# CROAC: Counting and Recognition using Omnidirectional Acoustic Capture

Marcel Gietzmann-Sanders

July 2, 2022

# Contents

**7 Divergent Degenerates**      **51**

**8 What the Fourier**      **55**

# Chapter 1

# The Pond

It's an hour past sunset and the rain has just subsided. I approach the pond - a vernal pool that will dry up sometime this summer - trying as best I can to be a ghost. Unfortunately the crinkling of leaves reveals my corporeality and a hush descends upon the pond. It is as if the frogs recognize that their rehearsal time is over and have gone backstage in a mist of whispers as they let me get seated in anticipation of the grand event. Groping about in the darkness, as I have put out my head torch, I find the most comfortable seat in the house - a patch of dirt nestled among the roots of an old maple tree - and settle in. On the walk of mile or so in I've already heard the spring peepers chirping in all their eagerness and bullfrogs announcing their grandiose dominion with their earthly croaks, but I know that was all but a taste of what is to come. So I settle back and wait for the concert to start.

It begins with solitary croaks as the bolder of the frogs begin to test the air, seeing whether their audience still stirs. But soon enough their comrades join in and the sound begins to mount. Layer by layer I hear different species enter the ensemble. Region by region the pond gets louder and louder as the frogs regain their composure. Before I know it the air is so thick with sound I feel as if I could breath it in.

In full chorus it is impossible for me to discern the individual voices that make up this extraordinary orchestra. Instead all my two ears receive is a wall of frenetic sound. Yet the physicist in me recognizes this as a mere illusion and as I sit there bathed in chirps and croaks and wheezes I relish the richness of the data before me. I know that what my ears hear as a single curtain is in fact a richly woven fabric of pitches, amplitudes, and phases - a mathematical puzzle waiting to be untangled and solved. I realize that every

spring night, and many a summer one too, these frogs broadcast into the ether a whole host of information on position, counts, energy, species, and perhaps even lineage. All waiting to be deciphered by a discerning soul. And so as I sat there at the base of that maple tree, bathed in the ciphered data of my amphibian friends, I wondered what it would take to break the code of frogs. The adventure into mathematics, computation, and herpetology that ensued has brought me untold joy, and I hope that in the subsequent paragraphs I can give you little taste of that adventure too.

# Chapter 2

# The Richness of Sound

We begin with a picture - specifically Figure 1. This is nothing more than a picture of the simplest wave possible. If you've ever plucked a guitar string you'll know that sound is the result of vibrations. When some object like a string, or a speaker, or your very own vocal chords vibrate that vibration translates into waves in the air which we then hear as sound. Given that all sound is composed of waves like these, we can begin our journey in trying to differentiate different sound sources by understanding what makes one wave different from the next. And an excellent way to do this is to look to the richest source of sound that our ears already find intelligible - music.

## 2.1   Blue Whales Yelling - Amplitude

What makes music such a great place to start is the fact that the human mind is already so used to picking out different sections, rhythms, and instruments - an activity quite similar to what we want to do with our frog chorus. When you go to a concert you can distinguish the vocals from the percussion, and the harmony from the melody. But, depending on the kind of concert it can be much more difficult picking out what your friends are trying to say when you're already having to yell to hear yourself. And by the end you'll all be speaking in muted whispers no matter how loud you try to speak because you'll have lost your voices half way through the performance. This contrast, conveniently, is our first and simplest discriminator of sound - volume. As you raise your voice the volume of the sound that you are producing increases. The fact that you can barely hear each other speaks to the enormous volume
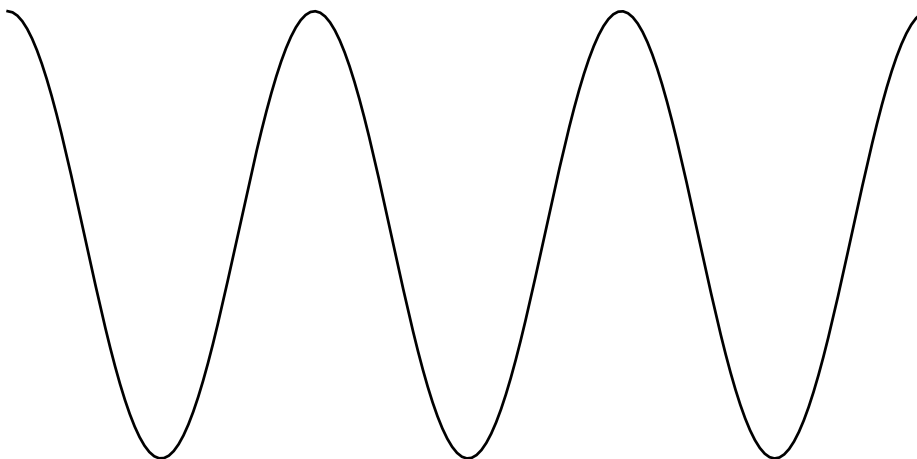
Figure 2.1: A Simple Wave

of sound coming from the stage. So how is this represented in our wave? Volume is determined by the height of our wave - the farther the undulation has to go between each peak and valley the louder the sound is. This height is referred to as the **amplitude** and to get a sense of just how variable amplitude can be let's bring a blue whale to the concert.

Blue whales are the largest animals to have ever lived. Rather unsurprisingly then, they can be rather loud. How loud? Well let's just say that blue whales would have no trouble hearing each other at a rock concert. In fact they may instead draw the ire of their fellow concert goers as they drown out the music with their voices. The typical sound level at a rock concert is around 110 decibels (dB) [1]. Blue whales on the other hand have been recorded at up to 180 dB [2]. This difference is actually far more ridiculous than it first seems because of what a decibel actually is. When a decibel measurement increases by 10 it means the sound is around 3 *times* louder. All to say that if our blue whales raised their voices, their chatter could get up to 21 times louder than the rock concert! Quite rude. Graphing this out in Figure 2 we can see that the amplitude of the blue whales yelling at one another (dashed line) is so large in comparison to the amplitude of the rock concert (solid line) that the rock concert's sound barely looks like a wave at all. Pretty incredible.
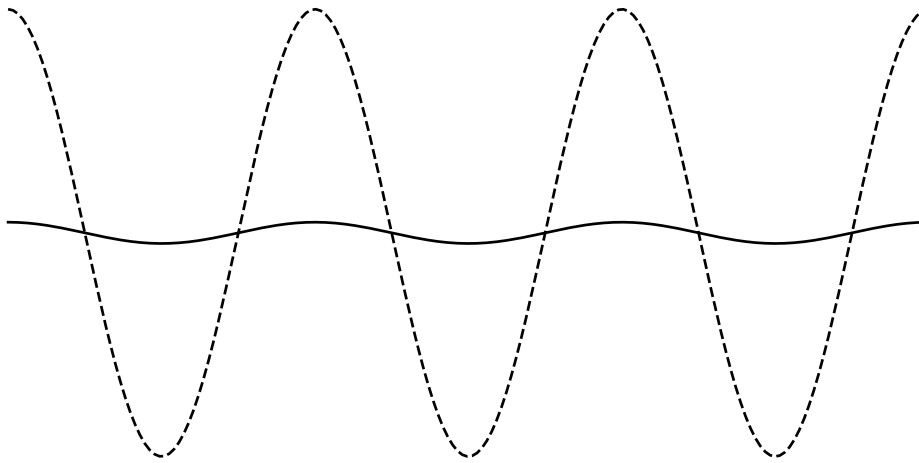
Figure 2.2: Blue Whale v. Rock Concert

## 2.2 To Tune or Not to Tune - Pitch

Okay, so amplitude is one way to distinguish one wave from another - what's up next? Well let's stop worrying about hearing our friends for a moment and turn back to the music. Remember how we said it's pretty easy for our brains to distinguish the vocalists from, say, the bass? Well there's a good reason for that - pitch. Pitch is how high or low a sound sounds. When that bass is rumbling all through your body you're experiencing a sound with a very low pitch. On the other hand when the singer hits that high note with a full crescendo that's a sound with an exhilarating high pitch. How does pitch show up in our wave? Well in contrast to amplitude - which was how high our waves got - pitch is all about how wide they are. The distance from one peak to the next is called the **wavelength** and the larger that distance is the lower the pitch of the sound. Pitch however is rarely measured in wavelengths. Instead the standard measurement used is something called Hertz (Hz). Hertz is a unit of **frequency** and frequency is simply the number of complete oscillations (peaks and troughs) per unit time. So, for example, when you hear that orchestras tune to A440 that 440 is specifically 440 Hz. Frequency and wavelength are interchangeable and tied together by a simple formula. Using the fact that we know the speed of sound $c$, if $w$ is the wavelength and $k$ is the frequency then the conversion is

simply:

$$w = c/k \tag{2.1}$$

So for example, if the speed of sound is 343 meters per second (m/s) then the wavelength of A440 is $343/440 \approx 0.78m$ or just over 2.5 feet. In comparison the highest note on a typical piano is at 7900 Hz [8] which corresponds to a *much* smaller wavelength ($343/7900 = 0.04m$). Figure 3 shows a comparison between the two with the solid line being A440 and the dashed line (which has a very very quick wiggle) being the highest note on a typical piano. Indeed the higher note has such a small wavelength it's becoming hard to see that it's a wave at all.
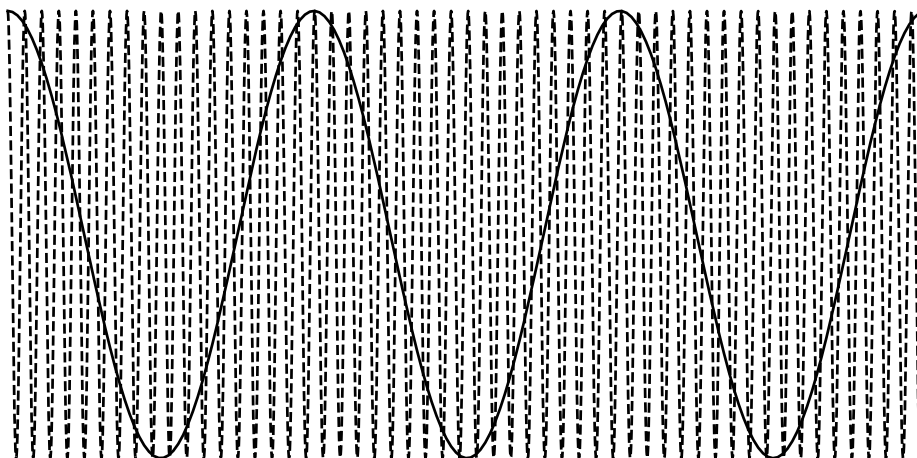
Figure 2.3: A440 v. The Highest Note on a Piano

## 2.3   The Quality of Sound - Superposition

Now while pitch can certainly explain how we can tell the bass and the vocalist apart, how about sounds that are in the same range of pitch? For example a piano and our vocalist can both end up in in the same range of pitches and yet you'll still be able to tell them apart. What's going on here? Well, so far we've been oversimplifying things a lot. Remember how I said our picture of wave was the simplest picture possible? Well one of the ways

in which it's super simple is that it's only got a single wavelength in it. In other words it has a single tone. Most sounds are not like this and are instead the composition of many wavelengths. This layering of wavelengths is called *superposition* and it's best illustrated by an example.

All sounds "begin" with a fundamental frequency - the lowest pitch in the sound. So let's begin with our simplest wave as the fundamental frequency (Fig 4.).
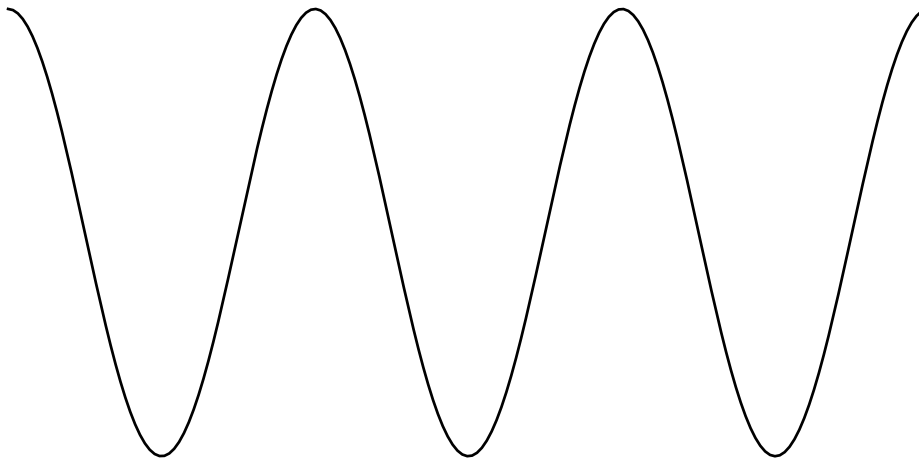


Figure 2.4: Fundamental Frequency

With our fundamental frequency in place, let's add in our first higher frequency. Note that when superimposing waves, not only can their frequencies be different, but their amplitudes can be different as well (and almost always are). So to illustrate this we'll superimpose a wave with half the wavelength and half the amplitude on top of our fundamental frequency. Figure 5 shows both the two base waves (dashed) and the resulting superimposed wave (solid).

Pretty wild right? (I find these kinds of waves starting looking really interesting and honestly kind of beautiful)

Alright, let's add one more frequency in, this time with one quarter the wavelength and equal amplitude in comparison to our fundamental frequency (Fig. 6).

