CROAC: Counting and Recognition using Omnidirectional Acoustic Capture

Marcel Gietzmann-Sanders

July 18, 2022

Contents

1 The Pond 7

2 The Richness of Sound 9
- 2.1 Blue Whales Yelling - Amplitude 9
- 2.2 To Tune or Not to Tune - Pitch 11
- 2.3 The Quality of Sound - Superposition 12
- 2.4 Vanishing Act - Phase 15
- 2.5 Sums and Summaries - Mathematical Formalization 17

3 To Phase Array 21
- 3.1 Back at the Pond 21
- 3.2 On Many Eared Aliens - Planar Phased Arrays 23
- 3.3 Enough Math, Show Me Something 27

4 How Tall is A-flat? 31
- 4.1 Getting Greedy - Number of Elements 31
- 4.2 Being Skinny Isn't Always Great - Element Spacing 33
- 4.3 Far A Field 36
- 4.4 Sound is Big - Designing the Antenna 37

5 Playing Clue 39
- 5.1 The Basic Idea 39
- 5.2 Looking at Prints 40
- 5.3 The Descent of Math 42
- 5.4 They Say Summaries are Good 44

6 A Very Derivative Section 47
- 6.1 The Model 47

6.2 The Gradient 48
    6.2.1 Position 49
    6.2.2 Phase 50
    6.2.3 Returning to the Angles 50
    6.2.4 A Quick Note 50
6.3 Where Was the Gradient? 51

7 Divergent Degenerates **53**
7.1 Run Away Steps 53
7.2 Invisible Walls 54
7.3 It's Alive! 55

8 What the Fourier **57**
8.1 Looking at Listening 57
    8.1.1 The Rate of Sampling 58
    8.1.2 Digital Discretion 60
8.2 Breaking out Amplitudes 63
8.3 Where's the Catch? 64
8.4 Scanning for Fingerprints 65

9 Mechanical Ears **67**
9.1 Facing Reality 67
9.2 Listening with Capacitance 68
9.3 Send Me the Deets 71
    9.3.1 Sensitivity 71
    9.3.2 SNR 72
    9.3.3 Frequency Response 72
    9.3.4 Electrical Properties 72
    9.3.5 Summary of the Deets 74
9.4 Hearing Voices 75
    9.4.1 Let's Not Listen to the Power Supply 75
    9.4.2 No DC Please 77

10 Digitize Me **81**
10.1 It's Complicated 81
10.2 One Step at a Time 83
10.3 Too Much *is* a Good Thing 84
10.4 Magic Filters 86

10.5 Devils in the Details 87

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

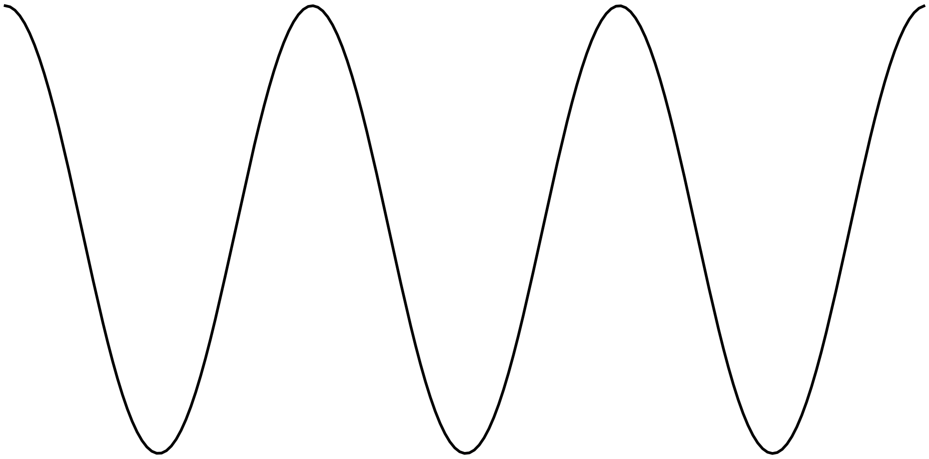

Figure 2.1: A Simple Wave

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.

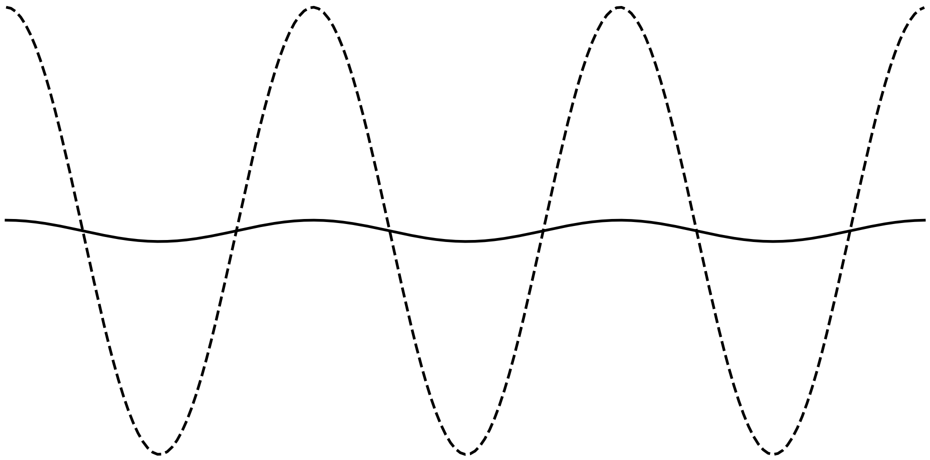

Figure 2.2: Blue Whale v. Rock Concert

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 [13] 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.

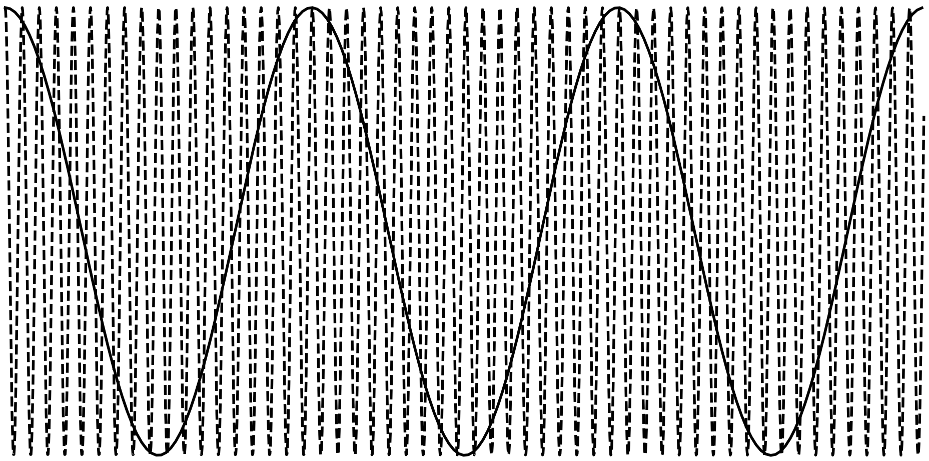

Figure 2.3: A440 v. The Highest Note on a Piano

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.).

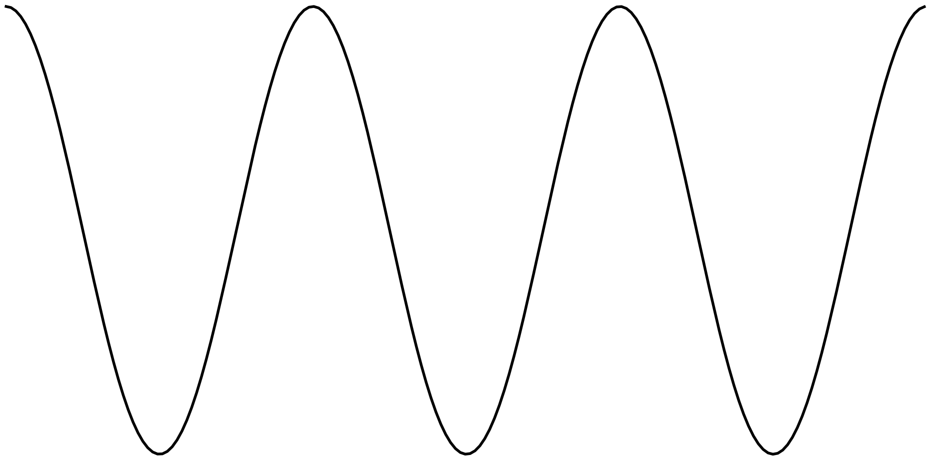

Figure 2.4: Fundamental Frequency

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

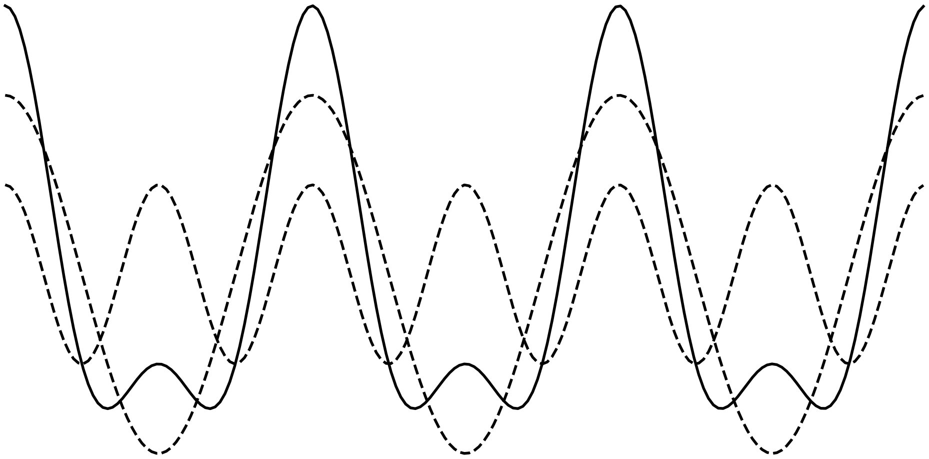

Figure 2.5: Super Imposing Two Waves

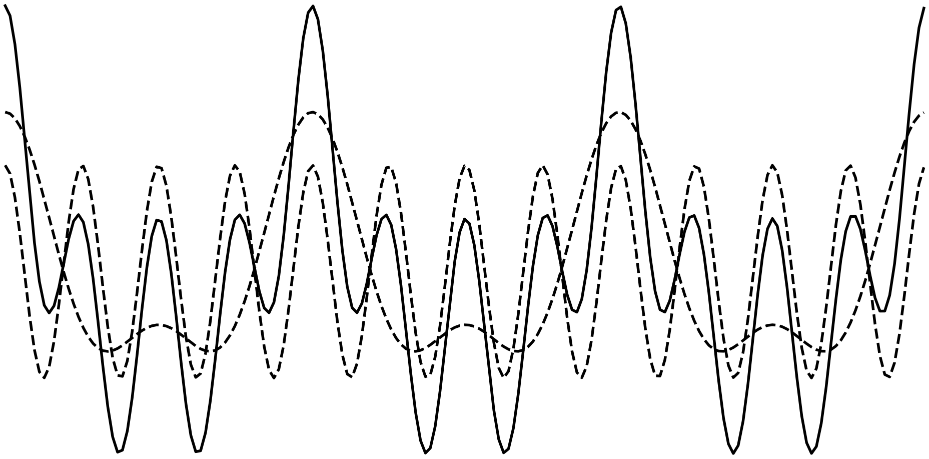

Figure 2.6: Three Frequencies

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.

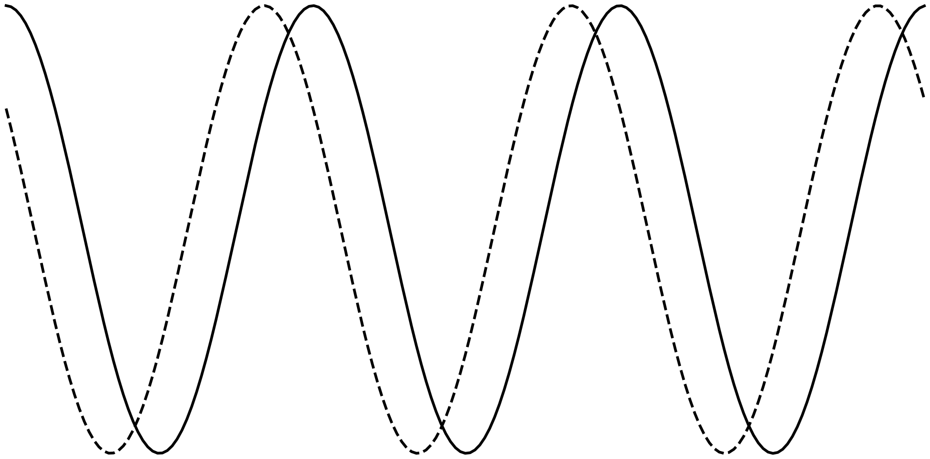

Figure 2.7: Two Waves with Different Phases

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**.

