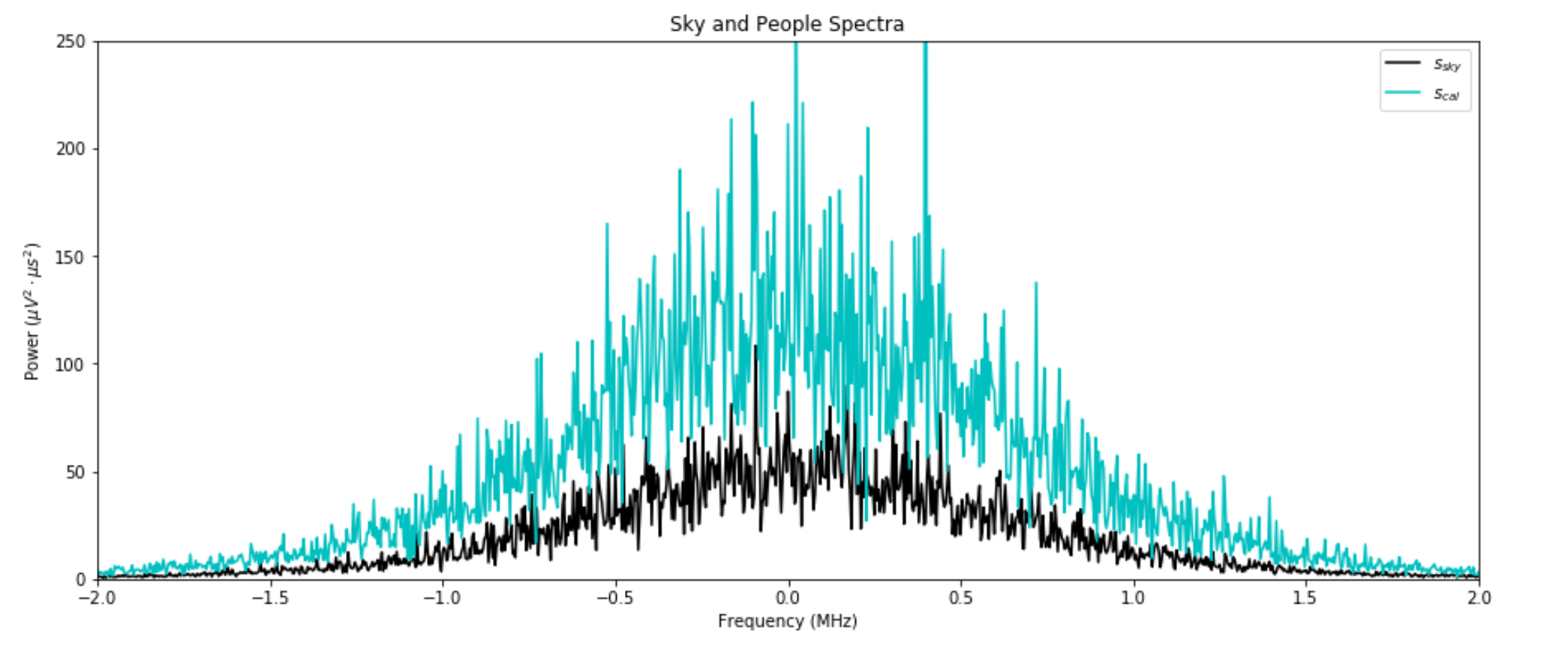
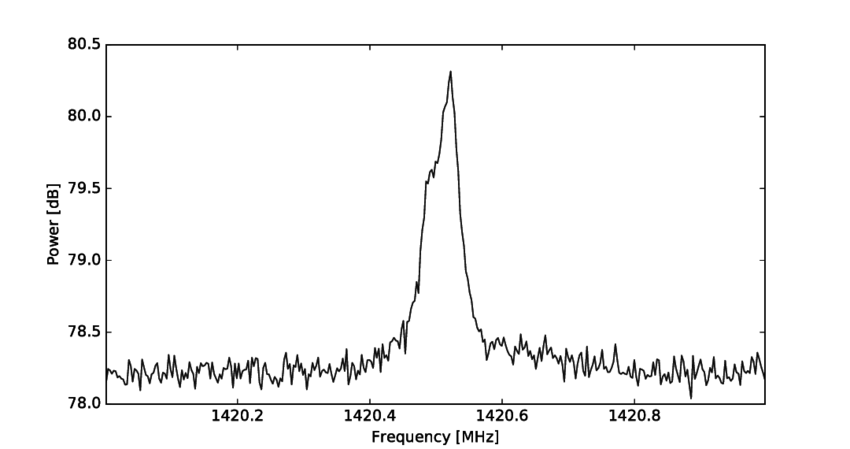
**Radio Astronomy Project**

**Introduction**

In this project, we will go through an extremely guided walk-through of handling data of emission spectra from the Milky Way. This data was taken by a telescope on the roof of Campbell Hall. In the same way we perform multiple trials in a science experiment, we pointed our telescope at the sky and took multiple “images” of the Milky Way to collect as much data as we can. “Images”, in this project, are actually emission spectra. They look like this:



And after extensive processing, they can look something like this (ignore units for now):



As we said in class, the Milky Way (as well as our entire universe) is permeated by a diffuse sea of hydrogen molecules. Hydrogen molecules emit light at a very defined frequency, which is 1420.4 MegaHerz (MHz or Herz), for that is its “chemical fingerprint”. Since there is so much hydrogen in our Milky Way, we will see a distinct peak in our data around at 1420.4 MHz, like in the image above. This peak is actually so well-known, it has a special name: **the 21-cm line**.

Overall, the majority of this project will surround the extensive process of taking raw data of hydrogen spectra and performing data reduction. *Data reduction* describes subtracting noise (extraneous pieces of information) from the data so that your interpretation of the data is entirely based on what you’re actually looking at, and not some random light you collected from a random supernova in a distant galaxy or whatever.

**The Goal of Project 2**

In this project, you will be practicing the following concepts from lectures and homework:

* Dictionaries
* Classes and Objects
* 1-D Arrays
* Plotting
* Importing and utilizing numpy, scipy, and matplotlib as other important packages
* 2-D Arrays
* File-io

We are on a giant rock swimming through a diffuse sea of neutral hydrogen which permeates the entire Milky Way. In this project, you will play with the emission spectra of neutral hydrogen that we have collected by pointing a telescope at a random direction in the sky. By the end of the project, you will be able to not only to estimate the brightness of that hydrogen source in that direction, but also estimate how fast it is moving relative to us.

