an alphabetical index at the end of this chapter.

6.1 Input Commands for the PWOM: PW Component

List of PW commands used in the PARAM.in file

6.1.1 Numerical scheme

#SCHEME command

```
#SCHEME
Godunov                 TypeSolver
Godunov                 TypeFlux
0.05                    DtVertical
F                       IsFullyImplicit
F                       IsPointImplicit
F                       IsPointImplicitAll
```

TypeSolver sets the type of solver which is to be used. Currently only two options are available: Godunov and Rusanov. These are actually both Godunov type schemes, but with a first order exact Riemann solver or a second order approximate Riemann solver. TypeFlux sets the intercell flux for the approximate Riemann solver and hence only matters when the Rusanov option is selected for the TypeSolver. The three choices are LaxFriedrichs, Rusanov, and HLL. DtVertical sets the time step for plasma propagating along the field line. IsFullyImplicit determines whether the fully implicit time stepping is used. If not, the last two parameters are also read. IsPointImplicit determines if the point implicit scheme is used or not for the collision terms. IsPointImplicitAll is true if all terms are point implicit.

The default values are shown.

#LIMITER command

```
#LIMITER
1.5                     LimiterBeta
```

Sets the beta parameter in the Rusanov scheme.

#VERTICALGRID command

```
#VERTICALGRID
390                     nPoints
2e6                     DeltaR
```

Sets the number of grid points in the vertical direction and the grid spacing.

#REGRID command

```
#REGRID
T                       DoRegrid (rest of parameters read if true)
T                       DtRegrid
F                       DoSavePoints
```

If DoRegrid is true then regrid the field footpoints every DtRegrid seconds. If DoSavePoints is true, save the footpoints and the triangulation before the regridding is done. The goal is to maintain a relatively uniform coverage of the polar region.

Default is DoRegrid false.

#TIMEACCURATE command

```
#TIMEACCURATE
T               IsTimeAccurate
```

DoTimAccurate determines if the field lines are solved in time accurate mode or in steady state mode.

#TIMESTEP command

```
#TIMESTEP
50              DtHorizontal
```

DtHorizontal is the time step for horizontal motion of the field line. Default value is shown.

#VARIABLEDT command

```
#VARIABLEDT
F                       IsVariableDt
```

When IsVariableDt=T then the vertical time step is variable based on the change in pressure.

#MOTION command

```
#MOTION
T                       DoMoveLine
```

This command determines which to move the field lines as determined by the horizontal convection, or to hold them in their initial locations. The default value is shown.

#ROTATION command

```
#ROTATION
T               UseCentrifugalForce
```

This command determines whether centrifugal forces should be included. The default is shown.

#FIELDLINE command

```
#FIELDLINE
1               nTotalLine
```

nTotalLine sets the number of field lines included in the simulation. The default is shown.

6.1.2 Input/output

#TEST command

```
#TEST
PW_move_line            StringTest
0                       iProcTest
2                       iLineTest
```

Set test parameters. The subroutines to be tested are listed in StringTest, the tested processor is iProcTest, and the tested field line is iLineTest. If iLineTest is 0, all field lines produce test output.

Default is an empty StringTest, i.e. no test output is produced.

#RESTART command

```
#RESTART
T                       IsRestart
```

If the IsRestart variable is true, then the PWOM uses a restart file. Otherwise, the PWOM uses a cold start routine. The default is shown.

#TYPEPLOT command

```
#TYPEPLOT
real4                   TypePlot (ascii/real4/real8)
```

The type (format) of the plot files written. The value "ascii" produces a simple text file, "real4" is single precision binary and "real8" is double precision binary format.

Default value is "ascii".

#SAVEPLOT command

```
#SAVEPLOT
10.0                    DtSavePlot
-1                      DnSavePlot
T                       DoSaveFirst
F                       DoAppendPlot
```

The frequency which plot files are written out are defined here. DoSaveFirst determines whether or not to save a plot on the first call. DoAppendPlot sets appending or not to previous plot file. Default values are shown.

#SAVEPLOTELECTRODYNAMICS command

```
#SAVEPLOTELECTRODYNAMICS
F                       DoPlotElectrodynamics
10                      DtPlotElectrodynamics
```

DoPlotElectrodynamics determines whether the electrodynmics plot information is saved. For instance the field aligned currents, and the polar cap potental can be saved with this command. The frequency of output is determined by the DtPlotElectrodynamics parameter.

#PLOTENERGYGRID command

```
#PLOTENERGYGRID
log                     TypeSpecGrid
0.1                     EminSpecGrid
100.0                   EmaxSpecGrid
25                      nSpecGrid
```

These are parameters for the flux spectrum particle output grid. It defines the type of grid (linear or log), the minimum energy, the maximum energy, and the number of grid points. Defaults are shown. Note that this is just the output grid and does not affect the calculation itself.

#NGDC_INDICES command

```
#NGDC_INDICES
ApFile.dat                      NameApIndexFile
f107.txt                        NameF107File
```

Read the Ap index and F10.7 from two files and calculate the appropriate array of AP indices to feed to MSIS.

#NOAAHPI_INDICES command

```
#NGDCHPI_INDICES
HpiFile.dat                     NameHpiFile
```

Read the HPI from a file. Used to get the Auroral ionization and heating

#HPI command

```
#HPI
0                       HemisphericPower
```

Set the HPI to a constant value. Used to get the Auroral ionization and heating

#AURORA command

```
#AURORA
F                       UseAurora
```

Set whether or not to include auroral ionization and bulk heating.