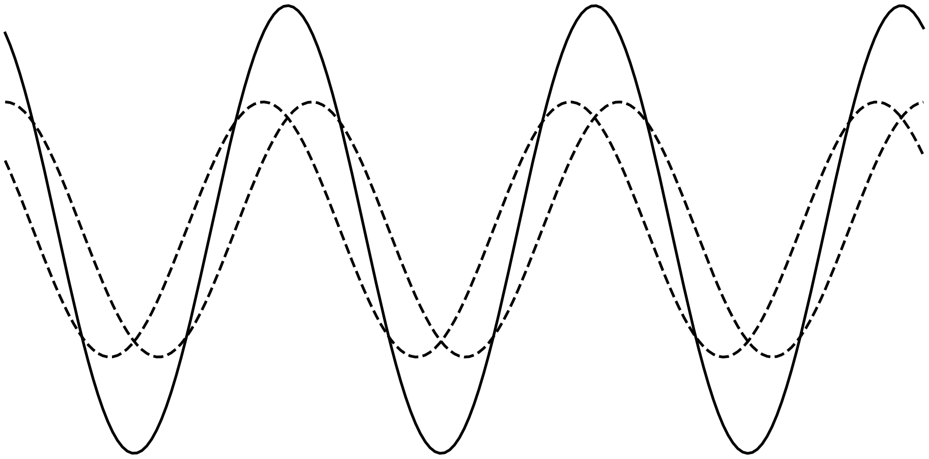

Figure 2.8: Constructive Interference

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

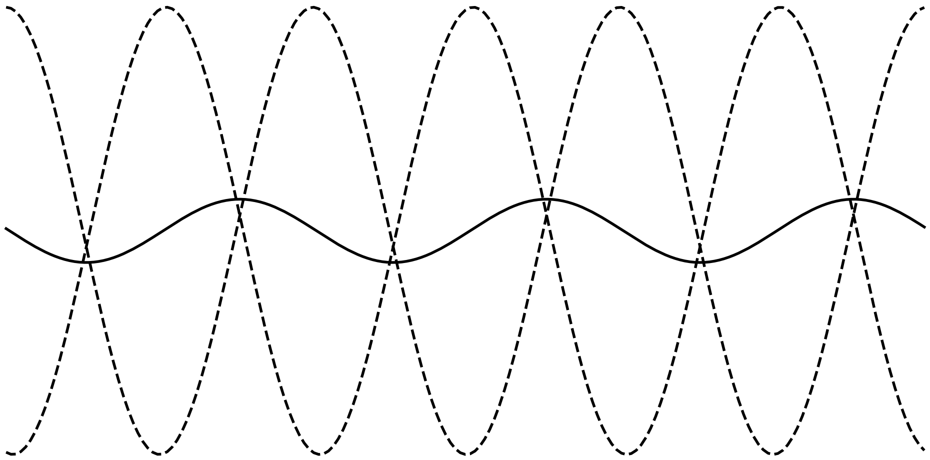

Figure 2.9: Destructive Interference

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! [14]. 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, ψ (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:

$$y = a_1e^{i\psi_1}e^{ik_1x} + a_2e^{i\psi_2}e^{ik_2x} + ... + a_Ne^{i\psi_N}e^{ik_Nx} \tag{2.3}$$

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 Σ.

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 Σ you would write:

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

With Σ 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 Σ) where n starts at 1 (expression below the Σ) and goes all the way to 100 (number above the Σ)". 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* [9]. 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) θ and ϕ.

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 jth element of our array as R_j. In other words:

$$y = ae^{ikR_j} \tag{3.2}$$

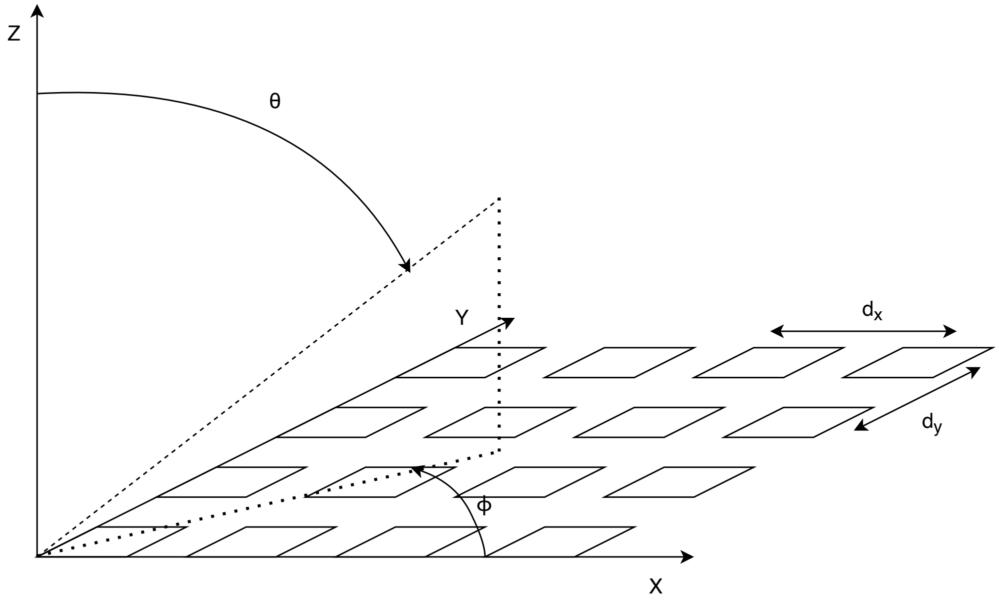

Figure 3.1: A Planar Phased Array

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 $(0,0)$, $\mathbf{\hat{r}}$ 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 jth element (see Figure 2). In the case where R_j is very large we find:

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

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 jth 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 jth 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.

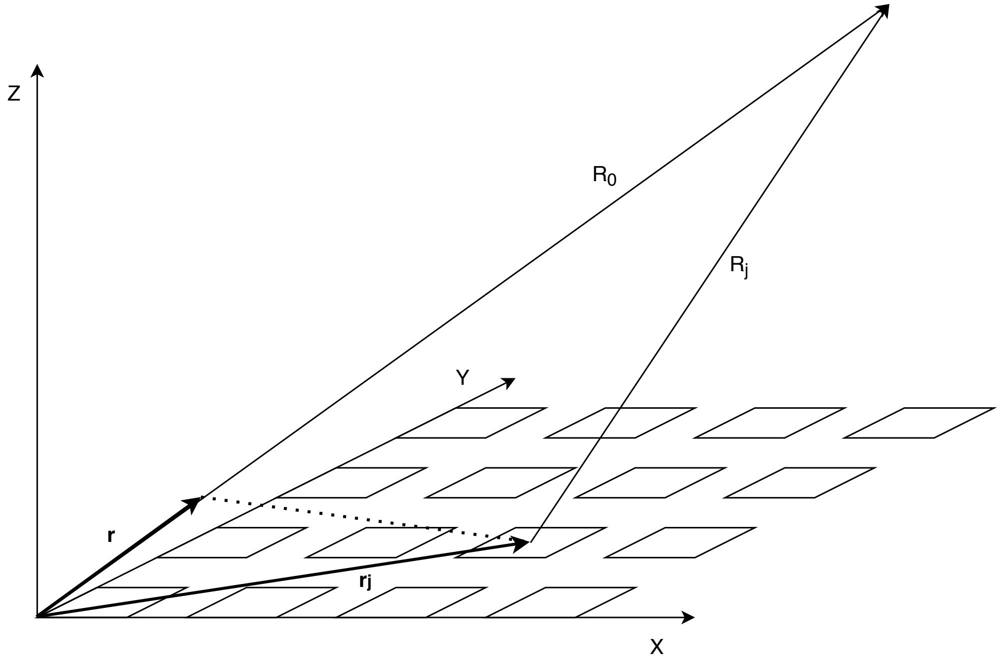

Figure 3.2: Approximation of R_j

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:

$$ae^{ik(md_xu+nd_yv)}e^{i\psi} \tag{3.12}$$

Now if we set ψ 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 θ and ϕ. Therefore:

$$\psi = -k(md_xu + nd_yv) \tag{3.13}$$

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 $\psi = -k(md_xu_0 + nd_yv_0)$ 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 Σ 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 θ (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

