HamSCI: Eclipse Simulator Documentation

Joshua S. Vega, WB2JSV

Last Updated: November 5, 2018

Contents

1 Introduction

HamSCI's eclipse simulator (colloquially known as "`eclipsesim`") is a software package developed by members of the HamSCI team at the New Jersey Institute of Technology used to simulate HF (3-30 MHz) through the ionosphere. Specifically, the package was developed in order to simulate the effects of the August 21, 2017 total solar eclipse on HF amateur "ham" radio communications. The bulk of the package is developed in MATLAB. However, the inputs and outputs are both in the comma-separated values (CSV) format in order to allow for easy interoperability with other analysis tools (such as those written in Python, R, or Julia).

This document is written in order to provide the reader with an understand of the inner workings of the simulation package as well as justify some of the design decisions that were made during development. In addition, this document is intended to be used as a helpful guide to using the package. Because of this, as the package undergoes future changes, so too should this document in order to maintain parity between the package and its documentation.

2 Dependencies

The `eclipsesim` package relies on a few third-party dependencies in order to implement some specialized functionality. This section provides a brief overview of the different dependencies and their function.

2.1 PHaRLAP

PHaRLAP[1]is a robust HF raytracing toolkit developed by Dr. Manuel Cervera of the Defence Science and Technology Group, Australia. It is used to compute the actual ray paths of radio transmission from transmitter to receiver. More information on how it is used is provided in section 4. Although the core of the toolkit is written in FORTRAN, a MATLAB application programming interface (API) is provided. PHaRLAP also provides a simple API for the IRI (see below) that is compatible with the ray path computation API.

2.2 SAMI3

SAMI3 is a three-dimensional global ionosphere model developed by the U.S. Naval Research Laboratory (NRL). In 2017, Joseph D. Huba and Douglas Drob published a paper detailing a modified SAMI3 model that simulated effects of the August 21, 2017 eclipse on the ionosphere [Huba and Drob, 2017]. This model was provided to HamSCI by Joseph Huba and have since been the *de facto* ionosphere model for eclipse simulations. For more information on SAMI3's data structure and how it has been integrated into the eclipse simulation, see Appendix B.

[1] *"The results published in this paper were obtained using the HF propagation toolbox, PHaRLAP, created by Dr. Manuel Cervera, Defence Science and Technology Group, Australia (manuel.cervera@dsto.defence.gov.au). This toolbox is available by request from its author."*

3 Usage

The `eclipsesim` package provides two methods of execution out-of-the-box: serially and in parallel. More information and specific instructions on each execution method is discussed in subsection 3.2 and subsection 3.3, respectively.

Regardless of the execution method used, `eclipsesim` is designed to be executed on a Linux-based machine. While limited success has been achieved on other operating systems (notably, Microsoft Windows), the execution scripts are developed with Linux in mind and may not execute as expected in incompatible environments. All instructions provided in this section are therefore intended for Linux-based systems and may require modification for other operating systems.

3.1 Setup

Before the eclipse simulation can be run, all external dependencies must be accessible to the simulation program. This section briefly describes what dependencies are required and how they must be configured. Some extension version of the simulation program may require additional dependencies. Please refer to Appendix A for extension-specific dependency information.

MATLAB: Since all computation is executed in MATLAB, it should come as no surprise that the most important dependency is MATLAB itself. The simulation is primarily designed to execute on version 2016a however later versions should also be compatible. When using older versions of MATLAB some compilation and runtime errors have been seen to occur due to breaking changes between older version and 2016a or later. Once MATLAB has been installed, it's important to ensure that the `matlab` program is available for execution through a modified `$PATH` variable.

PHaRLAP: As described in subsection 2.1, PHaRLAP is a vital part of the simulation program. Due to the restrictions on the distribution of PHaRLAP, it is not and should not be directly distributed with any of the eclipse simulation versions. Instead, it is recommended that prospective users of the simulation tool contant the author of PHaRLAP and receive a direct copy. Once a copy of PHaRLAP is acquired, it can be installed anywhere that is accessible to the simulation tool as long as the `PHARLAP_DIR` variable is updated appropriately in the execution script.

SAMI3: If SAMI3 will be used as the ionosphere model, then there are a few steps required in order to prepare it to work with the simulation tool. After acquiring the SAMI3 model it must be converted from the raw data format to the `*.m` file format which is far more efficient for MATLAB to load and parse. For detailed information about this step, refer to Appendix B. Once it has been properly converted, the SAMI3 `*.m` file(s) must be placed in a directory that is accessible to the simulation program. Be sure to update the `SAMI3_PATH` variable appropriately in the execution script.

