Low Earth Orbit Plasma Environment Visualization

Nick Bradley
May 4, 2010

I. Overview of Goals

The goal for the independent study effort this year was to develop a functional interface to an available low earth orbit (LEO) plasma environment modeling database, and to be able to easily extract pertinent information to charged spacecraft formation flying (plasma density, Debye length, etc.) The Debye length varies according to fluctuations in ion and electron temperatures and densities. There are some simple web-based interfaces and FORTRAN interfaces to some functional pieces of software that can provide some subset of the required data, but no easily accessible and user-friendly solution exists for quick numerical calculations and "sanity checks."

The specific goals for the fall semester were to:

- Research available models for the plasma environment in LEO and GEO
- Design, build, and complete a Matlab Graphical User Interface (GUI) to access data in the usable databases
- Model a reference craft and produce data for the craft given certain environmental conditions

The specific goals for the spring semester, given some setbacks and refined goals leading from the fall semester, were to:

- Continue researching available models and sources for plasma environment modeling
- Investigate anomalies in varying instances of the Debye length equation
- Produce Debye length functionality in Matlab

II. Summary of Fall Semester

The work completed during the fall semester was instrumental in developing a basic graphical user interface (GUI) to display an input of ionospheric data. It was found shortly into the spring semester that the data obtained from the MSIS source was not, as previously thought, both pertaining to neutral and ionic substances in LEO. No ionic or negative electron data is contained in the database, which is unfortunate given its ease of implementation as a Matlab aerospace toolbox function. However, the fall semester was not a complete waste, as the basic structure of Matlab display code was developed, as was much of the underlying physics and necessary basic knowledge.

III. Summary of Spring Semester

During the spring semester, a large amount of work was completed in researching other available atmospheric modeling libraries, interfacing them into Matlab, and generating useful output data for the end user. In the end, the International Reference Ionosphere

(IRI) model was chosen because of its reliable and broad database of ionospheric environmental information, its availability as source code, and its relevant data inputs and outputs. Through many different methods and trials, the IRI FORTRAN source code was interfaced with Matlab. A standalone Matlab function now exists to appropriately call the IRI FORTRAN file, and a GUI exists to interface with the Matlab library.

IV. Detailed Deliverable Description

The delivered code package has several levels of hierarchy, depending on the level to which the user wishes to control the model output. Figure 1 shows the hierarchy from highest level to lowest level. The GUI allows the user to choose from a list of possible data type and plot outputs. The parameters generated in the GUI are sent to the atmospheric modeling code, which can be run separately from the Matlab command line without control of the GUI. The modeling code takes in a varying set of parameters based on the desired output, and returns atmospheric environmental data based on the specified input as well. At a lower level is the Matlab-IRI interface, which simply takes in a set of values and calls the IRI FORTRAN code with the Matlab UNIX interface command. The interface code writes a text file with the parameters in the correct format for the IRI code to read in. The IRI code returns its own text file with the desired output parameters, which the Matlab-IRI interface code reads and returns as its own output. This means that the interface code can be run with its own inputs and outputs entirely within Matlab, even though it interfaces with the operating system and external FORTRAN code.

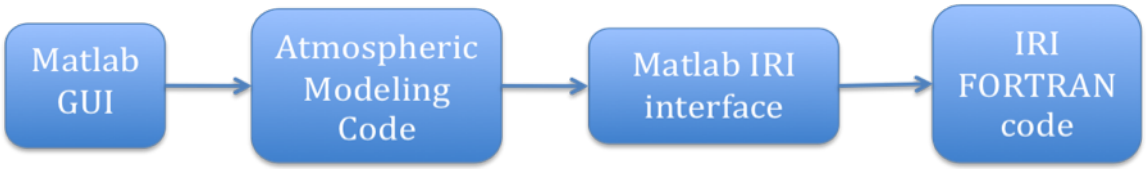


Figure 1: Code hierarchy

Each function contains its own extensive help text (inserted as Matlab comments). The text included here is to provide familiarity and examples with the code. The descriptions will begin with the lowest-level functionality and successively describe how specifically each higher level function interfaces with the lower-level functionality.