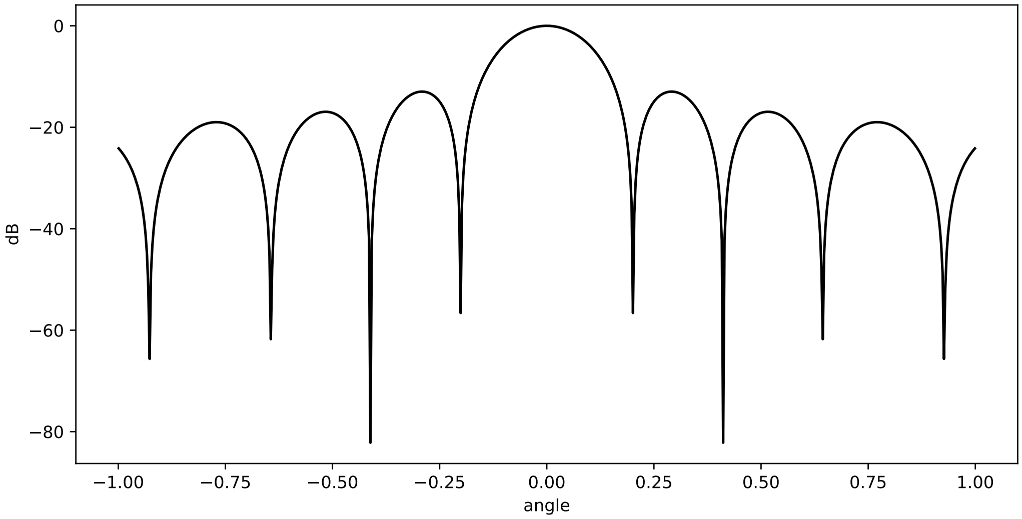

Figure 3.3: Steering Angle $\theta = 0$ with 10 Element Array

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

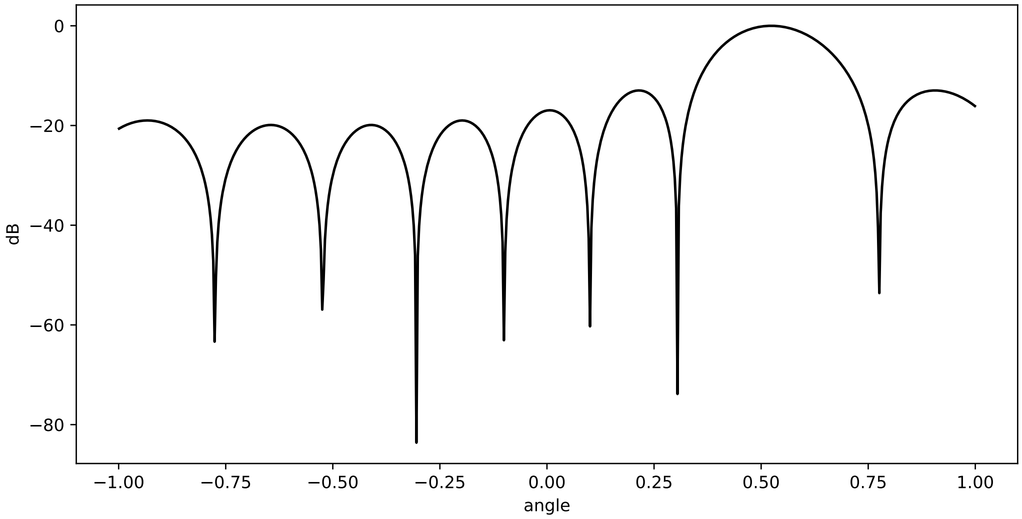

Figure 3.4: Steering Angle $\theta = \pi/6$ with 10 Element Array

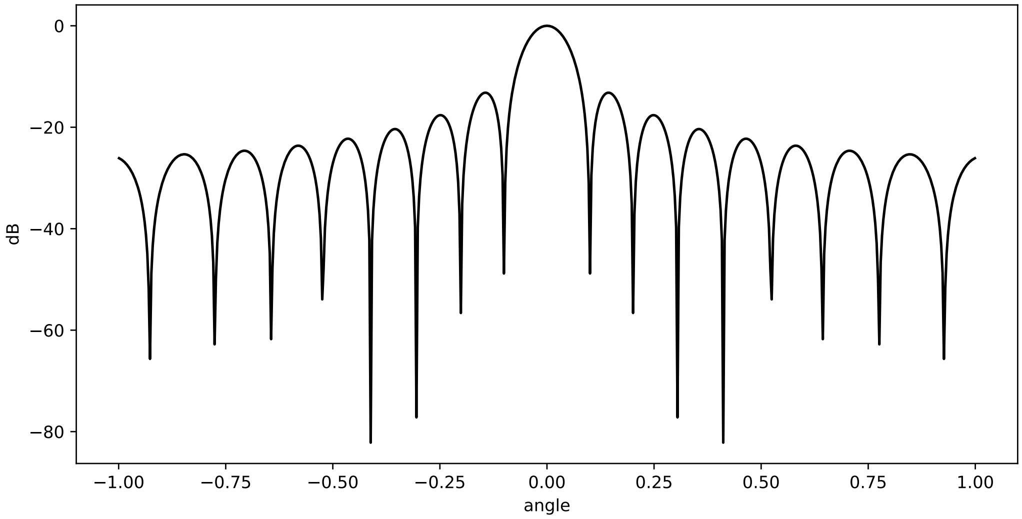

Figure 3.5: Steering Angle $\theta = 0$ with 20 Element Array

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

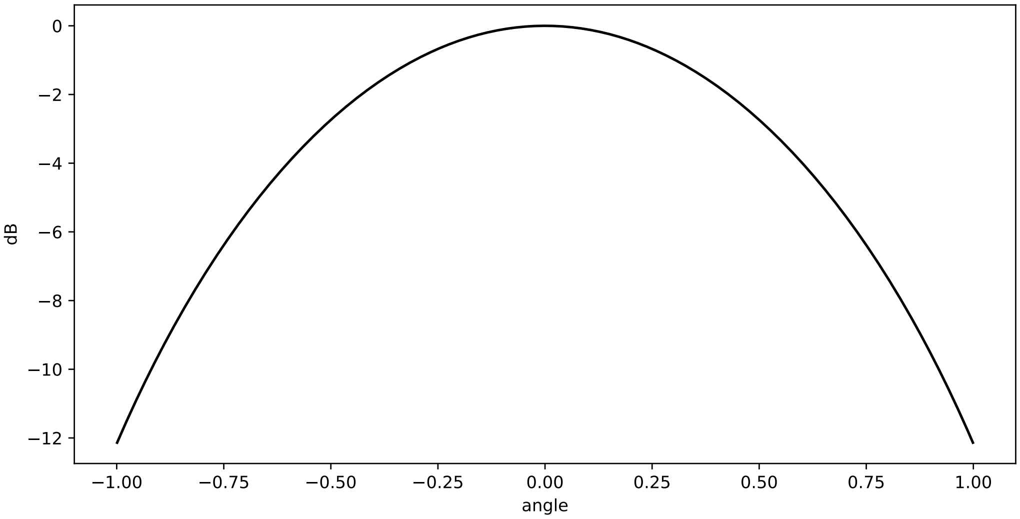

Figure 4.1: Steering Angle $\theta = 0$ with 2 Element Array

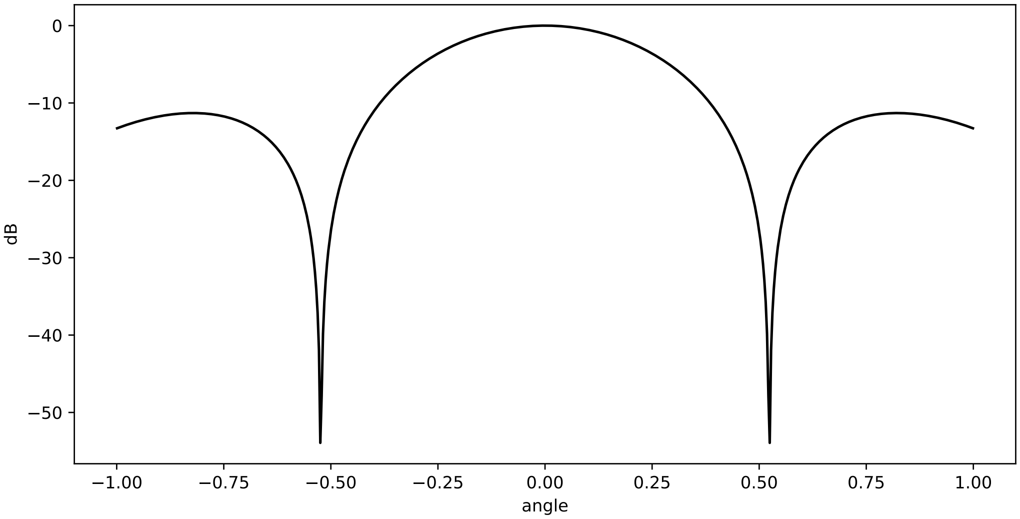

Figure 4.2: Steering Angle $\theta = 0$ with 4 Element Array

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

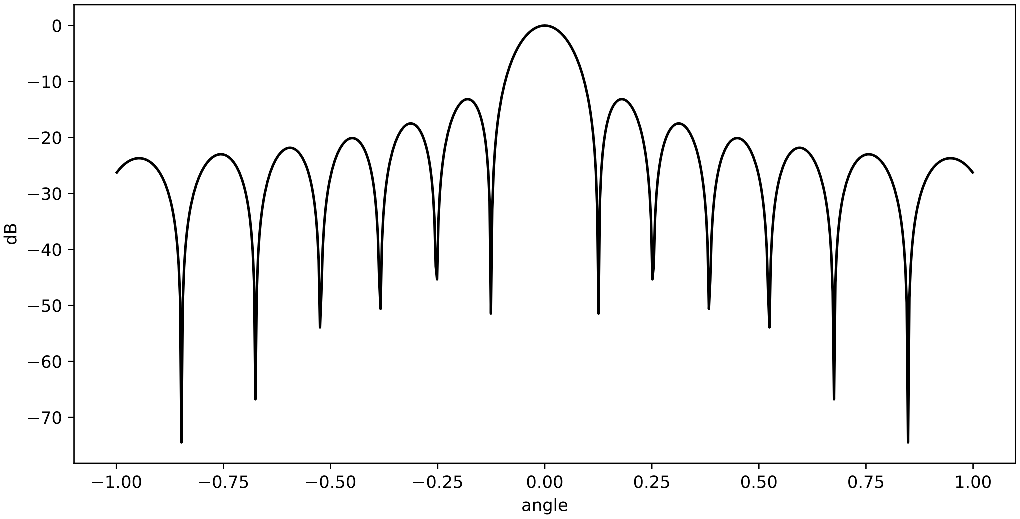

Figure 4.3: Steering Angle $\theta = 0$ with 16 Element Array

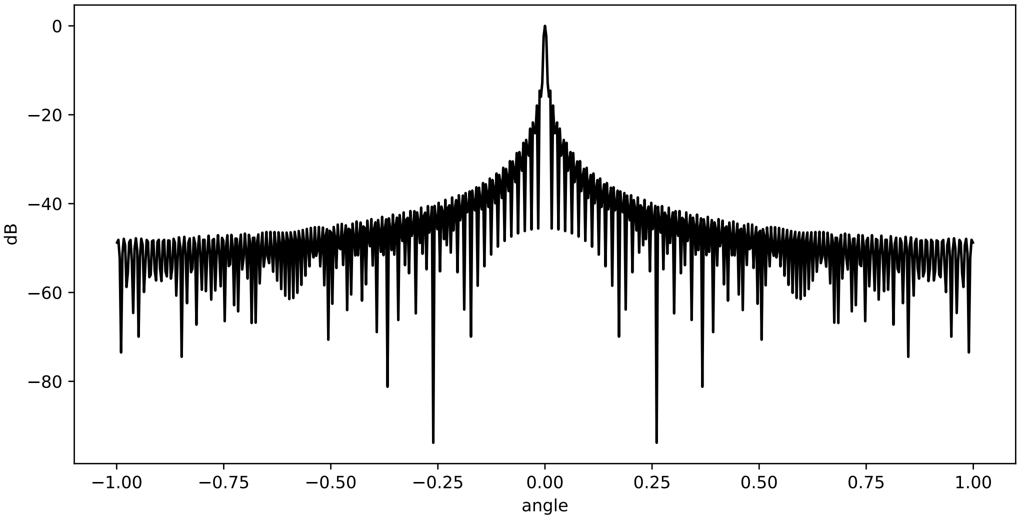

Figure 4.4: Steering Angle $\theta = 0$ with 256 Element Array

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):

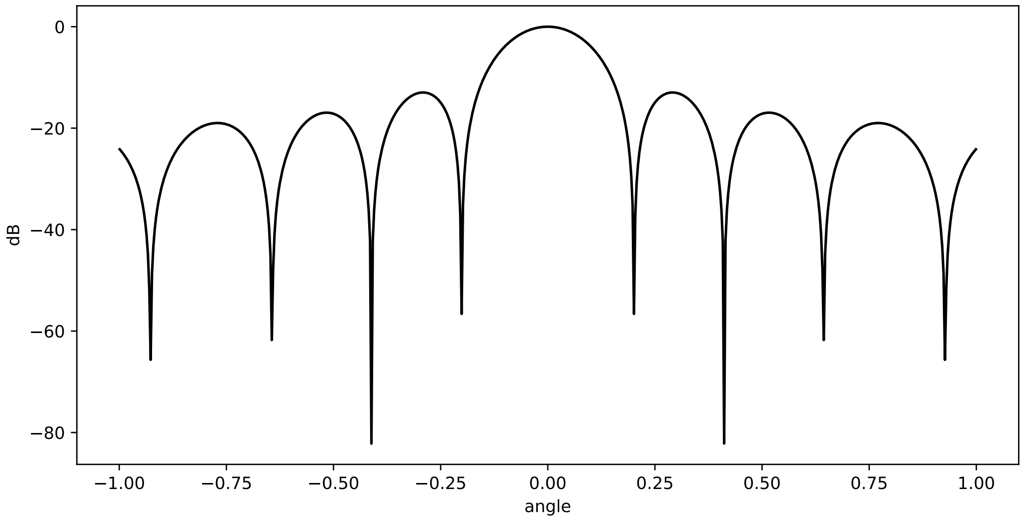

Figure 4.5: Half Wavelength Spacing

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 2π 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

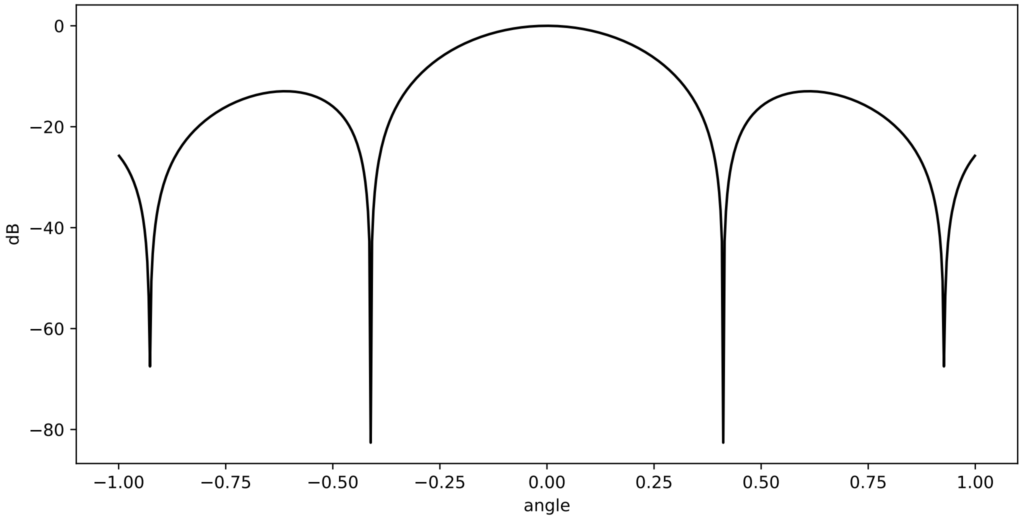

Figure 4.6: Quarter Wavelength Spacing

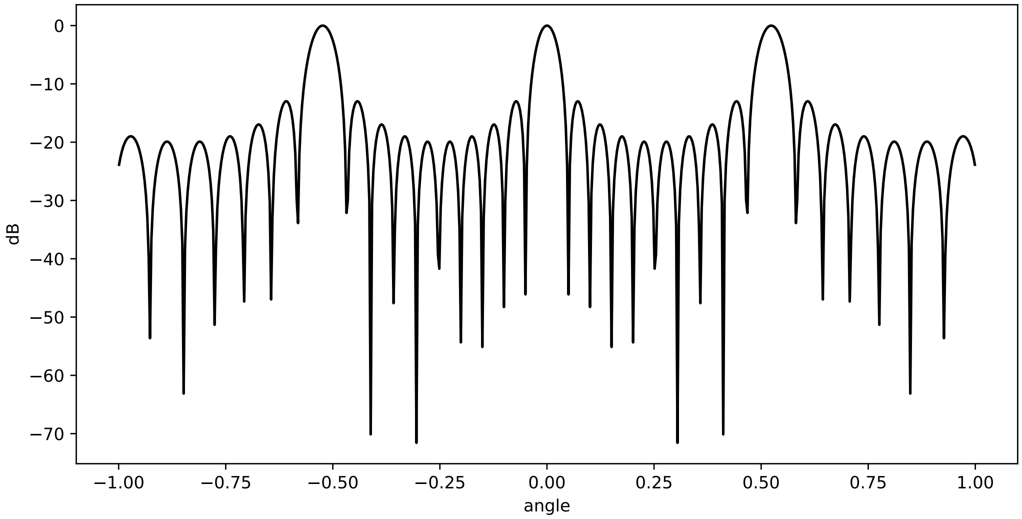

Figure 4.7: Two Wavelength Spacing

the spacing increases it becomes possible to hit multiples of 2π 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** [12]:

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

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 [10]. 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 (≈ 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 ≈ 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

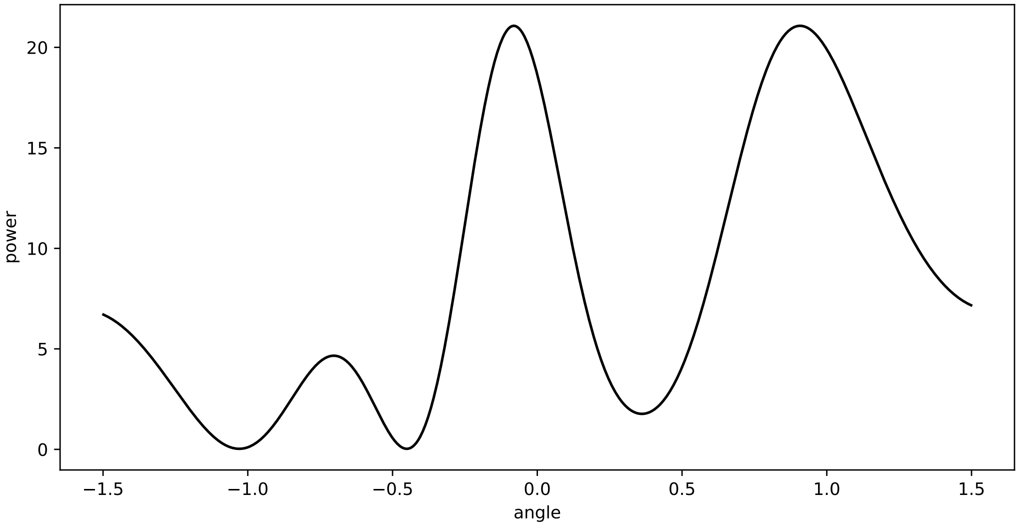

Figure 5.1: 0 and $\pi/4$ radians

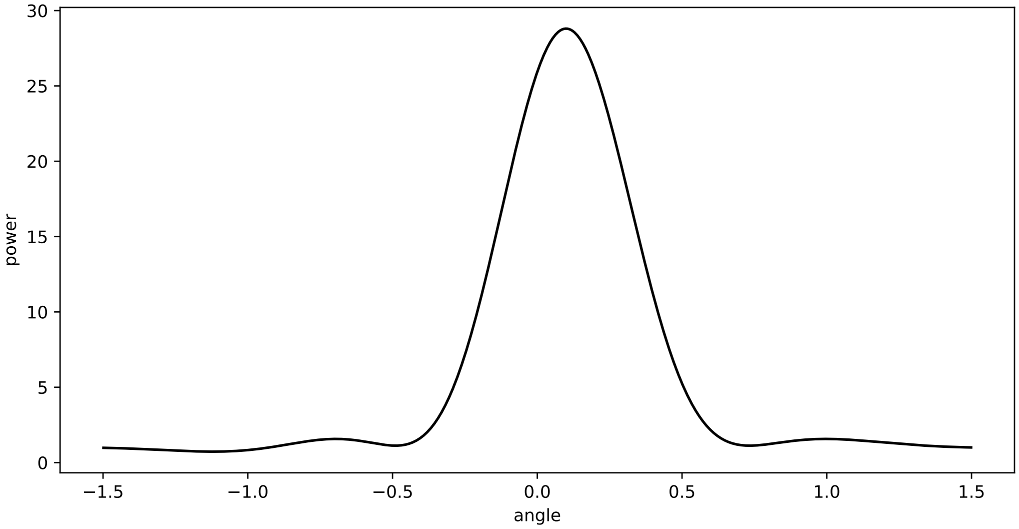

Figure 5.2: 0 and 0.2 radians

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).

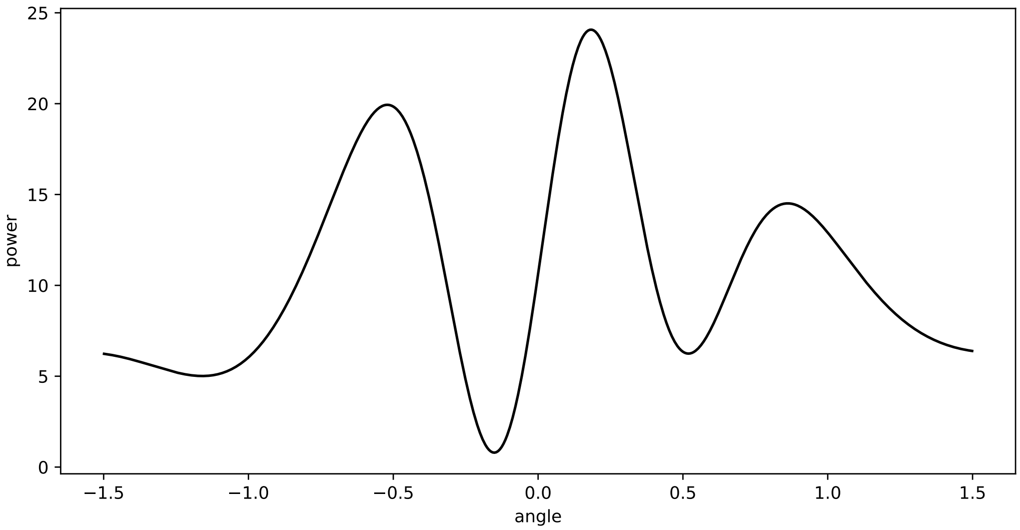

Figure 5.3: -1, -0.3, 0, 0.2, and 1.2 radians

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:

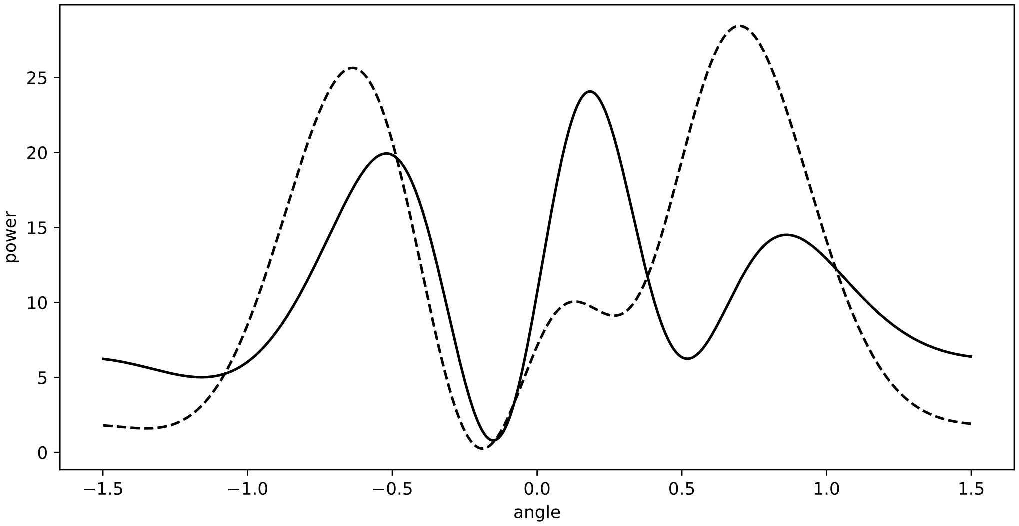

Figure 5.4: A First Guess (dashed line is the guess)

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.

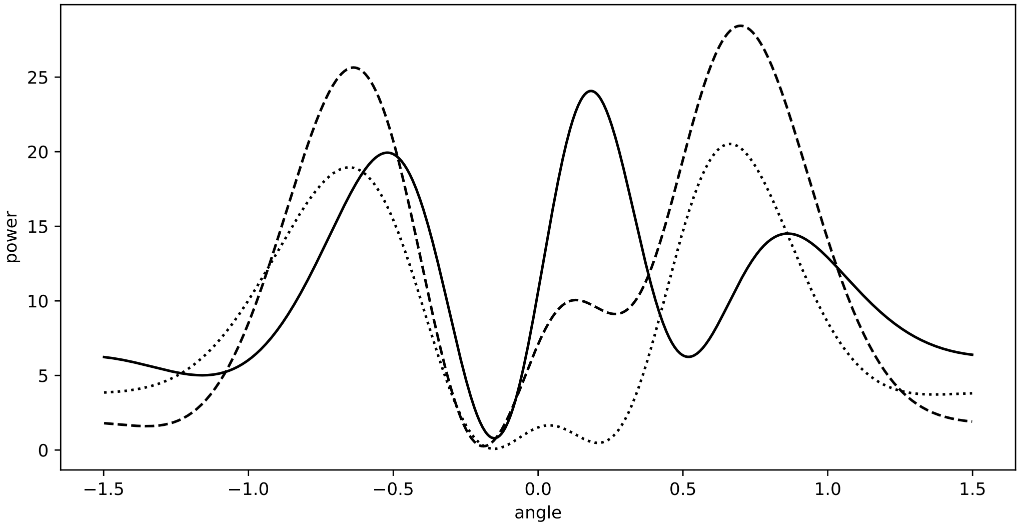

Figure 5.5: A Second *Educated* Guess (dotted line is the new guess)

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 θ, ϕ 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 θ_0, ϕ_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:

$$I(\theta,\phi) = \sum_{p=1}^{P} I_p(\theta,\phi) = \sum_{p,m,n} a_p \exp\{ik[md_x(u_p - u) + nd_y(v_p - v)]\}exp\{i\psi_p\} \tag{6.6}$$

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 θ, ϕ:

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

This then is the mathematical representation of our model which has parameters u_p, v_p, a_p, ψ_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:

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

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

Immediately we see that:

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

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

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.

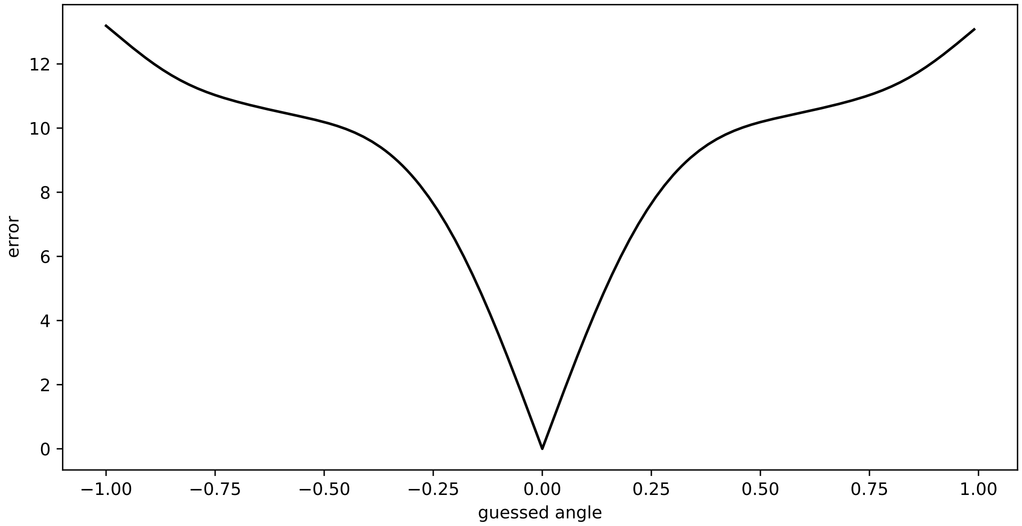

Figure 7.1: Error as a Function of the Guess Angle

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

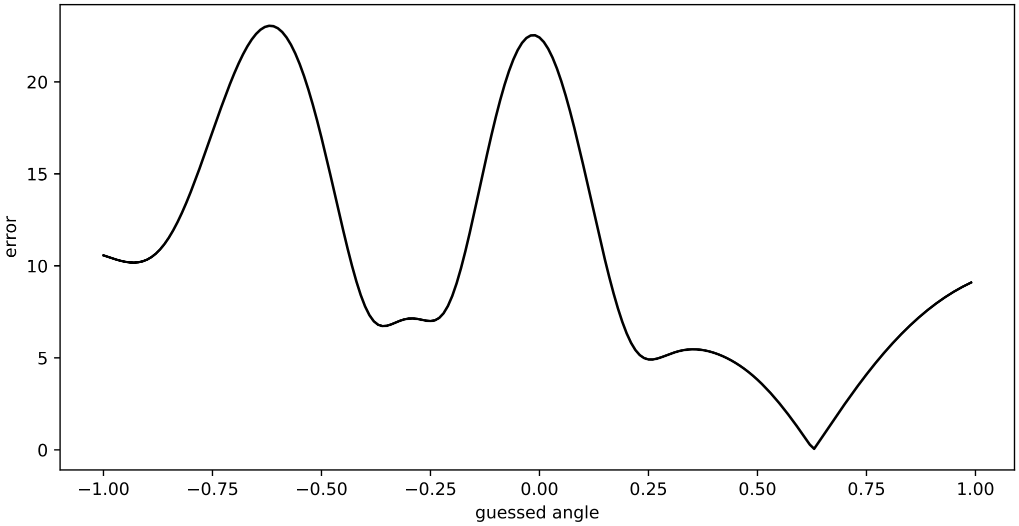

Figure 7.2: Error as a Function of the Guess Angle

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.

