NREIP CERF LAB INTERNSHIP 2016

DANIEL HALLMAN

Contents

1. Noise Floor and SNR 2
2. Angle Resolution Error 4
3. Inductance Patterns 6
4. Flux Bias Line Properties 10
4.1. Radiation 10
4.2. Analysis of Flux 14
4.3. Numerical Calculation of Flux 15
4.4. Superconductor Impedance 16
5. Solenoid Simulations 19
5.1. Regular Solenoids 19
5.2. Solenoid Quadrupoles 20
6. Direction-Finding 23
6.1. MUSIC performance on real SQUID output 23
6.2. MUSIC performance in a SQUIDy environment 27
6.3. Uni-Vector-Sensor ESPRIT 31
6.4. DR-MUSIC Method with Frequency Selection 32
References 39

Date: September 23, 2016.

1. Noise Floor and SNR

Relevant Programs: `SNRcode1.m`

One of the first things I did was try and estimate the noise floor of a SQUID signal I was given. This happened a long time and I did not document my findings, so this explanation will be a little bit hazy.

To my knowledge, calculating the power spectral density (PSD) of a given signal is the key to finding its noise floor and SNR. Mathematically, PSD is the Fourier Transform of the signal's autocorrelation function (the math makes sense if you look up PSD on Wikipedia). In simpler terms, the PSD is very similar to a Fourier Transform, except the resulting coefficients correspond to the power of a given frequency (as opposed to the amplitude). This is easily accomplished in MATLAB via a useful function called `pwelch`. In `SNRcode1.m`, a Blackman-Harris filter is used and a few other specific parameters are inputted into the `pwelch` function to create a "better" PSD. I regret to say that I have long since forgotten the references I used in motivating these decisions (a few websites played key roles in how I found all of this out).

With the `pwelch` function, the PSD is calculated along with a vector of frequencies. This vector of frequencies is exactly what you expect - each entry is the frequency that corresponds with the power coefficient that has been calculated in the PSD itself. The PSD and the vector of frequencies follow a certain resolution that determined by frequency bins. A frequency bin is simply an interval (or bin) full of frequencies. To calculate the power for every single frequency of a signal (say, between 1MHz and 100GHz) would be way too much work. So instead, the bandwidth of the signal is partitioned into equal bins, so that each coefficient in the PSD actually corresponds to the total summed power in an entire bin. The difference between any two consecutive entries in the vector of frequencies calculated from `pwelch` will give the width of the frequency bins. The frequency bins dictate the resolution of the PSD, and this resolution can be controlled in different ways (one of which is to just make the signal longer or shorter).

After the PSD has been calculated with `pwelch`, finding the noise floor and SNR of the given signal is fairly simple. In the example below, the data given is portraying a transmitted signal with a frequency of 29.9 MHz. This means that in order to calculate the power of the received signal from the data, the frequency bin that contains the frequency 29,900,000 (which corresponds to a specific entry in the vector of frequencies) has a corresponding entry in the PSD which states the total power of that frequency. This can be done in a few ways and is clearly stated in `SNRcode1.m`. To find the noise power, all the coefficients in the PSD are added up EXCEPT the coefficient that corresponds to the transmitted signal frequency. The units of the PSD are given in V^2/Hz, and so typically the SNR and the noise floor are calculated by first summing the relevant coefficients then taking a square root. There are a few other omitted details in the process, but they can be easily understood by studying the code of `SNRcode1.m`.

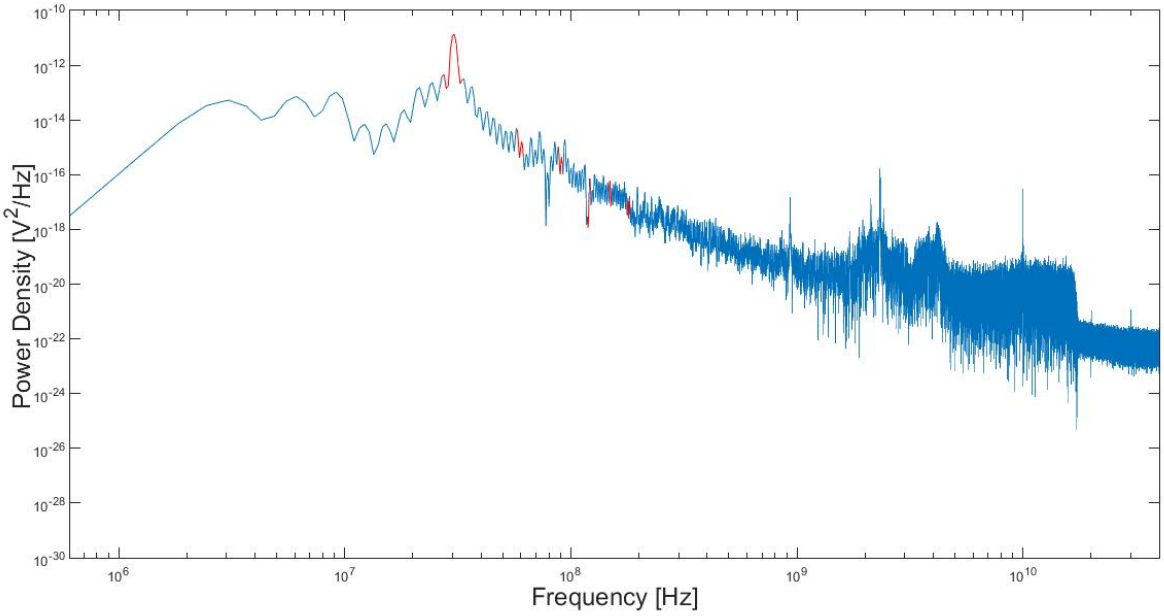

FIGURE 1.1. Power Spectral Density of SQUID data from a transmitted signal at 29.9 MHz

When running `SNRcode1.m`, figure 1.1 will be recreated (assuming the same SQUID data is used) along with estimates for the SNR, noise floor, and a few other things. In this example, we have

$$\text{Noise floor estimate} = 0.019\mu V/\sqrt{Hz}$$
$$\text{SNR} = 9.190 \text{ dB}$$
$$\text{MATLAB's SNR estimate (}\texttt{snr}\text{ function)} = 9.138 \text{ dB}$$

For most of the data I tried, `SNRcode1.m` and MATLAB's `snr` function gave roughly the same answer for the SNR which was comforting. I have not found a built-in function to find the noise floor and so for the time being I trust the calculation given by `SNRcode1.m`. Unfortunately, this data was obtained from an oscilloscope which has a noise floor that is higher than the noise floor of the SQUID. This means that the numbers given above actually correspond the oscilloscope instead of the SQUID. In order to get numbers that truly describe the SQUID, data must be obtained through a device that has a lower (or equal?) noise floor than the SQUID.

2. Angle Resolution Error

The following is a calculation for Ben's DF algorithm which involves one (or two?) SQUIDs and a Machzender interferometer. To be totally honest, I still don't really know how the algorithm is supposed to work. Regardless, I was given an equation of how the angle resolution is calculated and asked what the variance looks like if one of the terms in the equation itself varies. Fun fact: I completely misunderstood the problem at first and spent an entire week trying to solve the wrong problem.

The resolution of the estimated angle of arrival is given as

$$\begin{aligned}\theta &= \sin^{-1}\left[\left(\frac{c}{2\pi\nu d}\right)\frac{V_{MZ}}{a(\beta|\vec{S}|)^{1/2}}\right] \\ &= \sin^{-1}\left(A\frac{1}{|\vec{S}|^{1/2}}\right)\end{aligned}$$

Where A represents all the constants we don't need to worry about.
Since $|\vec{S}|$ has a certain level of uncertainty, we can instead represent $|\vec{S}|$ as $|\vec{S}|+\delta|\vec{S}|$. This invokes some uncertainty into θ. From [1], the uncertainty of θ can be calculated as

$$\begin{aligned}\delta\theta &= \left|\frac{d\theta}{d|\vec{S}|}\right|\delta|\vec{S}| \\ &= \frac{A}{2}\frac{1}{\sqrt{|\vec{S}|-A^2}}\frac{\delta|\vec{S}|}{|\vec{S}|} \text{ (after some rearranging)}\end{aligned}$$

From the calculation of $\delta\theta$, it is obvious that it will be favorable to increase $|\vec{S}|$ and decrease $\delta|\vec{S}|$ as much as possible.

We can fix $A=1$ and make a 2-dimensional plot of $\delta\theta$ against varying values of $|\vec{S}|$ and $\delta|\vec{S}|$. This seems like nice idea, but instead we can easily say that, in general, $\delta\theta \leq 1$ is considered satisfactory. This expectation gives us the following relation (again assuming $A=1$)

$$\begin{aligned}\frac{1}{2}\frac{1}{\sqrt{|\vec{S}|-1}}\frac{\delta|\vec{S}|}{|\vec{S}|} &\leq 1 \\ \rightarrow \frac{(\delta|\vec{S}|)^2}{(|\vec{S}|)^3-(|\vec{S}|)^2} &\leq 4 \\ \rightarrow \delta|\vec{S}| &\leq 2(|\vec{S}|^3-|\vec{S}|^2)^{1/2}\end{aligned}$$

Inserting A back into the inequality gives us

$$\delta|\vec{S}| \leq \frac{2(|\vec{S}|^3-A|\vec{S}|^2)^{1/2}}{A}$$

If instead you would like to know $\delta|\vec{S}|$ given $\delta\theta \leq C$, then use

$$\delta|\vec{S}| \leq \frac{2C(|\vec{S}|^3-A|\vec{S}|^2)^{1/2}}{A}$$

If we would like to look at the relative error, use

$$\frac{\delta\theta}{|\theta|} = \frac{1}{2}\frac{1}{\sqrt{|\vec{S}|-1}}\frac{1}{\sin^{-1}(|\vec{S}|^{-1/2})}\frac{\delta|\vec{S}|}{|\vec{S}|} \text{ , (A=1 here).} \tag{2.1}$$

Putting A back into the equation then turns 2.1 into

$$\frac{\delta\theta}{|\theta|} = \frac{A}{2}\frac{1}{\sqrt{|\vec{S}|-A^2}}\frac{1}{\sin^{-1}(A|\vec{S}|^{-1/2})}\frac{\delta|\vec{S}|}{|\vec{S}|}$$

Here is a table with some values. $\delta|\vec{S}|$ and A are kept constant at 1. $|\vec{S}|$ continues to increase which means the relative error in $|\vec{S}|$ decreases by an order of magnitude each time.

$\delta\theta/\|\theta\|$	S
9.04e-4	10
8.75e-5	100
8.72e-6	1000
8.72e-7	10000
8.72e-8	100000

3. Inductance Patterns

Relevant Programs: `betamaker.m, Testing.m`

The SQUID array consists of thousands of individual SQUIDs connected in a giant rectangle. When receiving a signal, the array outputs a voltage that is created by all of the individual SQUIDs working together. The individual SQUIDs consist of many different sizes and are currently distributed about the array in a totally random fashion. This is (I think) meant to effectively help cancel out persistent errors or biases and somehow creates the characteristic "anti-peak" that is crucial in the process of making an effective SQUID antenna.

Ben had an idea that the distribution of the distinctly sized SQUID loops might benefit the antenna performance if the sizes were placed in a more structured way. For example, perhaps putting all the big loops on the right-hand side of the array and then placing all the smallest loops adjacent to the large loops would create some kind of favorable effect and improve linearity in the voltage output. Or alternatively, put all the large loops in the center of the array and surround them with smaller loops or another strange combination. Basically, Ben needed a way to modify Susan's code in a way that would allow for any possible pattern of loops to be placed in the array to see what happens and spark a new innovation.

My solution to this problem was to create a template system. Instead of having a ton of parameters and predetermined patterns to tune and choose between, I opted to create a system in which the user designs a template themselves, then the program `betamaker.m` takes the template and scales it up to the large dimensions that are needed in the simulation. An example will be very useful in this situation.

Suppose the SQUID array to be simulated consists of 400 SQUID loops in series and 100 SQUID loops in parallel. In MATLAB notation, this corresponds to a 400×100 matrix of loops. If the user wishes to put all the large loops on the right-hand side with smaller loops next to them and all the leftover loops filling out the rest, they would input Susan's randomly generated vector of inductances (called `beta` in the code, I believe), along with this template into `betamaker.m`

$$\text{Template1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix}.$$

Selecting the 'expand' command on `betamaker.m` will stretch this template across 400 rows and 100 columns. The '2' on the right says "put large loops here" and the '1' says "put small loops here". The zeros everywhere else just mean that everything left over are placed in those positions. `betamaker.m` counts the numbers in the template and partitions the distribution from the inputted `beta` then rearranges everything to fit the specifications of the template. Note that this template is only a 1×10, while the desired SQUID array is 400×100. This means that each entry in the template will be repeated by a factor of 10 horizontally and 400 vertically. If a greater resolution is desired (i.e. less stretching), simply make the template larger which needs less stretching. Note that this template

$$\text{Template2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix}.$$

will create the exact same result as Template1. While this

$$\text{Template2} = \begin{bmatrix} 0 & 0 & 0 & 1 & 2 \end{bmatrix}.$$

will accomplish the same general pattern, but at a much lower resolution. It is important to note that the individual dimensions of the template MUST be divisors of the individual dimensions of the SQUID array itself, respectively (e.g. a template of size 10×17 will not work, but 10×20 will).

`betamaker.m` has a 'tesselate' command as well. Instead of expanding a given template to fit the SQUID array, 'tesselate' will repeat the template over and over across the SQUID array (another reason

why the template dimensions must divide the SQUID array dimensions). Here is a final example: suppose the SQUID array is a 6×10 and you enter this template

$$\text{Template2} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

then the result will be

$$\texttt{betamaker.m} \rightarrow (\text{`Expand'}) \longrightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\texttt{betamaker.m} \rightarrow (\text{`Tesselate'}) \longrightarrow \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

The output will be a vector that can be used in Susan's code just like the original vector was used before. Finally, here are some results on a 60×15 SQUID array. Figures on top are visual representations of the distribution. Red = large loops and blue = small loops. The figures on bottom are the voltage responses of the SQUID arrays against a linearly increasing amount of flux. The bottom axis lists 'time' as the x-axis but should read 'flux' instead.

