ENG 4350 Lab Report

Algonquin Radio Observatory

Diego Mateos Nikeet Pandit

Department of Earth and Space Science and Engineering York University
January 22, 2020

Contents

Purpose

The purpose of this virtual field trip is to field test the developed satellite tracking code, as well as gain experience with the use and operation of ground equipment such as a large parabolic radio antenna. This involves the usage of receiving equipment such as spectrum analyzer which is used to characterize the received signal and account for antenna gain. Furthermore, through a series of experiments we will experimentally attempt to determine the antenna dish pattern. Additionally, the purpose of this report is also to present all software validation methods, including the simulated software testing done with the Professor. In the total development of the ARO tracking section code, experience in creating a software package in Python was developed. In addition to the programming aspect, the tangible necessity of coordinate systems and coordinate transformations were shown on full scale project. Finally, there was tangible experience in using the SPG3 equation model for propagation of a GPS satellite.

Final Code Functionality

There are 6 modules and the main script (all .py files appended in zip file), three of which are Date fun, propagate, and FileIO, which have been discussed previously and there have been no updates to them and therefore will not be discussed. The three remaining high level modules and their respective functions and updates will be presented, and how their integration in the high level script yields the desired results.

List of modules/files:

1. Fileio
2. Datefun
3. Propagate
4. *Parse*
5. *HighLevel*
6. *Debugging*

Parse

The input parser simply contains four functions for the purpose of importing the source text files which obtains station, tracking interval, link inputs and TLE data.

High level

This module imports all the low level subroutines and serves as the main calculator for the propagation of the 32 satellites for a 30 minute time interval. Primarily, it calls on the input parser to obtain the required source code, and runs said data through a number of functions. These functions are listed below. Asides from Satlist, all functions have time as input unless otherwise specified. The reason for this will be discussed in the debugging module.

1. *Satlist*
2. *LookAngles*
3. *ECI*
4. *ECF*
5. *TOPO*
6. *DebugElements*
7. *SatName()*
8. *SatEpoch()*
9. *FileNames(Coord,N, StepSize)*

Satlist – This function calls on Fileio within a for loop with index N (defined as the number of the satellites in the source code tle file) to run the satellite class and create a list of 32 instances of this class for each satellite. In addition, outside this function on the main script, the parse module is called to create instances of the tle and station files, and obtain the required attributes of the station.

LookAngles – This function is a generator. It calls on all propagate functions and creates instances of all required vectors for calculating the look angles via the propagate topocentric to look angles function. This function runs a for loop of index N, yields azimuth elevation angles and rates, and is only defined within this module, it is not run as it will be used elsewhere for computation.

ECI, ECF, TOPO - These functions are all generators. They are only defined in the high level module. These functions; similar to LookAngles, call on the propagate functions, but only as required to yield results for their respective coordinate system state vectors. Other relevant

information is yielded as required, such as orbital elements or range and range rates for the Topocentric function. These yielded results can easily be changed by the user for debugging purposes.

DebugElements – This function is a generator. It yields results for propagate function outputs that the user would need for debugging. It outputs obital elements for checking them on STK.

SatName – Generator for yielding all satellite names. Calls on Satellite class attribute name.

SatEpoch – Generator for yielding all satellite epochs and UTC offset.

FileName – Generator to create output file names. Calls on SatName generator.

Debugging

This module obtains the functions to yield all output files required for the project purpose. The main script simply calls on debugging or high level modules to establish the program flow. The functions and their functionalities are as follows:

STKout (Coord,StepSize,N)
STKoutIndx(Coord,StepSize,N,Index)
STKsp (StepSize,N)
STKspIndx (StepSize,Index,N)
AOSLOS (StepSize,N)
PrintSatName()
TrackData (StepSize,Index)

STKout (Coord, StepSize,N) – Function to write an output ephemeris file for STK testing. This function is called in main script. It also takes as input which coordinate system is to be used for the ephemeris file. It can produce ephemeris files for ECI, ECF, or Topocentric coordinate systems, and orbital elements. All ephemeris files can be brought into STK for confirmation. Step Size input defines the length of time in seconds at which a calculation will be done, within a time interval determined in the Parse function. N is the number of satellites in the TLE. All inputs are defined in the main script. All functions in debugging, unless otherwise specified, run a for loop of size N to call on the high-level functions and calculate; for each satellite, the required propagation parameters at each time interval defined by Step Size. Meaning the high-level functions are called upon with each loop iteration, and that is why they have only time as input. **This is applicable to all functions in this module.**

STKoutIndx(Coord,StepSize,N,Index) – This function has the same functionality as STKout, however the index input is added to specify an ephemeris output for only one satellite, which is the chosen satellite from the list input by the user in the main script.