1. IRI FORTRAN CODE

The IRI FORTRAN source code is available online through an FTP site[1] maintained by NASA's Goddard Space Flight Center. This package contains all the code, libraries, and external data files necessary to enter a wide variety of input parameters to obtain a similarly broad variety of outputs. Because of the large amount of functionality of the IRI database, the source code was modified into a new library to suit the needs of spacecraft formation flying environmental modeling. Specifically, the source code was also modified to interface with text files so that Matlab could easily use this medium to

[1] http://iri.gsfc.nasa.gov/

communicate with the code library. An extensive amount of time was spent trying to interface to the FORTRAN code using .mex files in MATLAB, but these files proved extremely cumbersome and poorly documented. Instead of writing an entirely new function in MATLAB or developing enough FORTRAN to properly model the required parameters in the native modeling language, the text file interface was chosen instead.

The text-related code modifications were made simply by using the READ and WRITE commands in FORTRAN. The code downloaded from the FTP site includes a "test" function, which interfaces with the command line so that the user can input specific parameters that the code will then use to produce an output file of parameters. This code was simply modified for our desired purposes. Instead of having the capability to read in and produce a wide array of parameters, most of which are not useful for our purposes, the code was altered to work with a specific set of variables. Compiling the FORTRAN code (`calliri2.for`) requires the following command line text (using "gfortran" on a Mac).

```
gfortran  -o calliriF calliri2.for irisub.for irifun.for
iritec.for iridreg.for igrf.for cira.for
```

This creates an executable file called "calliriF" that can be executed from the command line. This executable, when run, looks for a file called "parameters.txt" with the input parameters for the IRI routine. This text file MUST be set up with the following format, including line breaks and commas. The variables that should NOT be changed are in square brackets.

```
jm,xlat,xlon
iy,imd,iut,hour
hx
ivar
vbeg,vend,vstp
htec_max
[jf1],[jf2],[jf3],[jf4],[jf5]
[jf6],[jf7],[jf8],[jf9],[jf10]
[jf11],[jf12],[jf13],[jf14],[jf15]
[jf16],[jf17],[jf18],[jf19],[jf20]
[jf21],[jf22],[jf23],[jf24],[jf25]
[jf26],[jf27],[jf28],[jf29],[jf30]
```

The first six lines are explained in Table 1. The "jf" terms are Boolean T/F flags that should not be altered. Their explanation is in the IRI source code "readme" file.

Table 1: FORTRAN code input parameter explanation

Variable	Value(s)	Notes
jm	0 (geographic coordinates) 1 (geomagnetic coordinates)	0 should usually be used
xlat	Latitude in deg. (+N, -S)	
xlon	Longitude in deg. (+E, -W)	
iy	Year	
imd	Day in MMDD format or (-ddd) for day of year	
iut	0 (local time) 1 (universal time)	1 should usually be used
hour	Hour of day	
hx	Height in km.	
ivar	Desired variable: 1 (altitude) 2 (latitude) 3 (longitude) 4 (year) 5 (month) 6 (day) 7 (day of year) 8 (hour)	This value changes based on which variable is desired. The Matlab interface code assigns this value based on the user input
vbeg	Variable initial value	
vend	Variable final value	
vstp	Variable step size	
htec_max	TEC integration height (?)	This value is not understood or documented… the value that the Matlab interface code uses for this variable is simply the height (hx)

The "calliriF" executable takes in the data from "parameters.txt" and produces a file called "output.txt." This file contains all of the information pertinent to our purposes, which is electron density and temperature, ion temperature, and specific ion species densities. Because of an anomaly in the IRI FORTRAN source code, the executable cannot directly output the ion species' densities in number density format, but only in relative percentage format. It does have a documented capability to produce number density, but it does not function correctly. However, this procedure has been implemented in the Matlab code that calls the executable function, and is modeled after the mathematics in the FORTRAN source code to achieve the same result. The IRI code assumes that ion and electron total number density will be equal, which is valid for singly-charged ion species in a plasma at equilibrium (i.e. no perturbing charges present at infinity). This is a valid assumption for our purposes, and so the total electron number

density is combined with the ion species' relative densities to produce the ion species' number densities. Each line in the output file is formatted as follows.

```
VAR, rhoE, Ti, Te, O+, N+, H+, He+, O2+, NO+
```

Table 2 contains an explanation of these output variables.