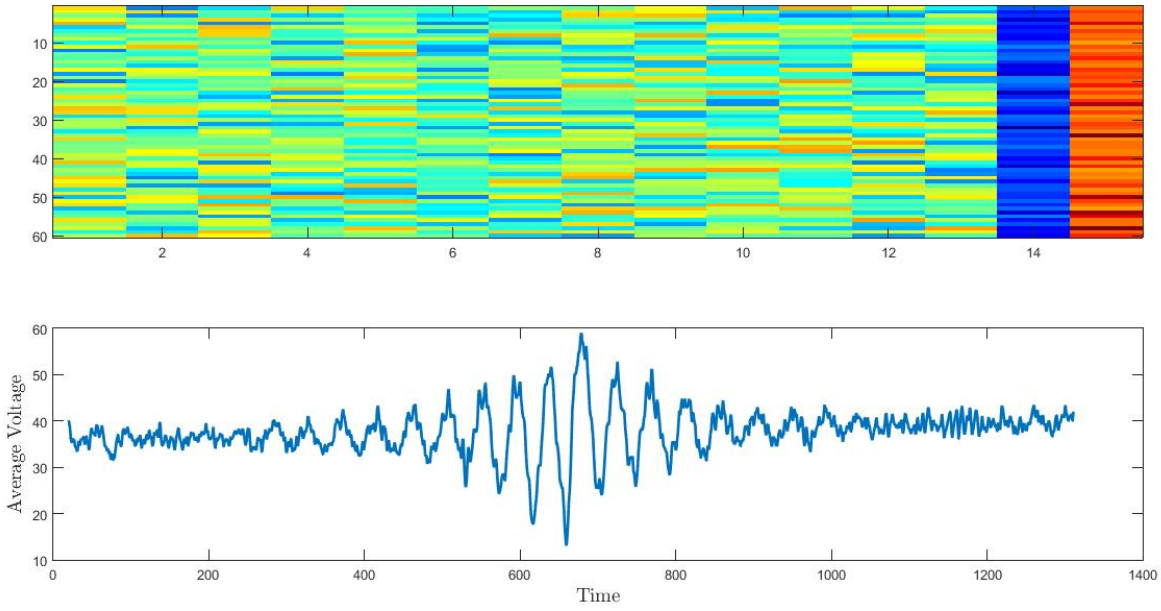

FIGURE 3.1. Large and small loops on the right

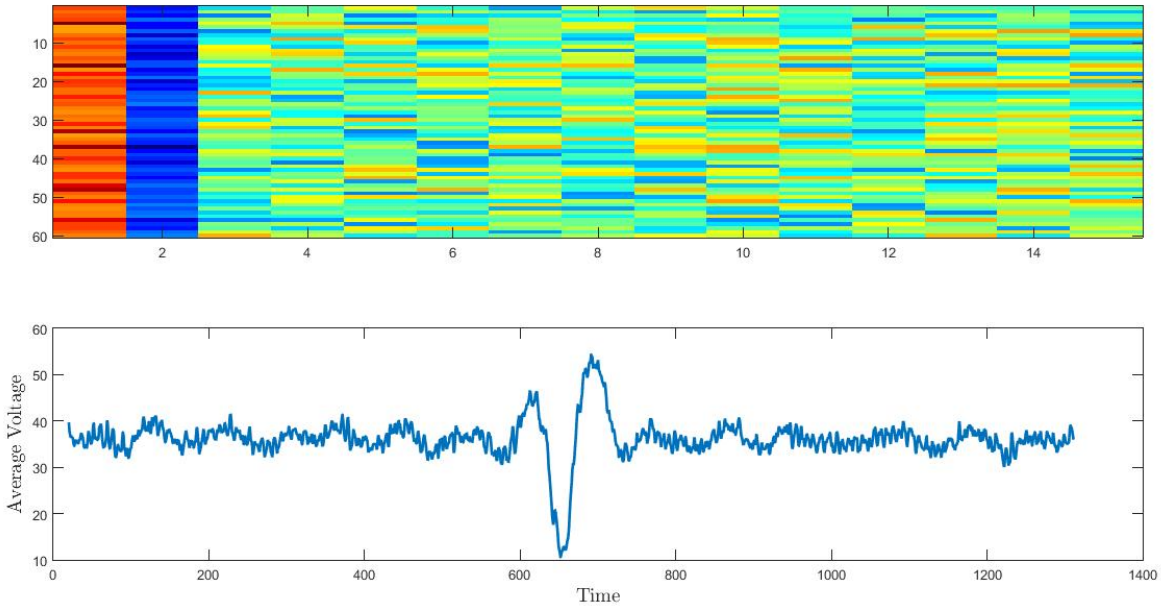

FIGURE 3.2. Large and small loops on the left

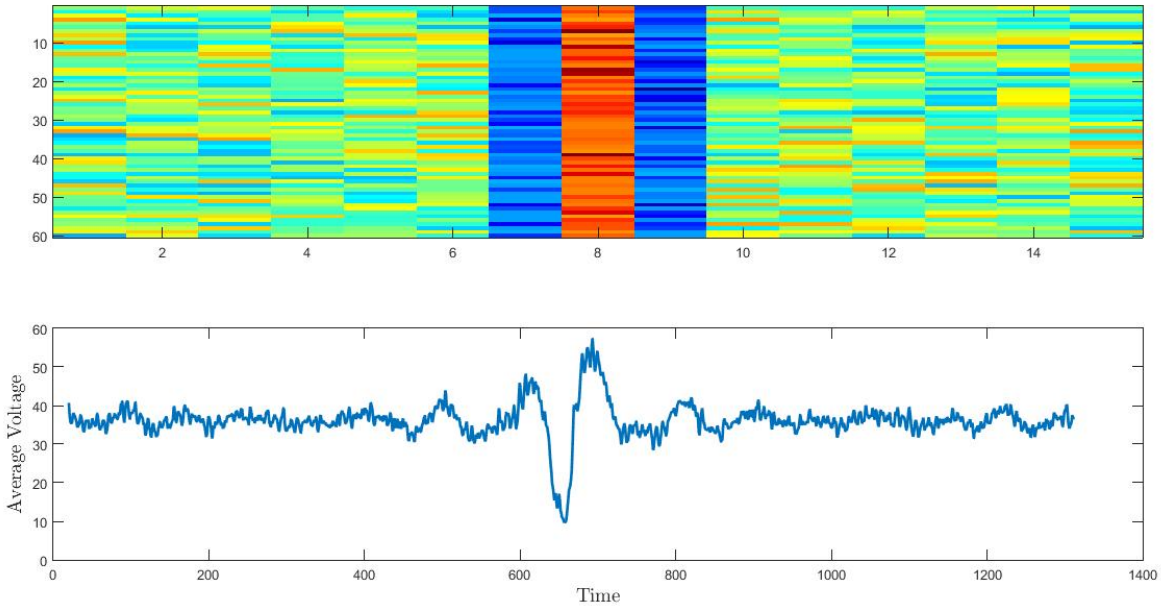

FIGURE 3.3. Large and small loops in the middle

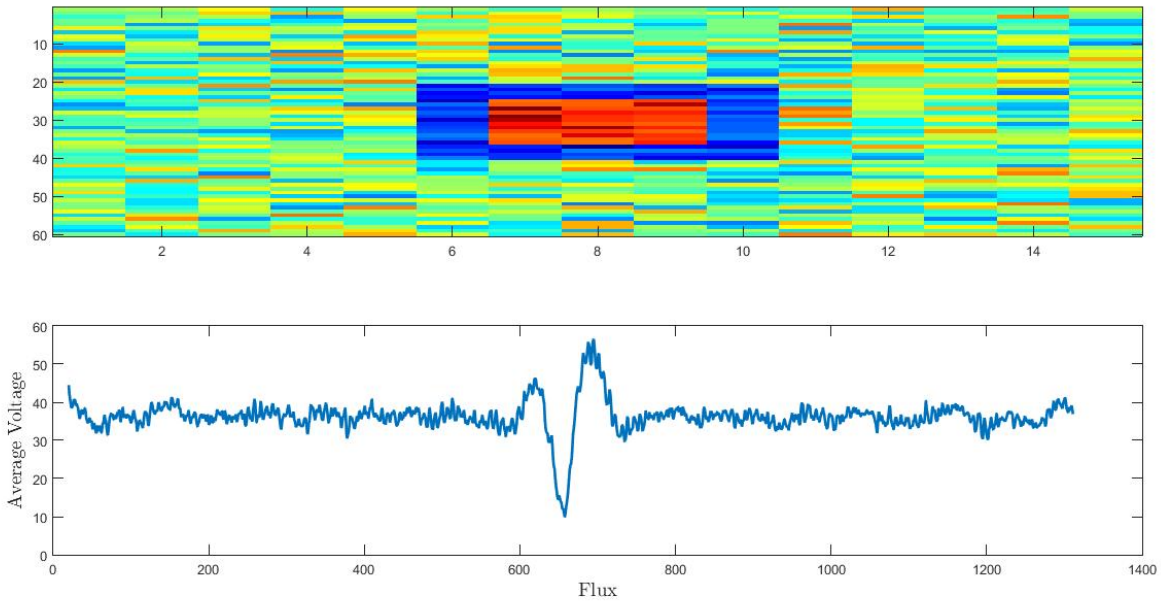

FIGURE 3.4. Large and small loops in the center

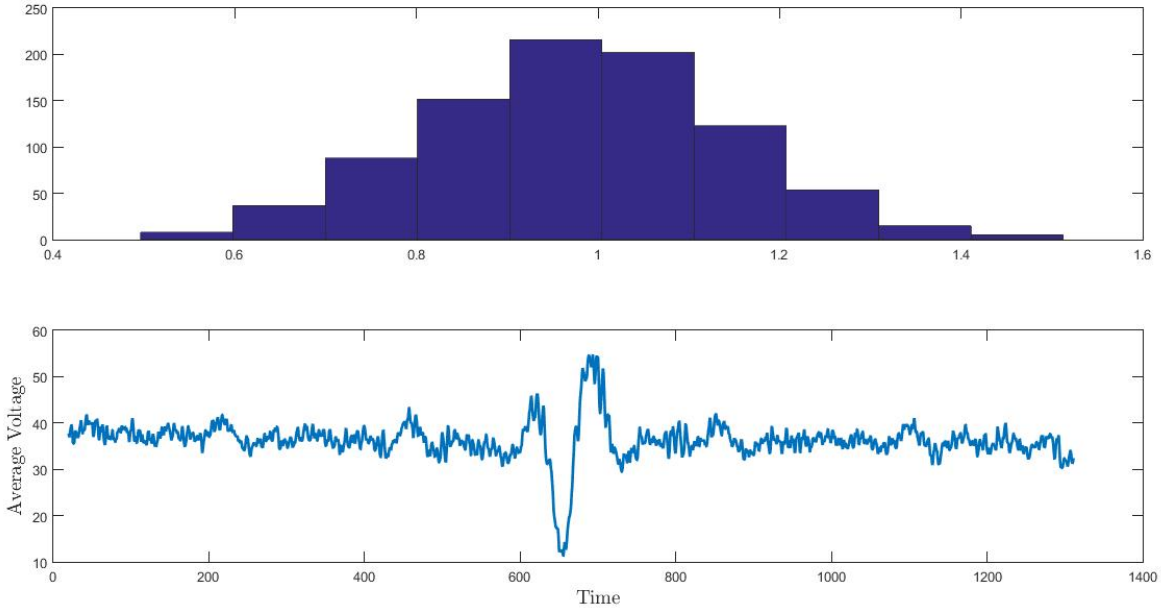

FIGURE 3.5. No pattern. Top graph is a histogram of the size distribution.

4. Flux Bias Line Properties

The SQUID array requires a precise amount of flux to pierce it during its operation. To accomplish this, a superconducting rectangular wire (referred to as a flux bias line here) supplied with DC current surrounds the SQUID array. This step is very necessary, as any shift in the array changes the flux from constant ambient magnetic fields. Controlling the current in the flux bias line allows for a consistent amount of field to go through the SQUID array. Any additional fluctuations in flux can then be entirely attributed to the RF target.

4.1. **Radiation.**

As with most things in real life, good ideas like the flux bias line become plagued by annoying complications. While the wire supplies a constant flux to flow through the SQUID, it itself electrically reacts with the RF signal, causing an unwanted EMF and ultimately induces its own magnetic field and gets picked up by the SQUID array. This unwanted field lags behind the original RF, and causes an "echo" to be heard by the array. Unfortunately, It seems that we may be stuck with the flux bias line for the time being. As troublesome as it is, can we use this property to our advantage? An idea is that we can run some sort of smart current through the flux bias line and increase the directivity of the array. This section attempts to look at this possibility. Sadly, the superconducting nature of the flux bias line is not taken into account until section 4.4 which largely makes this section obsolete (i.e. worthless). Regardless, you should go ahead and read it anyway because I worked really hard on it.

Here, we investigate the properties of the SQUID/flux bias line setup by modeling two nested square loop antennas with a centered outer loop and an off-center inner loop. A small loop surrounded by a large loop (even if both centered) will interfere with each other if the sizes or size difference between them is significant with respect to the wavelength being measured. Derivations of electromagnetic fields in loop antennas always assume that the radius of the antenna is very small compared to the wavelength of interest. This assumption makes many complicated terms go away and the derivation becomes much easier. This is NOT the case in the SQUID/flux bias line setup because the dimensions are large with respect to the wavelengths that they are sensing. As a first step, however, we assume that both antennas are very small (infinitesimal, in fact). This makes analysis much easier. Large antennas will be investigated after.

Consider an antenna emitting a signal, and you measure that signal to be $\sin(\omega t)$. Repeating the experiment, you then measure a signal of $\sin(\omega(t+a))$. What happened? If only distance is considered, then the antenna apparently moved some distance away from you which is what caused the phase shift. How is this any different from the antenna just deciding stay put and emit the same signal with the given phase difference? There is no difference, and I will try and show that analytically below.

For an electrically small loop antenna, shape is irrelevant. If we would like to analyze square loops, then we can easily treat them as circular loops and utilize well-established functions without any work (I did not originally copy such functions, I painstakingly calculated the vector-potential of a square loop and found it to be exactly the same as a circular loop). The only thing that sets small loop antennas apart are their areas. For square loop with side lengths a_0, b_0 and current $\mathbf{I}_0$, the potential function is given as

$$\begin{aligned}\mathbf{A}(x,y,z) &= \frac{\mu}{4\pi}\int_C \mathbf{I}_0 \frac{e^{-jkR}}{R} dl' \\ &= \frac{jk\mu I_0 a_0 b_0}{4\pi r} e^{-jkr}\sin\theta\ \boldsymbol{\phi}\end{aligned}$$

Here, C is the rectangle itself, and $\mathbf{R}$ is the vector that traces a point on the rectangle to the point (x, y, z). The $\mathbf{H}$ and $\mathbf{E}$ far-fields are calculated from this (via spherical curls, refer to [2]) to be

$$\begin{aligned} H_r &= 0 \\ H_\theta &= -\frac{I_0 a_0 b_0 k^2 e^{-jkr}}{4\pi r}\sin\theta \\ H_\phi &= 0 \\ E_r &= 0 \\ E_\theta &= 0 \\ E_\phi &= \frac{I_0 a_0 b_0 k^3 e^{-jkr}}{4\pi r \epsilon\omega}\sin\theta \end{aligned}$$

Now, we take the infinitesimal loop and move it in the x-direction by an amount d. Far from the source, this is nearly equivalent to changing the distance from the antenna to the reference point r to $r + d\sin\theta\cos\phi$. We replace this expression for r into the potential function and perform the same calculations. Note that this change only occurs in the phase term for r and not in the distance term. This is because the phase is the only term that is significantly affected. Since this is supposed to be the off-centered inner antenna, we give it new current and dimensions $I_1,\ a_1$ and b_1.

$$\mathbf{A} = \frac{jk\mu I_1 a_1 b_1}{4\pi r} e^{-jk(r + d\sin\theta\cos\phi)}\sin\theta\ \boldsymbol{\phi}$$

From this we obtain the $\mathbf{E}$ and $\mathbf{H}$ far-fields for the inner loop

$$\begin{aligned} H_r &\approx 0 \\ H_\theta &= -\frac{I_1 a_1 b_1 k^2 e^{-jkr} e^{-jkd\sin\theta\cos\phi}}{4\pi r}\sin\theta \\ H_\phi &\approx 0 \\ E_r &\approx 0 \\ E_\theta &\approx 0 \\ E_\phi &\approx \frac{I_1 a_1 b_1 k^3 e^{-jkr} e^{-jkd\sin\theta\cos\phi}}{4\pi r \epsilon\omega}\sin\theta \end{aligned}$$

Note that these are the *far*-fields, i.e., all terms of order larger than $O(1/r)$ have been eliminated. If this assumption is not made then the fields become extremely long and are too cumbersome to conduct effective analysis. Most terms are nonzero but they are not taken into consideration here. Something interesting has happened here. Notice that the only difference between these equations and the ones before is a factor of $e^{-jkd\sin\theta\cos\phi}$. The simplicity of this difference will provide more use later on, but keep in mind that the effect of this linear translation does indeed have more intricate effects when considering near and intermediate fields, so this"obviousness" really only reveals itself in the far-field. We now calculate the radiation intensity. This can be calculated using

$$U = \frac{r^2}{2\eta}\left(|E_\theta|^2 + |E_\phi|^2\right) = \frac{r^2}{2\eta}|E_\phi|^2$$

where again our far-field case has made things easier. Both loops are acting as one antenna, so we add both fields together to get

$$|E_\phi|^2 = \left|\frac{k^3 e^{-jkr}\sin\theta}{4\pi r\epsilon\omega}(I_0 a_0 b_0 + I_1 a_1 b_1 e^{-jkd\sin\theta\cos\phi})\right|^2$$

$$\longrightarrow U = \frac{\eta k^4 \sin^2\theta}{32\pi^2}[(I_0 a_0 b_0)^2 + 2I_0 a_0 b_0 I_1 a_1 b_1\cos(kd\sin\theta\cos\phi) + (I_1 a_1 b_1)^2]$$

We wish to know what happens if a phase-shift is placed into the current of the outer loop. Thus we replace I_0 with $I_0 e^{j\alpha}$.

$$\begin{aligned}
|E_\phi|^2 &= \left| \frac{k^3 e^{-jkr} \sin\theta}{4\pi r \epsilon \omega} (I_0 a_0 b_0 e^{j\alpha} + I_1 a_1 b_1 e^{-jkd \sin\theta \cos\phi}) \right|^2 \\
\longrightarrow U &= \frac{\eta k^4 \sin^2\theta}{32\pi^2} [(I_0 a_0 b_0)^2 + 2 I_0 a_0 b_0 I_1 a_1 b_1 (\cos\alpha \cos(kd \sin\theta \cos\phi) \\
&\quad - \sin\alpha \sin(kd \sin\theta \cos\phi)) + (I_1 a_1 b_1)^2] \\
&= \frac{\eta k^4 \sin^2\theta}{32\pi^2} [(I_0 a_0 b_0)^2 + 2 I_0 a_0 b_0 I_1 a_1 b_1 \cos(\alpha + kd \sin\theta \cos\phi) + (I_1 a_1 b_1)^2] \qquad (4.1)
\end{aligned}$$

As may have been expected, the radiation intensity is maximized when $\alpha = -kd \sin\theta \cos\phi$. Now, for two *centered* rectangular loops and no phase-shift forced on I_0, the expression for U would be exactly the same as 4.1, only without the cosine factor in the cross term. Thus, we may conclude that the directivity of an off-center inner rectangular loop actually decreases (unsurprisingly), unless a phase-shift is put onto the current in the outer loop. If that phase shift is equal to $-kd \sin\theta \cos\phi$, then U achieves the same value as was just described in the centered case for both loops. In conclusion, this means that the intensity (and hence directivity) of a rectangular loop with an off-center inner loop is at **most** as large as the both-centered case. They are only ever equal when the correct phase-shift is applied to I_0. This means that directivity cannot be increased or exploited in any useful way for off-center nested loops.