**Logistics**

This project is worth 20% of your grade. We are grading on correctness. However, you are able to gain extra credit by applying the Clean Code Tips that we have been going over in class.

Please be prepared to spend more time on this project than Project 1, which is not only more programming intensive, but conceptually intensive with regards to the physics and math that is involved. This project draws inspiration from a project in AY 121, which is our Radio Astronomy lab course. The Radio Astronomy Lab, requiring substantial programming skills, is the reason this DeCal exists. Many who have taken the lab consider it one of the hardest upper-division courses offered by the Astronomy Department; we therefore ask you to take what you learn in this project seriously as the skills will inevitably ease your problems in the future.

Again, please remember to reach out if you have any questions. In particular, if your questions are related to the astrophysics of this project, ask Oscar or Christine since both have taken AY 121.

**Some Notes Before We Begin**

**Writing a Draft**

Up until now, you have been using Jupyter Notebook in class and for homework, but now we will see how it’s used within a project with many moving parts. There are benefits and drawbacks to using Jupyter Notebook.

**Benefits:** Jupyter Notebook allows you to run portions of your code (called “cells”) out of order without having to run the entire program; this makes it easier to informally diagnose smaller sections of your code (which is why we use it for homework). In addition, Jupyter Notebook saves any old and new plots that you made and allows you to view them until you run the cell again. Most importantly, Jupyter Notebook places less pressure on you to write generalized functions from the get-go--instead you can write code that does something specific and generalize it to a function later.

**Drawbacks:** On the other hand, running cells out of order makes it easy to misremember what value a variable is assigned to. Due to it’s messy nature, it is often difficult for others to read and test.

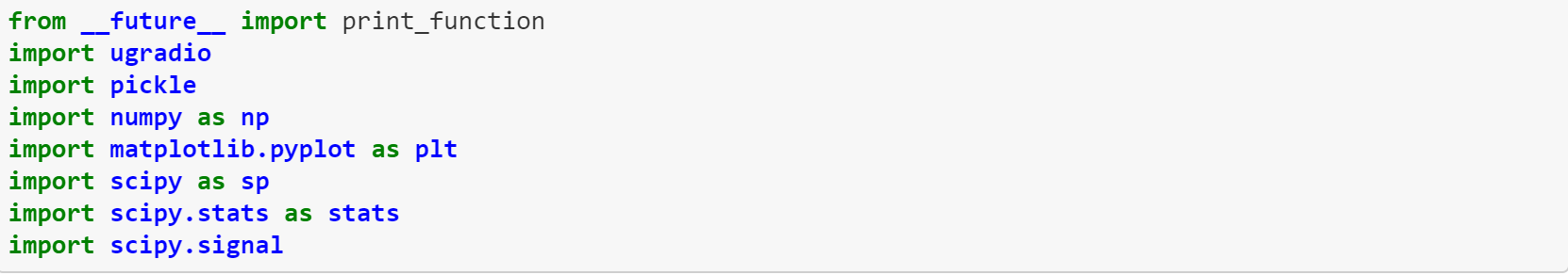
All-in-all, Jupyter Notebook is good for drafting code and testing what sections of your code is outputting...but bad for readability, running your entire program at once, and long-term software design. It is generally okay to use Jupyter Notebook when you are either only writing code for yourself or testing code that you will officialize in an actual .py file later.

In this project, you will use Jupyter Notebook to play around with some of the data we will be giving you. However, you will also turn in a .py file which is where you will write functions to automate the processes that you do in the Jupyter Notebook.

**A Note About Importing the Libraries You Need**

With more coding experience, you will find out that people across the internet have already written code for what you are trying to do. In order to use functions that other people write, you have to download “packages” or “libraries” and use “import” statements in your code to get access to the functions inside those packages or libraries. In this project, and likely in future Astro classes, you will be borrowing heavily from libraries like **numpy** and **scipy**.

In general, you will not know immediately what libraries you plan to import; however, whenever you decide to import a library, it is conventional practice to **collect all your import statements at the top of your file**, like so:



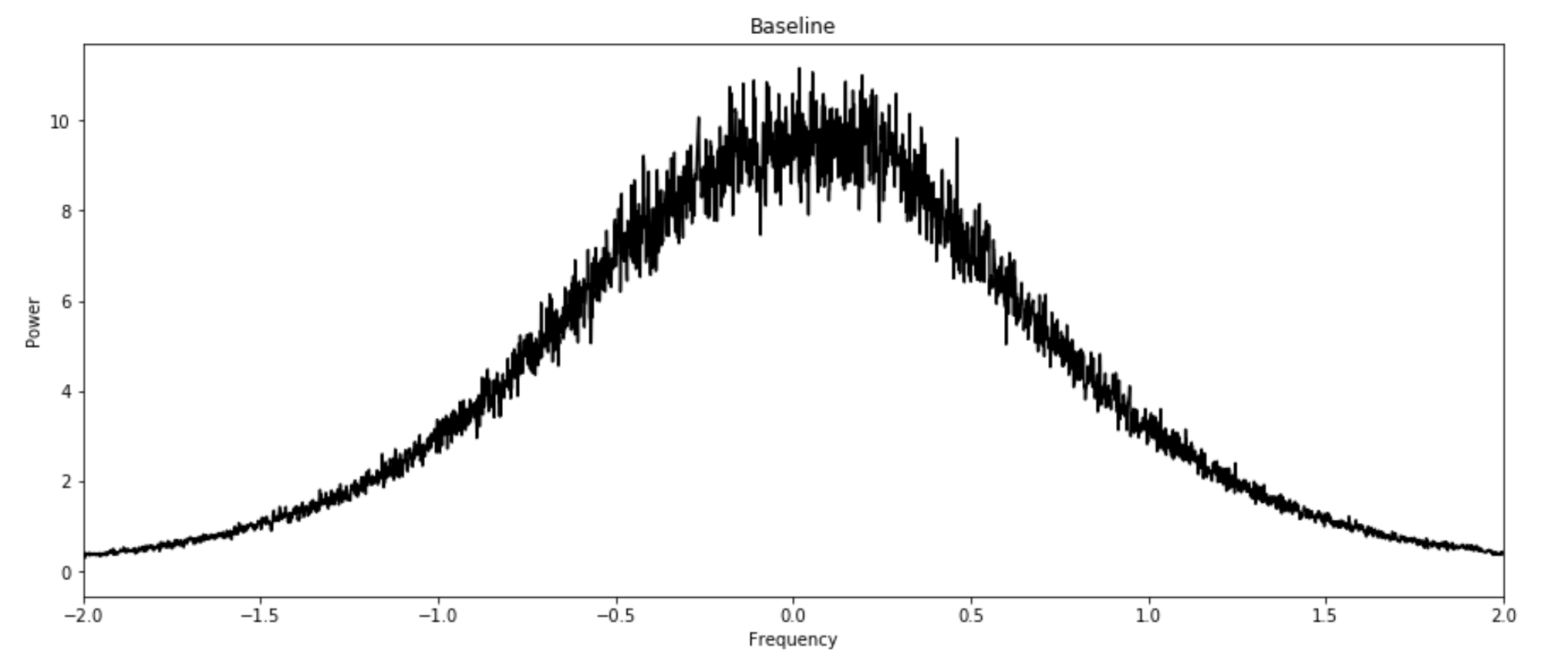
***Phase 1: Opening Data***

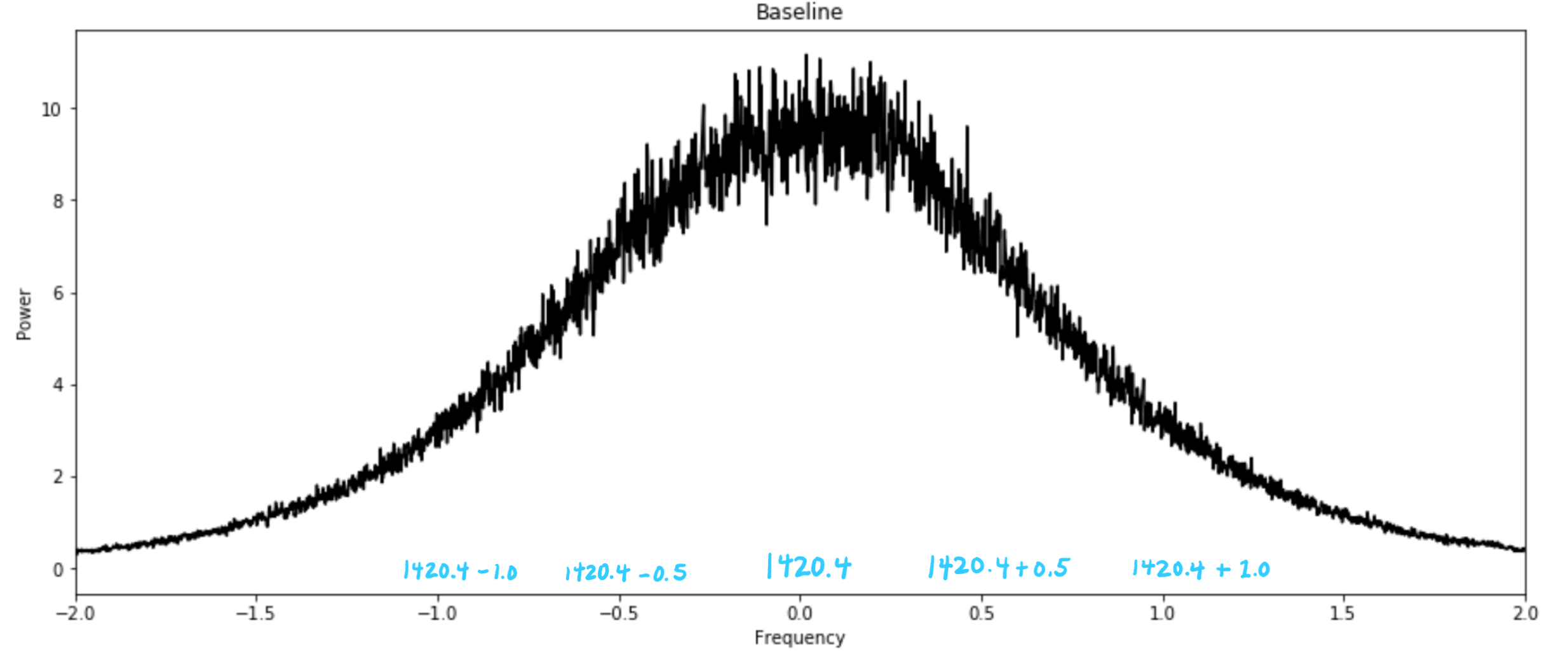
All spectra that you will be dealing with in this project are numpy arrays containing 16,000 data points. They will be stored in a file labeled project2.fits under different extensions, found here:

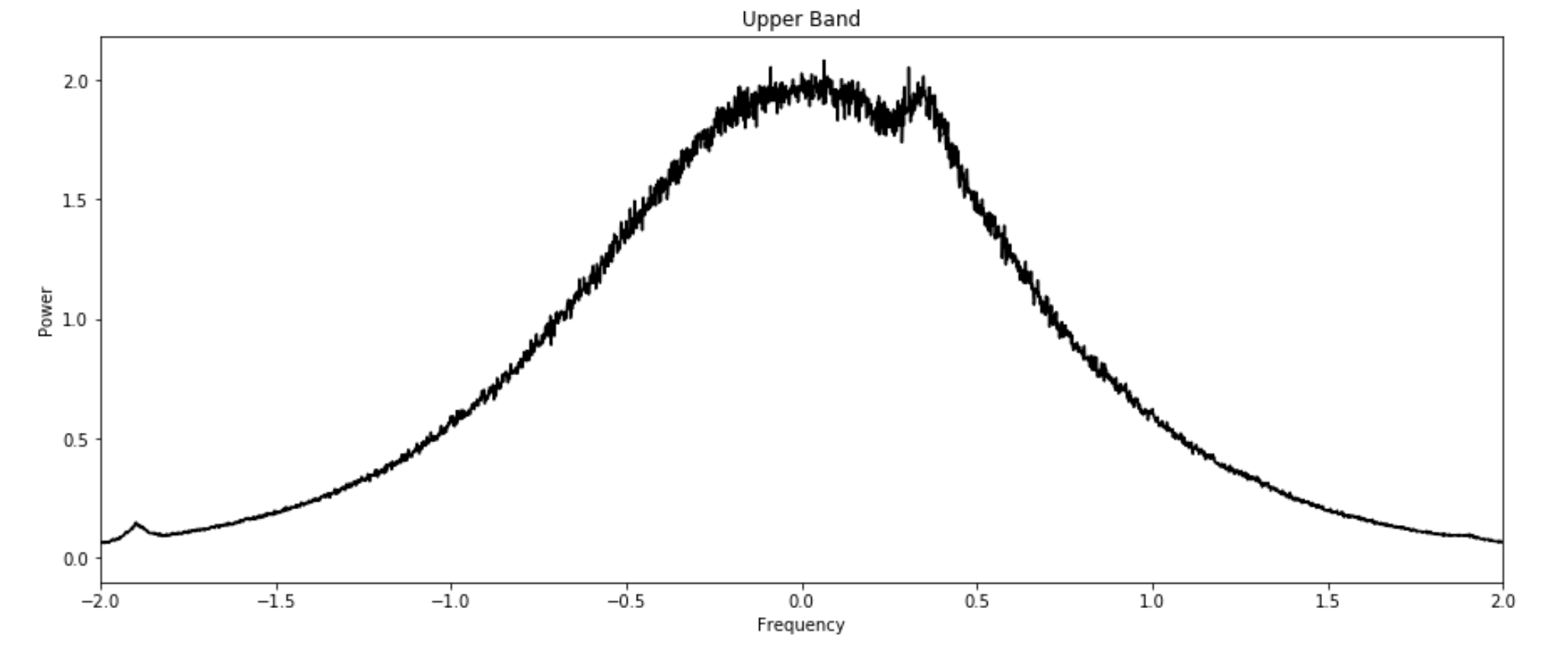
https://drive.google.com/file/d/14Cj3iQmYXgN2mMU9Tjljbq7WEnBgrhx6/view?usp=sharing