Table 2: FORTRAN code output parameter explanation

Variable	Value(s)	Notes
VAR	Variable step value	Represents the value of the step, e.g. "45" for the parameters at 45 degrees longitude if longitude is selected as the independent variable
rhoE	Electron density ($1/cm^3$)	Assumed to be equal to the total ion number density
Ti	Ion temperature (K)	
Te	Electron temperature (K)	
O+	O^+ relative density (%)	
N+	N^+ relative density (%)	
H+	H^+ relative density (%)	
He+	He^+ relative density (%)	
O2+	O_2^+ relative density (%)	
NO+	NO^+ relative density (%)	

2. MATLAB IRI INTERFACE

This interface simply provides a function that reads and writes the text files necessary for the IRI executable, described in the previous section. As inputs, this function takes in the latitude, longitude, year, month, day, universal time, height, variable initial value, variable final value, variable step size, and type of variable. These inputs are formatted appropriately for the text file "parameters.txt," as described in the previous section. The function uses the Matlab command to interface to the underlying OS to run the "calliriF" executable (`!./calliriF`). The IRI code produces the "output.txt" file, which this interface function then reads in to obtain the desired data. As mentioned previously, the ion species' relative densities are converted into absolute number densities by assuming that the total electron number density is equivalent to the total ion number density for singly-charged ion species, which is valid for the LEO region covered by the IRI code. The function returns a vector of the desired variable values (latitude, longitude, etc.) and a vector for each of the corresponding density and temperature parameters described in Table 2.