STKsp (StepSize,N) – Similar functionality and methodology to STKout, however an output pointing (.sp) file is generated for testing in STK. This pointing file has specifications in the

Topocentric coordinate system as it is what is relevant for the ARO to satellite link in this project.

STKspIndx (StepSize,Index,N) – Similar functionality to STKoutindx, with a (.sp) pointing file generated for a chosen satellite.

AOSLOS (StepSize,N) – Generates an access AOS/LOS table on the screen output for all satellites in a given tracking interval. This is an update from earlier versions of the software where it had initially outputted this information to an output text file. This function calls on high level functions such as SatName and look angles, and also on the parse module. The previous version of AOSLOS can actually determine satellites in line of sight for any interval. Specifically, it can determine when satellites enter access, leave access, and enter again for any X number of iterations. This functionality was not ported to the new AOSLOS that used for the ARO field trip, as formatting the output in the previous (more robust) functionality was found to be quite challenging, and unnecessary for this ARO visit.

PrintSatName() – This function prints out the satellite names. It runs a for loop for all satellites and calls on high level function SatName to print out the names from the SatList. This is used for debugging, as information about indexing will be given, and specific instances of any satellite in a TLE can be called.

TrackData (StepSize,Index) – Calls high level functions TOPO and Look Angles and SatName to output the tracking data file for a selected satellite.

Main script

The main script is responsible for calling in the debugging and high level modules to execute the program and prompt the user for full control of the program flow. It starts with calling in Fileio module to welcome the user, and prompt the user to ensure that source code is as required. TLE epoch, station name, and tracking interval are checked by the user, and only upon user input will the code continue it run, otherwise user can input 0 and the code will break out and stop execution. This logic follows through until the AOS/LOS table is generated by calling in the debugging module's AOSLOS function. The user can then check the AOSLOS listing on screen, select a satellite, and enter the index value of the satellite desired for further program execution. The main script will then call on the tracking data function form the debugging module to output the tracking data file for the ARO control computer. If the user would like debugging files, the main script prompts the user for whether or not the debugging file outputs are desired for the chosen satellite. If the user presses 1, then the program will output the debugging files for all ECI, ECF and TOPO coordinate systems of the chosen satellite via the debugging index functions.

Procedure & Observations

Simulated Software Testing

The following procedure reflects the code testing experience with Professor Chesser. Previously, as stated in lab reports 4 and 5, the team members had validated the different components of the code (AOS/LOS listing, azimuth elevation angles and rates, final tracking outputs and pointing files) on STK, for different satellites. This time the code would be tested with a non-developer of the code, the professor, to ensure the robustness of the software to be used at any instant in time with any of the 32 GPS satellites.

1. Professor Chesser selected a tracking interval and the team members created a text file for it.
2. The team members extracted the latest NORED TLE file from the celestrak website.
3. The TLE and tracking interval files were saved into the respective folder for the code.
4. The command window was reset and Professor Chesser assumed control of Nikeet's Computer.
5. The code welcomes the user, and prompts the user to press 1 if satisfied or 0 to abort the program.
6. The Professor pressed 1, the code displayed the TLE epoch times and user was prompted again.
7. The Professor pressed 1 and 1 again to confirm TLE epoch and station data as desired.
8. The code displayed the tracking interval and prompted the Professor again. The Professor stated "I am very satisfied" and pressed 1.
9. The code then printed the AOS/LOS files to a text file. This took some time.
10. The code printed the list of PRN satellites correlating each to a number from 1-32. The professor went back to the text file and chose a satellite.
11. The professor checked the access for PRN-23 on STK and did not find it on the access list.
12. Nikeet correctly identified that for some reason PRN-23 was not on the TLE list and user code was indeed robust.
13. The Professor double checked and confirmed this.
14. The Professor checked and confirmed all access times on STK to correlate very well with the python output.
15. The Professor protested that he had to open the text file, look for the satellite, correlate the satellite to its respective number (since they were not in PRN order).
16. The Professor selected a satellite and input into the code so the code could perform the tracking calculations for just one satellite.
17. Code prompted user if tracking output was desired, Professor pressed 0
18. Code prompted for debugging files, Professor pressed 1.
19. Debugging files were generated and sent to the professor.
20. Professor was satisfied with the performance of the code on STK when bringing in the ephemeris file and the pointing file, validating the robustness of the code. This can be seen in figure 1