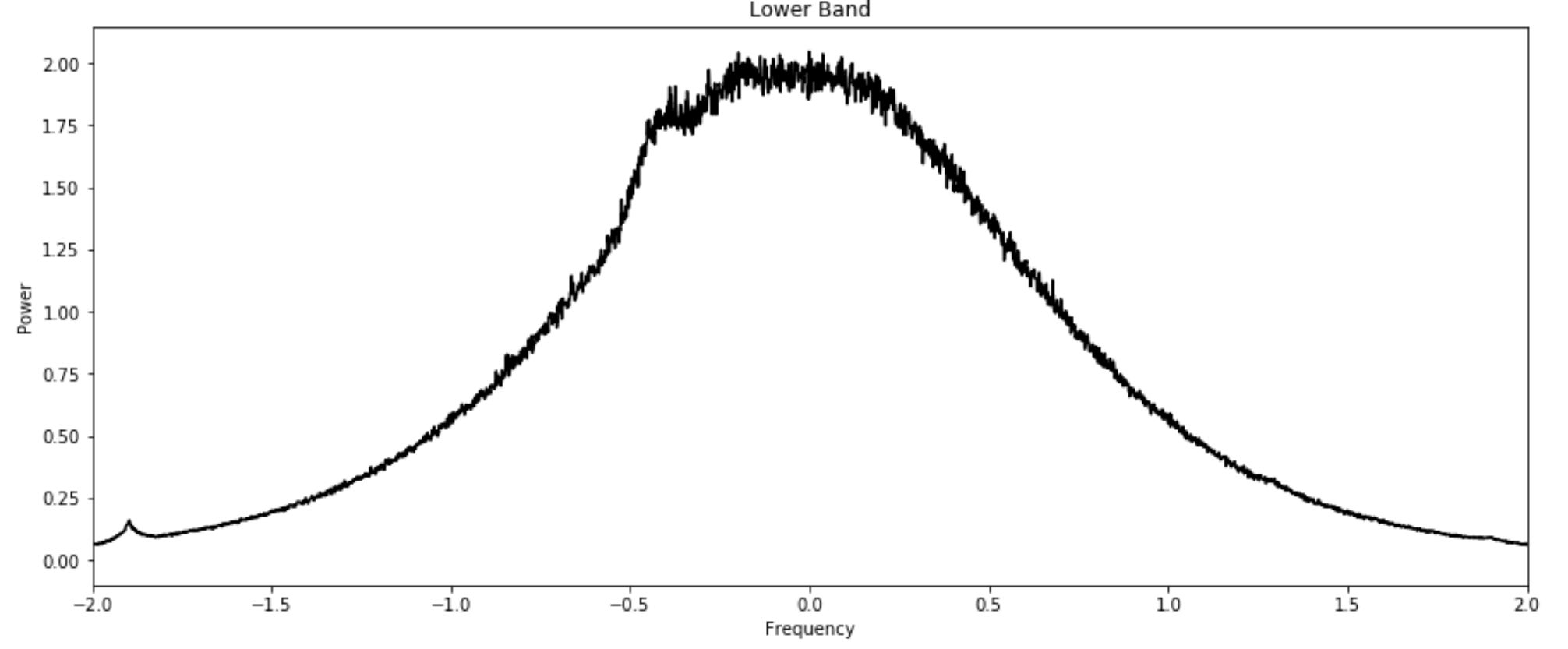
In this file, you will find that you have 4 data sets:

1. **The Baseline**: *This data is a 2-D array with dimensions (16,000, 99), meaning it is an array comprised of 99 1-D arrays of length 16,000.* The baseline is the spectra of the sky without any emission lines (peaks). In other words, it contains the background in the sky that is *not* the signal you're looking at (the 21-cm line). In an ideal scenario, the baseline would be flat, because this would mean that we are not getting any other light besides our emission lines; but in this case, the noise in our sky looks like a bell curve.



1. **The Frequency Array:** *This data is a 1-D array of length 16,000.* This frequency array serves as your x-axis in the above plot, as well as the plots below. The center of our frequency axis is labeled 0 MHz; in reality, the center is actually 1420.4 MHz, and the numbers we see are relative to 1420.4 MHz. So, for example, 0.5 MHz on our frequency axis is actually 1420.4 MHz + 0.5 MHz. -0.5 MHz on our frequency axis is 1420.4 MHz + (-0.5 MHz) (see below in blue) 
2. **The “Upper Band” and 4. “Lower Band”:** *These data sets are both 2-D arrays of dimension (16,000, 1000). 1000 1-D arrays of length 16,000.* Due to reasons that would complicate this project more than it would need to, we have two “bands”. Put as simply as possible, this just means we rigged the machinery we used to take the data to receive hydrogen’s emission peak 0.5 MHz higher (upper band) or lower (lower band) than 1420.4 MHz.

