MIST
Ionosphere Simulation

Erika Hornecker
McGill University
erika.hornecker@mail.mcgill.ca

June 24, 2021

1 Description

This ionosphere simulation code provides the user with refraction angles and attenuation factors for input azimuth and elevation points. To achieve this, ionospheric parameters specific to each point are collected using IRI2016.

The coordinates of the points are to be collected from a txt file, with two 1D arrays corresponding to the elevations and azimuths.

Currently, only the refraction due to the ionosphere's f-layer is calculated. The function for implementing tropospheric refraction is included in the code as `trop(theta)`. The constants used in this function still need to be verified, and the `theta` input angle is the zenith angle (NOT elevation).

Note: the f-layer refraction angles are only trustworthy for elevation angles $>10°$(zenith angles $<80°$). As you approach the horizon, the ray tracing method breaks down. This phenomenon is called the low-elevation cut-off and is described in Figure 8 of Vedantham et al. (2018).

2 How to Run

Before running the code a couple packages need to be installed:

- `iri2016` (information on how to install IRI can be found in Memo 46)
- `pymap3D` (information on how to install can be found here)

Now that the packages are installed, you are ready to run the code!

The main function that needs to be run is called `run_all (filename, lat0, lon0, h0, date_time, freq, f_lower, f_upper, num_f_layers, num_d_layers, col_freq, save_y_n, save_name)`.

It takes the following inputs (in order), and saves them to an hdF5 file:

- The filename containing the input coordinates
- Latitude [°], longitude [°], and altitude [meters] of instrument
- The date and time [UTC]
- The frequency [Hz], a single value
- The lower and upper limits of the f-layer [meters] (Recommended: 150km-500km)

- The number of layers in the f-layer (Recommended: $\sim$ 30-50)
- The number of layers in the d-layer (Recommended: $\sim$ 10-15)
- The collision frequency (0, 1, 2, 3, or a value):
  - 0 would use a default fixed value of 10MHz
  - 1 would use a height dependent model of collision frequency from Nicolet (1953)
  - 2 would use a height dependent model of collision frequency from Setty (1972)
  - 3 would use an average of both models
  - entering a fixed value would simply use that value
- 'y' or 'n' for saving the file
- The beginning of the name of the file to save to (it then appends information on the location and date/time of the measurement to the name as well)

It outputs the following information, and saves it to the same hdf5 file:

- D-layer attenuation (`attenuation`): a 1D array with each entry corresponding to an input coordinate
- D-layer electron densities (`d_e_density`): a 2D array with each row corresponding to an input coordinate and each column corresponding to a layer (in depth)
- F-layer electron densities (`f_e_density`): a 2D array with each row corresponding to an input coordinate and each column corresponding to a layer (in depth).
- Angle of the outgoing refracted beam at each layer (`phis`), the angle is measured from the normal to the layer at that point: A 2D array, same concept as previous
- The net deviation of the elevation angle for each coordinate (`delta_theta`): a 1D array, a diagram of what this value represents can be seen in Figure 4 of Vedantham et al. (2013)
- Refractive index at each layer: A 2D array, same concept as the previous 2D arrays

3 Example

Here is an example showing how to run this code for an instrument at the MARS station.

First, if the code is being run to make a map, meaning you want equally spaced azimuth and elevation points, then there is a function to create those points. Run `create_pts(num_points)` with the number of elevation points you would like between 0 and 90 (or between 0 and 360, for azimuth). This will create a txt file `'az_el_pts.txt'` with the points.

Now, to run the simulation on those points:

```
attenuation, d_e_density, f_e_density, phis, delta_theta, ref_indices = run_all( 'az_el_pts.txt',
79.433, -90.766, 0, '2012-08-15 06:00', 45e6, 150000, 500000, 30, 10, 1, 'y', 'Example_')
```

A separate function to make polar plots was also included: `polar_plot(el, az, data, n_rows, title, label)`. The value of n_rows is our previously defined `num_points`.
Note: this function is currently set up to be used on a 1D array dataset (but there is a commented section for averaging a 2D array into a 1D array, to plot the electron density for example), and it is set up to be used with coordinates obtained from the `create_pts` function (to make a map).

To study the f-layer electron density for a direct beam path (i.e. without taking into account refractions at each interface), a separate function was included: `get_direct_f_e_dens (filename, lat0, lon0, h0, date_time, f_lower, f_upper, num_f_layers)`