Alright, while this is cool and nerdy and all, how does this relate back to distinguishing a piano from a vocalist? Well, when you play a note on any
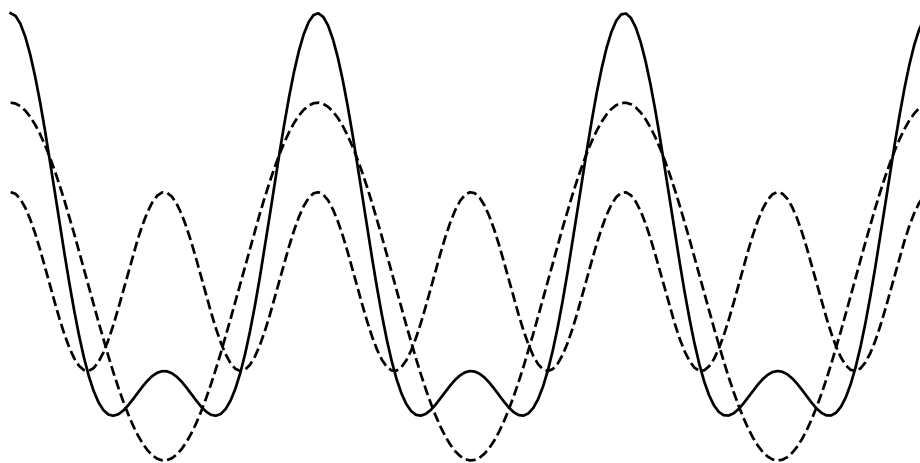
Figure 2.5: Super Imposing Two Waves



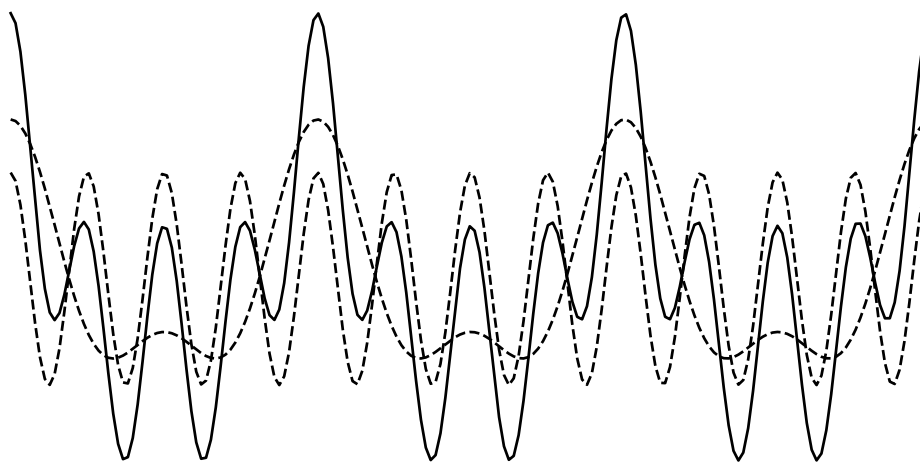Figure 2.6: Three Frequencies

instrument the resulting sound is the superposition of whole load of different frequencies grounded on top of that fundamental frequency. What makes each instrument sound different, even when they're playing similar notes, is the fact that their particular mix of frequencies and amplitudes are each different. The combination particular to one instrument is what gives it its

particular sound. In fact, for a trained ear, these differences can even allow you to identify one instrument from another. For example a piano with higher amplitudes at higher frequencies literally sounds brighter than one with muted amplitudes at higher frequencies. Superposition then, and the particular mix of frequencies and amplitudes that make up a specific sound, is yet another tool in our tool box of ways to distinguish one sound source from another.

## 2.4   Vanishing Act - Phase

So we've now got amplitude, wavelength, and superposition which brings us to the final component we can use in describing our waves. This one is probably the most abstract and weird of the bunch, but it's one we're going to take advantage of a *lot* and its name is **phase**.

Thus far we've been looking at all of these waves as being in one spot in our graph. But there's no reason why we can't start shifting them from left to right (Fig 7.). This shifting is phase.
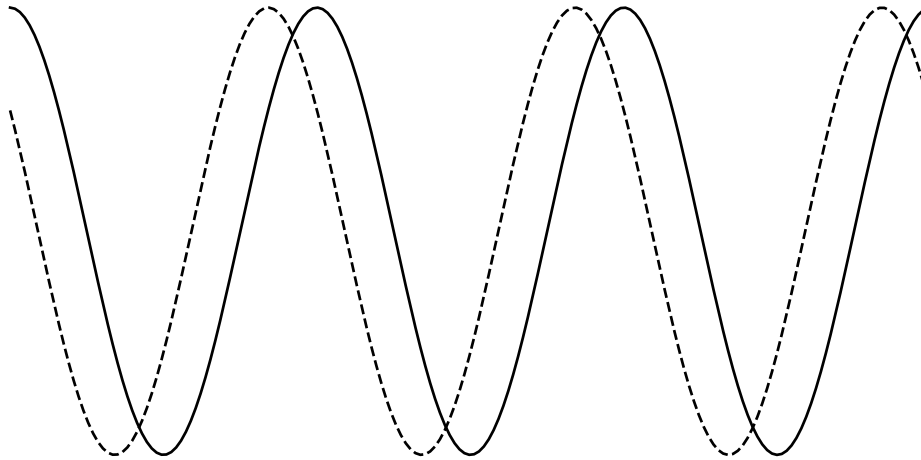


Figure 2.7: Two Waves with Different Phases

What gets really weird (and where we start to see the power of phase) is when we superimpose two waves with the same wavelength but different phases.

Figure 8 shows the super position of the two waves from Figure 7. Note how because the waves are nearly in sync (peaks match with peaks and troughs match with troughs) they add together to create a wave with much higher amplitude. This is known as **constructive interference**.
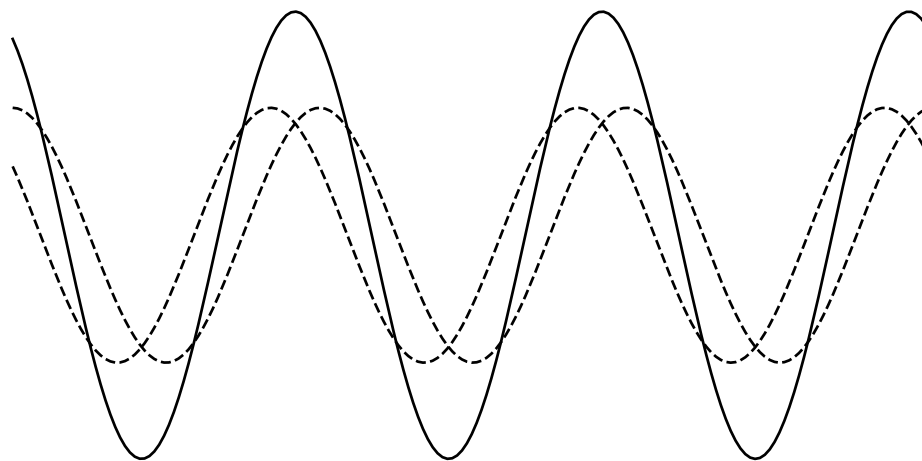


Figure 2.8: Constructive Interference

On the other hand, Figure 9 shows the superposition of two waves that are nearly out of sync (peaks at troughs and troughs at peaks). Note how in this case the resulting wave is much smaller than either of the constituents - this is **destructive interference**.

What's really weird is that if you get the two waves to be exactly out of sync, the resulting superposition *vanishes*! What does this mean physically? It means that the sound itself disappears! Yep, that's right, if two sound sources have just the right phase difference you'll suddenly no longer hear them even though the underlying sounds are there. Absolutely wild right? Well it turns out that this odd mathematical feature helps your brain figure out where sound is coming from.

Suppose a sound is coming directly from your right. That sound will obviously hit your right ear first. A very very short time later the same wave will hit your left ear, but at that point the undulation at your right ear will have changed (because the wave is constantly, well, waving). This difference in what your right and left ear are receiving is equivalent to the phase shift we were just talking about. Now suppose the sound is coming from directly
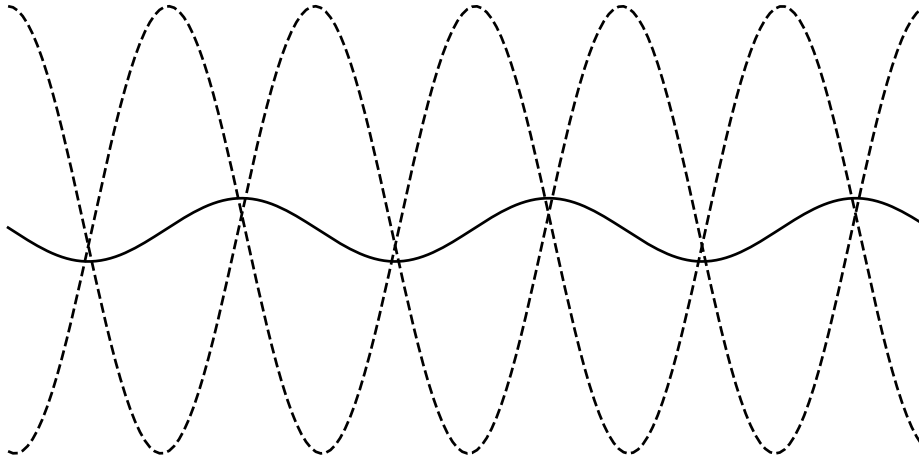
Figure 2.9: Destructive Interference

in front of you. In this case the sound hits both ears at the same time because the distance to each ear is the same. This means there is no phase difference. What these two examples show is that as a sound source moves around your head the phase difference between your two ears changes. And your brain can use these differences to help you pick out where the sound is coming from! [9]. Pretty amazing, right? This happens to be the simplest version of a very cool technology called a **phased array** - something we'll be diving into detail next. But before we get to that we need to pull together all the tools we've gathered up so far and formalize them mathematically because phased arrays get pretty technical. So let's step back and do just that.

## 2.5 Sums and Summaries - Mathematical Formalization

Alright, so we've got all our pieces:

- **Amplitude:** The height of a wave, representing volume.

- **Wavelength (or Frequency):** The width of a wave, representing pitch.

- **Superposition:** The particular mix of amplitudes and frequencies that make up a sound.

- **Phase:** An abstract sense of the "position" of a wave that leads to destructive or constructive interference.

How do we tie them all together mathematically? Well our simplest of waves is described by the following formula:

$$y = ae^{i\psi}e^{ikx} \tag{2.2}$$

$e = 2.71828$ is a mathematical constant known as Euler's number. $i = \sqrt{-1}$ is the imaginary number. $a$ is our amplitude. $k = 2\pi/w$ where $w$ is our wavelength. Last but certainly not least, $\psi$ (pronounced like *sigh*) is our phase.

That then gets us three out of our four. So what about superposition? Well remember superposition is just adding many simple waves together, so we can do just that:

$$y = a_1 e^{i\psi_1} e^{ik_1 x} + a_2 e^{i\psi_2} e^{ik_2 x} + ... + a_N e^{i\psi_N} e^{ik_N x} \tag{2.3}$$

Now writing out all these terms all of the time is going to get really burdensome, so we're going to take advantage of a little bit of mathematical notation that you may or may not be familiar with - the sum $\Sigma$.

If you're not familiar with this notation, let's demonstrate with a simpler example. Suppose that you were adding the numbers 1 through 100. Without $\Sigma$ you would write:

$$1 + 2 + 3 + ... + 100 \tag{2.4}$$

With $\Sigma$ this same expression becomes:

$$\sum_{n=1}^{100} n \tag{2.5}$$

which reads as - "add together all the $n$ (the thing to the right of the $\Sigma$) where $n$ starts at 1 (expression below the $\Sigma$) and goes all the way to 100 (number above the $\Sigma$)". I appreciate that this is probably pretty abstract and a little mind bending if this is your first time seeing it, but as we dive further into our little adventure you'll see just how useful this one bit of notation is. Alright, back to our waves.

Our superposition of waves goes from looking like this

$$y = a_1 e^{i\psi_1} e^{ik_1 x} + a_2 e^{i\psi_2} e^{ik_2 x} + ... + a_N e^{i\psi_N} e^{ik_N x} \tag{2.6}$$

to this:

$$y = \sum_{n=1}^{N} a_n e^{i\psi_n} e^{ik_n x} \tag{2.7}$$

which is far neater and, as will become clear later, far easier to work with.

That's it then! We've got our equation for a wave and are clear on the various components that make one sound different from another. With these tools in hand let's go and look at an absolutely incredible (and mathematically beautiful) technology for sound localization based on nothing more than our two ears - the phased array.

# Chapter 3

# To Phase Array

## 3.1   Back at the Pond

With a firmer understanding of the components of sound, let's return to the pond. Frogs in full chorus are a tricky bunch because while different species of frog may sound quite different, the individuals within a species are much more like instruments in the same section - while we find it easy to distinguish the violins from the cellos, telling individual cellos apart is extremely difficult (unless someone is playing terribly off tune). This is because they've all got roughly the same pitch, are playing at nearly the same time, and have similar if not identical volumes. Frogs, surprisingly are quite the same. The song for a specific species is very distinct, in chorus they tend to overlap with one another, and, in the competition to be heard, everyone gets rather loud. To complicate things even further they'll vary the length or repeats in their songs so we can't even try to identify a specific phrase as an individual. So, if frequency and amplitude are so unhelpful to us at this specific stage what are we to do?

What we do know is that each frog is going to be calling from a specific spot - so while they may overlap in terms of a lot of things, position is unlikely to be one of them. This brings us straight back to our two ears example from the last section. Recall how as you vary the position of what you're listening to, the difference in timing between your two ears changes and that these differences in timing correspond to differences in phase. Well this means that if you have multiple listening devices (in this case your two ears), the phase differences between those sources is a function of the position of the

sound source. How does this help us? Well remember that phase differences create destructive and constructive interference; so, if the phase difference is a function of location so too is whether the interference is constructive or destructive.

Let's take an example (and note this is an illustrative example, the extent to which your brain actually does this is up for debate). Suppose that we have two sound sources. One directly in front of you and one directly to the right. For the source directly in front of you, the distance from the source to each of your ears will be exactly the same. Therefore, the phase difference will be zero and as we saw in Section 2.4 this will mean perfect constructive interference - the combined wave will have twice the amplitude. Now consider the sound source at your right. Obviously the sound from that source will hit your right ear first and then your left. Suppose that the wavelength of this wave is such that when the sound hits your left ear it is at a peak but when it hits your right ear it is at a trough - i.e. they are completely out of sync. In this case we'll get perfect destructive interference and the superimposed sound will vanish. Now note this doesn't mean you'll just stop hearing the sound because your brain can do all sorts of funky things with the sounds coming to your ears; but, if we imagined two microphones reading in these sounds and superimposing them it would hear the sound in front as twice as loud and the sound to the right not at all which means we'd have a way of listening to one location while completely ignoring another which is exactly what we want!

Let's take this a step further. Returning to our digital ears (the microphones) we know that we can post-process the data from each microphone however we like. In other words, we can delay the signals coming from either microphone before combining them. So for example, we could delay the signal at the right microphone so that we get the two signals from the right sound source to line up and experience constructive interference. But this will of course cause the sound source at the front to go out of phase and vanish. In other words, using digital post processing we've switched which location we're listening too! And because it's digital, we can do both the delay and no delay at the same time as two separate processes which means we can isolate each sound source and listen to them simultaneously!

Now imagine that we had a setup like this at the pond. In theory it may be possible for us to vary the delays on our digital ears in such a way that we could isolate many different locations on the pond surface simultaneously and listen to them all at once. If we can get the size of those isolated blocks

to be small enough that they cover a single frogs territory then we could listen to all the frogs individually and have broken the frog cipher!

