[AUDIO PLAYBACK]

- Good morning, class.

[END PLAYBACK]

MICHALE FEE: Hey, let's go ahead and get started. So we're going to finish spectral analysis today. So we are going to learn how to make a graphical representation like this of the spectral and temporal structure of time series, or in this case, a speech signal recorded on a microphone. Well, actually let me just tell you exactly what it is that we're looking at here.

So this is a spectrogram that displays the amount of power in this signal as a function of time and as a function of frequency. So you remember we've been learning how to construct the spectrum of a signal. And today, we're going to learn how to construct a representation like this, called a spectrogram, that shows how that spectrum varies over time.

So as you recall, we have learned how to compute the Fourier transform of a signal. This is one of the signals that we actually started with. So if you compute the Fourier transform of this square wave, you can see that in the frequency domain, now we plot the amount of, essentially, the components of this signal at different frequencies. So the Fourier transform of this square wave has a number of peaks. Each of these peaks correspond to a cosine contribution to this time series, OK?

All right. And we also discussed how you can compute the power spectrum of a signal from the Fourier transform. So here what I've done is I've taken the Fourier transform of this square wave. And now we take the square magnitude of each of these values, and we just plot the spectrum, the square magnitude of just the positive frequency components.

For real-valued functions, the power spectrum is symmetric. The power in each of these-- at each of these frequencies in the positive, half of the-- for the positive frequencies is exactly the same as the power in the negative frequencies. So if we plot the power spectrum of that square wave, we can see that there are multiple peaks at regular intervals.

Now the problem with plotting power spectra on a linear scale here is that you often have contributions-- important contributions to signals that actually have a very small amount of power when you plot them on a linear scale. And so you can barely see them. You can barely see those contributions at these frequencies here on a linear scale. But if you plot this on a log scale, you can see the spectrum much more easily.

So for example, what we've done here is we've plotted the square magnitude of the Fourier transform, taken the log base 10 of that spectrum-- spectrum is the square magnitude of the Fourier transform. And now we can take the log base 10 of the power spectrum to get the power in units of bels, and multiply that by 10 to get the power in units of decibels, OK?

So each tick mark here of size 10 corresponds to one order of magnitude in power, OK? So this peak here is about 10 dB lower than that peak there, and that corresponds to about a factor of 10 lower in power. OK, any questions about that? I want to be able-- want you to understand what these units of decibels are. They're going to be on the test.

OK. Questions about that? You want to just ask me right now? OK. Remember this. OK. And keep in mind that the power in a signal is proportional to the square of the amplitude, OK?

So if I tell you that a signal has 10 times as much amplitude, it's going to have 100 times as much power. 100 times as much power is 10 to the 2 bels, which is 20 decibels. Does that make sense? OK.

All right. So we also talked about some Fourier transforms of different kinds of functions. So this is the Fourier transform of a square pulse. So here I showed you a square pulse that has a width of 100 milliseconds. The Fourier transform is this sinc function, and for a square pulse of width 100 milliseconds, the sinc function has a half-- sorry-- has a full width at half height of 12 hertz.

If we have a square pulse that's five times as long, 500 milliseconds long, the Fourier transform is the sinc function again, but it has a width of this central lobe here of 2.4 hertz. So you can see that the longer the pulse, the shorter-- the narrower the structure in the frequency domain. Up here, if we'd look at the Fourier transform of a square pulse that's 25 milliseconds long, then the Fourier transform is again a sinc function, and the width of that central lobe is 48 hertz.

So you can see that the width in the time domain and the width in the frequency domain are inversely related. So the product of the width in the time domain and the width in the frequency domain is constant. And that constant is called the time-bandwidth product of that signal.

The time-bandwidth product of this square pulse and sinc function is constant. It's independent of the width. The time-bandwidth product is a function of that. It's a characteristic of that functional form.