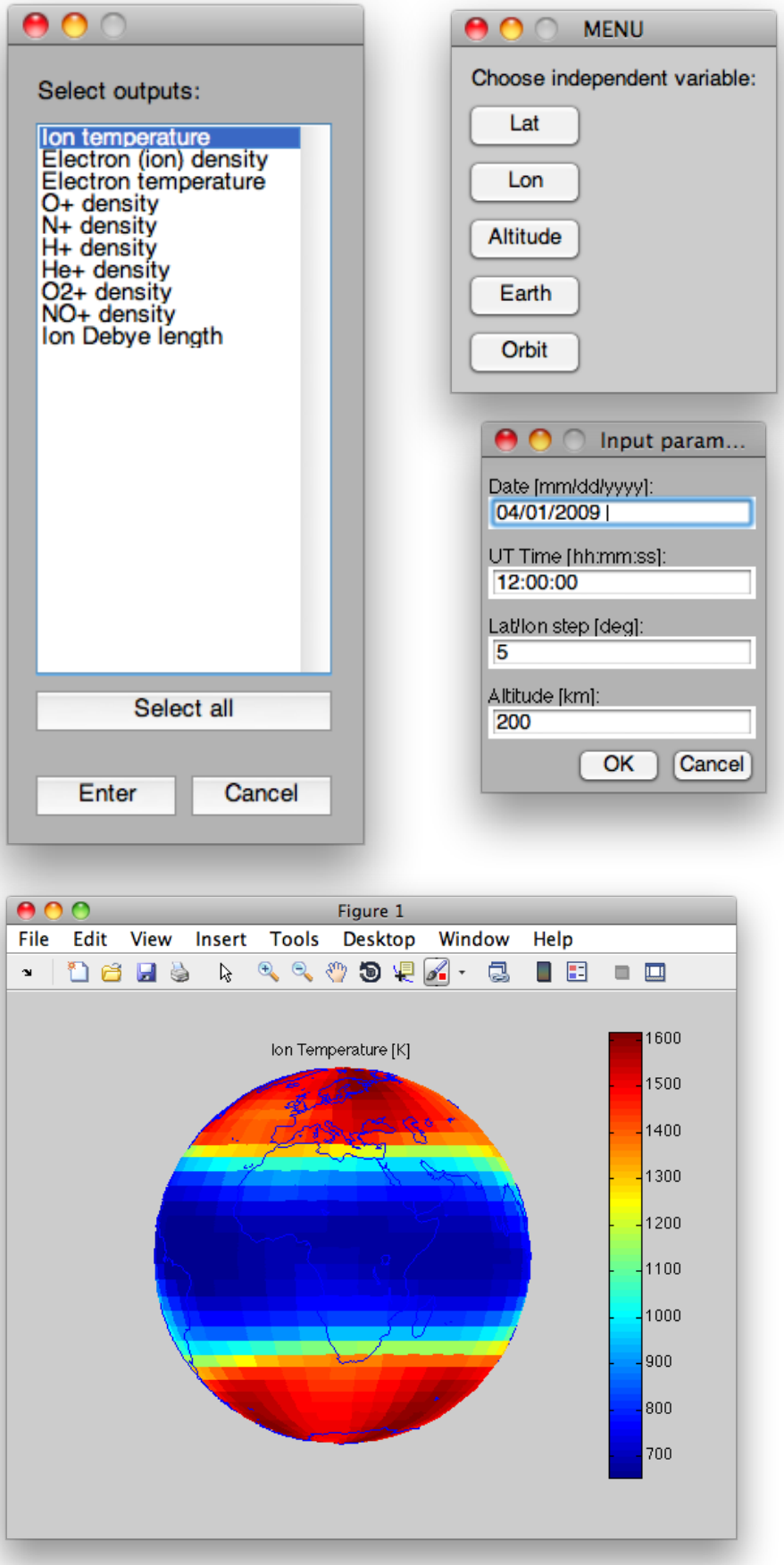
3. ATMOSPHERIC MODELING CODE

This function produces plots and sets of Matlab outputs for a desired large set of data, such as the entire earth or visualizing a set of multiple ion densities. In its simplest definition, it is essentially a "FOR" loop that iterates over a given set of data to call the IRI interface function, which successively produces sets of text files that call the IRI executable. The function is very versatile in its input and output parameters, since the required parameters and variables are different for different types of visualizations. The visualizations available are latitude varying 2D, longitude varying 2D, altitude varying 2D, global 3D, and orbit varying 2D. With a user-specified set of input and output parameters, including a desired independent variable, the function takes the data and passes it to the IRI executable interface, which returns the results from the model. These results are formatted, plotted, and written to a structure that is returned as a function output.

This function can be separately embedded or called from any other Matlab function, as long as the appropriate input and output format is followed. For example, if a user wishes to observe the variation in Debye length over the South Atlantic Anomaly by altitude, a simple "FOR" loop can be written to loop through specific altitudes at the desired latitude/longitude for the South Atlantic Anomaly. This atmospheric modeling code can be called at every desired altitude (with the plot option turned either on or off) to obtain a set of data that can be evaluated by the user.

4. MATLAB GUI

This GUI is simply a user interface to the atmospheric modeling code. The interface provides a series of dialog boxes from which the user can choose the desired modeling variable, the desired outputs, and specify the values of the parameters for the modeling simulation. An example of the GUI process with its parameter inputs is shown in the following figures. The atmospheric modeling code will produce plots consistent with the selected outputs in the first GUI window according to the independent variable chosen in the second window. The parameters are specified in the third window, and are customized for the independent variable type.


Select outputs:
Ion temperature
Electron (ion) density
Electron temperature
O+ density
N+ density
H+ density
He+ density
O2+ density
NO+ density
Ion Debye length
Select all
Enter
Cancel
MENU
Choose independent variable:
Lat
Lon
Altitude
Earth
Orbit
Input param...
Date [mm/dd/yyyy]:
04/01/2009
UT Time [hh:mm:ss]:
12:00:00
Lat/lon step [deg]:
5
Altitude [km]:
200
OK
Cancel
Figure 1
File Edit View Insert Tools Desktop Window Help
Ion Temperature [K]
1600
1500
1400
1300
1200
1100
1000
900
800
700

V. Extensions and Known Issues

Many extensions are possible with this library of code. Data smoothing, interpolation, and the addition of GEO environmental data are all extensions that would greatly benefit the user.

There are several known issues, mostly stemming directly from unexplained anomalies in the IRI FORTRAN source code.

1. When the whole earth is modeled at a specific time, it is apparent that there is a data anomaly at longitudes approximately ±20 degrees from the International Date Line (180 degrees longitude). Data smoothing has been investigated for this anomaly, but is not a fix to the problem of not being able to model actual data.
2. Along with the previous anomaly, the IRI FORTRAN code does not accept a longitude of 0 or 360 degrees. In the implementation of the Matlab code, longitudes of 0.01 and 359.99 degrees are used, which seem to work mathematically with the FORTRAN code.
3. Previously mentioned, the FORTRAN code seems to have trouble creating density outputs in absolute units rather than relative units. The work-around for this has been implemented in Matlab by using the electron density as the absolute reference.
4. The IRI code does not have solar activity data (f10.7cm, ap index) for dates past early 2009. The modeling is not completely valid for more recent dates, since the environment in the ionosphere is highly dependent on solar activity. Most likely, if a new version of the IRI source code is downloaded every so often from the FTP site, the data files can be replaced to provide updated solar data, but this has not been tested. Dates before early 2009 work well, and if simple modeling and rough estimates are desired for environmental behavior, older dates work well.