Well turns out this kind of setup is an already well developed technology known as a **phased array**. But phased arrays require a great deal of precision and far more than two digital ears. So in order to figure out how this would work in reality we're going to need to go through the mathematics of phased array antennas - the subject of the rest of this section. Then in the following section we'll be able to return to the question of what it would take to build a phased array antenna that could listen to individual frogs on a pond simultaneously. So let's dive into the math.

## 3.2 On Many Eared Aliens - Planar Phased Arrays

The following derivations and treatment come almost entirely from an absolutely incredible book - *Phased Array Antenna Handbook* [4]. If you're wanting to go into more detail or learn more about phased array antennas I'd encourage you to check it out.

We begin with a large grid sitting, as all things mathematical tend to do, on a plane (a big, perfectly flat surface). This grid is composed of lovely, omni-directional antennas. Specifically they are layered $M$ deep with a spacing of $d_x$ along the $x$-axis and $N$ deep with a spacing of $d_y$ along the $y$-axis which means we have $M \times N$ antennas in total. This setup is known as a planar phased array antenna. We denote the position of our sound source by two angles (illustrated in Figure 1) $\theta$ and $\phi$.

You'll remember from the prior section that our overall wave equation for a single frequency was given by:

$$y = ae^{i\psi}e^{ikx} \tag{3.1}$$

Our independent variable here was $x$ which in that context was the distance from the sound source to the position you were observing the wave at. Given we're using $x$ for something else in this context (one of the dimensions of our planar phased array) we'll instead designate the distance from our sound source to the $j$th element of our array as $R_j$. In other words:
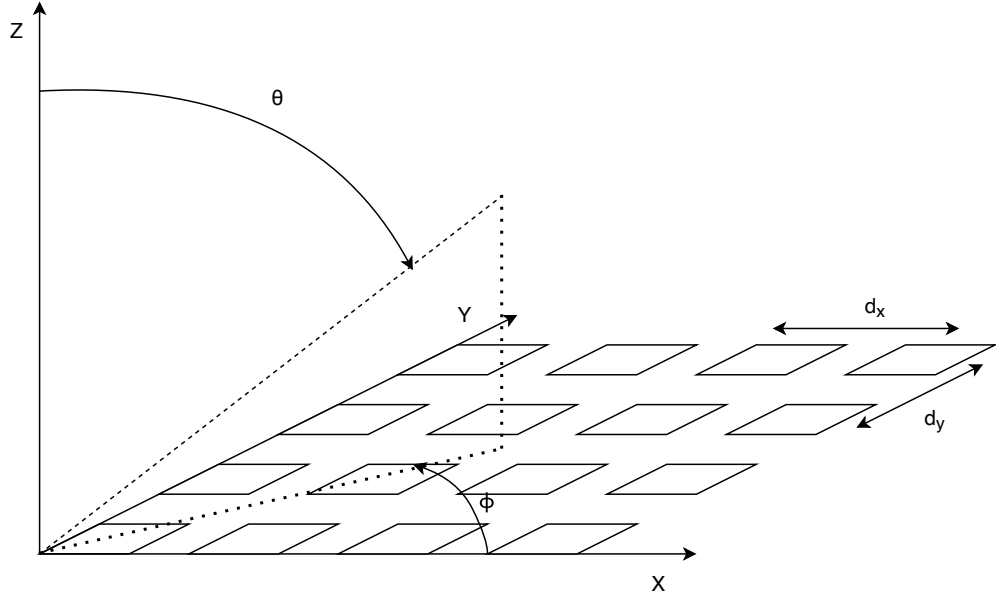
$$y = ae^{ikR_j} \tag{3.2}$$

Figure 3.1: A Planar Phased Array

Note that we're ignoring the phase term here because for a single wave it's just a constant and will therefore have nothing to do with our constructive or destructive interference.

Now it's important to note that we're making a few simplifications here. First, in reality as you get farther from a sound source the amplitude shrinks but we're just assuming we're only interested in the amplitude near the antenna $a$ so we don't need to account for this reduction in sound level. Second we're implicitly making what's called a *far field* assumption here - namely that the source is far enough away that we can use our simple wave equation from the last section (if the sound source is very close the equation gets far more complicated). We'll get into what this means for our antenna design in the next section, but I figured it would be useful to mention now.

Let's next go ahead and designate $R$ to be the distance from the sound source to our our origin $(0,0)$, $\hat{\mathbf{r}}$ as the unit vector from the origin in the direction of our sound source, and $\mathbf{r_j}$ as the vector from the origin to the $j$th element (see Figure 2). In the case where $R_j$ is very large we find:

$$R_j \approx R - \hat{\mathbf{r}} \bullet \mathbf{r_j} \tag{3.3}$$

Why is this useful? Well let's look at what $\hat{\mathbf{r}} \bullet \mathbf{r_j}$ ends up being. Using Figure 1, some trigonometry, and the definition of our planar array we find that:

$$\mathbf{r_j} = \hat{\mathbf{x}}x_j + \hat{\mathbf{y}}y_j + \hat{\mathbf{z}}z_j \tag{3.4}$$

$$\hat{\mathbf{r}} = \hat{\mathbf{x}}\sin\theta\cos\phi + \hat{\mathbf{y}}\sin\theta\sin\phi + \hat{\mathbf{z}}\cos\theta \tag{3.5}$$

where $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, and $\hat{\mathbf{z}}$ are our coordinate vectors and $x_j, y_j, z_j$ are the coordinates of our $j$th antenna. Let's define $u = \sin\theta\cos\phi$ and $v = \sin\theta\sin\phi$ so that we can shorten the latter equation to:

$$\hat{\mathbf{r}} = \hat{\mathbf{x}}u + \hat{\mathbf{y}}v + \hat{\mathbf{z}}\cos\theta \tag{3.6}$$

Next if we define $m$ to be the number of antennas in our $j$th antenna is along the $x$-axis and $n$ the same along the $y$-axis then we have:

$$\hat{\mathbf{x}}x_j + \hat{\mathbf{y}}y_j + \hat{\mathbf{z}}z_j = \hat{\mathbf{x}}md_x + \hat{\mathbf{y}}nd_y \tag{3.7}$$

So now we can go ahead and compute the scalar product:

$$\hat{\mathbf{r}} \bullet \mathbf{r_j} = qd_xu + pd_yv \tag{3.8}$$

Plugging this into our approximation we have:

$$R_j \approx R - qd_xu + pd_yv \tag{3.9}$$

Still unsure what all this juggling has been for? Well let's throw this back into our wave equation and see what this is all about:

$$ae^{ikR_J} \approx ae^{ik(R-qd_xu+pd_yv)} = ae^{ikR}e^{-ik(qd_xu+pd_yv)} \tag{3.10}$$

Note that $e^{ikR}$ is once again just a constant phase term that we don't care about here and that because we're interested in the value of the phase and not its sign we don't care about the negative in front of $ik(qd_xu + pd_yv)$ either. Therefore this all simplifies to:

$$ae^{ik(qd_xu+pd_yv)} \tag{3.11}$$

Which means our whole wave equation is now defined entirely in terms of values we know - pretty slick.
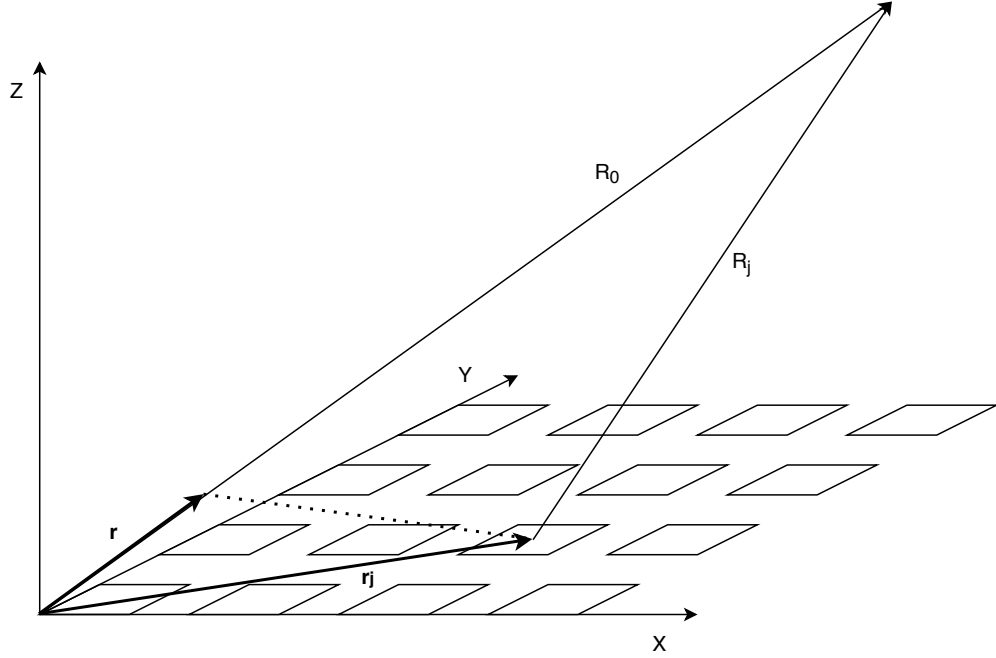
Figure 3.2: Approximation of $R_j$

Alright, thus far we've only been talking about what the microphone sees without modifications. But remember that the whole point here is going to be introducing phase shifts in order to create the constructive or destructive interference that we want. A phase shift is simply represented by adding (or subtracting) a value from our exponent:

$$ae^{ik(md_x u + nd_y v)}e^{i\psi} \tag{3.12}$$

Now if we set $\psi$ so that it cancels the other exponent we'll find that all antennas (regardless of the specific $m$ or $n$) will have the same phase. I.e. we will be creating the maximal constructive interference for the direction defined by $\theta$ and $\phi$. Therefore:

$$\psi = -k(md_x u + nd_y v) \tag{3.13}$$

Let's generalize this a little bit. Let's create two new variables $u_0 = \sin\theta_0 \cos\phi_0$ and $v_0 = \sin\theta_0 \sin\phi_0$ that designate our **steering angle**. Then let's set $\psi = -k(md_x u_0 + nd_y v_0)$ so that our wave becomes

$$ae^{ik(md_x(u-u_0)+nd_y(v-v_0))} \tag{3.14}$$

From here it becomes clear that when the steering angle and the source direction coincide we get the most constructive interference (because all of the antennas experience the same phase regardless of $m$ and $n$).

You may now be wondering what happens when the steering angle and the source direction don't coincide. Well the contributions from all of our $M \times N$ antennas is:

$$I = \sum_{m=1}^{M} \sum_{n=1}^{N} ae^{ikqd_x(u-u_0)} e^{ikpd_y(v-v_0)} \tag{3.15}$$

so let's plot out some examples and see! (Also imagine trying to write out that equation without our $\Sigma$ notation...)

## 3.3 Enough Math, Show Me Something

We've got a mathematical formalization for our phased array so it's time to check out what things actually look like! Let's take a simple example of a single row of antennas (i.e. $N = 1$) and graph things out!

Now for those of you who have been playing close attention you'll have noticed that the $I$ from the preceding subsection is an imaginary number. Obviously we won't be plotting anything imaginary so what are we going to be plotting? We'll be plotting the power as a function of $\theta$ (the source location). Power is defined as:

$$P = I\bar{I} \tag{3.16}$$

or as $I$ multiplied by it's conjugate.

As a final note before we get plotting we're going to be plotting the power in $dB$ relative to the maximum power. That just means we're going to be using a log scale rather than a linear one. Alright let's get to it.

We'll start with the simplest case where $u_0$ (indicative of the **steering angle**) is steered to $\theta = 0$ (Figure 3). We'll include 10 elements in our array and space them half a wavelength apart (more on this in the next section).

Right away there's several things to take away from this graph. First while we have steered toward $\theta = 0$ and are getting our most constructive interference at that angle, it's not as if the constructive interference just falls
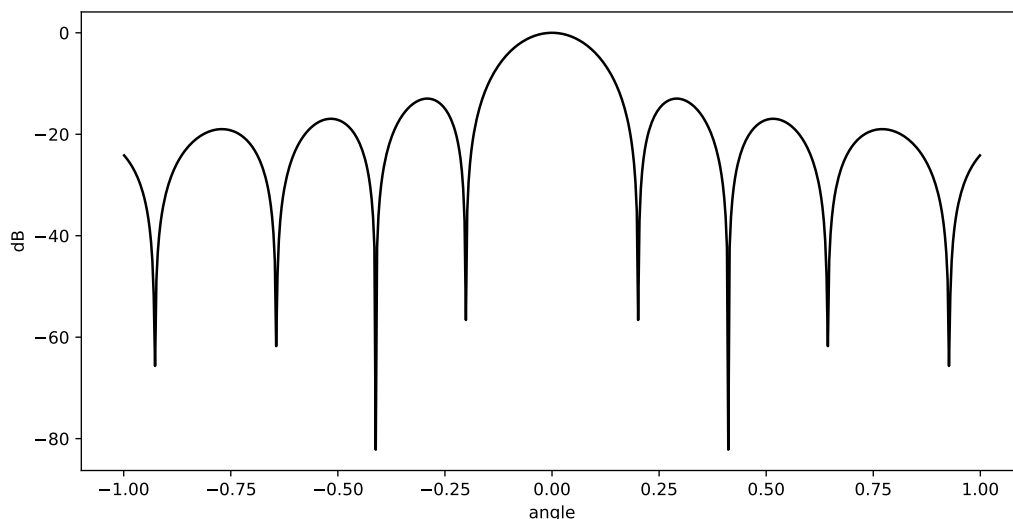
Figure 3.3: Steering Angle $\theta = 0$ with 10 Element Array

off immediately as we move away from $\theta = 0$. Instead we see that we have this whole area around $\theta = 0$ where we still have considerable power before it drops off sharply. That area is known as the **beam** and its width (measured in various ways) is the **beam width**. Put another way, we won't just be hearing things at $\theta = 0$ but also things throughout the beam. How narrow our beam is (and thus how small the beam width) determines how localized our listening will be.

Second, in the graph we see these steep drop offs followed by subsequent peaks. These other peaks are known as **sidelobes** and are other areas in which we'll get non-negligible signal from.

Let's now look at a couple other examples. First let's see the effect of changing the steering angle by modifying our $u_0$ (Figure 4).

This is exactly as we'd expect, as the steering angle changes the position of our beam should change as well.

Finally let's look at what happens when we increase the number of elements to 20 (Figure 5).