OK. Now we also talked about the convolution theorem, which relates the way signals get multiplied in time or convolved in frequency domain. So for example, if we have a square pulse in time multiplied by a cosine function in time to get a windowed cosine-- so this function is zero everywhere except it's cosine within this window-- we can compute the Fourier transform of this windowed cosine function by convolving the Fourier transform of the square pulse with the Fourier transform of the cosine function, like this.

So the Fourier transform of the square pulse is, again, this sinc function. The Fourier transform of the cosine are these two delta functions. Now if we convolve the sinc function with those two delta functions, we get a copy of that sinc function at the location of each of those delta functions.

And that is the Fourier transform, OK? Any questions about that? Just a quick review of things we've been talking about. All right.

So we can look at this Fourier transform here. We can look at the power spectrum of this windowed cosine function, like this. So there's the windowed cosine function. The power spectrum-- the power spectrum plotted on a linear scale is just the square magnitude of what I've plotted here. And we're just going to plot the positive frequencies.

That's what the power spectrum of that signal looks like on a log scale. So you can see that it has a peak at 20 hertz, which was the frequency of the cosine function. You see some little wiggles out here. But if you look on a log scale, you can see that those wiggles off to the side are actually quite significant.

The first side lobe there has a power that's about 1/10 of the central peak. That may not matter, sometimes, when you're looking at the spectrum of a signal, but sometimes it will matter because those side lobes there will interfere. They'll mask the spectrum of other components of this signal that you may be interested in.

We also talked about how this spectrum depends on the function that you multiply your cosine by here. So for example, if you take a cosine and you multiply it by Gaussian, the power spectrum has the shape of a Gaussian, it's a Gaussian that has a peak at 20 hertz. And if you look at the-- if you look at that spectrum on a log scale, you can see that it loses it-- you've lost all of these high-frequency wiggles, up here.

All of those wiggles come from the sharp edge of this square pulse windowing function, OK? So the shape of the spectrum that you get depends a lot on how you window the function that you're looking at. Questions about that? Right, again, more review.

OK. So we talked about estimating the spectrum of a signal. If you have many different measurements of some signal, you can actually just compute the spectrum of each one. This little hat here means an estimate of the spectrum. You compute some estimate of the spectrum of each of those trials, samples of your data, and you can just average of those together.

OK, now if you have a continuous signal, you can also-- you could estimate the spectrum just by taking the Fourier transform of a long recording of your signal. But it's much better to break your signal into small pieces, compute a spectral estimate of each one of those small pieces and average those together. Now how do you construct a small sample of a signal? If you have a continuous signal, how do you take a small sample of it?

Well, you can think about that as taking your continuous signal and multiplying it by a square window. Setting everything outside that window to zero and just keeping the part that's in that window. And you know that when you take a signal and you multiply it by a square window, what have you done? You've convolved the spectrum of this original signal with the spectrum of this square pulse.

And that spectrum of the square pulse is really a nasty looking thing, right? It is this what we call the Dirichlet kernel, which is just the power spectrum of a square pulse that we just talked about, OK? So that's called the Dirichlet kernel. And using a square pulse to select out a sample of data introduces two errors into your spectral estimate, narrowband bias. It broadens your estimate of the spectrum of, let's say, sinusoidal or periodic components in your signal. And it also introduces these side lobes.

So the way we solve that problem is we break our signal into little pieces, multiply each of those little pieces by a smoother windowing function by something that isn't a square pulse, multiply it by something that maybe looks like a Gaussian, or a-- or half of a cosine function. That gives us what we call tapered segments of our data. We can estimate the spectrum of those tapered pieces and averaged those together, OK? Any questions? OK, again, that's a review.

And I showed you briefly what happens if we take a little piece of signal. The blue is white noise with a little bit of this periodic sine function added to it. And if you run that analysis, you can see that there is a large component of the spectrum that's due to the white noise. That's this broadband component here. And that sinusoidal component there gives you this peak in the spectrum, OK? And there's a-- I've posted-- or Daniel's posted a function called wspec.m that implements this spectral estimate like this.

So now today, we're going to turn to estimating time varying signals, estimating the spectrum of time varying signals. So this is a microphone recording of a speech signal. Let me see if I can play that.

