ECE 480 SENIOR DESIGN

Using GNU Radio for Signal Phase Measurements

George Godby
3/27/2014

Abstract

This document focuses on how to set up a flow graph in GNU Radio Companion that will measure the phase of an RF signal using a Software Defined Radio (SDR) . The goal is to help the user better understand how signal processing works by explaining the purpose of each component in the flow graph. Additionally,

information is given on how to use computational software like MATLAB and scripting languages, such as, Python which can be used to perform post processing analysis of the phase data.

Introduction

This application note explains how to extract phase information from an RF waveform. The motivation for the topic is based on the fact that signal phase information may be used for signal direction finding and angle of arrival.

There are two different methods for measuring the phase of a signal. One method involves using a highly accurate reference signal as a standard. The reference signal phase can be compared to the phase of an incoming received signal and the difference between the two is the phase offset. The other method, which is demonstrated in this document, is simpler and relies on only using to antennas.

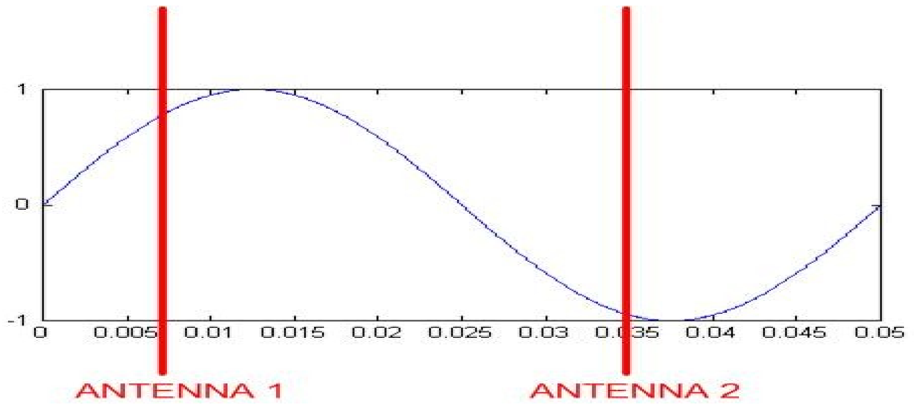
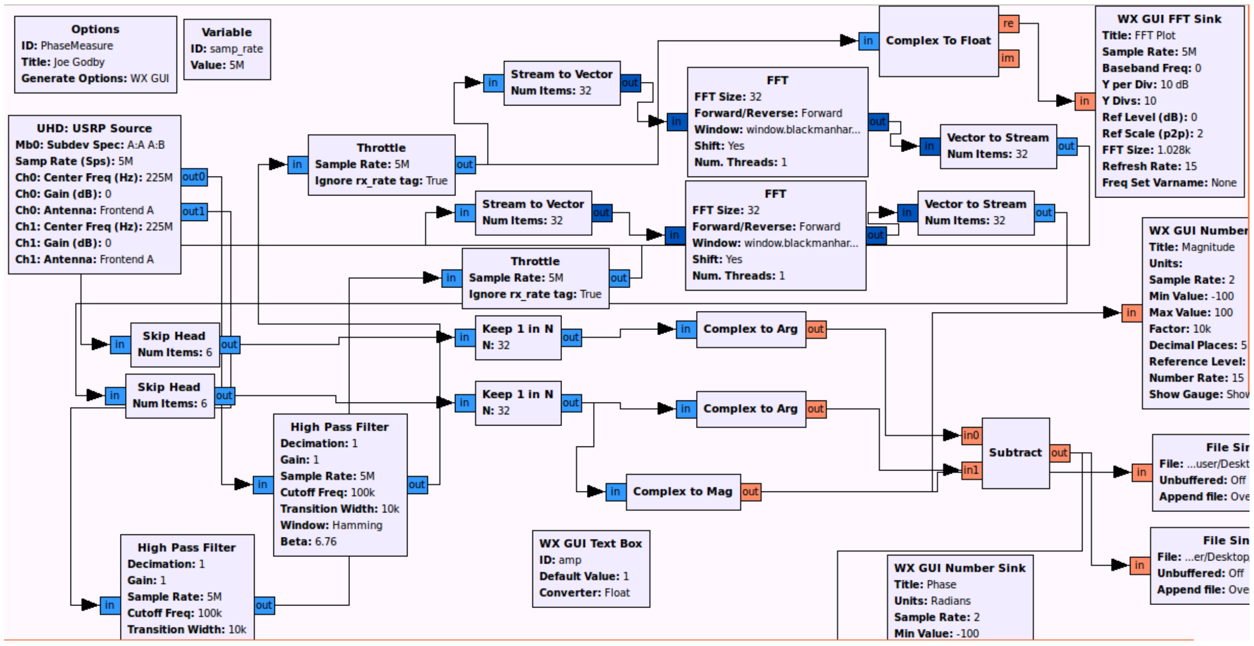