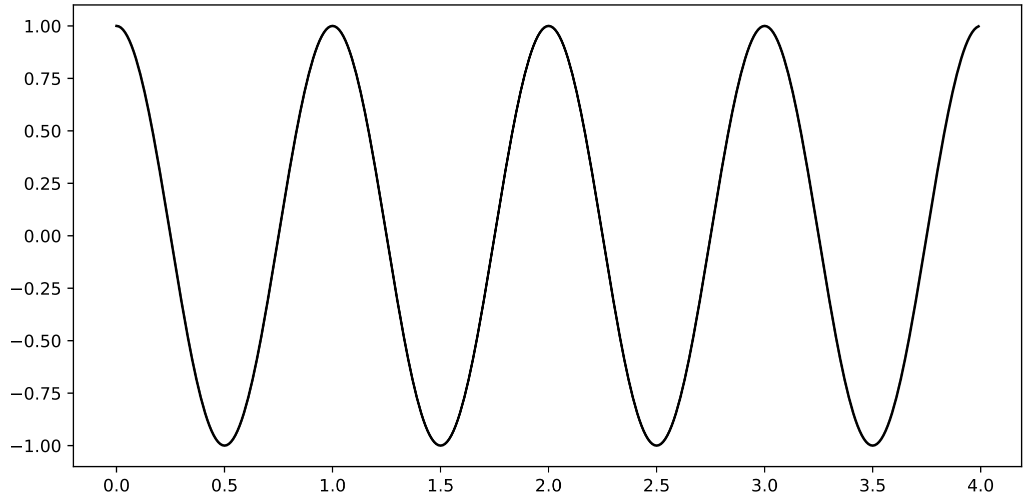

Figure 8.1: The Continuous Signal

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* [7] 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).

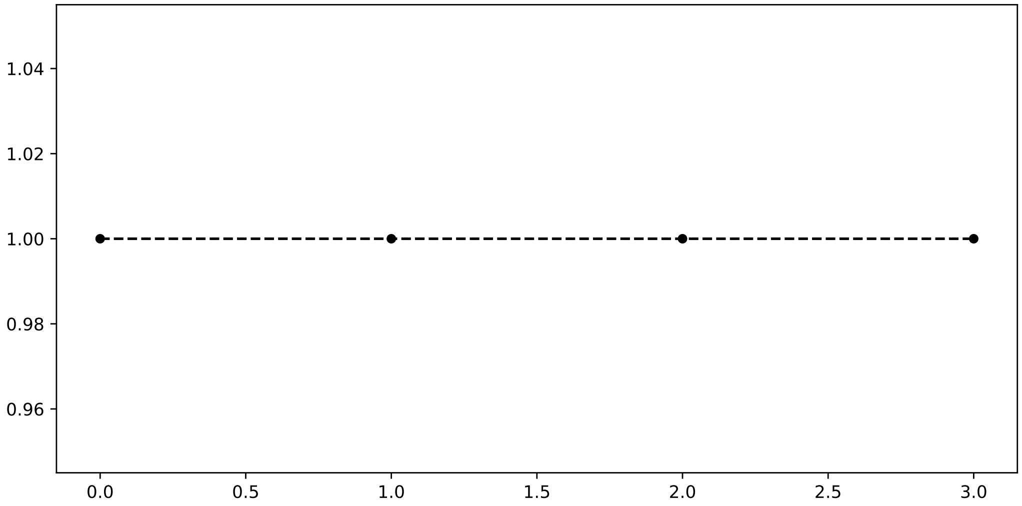

Figure 8.2: Sampling Once Per Second

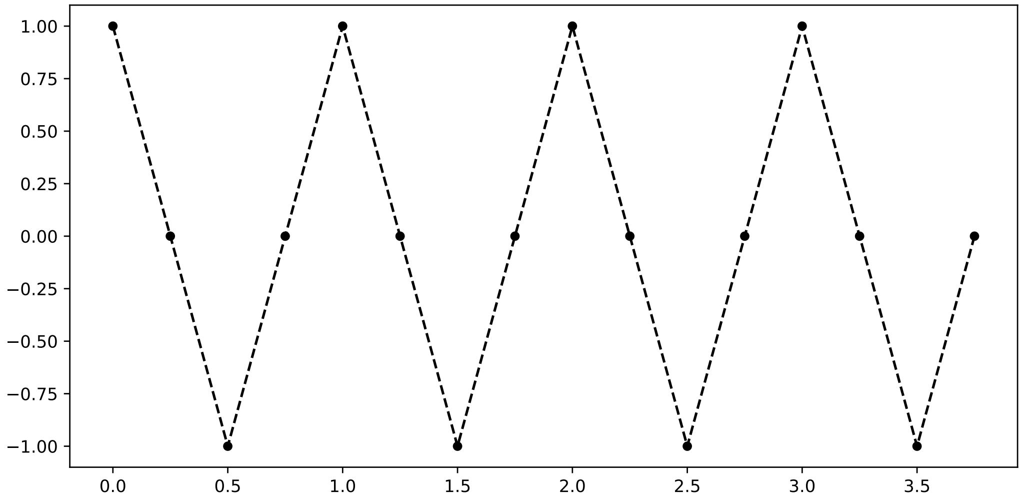

Figure 8.3: Sampling Four Times Per Second

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.

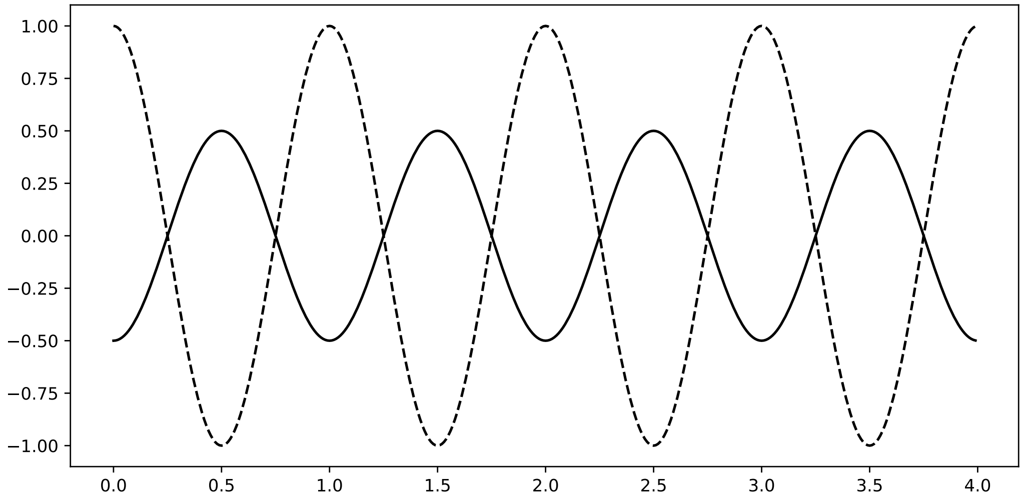

Figure 8.4: Two Signals

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

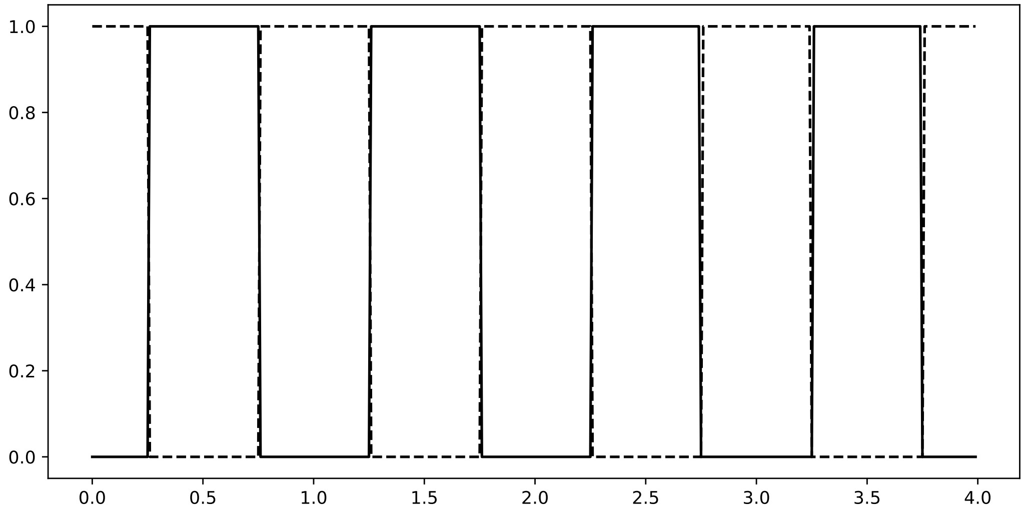

Figure 8.5: Single Bit

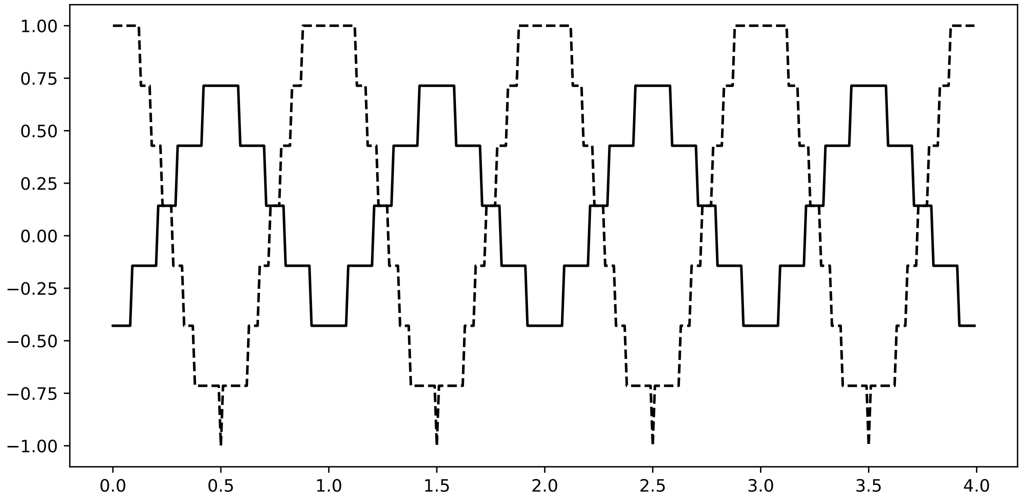

Figure 8.6: Three Bits

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.

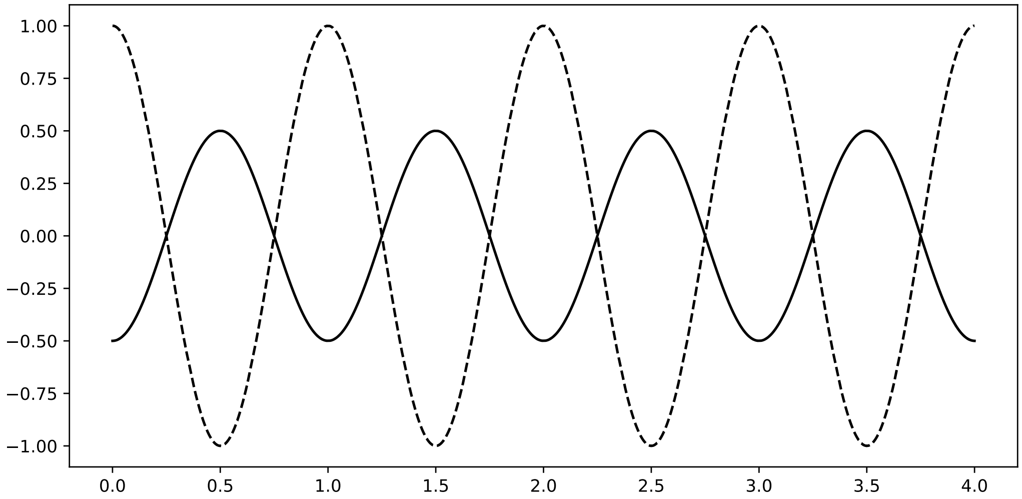

Figure 8.7: Twelve Bits

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* [11].

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× 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.

Chapter 9

Mechanical Ears

9.1 Facing Reality

It's time to face the real world. Thus far we've existed almost entirely in the space of software. We've been playing with signals - chopping them up and putting them back together in more useful ways - but we've yet to actually discuss how we're going to really get those signals from the real world. That then is what the next few chapters are about - the hardware.