[AUDIO PLAYBACK]

- Hello.

[END PLAYBACK]

MICHALE FEE: All right. So that's just me saying hello in a robotic voice. OK. So that is the signal.

That's basically voltage recorded on the output of a microphone. It's got some interesting structure in it, right?

So first these little pulses here, you see this kind of periodic pulse. Those are called glottal pulses. Does anyone know what those are? What produces those? No?

OK, so when you speak a voiced sound, your vocal cords are vibrating. You have two pieces of flexible tissue that are close to each other in your trachea. As air flows up through your trachea, the air pressure builds up and pushes the glottal folds apart.

Air begins to flow rapidly through that open space. At high velocities, the velocity flowing through the constriction is higher than the velocity of air anywhere else in the trachea because it's flowing through a tiny little space. At high velocities, at constrictions where you have a high fluid flow, the pressure actually drops. And that pulls the vocal folds back together again.

When they snap together, all the airflow stops, and you have a pulse of negative pressure above the glottis, right? Imagine you have airflow coming up. And all of a sudden, you pinch it off. There's a sudden drop in the pressure as that mass of air keeps flowing up, but there's nothing more coming up below.

So you get a sharp drop in the pressure. Then the air pressure builds up again. The glottal folds open, velocity increases, and they snap shut again. And so that's what happens as you're talking, OK?

And so that periodic signal right there, those pulses in pressure-- the microphone is recording pressure, remember. So those pulses are due to your glottis snapping shut each time it closes during the cycle, during that oscillatory cycle. The period of that-- those glottal pulses is about 10 milliseconds in men and about 5 milliseconds in women.

OK. But you can see there's a lot of other structure changes in this signal that go on through time. But let's start by just looking at the spectrum of that whole signal. Now what might we expect? So if you have periodic pulses at 10 millisecond period, what should the spectrum look like? If you have a train of pulses, let's say delta functions with 10 millisecond period, what would the spectrum of that look like? Anybody remember what the spectrum of a train of pulses looks like?

Almost, yes. There would be. But there would be other things as well. What would a signal look like that just has a peak at 100 hertz? What is that? Has one peak at 100 hertz? Or let's say [INAUDIBLE] Fourier transform would have a peak at 100 and at minus 100.

That's just a cosine. That's not a train of pulses. What's the Fourier transform of a train of pulses? Those of you who are concentrating on this right now are going to be really glad on the midterm. What's the Fourier transform of a train of pulses? OK, let's go back to here because there was a bit of a hint here at the beginning of lecture.

What's the Fourier transform of a square wave? Any idea what happens if we make these pulses narrower and narrower? The pulses get more and more narrow, these peaks get bigger and bigger. And as we go to a train of delta functions, you just get Fourier transform of a train of delta functions in time, is just a train of delta functions in frequency. The spacing between the peaks in frequency is just 1 over the spacing between the peaks in time, right? Make sure you know that.

OK, so now let's go back to our speech signal. These are almost like delta functions. Maybe not quite, but for now, let's pretend they are. If those are a train of delta functions spaced at 10 milliseconds, what is our spectrum going to look like? I just said it. What is it going to look like?

Yep. Spaced by?

AUDIENCE: One.

MICHALE FEE: Which is? 100 hertz. Good. So here's the spectrum of that speech signal. What do you see? You see a train of delta functions separated by about 100 hertz, right? That's a kilohertz, that's 500 hertz, that's 100 hertz. So you get a train of delta functions separated by 100 hertz, OK? That's called a harmonic stack.

OK. And the spectrum of a speech signal has a harmonic stack because the signal has these short little pulses of pressure in them. OK, what are these bumps here? Why is there a bump here, a bump here, and a bump here? Does anyone know that?

What is it that shapes the sound as you speak? That makes an "ooh" sound different from an "ahh?" [INAUDIBLE] This is hello. Sorry, I'm having trouble with my pointer. That's hello. What is it that makes all things sound different?