Figure 1. Using two antennas

Figure 1 demonstrates two antennas spaced sufficiently far apart for the wavelength under measure; here, each antenna will receive a different phase of the signal. We can exploit this fact by using a Software Defined Radio (SDR) that is capable of receiving two separate signals and by building a software circuit (flow graph), to extract this phase information from each antenna separately.

GNU Radio Flow Graph

GNU Radio is open source development software that is used in the design of RF software circuits. An additional feature of GNU Radio is the “companion” feature; this allows the user to create a “flow graph” of the circuit they are building. Figure 2 shows the flow graph of the phase circuit for this project. The end product can prove to be somewhat overwhelming for the beginner and this documents attempts to break the circuit up and explain what each section is doing.


Options
ID: PhaseMeasure
Title: Joe Godby
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 5M

UHD: USRP Source
Mb0: Subdev Spec: A:A A:B
Samp Rate (Sps): 5M
Ch0: Center Freq (Hz): 225M
Ch0: Gain (dB): 0
Ch0: Antenna: Frontend A
Ch1: Center Freq (Hz): 225M
Ch1: Gain (dB): 0
Ch1: Antenna: Frontend A

Throttle
Sample Rate: 5M
Ignore rx_rate tag: True

Stream to Vector
Num Items: 32

FFT
FFT Size: 32
Forward/Reverse: Forward
Window: window.blackmanhar...
Shift: Yes
Num. Threads: 1

Complex To Float

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 5M
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.028k
Refresh Rate: 15
Freq Set Varname: None

Vector to Stream
Num Items: 32

Stream to Vector
Num Items: 32

FFT
FFT Size: 32
Forward/Reverse: Forward
Window: window.blackmanhar...
Shift: Yes
Num. Threads: 1

Vector to Stream
Num Items: 32

WX GUI Number
Title: Magnitude
Units:
Sample Rate: 2
Min Value: -100
Max Value: 100
Factor: 10k
Decimal Places: 5
Reference Level:
Number Rate: 15
Show Gauge: Show

Throttle
Sample Rate: 5M
Ignore rx_rate tag: True

Skip Head
Num Items: 6

Skip Head
Num Items: 6

Keep 1 in N
N: 32

Keep 1 in N
N: 32

Complex to Arg

Complex to Arg

High Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 5M
Cutoff Freq: 100k
Transition Width: 10k
Window: Hamming
Beta: 6.76

Subtract

File Sin
File: ...user/Deskt
Unbuffered: Off
Append file: Ove

Complex to Mag

File Sin
File: ...er/Desktop
Unbuffered: Off
Append file: Ove

High Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 5M
Cutoff Freq: 100k
Transition Width: 10k

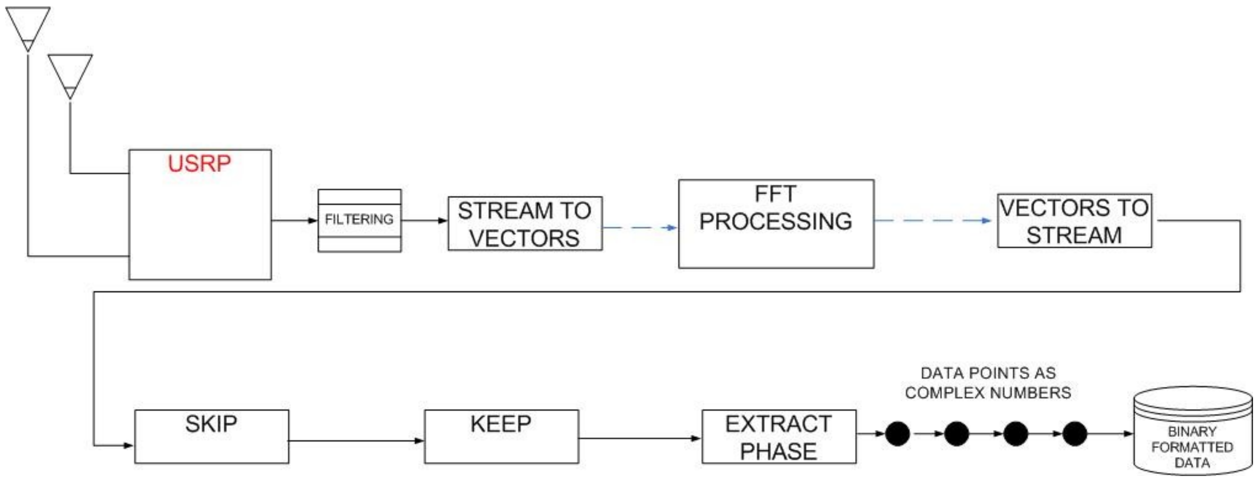
WX GUI Text Box
ID: amp
Default Value: 1
Converter: Float