Now it is time to investigate the case when the rectangular loops are large. We start off simple and first analyze the interference due to two nested *centered* rectangular loops. For one rectangular loop in the xy plane with side lengths a and b (a runs along the $\mathbf{x}$ direction and b runs along the $\mathbf{y}$ direction. This loop consists of four pieces, two dipoles facing in the $\mathbf{x}$ direction and two dipoles in the $\mathbf{y}$ direction. For an $\mathbf{x}$ facing dipole that is translated down the y-axis by an amount $b/2$, the vector potential function is given as

$$\mathbf{A} = \frac{\mu I a}{4\pi r} e^{-jkr} e^{-jkbsin\theta sin\phi/2} \mathbf{x}$$

converting this to spherical coordinates and then calculating the $\mathbf{E}$ in the far field gives us

$$\begin{aligned}
dE_r &\approx 0 \\
dE_\theta &= \frac{-jIk^2 e^{-jkbsin\theta sin\phi/2}}{4\pi r \epsilon \omega} \cos\theta \cos\phi e^{-jkr} dx' \\
dE_\phi &= \frac{jIk^2 e^{-jkbsin\theta sin\phi/2}}{4\pi r \epsilon \omega} \sin\theta e^{-jkr} dx'
\end{aligned}$$

Recall that the differentials are due to the fact that we are dealing with an infinitesimal dipole . To create a finite length dipole, we integrate each dE component over the length of our desired dipole, which is $[-a/2, a/2]$ (technique from [2]). Here is the calculation for the θ component. ϕ is similar.

$$E_\theta = \frac{-jIk^2 e^{-jkbsin\theta sin\phi/2}}{4\pi r \epsilon \omega} \cos\theta \cos\phi \int_{-a/2}^{a/2} e^{-jkR} dx' \qquad (4.2)$$

Because the integration itself introduces a factor of a, we can, to first order, replace the integrand with its value at the center of the translated dipole (I took this technique from [3], however I can't be sure that it can be applied here - all I know is that without this shortcut the calculation becomes impossible). This technique may seem simple, but the approximation requires its own (binomial) approximation (again

from [3]).

$$R^2 = r^2 + (b/2)^2 - 2r(b/2)\cos\psi = r^2(1 + (\frac{b/2}{r})^2 - 2\frac{b/2}{r}\sin\theta\sin\phi)$$
$$\longrightarrow R \approx r\left(1 + \frac{1}{2}\frac{(b/2)^2}{r^2} - \frac{b/2}{r}\sin\theta\sin\phi - \frac{1}{8}\left[(\frac{b/2}{r})^4 - 4\frac{(b/2)^3}{r^3}\sin\theta\sin\phi + 4\frac{(b/2)^2}{r^2}\sin^2\theta\sin^2\phi\right]\right)$$
$$\approx r + \frac{(b/2)^2}{2r} - (b/2)\sin\theta\sin\phi - \frac{(b/2)^2}{2r}\sin^2\theta\sin^2\phi \tag{4.3}$$

Now we can skip the integral in (4.2) by replacing R with its approximate value in the center of the dipole given by (4.3). (4.2) then simplifies to

$$\begin{aligned} E_\theta &\approx \frac{-jaIk^2 e^{-jkbsin\theta sin\phi/2}}{4\pi r\epsilon\omega}\cos\theta\cos\phi e^{-jk(r+\frac{(b/2)^2}{2r} - (b/2)\sin\theta\sin\phi - \frac{(b/2)^2}{2r}\sin^2\theta\sin^2\phi)} \\ &= \frac{-jaIk^2}{4\pi r\epsilon\omega}\cos\theta\cos\phi e^{-jkr}e^{-jk\frac{(b/2)^2}{2r}(1-\sin^2\theta\sin^2\phi)}. \end{aligned} \tag{4.4}$$

For the other dipole facing the $\mathbf{x}$ direction, we just replace $b/2$ in (4.4) with $-b/2$. Adding the expressions for both $\mathbf{x}$ facing dipoles together gives us

$$E_\theta \approx \frac{-jaIk^2}{2\pi r\epsilon\omega}\cos\theta\cos\phi e^{-jkr}e^{-jk\frac{(b/2)^2}{2r}(1-\sin^2\theta\sin^2\phi)}.$$

The solution for the ϕ component is similarly obtained,

$$E_\phi \approx \frac{-jaIk^2}{2\pi r\epsilon\omega}\sin\phi e^{-jkr}e^{-jk\frac{(b/2)^2}{2r}(1-\sin^2\theta\sin^2\phi)}.$$

Now we do the same for the dipoles facing the $\mathbf{y}$ direction and arrive at

$$\begin{aligned} E_r &\approx 0 \\ E_\theta &\approx \frac{-jbIk^2}{2\pi r\epsilon\omega}\cos\theta\sin\phi e^{-jkr}e^{-jk\frac{(a/2)^2}{2r}(1-\sin^2\theta\cos^2\phi)} \\ E_\phi &\approx \frac{jbIk^2}{2\pi r\epsilon\omega}\cos\phi e^{-jkr}e^{-jk\frac{(a/2)^2}{2r}(1-\sin^2\theta\cos^2\phi)} \end{aligned}$$

Adding everything together, we arrive at the E far-field components for a large rectangular loop

$$E_r \approx 0 \tag{4.5a}$$
$$E_\theta \approx -\frac{jIk^2}{2\pi r\epsilon\omega}\cos\theta e^{-jkr}[a\cos\phi e^{-jk\frac{(b/2)^2}{2r}(1-\sin^2\theta\sin^2\phi)} + b\sin\phi e^{-jk\frac{(a/2)^2}{2r}(1-\sin^2\theta\cos^2\phi)}] \tag{4.5b}$$
$$E_\phi \approx \frac{jIk^2}{2\pi r\epsilon\omega}e^{-jkr}[b\cos\phi e^{-jk\frac{(a/2)^2}{2r}(1-\sin^2\theta\cos^2\phi)} - a\sin\phi e^{-jk\frac{(b/2)^2}{2r}(1-\sin^2\theta\sin^2\phi)}]. \tag{4.5c}$$

Now, translating a loop in any direction and doing this all over again would be an absolute nightmare. The resulting equations would likely be pages long. Fortunately, recall back in the previous case with infinitesimal loops - when we translated the loop, the resulting equations where precisely the same, except r had been replaced with $r + d\sin\theta\cos\phi$ in the phase term only (not in the length term). So

that is what will be done here.

$$E_r \approx 0 \tag{4.6a}$$

$$E_\theta \approx -\frac{jIk^2}{2\pi r\epsilon\omega}\cos\theta e^{-jkr}e^{-jkd\sin\theta\cos\phi}\left\{a\cos\phi\exp\left[-jk\frac{(b/2)^2}{2(r+d\sin\theta\cos\phi)}(1-\sin^2\theta\sin^2\phi)\right]\right.$$
$$\left. + \; b\sin\phi\exp\left[-jk\frac{(a/2)^2}{2(r+d\sin\theta\cos\phi)}(1-\sin^2\theta\cos^2\phi)\right]\right\} \tag{4.6b}$$

$$E_\phi \approx \frac{jIk^2}{2\pi r\epsilon\omega}e^{-jkr}e^{-jkd\sin\theta\cos\phi}\left\{b\cos\phi\exp\left[-jk\frac{(a/2)^2}{2(r+d\sin\theta\cos\phi)}(1-\sin^2\theta\cos^2\phi)\right]\right.$$
$$\left. + \; a\sin\phi\exp\left[-jk\frac{(b/2)^2}{2(r+d\sin\theta\cos\phi)}(1-\sin^2\theta\sin^2\phi)\right]\right\} \tag{4.6c}$$

At this point, to get the full electric field components of the nested off-center loops, just add equations (4.5) and (4.6) (a-c) together. Finally, we are ready to find the intensity U to get some insight into the directivity. However, I don't really feel like doing that (and you can't really blame me, either). Hopefully these equations will find some use on their own.

4.2. **Analysis of Flux.**

For this section, we are interested in how exactly the induced EMF in the flux bias line is influencing the magnetic flux through the SQUID array with respect to incoming signal frequency and phase changes. Just like last section, I realized too late that this calculation of mine will not actually answer this question (since, to my knowledge, mutual inductance and total flux really just quantify the strength of the mutual flux between both loops and will not give any answers about phase delays). Regardless, I had already done it and wrote it up, so here you go.

We try and calculate the flux through a rectangular loop that is centered inside a square loop being supplied with DC current. The magnetic field at a point above a finite length wire of length L and current I is given as

$$B = \frac{\mu_0 I}{4\pi a}(\sin(\theta_2) - \sin(\theta_1))$$

Here, the angles $\theta_{1,2}$ measure the angles between the point of interest and each end of the wire. Since the point of interest is always inside the breadth of the wire, θ_1 actually takes on a negative value. Our equation then becomes

$$B = \frac{\mu_0 I}{4\pi a}(\sin(\theta_2) + \sin(\theta_1))$$

Now, if our point of interest is a distance a above the wire and a distance b from the left end of the wire, then the equation becomes

$$B = \frac{\mu_0 I}{4\pi a}(\sin(\arctan((L-b)/a)) - \sin(\arctan(b/a)))$$

Using the identity $\sin(\arctan(x)) = \frac{x}{\sqrt{1+x^2}}$, we arrive at

$$B = \frac{\mu_0 I}{4\pi}\frac{1}{a^2}\left[\frac{L-b}{\sqrt{1+((L-b)/a)^2}} + \frac{b}{\sqrt{1+(b/a)^2}}\right] \tag{4.7}$$

We consider our point to now be the lower left corner of a rectangle which stretches horizontally from b_1 to b_2 and vertically from a_1 to a_2 which traces out a rectangle. Thus to find the total flux through

the rectangle, we integrate (4.7) across it

$$\begin{aligned}\Phi &= \frac{\mu_0 I}{4\pi}\int_{b_1}^{b_2}\int_{a_1}^{a_2}\frac{1}{a^2}\left[\frac{L-b}{\sqrt{1+((L-b)/a)^2}}+\frac{b}{\sqrt{1+(b/a)^2}}\right]\mathrm{d}a\mathrm{d}b\\ &= \frac{\mu_0 I}{4\pi}\int_{b_1}^{b_2}\sinh^{-1}\left(\frac{b-L}{a}\right)-\sinh^{-1}\left(\frac{b}{a}\right)\Bigg|_{a=a_1}^{a=a_2}\mathrm{d}b\\ &= \frac{\mu_0 I}{4\pi}\Bigg[-a\sqrt{G/a^2}-\frac{aL\sqrt{G/a^2}\ln(\sqrt{a^2+(b-L)^2}+b-L)}{\sqrt{G}}+a\sqrt{\frac{a^2+b^2}{a^2}}\\ &\qquad -b\left(\sinh^{-1}\left(\frac{b}{a}\right)-\sinh^{-1}\left(\frac{b-L}{a}\right)\right)\Bigg]\Bigg|_{a=a_1}^{a=a_2}\Bigg|_{b=b_1}^{b=b_2},\end{aligned}$$

where $G = a^2 + b^2 - 2bL + L^2$.

Now we extend this concept to placing our rectangle into the center of a large square conducting loop of side length L (I could have done this for an actual rectangle but didn't feel like it). We just calculated the flux from the bottom part of the loop. The top part of the loop contributes the exact same amount of flux, so we just multiply the above equation by 2. For a side (say the left side), we can do the calculation all over again. However, doing this problem on the side of the square is the same as making the calculation on the bottom of the square, only now the rectangle has been rotated 90 degrees. We pull off the exact same calculation, only now we are integrating from $a = a_1'$ to $a = a_2'$ and from $b = b_1'$ to $b = b_2'$. This yields the same exact same long equation above, only the bounds are switched to their primed counterparts. Finally, we realize that $a_1' = b_1,\ a_2' = b_2,\ b_1' = a_1$, and $b_2' = a_2$. Multiplying this result by 2 (as before with the bottom and top parts of the square) and adding everything up gives us the final flux through the inner rectangle due to the square outer loop with current I,

$$\begin{aligned}\Phi = \frac{\mu_0 I}{2\pi}\Bigg[&-a\sqrt{G/a^2}-\frac{aL\sqrt{G/a^2}\ln(\sqrt{a^2+(b-L)^2}+b-L)}{\sqrt{G}}+a\sqrt{\frac{a^2+b^2}{a^2}}\\ &-b\left(\sinh^{-1}\left(\frac{b}{a}\right)-\sinh^{-1}\left(\frac{b-L}{a}\right)\right)\Bigg]\left(\Bigg|_{a=a_1}^{a=a_2}\Bigg|_{b=b_1}^{b=b_2}+\Bigg|_{a=b_1}^{a=b_2}\Bigg|_{b=a_1}^{b=a_2}\right)\end{aligned}$$

4.3. **Numerical Calculation of Flux.**

Relevant programs: `FluxGradient.m`

The equations in the previous section are a little bit too scary to be of much use. In this section, those equations will be replaced with actual numerical calculations. The results are simply better and the work itself was much easier to come by.

Finding out how to calculate flux is common knowledge. In my case, I primarily used [3] as a reference. In MATLAB, two simplifications were made to make the calculation very easy. The first is that the outer loop is considered to be square. The second is that the inner (SQUID array) loop is taken to be one-dimensional. This second assumption is due to the fact that the SQUID array is indeed much longer than it is wide and so this shortcut should not take us too far off from the true answer.

I unfortunately do not remember the details of how I arrived at the equations that are used in `FluxGradient.m`, but I can say that the process itself was not too difficult. If you followed the last section, then making the same calculation using the assumptions above should be relatively easy. Glancing at the code, it appears that I divided up the SQUID array into many pieces, then each piece was evaluated in a function (fairly similar to (4.7) actually) and added up to mimic an integral.

For the experiment, a one-dimensional SQUID array of length 2.5 mm is placed in the center of a square flux bias line with a side length of 8 mm and a DC current running through it at 0.1 amps. Here is the result.

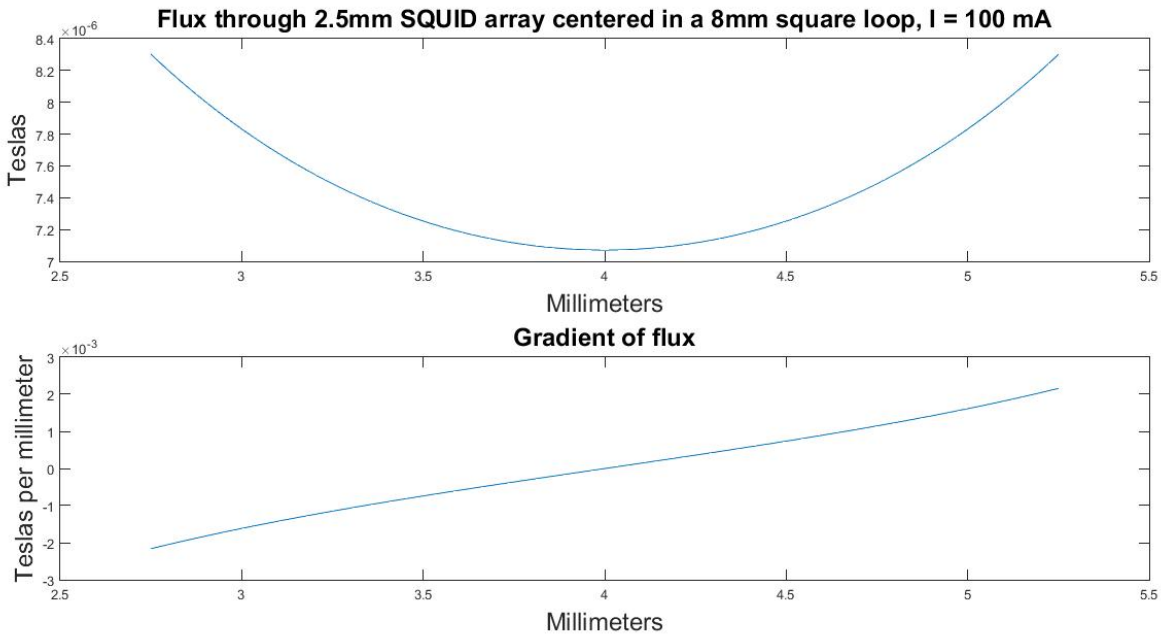

FIGURE 4.1. Magnetic flux through a SQUID array induced by a square flux bias line

From the graph, the maximum amount of flux occurs at the ends of the SQUID array and the minimum flux occurs in the middle. This represents about a 15% total decrease in flux. This is an important result, because this flux gradient will affect the performance of the SQUID array. Ideally, the flux going through the SQUID array should be constant throughout its length. We now know that any alternative solution to the flux bias line should exhibit a relative flux difference that is at most 15% throughout the SQUID array (since obviously we want the alternative to provide better results than the flux bias line). The next section should provide an argument that an alternative to the flux bias line is definitely needed.

4.4. **Superconductor Impedance.**

This section is an attempt to explain what is happening when the flux bias line reacts with the surrounding RF and induces an unwanted magnetic flux through the SQUID array. To make matters worse, the phase delay on this induced flux changes erratically with respect to the frequency of the signals. This phase-frequency relation is fairly simple to understand in a normal conducting loop of wire. However, the flux bias line itself is superconducting, which creates many complications and is described below. The following is paraphrased from pages 40-42 of [4] (although to my credit, I've corrected some errors and my exposition should be a bit easier to follow than in the book).

In the following analysis we use the two-fluid model, meaning that we assume that there are normal electrons of density n_n and superconducting electrons of density n_s. The total charge carrier density can then be given as $n = n_n + n_s$. Furthermore, we assume that the electron mean free path l and the frequency are low enough such that l is less than the penetration depth of the electromagnetic field. The frequency of electron collisions is represented by τ^{-1}. We now calculate the complex conductivity of a superconductor.

For normal electrons, we use Newton's second law:

$$e\mathbf{E} - \frac{m}{n_n e}\frac{\mathbf{j}_n}{\tau} = \frac{m}{n_n e}\frac{d\mathbf{j}_n}{dt} \tag{4.8}$$

where e and m are the charge and mass of an electron, and $\mathbf{j}_n$ represents the current density caused by normal electrons. Note that the first term is the force, and the second is the average"friction" due to electron collisions. The term on the right is the product of electron mass and acceleration. For one normal electron, we can rearrange this equation into

$$\mathbf{E} = \frac{n_s}{n_n}\Lambda\frac{d\mathbf{j}_n}{dt} + \frac{n_s}{n_n}\Lambda\frac{\mathbf{j}_n}{\tau}$$

Here, $\Lambda = m/n_s e^2$. We use the first London equation to represent the motion of the superconducting electrons.

$$\mathbf{E} = \Lambda\frac{d\mathbf{j}_s}{dt} \tag{4.9}$$

where $\mathbf{j}_s$ is the current density from the superconducting electrons. For an AC current, we let $\mathbf{j}_s \propto e^{i\omega t}$ and $\mathbf{j}_n \propto e^{i\omega t}$, with ω being the frequency. This leads (4.8) and (4.9) to become

$$\begin{aligned}
\mathbf{j}_n &= \frac{n_n}{n_s}\frac{\tau}{\Lambda}\frac{1-i\omega\tau}{1+(\omega\tau)^2}\mathbf{E}, \\
\mathbf{j}_s &= -i\frac{1}{\Lambda\omega}\mathbf{E}.
\end{aligned}$$

Thus we arrive at an expression for conductivity:

$$\begin{aligned}
\mathbf{j} &= \sigma\mathbf{E}, \quad \sigma = \sigma_1 - i\sigma_2, \\
\sigma_1 &= \frac{n_n}{n_s}\frac{\tau}{\Lambda}\frac{1}{1+(\omega\tau)^2}, \\
\sigma_2 &= \frac{1}{\Lambda\omega}\left[1 + \frac{n_n}{n_s}\frac{(\omega\tau)^2}{1+(\omega\tau)^2}\right]
\end{aligned}$$

We would like to calculate the surface impedance now, but first we need to know to what extent the surface of the superconductor will be penetrated by an electromagnetic field. We start with some of Maxwell's equations:

$$\nabla \times \mathbf{H} = \frac{4\pi}{c}\sigma\mathbf{E}, \tag{4.10}$$

$$\nabla \times \mathbf{E} = -\frac{1}{c}\frac{\partial\mathbf{H}}{\partial t} \tag{4.11}$$

Taking the curl of both sides of (4.10) and using (4.11) gives us

$$-\nabla^2\mathbf{H} = -\frac{4\pi}{c^2}\sigma\frac{\partial\mathbf{H}}{\partial t} \tag{4.12}$$

where we have used $\nabla \times (\nabla \times \mathbf{H}) = -\nabla^2\mathbf{H}$. Now as before, for an AC current, we let $\mathbf{H} \propto e^{-i(kx-\omega t)}$. The additional term in the exponent represents the exponential decay of the field as it enters the superconductor. For reference, our superconductor coincides with the plane $x = 0$. With this substitution, (4.12) becomes

$$-k^2\mathbf{H} = i\frac{4\pi\sigma\omega}{c^2}\mathbf{H}. \tag{4.13}$$

This allows us to solve for k,

$$\begin{aligned} k &= \frac{1+i}{\delta}, \\ \text{where } \delta &= \left(\frac{c^2}{2\pi\sigma\omega}\right)^{1/2}. \end{aligned}$$

Now we are ready to find the impedance of the superconductor. By definition, the surface impedance is given by

$$Z = \frac{4\pi}{c}\frac{E}{H}. \tag{4.14}$$

With the substitution $\mathbf{H} \propto e^{-i(kx-\omega t)}$, (4.10) turns into

$$ikH = \frac{4\pi}{c}\sigma E. \tag{4.15}$$

Then (4.14) combined with (4.15) gives us

$$Z = \frac{4\pi}{c}\frac{E}{H} = \frac{ik}{\sigma} = \frac{1+i}{\sigma\delta} = R + iX.$$

I will procrastinate on finding the values for R and X explicitly for the time being (it's a bit of a pain). As is tradition, we attribute R to be resistance and X to be (kinetic) inductance. It should be clear that the role that ω plays in this impedance is quite complex, and so one should expect current to lag behind voltage in a very chaotic manner in response to varying frequencies.