So the sound, those pulses, are made down hearing your vocal tract. As those pulses propagate up from your glottis to your lips, they [AUDIO OUT] filter, which is your mouth. And that the shape of that filter is controlled by the closure of your lips, by where your tongue is, where different parts of your tongue are closing the opening in your mouth.

And all of those things produce filters that have peaks. And the vocal filter has three main peaks that move around as you move the shape of your mouth. And those are called formants, OK?

OK. Now you can see that this temporal structure, this spectral structure isn't constant in time. It changes-- right-- throughout this word. So what we can do is we can take that signal, and we can compute the spectrum of little parts of it, OK? So we can take that signal and multiply it by a window here, a taper here, and get a little sample of the speech signal and calculate the spectrum of it just by Fourier transforming, OK?

We can do the same thing. Shift it over a little bit and compute the spectrum of that signal, all right? So we're going to take a little piece of the signal that has width in time, capital T-- OK-- that's the width of the window. We're going to multiply it by a taper, compute the spectrum. And we're going to shift that window by a smaller amount, delta t, so that you have overlapping windows.

Compute the spectrum of each one, and then stack all of those up next to each other So now you've got a spectrum that's a function of time and frequency, OK? So each column is the spectrum of one little piece of the sound at one moment in time. Does that make sense? OK. And that's where this spectrogram comes from.

Here in this spectrogram, you can see these horizontal striations are the harmonics stack produced by the glottal pulse. This is a really key way that people study the mechanisms of sound production and speech, and animals vocalizations, and all kinds of signals, more generally, OK? All right, any questions about that?

All right. Now what's really cool is that you can actually focus on different things in a signal, OK? So for example, if I compute the spectrogram with signals where that little window that I'm choosing is really long, then I have high frequency-- high resolution and frequency, and the spectrogram looks like this.

But if I compute the spectrograph with little windows in time that are very short, then my frequency resolution is very poor, but the temporal resolution is very high. And now you can see the spectrum. You can see these vertical striations. Those vertical striations correspond to pulse of the glottal pulse. And we can basically see the spectrum of each pulse coming through the vocal tract.

Pretty cool, right? So how you compute the spectrum depends on whether you're actually interested in. If you want to focus on the glottal pulses, for example, the pitch of the speech you look with a longtime window. If you want to focus on the formants, here you can see the performance very nicely, you would use shorttime window. Any questions?

So now I'm going to talk more about the kinds of tapers that you use to get the best possible spectral estimate. So a perfect taper, in a sense would give you perfect temporal resolution. It would give you really fine temporal resolution. And it would give you really fine frequency resolution, but because there is a fundamental limit on the time bandwidth product, you can't measure frequency infinitely well with an infinitely short sample of a signal.

Imagine you have a sine wave, and you took like two samples of a sine wave. It would be really hard to figure out the frequency, whereas if you have many, many, many samples of a sine wave, you can figure out the frequency. So there's a fundamental limit there. So there's no such thing as a perfect taper.

If I want to take a sample of my signal in time, if I have a sample that's limited in time, if it goes from one time to another time and a zero outside of that, then in frequency, it's spread out to infinity. And so all we can do is choose how it is. We can either have things look worse in time and better in frequency or better in time and worse in frequency.

So the other problem is that when we taper a signal, we're throwing away data here at the edges. But if you take a square window and you keep all the data within that square window, well, you've got all the data in that window. But as soon as you taper it, you're throwing away stuff at the edges. So you taper it to make it smooth and improve the spectrum, the spectral estimate, but you're throwing away data.

So you can actually compute the optimal taper. Here's how you do that. What we're going to do is we're going to think of this as what's called the spectral concentration problem. We're going to find a function W.

This is a tapering that is limited in time from some minus T/2 to plus T/2. So it's 0 outside of that. It concentrates the maximum amount of energy in it's Fourier Transform, in its power spectrum within a window that has widths 2W. So W is this [INAUDIBLE]. Does that makes sense?

We're going to find a function w that concentrates as much energy as possible in square window. And of course, that's going to have the result that the energy in the side lobes is going to be as small as possible. And there are many different optimizations you can do in principle. But this particular optimization is about getting as much of the power as possible into a central low.

