Sabrina DeSoto

Computation in the Physical Science

9 November, 2015

**PROJECT SEARCH FOR SPACE TREASURE**

10 MINUTE PLAN

INTRODUCTION

The purpose of this project is to use fringe rate mapping via a python code to analyze any microwave amplification by stimulated emission of radiation (MASER) data and return a location on a fringe (a wave like pattern on the sky) for where the MASER is. This work can be divided into two major sections. The first part is to take the raw data collected and confirm a detection was made. The second part is to take that detection data and to use it to establish the location of the MASER on the fringe. Both these results were achieved on a small scale with the sample data provided (located in Data/antena.dat) and can be seen from the outcomes of running the projects’ shell script.

RUN SCRIPT

Once the main project directory (Sab\_Final\_Project) and all it’s subdirectories are downloaded the project can be ran from inside this main directory. To run the project for the sample data provided open a terminal, move into the main project directory and enter:

./run\_script.sh Data/antena.dat project\_final\_product.csv

Both the ‘Data/antenna.dat’ and ‘of\_project’ are arguments so they may be changed to run the python script for other data sets. The first argument must be a path to the input data file and the second argument is any name that will be assigned to the most important out come of the project, a dataframe of average phase and this phase converted to a location on a fringe. In general the python can script can be run by:

./run\_script.sh path\_to\_input\_data output\_name.file\_type

However the loading function of this project is specific to raw antena data for 24 observation periods, that are separated by headers. To run this program for data not formatted this way the load\_data function in the python script itself would need to be changed (most importantly the length\_of\_file=12624).

Running the python script for the file types it was intended for (with the first command above)produces the following outcomes from each function:

load\_data(): the outcomes from this function are chunks of antenna data sorted by observation period saved in Data/Analyzed\_data as . obv\_period#\_data.csv files

make\_phase\_diagrams(): the outcomes of this function are all the phase diagrams for each observation period saved in the Results directory as Plot\_obv\_period#.pdf files

signal\_phase(): the outcome for this function is one file for all the average signal phases and phase locations saved in the Results directory under whatever the second argument put in was (example was project\_final\_product.csv)

As the python scripts run it will first print a message that says the number of observation periods there are and then print ‘done’ when it’s finished running. To see all the results one may navigate to either the Data/Analyzed\_data subdirectory or the Results directory using using cd. Each time the script is ran the outputs of each function pile up in those folders, thus before running the script again for the same input data file it is good to use the rm function in both subdirectories to delete the unwanted old outputs. To view any of the outcomes one need only type open and that files name. Bellow states the method the script runs on to produce these results.

METHOD

The script begins with creating an imports library containing all necessary coding aides. Following this the arguments and variables are defined before starting the main(): function. Under the main function the order of execution for the three functions that produce an outcome are stated. The first function called is load\_data() which, loads the input data file using pandas and a while loop to separate each chunk of observation period data that’s separated by headers. An assertion is added to check that the function separates 24 chunks (12hrs of data collection/ 30minute observation periods) of data. These data chunks are then appended to table (defined in variables) to be referenced later and also saved separately as mentioned.

Next the make\_phase\_diagrams() function takes this data from the table and similarly loops through each chunk, creating two subplots that together are a phase diagram. The first of the subplots is a scatter plot of the phase vs. frequency, which will appear as the top half of the phase diagram. The bottom half is a line plot for the amplitude vs. frequency.

The third function defined as signal\_amplitude() is not called on it’s own. The purpose of this function is to set an amplitude threshold value witch defines the a signal amplitude is any detection greater than this threshold. It then appends all these amplitudes, separated for each observation period, to a list (defined before main() as signal\_ampl\_list). The last function called is signal\_phase(). This function also takes in the list ‘table’ and then merges it to the signal amplitudes saved to ‘signal\_ampl\_list’ in order to make a data frame of all values corresponding to a MASER detection. From this data frame the phase values are extracted and averaged for each observation period. Finally these values are converted to a location on a fringe pattern (by dividing them by 360 degrees) both the average phase and phase location are saved as a data frame assigned to argument 2. This concludes the python script. For more details see the python and notebook scripts located in the Bin folder or the project proposal and report located in the Doc folder.

KEY VARIABLES AND FUNCTIONS IN PYTHON SCRIPT

Libraries:

pd = pandas library

plt = matplotlib.pyplot library (for plotting)

sys = sys library

Arguments:

input\_data\_file = sys.argv[1]

output\_name = sys.argv[2]

Variables:

length\_of\_file=12624

table=[]

signal\_ampl\_list=[]

threshold\_ampl=0.010

Functions, inputs and outputs:

load\_data(input\_data\_file) = separated data chunks

make\_phase\_diagrams(table) = phase diagrams

signal\_phase(table) = phase average and phase location data frame