5. Solenoid Simulations

5.1. **Regular Solenoids.**

Relevant Programs: `SolenoidTiltNew.m`

After the previous section, it should be fairly clear that the flux bias line may not be the best solution to providing a constant amount of flux through the SQUID array. This section introduces an alternative to the flux bias line. This alternative is to place a solenoid somewhere around the SQUID array and accomplish exactly what the flux bias line did but at a much farther distance from the SQUID array. The hope is that this approach will not bring along the troublesome induced magnetic flux that the flux bias line has.

Prior to buying or building a bunch of solenoids to check out which ones work best, it is smart to just try and simulate some results to find a suitable starting point. Luckily for me, some very smart people a long time ago analytically derived the magnetic field components of a solenoid given any shape and current distribution.

$$\begin{aligned} B_z &= -\frac{\mu_0 I}{4\pi}\frac{1}{2L\sqrt{a\rho}}\left[\zeta k\left(K(k^2)+\frac{a-\rho}{a+\rho}\Pi(h^2,k^2)\right)\right]\Bigg|_{\zeta_-}^{\zeta_+} \\ B_\rho &= \frac{\mu_0 I}{4\pi}\frac{1}{L}\sqrt{\frac{a}{\rho}}\left[\left(\frac{k^2-2}{k}K(k^2)+\frac{2}{k}E(k^2)\right)\right]\Bigg|_{\zeta_-}^{\zeta_+} \end{aligned}$$

Now, there are a lot of variables and things going on here and none of them really matter in this report. The important thing to know is that these equations involve some elliptic integrals and some funny complicated abbreviations and such - all of which I was able to offload into MATLAB to do all the heavy lifting. The result is `SolenoidTiltNew.m`, a program that simulates the magnetic field of a solenoid that can be moved and tilted into any position. If you paid attention in section 4.3, you'll know that in order to replace the flux bias line we'd need something that performs better than it. This means providing a magnetic flux through the SQUID array that does not exhibit a variance in flux that is over 15%. The second requirement is that the flux through the SQUID array needs to reach at least 65 microteslas in order to combat the effects of Earth's magnetic field.

The SQUID array itself is modeled to be a one-dimensional line that is around 2.23 mm tall and stands vertically. The array is shielded and is supported by some type of pillar, meaning that a solenoid will bump into it before it gets too close. The details I was given were that the solenoid needs to be around 2.5 cm below the array and around 4 cm to the left or right of it. This creates a problem because the magnetic field strength of a solenoid greatly decreases with distance, so it's desirable to nudge the solenoid up as close to the array as possible. Assuming a standard permeability of free space and a maximum current of 5 amps, we wish to know the shape and position the solenoid should take in order to satisfy the necessary requirements.

What I found was that a fat solenoid is preferable to a long one because the field becomes much stronger (but maybe you already knew that). With a fat solenoid, the field that is strongest comes right out of the top. So, for a solenoid that is placed below and to the left or right of a SQUID array, it is best to tilt it in such a way as to "aim" the magnetic field coming from the top of the solenoid into going right through the array. I tried many other variations of solenoid placement, but the one just described provides the best results. The nicest property is that the variation in field strength is far below 15% throughout the SQUID array. Unfortunately, with the given solenoid specifications (current and permeability limitations), I have not been able to find a size or placement configuration that produces a 65 microtesla field strength. For a fat tilted solenoid, the flux through the SQUID hovers around 2-4 microteslas. This flux can be doubled by placing another solenoid opposite to the first solenoid and with

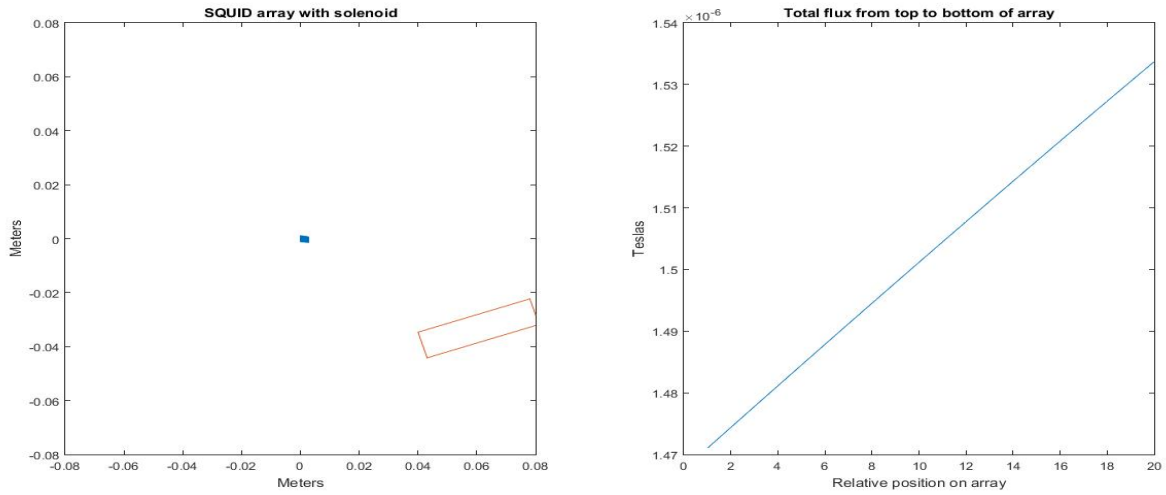

FIGURE 5.1. Solenoid length = 1 cm, radius = 2 cm, I = -5 amps centered at $(x, y) = (6.06, -3.32)$ cm at an angle of 18 degrees. Average flux = 1.503 microteslas and maximum flux variance = 4.08%

a reversed current. Other solutions to getting more flux is to use a solenoid with more layers, using a ferromagnetic core, or simply using a larger current. These possibilities are not considered here.

To find the best placement, we try and get the solenoid as close as possible to the SQUID array while obeying some restrictions: the *extremities* of the solenoid must stay 2.23 cm below the bottom of the array and 4 cm to the right of the array (the specific numbers here are just placeholders and can be replaced if needed). The restriction to stay below a certain point is important because we don't want anything standing between the SQUID array and a signal. From a specific angle, the solenoid is just a rectangle and so these restrictions are dealt with by using some trigonometry (lines 87-88 in `SolenoidTiltNew.m` if you're interested). The size of the solenoid is actually chosen arbitrarily, because the exact details on size restrictions and preferences are still up in the air. As a result, I just went with random fat solenoids. After a size is chosen, `SolenoidTiltNew.m` loops through all the angles between 0 and 90 while recording the maximum flux variation and average flux for each angle. Throughout the experiment, the solenoid is kept nudged up against the SQUID array housing. Since the variation in strength is always below 15%, the optimal angle is chosen to be the one that provides the most strength. Here are a few examples. The speck in the center of each left-hand subplot is a group of magnetic field vectors coming from the SQUID array. As of this writing, I am not aware of any other good solenoid placement/combinations that would give nicer results.

5.2. **Solenoid Quadrupoles.**

Relevant Programs: `Quadrupole.m`

An alternative to using just one solenoid is to couple two solenoids together in order to form a magnetic quadrupole. The main draw of this is that a strong uniform magnetic field not only forms inside the quadrupole, but also shoots out of the side like an arrow. We can possibly use this to better aim the field to go right through the SQUID array and produce a strong flux. The same exact experiment is conducted here, just note that the physical placement corresponds to the center of the solenoid.

This experiment is set up in the same way - the quadrupole is nudged up as close to as possible to the SQUID array and then looped through every angle between horizontal and vertical. Each angle however features another step. The current in one solenoid can be varied with respect to the other and

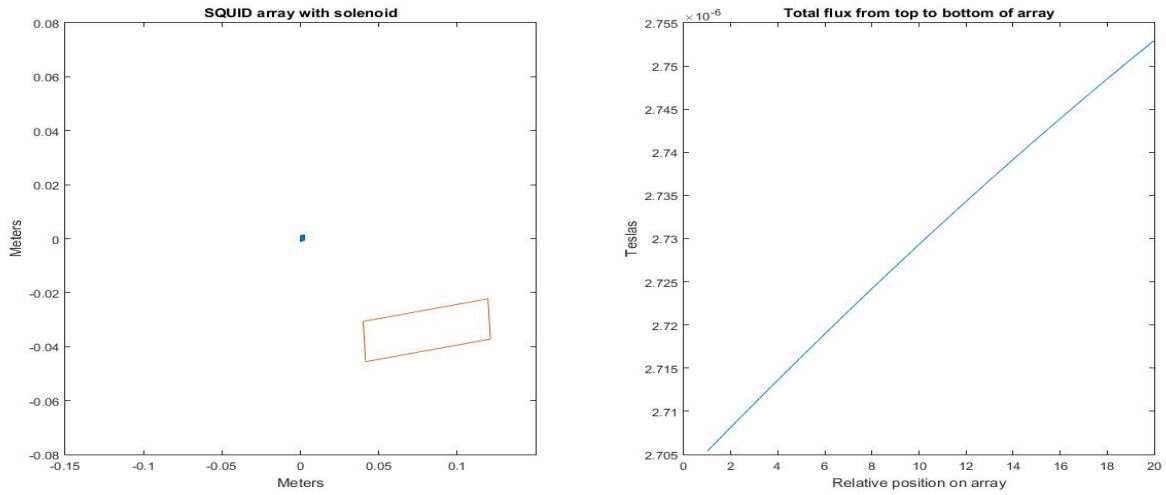

FIGURE 5.2. Solenoid length = 1.5 cm, radius = 4 cm, I = -5 amps centered at $(x, y) = (8.06, -3.39)$ cm at an angle of 6 degrees. Average flux = 2.7301 microteslas and maximum flux variance = 1.728%

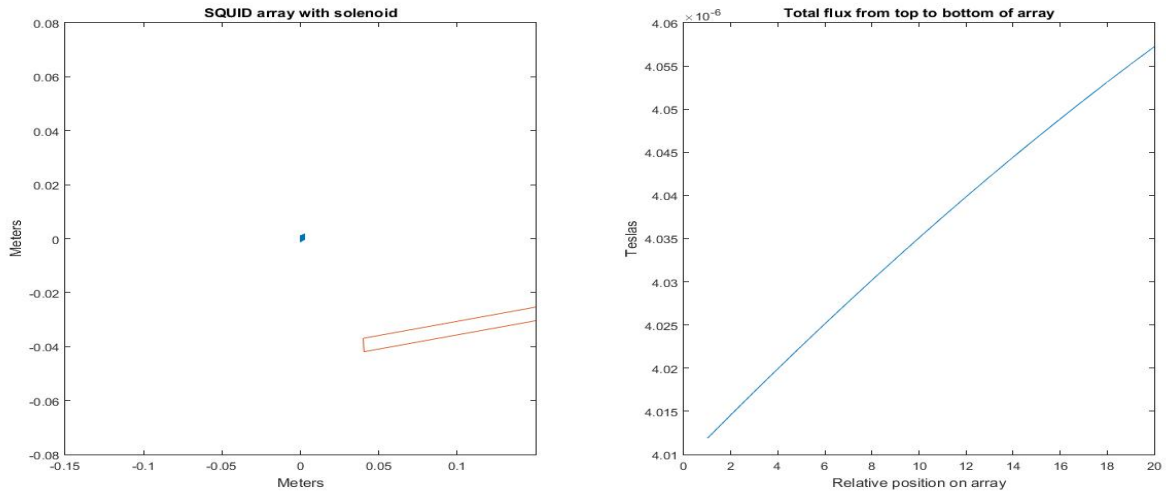

FIGURE 5.3. Solenoid length = 0.5 cm, radius = 7 cm, I = -5 amps centered at $(x, y) = (10.99, -3.21)$ cm at an angle of 6 degrees. Average flux = 4.034 microteslas and maximum flux variance = 1.118%

will cause the straight magnetic field out the side to become skewed slightly. Thus for each angle, the current in the top solenoid is varied in strength to be between 50% and 150% of the bottom solenoid. The motivation for this step is that steering the magnetic flux relative to the quadrupole's angle may prove to be useful. If you zoomed in on the blue speck on the figures in section 5.1, you'd see that the flux vectors are always pointing diagonally downwards. This is bad because we want them to be horizontal for full strength. Thus it may be possible to angle the solenoid at a higher angle (so that the straight flux off shoots above the SQUID array), then tweak the current in the top solenoid so that the flux steers downward resulting in flux going right horizontally through the SQUID array.

After looping through all angles, varying the current in the top solenoid by 10% between -100% to 100% of the current in the bottom solenoid for each angle, the results are not very satisfying. The simulation results show that the strongest power occurs at an angle of 0 degrees, with the currents being

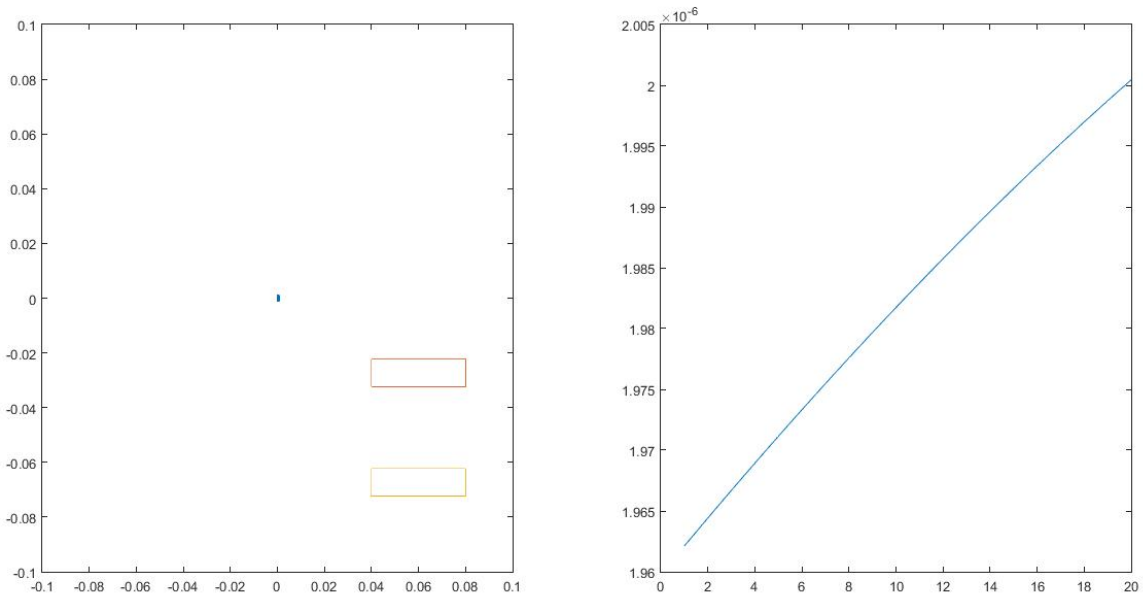

FIGURE 5.4. Solenoid length = 1 cm, radius = 2 cm, I = 5 on both solenoids, centered at $(x, y) = (6, -6.73)$ cm at an angle of 0.01 degrees. Average flux = 1.982 microteslas and maximum flux variance = 1.915%

equal. Quite a far cry from aiming the solenoid at some nice angle, but it is what it is. I suspect that when the quadrupole tilts, it is adjusted to be further from the SQUID array (since its extremities bulge out farther at angle and hit the housing) and may account for the maximum power occurring when there is no angle. It is clear that this is somehow not a viable method for supplying the SQUID array with flux, because it can be achieved just as well or better with a single solenoid. I will admit that there is room for error with my positioning criteria, but after doing other tests with closer quadrupoles, I have not been convinced that it will ever outperform a single tilted solenoid.

6. Direction-Finding

6.1. **MUSIC performance on real SQUID output.**

Two SQUID arrays spaced 2.5mm apart (give or take) were placed on a Lazy-Susan-type apparatus and subjected to a 1GHz signal. The signal was recorded for both arrays in 10 degree increments for orientations between 0 and 170 degrees. The measured angle is between the transmitter and some reference point on the two antennas. Below is the raw output of this experiment with the flux bias line turned on.