****

****

Don’t try to understand why we do this. It will hurt the brain. Just follow our instructions later.

**Problem 1: Download and Read In the Data (10 points)**

Using the package astropy.io, open the fits file ‘project2.fits’ and read its info. For each extension, print the type() of the data in that extension.

**Problem 2: Plot the Raw Data to See What’s There (10 points)**

Everytime you download data, you should look at the data to see what’s there. Using matploblib.pyplot, plot just one spectra from the baseline data set, the upper band data set, and the lower band data set. You should have three separate plots and title each one. We have done the basic structure of plotting the baseline data set in the notebook as an example. Apply aesthetics as you please.

Note: The units for frequency are MHz, and the units for the spectra are microvolts-squared [].

**Problem 3: Cleaning Up the Raw Data a Little (15 points)**

The raw data looks very strange, no? Here are three things we can do right now:

* **Take the average:** Write a function that takes all the spectra in a data set as a parameter. It should return one spectra which is an average of all the other spectra in the data set. For example, the baseline data set consists of 99 spectra. We want a new averaged spectra where each element is the average of that element across all 99 spectra in baseline. (Hint: there is a numpy function for this)
* **Shift the Array:** The data looks a bit wonky right now because of the way the data points are ordered. This data was collected by using a function called fft and the output of the fft is an array with positive values in the first half of the array and then negative values in the second half. So if you print out your frequency array you may see something like [0, ... ,6, -1, … ,-6]. To fix this apply, np.fft.fftshift() to the frequency array, online and offline spectrum and to the base shape storing them to variables of an appropriate name. You can make a function for this or apply np.fft.fftshift to each of the data.
* **Filter Out Outliers:** One other thing that is wrong with the data is that it is full of outliers (really tall, skinny peaks). To get rid of these outliers, try using a function called medfilt() from a package called scipy.signal. Read the documentation to understand how it works.

Once you’ve written these functions, try plotting your raw data again, but only after you’ve processed it through the last three steps.

**Problem 4: Write a Class (15 points)**

If this were a real lab class, it’s likely that you would have to deal with more than just one fits file like ‘project2.fits’. Instead of doing problems 1-3 for each fits file, we’re going to write a class which will automate what you did in problems 1-3. Write a File class that:

* Opens the fits file.
* Has methods get\_average\_spectra, shift\_data, and filter\_outliers.
* Has object attributes for the cleaned up baseline spectra, upper-band spectra, and lower-band spectra.

To merge the new jupyter notebook with the old jupyter notebook, you can use the shift key to select all the cells in the new jupyter notebook and then simply copy and paste into the old jupyter notebook.

***Phase 2: Getting Rid of the Noise***

Like we talked about in class, what we want to see is our 21-cm line. However, what we get instead when we take our data is the 21-cm line *plus* noise.

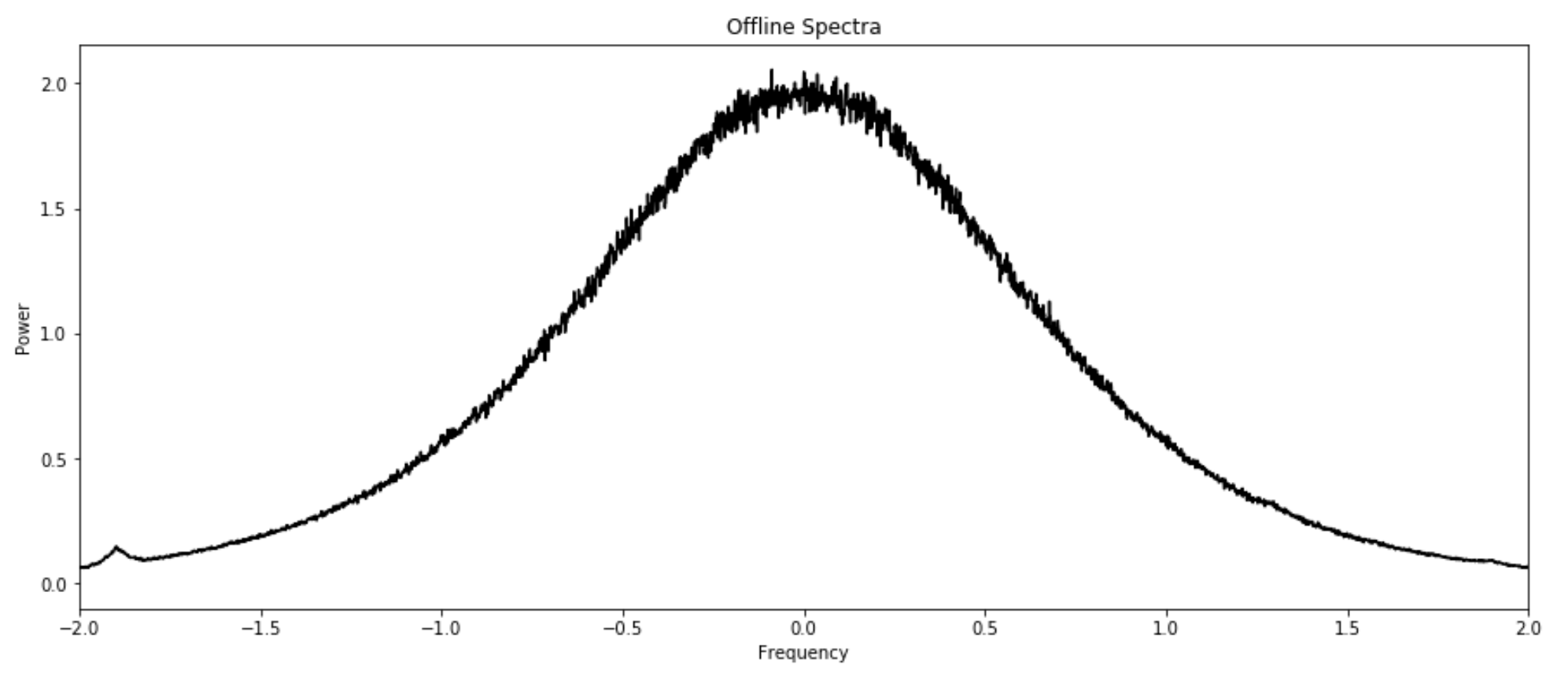
Now it’s time to explain why we have an “upper” and “lower” band. Without going into much detail, we rigged the machines to receive the 21-cm line 0.5 MHz above (upper band) and 0.5 MHz below (lower band) 1420.4 MHz. Why not just receive it right at the center, 1420.4 MHz?