Here's this function of time. We simply calculate the Fourier Transform of W. We call that U of f. And now we just write down a parameter that says how much of that Fourier-- how much of the power in U is in the window from minus w to w compared to how much power there is in U overall, overall frequencies?

So if lambda is 1, then all of the power is between minus w and w. Does that make sense? So you can actually solve this optimization problem, maximize lambda, and what you find is that there's not just one function that gives very good concentration of the power into this band. There's actually a family of functions.

There's actually k of these functions, where k is twice the bandwidth times the duration of the window minus 1. So there are a family of k functions called Slepian functions for which lambda is very close to 1. There are also discrete probate spheroid sequence functions, dpss. And that's the command that Matlab uses to find those functions dpss.

Here's what they look like. So these are five functions that give lambda close to 1 or for a particular bandwidth in a particular time window. The n equals 1 function is a single peak. It looks a lot like a Gaussian, but it's not a Gaussian.

What's fundamentally different between this function and a Gaussian? This function goes to 0 outside that time window, whereas a Gaussian goes on forever. The second slepian in this family has a peak, a positive peak in the left half, a negative peak in the right. The third one has positive, negative, positive, and then goes to 0. And the higher order functions just have more wiggle.

They all have the property that they go to 0 at the edges. And the other interesting properties that these functions are all orthogonal to each other. That means if you multiply this function times that function and integrate, you get 0. Multiply any two of these functions and integrate over the window minus T/2 to plus T/2 the integral [INAUDIBLE]

What that means is that the spectral estimate you get by windowing your data with each of these functions separately are statistically independent. You actually have multiple different estimates of the spectrum from the same little piece of [AUDIO OUT] The other cool thing is that remember the problem with windowing our [AUDIO OUT] with one peak like this is we were throwing away data at the edges. Well, notice that the higher order slepian functions have big peaks at the edges. And so they are actually measuring the spectrum of the parts of the signal that are at the edge of the window.

Now notice that those functions start crashing into the edges. So you start getting sharp, sharp edges out here, which is why the higher order functions have worse ripples outside that central lobe. Any questions about that? It's a lot [AUDIO OUT] Just remember that for a given width of the window in time and within frequency, there are multiple of these functions that put the maximum amount of power in this window 2W.

So that's great. So good question. What would you do if you're trying to measure something and you measure it five different times, how would you get an estimate of what the actual number is? How would you get an error bar on how good your estimate is?

For deviation of your estimates, right. And that's exactly what you do. So not only can you get a good estimate of the average spectrum by averaging all of these things together, but you can actually get an error bar. And that's really cool.

So here's the procedure that you use. And this is what's in that little function W spec.m. So you select a time window of a particular width. How do you know what with to choose? That's part of it.

The other thing is if your signal is changing rapidly in time and you actually care about that change, you should choose-- you're more interested in temporal resolution. If your signal is really constant, like it doesn't change very fast, then you can use bigger windows.

So we're going to choose a time width. Then what you're going to do is you're going to select this parameter p, which is just the time-bandwidth product And if you've already chosen T, what you're doing is you're just choosing the frequency resolution.

Once you compute p and you know T, you just stuff those numbers into this Matlab function dpss, which sends back to you this set of functions here. It sends you back k of those functions that once you've chosen p, k is just 2p minus 1. And then what do you do? You just take your little snippet of data.

You multiply it by the first taper, compute the spectrum, compute the Fourier Transform. And then take your little piece of data, multiply it by the second one, compute the Fourier transform, and the power spectrum. And then you're just going to average.

This square magnitude should be inside the window, your piece of data. You Fourier transform it, square magnitude, and then average all those spectra together. I see this is Fourier transform right here.

This sum is the Fourier transform that. We square magnitude that to get the spectral estimate of that particular sample. Then we're going to average that spectrum together for all the different windowing tapering function.

Now you get then multiple spectral estimates. You're going to average them together to get the mean. And you can also look at the variance to get the standard deviation.