When it comes to hardware there's really only two functions that we want. (1) A way to turn sound into a high resolution signal. (2) A way to turn that signal digital so a computer can consume and manipulate it. In other words all we want is a microphone attached to an analog to digital converter (an ADC). There of course will be more involved here in getting those two things to behave well and be able to communicate with our computer, but more or less that's all that's really involved here. This chapter then will be about the microphone side of things while the next will be about the ADC. But before we move onto learning about how mics work let's get a couple desired outcomes clarified that will help direct the choices of mic and ADC to talk about.

In my mind CROAC is its most powerful when it also its cheapest. Rather than setting up some super expensive sound equipment up near a pond, imagine if we could just build small, cheap, easily reproducible circuit boards that we could hang up next to any odd pond or vernal pool that we find? Then we'd be able to take measurements all across a particular region at really high resolution and thereby have information we could tie to all sorts of other data like elevation, temperature, noise and light levels, etc. It's scale that brings power and scale requires technology that is very cost effective, portable, and easy to use. So our goal here is not to buy the fanciest sound equipment possible and tie it together, but rather to be able to find relatively inexpensive parts that can be thrown together into a relatively small scale circuit that's easy to set up and not going to get in anyone's way.

So with that in mind, let's talk about MEMS mics (Micro-Electro-Mechanical-Systems).

9.2 Listening with Capacitance

There are lots of different kinds of mics out there, but the tiny ones you'll find inside things like your phone are MEMS mics. MEMS mics themselves belong to a breed of microphones called condenser mics. Condenser mics use capacitance to turn motion in the air into an electric signal so obviously before we can understand them properly we've got to talk a little about capacitance.

First of all, what is a capacitor? A capacitor (illustrated in Figure 1) is simply two conductive plates (with a specific area A) separated by a distance d. This gap is filled with something non conductive like a vacuum or some *dielectric* (air is a dielectric). When a voltage is applied across the capacitor electrons try to flow through the capacitor but can't due to the non-conductive gap. Instead they just accumulate on one side which sets up an electric field across the dielectric. That electric field pushes on like charges on the other plate of the capacitor so that both plates have equal but opposite charges. This process doesn't go on forever so eventually (for the circuit but quite quickly to us humans) a static charge Q is accomplished and the capacitor stops accumulating new charge.

Therefore, given a voltage V across the capacitor and a charge Q the capacitance C is defined as the ratio $C = Q/V$. It's units are in Farads.

Now it turns out that if you have a capacitor with a very small gap and relatively large plates (i.e. $A >> d$) that the capacitance is given by:

$$C = \frac{\epsilon A}{d} \tag{9.1}$$

where ϵ is the dielectric constant and more or less just quantifies the "non-conductiveness " of whatever is in the gap. Putting this back into our definition of capacitance and rearranging for voltage we have:

$$V = \frac{Qd}{\epsilon A} \tag{9.2}$$

Now suppose we have a capacitor where one of the plates is "flimsy", i.e. it can vibrate like a diaphragm. That would mean that while the area A and the dielectric constant e would remain the same d would be changing with the vibrations. Let's then differentiate our equation with respect to d:

$$\partial_d V = \frac{Q}{\epsilon A} + \frac{d}{\epsilon A}\partial_d Q \tag{9.3}$$

Now if we had some way of keeping Q constant this equation would reduce to:

$$\partial_d V = \frac{Q}{\epsilon A} \tag{9.4}$$

which means our voltage follows the changes in d linearly! This is perfect because it means that our vibrations (in this ideal case) have been turned from a mechanical signal into an electric signal which is exactly what we wanted.

Turns out this is (more or less) exactly how MEMS mics work [4]. Inside the tiny chip you receive when you purchase one is a capacitor composed of a solid plate on top with perforations that allow sound waves to pass through to the lower plate that can move and acts like a diaphragm. Then onboard the same chip is an ASIC (application specific integrated circuit) that acts as a charge pump to keep the Q constant. Then as sound waves hit the tiny little microphone the changes in d result in changes in V just as we described which can then be measured by a circuit which can convert that into a stronger (and eventually digital) signal. Pretty sweet!

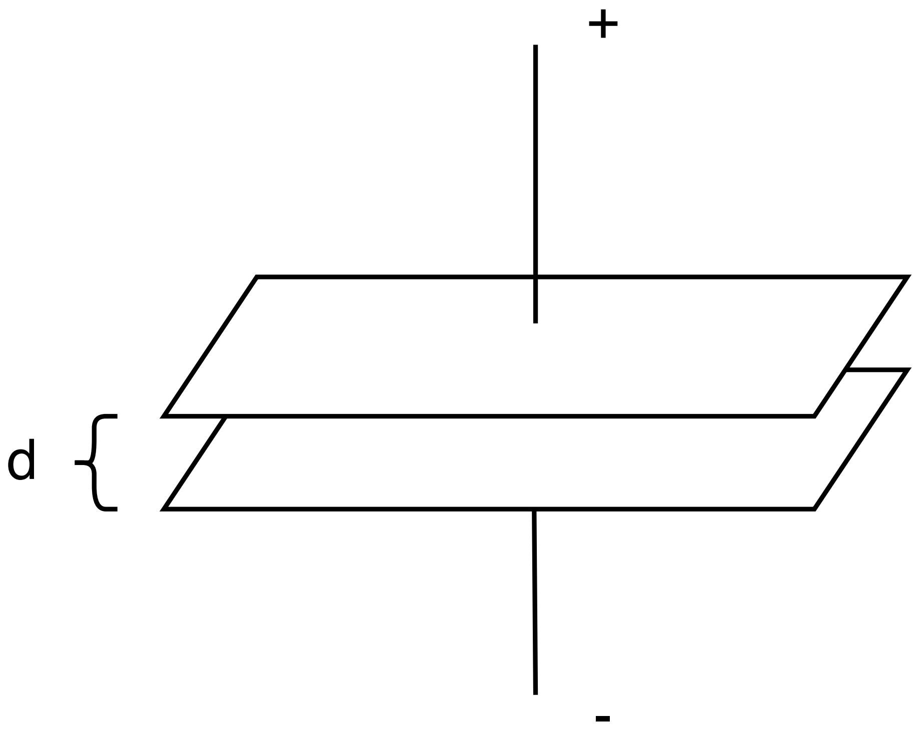

Figure 9.1: A Capacitor

9.3 Send Me the Deets

Now that we have a general understanding of how one of these mics work, let's talk about the characteristics that make mics different from one another - i.e. let's talk about specs.

9.3.1 Sensitivity

The first question that we have to answer about our mic is what kind of signal we can expect given a particular input audio signal. Remember we will be converting from sound pressure (measured in Pascals) to electric potential (measured in Volts). So a pretty key measurement is the how many Volts I can expect given a certain number of Pascals - this ratio is the *sensitivity*.

Because we're measuring pretty much everything in decibels and decibels is a relative quantity (it tells you ratios between a measurement and a reference) we need a reference in order to express the sensitivity. In addition to this, as we'll see soon, different frequencies behave slightly differently, so when we choose our reference point we must also choose a reference frequency. The typical choices are a 1 Volt, 1 Pascal reference at 1kHz. With this we can then define the sensitivity in dBV as [8]:

$$S_{dBV} = 20\log_{10}(V/P) \tag{9.5}$$

where V and P are the actual measured voltage and pressure. Note that because we chose a reference where $V/P = 1$ we don't have to include it in the $\log_{10}$.

Couple of additional details that are worth pointing out.

First there's the question of why we're multiplying our logarithm by 20 instead of 10. This is a matter of definitions. When dealing with power we use the formula $10\log_{10}$ but when dealing with voltage or signal amplitude we instead use $20\log_{10}$ and because right now we're talking about voltage, we use the 20. If you're wondering why 20 and not something else, it's because power is related to the square of the amplitude. In other words we have:

$$dB = 10\log_{10}(\frac{P}{P_{ref}}) = 10\log_{10}(\frac{A^2}{A_{ref}^2}) = 20\log_{10}(\frac{A}{A_{ref}}) \tag{9.6}$$

Second you'll often read the reference as something like 1kHz at 94 dB SPL. What does that mean? Well it actually means the same thing as our

1kHz at 1Pa it's just getting expressed in different units (note also the 1V reference voltage is just implied here) [3]. So how is 94 dB SPL the same thing? Well SPL is the Sound Pressure Level and when measured in dB it is measured in reference to the reference pressure in air which is ≈ 0.00002Pa. And $20\log_{10}(1Pa/0.00002Pa) \approx 94$dB so they're the same thing.

9.3.2 SNR

Alright, so we now have a sense for how many volts we'll get for a specific number of pascals which gives us our sensitivity. But as with any instrument there's going to be a certain level of noise introduced. So the next specification of interest is the Signal to Noise Ratio or the SNR. So remember how we had to switch our dB multiplier to 20 in the last section because we were dealing with volts? Well we're going back to power as SNR is defined as the ratio between the signal and noise power. So:

$$SNR_{dB} = 10\log_{10}(\frac{P_{signal}}{P_{noise}}) \tag{9.7}$$

9.3.3 Frequency Response

In order to get the cleanest signal possible we would like that every frequency is measured the same. You know how people hear different frequencies as being louder or quieter? Well it's the same thing for microphones. So one of the key properties of microphones is the frequency range for which you can expect a flat frequency response. A flat frequency response means that the magnitude of the signal (in power) stays within a specific range. The range that's typically chosen is up to -3dB from peak power (i.e. up to one half of peak power). So if your mic says it has a flat frequency response between 1kHz and 20kHz that means at 1 and 20kHz the power is half of peak power (where the peak is somewhere between those two ends). The frequency response therefore measures the effective range of listening that your mic has.

9.3.4 Electrical Properties

Sensitivity, SNR, and Frequency Response give us the acoustic properties of our mic, but what about it's electric properties? Well there's a couple things to pay attention to.

First is the input supply voltage and current, these are obviously important characteristics to look at when you're designing how your overall circuit should work. But equally important are the output characteristics - impedance and max output voltage.

Max output voltage is pretty easy to understand - it's just the most you'll get out of the mic in volts. But to understand why impedance is important requires us to talk about voltage dividers [5].

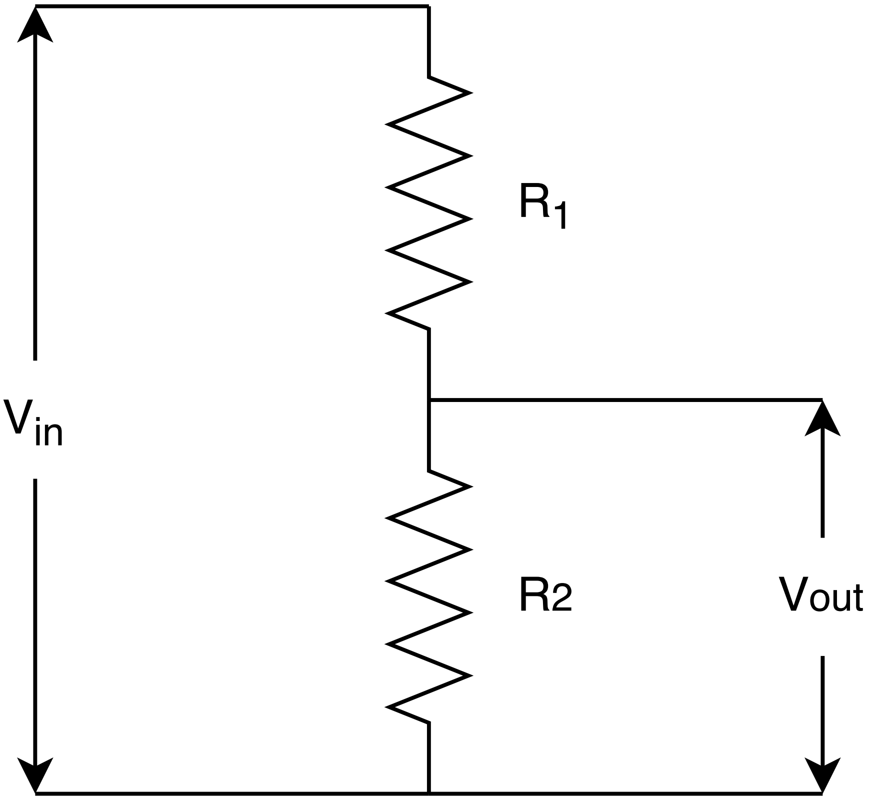

Figure 9.2: A Voltage Divider

Figure 2 shows us the simplest voltage divider you can construct. How does this work to "divide" voltage? Well let's run the math and find out.

From Ohm's law we know that:

$$V = IR \tag{9.8}$$

I is going to be the most useful quantity to keep track of because it should remain constant in this circuit so we can use it to relate V_{in} to V_{out}. For V_{in} we have:

$$I = \frac{V_{in}}{R_1 + R_2} \tag{9.9}$$

This is because the accumulative resistance of resistors in series is just the sum of their individual resistances.