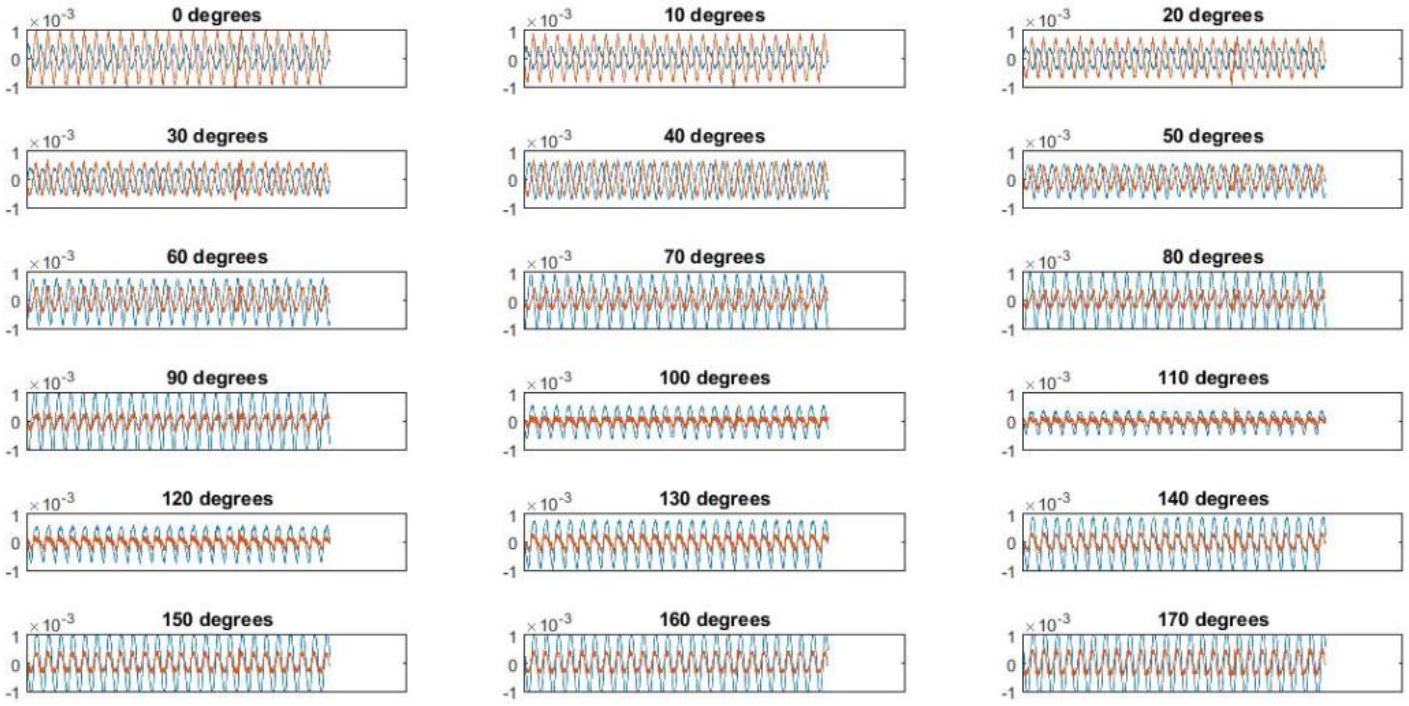

Figure 6.1. SQUID output for each angle

The red signal is much noisier than the blue signal so it is smoothed with a moving average filter before being fed into MUSIC. The varying amplitudes are also repaired so that both signals have the same maximum amplitude. The results are listed in table 1.

The huge discrepancy between the true answers and the estimates should be no immediate cause for concern. MUSIC assumes 0 degrees to be along the axis that the antennas are placed on while this was likely not the case during the experiment. Hence the two definitions for angle-of-arrival are different. With this in mind, it appears that the performance between 0 and 40 degrees is actually fairly good for the first column, each 10 degree increment is complemented by a 10 degree increase in the estimate. Between 40 and 50 degrees, the estimate jumps by almost 30 degrees (mod 180) which is troubling. After 50 degrees, the performance becomes progressively worse. Each 10 degree increment results in estimates that advance by smaller and smaller amounts (not good!).

An important thing to note here is that MUSIC has been set to search for the signals as if the two antennas were placed exactly one-half of a wavelength apart from each other. This is entirely incorrect. A 1 GHz signal has a wavelength of approximately 30 cm, meaning that the 2.5 mm separation of the antennas amounts to a measly 8.3 percent of a wavelength. It is not common for MUSIC to work well with such a small separation. In fact, when MUSIC is set to have the correct physical separation of the experiment, each estimate is either 0 or 180 degrees (essentially a complete failure).

True answer	MUSIC estimate (red signal filtered)	MUSIC estimate (both signals filtered)
0	121	145
10	130	163
20	142	25
30	152	36
40	164	43
50	13	48
60	23	52
70	34	59
80	43	65
90	52	72
100	60	79
110	64	85
120	70	89
130	74	93
140	78	96
150	81	99
160	84	101
170	87	104

TABLE 1. My caption

This raises a natural question: why does MUSIC work more effectively with a false sense of antenna separation? A picture of the signals should provide the answer. Below is a hypothetical picture of what happens when two antennas are spaced a half-wavelength apart (the frequency is arbitrary)

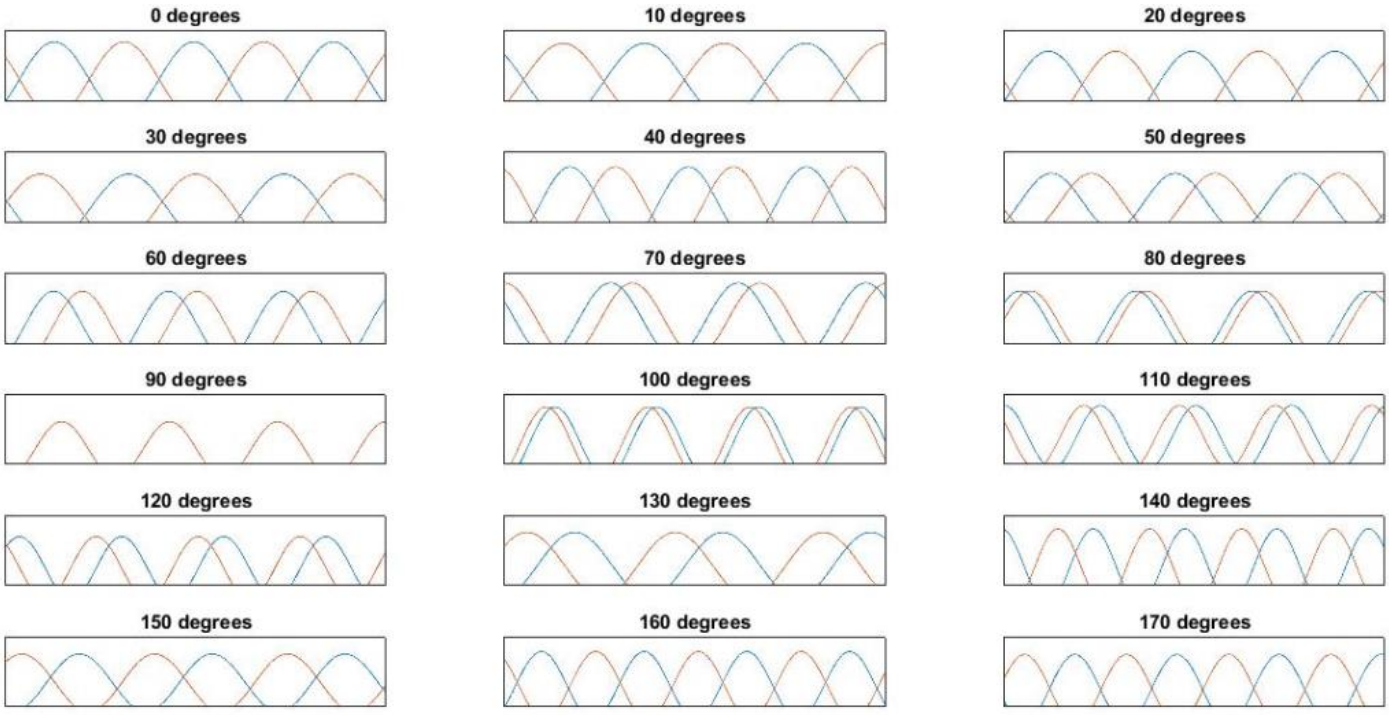

FIGURE 6.2. Signal output from two antennas that are placed a half-wavelength apart.

The waves have maximum spacing at 0 degrees (as a bonus, this should answer the question as to why a half-wavelength separation is ideal) and smoothly come together, becoming totally overlapped at 90 degrees, then moving further past each other with each subsequent increment. In stark contrast, here is an ideal picture of two antennas spaced 8.3 percent of a wavelength apart (essentially mimicking what we SHOULD see with the experiment):

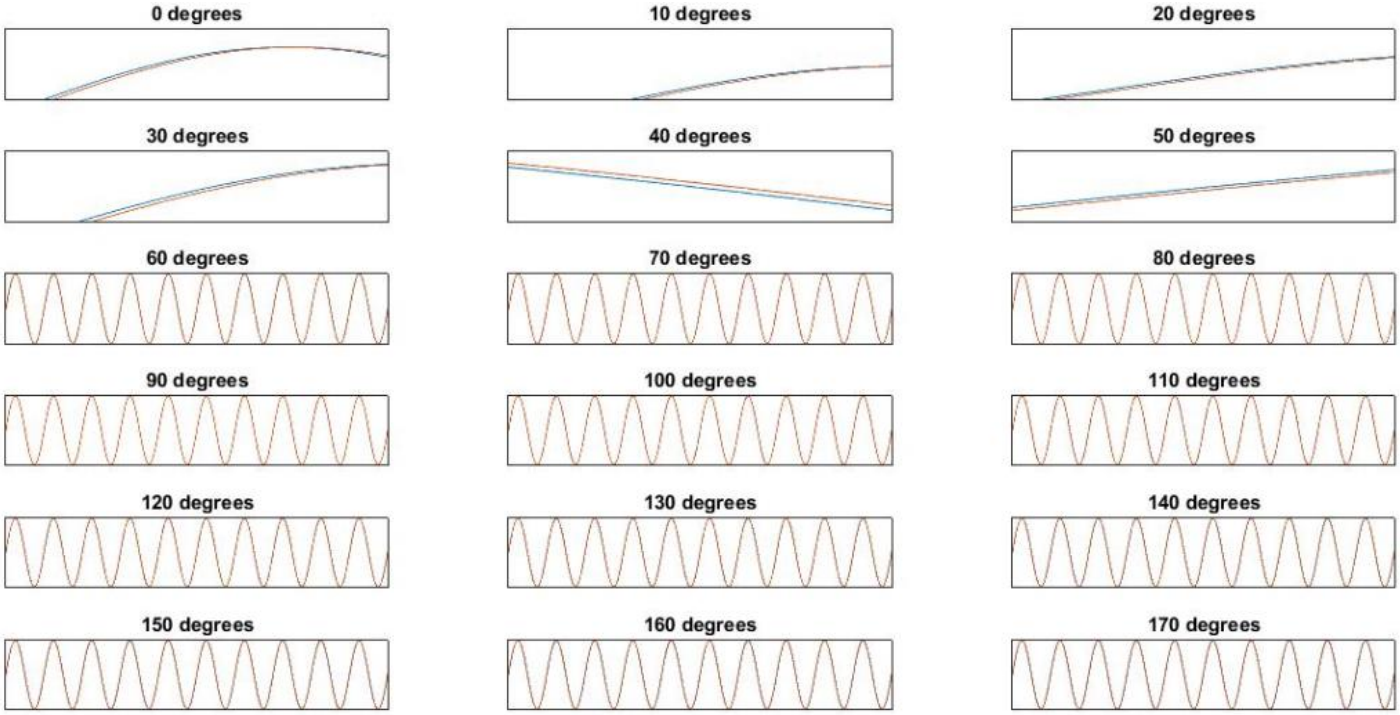

FIGURE 6.3. Waveforms that mimic what should be occuring in the SQUID experiment.

Here, the first 6 pictures have been zoomed in on in an attempt to magnify the extremely tiny phase difference between the two signals (this is why very small antenna separation is bad!). The latter 12 pictures show that the phase differences are so small that the two signals appear to be one. Now, figure 6.4 is a picture of the experiment data with a smoothed red signal and fixed amplitudes (the filter causes a phase difference all on its own; this has been accounted for and reversed):

This looks nothing like the picture that is supposed to emulate the experiment. It appears as if the antennas truly are spaced close to a half-wavelength apart. In fact, at 10 degrees it appears that the two signals are almost exactly 90 degrees out of phase with each other which is NOT possible with a very small separation. This means that one or more of the parameters of the experiment has been calculated in error, or some other problem has happened (or maybe I just dont know what Im doing). There are a few other things to note here: the phases between 0 and 40 degrees appear to move along very well, while other angles do not demonstrate the same behavior. Between 80 and 140 degrees the increments are very small, with 110-140 looking almost identical. Some of the behaviors here are reflected in the MUSIC estimates, however there are other quirks in the estimations that do not match the pictures shown here which I cannot explain at the moment.

A glance at figure 6.1 holds another peculiar characteristic- the respective amplitudes between both signals vary significantly between measurements. As far as I know, this can be attributed to two causes. The first is that a SQUID antenna is directional to begin with. The amplitude is maximized when a signal impinges on the front of the antenna, and should reduce to zero when a signal impinges perpendicularly with respect to the front. The second reason is that there is a signal echo that the SQUID is picking up caused by the flux bias line which is itself acting like an antenna and responding to the transmitted

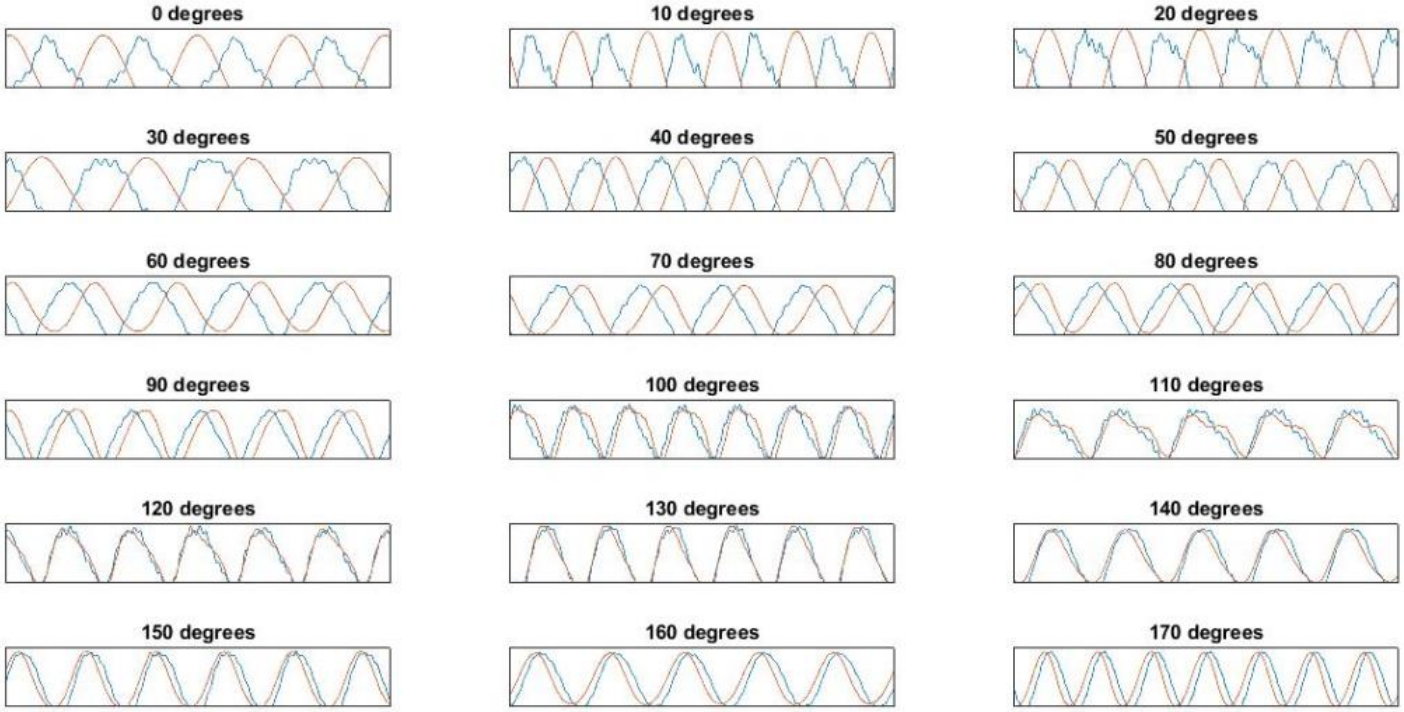

FIGURE 6.4. Smoothed and zoomed-in picture of SQUID outputs.

signal. This second signal is identical to the first but is delayed by a small amount due to the impedance (I believe) of the flux bias line. The amplitudes in 6.1 are NOT going to zero for certain directions as one would expect, which means that the flux bias line likely is responding to the signal and screwing up the measurements. For some comparison, here is a picture of two antennas spaced half a wavelength apart, with the SAME delayed signal added to each (this should not entirely make sense, since the delayed flux bias line signal should be FURTHER delayed for the second antenna due to the additional physical separation. If that additional phase is accounted for then nothing interesting happens and the graph looks just like figure 6.2. This is just an attempt to try and recreate what is observed in figure 6.1.)