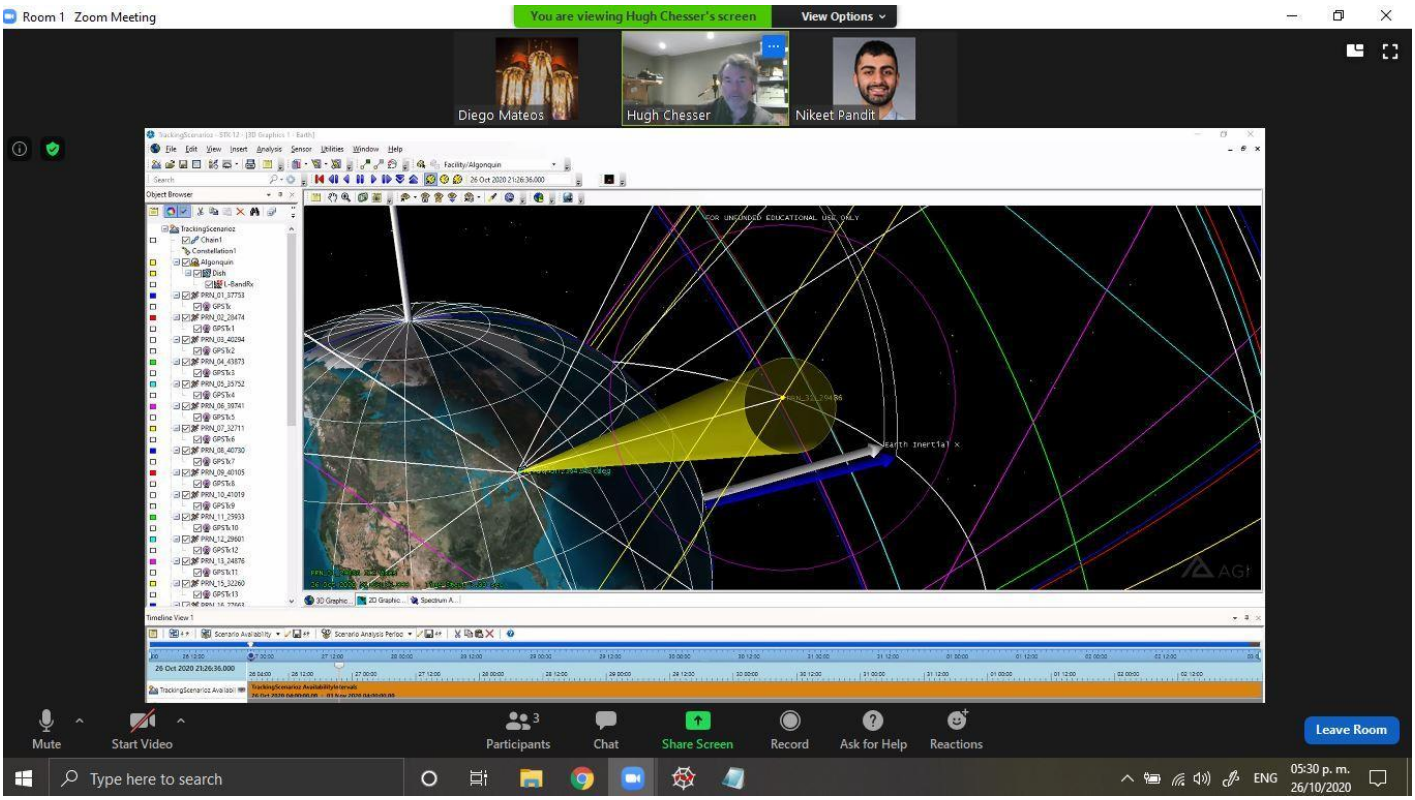

Figure 1: Validation of code on STK with the Professor

ARO Activities

In order to account for antenna gain, the signal was characterized by using existing GPS tracking code at ARO, in which the beam bandwidth and power levels were measured. The following procedure was done for this process.

