# Dark Side Of The Dishy

## Task

* Category: Can’t Stop the Signal, Mal
* Points: 257
* Solves: 8

Description:

Phased arrays are a pathway to many abilities some consider to be unnatural.

Attachements:

* readme.txt
* antennas.txt
* Receiver\_{0-8}.bin

Readme:

Im sending you the flag and only the flag….but its being jammed….please recover.

Symbol Table: [(1+1j), (-1+1j), (-1-1j), (1-1j)]

Symbol rate is 1/2 times the sample rate

All flags follow format flag{YourFlagIsABunchOfAsciiHere!!!}

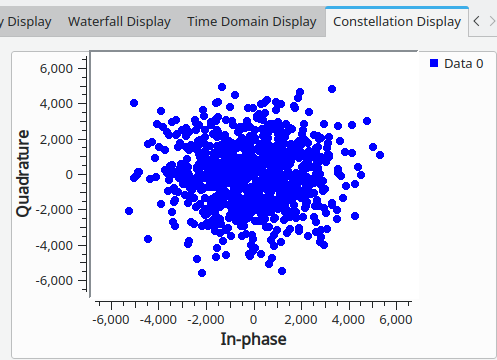
## Solution

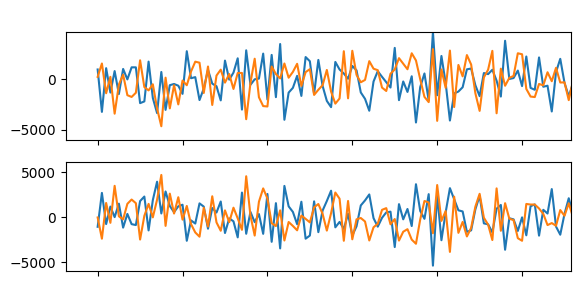
In this task we are given 9 receiver files, each 7.1KiB in size. Similar to many other signal processing challanges, they contain complex IQ samples. These can easily be read in GNURadio or numpy using the np.fromfile('Receiver\_0.bin', np.complex64).

From the antennas file we know the 9 receivers are placed in a 3x3 grid with sidelength ~60cm. It also tells us that we are transmitting at 500MHz, which has a wavelength of ~60cm, exactly the same as the grid size.

The Readme tells us the flag is being jammed. We thus likely have two senders, one jammer and one flag-sender, at different locations. The Task is then to use the delay information inferred by looking at different receivers to filter out the jammer and receive the flag sender cleanly.

Indeed, looking at a constellation plot in GNURadio shows that the original signal of an individual receiver is just noise.



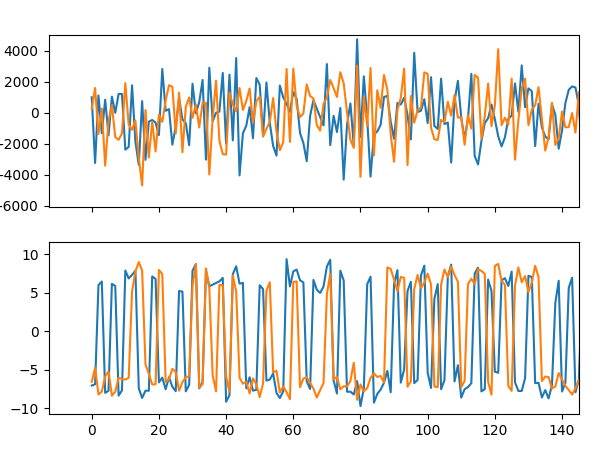
Plotting 2 receivers over time shows us that they are somewhat correlated. Not easy to see, but spikes in the top graph (receiver0) also have spikes in the bottom graph (receiver1): 

Now, accurately implementing a direction-based filter is somewhat hard. So we tried the “easy” approach first and just poked at the data a bit to see what falls out. At first, we tried to cross-correlate the receivers, which didn’t work out so well since the delay was too small. Once we noticed that our antenna array is only 1 wavelength in size, we figured the delay would be only in the phase of the signal.

To get that phase delay, we simply calculate np.mean(np.angle(receiverX / receiver0)) for each receiver. As the jammer is way stronger than the flag sender, this gives us the phase delay of the jammer at each receiver compared to receiver0. We can then correct for the phase-delay, by computing receiverX \*= np.exp(-1j \* mean\_phase). Now we have 9 receivers which receive the jammer simultaneously, but are still phase-incorrect for the flag-sender.

By computing the average over all 9 receivers we get a good guess for the jammer signal. Subtracting that mean jammer signal from receiver0 gives a clean flag signal.

Plotting the mean jammer signal (top) and clean flag signal (bottom) gives us:



Now the only thing left is decoding the flag signal. The readme tells us it is 4QAM, and the signal is clean enough that we can simply use a “is I/Q greater or less than 0” decoder to map all samples into the 4 quadrants of the I/Q plane (each is one symbol). From the readme we know that we have two samples per symbol.

We know the flag starts with flag, which allows us to find the symbol to bit mapping by hand:

flag{yankee221281foxtrot4:GG0TlpIcPdbFX03jUrOdlOtFBpRRIRywbqxZSh2aAIHz7thDd9MZoCI5Of3aER1ZYViacsU6LUWm\_fdBG-Di49Y}