WX GUI Number Sink
Title: Phase
Units: Radians
Sample Rate: 2
Min Value: -100

Figure 2. Phase flow graph in GNU Radio Companion

High Level Overview

The circuit shown in figure 2 can best be summed up in the simplified block diagram seen in figure 3.



Before diving too deep into how the circuit works, let’s take a moment to look at the big picture and consider what must happen first before any signal processing may take place. Figure 4 shows the RF spectrum which will be presented to our signal processing circuit.

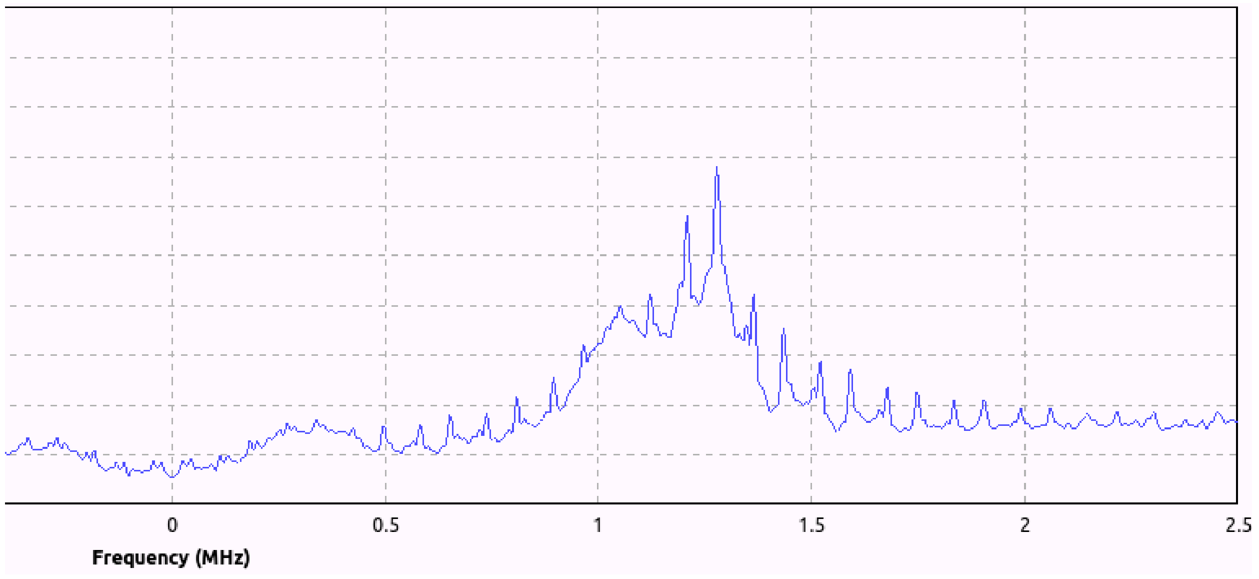


Figure 4. RF Spectrum

Recall that the SDR is not a tuner, so we are given 2.5 MHz of bandwidth here to sort through to recover our signal, which is at 1.3 MHz. As a side note, the amount of bandwidth is based on one half of the sample rate, which is 5 MHz in this case. Now we must solve our first problem: How to recover the signal at 1.3 MHz aside from all the other signal available in the given spectrum of 0 to 2.5 MHz. There are two ways to do this: (1) Create a quadrature demodulation to bring in the signal or (2) Using Fast Fourier Transform (FFT) signal processing. For this document, we will use FFT approach.