Notice how in comparison to our 10 element array our beam width has shrunk dramatically. In general, as we add elements our localization improves and where we receive signal from shrinks. This is our first aspect of antenna design - a subject we'll now turn to in more detail in order to understand
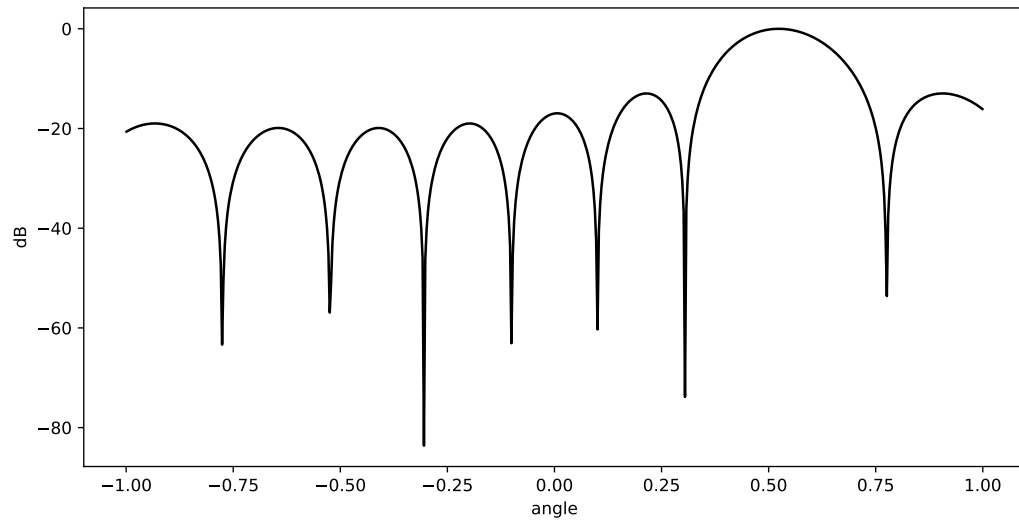
Figure 3.4: Steering Angle $\theta = \pi/6$ with 10 Element Array
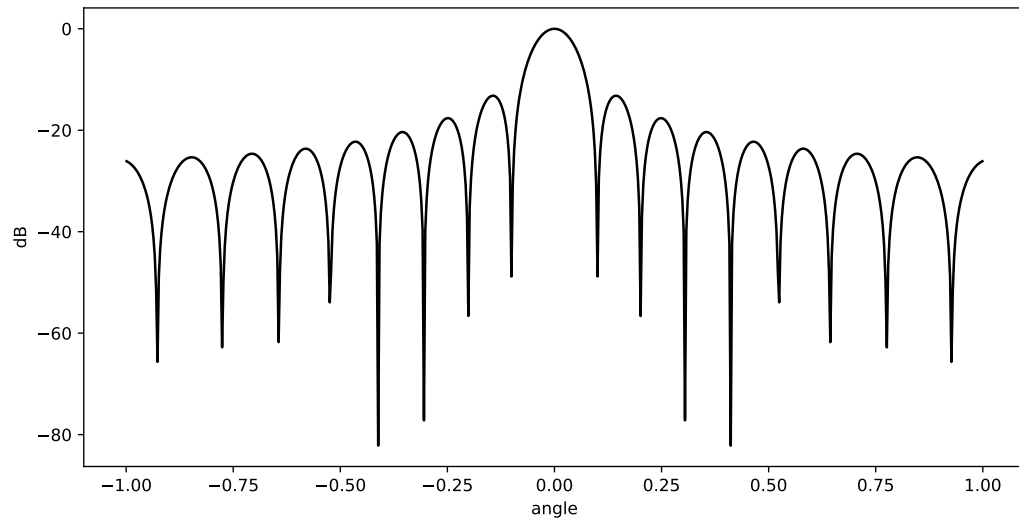


Figure 3.5: Steering Angle $\theta = 0$ with 20 Element Array

what it would take to localize all the frogs on the surface of a pond.

# Chapter 4

# How Tall is A-flat?

We have formalized our phased array with the following equation:

$$I = \sum_{m=1}^{M} \sum_{n=1}^{N} a e^{ikqd_x(u-u_0)} e^{ikpd_y(v-v_0)} \tag{4.1}$$

Which shows us pretty summarily that the two things we have control over are (1) the number of elements ($M$ and $N$) and (2) the spacing of those elements ($d_x$ and $d_y$). So let's understand how changing each of these changes the properties of our phased array antenna.

## 4.1  Getting Greedy - Number of Elements

The first of our modifiable parameters is the number of elements. We already saw the main gist of what happens here when we went from 10 to 20 elements in the last section and saw the beam width shrink. But, to firm up our intuition and understanding, let's go through a few more examples.

We'll iterate from the simplest array possible at 2 elements through 4, 16, and then 256 elements (Figures 1 - 4).

The first take away from this sequence is both how wide the beam is in the case of two elements and how we can make it more or less indefinitely small by adding more and more elements (look at how small the beam width is in the 256 element case!). The second point of note, though, is that as the beam width shrinks the number of distinct side lobes increases. While an interesting observation this is not going to be of huge concern to us. Main
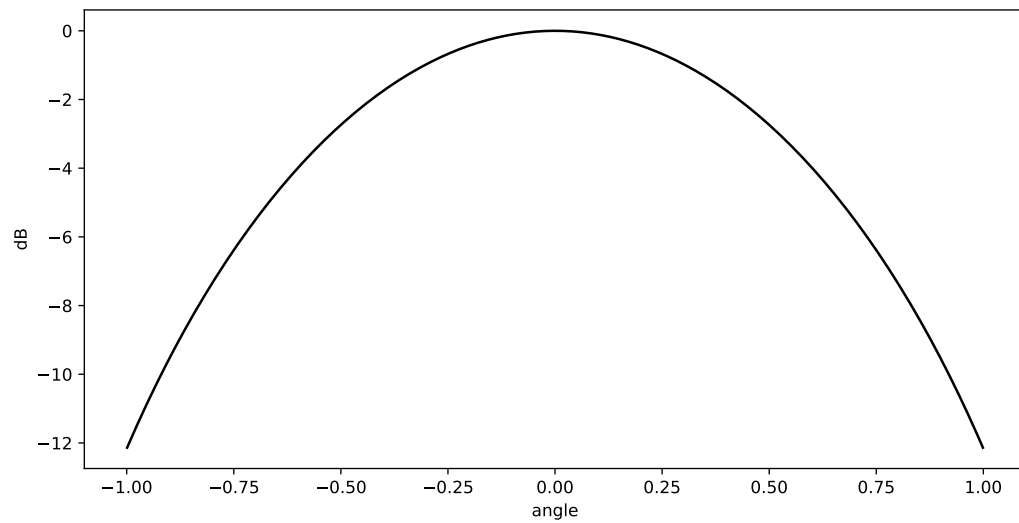
Figure 4.1: Steering Angle $\theta = 0$ with 2 Element Array
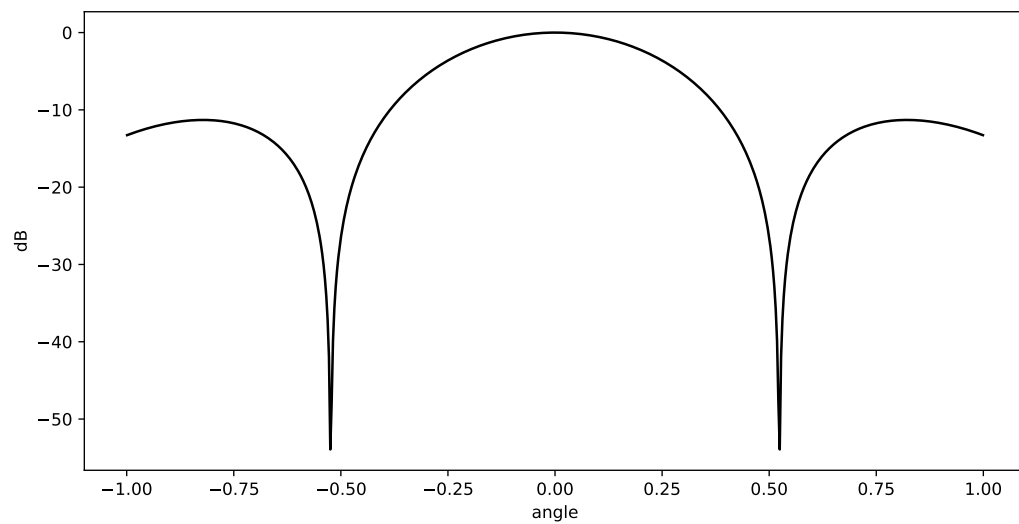
Figure 4.2: Steering Angle $\theta = 0$ with 4 Element Array

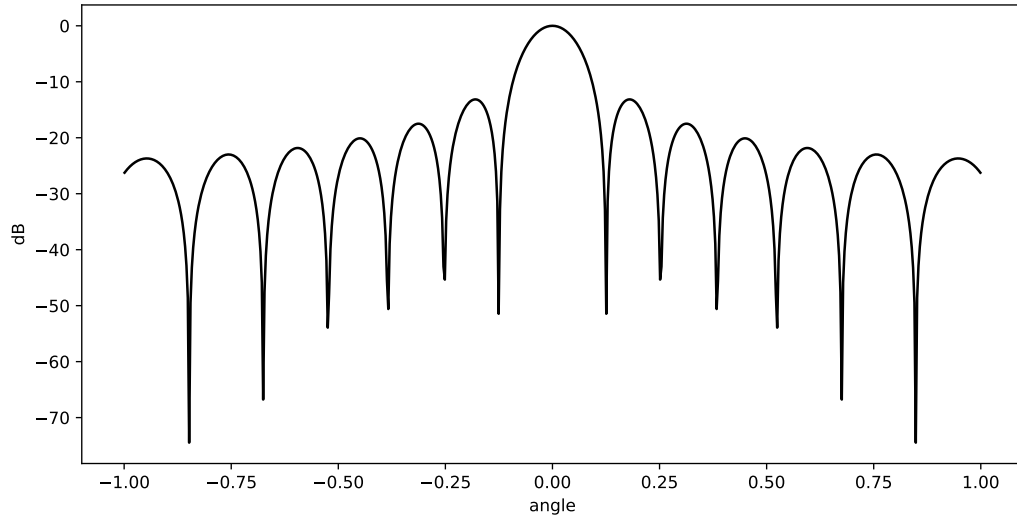takeaway should be that if you want to shrink your beam width - add more elements.

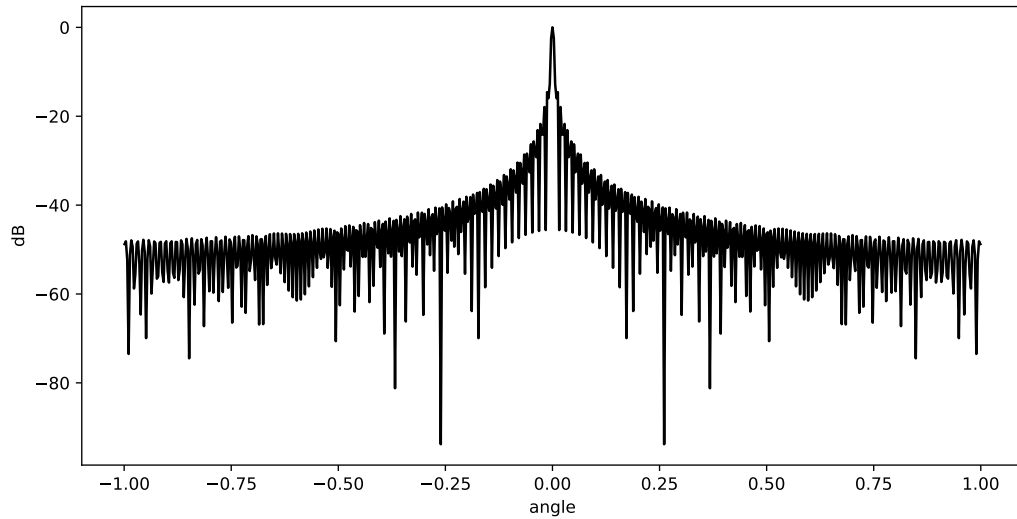Figure 4.3: Steering Angle $\theta = 0$ with 16 Element Array



Figure 4.4: Steering Angle $\theta = 0$ with 256 Element Array

## 4.2 Being Skinny Isn't Always Great - Element Spacing

Next we have our good ole spacing parameters $d_x$ and $d_y$ (or just $d_x$ as we're considering a one dimensional phased array at the moment). Let's return to

our 10 element phased array with what we called half wavelength spacing
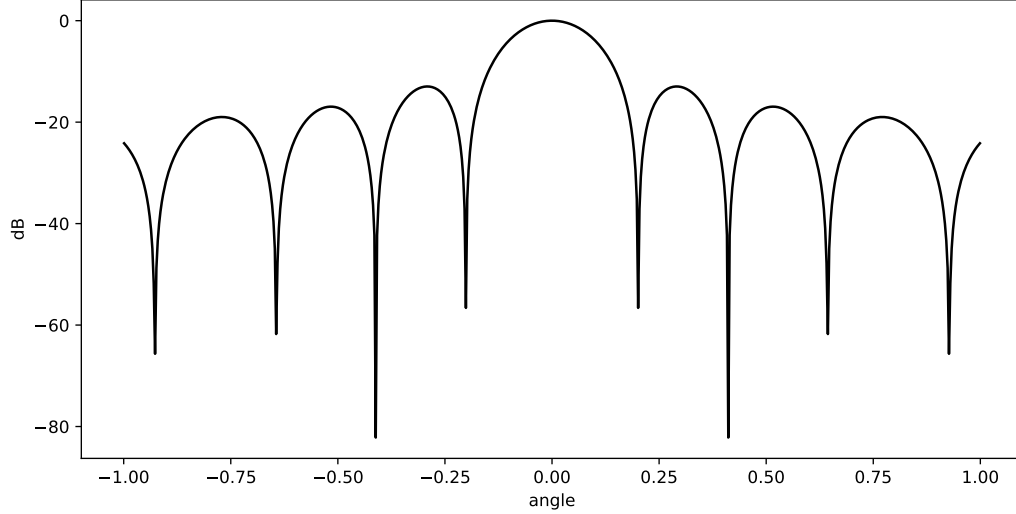(Figure 5):



Figure 4.5: Half Wavelength Spacing

First let's drop the spacing to, say, a quarter wavelength (Figure 6). Note
our beam width just increased! This makes sense because as our elements
get closer and closer they become more and more like one single antenna
rather than a phased array. Does that then mean that the larger our spacing
the better? Not quite. To illustrate why consider the case of two wavelength
spacing (Figure 7).