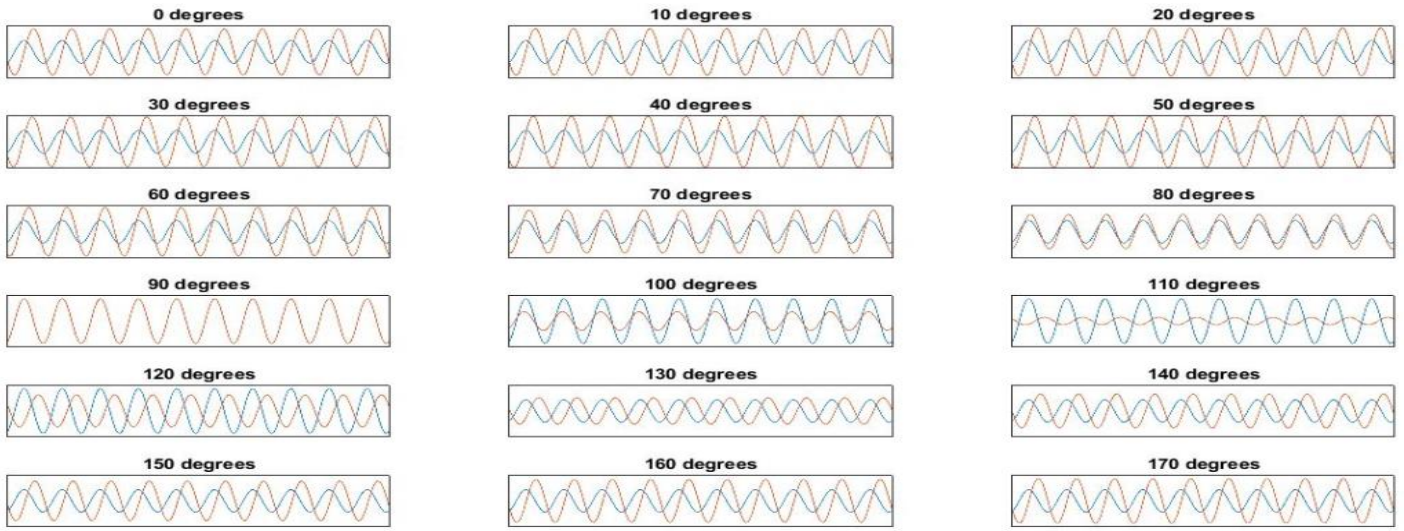

FIGURE 6.5. An attempt to recreate the SQUID outputs by interfering phases.

The blue signal is not changing here (since this is the first reference antenna) even with the added delayed signal on top of it. The red signal, however, changes phase for each angle. Thus the fixed delayed signal on top of it causes varying degrees of interference and hence makes the amplitude fluctuate. What results is behavior that is similar to what happens in the experiment.

6.2. **MUSIC performance in a SQUIDy environment.**

Relevant Programs: `Baseline2meters.m, SingleFrequencies.m, MUSICtriangle.m`

In this section, a general analysis of performance is done on the MUSIC algorithm. Performance studies of a popular method like MUSIC have been done already, so the point of this section is to probe just how effective MUSIC is under very bad circumstances. These bad circumstances are specific to a SQUID setup - they mainly involve a low number of antennas and a large number of interference signals since the SQUID has an enormous bandwidth and such complications are inevitable. The main accomplishment here is that a fairly good metric for performance is introduced (in my opinion) that sums things up nicely. To the person reading this, I challenge you to find a DF paper that details the performance of an algorithm and specifically talks about performance under the stress of multiple interference signals. I say this because of all the research I have done, I have never seen such a paper. Why is this? Is it because interfering signals are not a problem and can just be filtered out? Or maybe because normal antennas have a small bandwidth, so they don't pick up signals from different sources? I'm really not sure. The fact that this practice is not prevalent in published papers makes me think that the focus of this section is somehow pointless or wrong. Regardless, it makes sense to me to shine a light on this issue and see what happens.

For the first experiment, we model a linear array of antennas (SQUID antenna or not, the distinction is arbitrary) spaced at exactly 2 centimeters apart. A total of 9 signals are generated: the first being the "target" signal that we wish to find, and 8 others coming from azimuths of 20, 40, 60, 80, 100, 120, 140, and 160 degrees respectively. The frequency of the target signal is decided, then the 8 other interference signals' frequencies are uniformly distributed in a band that is within 10% of the target frequency. A signal-to-interference ratio (SIR) is the main parameter that determines the effectiveness of MUSIC under the stress of having interference signals. The SIR is controlled simply by varying the amplitude of the signal sinusoids. After the signals are made, MUSIC is activated and tries to find the azimuth of the target signal. This constitutes a single trial. There are too many possible parameters to vary here - number of antennas, SIR, SNR, choice of target and interference frequencies, and choice of target and interference azimuths. To deal with this, SNR is set in stone at 20 dBs, and the number of antennas (2, 3, 5, and 10) constitute new experiments. For the frequency and azimuths, I tackled this by randomly generating everything, then repeating each trial 1000 times and averaging the results. I felt that this would give a good, general view of the error results.

Figure 6.6 shows what happens when the SIR is decreased into oblivion. MUSIC performs well until around 20 decibels and then slowly breaks. Figure 6.7 shows frequencies that are entirely inappropriate for such a small baseline, hence the terrible errors. Plenty of more graphs of other frequency ranges will be available with the data provided with this report. They are omitted here mainly because I find them largely redundant - every other example is a slight variation on this and can be largely predicted even without looking at a graph. For instance, the 2 centimeter baseline will be worse and worse for frequencies below the GHz range, because it is too small with respect to the wavelength. Also, for frequencies in the 10's of GHz and above, the errors will be extremely bad because the 2 centimeter baseline exceeds the half-wavelength limit of phase-based direction finding. I will not expound on the exact errors because they can be easily lifted from the graphs and also because the performance is fairly subjective based on the application it is to be used for.

If a different viewpoint is desired, then some more graphs have been prepared that keep the SIR and SNR fixed at 20 dBs but varies the baseline itself. These data points are not averaged over thousands

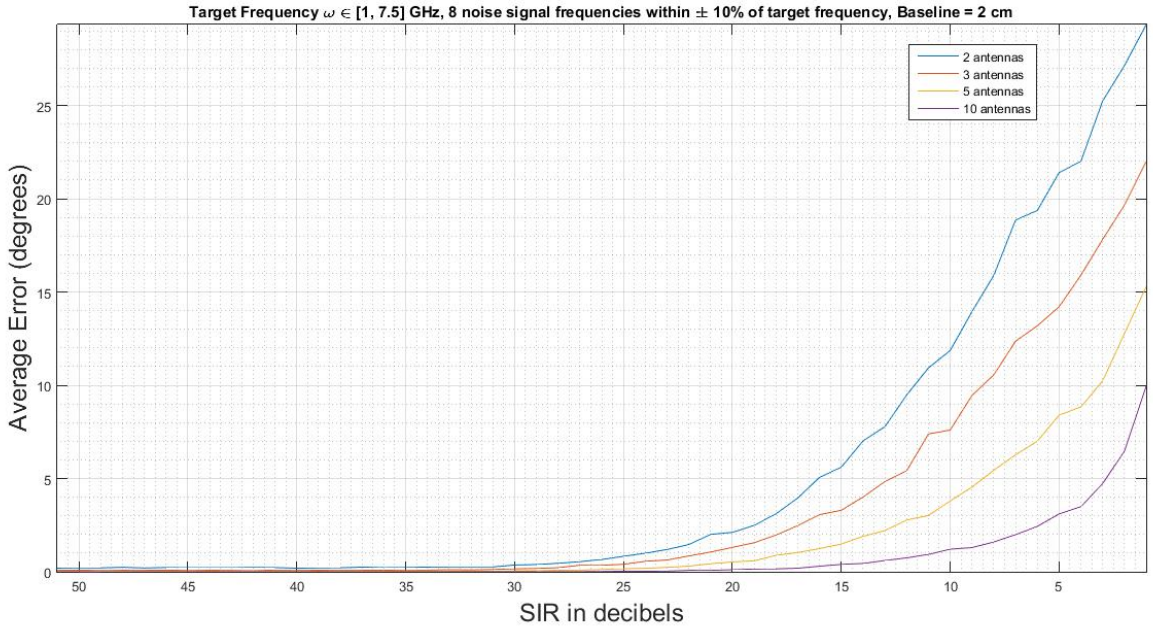

FIGURE 6.6. Target frequency within [1, 7.5] GHz, with a baseline of 2 cm. Lines represent 2, 3, 5, and 10 antennas

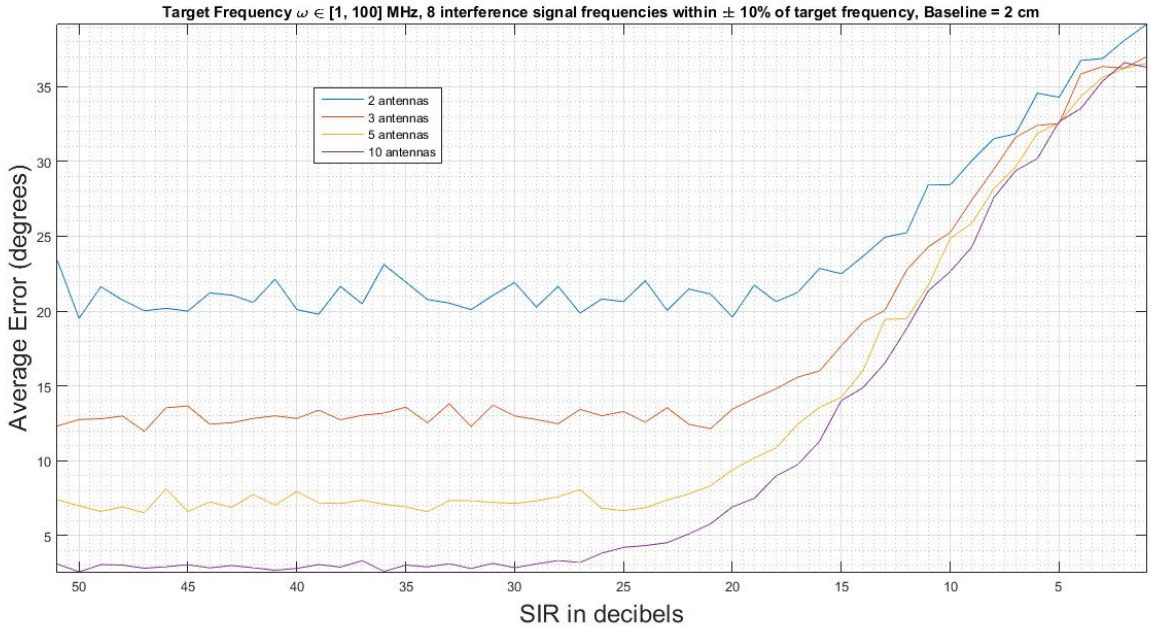

FIGURE 6.7. Target frequency within [1, 100] MHz, with a baseline of 2 cm. Lines represent 2, 3, 5, and 10 antennas

of trials. Figures 6.8, 6.9, and 6.10 show what to expect under certain circumstances. Again, there are many plots available (including different frequency ranges, baseline ranges, antennas etc.), but most are omitted.

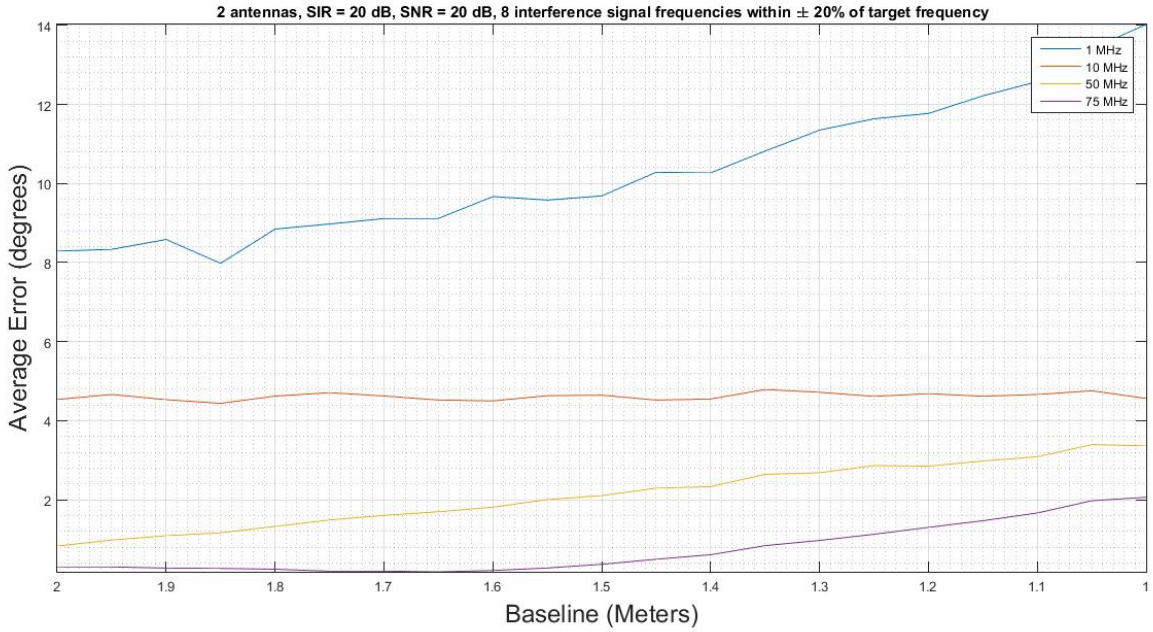

FIGURE 6.8. Baseline that decreases from 2 m to 1 m. Lines represent 1, 10, 50, and 75 MHz target frequencies.

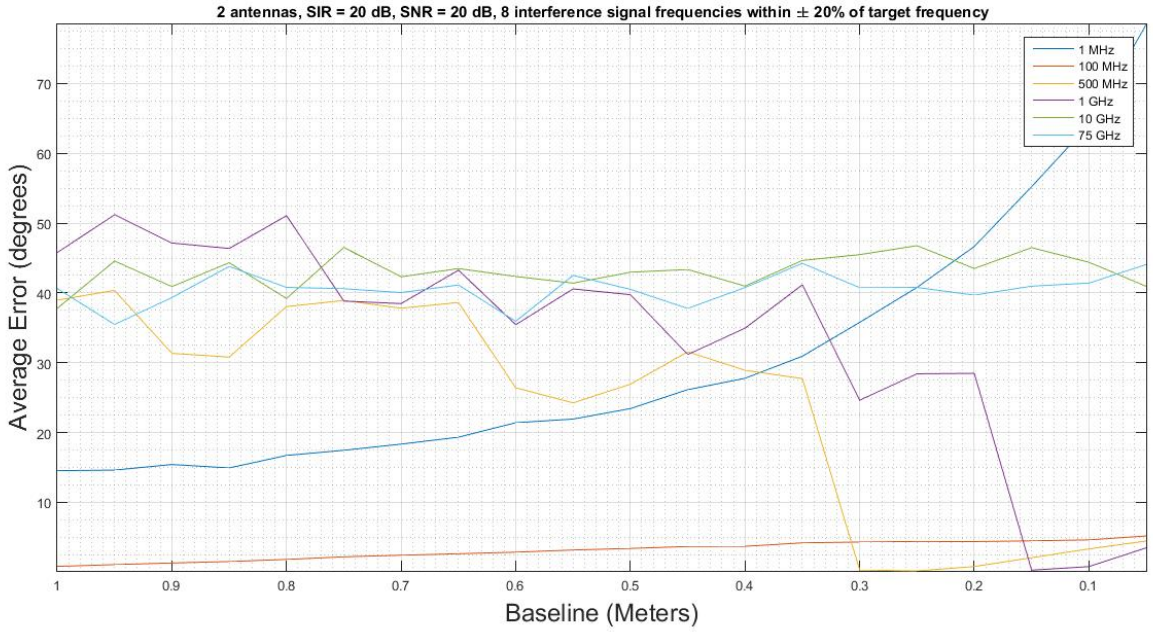

FIGURE 6.9. Baseline that decreases from 1 m to 1 cm. Lines represent 1, 100, 500, 1000, 10000, and 75000 MHz target frequencies.

These graphs were hand-picked to showcase the variation in performance in different situations. If anything, these graphs should both comfort and scare someone who wants to use phase-based MUSIC with SQUID antennas. This is because a fixed baseline

A very easy way to improve these numbers is to use an array of antennas with several different baselines, e.g., two closely spaced antenna subarrays that have a large separation between them. Systems

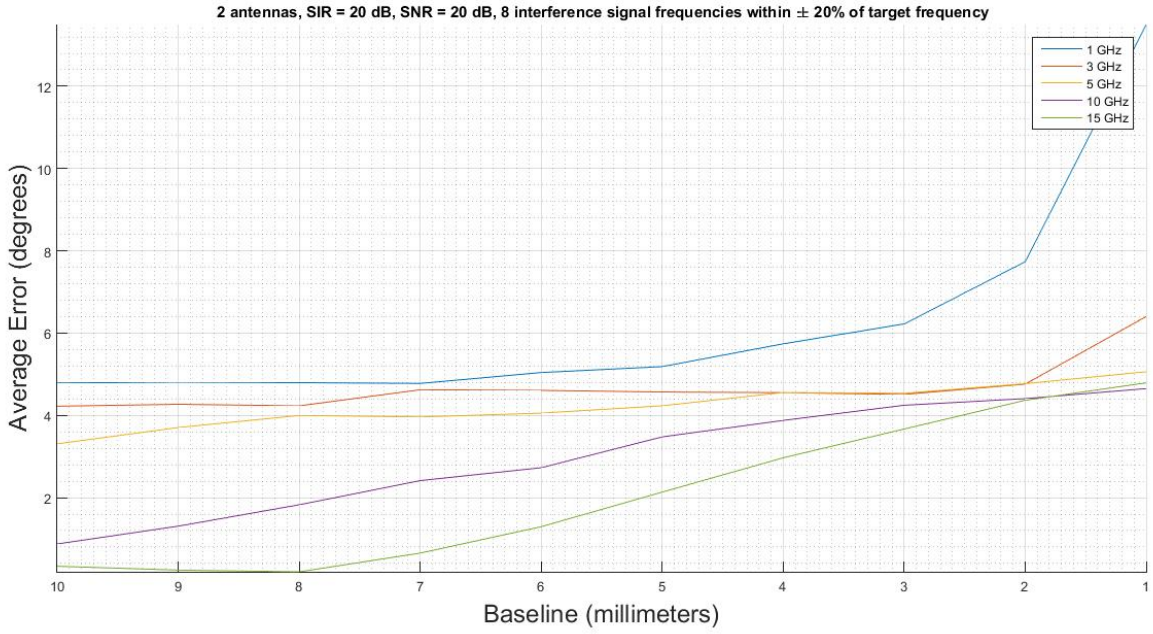

FIGURE 6.10. Baseline that decreases from 10 m to 1 mm. Lines represent 1, 2, 5, 10, and 15 GHz target frequencies.

like this can easily be made to activate pairs of antennas individually for according to a give frequency in order to maximize the array's effective frequency range. In fact, I did something quite similar with a triangular array of antennas and found that it worked just fine. I would say that this is something to consider if a linear array of SQUID antennas is being seriously considered.