Flow Graph Components

Filter

This is pretty standard for any RF circuit, as it helps to keep noise from getting into the circuit

Stream to Vectors

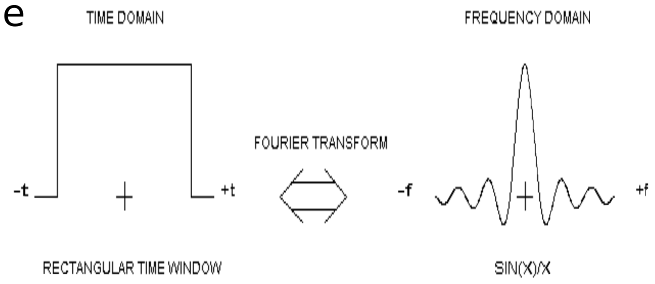
In order to prepare for our conversion from Time Domain to Frequency Domain, we need to break our waveform down into an array of Time Domain samples

0	**1**	**2**	**3**		**bin**				**N**

Where N must be a power of 2, such as, 64, 128, 256…. The squares are referred to as ‘bins’.

FFT

Once our signal is in vector format, it is run thru the Fast Fourier Transform (FFT). This is the signal processing block which converts the signal from the Time Domain to the Frequency Domain, by using algorithms beyond the scope of this writing. What is important here is that the FFT output will give us an array of data which we will later index in order to get only the specific frequency of interest and discard all of the other frequencies.



Vector to Streams

This block just strings everything back together again so the data is in one continuous flow.

Skip

Figure 5 depicts where we are now in the signal process. The squares represent the bins, keep in mind that everything here has been magnified for the purposes of illustration. Typically the numbers bins would be on the order of 1024, not 16.

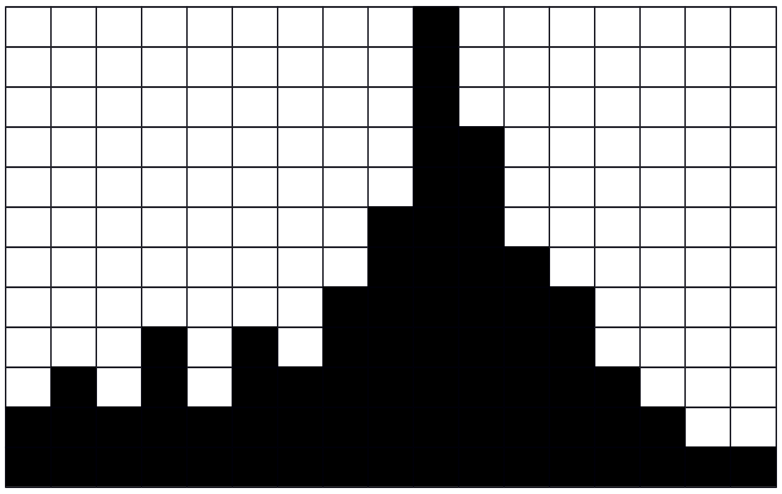


Figure 5. FFT output showing bins

Our goal here is to extract only the bin with our frequency or ‘skip’ to that bin. In order to calculate this, first let’s first consider the size of each bin.

$$Bin\,\frac{\overset{FFT¿¿}{¿\,Sample\,Rate}}{¿}$$

Sample rate is how many times you originally sampled the signal back in the Time Domain.   Figure 6 shows us sampling our input signal 16 times over the period of one wavelength and since or frequency of interest is 1.3 x 10^{6}, our sample rate is 20.8 x 10^{6} or 20.8 MHz.