What the on earth is going on here?! We now see multiple beams! This
comes down to a degeneracy in any kind of cyclic function. Remember how
we were getting constructive interference because the phases were all the
same across all the antenna elements? Well this is not the only way to get
perfect constructive interference. Phases that are different by exact multiples
of $2\pi$ will also create perfect constructive interference. Looking back at our
equation for our phased array

$$I = \sum_{m=1}^{M} \sum_{n=1}^{N} a e^{ikqd_x(u-u_0)} e^{ikpd_y(v-v_0)} \tag{4.2}$$

we can see that as $d_x$ or $d_y$ increase, changes in the $u - u_0$ or $v - v_0$
have a larger overall impact on the phase of the exponential. Therefore as
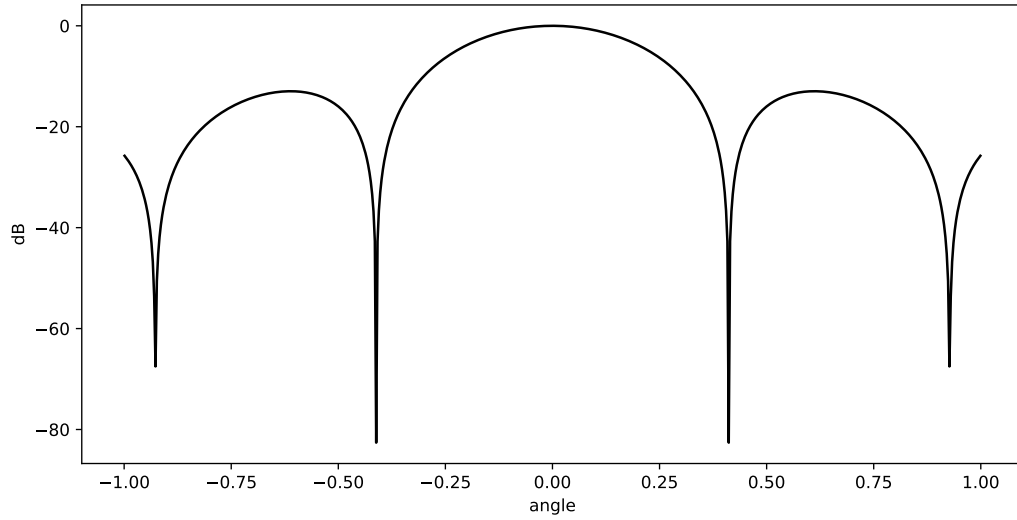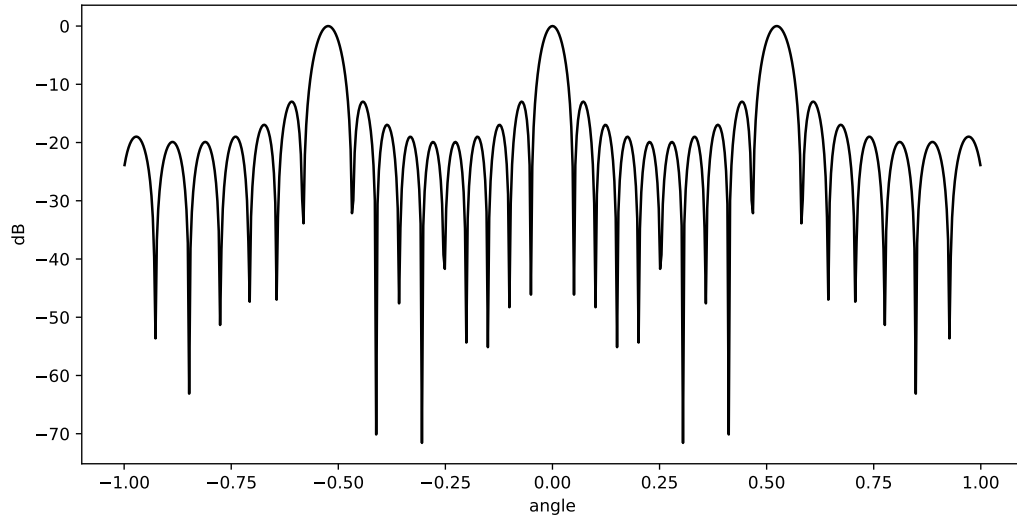
Figure 4.6: Quarter Wavelength Spacing



Figure 4.7: Two Wavelength Spacing

the spacing increases it becomes possible to hit multiples of $2\pi$ and thereby create other areas of perfect constructive interference (our other beams in the graph). These other beams are known as **grating lobes**. Note too that $k$ plays a similar role to $d_x$ and $d_y$ which is why spacing has to be a function

of the wavelength. In general, the rule of thumb is that larger spacing is always better but to avoid grating lobes you need to keep the spacing $\leq$ to half the wavelength.

## 4.3   Far A Field

Okay, we've talked about the number of elements and the spacing of elements. That's it right? Not exactly. Remember how I mentioned in the last section that we were making a far field assumption? Well it's time to talk about that now and what it means for the design of our antenna.

In almost every case thus far we've been talking about planar waves - that is waves that look like they're coming in a single front. To get a sense of what I mean by this, a planar wave is kind of like what happens if you were to take a sheet and wiggle it. Each of the waves travels straight down the sheet without radiating in some new direction. Compare this on the other hand to what happens when you drop a stone in a pond. Those waves radiate outwards in all directions in rings.

Close to a sound source waves are much more like what you see on the surface of a pond and, unsurprisingly, the math involved is a lot more complicated than what we've been going over here. But if you imagine that circular wave radiating out really really far and looking at only a small part of that circumference you'll note that as the radius of the circle gets larger, the edge gets flatter - in other words it becomes more and more like a planar wave. This is the far field assumption (or perhaps the better word would be simplification) - as you get far from the source of the wave the wave becomes approximately planar.

How far away do you need to be for this to take effect? The answer is known as the **Fraunhofer distance** [7]:

$$\frac{2D^2}{w} \tag{4.3}$$

where $w$ is the wavelength and $D$ is the largest dimension of the antenna. If we assume a half wavelength spacing between elements then for a linear array this becomes:

$$\frac{2(w/2(M-1))^2}{w} = \frac{w}{2}(M-1)^2 \tag{4.4}$$

In other words as either our wavelength or the number of elements we use increases, the far field gets farther and farther away.

Okay so let's take everything we've learned and put it together to see whether simultaneous listening with a phased array antenna is even possible.

## 4.4 Sound is Big - Designing the Antenna

As has probably become clear from the last few subsections, the two most important parameters for us are - what's the wavelength we're going to be listening in at and how tight does our beam need to be?

Let's start with a rather well known critter - the Coqui (for which I've read several studies on automated call recognition). A reasonable middle for their frequency range is 2000 Hz [5]. Given the speed of sound is roughly 343 m/s this corresponds to a wavelength of $343/2000 \approx 0.17$m (about 6.7 inches). Right off the bat then we know that we're spacing our antennas 3.3inches apart.

Next, what's the beam width we want? Let's suppose the frogs are, at a maximum, 10 meters away. Using some trigonometry we can derive that a beam width of 5 degrees ($\approx 0.09$ radians) would therefore correspond to $2 \times 10 \tan(0.09/2) \approx 0.9$m. This is probably too large so let's try a 2.5 degree beam which will give us about half a meter width at 10 meters distance.

How many elements are required to get a 2.5 degree width? Well looking 3dB down from peak (half power down) and varying the number of elements in our linear array the answer is $\approx 45$. 45 elements to a side means $45 \times 45 = 2025$ distinct antennas! It also means an antenna that's $(45-1)*0.17 \approx 7.5$m wide! Not only is this just far too wide to handle it also means our far field would only begin far past our 10m limit that we were talking about earlier.

So what's going on here? Why have we come to the conclusion that we're going to need a house sized antenna? The answer is unfortunately quite simple. Sound waves are actually quite large and therefore don't allow us to provide very high resolution. It's like the difference between a light microscope and an electron microscope. An electron microscope can allow you to see individual atoms because the wavelengths used by the microscope are so much smaller (and therefore much higher resolution) than the ones used in a light microscope. Meanwhile we're over here trying to use wavelengths that are the size of a hand and larger! Clunky resolution indeed.

So is that it? Game over? Not at all! All we know right now is we can't

*just* use the phased array by itself and call it good. We're going to need some additional machinery to help unpack all the useful information the phased array is giving us. Doing so is going to require some careful thinking and a lot of really cool tools - so let's get to it!

# Chapter 5

# Playing Clue

## 5.1 The Basic Idea

We now know a phased array on its own isn't going to get us anywhere. So, what to do? Well let's go back to our basics - amplitude, frequency, superposition, and phase - and focus on the two in the middle.

There is one very big difference between a pond in chorus and a choir. It is the reason why a choir sounds ordered and a chorus sounds like chaos - no one (at least as far as I know) is trying to sing to the same score. Whereas in a choir your basses, tenors, altos, and sopranos are each trying to blend into their group, frogs try to distinguish themselves by singing at different times than their neighbors. And this should lead (in theory) to something quite useful for us - not everyone is singing the same frequency at the same time.

Let's consider the luckiest of cases where at any particular time, in the pond, no one is singing quite the same pitch. Then if we broke up the sound we were reading into its constituent frequencies and then applied our lovely phase delays (to localize our recording), then as we sweep through the angles we'd find a very specific direction get the highest amplitude (the angle corresponding to one our fellow singing at that pitch at that time). Then, at another moment we'd find the maximum power direction as our first frog moves onto another pitch and a second one moves in.

In a somewhat less lucky case let's imagine that several (but not loads) of frogs are singing at the same frequency at the same time. Then we'd see a series of peaks as we swept our array. But they would still look distinct

which suggests that we may be able to use all of the math we've so far derived to deduce where our various frogs are and what amplitude they are singing away at (with regards to that particular frequency of course).

If we did this over all the various frequencies and throughout the listening period then we'd end up with loads of amplitude, frequency, and position tuples which we could then group back together to reconstitute the original, distinct songs!

This then is how we're going to attempt to do simultaneous listening with a "too small" phased array antenna. (1) Capture all the sound with a phased array antenna. (2) Break it up into small windows of time and small bands of frequency. (3) Sweep through all the steering angles to come up with a "fingerprint" for the frequency/window in question. (4) Deduce what the sources would need to be to generate that sweep "fingerprint". (5) Reconstitute the original songs by grouping over positions. The key then is understanding how we're going to deduce our sources from our fingerprints.

## 5.2   Looking at Prints

Let's begin by first looking at some fingerprints. Before we start doing this we're going to switch things up a little. Rather than looking at power in relative dB as we've been doing this whole time, we're just going to look at straight power because I find it far more illustrative when it comes to this fingerprinting business. The other thing worth noting is that all of these figures are going to be based on a 5 element array (so we can show how this all still works even with a really low resolution phased array).

Alright let's begin with a simple example (Figure 1) - two sources at the same amplitude, one at 0 radians and the other at $\pi/4$ radians (45 degrees).

This is exactly as we'd expect. Two large peaks in power at roughly the locations of our sources (although take close note that the positions of the maximums are not exactly those of our sources).

Next let's look at what happens when we bring two sources really close together (Figure 2).

Where are the two peaks? Well thanks to how low the resolution is on our phased array they've merged. But don't panic - note how the width of the hump is so much larger than the width of our source centered at zero in Fig. 1! That means there's still something hinting to us that another source is there and where it might be. The only real difference in this case is that
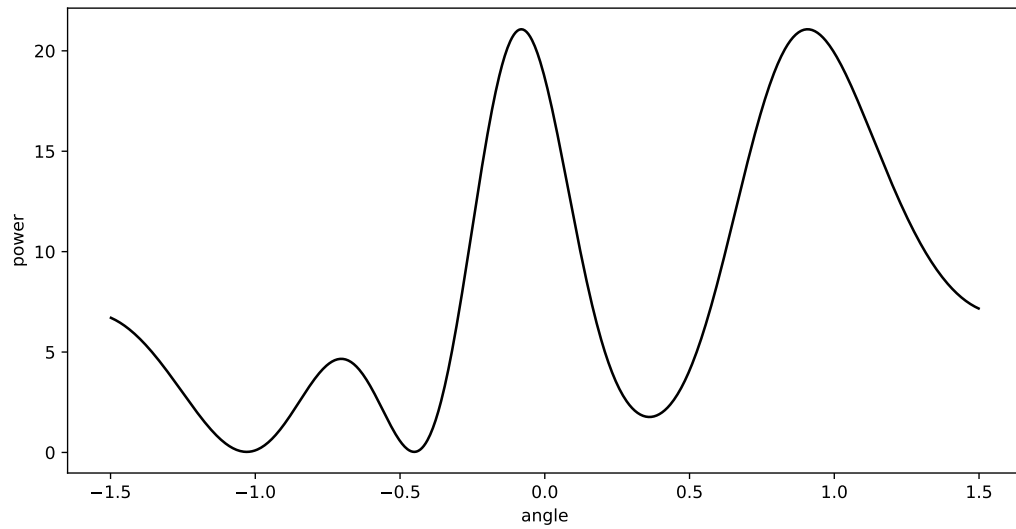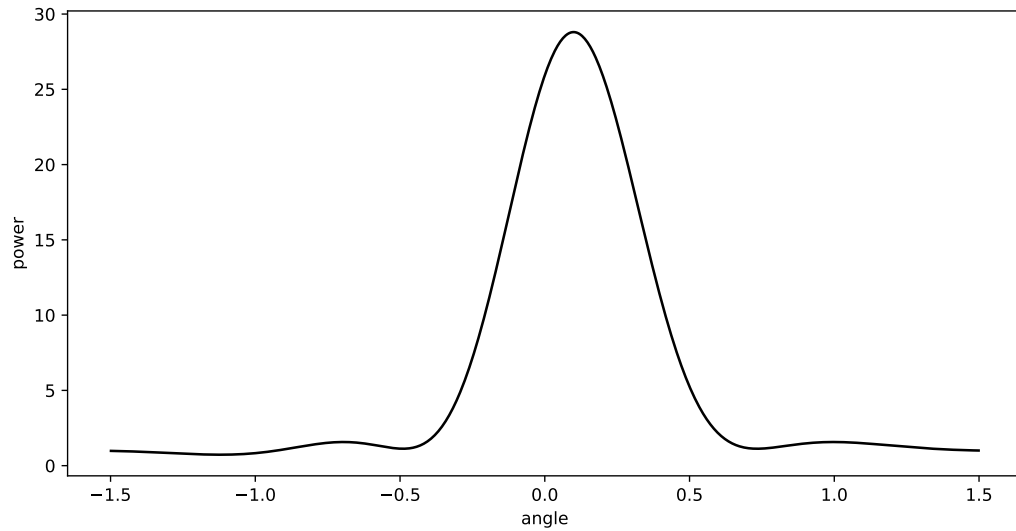
Figure 5.1: 0 and $\pi/4$ radians



Figure 5.2: 0 and 0.2 radians

we'll need more math to figure out where (more on that in a moment).