3.2 Serial

The serial execution model is designed for testing or very short simulations. The serial execution model only executes the simulation for a single point in time. This can obviously be adjusted by modifying the bash execution script. However for this sample, it is assumed that only one time-step is being simulated. It is also assumed that all dependencies have been properly installed and configured and that the appropriate modifications have been made to the execution script.

To execute the simulation tool in serial mode, you'll be executing the `eclipsesim_local.sh` file in as bash script without any arguments (ensure that it has been configured to have the execution permissions bit enabled):

```
$ ./eclipsesim_local.sh
```

This script will the first job file available in the input directory (see subsection 3.4 for information on inputting data to the simulation tool). When the simulation is complete the terminal will be prepared for a new command and the outputted data will be available in the output directory (see subsection 3.5 for information on outputting data from the simulation tool). If an error occured during the simulation, MATLAB will halt and indicate where in the MATLAB code the error occurred.

3.3 Parallel

The parallel execution model is designed to be executed on a computing cluster running a Sun Grid Engine (SGE; now Oracle Grid Engine) compatible scheduling system. One such system is Son of Grid Engine (also SGE). When submitting a job, SGE adds it to the job queue. When a compute node with the requested resources is made available, the next job is assigned to be executed on said node. SGE also allows multiple compute nodes to be requested by one job using the array option. For more information about the many options made available to jobs, please refer to both the SGE documentation as well as any cluster-specific documentation.

To execute the simulation tool in parallel mode, execute the `eclipsesim_sge.sh` file as an argument to the `qsub` application.

```
$ qsub ./eclipsesim_sge.sh
```

This command will submit the job file to the SGE queue. Once the job has been selected for execution, it will ensure that MATLAB is loaded and then begin the process of preparing the simulation for execution (this step is identical to the equivalent step in the serial execution model). Due to the parallel nature of this execution model, no response will be printed to the terminal. Instead, a log file for each job submitted job (if the job is an array job, one log file will be generated for each sub-job) containing all output information for that individual job (or sub-job). For this reason it is not recommended that the simulation be tested or debugged in the parallel execution mode (unless it is directly related to the parallel nature of the execution).

3.4 Input

In order for the simulation tool to execute, it requires some input data. This data is usually in the form of a transmitting station location, receiving station location, frequency, and time of exchange. Because the simultion is divided into timestamps, the time of exchange is encoded into the input filename. This is done using the format ‘‘`<YYYY>-<MM>-<DD> <HH>:<mm>:<ss>.csv`’’ (e.g. ‘‘`2017-08-21 16:00:00.csv`’’ for August 21, 2017 at 4:00 PM UTC).

Within the input file, the remainder of the input data is stored. This is done by entering the data as rows into the file using the Comma-Separated Values (CSV) format. The first line of file should be ‘‘`RX_CALL,TX_CALL,MHZ,RX_LAT,RX_LON,TX_LAT,TX_LON`’’. Table 1 shows the different input fields (all fields are required).

Name	Datatype	Description	Example
RX_CALL	alphanumeric string	The name (or callsign) of the RX station.	WE9V
TX_CALL		The name (or callsign) of the TX station.	N8PW
MHZ	floating-point number	The frequency of the transmission in MHz.	7.03
RX_LAT		The latitude of the RX station.	42.5625
RX_LON		The longitude of the RX station.	-88.0417
TX_LAT		The latitude of the TX station.	40.8542
TX_LON		The longitude of the TX station.	-81.3750

Table 1: The input data fields for the simulation.

Below is an example row from an input file:

```
WE9V,N8PW,7.03,42.5625,-88.0417,40.8542,-81.3750
```

Once the correct input files have been created (or possibly generated), they should be placed into the `./jobs` sub-directory of the simulation tool's directory.

3.5 Output

PHaRLAP outputs a wide array of simulation values for each simulated ray. However, for the purposes of this simulation tool, only the values related to the "best" ray-path between the transmitting station and receiving station are taken into account. Yet still there are many outputted values that are recorded into the output CSV files. Table 2 details them.

These output files will be located in the directory specified by the `OUT_PATH` variable in the execution directory (ensure that this directory exists prior to execution or else the simulation will crash). The filename indicates the time at which the simulation was executed using the form ''`simulated_<YYYY>-<MM>-<DD>_<HH>-<mm>.csv`''(e.g. ''`simulated_2017-08-21_16-00.csv`'' for August 21, 2017 at 4:00 PM UTC).