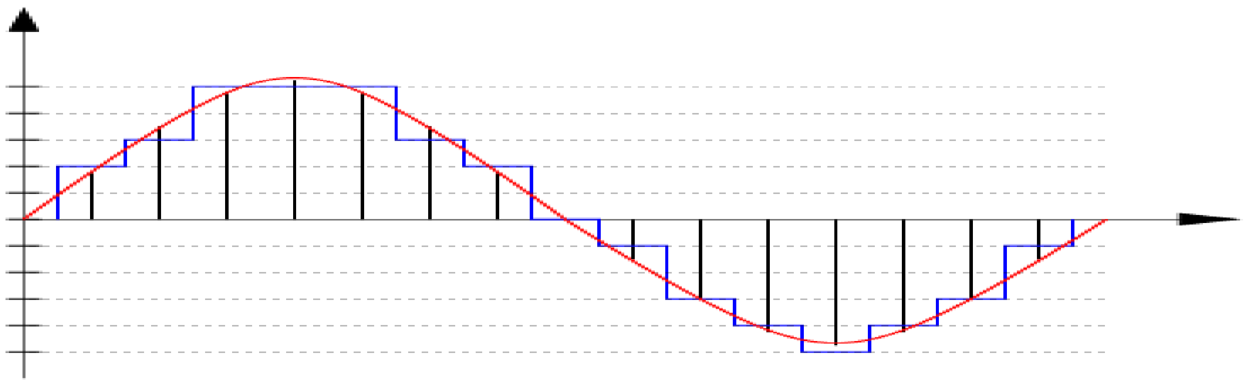


Figure 6. Sampling of input signal

With regards to FFT size, let’s give this some though (remember it must be a power of 2), if FFT is small than we will have big bins and if FFT is large than we will have smaller bins, which means more resolution. On a practical note here, theoretically we can pick a very high number for the FFT size and have excellent signal resolution, however, if you are planning to actually implement this into receiving a real signal, keep in mind that real RF is somewhat of a moving target. If your bin sizes are too small you will never be able to hit the right bin, do to signal drift. So for practical purposes let’s make FFT size = 128.

BIN SIZE: $\frac{20.8x\ 10^{6}}{128} = 162.5\,x\ 10^{3}$

Now to find our bin of interest simply divide the frequency that we wish to recover by the bin size.

$$\frac{1.3\,MHz}{162.5\,KHz} = 8$$

One last thing, since we are using a zero based system here, we need to add 1. The actual bin of interest is bin 9. Figure 7 shows we have given ourselves some room for signal drift.