Questions? Let's stop there. That was that was a lot of stuff. Let's take a breath and

[INAUDIBLE] to see whether we [AUDIO OUT] Questions? No.

Don't worry about it. This is representation of the Fourier transform. You sum over all the time samples. This, you will just do as fast Fourier transform.

So you'll take the data, multiply it by this taper function, which is the slepian and then do the Fourier transform, take the square magnitude. We just want make sure that we've got the basic idea.

So you've got a long piece of data. You're going to lose some time window, capital T. You're going to choose a bandwidth W or this time bandwidth product p. Bend T and p to this dpss function. It will send you back a bunch of these dpss functions that fit in that window.

Now you're going to take your piece of data. You're going to break it into little windows of that length, multiply them by each one of the slepian functions Do the Fourier transform of each one of those products. Average them all altogether. Take the square magnitude of each one to get the spectrum, and then average all those spectra together.

So now so what does p do? p chooses bandwidth of the slepian function in that window. So if you have a window that's 100 milliseconds wide-- so we're going to take our data and break it into little pieces that's milliseconds long. It's goes from minus 50 to plus 50 milliseconds. Choose a window that has a narrow bandwidth, the small p, then the bandwidth is narrow.

The bandwidth is narrow, because the function is wide, or you can choose a large bandwidth. What does that mean? It's a narrower function in time.

Now if p is 5, you have a broader bandwidth. And that means that the window, the tapering function is narrower in time. Look at the Fourier transform of each of two different tapering functions. You can see that if p equals 1.5, the tapering function is broad. But that Fourier transform, a kernel in frequency space is narrower.

Take the p equals 5 function, a broader bandwidth, it's narrower in time and broader in frequency. Does that makes sense? p just for a given size time window tells you how many different samples we're going to take within that time window. no

So let me just go back to this example right here. So I took this speech signal that I just showed you that was recorded on the microphone. I chose a time window of 50 milliseconds. So I broke the speech signal down into little 50 millisecond chunks.

I chose a bandwidth of 60 hertz. That corresponds to p equals 1.5 and k equals 2. That gives me back a bunch of these little functions. And I computed this spectragram.

For this spectragram, I chose a shorter time window, eight milliseconds, choose a bandwidth of 375 hertz, which also corresponds to p equals 1.5 and k equals 2. And if you Fourier transform the spectragram with those parameters, you get this example right here.

So in this case, I kept the same p, the same time-bandwidth product, but I made the time shorter. So the best way to do this, when you're actually doing this, practically is just to take a signal by some of these different things [INAUDIBLE] That's really the best way to do it. You can't-- I don't recommend trying to think through beforehand too much exactly what it's going to look like if you choose these different values when it's easier just to try different things and see what it looks like. Yes.

AUDIENCE: What are looking for?

MICHALE FEE: Well, it depends on what you're trying to get out of the data. If you want to visualize formants, you can see that the formants are much clearer. These different windows give you a different view on the data. So just look through different windows and see what looks interesting in the results. That's the best way to do it.

So I just want to say one more word about this time-bandwidth product. So the time-bandwidth product of any function is greater than 1. So you can make time shorter, but bandwidth is worse.

The way that you can think about this as that you're sort of looking at your data through a window in time and frequency. What you want is to look with infinitely fine resolution in both time and frequency, but really you can't have infinite time and frequency resolution. You're going to be smearing your view of the data with something that has a minimum area, the time-bandwidth product which has a minimum of size of 1.

You can either make time small and stretch the bandwidth out, or you can stretch out time and make the bandwidth shorter or make time short and make the bandwidth long. But you can't squeeze both, because of this fundamental limit on the time bandwidth product. This all depends on how you're measuring the time and the bandwidth. These are kind of funny shaped functions.

So there are different ways of measure what the bandwidth of a signal is or what the time width of a signal is. Now, the windows that you're looking in time and frequency with are the smallest time bandwidth product. So notice that if the time-bandwidth product is small, close to 1, the number of tapers you get in this dpss, this family of functions you get is just 1.