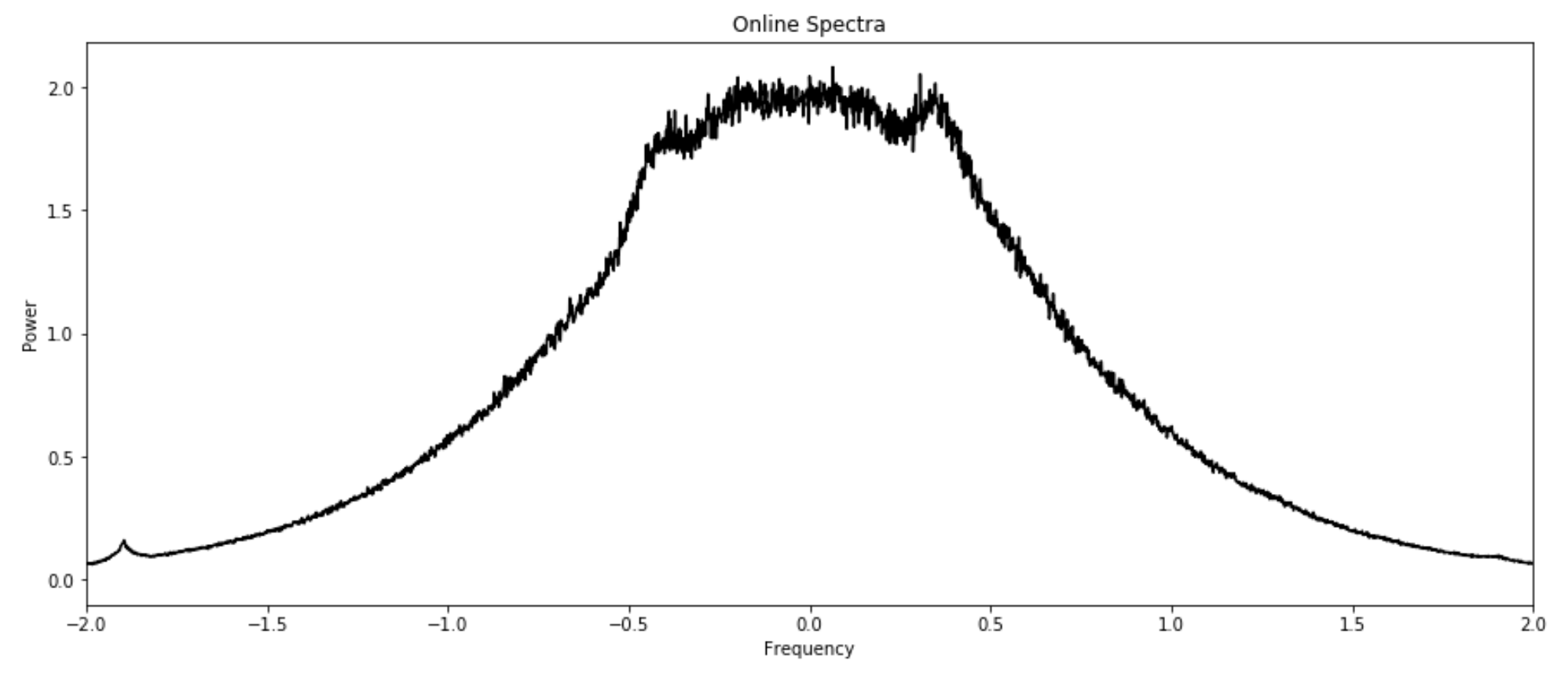
It has to do with getting rid of the noise. We showed you what the noise looks like (the baseline). In practice, however, it’s not actually easy or possible to set up the telescope just to receive the baseline. This is because no matter where we point our telescope, we will always see the 21-cm line too. We have to isolate baseline a different way.

**Problem 1: Getting the Online and Offline Spectra**

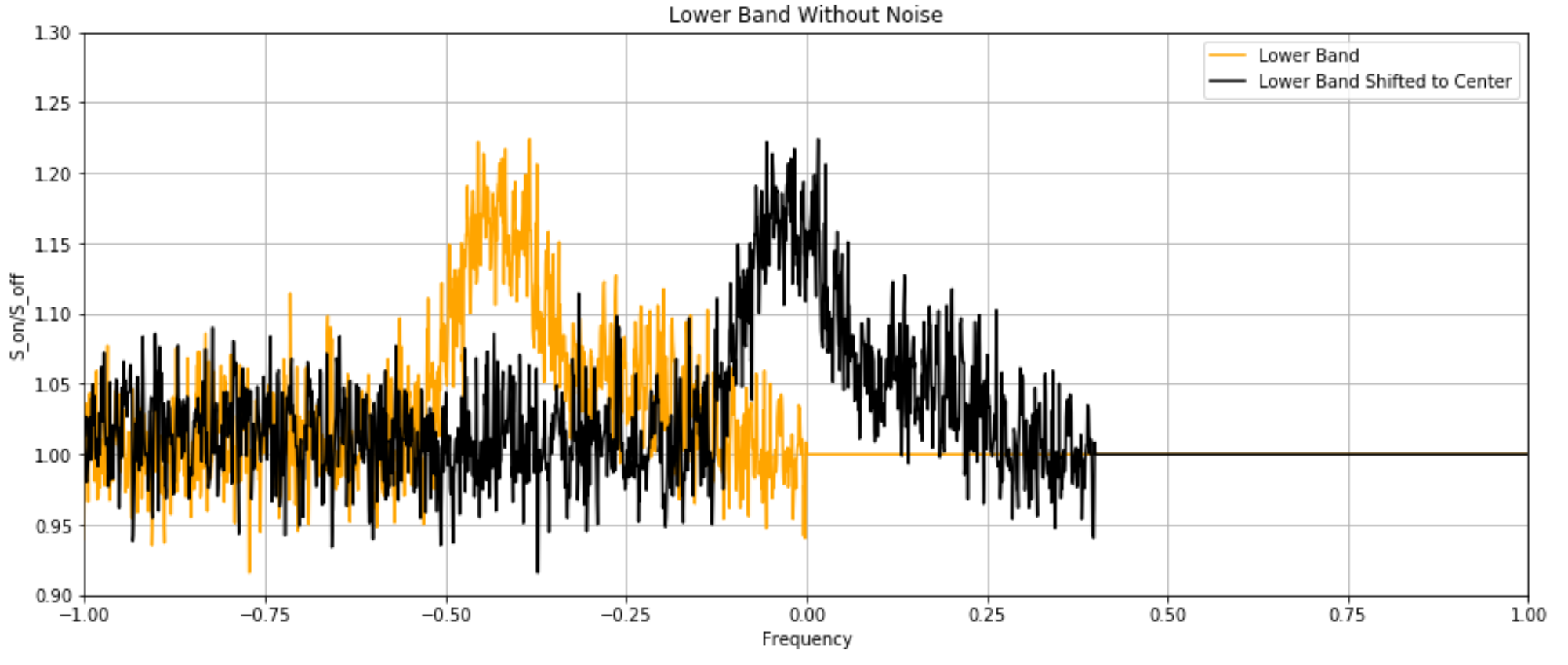
**This part is optional because you are already provided a baseline in the original fits file. If you think it would be helpful for your learning, go ahead and do this part.**