Now let's look at a case with several sources (once again all at equal amplitude - I hope it's clear that changing the amplitude largely just changes

the heights of these peaks so it's not particularly informative for me to show that).
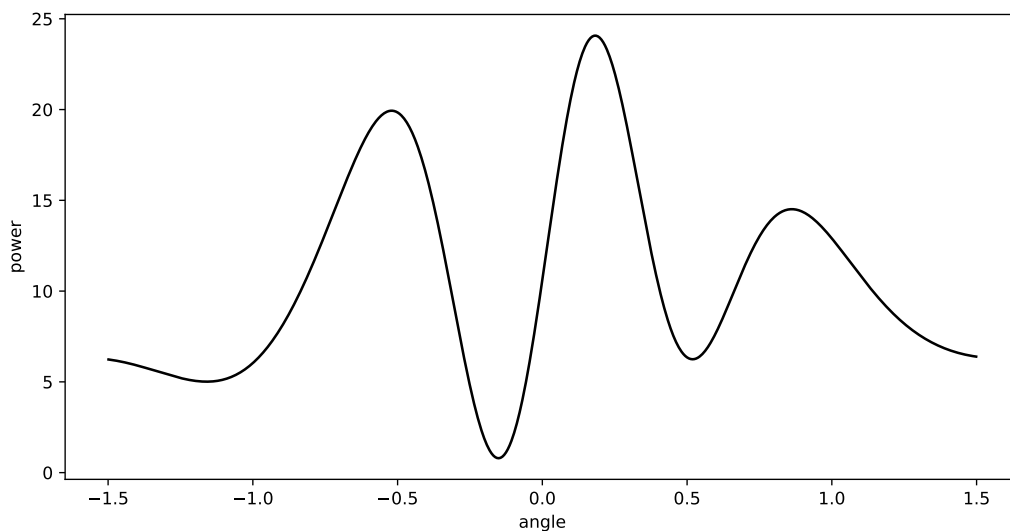


Figure 5.3: -1, -0.3, 0, 0.2, and 1.2 radians

Okay, so this is just to point out that as more and more sources get added to the mix things become harder and harder to sort out (at least for us humans). We can sort of guess that there are two sources to the left of 0 because of the width of that one hump, that there are maybe one or two sources near zero, and then another one out to the right, but the precise positions and amplitudes are becoming much less clear.

So how are we going to do this? How are we going to deduce from these fingerprints who they belong to? The answer to that is known as gradient descent.

## 5.3    The Descent of Math

Suppose that we took our complicated five source case and made some guesses about where the sources were and what their amplitudes are. Well thanks to our handy dandy equation:

$$I = \sum_{m=1}^{M} \sum_{n=1}^{N} a e^{ikqd_x(u-u_0)} e^{ikpd_y(v-v_0)} \tag{5.1}$$

we could compute what the finger print would look like for those sources. For example we might get something like the Figure 4:
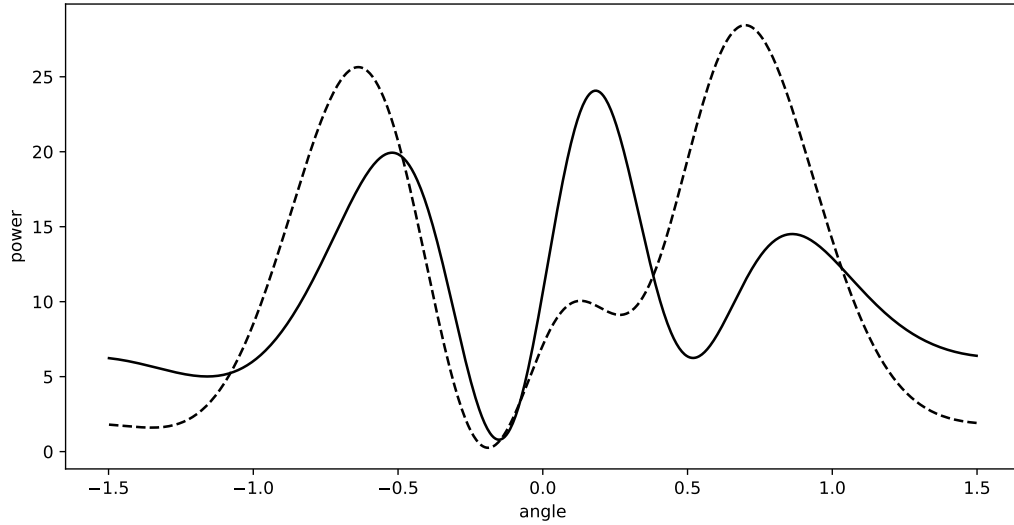


Figure 5.4: A First Guess (dashed line is the guess)

The difference between our dashed curve (our guess) and the actual finger print can be computed and would be the **error** in our guess. Turns out, thanks to derivatives, we could then compute how quickly the error would change and in what direction if we changed our guess just slightly. Given this information we could then "step" our guesses in a direction that should lead to less error and get something like Figure 5:

which is obviously a better fit (although still quite off). With this new guess we could once again compute the error, calculate the direction of best change, and make a new guess which would hopefully once again be better. By repeating this process over and over again we'd eventually converge on the actual set of sources (or something very close). This whole process is called **gradient descent**. Why the weird name? Because computing the direction of greatest change in terms of your independent variables is the same thing as computing the mathematical gradient and we're descending to the lowest error we can manage. So all in all we're doing quite literal gradient descent.
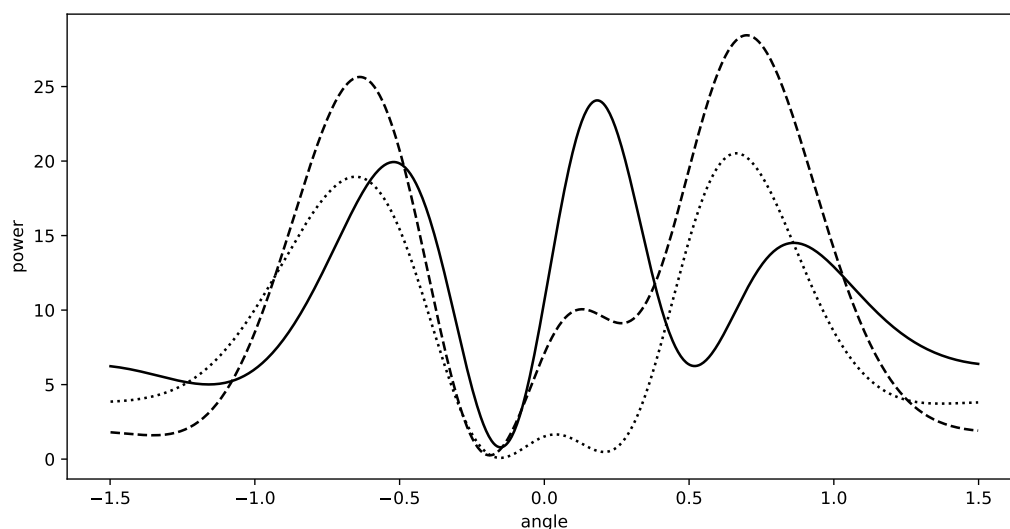
Figure 5.5: A Second *Educated* Guess (dotted line is the new guess)

Gradient descent, by the way, is behind much of machine learning and is a common algorithm used to tune the models that allow computers to see, talk, classify, and much more. So if you're interested in the details definitely go check it out!

For us though it comes down to nothing more than this. We're going to use gradient descent to: make a guess, find the direction that would reduce our error, and re-guess based on that until we converge (stop improving). But to do that we're going to need to calculate that gradient - the subject of the next section.

## 5.4   They Say Summaries are Good

Alright so let's summarize our proposed process:

1. We record the sound at a pond using a pretty small phased array antenna

2. We break up the data into small windows of time and small frequency bands in the hopes that only a few frogs are singing at a particular frequency at a particular time

3. We digitally sweep our phased array for each of these window/bands to create a finger print

4. We use gradient descent to find the best $n$ sources that fit the particular fingerprint. We sweep from small to large $n$ until adding sources doesn't help us to determine the correct $n$

5. From our source finder we collect frequency,time,position, and amplitude tuples

6. We group by position and reconstitute the individual songs

7. We celebrate a new day in frog surveillance

Let's get onto computing that gradient.

# Chapter 6

# A Very Derivative Section

## 6.1 The Model

Before we can consider taking derivatives we must first clarify the model that is going to allow us to calculate our error term. Taken straight from what we derived earlier we know that the power received from a specified direction $\theta, \phi$ is given by:

$$P(\theta, \phi) = |I(\theta, \phi)|^2 \tag{6.1}$$

$$I(\theta, \phi) = \sum_{m,n} a_{m,n} \exp\{ik[md_x(u - u_0) + nd_y(v - v_0)]\} \tag{6.2}$$

$$u = \sin\theta \cos\phi \tag{6.3}$$

$$v = \sin\theta \sin\phi \tag{6.4}$$

where $\theta_0, \phi_0$ would be the current scan direction.

In our case we will be varying the scan direction while the sources stay fixed (and are potentially numerous). Specifically for each point source $p$ we have it's contribution:

$$I_p(\theta, \phi) = \sum_{m,n} a_p \exp\{ik[md_x(u_p - u) + nd_y(v_p - v)]\}exp\{i\psi_p\} \tag{6.5}$$

where we are now using $u, v$ to designate the scan direction. $u_p, v_p$ are now fixed so it makes more sense to allow $I$ to be a function of the scan direction.

Now in general for $P$ different sources we have:

$$I(\theta, \phi) = \sum_{p=1}^{P} I_p(\theta, \phi) = \sum_{p,m,n} a_p \exp\{ik[md_x(u_p - u) + nd_y(v_p - v)]\}exp\{i\psi_p\}$$

(6.6)

where the ultimate prediction we are making is that of (29).

As a final simplification let's represent $I$ as a function of $u, v$ rather than $\theta, \phi$:

$$I(u, v) = \sum_{p,m,n} a_p \exp\{ik[md_x(u_p - u) + nd_y(v_p - v)]\}exp\{i\psi_p\} \qquad (6.7)$$

This then is the mathematical representation of our model which has parameters $u_p, v_p, a_p, \psi_p$.

## 6.2   The Gradient

Let's begin by considering an arbitrary parameter $w_p$. The first question we must ask ourselves is what are we taking the derivative of? In general with any kind of modeling we are interested in reducing the overall error. So our first requirement is a representation of the error itself.

An easy choice thanks to the presence of its derivatives as well as its ubiquitous use is the mean squared error (specifically in the original units of the prediction). Given we are interested in the mean squared error over all of the measured scan angles we have:

$$E = \sqrt{\frac{\sum_{u,v}(P(u, v) - O(u, v))^2}{|O|}} \qquad (6.8)$$

where $P$ are the predictions and $O$ are the observations at each scan direction.

Immediately we see that:

$$\partial_{w_p}E = \frac{1}{2E|O|} \sum_{u,v} \partial_{w_p}[(P(u, v) - O(u, v))^2] \qquad (6.9)$$

$$\partial_{w_p}E = \frac{1}{E|O|} \sum_{u,v}(P(u, v) - O(u, v))\partial_{w_p}P(u, v) \qquad (6.10)$$

Now we know that

$$P(u, v) = I(u, v)\overline{I(u, v)} \tag{6.11}$$

and therefore

$$\partial_{w_p} P(u, v) = \partial_{w_p} I(u, v)\overline{I(u, v)} + I(u, v)\overline{\partial_{w_p} I(u, v)} \tag{6.12}$$

So it follows that what we're really interested in computing is $\partial_{w_p} I(u, v)$ as once we know that we can simply plug it into the above equations to get the derivatives of $E$.

## 6.2.1 Position

Given the symmetry of our problem we can consider $u_p$ alone and get the derivatives with respect to the $v_p$ at the same time. So let us consider:

$$\partial_{u_p} I(u, v) = \partial_{u_p} \sum_{q,m,n} a_q \exp\{ik[md_x(u_q - u) + nd_y(v_q - v)]\}exp\{i\psi_q\} \tag{6.13}$$

Right away we know all the terms where $p \neq q$ will drop away under differentiation. So this simplifies to:

$$\partial_{u_p} I(u, v) = a_p exp\{i\psi_p\}\partial_{u_p} \sum_{m,n} \exp\{ik[md_x(u_p - u) + nd_y(v_p - v)]\} \tag{6.14}$$

$$\partial_{u_p} I(u, v) = a_p exp\{i\psi_p\} \sum_{m,n} \exp\{iknd_y(v_p - v)\}\partial_{u_p} \exp\{ikmd_x(u_p - u)\}$$

$$\tag{6.15}$$

$$\partial_{u_p} I(u, v) = V_p \sum_m \partial_{u_p} \exp\{ikmd_x(u_p - u)\} \tag{6.16}$$

where

$$V_p = a_p exp\{i\psi_p\} \sum_n \exp\{iknd_y(v_p - v)\} \tag{6.17}$$

Now that we've gotten all of the constant terms out of the way let's actually differentiate.

$$\partial_{u_p} I(u, v) = V_p \sum_m \exp\{ikmd_x(u_p - u)\}\partial_{u_p}\{ikmd_x(u_p - u)\} \tag{6.18}$$

$$\partial_{u_p} I(u, v) = V_p \sum_m ikmd_x \exp\{ikmd_x(u_p - u)\} \tag{6.19}$$

By symmetry we then know that:

$$\partial_{v_p} I(u, v) = U_p \sum_n ikn d_y \exp\{ikn d_y(v_p - v)\} \tag{6.20}$$

$$U_p = a_p exp\{i\psi_p\} \sum_m \exp\{ikm d_x(u_p - u)\} \tag{6.21}$$

**Amplitude**

Given the amplitude is just a coefficient, this one is easy:

$$\partial_{a_p} I(u, v) = exp\{i\psi_p\} \sum_{m,n} \exp\{ik[m d_x(u_p - u) + n d_y(v_p - v)]\} \tag{6.22}$$

## 6.2.2   Phase

This too is so straightforward we can just state it here:

$$\partial_{\psi_p} I(u, v) = a_p i exp\{i\psi_p\} \sum_{m,n} \exp\{ik[m d_x(u_p - u) + n d_y(v_p - v)]\} \tag{6.23}$$

## 6.2.3   Returning to the Angles

Getting the gradients in terms of angles is really easy now that we have the gradients for $u$ and $v$. We simply apply the chain rule for partial derivatives:

$$\partial_{\theta_p} I = \partial_{u_p} I \partial_{\theta_p} u_p + \partial_{v_p} I \partial_{\theta_p} v_p \tag{6.24}$$

$$\partial_{\phi_p} I = \partial_{u_p} I \partial_{\phi_p} u_p + \partial_{v_p} I \partial_{\phi_p} v_p \tag{6.25}$$

## 6.2.4   A Quick Note

One note worth making is that we can see that there are several terms in the above that are shared across the predictions and the gradients. So carefully holding onto these things in memory will allow us to reuse much of the computation in doing both.