If p is 1, then k is 2P minus 1, which is 1. So you only get one window. You only get one estimate of the spectrum, but you can also choose to look at your data with worse, [AUDIO OUT] that have a worse time frequency time-bandwidth product. Why would you do that? Why would you ever look at your data with functions that have a worse time-bandwidth product?

Well, notice that if the time-bandwidth product is 2, how many functions do you have? Why does that matter, because now you have three independent estimates of what that spectrum is. So sometimes you would gladly choose to have a worse resolution in time and frequency, because you've got more independent estimates means better.

So sometimes your signal might be changing very slowly. And then you can use a big time-bandwidth product. It doesn't matter. Sometimes your signal is changing very rapidly in time. And so you want to keep the time-bandwidth product small.

Does that begin to [INAUDIBLE] bigger time-bandwidth products and now you get even more independent estimates. Most typically, you choose p's that go from 1.5 to multiples of 0.5, because then you have an integer number of k's. But usually, you choose p equals 1.5 or higher in multiples of 0.5.

If you really care about temporal resolution and frequency resolution, you want that box that's smearing out your spectrum to be as small as possible. Small as possible means it has an area of 1. That's the minimum area it can have, but that only gives you one taper. But if you really care about both temporal and frequency-- time and frequency resolution, then that's the trade-off you have to make.

Slowly you can air out more in time, maybe more in frequency, and you can choose your time-bandwidth product, in which case you get more tapers and a better estimate of the spectrum. So this is state of the art spectral estimation. It doesn't get better than this.

To put it like this, you're doing it the best possible way, a bit of digesting. So lets spend the rest of the lecture today talking about filtering. So Matlab has a bunch of really powerful filtering tools.

So here's an example of the kind of thing where we use filtering. So this is a [INAUDIBLE] finch song recorded in the lab. OK, so now I want you to just listen-- so you were probably listening to the song, but now listen at very low frequencies. Tell me what you hear. Listen at very low frequen--

[AUDIO PLAYBACK]

[FINCH CHIRPING]

[END PLAYBACK]

[HUMS]

--background, that's hum from the building's air conditioners, air handling. It all makes this low rumbling, which adds a lot of noise to the signal that can make it hard to see where the syllables are in the [AUDIO OUT] time series. Here are the syllables right here. And that's the background noise. But the background noise is at very low frequencies.

So sometimes you want to just filter stuff like that away because we don't care about the air conditioner. We care about the bird's song. OK, so we can get rid of that by applying-- what kind of filter would we apply to this signal to get rid of these low frequencies?

[INAUDIBLE]

AUDIENCE: A high pass.

MICHALE FEE: A high-pass filter, very good. OK, so let's put a high pass filter on this. Now, in the past, previously we've talked about using convolution to carry out a high pass filtering function. But Matlab has all these very powerful tools. So I wanted to show you what those look like and how to use them.

OK, so this is a little piece of code that implements a high-pass filter on that signal. Now, you can see that all of that low frequency stuff is [AUDIO OUT] You have a nice clean, silent background. And now you can see the syllables on top of that background. Here's the spectrogram. You can see that all of that low frequency stuff is gone.

And this is a little bit of sample code here. I just want to point out a few things. You give it the Nyquist frequency, which is just the sampling rate divided by 2. I'll explain later what that means. You set a cutoff frequency. So you tell it to cut off below 500 hertz. You put the cutoff and Nyquest frequency together, you get a ratio of those two that's basically the fraction of the spectral width that you're going to cut off.

And then you tell it to give you the parameters for a Butterworth filter. It's just one of the kinds of filters that you use. Tell it whether it's a high-pass or low-pass. Send that filter, those filter parameters, to this function called filtfilt. You give it these two parameters, B and A, and your data vector.

And when run that, that's what the result looks like, OK? Let me play that again for you after filtering. All that low frequency hum is gone.

All right, so here's an example of what-- I mean, we would never actually do this in the lab. But this is what it would look like if you wanted to emphasize that low frequency stuff. Let's say that you're the air conditioner technician who comes and wants to figure out what's wrong with the air conditioner.