BIN

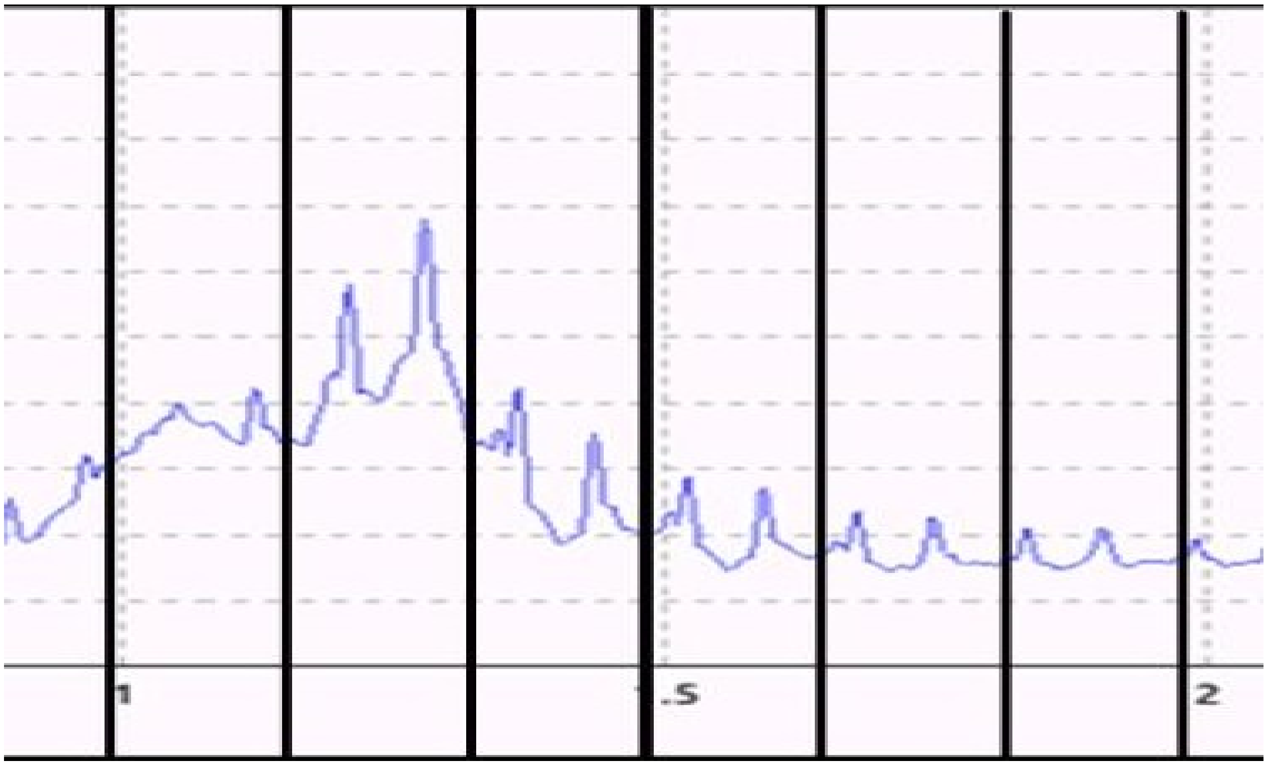


Figure 7. Bins laid over FFT

Keep

This part of the software is called ‘Keep 1 in N’ and that is just what it does, where N is our bin of interest, bin 9. It passes only bin 9 along to the phase extraction circuit.

Phase

The phase block out outputs the raw phase data points in binary format in the form of a complex number, which is good since we can use this in our post processing script to compute both phase and magnitude.

Circuit as a Whole

If we look at the GNU flow graph we see there are actually two phase circuits, one for each antenna. At the end the extracted data is just subtracted from each other, giving a phase difference. The phase is saved to a file in binary format.

Handling Captured Data

Once the raw binary data has been captured into a data file, a handler program must be created to extract the data from the file and do something useful with it. These files accumulate millions of data points very quickly and therefore must be processed according the so the user is not overwhelmed with so much information that nothing makes since.

Phython Scripts

The recommend language for handling the data files is Python. Python makes a good choice since it is free and comes with very effective mathematical computational add-on libraries like SciPy and NumPy.

The following will provide a high level over view of the data handling script, such that it demonstrates the writers understanding of the coding process rather than attempting to teach Python to the reader.

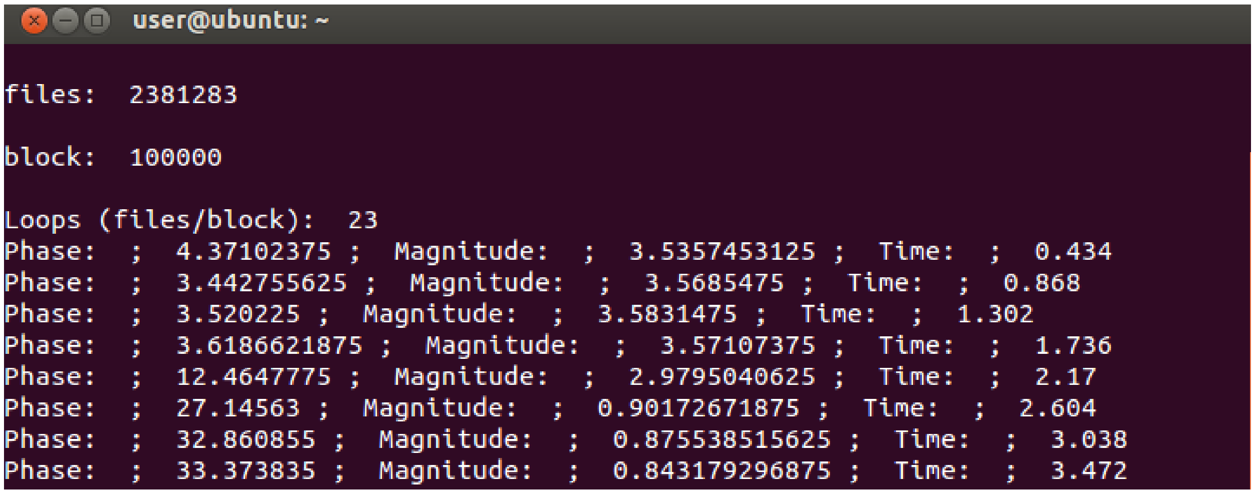
File Handling Method

The NumPy library provides us with a file handler method that when called takes in the file name and options. An options parser handles all of the options, the only option relevant here is the Block option. The Block option allows us to choose how many data point to bring in at one time and defaults to 1000. This will be useful in determining our array size in the Get Data method.