## 6.3 Where Was the Gradient?

Okay so we just computed a slew of derivatives - did we get the gradient somewhere in all of that too? We sure did! The gradient is just a vector containing the partial derivatives with respect to each of the parameters. So by calculating the individual derivatives we calculated the gradient as well. Now we'll be able to take our guess, determine the error, compute the gradient, take our step and repeat! Let's see how all of that works out.

# Chapter 7

# Divergent Degenerates

While implementing gradient descent I ran into two central issues. Let's illustrate each in turn.

## 7.1   Run Away Steps

The first issue I ran into was that of divergence. This is when the gradient just keeps rising near the point where you expect it to converge and is usually a sign that the derivative isn't actually *defined* at the point of convergence.

So, for example if you take the case where the actual source location is at $\theta = 0$ and then you plot the error as a function of the guessed source location you'll get Figure 1:

Clearly the derivative is not defined at 0 and the gradient is very steep in the vicinity of 0. Practically this meant that my gradient descent algo would rush toward zero and then totally overshoot, then rush toward zero again and overshoot in the other direction, and if it didn't just diverge completely would instead ping-pong back and forth forever.

To resolve this I decided to use only the direction of the gradient (instead of both the direction and the magnitude) and then every time the error swung (as it would do during the ping-ponging) reduce the magnitude of the step I was taking. This would in turn mean that every time we overshot we'd step more lightly, thereby allowing us to converge.
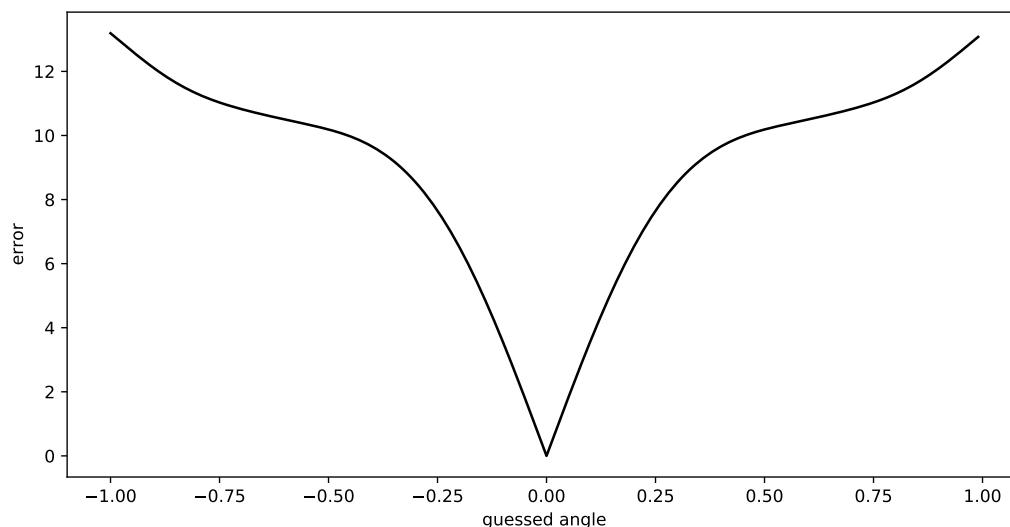
Figure 7.1: Error as a Function of the Guess Angle

## 7.2   Invisible Walls

The second issue I ran into was that depending on the guess I initially made, sometimes we'd converge to the right solution and somethings we'd converge nowhere near the solution. This turned out to be the result of degeneracies in the error function. Let's once again illustrate this with an example.

In this case we're going to have 3 sources. In reality they'll be at $\theta = 0$, $\theta = -\pi/5$, and $\theta = \pi/5$ but we're going to vary the last guess across theta and look at our error (Figure 2):

Note first that as we'd expect we see our dive to zero around the correct angle $\pi/5$. But look at those two peaks to the left! If we guessed between them our gradient would never tell us to climb those mountains of error! In other words the valley between those peaks is a local minimum - if we guess in its vicinity we'll never end up finding the true solution.

These are pretty common features of any optimization problem (and finding minimal error is an optimization problem). It's easy to find local minimums, but very hard to find the global minimum. The solution, as it turns out, is to keep making new guesses, tabulate all the spots you converged to and pick the one with the lowest minimum. Then, if you've made enough guesses (and had a little bit of luck) your lowest of the low will be the global
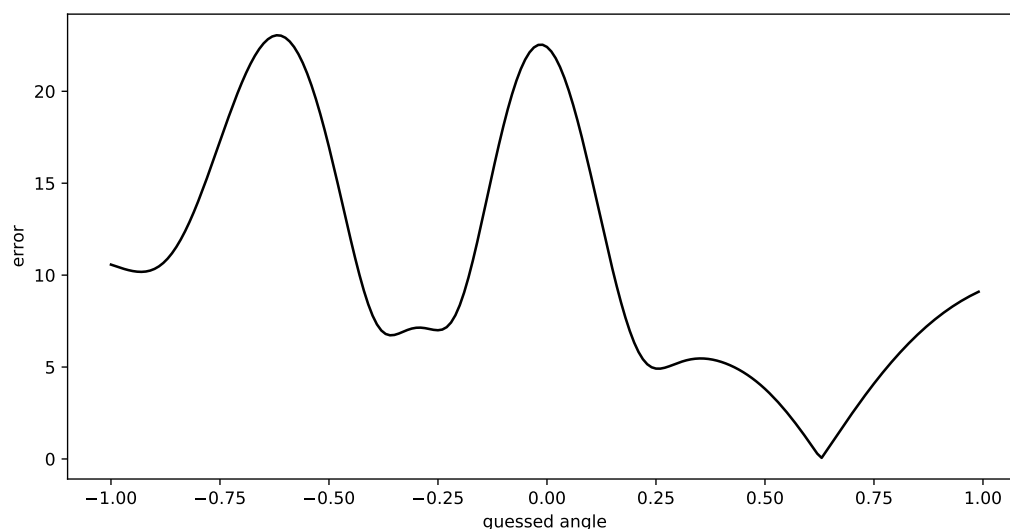
Figure 7.2: Error as a Function of the Guess Angle

minimum.

What I found was that if I picked my guesses from a probability distribution that had the same shape as my observed values and then just kept re-guessing I could pretty quickly find the global minimum - so, another issue resolved.

## 7.3 It's Alive!

And just like that we've got gradient descent working! We can input a set of observations and a guess at the number of sources and our little algo will churn away and find the lowest error it can hit. This then gives us the best set of sources and amplitudes to match our observed sweep. Pretty darn cool.

I think it's worth pausing for just a moment and appreciating the power of this. While our phased array itself didn't have the resolution to isolate sources, by building some machine learning around it we've been able to do just that. So now we can build far less grand (and therefore in our case actually usable) phased arrays while still being able to isolate (at least in theory) the particular sources generating the sound! With this tool and the bandwidth filters we'll discuss shortly, we should have a reasonable chance

at the simultaneous, omni-directional listening we've been shooting for.

Finally if you're interested in using this tool or checking out the code, it can be found at my GitHub Project *Project Fjorgyn* in the following repository: https://github.com/Project-Fjorgyn/croac.

# Chapter 8

# What the Fourier

In all of this discussion of locating point sources using gradient descent you may have noticed that we've been working with a single frequency at a time. That was part of original design - divide and conquer. While we've made good progress in understanding how to position things once that division has been made, we've conveniently ignored how we're even going to create the division in the first place? How are we going to take a very complicated superimposed amalgam of sound waves and filter down to a single frequency's contribution? Well that, conveniently, is the purpose of this section.

## 8.1  Looking at Listening

So far we've been talking about continuous complex functions and while that may in fact be a good model of what's going on it certainly doesn't capture what measurements look like. There are three basic reasons for this. First of all, as should be pretty obvious, it's not actually possible for us to take complex measurements - our measurements will always be real valued. Second we can't actually measure a continuous signal - doing so would require an infinite sampling rate and produce infinite amounts of data. Finally, because we'll be working in the digital world of computers, we have to represent our signal in bits. Given each record of our signal will need to represented by a finite number of bits, that signal will be limited to a finite set of values. For example if we were using only one bit to record each sample of our signal we could only record *on* and *off*. If we instead use two bits we'll end up with four possible values. In general for $n$ bits we'll have $2^n$

distinct values we can record. So the greater the bits per sample, the higher the fidelity, but also the larger the overall data. Overall then with both our sample and our bit-count we're doing a kind of trade between space (data storage) and fidelity. Let's get a better sense of this through examples.

### 8.1.1   The Rate of Sampling

We'll start with a very simple continuous wave to get us going. If we treat the $x$-axis as time in seconds and the $y$-axis as our observed amplitude the Figure 1 gives us the continuous *real* signal from a 1 Hertz wave of amplitude 1.
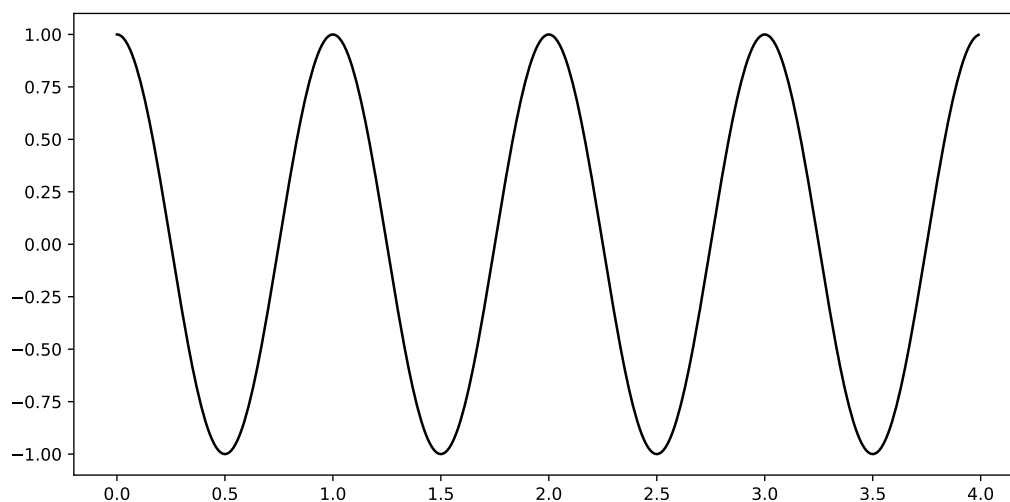


Figure 8.1: The Continuous Signal

Let's begin by sampling this wave. If we sample the wave every second we'll get Figure 2.

It doesn't even look like a wave anymore! This is because we're sampling too slow to capture the characteristics of a 1 Hz wave. Well according to *Shannon's sampling theorem* [3] if the highest frequency in our signal is $f$ our sampling frequency $f_s$ must obey:

$$f_s \geq 2f \tag{8.1}$$

So in our case we need to sample at least twice per second. We'll go a little farther than this and sample four times a second (Figure 3).
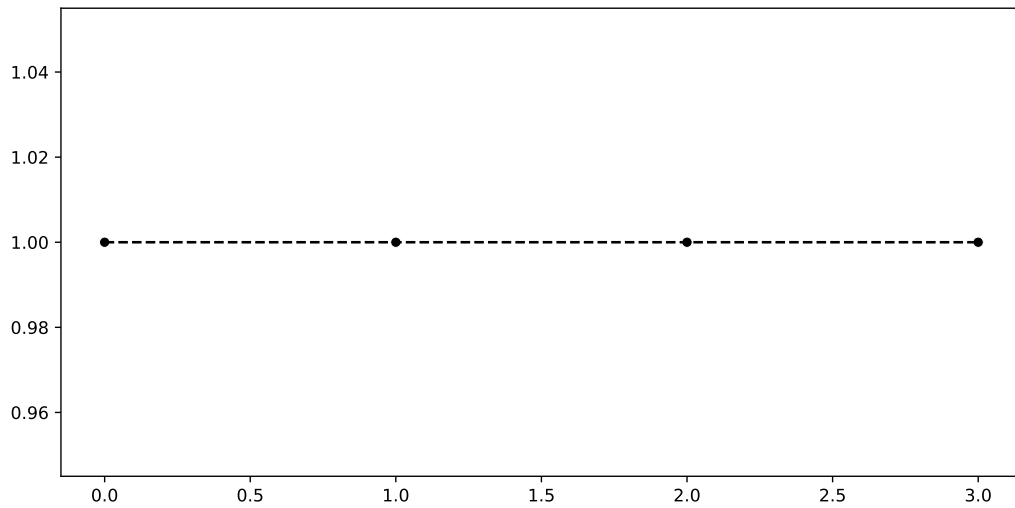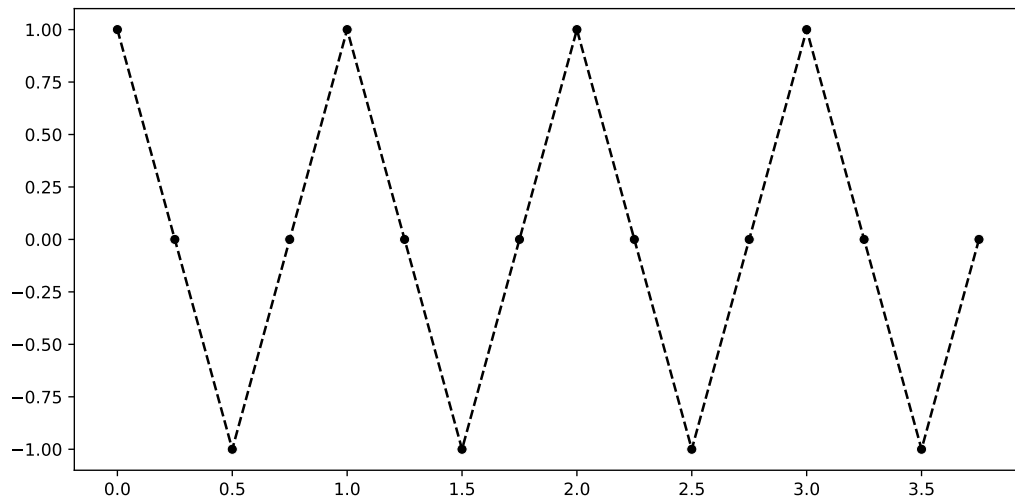
Figure 8.2: Sampling Once Per Second



Figure 8.3: Sampling Four Times Per Second

This is a much better representation of our wave. Obviously as we keep adding more samples we'll get better and better resolution but this is enough sampling to determine our amplitude, frequency, and phase. In general the takeaway is that in order to capture our full signal we need to sample at a frequency at least twice that of our highest frequency. If we sample lower

than that we'll start loosing information on those higher frequencies.