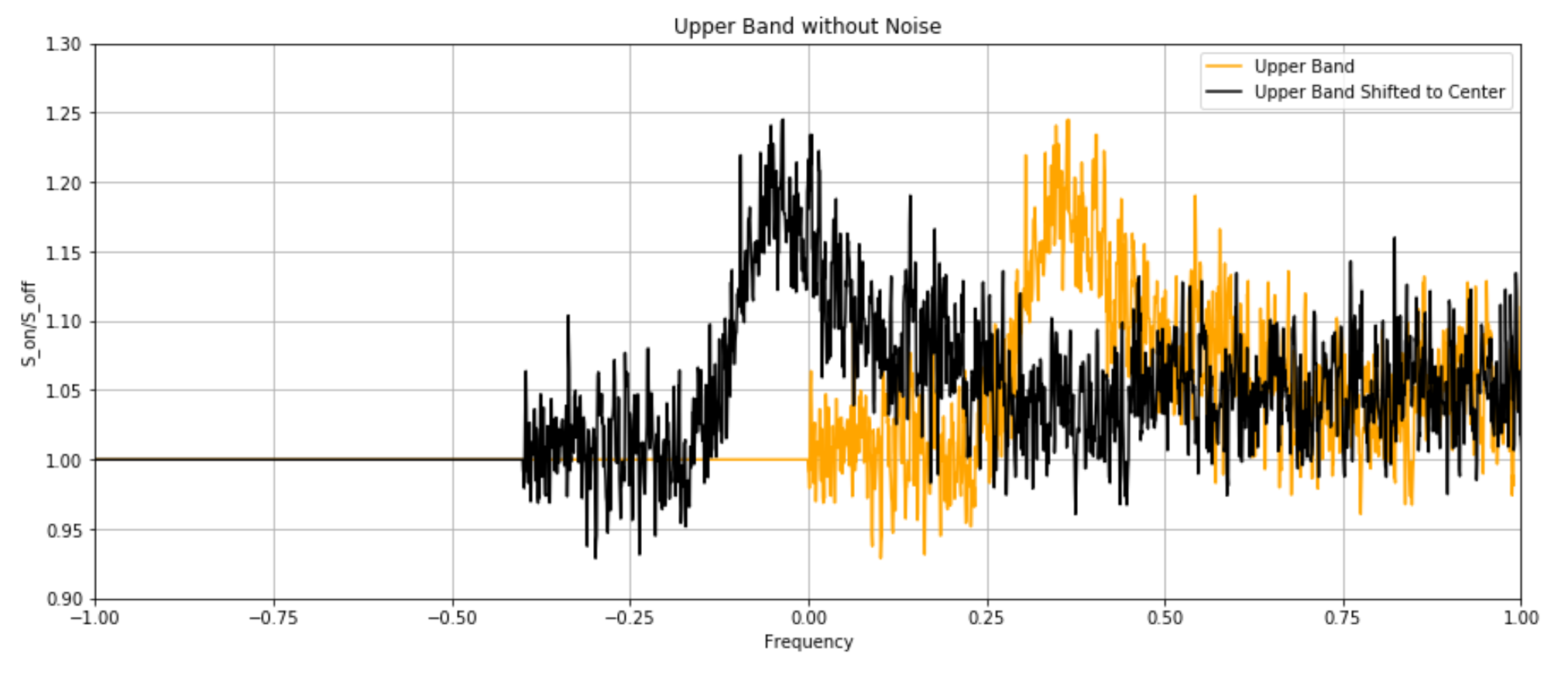
* Using np.split(), split the upper band and lower band spectra into four pieces: upper band with no peak, upper band with peak, lower band with no peak, and lower band with peak.
* Put the pieces with no peaks together (called the offline spectra) and the pieces with peaks together (called the online spectra).
* Voila, you isolated the noise (offline spectra)! Try dividing the online spectra by the offline spectra and see what happens.