Get Data Method

The get data method converts the binary data to ascii format so that can be processed. The amount of data processed at one time is determined by the block option. For instance, if Block = 10000, then 10000 data points are loaded into an array. Even though 10000 seems large it is less than 500 mS worth of data, so the best thing to do is take the average of all 10000 point and loop to the next block of 10000 points. The data iteration is done with a for loop: `for count in range (self.files/self.block_length):`

Where the *range* is determined by (file size/block size). Count is initialized at zero and then incremented by one on each loop. Using a loop will allows us to accumulate enough average data points to begin drawing some conclusions about what we are measuring. A note about measuring phase is that the data points are prone to random spikes for short periods of time, so that data is best view averaged out over time. Additionally, it can be shown experimentally that it takes 434 mS to log 10000 data points for one loop and thus a timestamp can be added to each loop. Figure 8 shows a portion of the final data as it is output from the program. Note that semicolons have been concatenated to the lines of data so the information can be parsed out later by a plotting program or imported into a spreadsheet.



```
files:  2381283

block:  100000

Loops (files/block):  23
Phase:  ;  4.37102375 ;  Magnitude:  ;  3.5357453125 ;  Time:  ;  0.434
Phase:  ;  3.442755625 ;  Magnitude:  ;  3.5685475 ;  Time:  ;  0.868
Phase:  ;  3.520225 ;  Magnitude:  ;  3.5831475 ;  Time:  ;  1.302
Phase:  ;  3.6186621875 ;  Magnitude:  ;  3.57107375 ;  Time:  ;  1.736
Phase:  ;  12.4647775 ;  Magnitude:  ;  2.9795040625 ;  Time:  ;  2.17
Phase:  ;  27.14563 ;  Magnitude:  ;  0.90172671875 ;  Time:  ;  2.604
Phase:  ;  32.860855 ;  Magnitude:  ;  0.875538515625 ;  Time:  ;  3.038
Phase:  ;  33.373835 ;  Magnitude:  ;  0.843179296875 ;  Time:  ;  3.472
```

Figure 8. Results

Final Results

Figure 9 show the final results with all phase data being graphed as a function of time. Table 1 shows the test bed parameters.

TIME IN SECONDS	**ANTEN NAS ANGLE**
0 TO 40	0
41 TO 60	45
61 TO 80	90

81 TO 100	45
101 TO 120	0

Table 1. Testing Parameters

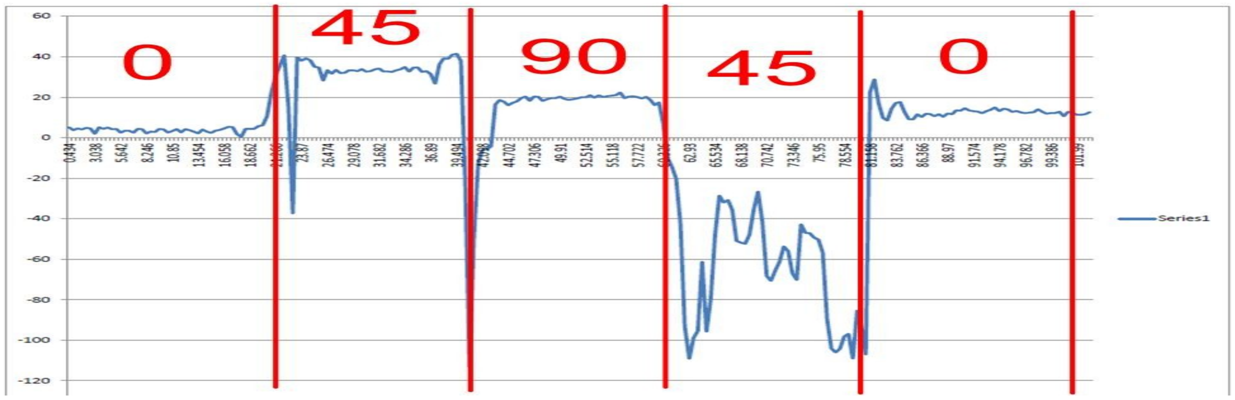


Figure 9. Data plot. Time is printed on the x axis.

Conclusion

It is clear from the data in figure 9 that there is a direct correlation between the antennas position and the recorded phase as it relates to time. An improvement needs to be made in calibrating the scale so the results correlate directly to the angle of the arriving signal.