It's important to remember that each row is not a unique ray-trace. Instead, a row only provides the information relevant to a single hop of a ray-trace. Therefore, if for example a ray required 3 hops to travel the distance between the transmitter and receiver then 3 rows will appear in the output file. In this case, the `srch_rd_hop_idx` field will indicate the order in which the hops occurred (e.g. 1 being the first hop and 3 being the last). In this case, it's also important to realize that the field `srch_rd_nhops_attempted` is not necessarily the number of resulting number of rows that will appear in the output file. If the maximum number of hops is set at 10 for example but if only 3 hops are needed then only 3 rows will appear for that ray-trace.

4 Design

This section discusses the inner-workings of the eclipse simulator package. However, it does not discuss how PHaRLAP or any other dependency function and instead focuses on how such functionality is being used. Please refer to the respective documentation for design details about any dependencies.

tx_call	The name (or callsign) of the TX station.
rx_call	The name (or callsign) o fthe RX station.
tx_lat	The latitude of the TX station.
tx_lon	The longitude of the TX station.
rx_lat	The latitude of the RX station.
rx_lon	The longitude of the RX station.
freq	The frequency of the transmission in MHz.
srch_rd_lat	The latitude of the end point of the ray.
srch_rd_lon	The longitude of the end point of the ray.
srch_rd_ground_range	The ground range of the ray in km.
srch_rd_group_range	The group range of the ray in km.
srch_rd_phase_path	The phase path of the ray in km
srch_rd_geometric_path_length	The physical distance along ray path in km.
srch_rd_initial_elev	The initial elevation of this hop in degrees.
srch_rd_final_elev	The final elevation of this hop in degrees.
srch_rd_apogee	The maximum altitude of the ray in km.
srch_rd_gnd_rng_to_apogee	The ground range to the apogee in km.
srch_rd_plasma_freq_at_apogee	The plasma frequency at the ray apogee MHz.
srch_rd_virtual_height	The virtual height of the ray in km.
srch_rd_effective_range	The effective range of the ray in m.
srch_rd_deviative_absorption	The ionospheric deviative absorption in dB.
srch_rd_TEC_path	The integrated electron density along the ray path (number of electrons in $1\,m^2$ cross-section tube).
srch_rd_Doppler_shift	The Doppler shift in Hz.
srch_rd_Doppler_spread	The Doppler spread in Hz.
srch_rd_FAI_backscatter_loss	The backscatter loss in dB for the last hop due to field aligned irregularities.
srch_rd_frequency	The carrier frequency used for the ray.
srch_rd_nhops_attempted	The number of hops in this ray.
srch_rd_rx_power_0_dB	*These fields should be ignored.*
srch_rd_rx_power_dB	
srch_rd_rx_power_O_dB	
srch_rd_rx_power_X_dB	
srch_rd_hop_idx	The index for this hop.
srch_rd_apogee_lat	The latitude of the apogee.
srch_rd_apogee_lon	The longitude of the apogee.

Table 2: The output data fields for the simulation.

4.1 Rough Pass

The most basic functionality of the eclipse simulator package is the initial raytrace calculation. Here, PHaRLAP is used to raytrace several ray paths at different takeoff angles (the initial angle above the horizon from the transmitting station). For a usual simulation, the rays are lauched at 0.5° increments from 5° to 60°. Each ray is simulated for 3 hops. From this set of rays, it is possible to determine at which takeoff angle and hop count the ray landed closest to the receiver. Two rays are selected: the closest ray that landed before the receiver, and the closest ray that landed beyond the receiver. These two rays are the boundaries for the "Fine Pass" described in subsection 4.2.

4.2 Fine Pass

The fine pass works very similarly to the rough pass. Asside from the boundaries set by the rough pass, the fine pass is also distinct in that it iterates using an increment that is a fifth of the difference between that of the two bracketing rays (e.g. if the selected rays have takeoff angles of 10° and 10.5°, then the new inrecment would be 0.1°. It is assumed that the two selected takeoff angles are within twice the rough increment step from one another (when the rough increment is 0.5°, the difference in takeoff angles cannot be more than 1°).

After the fine pass has been raytraced, linear interpolation is used to finally find the "best" take-off angle for a transmission between a transmitter and receiver. The raytracer is executed with this "best" takeoff angle and the outputted values are saved to the output CSV file in the format described in subsection 3.5.

4.3 Plotting

5 Conclusion

References

J. D. Huba and D. Drob. SAMI3 prediction of the impact of the 21 August 2017 total solar eclipse on the ionosphere/plasmasphere system. *Geophysical Research Letters*, 44:5928–5935, 2017.

A Appendix: SAMI3 Data Format