1. The ARO operator loaded the pre-existing software to the control computer to track a GPS satellite.
2. The spectrum analyzer instrument which received the digitized signal was virtually shared with the class on zoom, which enabled the class to see the magnitude vs frequency plot on the analyzer.
3. It was noted that the spectrum analyzer had a RBW frequency of -49.762 kHz.
4. To get a clear signal on the spectrum analyzer, the centre frequency was set to the L band (1575.42 mHz). The no mixing setting was selected.
5. Then averaging was selected to remove high frequency noise in the spectrum analyzer, where about 50 sample averaging was selected to provide a good view.
6. The spectrum analyzer had a reference level, so we lowered this closer to the peak level which we were seeing. This was set to about -80 dBm.
7. We set the range then to 30 dB.
8. Offsets in azimuth and elevation were commanded by a voluntary student taking virtual control over the ARO software to interact with the antenna.

9. These offsets aided in characterizing the beam shape and power, the following table was acquired after multiple variations of azimuth at constant elevation, vice versa, and then both combined.

Azimuth offset (deg)	Elevation offset (deg)	Power (dBm)
-2	0	-106
-1	0	-92
1	0	-93
2	0	-104
0	-1	-92
0	-2	-100
0	-1.5	-110
0	-0.5	-90
0	0.5	-90
0	1	-100
0	1.5	-105
0	0.5	-90
-1	0.5	-94
-1	-0.5	-89

10. With the above observations using the ARO standard tracking code, the expected maximum power level is -89 dBm with -1 AZ offset and -0.5 EL offset.
11. The developed python code was now run by the team members, PRN-26 was selected and the tracking file output was generated in order to compare the results with the spectrum analyzer measurements.
12. The python code Doppler shift was not determined experimentally, and represents a deviation from the lab manual.
13. The radio frequency power levels were compared to those of the spectrum analyzer at its max level -89 dB. As required, the system gain was updated to account for the antenna gain.
14. The following plot of the antenna power as a function of AZ & EL offsets can be seen below:

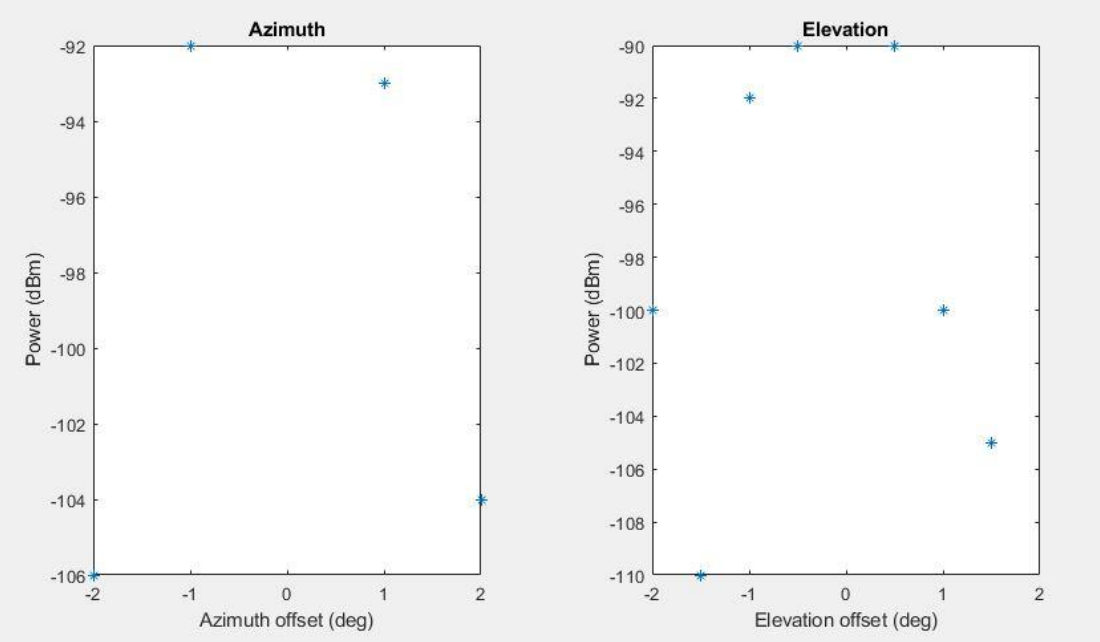

Figure 2: Azimuth & Elevation Antenna Gain Plots

15. Additionally, polar plots are provided in an attempt to show the antenna shape. It is likely that not enough resolution or offset variables were tested to obtain an actual physical representation of the beam width.