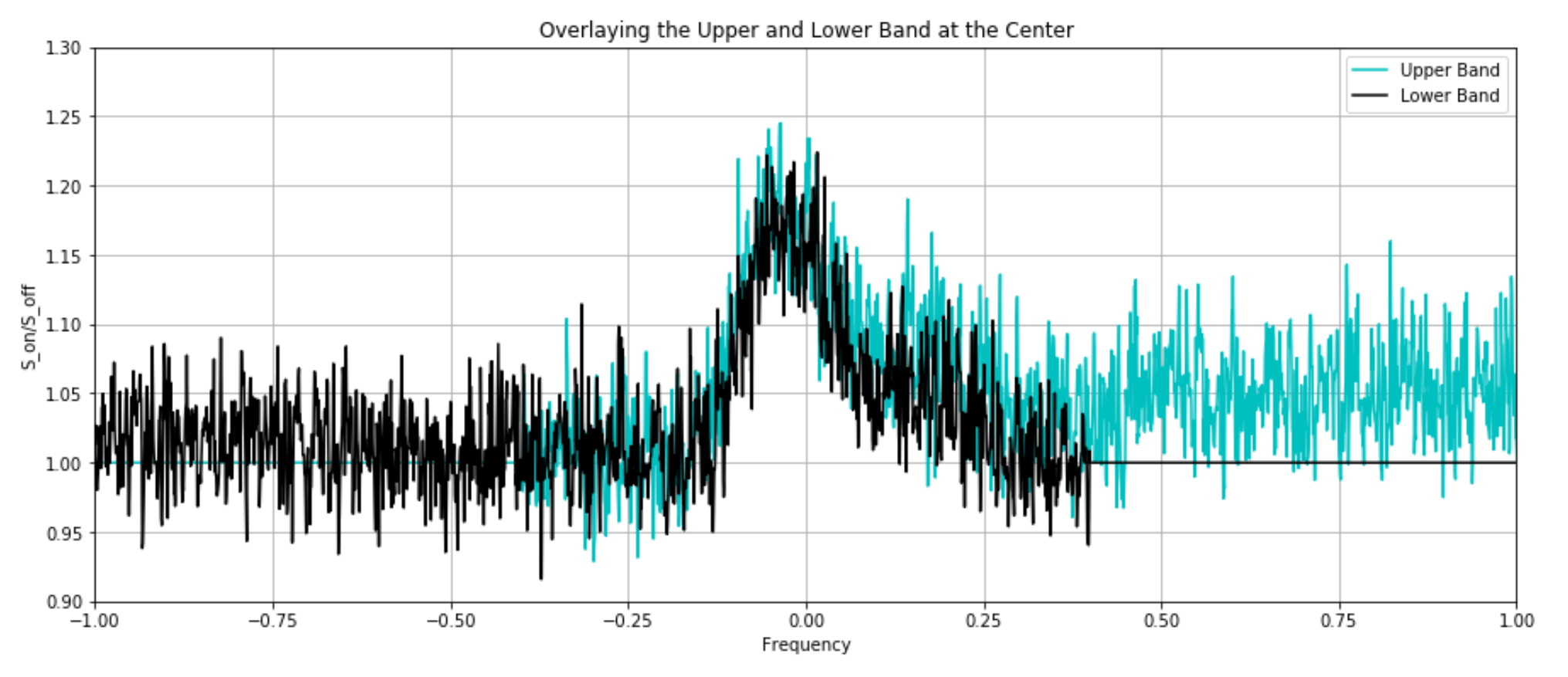




**Problem 2: Combining the Upper and Lower Band at the Center (15)**

****

****

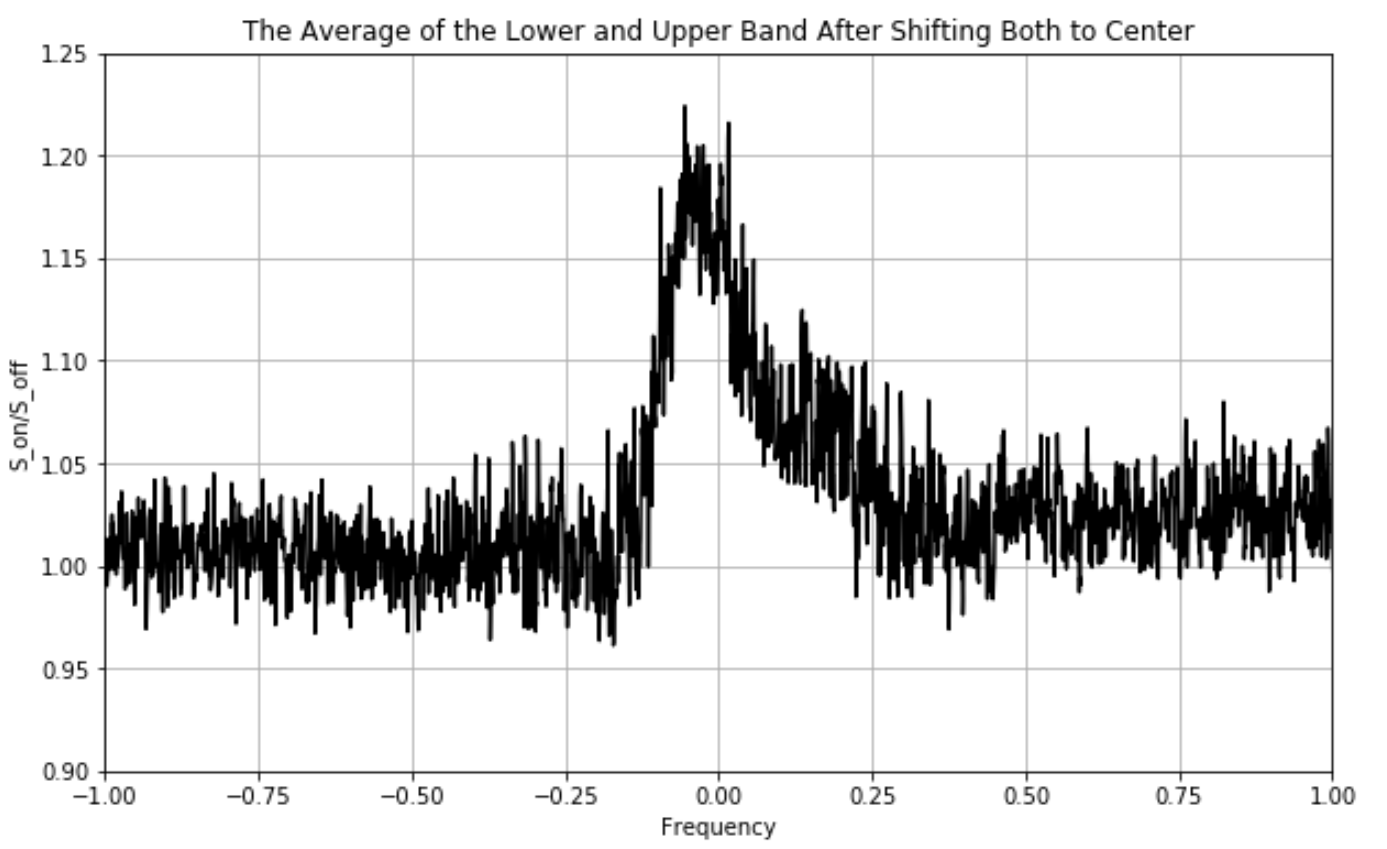
****

It’s time to look at our peak where it’s supposed to be, which is at the center (0 MHz). This means that we have to take our peak in the upper band and move it from 0.5 MHz to 0 MHz, and vice versa for the lower band.

In this project, we have provided some functions that you may use or fill in, but you are not limited to these. You have the creative freedom to search up any resources online that will help you accomplish this task of shifting arrays over.

**Problem 3: Averaging the Peak at the Center (10 points)**

When you are able to shift the upper band peak and lower band peak to the center, try averaging the signals together to get something that looks like this!



***Phase 3: Science!***

**Problem 1:**