A more effective way to achieve better results is to ditch the phase-shift linear array way of direction-finding entirely. This would entail mashing 3 orthogonally oriented SQUID antennas together into a single vector sensor. This creates manufacturing complications (not to mention biasing each with the same amount of magnetic flux might turn out to be extremely hard) but the payoff would be much more effective and compact direction-finding machine. Major perks would be that the baseline is rendered nonexistant and the antenna becomes entirely frequency independent - essentially eliminating all the problems seen in this section. Colocating 3 SQUIDs might be very hard to do, but I am confident that tweaks to certain algorithms will allow for them to be separated without much issues. There are quite a few direction-finding algorithms for vector sensors out already. One is detailed in the next section (although is not ideal for SQUIDs) and another is derived from DR-MUSIC and is modified by myself.

6.3. **Uni-Vector-Sensor ESPRIT.**

Relevant Programs: `UniVectorSensor_ESPRIT.m, UVS_ESPRIT_many_signals.m`

The following is not my work and is simply being included as a fun bonus because I made a working program of it. The theory is lifted from [6]. Uni-Vector-Sensor ESPRIT is exactly what its name entails: using the ESPRIT algorithm on multiple antennas mashed together into a single 'vector' sensor to find the direction of a signal. It is important to note that in general, ESPRIT is preferable over MUSIC because it is faster (does not require searching) and performs just as well. This algorithm is cool because ESPRIT typically operates on two separated arrays of antennas, whereas this algorithm works around that. In [234], two time-delayed sets of data are collected, and these two data sets function in a similar way as if the data sets were collected from two spatially separated arrays of antennas. This is a great technique, and is what inspired my modification to DR-MUSIC that is detailed in section 6.4. Here is a very quick run through of the theory behind Uni-Vector-ESPRIT. Refer to [234234] for more details

Uni-Vector-Sensor ESPRIT uses 3 orthogonally oriented magnetic field and 3 orthogonally oriented electric field sensors all colocated into a single vector sensor. The steering vector for the kth incoming signal $\mathbf{a}_k$ is thus a 6×1 vector that contains complete electric and magnetic information of the signal. The parameters contained in $\mathbf{a}_k$ include $\theta_k, \phi_k, \gamma_k$, and η_k, which correspond to the kth signal's elevation, azimuth, auxiliary and phase polarization angles, respectively. A total of K steering vectors relating to K signals are grouped together to form a steering matrix $\mathbf{A}$. N snapshots of data are collected from the vector sensor in two bursts with an arbitrary time-delay in between. These two data sets are called $\mathbf{Z}_1$ and $\mathbf{Z}_2$ which can are spanned by steering matrices $\mathbf{A}_1$ and $\mathbf{A}_2$. Since $\mathbf{A}_1$ and $\mathbf{A}_2$ are distinguished by a single time-delay, we can say that $\mathbf{A}_2 = \mathbf{A}_1\mathbf{\Phi}$, where $\mathbf{\Phi}$ is a diagonal matrix of phase-shifts. The data covariance matrices of both $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are estimated and then eigendecomposition is performed, yielding matrices $\mathbf{E}_1$ and $\mathbf{E}_2$. These matrices are both comprised of K eigenvectors corresponding to the K largest eigenvalues of the first and second data sets, respectively (dubbed signal-subspace eigenvector matrices). Since $\mathbf{E}_1$ and $\mathbf{E}_2$ span the same spaces as $\mathbf{A}_1$ and $\mathbf{A}_2$ (respectively), there exists a nonsingular matrix $\mathbf{T}$ such that

$$\begin{aligned} \mathbf{E}_1 &= \mathbf{A}_1\mathbf{T}, \\ \text{and } \mathbf{E}_2 &= \mathbf{A}_2\mathbf{T} = \mathbf{A}_1\mathbf{\Phi}\mathbf{T} \end{aligned}$$

Because $\mathbf{E}_1$ and $\mathbf{E}_2$ are full-rank, there exists a unique nonsingular matrix $\mathbf{\Psi}$ such that

$$\begin{aligned} \mathbf{E}_1\mathbf{\Psi} = \mathbf{E}_2 &\rightarrow \mathbf{A}_1\mathbf{T}\mathbf{\Psi} = \mathbf{A}_1\mathbf{\Phi}\mathbf{T} \\ &\rightarrow \mathbf{\Psi} = (\mathbf{E}_1^H\mathbf{E}_1)^{-1}(\mathbf{E}_1^H\mathbf{E}_2) = \mathbf{T}^{-1}\mathbf{\Phi}\mathbf{T} \\ &\rightarrow \mathbf{\Phi} = \mathbf{T}\mathbf{\Psi}\mathbf{T}^{-1}. \end{aligned} \tag{6.1}$$

Since $\mathbf{\Phi}$ is diagonal, linear algebra and (6.1) say that $\mathbf{T}$ is composed of the eigenvectors of $\mathbf{\Psi}$. Thus we reach the main result:

$$\begin{aligned} \hat{\mathbf{A}}_1 &= \mathbf{E}_1\mathbf{T}^{-1} = \mathbf{E}_2\mathbf{T}^{-1}\mathbf{\Phi}^{-1} \\ &= \frac{1}{2}\{\mathbf{E}_1\mathbf{T}^{-1} + \mathbf{E}_2\mathbf{T}^{-1}\mathbf{\Phi}^{-1}\} \end{aligned} \tag{6.2}$$

(6.2) is apparently necessary to achieve the best estimate. After the estimated steering matrix $\hat{\mathbf{A}}_1$ is obtained, a straight-forward cross product is performed which produces the Poynting vector and subsequently provides the azimuth and elevation estimations.

`UniVectorSensor_ESPRIT` is a bare program that very clearly follows along this theory and produces extremely reliable and fast estimates. However, how does it perform when other unwanted signals are bombarding it? This was attempted in `UVS_ESPRIT_many_signals.m` which simulates the effects of up to 5 signals at once. For finding all signals simultaneously, the program performs similarly to MUSIC; namely, it locates all the signals even up to very low SNR levels. When trying to find one signal amongst up to 4 other interference signals, the program can reliably find the signal up to around an SIR of 15

decibels. This is again fairly similar performance to MUSIC (although this program performs much faster). In general, this algorithm seems to be a viable alternative to MUSIC for direction finding. In fact, many people prefer ESPRIT.

Some bugs that I should mention are that the results stated above can only be applied to the program's ability to locate a signal's azimuth. In many cases, the program cannot locate the signal's elevation correctly. I refer to it as 'the program' and not the algorithm itself because it is entirely possible that my execution of the algorithm was poor and is to blame for the bad results. The main reason I consider this possibility is because the transition from `UniVectorSensor_ESPRIT` to `UVS_ESPRIT_many_signals` was rife with errors. I could not get the latter program to work at all whenever I input more than one signal. After much trial and error, I was finally able to fix it by changing (6.2). In `UVS_ESPRIT_many_signals`, equation (6.2) is actually written as $\hat{\mathbf{A}}_1 = \frac{1}{2}(\{\mathbf{E}_1\mathbf{T}+\mathbf{E}_2\mathbf{T}\mathbf{\Phi}^{-1}\})$. In case you didn't catch it, the program uses $\mathbf{T}$ instead of $\mathbf{T}^{-1}$. Weird! Why does this happen? I honestly have no idea. But it seems to get the job done for the most part so I don't care too much.

In conclusion, this section provides a viable alternative to MUSIC and hopefully makes the reader consider the possibility of using a vector sensor over a linear array (In fact, if using a linear array you can still use ESPRIT with `NewESPRIT.m`). Downsides to this approach are that electric sensors are used and the fact that the programs I wrote may not be the perfect implementation to it.

This brings us to the final section. This next section introduces a DF method that I fashioned from DR-MUSIC coupled with the time-delayed data samples used in the algorithm above. I went through the trouble of doing this for a few reasons: 1) Uni-Vector-Sensor ESPRIT requires electric sensors which does not work in a SQUID setup, 2) ESPRIT itself in other forms does not perform very well with interference signals in general, 3) MUSIC in a linear array does not perform fantastically well and requires large antenna separations for low frequency signals, 4) DR-MUSIC is a frequency-independent and works with only magnetic field components - but is completely incapable of finding a signal when interference signals are present. The last point was not discussed in this report, but I did do some tests with DR-MUSIC and found that it breaks when unwanted signals are present in the data. I wanted to take DR-MUSIC and give it the capability to discriminate between signals when it is trying to do a search. Luckily, the experiment succeeded. I have considered doing the same with an ESPRIT algorithm but unfortunately I am at the end of my internship and do not have the time to do so. I have a few ideas on how I could create more effective algorithms, so hopefully I will get the chance to work on that in the future.

6.4. **DR-MUSIC Method with Frequency Selection.**

Relevant Programs: `Selective_DR_MUSIC.m, MultipleDRMUSIC.m`

The following is a direction-finding algorithm that has been specifically crafted for use in a SQUID sensor array of antennas. The desired properties of such an antenna system involve: 1. a very small or no baseline, 2. an ability to locate signals of any frequency and 3. a strong capability to locate specific signals when many other interfering signals are present. To accomplish this, DR-MUSIC is used in conjunction with many time-delayed data sets (very similar to what is found in the previous section). The result is a method that accomplishes the above three criteria even in sub-zero SNR and SIR dB ranges. The following theory is presented for a electromagnetic vector sensor featuring 3 electric and 3 magnetic sensors. Comments about the strictly magnetic case will be made in the last section.

6.4.1. *Array Model.*

This method involves the use of three spatially colocated identical but orthogonally oriented, electrically short dipoles and magnetically small loops measuring all three electric field components and all three magnetic components of incident signals.

Suppose that the kth ($1 \leq k \leq K$) narrow-band, transverse completely polarized electromagnetic plane-wave signal, having traveled through a homogenous isotropic medium, impinges upon the vector sensor. For an incoming unit-power signal, the vector sensor manifold can be expressed as

$$\begin{aligned}
\mathbf{a}_k &= \begin{bmatrix} e_x(\theta_k,\phi_k,\gamma_k,\eta_k) \\ e_y(\theta_k,\phi_k,\gamma_k,\eta_k) \\ e_z(\theta_k,\phi_k,\gamma_k,\eta_k) \\ h_x(\theta_k,\phi_k,\gamma_k,\eta_k) \\ h_y(\theta_k,\phi_k,\gamma_k,\eta_k) \\ h_z(\theta_k,\phi_k,\gamma_k,\eta_k) \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} \cos\theta_k\cos\phi_k & -\sin\phi_k \\ \cos\theta_k\sin\phi_k & \cos\phi_k \\ -\sin\theta_k & 0 \\ -\sin\phi_k & -\cos\theta_k\cos\phi_k \\ \cos\phi_k & -\cos\theta_k\sin\phi_k \\ 0 & \sin\theta_k \end{bmatrix}}_{\mathbf{\Omega}(\theta_k,\phi_k)} \underbrace{\begin{bmatrix} \sin\gamma_k e^{j\eta_k} \\ \cos\gamma_k \end{bmatrix}}_{\mathbf{g}(\gamma_k,\eta_k)}
\end{aligned}$$

where $\theta_k \in [0,\pi]$ is the signals elevation measured from the z-axis, $\phi_k \in [0,2\pi]$ is the azimuth angle, $\gamma_k \in [0,\pi/2]$ is the auxiliary polarization angle and $\eta \in [-\pi,\pi]$ is the polarization phase difference.

6.4.2. *Data Model.*

Any array of antennas will be subjected to signals that it does not intend to locate. This method specifically finds any number of target signals via discrimination of the carrier frequencies of all signals. For K impinging signals (target and interference signals), L time-delayed sets of data are collected by one vector sensor, each consisting of N snapshots. This method can locate up to any number of signals with distinct frequencies amongst any number of ambient interference signals if a sufficient number of data sets are recorded. However, this method can locate a maximum of 5 signals of the same carrier frequency at once. Thus, the kth monochromatic signal creates L $6 \times N$ data sets

$$\begin{aligned}
\mathbf{a}_k s(t_n, f_k), \quad n = 1, ..., N \\
\mathbf{a}_k s(t_n + \Delta_1, f_k) &= \mathbf{a}_k s(t_n, f_k) e^{j2\pi f_k \Delta_1} \\
\mathbf{a}_k s(t_n + \Delta_2, f_k) &= \mathbf{a}_k s(t_n, f_k) e^{j2\pi f_k \Delta_2} \\
&\vdots \\
\mathbf{a}_k s(t_n + \Delta_{L-1}, f_k) &= \mathbf{a}_k s(t_n, f_k) e^{j2\pi f_k \Delta_{L-1}}
\end{aligned}$$

where f_k is the kth signal's frequency, $s(t_n, f_k)$ is the signal itself, and $\{\Delta_1, ..., \Delta_{L-1}\}$ is a set of uniformly distributed random time delays. Give sufficiently small time-delays, the data sets can be overlapped. Note that when f_k and Δ_i are known, all $L-1$ phase differences are exactly known and is a key to the method's selective process.

Given an unbounded number K of signals impinging on the array with additive zero-mean Gaussian white noise, the data taken in by the vector sensor takes the form

$$\mathbf{z}(t_n) = \sum_{k=1}^{K} \begin{bmatrix} \mathbf{a}_k s(t_n, f_k) \\ \mathbf{a}_k s(t_n, f_k) e^{j2\pi f_k \Delta_1} \\ \vdots \\ \mathbf{a}_k s(t_n, f_k) e^{j2\pi f_k \Delta_{L-1}} \end{bmatrix} + \mathbf{n}(t_n)$$

With the following definitions,

$$\begin{aligned}
\mathbf{A}_1 &\triangleq [\mathbf{a}_1, ..., \mathbf{a}_K], \\
\mathbf{A}_2 &\triangleq [\mathbf{a}_1 e^{j2\pi f_1 \Delta_1}, ..., \mathbf{a}_K e^{j2\pi f_K \Delta_1}], \\
&\vdots \\
\mathbf{A}_L &\triangleq [\mathbf{a}_1 e^{j2\pi f_1 \Delta_{L-1}}, ..., \mathbf{a}_K e^{j2\pi f_K \Delta_{L-1}}] \\
\mathbf{s}(t_n) &\triangleq \begin{bmatrix} s(t_n, f_1) \\ \vdots \\ s(t_n, f_K) \end{bmatrix} \\
\mathbf{n}(t_n) &\triangleq \begin{bmatrix} n_1(t_n) \\ \vdots \\ n_{3\times L}(t_n) \end{bmatrix}
\end{aligned}$$

we can compactly write the entire data collected by the vector sensor as

$$\begin{aligned}
\mathbf{z}(t_n) &= \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_L \end{bmatrix} \mathbf{s}(t_n) + \mathbf{n}(t_n) \\
\mathbf{Z} &\triangleq [\mathbf{z}(t_1), ..., \mathbf{z}(t_N)].
\end{aligned}$$

Here, $\mathbf{Z}$ is a $(6 \times L) \times N$ matrix of data.

6.4.3. *DR-MUSIC method.*

We first begin with an exposition of the normal DR-MUSIC method [5] to gain perspective and form a basis of understanding. This will make the modification made by DR MUSIC with frequency selection more obvious. Note that DR-MUSIC does not take several data sets, meaning that $L = 1$ in this section. The covariance matrix of the data $\mathbf{Z}(t)$ is given by

$$\mathbf{R}_Z = \mathrm{E}[\mathbf{Z}\mathbf{Z}^{\mathrm{H}}] = \mathbf{A}\mathbf{R}_s\mathbf{A}^{\mathrm{H}} + \sigma^2\mathbf{I}$$

where $(\cdot)^{\mathrm{H}}$ denotes complex conjugate transpose, σ^2 is the white noise power, and $\mathbf{R}_s = \mathrm{E}[\mathbf{s}^{\mathrm{H}}(t)\mathbf{s}(t)]$ is the source covariance matrix. Due to the ergodicity of the noise, $\mathbf{R}_Z$ can be estimated as

$$\hat{\mathbf{R}}_Z = \frac{1}{N}\mathbf{Z}\mathbf{Z}^{\mathrm{H}}.$$

DR-MUSIC begins by executing an eigendecomposition on the matrix $\hat{\mathbf{R}}_Z$. The resulting eigenvectors decompose the matrix into a K-dimensional signal subspace and an $(6-K)$ dimensional noise subspace. Let $\mathbf{U}_n$ represent the $6 \times (6-K)$ matrix of noise eigenvectors associated with the $6-K$ smallest eigenvalues of $\hat{\mathbf{R}_Z}$. The MUSIC spectrum is then given as

$$\mathbf{P}_{\mathrm{MU}}(\hat{\theta_k}, \hat{\phi_k}, \hat{\gamma_k}, \hat{\eta_k}) = \left[\frac{\mathbf{a}^H\mathbf{U}_n\mathbf{U}_n^H\mathbf{a}}{\mathbf{a}^H\mathbf{a}^H}\right]^{-1}$$

The peaks of this spectrum will correspond to the desired angles to be found. Thus, the searching involved in MUSIC equates to

$$\mathbf{P}_{\mathrm{MU}}(\hat{\theta_k}, \hat{\phi_k}, \hat{\gamma_k}, \hat{\eta_k}) = \left[\min_{\theta,\phi,\gamma,\eta} \mathbf{a}^H\mathbf{U}_n\mathbf{U}_n^H\mathbf{a}\right]^{-1}. \tag{6.3}$$

This four-dimensional search is very computationally expensive. DR-MUSIC shrinks this search to only two dimensions via use of the Rayleigh-Ritz theorem. Since

$$\mathbf{a}(\theta, \phi, \gamma, \eta) = \mathbf{\Omega}(\theta, \phi)\mathbf{g}(\gamma, \eta),$$

and $\mathbf{g}^H\mathbf{g} = 1$, (6.3) can be rewritten as

$$\mathbf{P}_{\text{MU}}(\hat{\theta_k}, \hat{\phi_k}, \hat{\gamma_k}, \hat{\eta_k}) = \left[\min_{\theta,\phi,\gamma,\eta} \frac{\mathbf{g}^H\mathbf{\Omega}^H\mathbf{U}_n\mathbf{U}_n^H\mathbf{\Omega}\mathbf{g}}{\mathbf{g}^H\mathbf{g}}\right]^{-1}. \tag{6.4}$$

Letting $\mathbf{B}(\theta, \phi) = \mathbf{\Omega}^H\mathbf{U}_n\mathbf{U}_n^H\mathbf{\Omega}$, (6.4) becomes

$$\mathbf{P}_{\text{MU}}(\hat{\theta_k}, \hat{\phi_k}, \hat{\gamma_k}, \hat{\eta_k}) = \left[\min_{\theta,\phi,\gamma,\eta} \frac{\mathbf{g}^H\mathbf{B}(\theta,\phi)\mathbf{g}}{\mathbf{g}^H\mathbf{g}}\right]^{-1}. \tag{6.5}$$

Since $\mathbf{B}$ is Hermitian and $\mathbf{g}^H\mathbf{g} = 1$, the right-hand side of (6.5) is a perfect candidate for the Rayleigh-Ritz theorem, which then transforms the minimization problem to finding the minimum eigenvalue of $\mathbf{B}$, i.e.,

$$\mathbf{P}_{\text{DR}}(\hat{\theta}_k, \hat{\phi}_k, \hat{\gamma}_k, \hat{\eta}_k) = \left[\min_{\theta,\phi} \lambda_{min}(\mathbf{B}(\theta,\phi))\right]^{-1}.$$

Since $\mathbf{B}$ only depends on θ and ϕ, the 4-dimensional search has been shrunk to 2 dimensions.