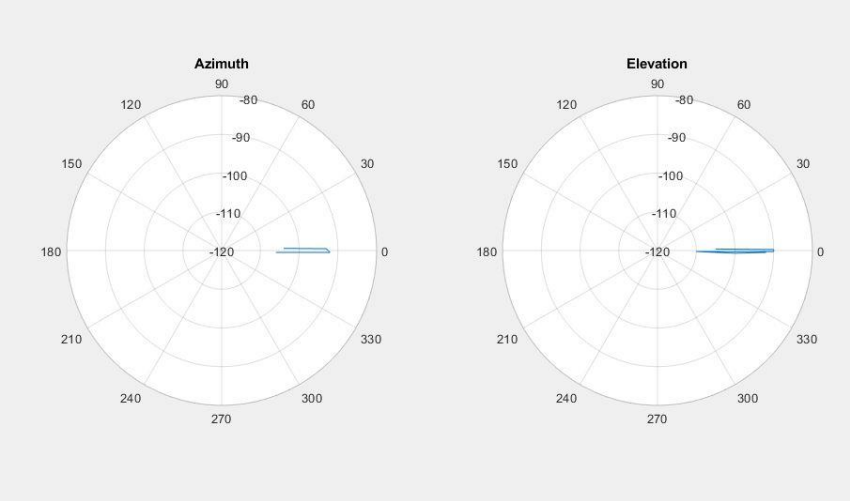

Figure 3: Polar plots of antenna beam width (AZ & EL)

16. Once it was the team's turn, the updated code was run, and PRN-26 was selected for tracking, as recommended by the ARO operator due to close proximity to the direction at which the antenna was already pointing.
17. The satellite was selected for tracking, the tracking output ASCII file was generated and sent to the ARO operator.
18. The file was uploaded to the ARO control computer, and the team members were happy to

see an antenna peak on the spectrum analyzer of -89 dBm at the optimized offsets determined in steps 5 and 6, indicating the successful tracking of the GPS satellite to an extremely high degree of precision. In this figure (4), however, the received level is -93 dB, which is slightly lower than the peak level seen over the duration of tracking.

19. It was observed that the power level was fluctuating from about -89 dBm to -94 dBm.

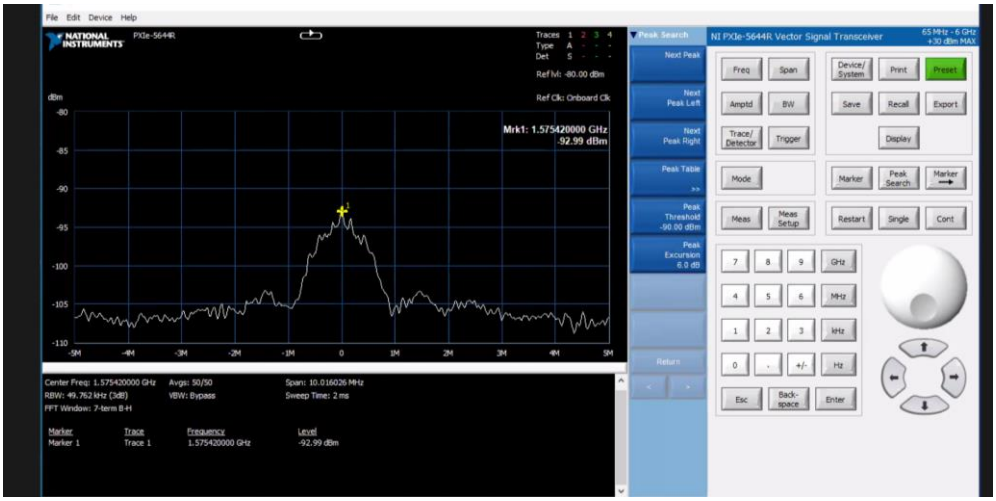

Figure 4: Polar plots of antenna beam width (AZ & EL)

20. This represents successful completion of the tracking section. There was no issue with formatting of the ASCII control code whatsoever.