## 8.1.2   Digital Discretion

Alright let's now look at the effect digitization has. We'll once again start with a super simple example - two waves of the same frequency but different amplitudes and phase. We're also going to keep our sampling rate infinite, just to get a better sense of what digitization does. Figure 4 shows us our two continuous, un-digitized signals.
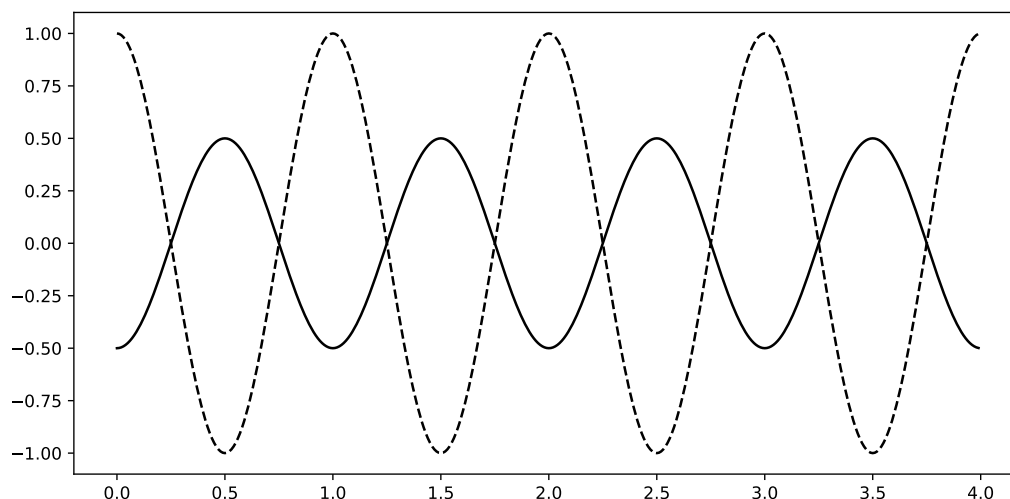


Figure 8.4: Two Signals

Now let's take the most extreme example - a single bit representation of these signals. With bits we have to choose what each value should mean so in this case we're going to say that the bit being on means we're observing a signal greater than 0 and the bit being off means the opposite. Figure 5 shows us our observed signals under this representation.

Immediately we can see we've lost any information our amplitude. Both of our signals look exactly the same just offset by phase! So we obviously need more bits in our representation? What happens if we use 3 bits instead? This will give us $3^4$ or 8 distinct values (Figure 6).

This is much better. We can see there's a difference in amplitude, and at least for one of the waves that measurement is accurate. But if you look at the solid (not dashed) signal you'll see that the amplitude is being measured
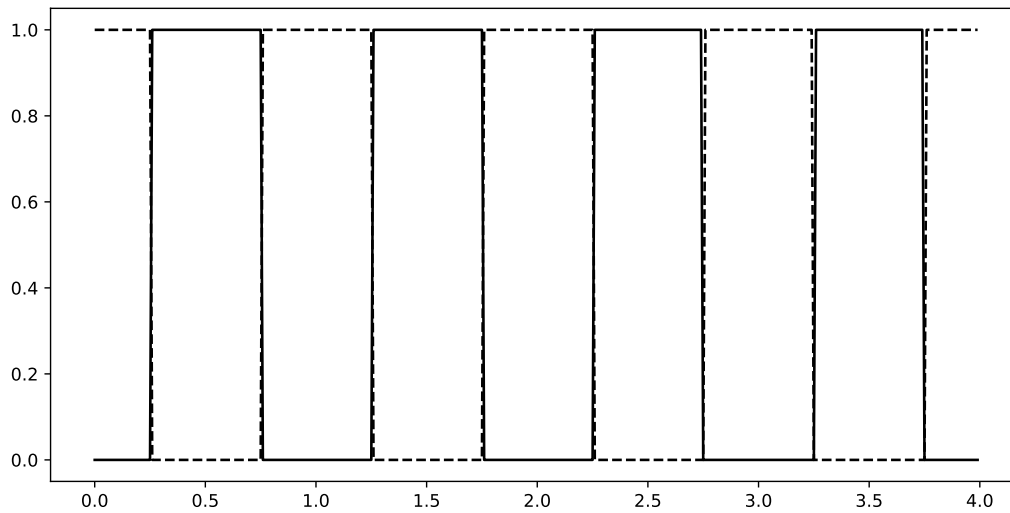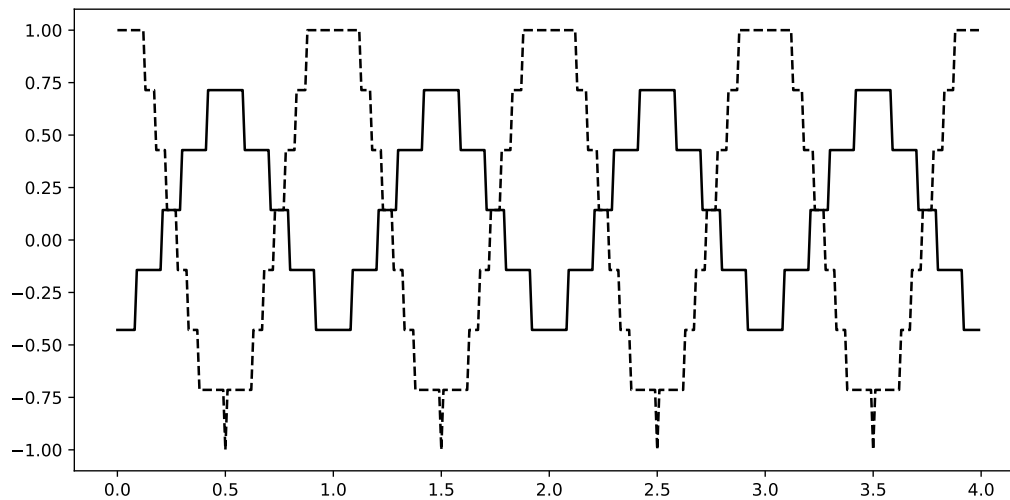
Figure 8.5: Single Bit



Figure 8.6: Three Bits

as larger than it actually is! This is once again because our digitization is turning our signal into discreet bins of values - so we lose fidelity on the actual amplitude of the signal.

In general people tend to use 12 bit representations and higher which have a pretty high level of fidelity (Figure 7). So we should be generally alright,

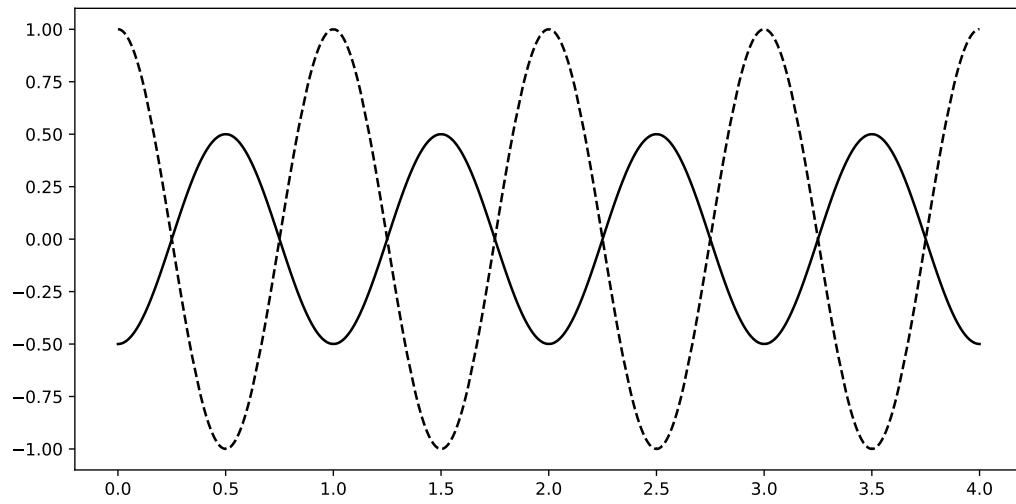but it's just an important thing to note and keep an eye on.



Figure 8.7: Twelve Bits

## 8.2 Breaking out Amplitudes

Now that we've got what the signal looks like under our belts it's time to move onto how we're going to break out the different frequencies and amplitudes in our underlying superimposed signal. We're already seen that we need to make sure we're sampling at at least twice the rate as the highest frequency we're interested in, so from now on we'll be assuming that we are doing just that. So how are we going to decompose our signal into its various pieces? The answer is a wonderful piece of mathematics called the *Discrete Fourier Transform* [6].

This transform takes a series of $N$ samples $\{x_n\}$ and transforms them into a second set of complex numbers $X_k := X_0, X_1, ..., X_{N-1}$ using the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi}{N}kn} \tag{8.2}$$

How is this transform going to help us sort out what's going on per frequency? Well suppose that we gather our samples by taking a series of time samples of our signal. Our first sample would correspond to $n = 0$ and our last sample would correspond to $n = N - 1$. Suppose we were sampling at 100Hz (i.e. we're interested in frequencies below 50Hz). That would mean that the time our sample was taken at would be $t = n/100$. Suppose we sampled for 2 seconds, i.e. $N = 200$. These two specifications means we can rewrite the exponential term in our formula above as:

$$e^{-\frac{i2\pi}{N}kn} = e^{-\frac{i2\pi}{200}k100t} = e^{-i2\pi kt/2} \tag{8.3}$$

Which is just the equation for a wave operating at frequency $k/2$ Hz. If we rephrase our original formula like this where $x_n = x(t = n/100)$:

$$X_k = \sum_{n=0}^{N-1} x(t = n/100)e^{-i2\pi kt/2} \tag{8.4}$$

things start becoming a little clearer. What this formula is doing is multiplying a wave of amplitude 1 (our exponential) by our samples. Now if our samples have a component from a wave of frequency $k/2$ then that component and our exponential will "correlate". On the other hand, those components with different frequencies won't "correlate". Therefore, intuitively our $X_k$ will preferentially capture information about the component of our wave at

frequency $k/2$. More generally if we took measurements for $m$ seconds then we'd expect $X_k$ to capture information on wave components at frequency $k/m$. Given $k$ ranges from 0 to $N-1$ where $N = f_s m$ we can see that we're going to capture information on frequencies up to but not including $N/m = f_s$.

Continuing along this line of intuition we know that our exponential will not contribute to the magnitude of our sum. Therefore the magnitude will be entirely determined by our samples, and, if our intuition is correct, specifically the component of our samples that corresponds to a frequency of $k/m$. Therefore the relative size of the $X_k$ should be related to the amplitude of our $k/m$ component.

Alright, that's the intuition. Does it pan out? Indeed it does. This transform takes us from the time domain (time samples) to the frequency domain (frequency samples if you will). Specifically the inverse of this transform is:

$$x_n = \sum_{k=0}^{N-1} \frac{X_k}{N} e^{i2\pi kn/N} \tag{8.5}$$

which is just a superposition of waves of frequencies $k/m$ and amplitudes $|X_k|/N$. So the $|X_k|/N$ correspond to the amplitudes of our various components! Decomposition accomplished!

## 8.3   Where's the Catch?

Okay before moving onto how this relates to our phased array fingerprints let's take a step back and note the design implications we've run into thus far.

We know that in order to get *any* information on a particular frequency we need to sample at a rate at least twice as great as that frequency. But we also know that if we sample at a rate $f_s$ our discrete fourier transform can tell us about all the frequencies up to but not including $f_s$. Obviously then our requirement for a $2\times$ sampling rate is going to overcome our transform limitations but it does point out that while we will be given information on frequencies near our sampling rate from the transform that we shouldn't actually trust those values. Therefore in general our Shannon sampling rate takes precedence.

On the other hand our transform has a variable beyond our sampling rate

- the sampling time. Now if our sampling time is very short we'd expect to start losing information on the lower frequencies as wouldn't have enough time to actually see them. In general we'll want to use as a rule of thumb that we should sample for a considerably longer time than it takes for any of the frequencies we're interested to cycle. For example if we were interested in 100Hz signals as our floor than sampling for only a hundredth of a second would be far too little time to gather information for our transform.

Alright, back to the scheduled programming.

## 8.4 Scanning for Fingerprints

We've now got a way to decompose our signal into frequency and amplitude pairs. So how are we going to get our phased array fingerprints? Easy peasy - we'll take the following steps:

1. We record the signal from each of our microphones at a high enough frequency to capture the full range we're interested in

2. For each of our scan angles we introduce our time delays and superimpose the signals

3. We break these signals into windows that are large enough to capture the lower range of the frequencies we're interested in

4. We decompose each of these window/scan-angle pairs into frequencies using our DFT (discreet fourier transform)

5. Using the derived amplitudes per scan-angle we reconstruct our fingerprints

And as a final note, it turns out that a straight DFT is quite slow computationally so we'll be taking advantage of Fast Fourier Transforms to get some speedup.

# Bibliography

[1] Chris. Level of noise in decibels (db) level comparison chart.

[2] Zoe Cormier. The loudest voice in the animal kingdom.

[3] Edmund Lai. *Practical Digital Signal Processing*. 2003.

[4] Robert J. Mailloux. *Phased Arrays in Radar and Communication Systems*. 1994.

[5] et. al Miguel A. Acevedo. Automated classification of bird and amphibian calls using machine learning: A comparison of methods. 2009.

[6] Wikipedia. Discrete fourier transform.

[7] Wikipedia. Near and far field.

[8] Wikipedia. Piano key frequencies.

[9] Wikipedia. Sound localization.