6.4.4. *Selective DR-MUSIC method.*

Now, we attempt to make a major modification to DR-MUSIC in order to enable it to find signals amongst any number of interference signals. The first step is to record $L > 1$ sets of data in order to gain more information about the electromagnetic spectrum as was described in section 6.4.2. This causes the matrix $\mathbf{R}$ to grow to a $(6\times L)\times(6\times L)$ matrix and $\mathbf{U}$ to increase to size $(6\times L)\times(6\times L)-K$. This increase in $\mathbf{U}$ has huge impact on the performance and amount of information at the method's disposal. For DR-MUSIC, $\mathbf{U}$ is a $6\times(6-K)$ matrix, meaning that if at least one noise vector is to be used for direction-finding, then no more than $K = 5$ signals can be located at the same time. By taking L data sets, the requirement for having at least one noise eigenvector for direction finding now depends on L, namely, $L \geq (1+K)/6$ (although it is recommend that L be greater). This is incredible, because a sufficiently large enough number of data sets L means that any number K of signals with distinct frequencies can be located simultaneously. The method is still limited to 5 simultaneous signals of the same carrier frequency. However, any number of frequency groups can still be located, e.g., 5 signals at 30 MHz and 5 signals at 31 MHz can still be located simultaneously with no issues.

With $\mathbf{U}_n$ now a $(6\times L)\times(6\times L)-K$ matrix, this means that $\mathbf{\Omega}$ must now be a $(6\times L)\times M$ matrix, where M is any number. We accomplish this by injecting frequency-specific phase information into $\mathbf{\Omega}$. For the set of frequencies $F = \{f_1, ..., f_m\}$ where m is the number of *distinct* frequencies in the set $\{f_1, ..., f_K\}$ of all signals, choose a set $A \subseteq F$ of frequencies corresponding to the signals to be located. For $|A| = D$, order these frequencies into the set $\{f_1, ..., f_D\}$. Now, we construct the matrices

$$\mathbf{\Psi} \triangleq \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \mathrm{e}^{j2\pi f_1\Delta_1} & \mathrm{e}^{j2\pi f_2\Delta_1} & \cdots & \mathrm{e}^{j2\pi f_D\Delta_1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{e}^{j2\pi f_1\Delta_{L-1}} & \mathrm{e}^{j2\pi f_2\Delta_{L-1}} & \cdots & \mathrm{e}^{j2\pi f_D\Delta_{L-1}} \end{bmatrix}$$

$$\text{and} \qquad \widehat{\mathbf{\Omega}} \triangleq \mathbf{\Psi} \otimes \mathbf{\Omega}.$$

$\widehat{\mathbf{\Omega}}$ is now a $(6\times L)\times(2\times D)$ matrix that is suitable for searching and contains all the phase information of each signal relative to its frequency at every time-delay. It should be clear now that including only distinct frequencies from the K signals was essential to ensure that $\mathbf{\Psi}$ and $\widehat{\mathbf{\Omega}}$ are full rank. With $\widehat{\mathbf{B}}(\theta,\phi) = \widehat{\mathbf{\Omega}}^H\mathbf{U}_n\mathbf{U}_n^H\widehat{\mathbf{\Omega}}$, the new spectrum becomes

$$\mathbf{P}_{\text{SDR}}(\hat{\theta}_k, \hat{\phi}_k, \hat{\gamma}_k, \hat{\eta}_k) = \left[\min_{\theta,\phi} \lambda_{min}(\widehat{\mathbf{B}}(\theta,\phi))\right]^{-1}.$$

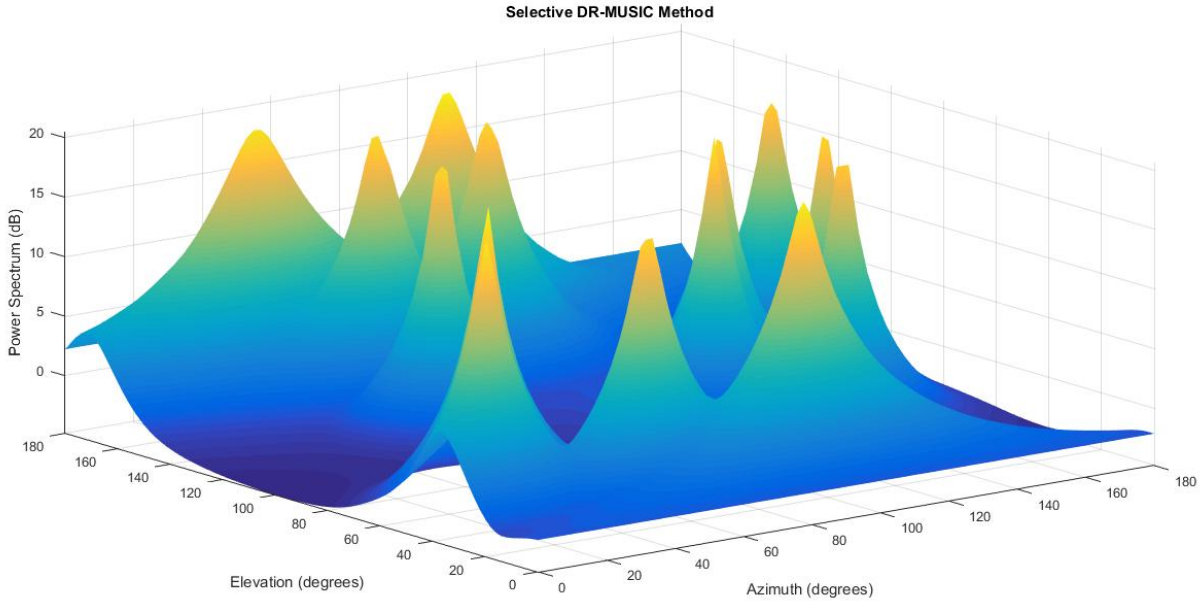

FIGURE 6.11. New algorithm locating 15 signals out of 20 ambient signals with random directions where SIR $= -100$ dBs and SNR $= -10$ dBs. This performance is several orders of magnitude better than other methods I am aware of. You will never see another spectrum like this!

With this modification, Selective DR-MUSIC effectively finds the directions of D signals even in sub-zero SNR and SIR decibel levels. At the time of this writing, I am not aware of any other direction-finding algorithm in literature that is capable of this. Direction-finding algorithms are typically quite sensitive to these decibel levels and can never find more than $T-1$ signals simultaneously, where T is the number of antenna elements. It is actually quite unbelievable how effective this method is.

6.4.5. *Difficulties and SQUID implementation.*

Theoretically, for T antennas, the algorithm is able to locate up to $T-1$ signals of the same carrier frequency simultaneously. The writing here does not explicitly show this. I state this now and am confident of this fact because every direction-finding algorithm capable of locating multiple signals (including the ones this new algorithm is built upon) state this fact. However, if you run `Selective_DR_MUSIC` with signals where more than 4 signals share the same carrier frequency, the algorithm will break. My hypothesis is that this new $T-2$ limit is not specific to this algorithm but is actually a bug in my personal implementation of the code. My evidence for this is that the publication for the DR-MUSIC method itself states that $T-1=5$ signals of the same carrier frequency can be located simultaneously. However, a look at figure 6.13 shows what happens when I attempt this. The result is a totally broken spectrum. For some mysterious reason, my program can only locate $T-2$ signals at a time. Why is this happening? I have no idea. This same problem also plagues `UVS_ESPRIT_many_signals.m` whos publication also states that it should be able to locate 5 signals concurrently. Since `Selective_DR_MUSIC.m` is built off of my existing code for DR-MUSIC, I am led to believe that the $T-2$ signal limit is an artifact of faulty code.

Despite the faults in the code, this method also works for magnetic sensors only. Doing this is incredibly simple, just delete the top 3 entries of $\mathbf{a}_k$ and change every '6' you see with a '3'. A command in `Selective_DR_MUSIC.m` streamlines this process. Due to the bug in the code, it is only possible to find $3-2=1$ signals of the same carrier frequency simultaneously, even though it should be able to find

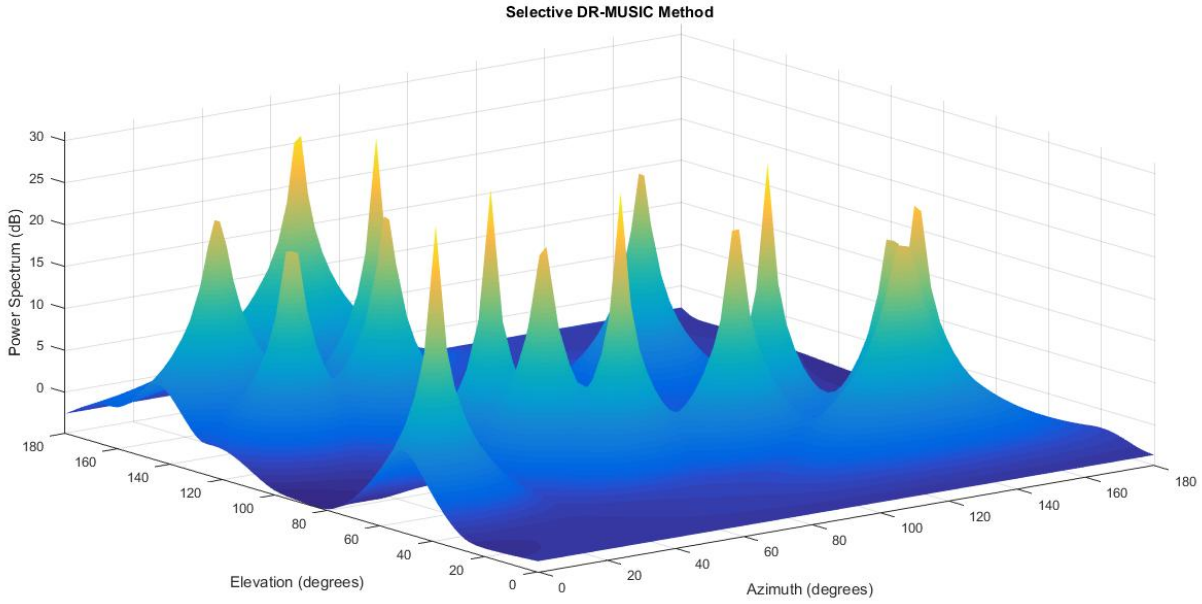

FIGURE 6.12. Similar to figure 6.11 but with SIR = 0 dBs and SNR = 0 dBs.

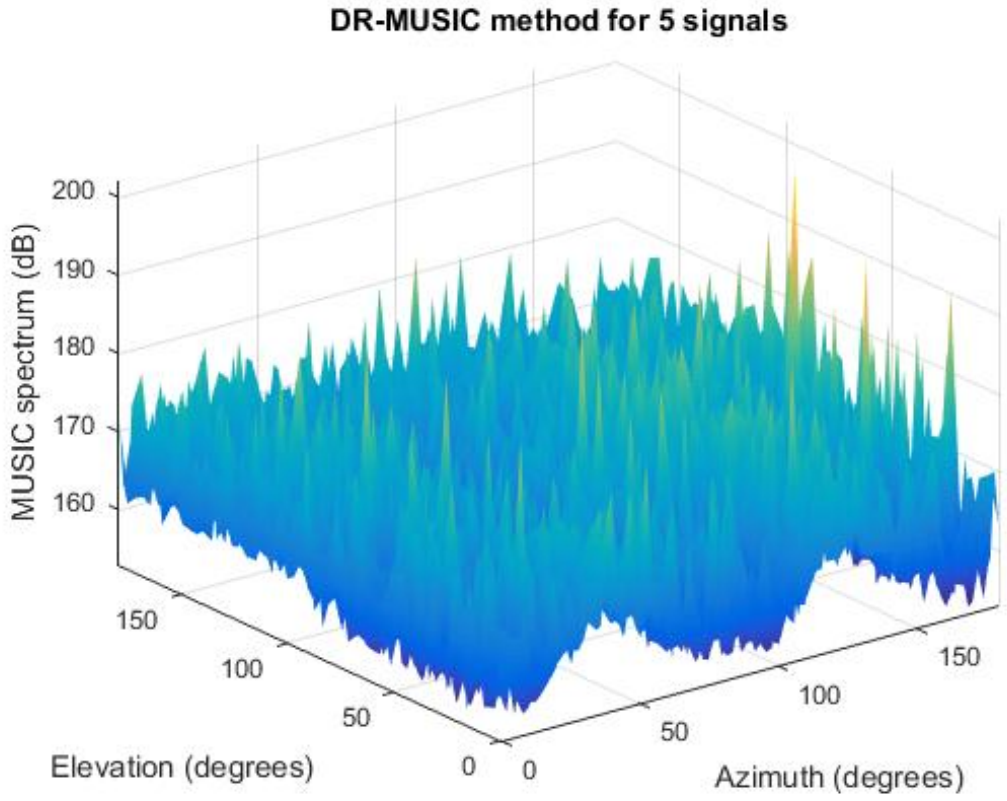

FIGURE 6.13. My faulty DR-MUSIC program trying to find 5 signals unsuccessfully.

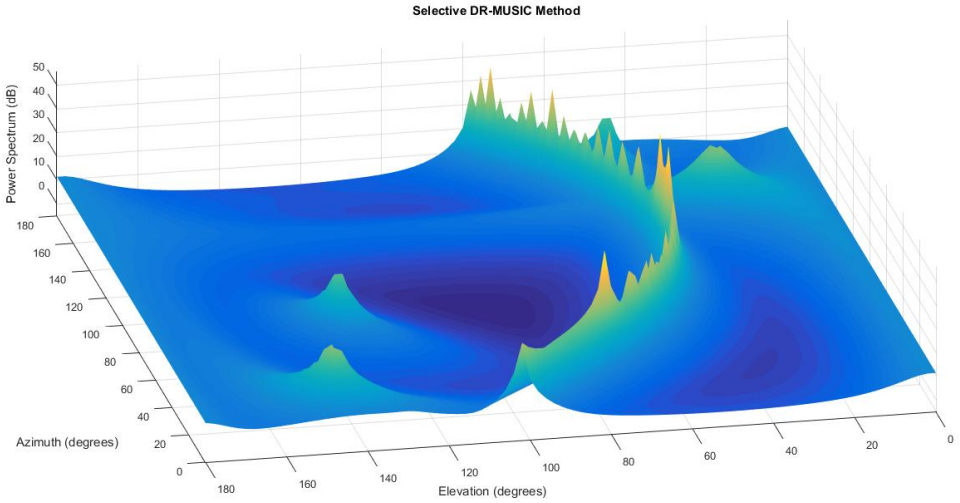

FIGURE 6.14. Fringe effects occur sometimes when attempting to find multiple signals with just a magnetic sensor.

2. This method in the strictly magnetic case performs just fine with finding a single signal at a time among many ambient interference signals. However, even though it is capable of finding more than one signal with distinct frequencies at once, the results can be strange. Specifically, an interesting 'fringing' effect occurs. Figure 6.14 showcases this. It is quite ugly, but keep in mind that it took me about 5 runs to come up with a result this bad - usually the peaks are clean and usable. I find this effect to be fascinating, as the fringe effects always create clear, sweeping lines that intersect each other at the desired peaks. I imagine that there is a cool theoretical explanation for this. Lastly, the magnetic sensor cannot sweep the full 360 degree azimuth spectrum as is the case in the full electromagnetic case. The magnetic sensor is limited to a 180 degree view.

And that concludes just about every significant thing I did during my 14 weeks here. I hope that it was worth the $15120 I was paid for it!

References

[1] Taylor, John R. *An Introduction to Error Analysis*. University Science books, 1982. (page 73)
[2] Balanis, Constantine A. *Antenna Theory: Analysis and Design*. John Wiley & Sons, 2005.
[3] Griffiths, David J. *Introduction to Electrodynamics, 3rd Edition*. Prentice Hall, 1999.
[4] Schmidt, V. V. *The Physics of Superconductors*. Springer, 1997.
[5] Lanmei Wang, Le Yang, Guibao Wang, Zhihai Chen, and Minggao Zou, "Uni-Vector-Sensor Dimensionality Reduction MUSIC Algorithm for DOA and Polarization Estimation," *Mathematical Problems in Engineering*, vol. 2014, Article ID 682472, 9 pages, 2014. doi:10.1155/2014/682472
[6] Kainam Wong, Michael Zoltowski, "Uni-Vector-Sensor ESPRIT for Multisource Azimuth, Elevation, and Polarization Estimation," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 10, October 1997