Discussion & Interpretation

Simulated Software Testing

Since the group had performed objectives ahead of pace, as discussed earlier, there has been minimal adjustments and additions to the software package. Instead, the group presented the results of the code testing on this report. The results from this test were as expected. The selected satellite was tracked nominally, and the pointing file provided to the professor resulted in tracking success. This team has seen success in the tracking code since November. The only fundamental change to the code, as mentioned in the code functionality section, was that the AOS/LOS table now prints to the console, as per Professor Chesser's request.

ARO Activities

The ARO standard software, upon execution and obtaining the tracking signal had a power level of -89 dB antenna gain. The user developed code had an expected power level of -155 dB. The

difference between the expected power level and the antenna gain is 66 dB, and therefore represents the system gain. It is important to note that upon attempting to characterize the received beam shape in elevation and azimuth planes, nulls were observed, and care had to be taken so as to not confuse a null for a misreading or the end of the beam width. For example, in the elevation plot, the second point at an offset of -1.5 deg of elevation is a clear null, since the further offset point at -2 deg has a higher power level and so does the adjacent point at -1 deg offset beside it. Upon characterizing the beam shape and antenna gain, it could be confidently confirmed that the antenna losses such as temperature dependence and loss of signal through the hardware setup (instruments, power lines etc.) were accounted for before reaching the spectrum analyzer.

It is important also to note that characterization of this beamwidth is not required to track and receive the GPS satellite signal, as the ARO antenna does require this instruction, nor accounting for it when sending signals. It is important to note that in order to characterize the beam bandwidth and shape to ensure that sufficient power is received and bandwidth resolution to ensure that all of the modulated signal information within a radio frequency carrier signal are accounted for and not dropped. It is also interesting to note that more tests could be performed to account for the full beam shape as mentioned in step 15 of the ARO activities procedure. This could include accounting for higher azimuth resolution and involving greater azimuth and elevation offsets to characterize the side lobes. In addition, more decimal places at the high received power sections could be tested to observe the main lobe power shape. It also must be noted that calibrating the system gain must be done for each observatory individually. At the end it was suggested by this team to perform experiments by: Starting at +2 degree Az offset, and ranging it to -2 degree offset and screen recording. The same operation could have been performed with El offset, and both simultaneously. Then the results could be tabulated, and an antenna beamwidth could be characterized more precisely.

An observation of notable consideration is the fluctuation of power throughout tracking the satellite. This was observed and mentioned in step 19 where the power fluctuated from a peak of about -89 dBm to a low of about -94 dBm. This could possibly be for a number of different reasons. For example, the azimuth and elevation rates of the tracking code are not completely perfect, and upon following the spacecraft, there is perhaps some offset with the position of the spacecraft relative to the center of the antenna. Perhaps the antenna was hitting nulls. In addition, there could be increased/decreased ionospheric and atmospheric interference as the spacecraft moves quite fast in relation to the earth. In addition, there could possibly be fluctuations involved with the antenna hardware equipment temperature and impedance.

Conclusion

In conclusion, the team members were able to successfully develop a collaborative object-oriented software that given station parameters, TLE data, and tracking interval inputs, was able to output a standardized ASCII tracking file compatible with the Algonquin Radio Observatory's control computer, to track and receive a GPS satellite signal. This is a remarkable accomplishment and the team members have been very happy with the exponential learning experience. The multiple different methods of testing and validating the code were done to verify the robustness of the

software, all of which were successful and useful for understanding the development of tracking software. STK simulates software testing was an integral part of the process to debug and validate the code, and the final ARO testing was required to validate the code for its intended purpose. Further, through experimentation we determined the actual system gains at ARO to properly calibrate our tracking model. Through these experiments and experiential testing, we determined signal level received which was based on the precision and accuracy of the developed code's pointing parameters. The project has provided insight into radio science, ground operations and astronomy, in the sense of what is required to receive and characterize signals coming from outer space.