For V_{out} we have:

$$V_{out} = IR_2 \tag{9.10}$$

and therefore:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in} \tag{9.11}$$

Therefore $V_{out} \leq V_{in}$ and we have ourselves a voltage divider!

Okay so what does this have to do with impedance? Well impedance is a generalized measure of "resistance" that can be applied to any kind of circuit and circuit component (not just resistors). So when we talk about output impedance we're effectively saying that our component is providing an R_1 in our voltage divider component above. Then when we attach a new circuit component after our mic, the input impedance of that component represents R_2. So that means the voltage across our new component is going to drop - and that means our signal is going to drop!

We can minimize my this by making R_2 far larger than R_1. I.e. we want the output impedance from our mic to be as low as possible and the input impedance from our next component to be quite high. This is why input impedance is such an important quantity to keep track of.

9.3.5 Summary of the Deets

Okay let's summarize the important specs:

1. **Sensitivity** - The ratio of the output signal in Volts to the input signal in Pascals (usually expressed in dB).

2. **SNR** - The ratio of the signal power to the noise power (usually expressed in dB).

3. **Frequency Response** - Usually, the range of frequencies for which we're within half of maximum power.
4. **Supply Voltage and Current** - The supply specs required to run the mic.
5. **Max Output Voltage** - How large the output signal can get.
6. **Output Impedance** - The ”generalized resistance” the mic is adding to the circuit (relevant for protecting against voltage drops in the next stage of the circuit).

9.4 Hearing Voices

So we've got the intuition of how a MEMS mic works down. We've got the parameters to pay attention to that distinguish one mic from the next. But how on earth do we put this into a circuit? To understand this we've got to introduce two concepts - a low pass filter and an AC filter. Let's start with the first.

9.4.1 Let's Not Listen to the Power Supply

So the last thing we want is signals coming from anything besides our mics. But here's the problem, there's nothing to actually prevent some amount of alternating current (AC) coming from our power supply. So how do we make sure none of this reaches our mic? Well we can add what's called an AC filter. The idea is wonderfully simple.

Consider Figure 3. We have voltage in on the left, then a capacitor that connects to ground followed by a resistor connected to ground. Now electrons follow a law very similar to us human beings - they will follow the path of least resistance. In other words if the resistance of the capacitor is somehow lower than the resistance of the resistor (we're using the term resistance loosely here) the electrons will choose to move through the capacitor instead of the resistor.

Now remember from our discussion of what a capacitor is that charge cannot actually flow through it - instead it just sort of accumulates on the edges of the capacitor. So wouldn't that mean we've got infinite resistance?

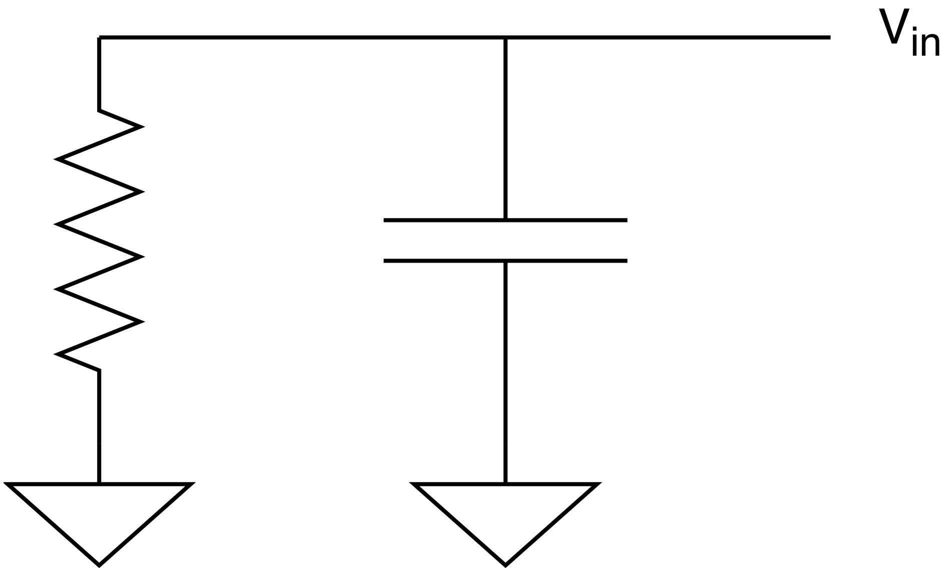

Figure 9.3: An AC Filter

For a DC current this is definitely true. Therefore all of our DC signal will "choose" the resistor over the capacitor.

For AC however, the story is more complicated. For an AC current the voltage keeps switching back and forth. This means the "desired" charge on the capacitor keeps switching back and forth. More mathematically if we remember the equation from earlier:

$$C = Q/V \tag{9.12}$$

then differentiating both sides with respect to time and rearranging a little gives us:

$$\partial_t Q = C\partial_t V \tag{9.13}$$

But the derivative of charge is just the current. So we have:

$$I = C\partial_t V \tag{9.14}$$

This means that so long as the voltage is changing, current *can* pass through a capacitor. So while DC current sees a capacitor as infinitely resistave wall, AC current just sees it as another resistave element it can pass through. And if that resistance is less than the resistance of the resistor it'll go through the capacitor instead. Which now means we've passed the AC and DC parts of our signal to two different elements!

Now the resistor could be replaced by just about any other resistave element - including the rest of our circuit - and so long as it's resistance is higher than the "resistance" of our capacitor (once again speaking very loosely here), the DC will pass to our circuit and the AC will effectively "short" through the capacitor. Pretty cool!

9.4.2 No DC Please

Alright, so we've passed DC current into the mic and now we're getting an AC current out the other end - this of course is our signal. Problem is the mic will also have likely introduced some base DC into our signal as well and we'd rather like to get rid of this. Now given the AC filter we just described it may seem as if the obvious solution is to have our current pass through a capacitor before heading off to other components as we already saw how DC simply can't pass through a capacitor. But life is rarely so simple.

When Your Filter Gets High

To get a sense of the issue that can arise here we've got to talk about a high pass filter. So time for another diagram!

Here in Figure 4 we have a capacitor and resistor in series and we're measuring the voltage across our resistor. Now to do this math right we're going to have to stop talking loosely about impedance and resistance and talk about them precisely instead. So prepare yourself for some cool math!

If we represent our alternating voltage as a complex number like we did for our sound waves earlier it turns out that we can represent all three basic circuit elements (resistors, capacitors, and inductors) by the same thing - a complex number call impedance which we'll denote by a Z. Z is the sum of two numbers, the resistance R and the reactance X:

$$Z = R + iX \tag{9.15}$$

What's so powerful about this representation is that it turns out Ohm's law just follows for impedance in general, regardless of whether it's a resistor or a capacitor or some strange combination of both. Along with that the

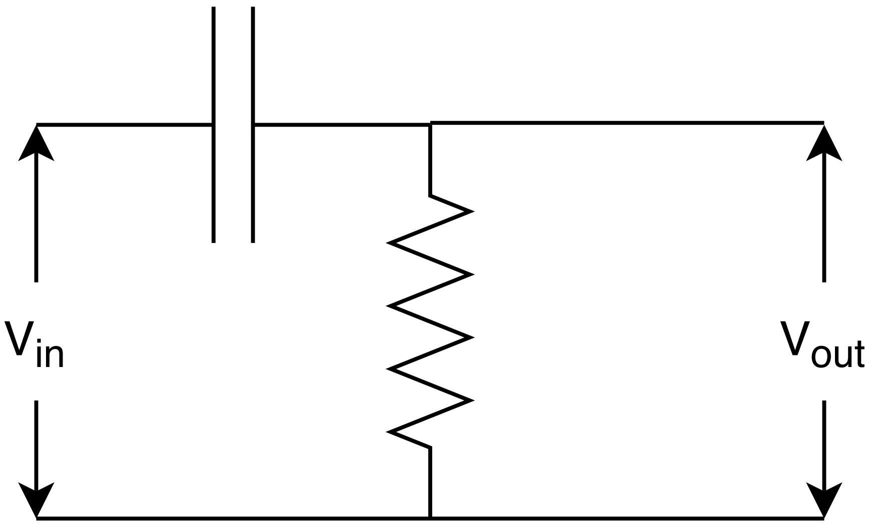

Figure 9.4: An High Pass Filter

rules for computing the result of a combination of impedances is the same as that for resistors. In series:

$$Z = Z_1 + Z_2 + ... + Z_n \tag{9.16}$$

and in parallel

$$Z = \frac{1}{1/Z_1 + 1/Z_2 + ... + 1/Z_n} \tag{9.17}$$

Pretty awesome generalization in my opinion!

Okay so how on earth does this help us with understanding our high pass filter? Well remember our voltage divider was just two resistors in parallel and this new circuit is just two "impeders" in parallel. So we've simply got a more complicated voltage divider (I don't know about you but I love when things that are different suddenly become the same) [5]. Therefore we have:

$$V_{out} = \frac{Z_2}{Z_1 + Z_2} V_{in} \tag{9.18}$$

We know that $Z_2 = R$ because that's just our resistor. So all we need to know is the impedance of the capacitor. Well it turns out the impedance of a capacitor is given by:

$$Z = \frac{-i}{\omega C} \tag{9.19}$$

where C is our capacitance. First off we can see that obviously a capacitor only has reactance (there's no resistance which is no surprise). But what is with the ω in the denominator? That's the frequency of the alternating voltage. So as the frequency increases the magnitude of the reactance of the capacitor decreases. Perhaps you're beginning to guess how this all ends up being a high pass filter, but let's follow the math through.

Plugging what we know into our equation relating our input and output voltage we have:

$$V_{out} = \frac{R}{R - i/\omega C} V_{in} \tag{9.20}$$

Now as before with our waves we can get the power with:

$$P_{out} = V_{out} V_{out}^{*} = \frac{R^2}{R^2 + 1/\omega^2 C^2} P_{in} \tag{9.21}$$

Let's look at extremes. If ω is really really large then $P_{out} \approx P_{in}$. But as ω approaches zero P_{out} goes to zero as well. I.e. high frequencies are passed and low frequencies are filtered out - a high pass filter.

While the degree to which the power is attenuated is obviously a continuous function we can still define a "line in the sand" by looking for the point at which the power is cut in half (the 3dB cutoff we've seen before). Looking at our equation this is simply where:

$$\omega = 1/RC \tag{9.22}$$

Back to DC

Okay so let's return to our circuit where we were trying to filter out the DC by placing a capacitor right after the mic. Well remember after the capacitor is going to come the rest of the circuit (including our ADC) which will have some level of impedance. The rest of our circuit is pretty much equivalent to the resistor in our high pass filter and so we've got a capacitor and resistor in series - i.e. a high pass filter. So if we set the capacitance too low and the impedance from the rest of our circuit is also very low our fraction $1/RC$ will be a very large number and we'll start filtering out our mic's signal! So we've got to make sure that not only do we put a capacitor there to filter out the DC signal but that the capacitance is high enough to not also filter out a bunch of the mic signal we're wanting to process.

Alright, so in summary we want to make sure we have an AC filter before the mic to remove any signal from our power supply while also adding a high pass filter (of sufficient capacitance) to filter the DC out of our mic output. But with our sound signal now turned into a lovely analog voltage signal it's time to understand how we're going to make that signal digital. Time to move onto ADCs.

Chapter 10

Digitize Me

10.1 It's Complicated

Before we can even begin talking about Analog to Digital Converters (ADCs) we need to address a very specific question - what does a digital signal look like? We've spoken before a digital signal in terms of how it quantizes our input, but we haven't really talked about what a digital signal looks like as it's flying our circuit.

All digital signals are represented, at their core, by two logic states - HIGH and LOW. This plants our feet firmly in the world of binary numbers. To get a sense of what binary means we start with our own number representation which is in *base* 10. What does base 10 mean? It means I'm writing my number as a series of powers of ten. So for example if I wanted to write 93, I'd start with the highest power of 10 that's in this number - in this case 10^1. Then I'd determine how many multiples of 10^1 I have - 9 - and what the remainder is - 3. Then I'd drop to the next power of 10 - 10^0 - and repeat. This would then give me:

$$93 = 9 \bullet 10^1 + 3 \bullet 10^0 \tag{10.1}$$

and that gives us the written 93 because we start with the highest powers first and make our way down. Now if that seems a little weird or circular to you, don't worry, let's see what this looks like when we turn to binary.