#SOLARWIND command

```
#SOLARWIND
0                       Bx
0                       By
0                       Bz
400                     Vx
```

This command sets the solar wind parameters for the Weimer model. The solar wind parameters are constant if this command is used

#MHD_INDICES command

```
#MHD_INDICES
IMF.dat                 NameUpstreamFile
```

Read the IMF from a file. Unlike the SOLARWIND command, the IMF can be time dependant in this case.

6.1.3 Control

#END command

```
#END
```

The #END command signals the end of the included file or the end of the PARAM.in file. Lines following the #END command are ignored. It is not required to use the #END command. The end of the included file or PARAM.in file is equivalent with an #END command in the last line.

#STARTTIME command

```
#STARTTIME
2006                    iYear
7                       iMonth
19                      iDay
0                       iHour
0                       iMinute
0                       iSecond
```

The STARTTIME command sets the integer year, month, day, hour, minute and second at which the simulation begins. This command is only used in standalone mode.

#STOP command

```
#STOP
10                      Tmax
-1                      MaxStep
```

The #STOP command sets the stopping condition for the PWOM during standalone execution. Tmax sets the stop time in timeaccurate mode. and MaxStep sets the maximum number of iterations for non-timeaccurate mode.

6.1.4 Physics

#FLUIDPWI command

```
#FLUIDPWI
T               UseFluidPWI
```

Use wave-particle interaction for the fluid version of PWOM. Default is false.

#FAC command

```
#FAC
F               UseJr
```

UseJr determines whether to use field aligned currents to affect the ion outflow. The default is shown.

#STATICATMOSPHERE command

```
#STATICATMOSPHERE
T                       UseStaticAtmosphere
```

To keep the MSIS atmosphere constant set UseStaticAtmosphere to True.

#MSISPARAM command

```
#MSISPARAM
60                      F107
60                      F107A
4                       AP1
4                       AP2
4                       AP3
4                       AP4
4                       AP5
4                       AP6
4                       AP7
```

This command sets the MSIS parameters for the the neutral atmosphere. The defaults are shown.

#SE command

```
#SE
T               DoCoupleSE (rest is read if true)
T               UseFeedbackFromSE
F               IsVerboseSE
120             DtGetSe
```

#SE commands DoCoupleSE and UseFeedbackFromSE tells if you should include SE coupling (works for Earth and Jupiter) and if feedback should be included. IsVerboseSE option tells if SE should write lots of output (good for debugging one line) or run quietly (good for lots of lines). DtGetSe tells how frequently SE should be called.

#SETPRECIP command

```
#SETPRECIP
T                       UseFixedPrecip (rest is read if true)
100.0                   PrecipEnergyMin[eV]
1000.0                  PreciptEnergyMax[eV]
400.0                   PrecipEnergyMean[eV]
2.0                     PrecipEnergyFlux[ergs/cm2/s]
```

Sets electron precipitation from a given distribution at the top of the simulated magnetic field line. In PWOM stand alone mode UseFixedPrecip define if we include precipitation, for coupled runs the precipitations is given by IE module. In the case of coupled runs you can force all field lines to have the same precipitations with UseFixedPrecip. The precipitation is defined as a Maxwell energy spectrum covering energies between PrecipEnergyMin and PreciptEnergyMax with mean energy of PrecipEnergyMean. The inflow precipitation flux at the top of the field line is given by PrecipEnergyFlux. See also #POLARRAIN.

Default is UseFixedPrecip false.

#POLARRAIN command

```
#POLARRAIN
T           UsePolarRain (rest is read if true)
80.0        PolarRainEMin[eV]
300.0       PolarRainEMax[eV]
100.0       PolarRainEMean[eV]
1.0e-2      PolarRainEFlux[ergs/cm2/s]
```

Sets a constant electron precipitation from a given distribution at the top of the simulated magnetic field line Applied to all field lines. The precipitation is defined as a Maxwellian energy spectrum covering energies between PolarRainEMin and PolarRainEMax with mean energy of MeanPolarRainEMean. The inflow precipitation flux at the top of the field line is given by PolarRainEFlux. See also #SETPRECIP.

Default is UsePolarRain false.

Index

CON
- #AURORA, 24
- #END, 25
- #FAC, 26
- #FIELDLINE, 23
- #FLUIDPWI, 26
- #HPI, 24
- #LIMITER, 21
- #MHD_INDICES, 25
- #MOTION, 22
- #MSISPARAM, 26
- #NGDC_INDICES, 24
- #NOAAHPI_INDICES, 24
- #PLOTENERGYGRID, 24
- #POLARRAIN, 27
- #REGRID, 22
- #RESTART, 23
- #ROTATION, 22
- #SAVEPLOT, 23
- #SAVEPLOTELECTRODYNAMICS, 24
- #SCHEME, 21
- #SE, 26
- #SETPRECIP, 27
- #SOLARWIND, 25
- #STARTTIME, 25
- #STATICATMOSPHERE, 26
- #STOP, 25
- #TEST, 23
- #TIMEACCURATE, 22
- #TIMESTEP, 22
- #TYPEPLOT, 23
- #VARIABLEDT, 22
- #VERTICALGRID, 22