Binary, as we just mentioned, has two states - HIGH and LOW - and is therefore base 2. So let's follow the exact same process to write 93 in base 2. The highest power of 2 present in 93 is $2^6 = 64$ and we obviously only have one multiple of this so the first digit in our base 2 number is going to be 1 and our remainder will be $93 - 64 = 29$. The next power of 2 is $2^5 = 32$ and there's obviously zero of these in our remainder, so our next digit is zero. Repeating this process until we get all the way down to 2^0 gives us:

$$93 = 1 \bullet 2^6 + 0 \bullet 2^5 + 1 \bullet 2^4 + 1 \bullet 2^3 + 1 \bullet 2^2 + 0 \bullet 2^1 + 1 \bullet 2^0 \tag{10.2}$$

which gives us the written representation of 1011101, a binary number! Note that this is the same thing as 93 just written in base 2 instead of base 10.

Okay so why do all this? Because now we can represent all of our numbers in terms of HIGH and LOW states alone. And why is that useful? Because it means our circuit only needs to be able to read a lot of voltage vs very little voltage and that makes it pretty much noiseless. Remember if we are reading the *precise* voltage coming out of a circuit to get a signal noise is just going to throw us off. But if we are only wondering whether we're getting some voltage above a value, or a voltage below our signal can be pretty noisy without throwing us off. This means digital circuits can get really really complicated or communicate over large distances without introducing all sorts of errors which is exceptionally useful. So you can now think of a digital signal as really just a series of HIGH and LOW pulses traveling through our circuit.

At this point there are probably two questions floating around in your brain (they certainly were in mine when I first learned about this stuff). How on earth would we tell two consecutive HIGH or LOW states apart and more importantly how do I figure out where my binary number begins and ends? I.e. how does the receiver of this digital signal stay in sync with the sender?

The answer to that is twofold. First when it comes to telling apart successive bits a clock is involved. A series of consecutive HIGH states looks like just one single high state until a clock defines a cycle rate at which each bit measurement is taken. So now with each clock cycle we can take a measurement and even if the state remains HIGH we know it's a new state and therefore a new bit.

On the other hand when it comes to figuring out where my numbers begin and end that comes down to defining a communication *protocol*. Basically the sender and receiver have to have an agreed upon method of communicating which will include things like how to mark a message as starting and ending, how long those messages will be, how to interpret them, and so forth. There's a lot of different protocols that are out there so I'm not going to cover them here (at least not until we choose one to use) but suffice it to say our ADC is going to have, at it's output, a little microprocessor that's going to handle communicating with a particular protocol and then we need to make sure our computer is on the same page with that microprocessor so that it can interpret the stream of HIGH's and LOW's appropriately. It obviously also needs to be in sync with the clock on that microprocessor as well.

Okay so to summarize our ADC is going to take the following steps:

1. Receive the analog signal
2. Quantize the analog signal
3. Convert that analog signal to a binary representation
4. Store that binary representation somewhere that's accessible to the sender of the digital signal
5. Communicate that signal out using a specific protocol tied to a specific clock cycle

Alright with that general orientation out of the way let's dive in!

10.2 One Step at a Time

There are several kinds of ADC architectures out there. We're going to be focusing one called a Delta-Sigma ADC which is notable for its super high resolution (you can easily get ones that are 24 bits or more) while also being able to sample at rates much higher than the limits of human hearing - so perfect for our application!

Figure 1 shows us the general components of a $\Delta\Sigma$ ADC. We start with a $\Delta\Sigma$ whose job is to over sample the input analog signal in such a way as to push the quantization noise (more on that in a moment) to higher than desired frequencies. The modulator's output is pulses of HIGH/LOW signals whose average value is the level of the input analog signal. The digital filter then does this averaging to get a quantized representation of the input signal, but it does this using digital logic so we get a nice binary representation. After that the decimator removes our over sampling and

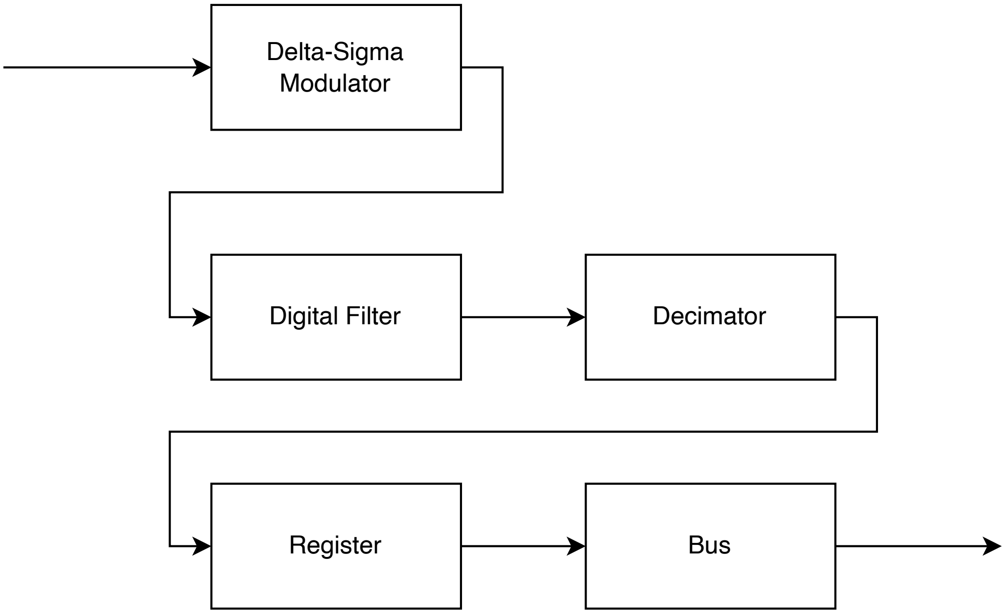

Figure 10.1: Delta Sigma ADC Components

drives us back to the desired frequencies. Then the register takes and stores the current reading while the bus communicates what's been measured to some receiver elsewhere.

The latter two stages (register and bus) are handled differently in different ADCs, but their purpose is always the same - communicate the findings out to the rest of world. So we're going to focus on what makes an $\Delta\Sigma$ ADC special - the modulator and the filtering.

10.3 Too Much *is* a Good Thing

We start with the $\Delta\Sigma$ Modulator.

Figure 2 shows how a $\Delta\Sigma$ Modulator is constructed. Let's start by walking through the components. The difference amplifier takes in two inputs and outputs their difference - no surprises there. The integrator sums the inputs being passed into it. The comparator can be thought of as a 1 bit ADC. If the voltage being passed in is higher than the reference it will output a HIGH signal and a LOW signal otherwise. Finally our 1 bit DAC just converts our 1 bit back to analog. Alright let's walk through how this thing

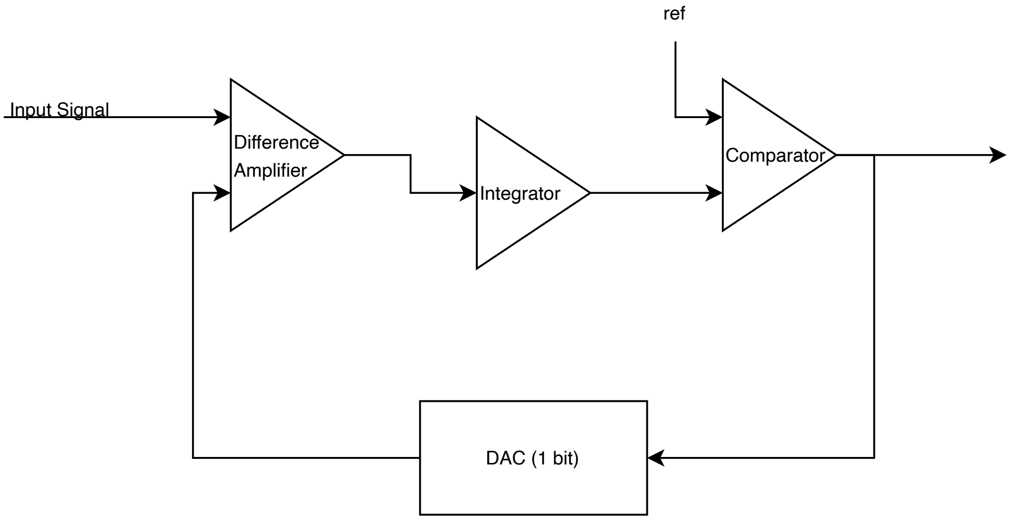

Figure 10.2: Delta Sigma Modulator

works by following an example.

Suppose our input voltage is 0V and our DAC output starts at +5V. Then our difference is -5V. Our integrator works by summing this -5V over time so after a short period our comparator is going to see a voltage of -5V itself which is going to cause it to output the most negative voltage it can thus throwing our DAC into a LOW state (let's say -5V). Our difference is therefore now 5V instead of -5V and so the integrator will start summing up to 5V which will cause the comparator to throw itself to a positive voltage and DAC back into a high state. This will just keep happening and so we'll have a pulse of LOWs and HIGHs where we spend as much time in the HIGH as the LOW state. In other words the average value of our pulse is precisely 0V.

Now suppose our input voltage is 3V. The same process is going to ensue except that this time when the DAC is in a +5V state our difference will be -2V and when our DAC is at a -5V state our difference will be 8V. This means that when our DAC is at a positive value our integrator will take longer to flip it the other way than in our prior example and that when our DAC is in a negative state it will take our integrator less time to flip it than in the prior example. Which means that the average of our pulse will be > 0. In fact it will be 3V!.

Okay so cool we've got a pulse modulation of our original analog input - so what? Here's where the magic starts. As I mentioned earlier the $\Delta\Sigma$ samples at a rate higher than our overall ADC. I.e. it oversamples. Now it turns out that the quantization error gets smeared across the whole range of possible frequencies in our quantized signal. So by oversampling we extend the range of frequencies beyond the actual ones we'll measure at the end of the day thereby pushing a whole slew of our quantization error into frequency bands we'll later delete. And when we remove those frequencies we'll remove the error too! This allows a $\Delta\Sigma$ ADC to produce far less noise than equivalent ADC architectures.

Let's move onto the digital filter and decimator.

10.4 Magic Filters

Okay so we've now got this pulse modulated (i.e. digital) signal flying into our digital filter. We know the average value of that pulse is equal to our input signal (plus some error) so we just average and move on, right? Not quite. Suppose our over sampling rate was 16x - that is we have 16 samples to average per output sample - then we'd have 16 1's and 0's to average. But this only allows for 16 different possible values - which is a 4 bit signal. To get a 24 bit signal that'd mean we'd need an over sampling rate (OSR) of over 16 million! That would take our 20MHz limit of human hearing to 334 Tera-Hertz! So how on earth does the digital filter create a 24 bit signal? By taking a running weighted average over far more than just the bits from our OSR. This then allows for far, far more values and thereby a 24 bit signal.

But, because it's a running average we still have a signal coming through at the same rate as our OSR. This is where the decimation comes in. The decimation simply throws out most of our samples leaving us with a single sample per output sample cycle. And in so doing it thrusts us back into our normal frequency range while saying goodbye to all the quantization error we created during our oversampling. Honestly magic.

Alright all of that was pretty "hand-wavy" and imprecise but I think it gets across the salient points. If you're interested in learning more check out the sources I learned all this from [15] and [6].

10.5 Devils in the Details

Now that we understand the magic it's time to go over the specs to pay attention to.

1. **Input Voltage Range** - this is the accepted voltage range for the input as extremely important as this is also the range that is being transferred into bits. So for example if your range is ±5V and you know for a fact that your input will be in the range of ±2.5V then you're only going to be using half of your bit resolution!
2. **Impedance** - this is important for all the voltage splitting reasons we talked about in the prior chapter.
3. **Data Rate** - this is super important as it's going to give you the maximum sampling rate the ADC can give you. As we spoke about in the chapter on Discrete Fourier Transforms having a high enough sample rate is exceptionally important.
4. **Noise** - not much to say here - less noise is obviously better.
5. **Resolution** - this gives you the resolution in bits you can expect from the ADC.
6. **Logic Family** - this determines how HIGH and LOW are specified by the ADC.
7. **Data Format** - the specific binary representation family.
8. **Communication Protocol** - got to make sure you're all speaking the same language.

Note that when it comes to integrating one of these things into your overall circuit you really have to pay attention to the specifics of that *particular* ADC. How is it expecting its inputs? How do you communicate with the bus? What specific voltage references is it going to need, or is that build in? Lots and lots of questions - so make sure you read the data sheet thoroughly.

As a final note, I want to just stress that your ADC's resolution is going to be applied over the full range of accepted input voltage. So if your mic's output is going to far below that full range use an amplifier to make sure you don't waste your bit resolution.

Bibliography

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

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

[3] Robert Dunn. What is sound pressure level (spl) and how is it measured?

[4] Arthur Fox. What is a mems microphone?

[5] Paul Horowitz and Winfield Hill. *The Art of Electronics*. 2015.

[6] Texas Instruments. Sar and delta-sigma: Basic operation.

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

[8] Jerad Lewis. Understanding microphone sensitivity.

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

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

[11] Wikipedia. Discrete fourier transform.

[12] Wikipedia. Near and far field.

[13] Wikipedia. Piano key frequencies.

[14] Wikipedia. Sound localization.

[15] Fernando Zigunov. Delta-sigma modulator basics.