And it turns out that the way it sounds really is helpful. So you now do a low-pass filter. And you're going to keep the low frequency part. Because all those annoying birds are making it hard for you to hear what's wrong with the air conditioner. OK, so here's-- Didn't quite get rid of the birds. But now you can hear the low frequency stuff much better.

OK, all right, so now we just did that by, again, giving it the Nyquist. The cutoff, we're going to cut off above 2,000, pass below 2,000. We're going to tell it to use a Butterworth filter, now low-pass. And again, we just pass it the parameters and the data. And it sends us back the filtered data.

OK, you can also do a band-pass. OK, so a band-pass does a high-pass and a low-pass together. Now you're filtering out everything above some number and below some number. And here we give it a cutoff with two numbers. So it's going to cut off everything below 4 kilohertz and everything above 5 kilohertz.

Again, we use the Butterworth filter. You leave off the tag to get a band-pass filter. And here's what that sounds like.

[AUDIO PLAYBACK]

[BIRDS CHIRPING]

[END PLAYBACK]

And thats a band-pass filter. Questions? Yeah, so there are many different ways to do this kind of filtering. Daniel, do you know how filtfilt actually implements this? Because Matlab has a bunch of different filtering functions. And this is just one of them. [INAUDIBLE] how it's actually implemented [AUDIO OUT]

Right, so there's a filt function, which actually does a convolution in one direction. And filtfilt does the convolution one direction and then the other direction. And what that does it [AUDIO OUT] output center with respect to the input, centered [AUDIO OUT] Anyway, there are different ways of doing it. And the nice thing about-- yeah?

AUDIENCE: [INAUDIBLE]

MICHALE FEE: Well, for the bird data, it doesn't necessarily make all that much sense, right, on the face of it? But there are applications where there's some signal at that particular band. So for example, let's say you had a speech signal and you wanted to find out when the formants cross a certain frequency.

Let's say you wanted to find out if somebody could learn to speak [AUDIO OUT] if you blocked one of their formants whenever it comes through a particular frequency. OK, so let's say I have my second formant and every time it crosses 2 kilohertz I play a burst of noise. And I ask, can I understand if I've knocked out that particular formant?

I don't know why you'd want to do that. But maybe it's fun, right? So I don't know, it might be kind of cool.

So then you would run a band-pass filter right over 2-kilohertz band. And now, you'd get a big signal whenever that formant passed through that band, right? And then you would send that to an amplifier and play a noise burst into the person's ear. All right, we do things like that with birds to find out if they can learn to shift the pitch of their song in response to errors. OK, so yes, they can. Yes--

AUDIENCE: [INAUDIBLE]

MICHALE FEE: Formants are the peaks in the filter that's formed by your vocal tract by the [AUDIO OUT] air channel from your glottis to your lips. The location of those peaks changes [INAUDIBLE] So ahh, ooh, the difference between those is just the location of those formant peaks. All those things just have formants at different location.

AUDIENCE: [INAUDIBLE]

MICHALE FEE: So explain a little bit more what you mean by analog interference.

AUDIENCE: [INAUDIBLE]

MICHALE FEE: Oh, OK, like 60 hertz. OK, so that's a great question. So let's say that you're doing an experiment. And you [AUDIO OUT] contamination of your signal by 60 hertz noise from the outlet. OK, it's really better to spend the time to figure out how to get rid of that noise.

But let's say that you [INAUDIBLE] advisor your data [AUDIO OUT] quite figured out how to get rid of the 60 hertz yet [AUDIO OUT] How would you get rid of the 60 hertz from your signal? You could make what's called a band-stop filter where you suppress frequencies within a particular band. Put that band-stop filter at 60 hertz. The thing is, it's very hard to make a very narrow band-stop filter.

So we learned this in the last lecture. How would you get rid of a particular [AUDIO OUT] Yeah, so take the Fourier transform of your signal, that 60 hertz [AUDIO OUT] one particular value of the Fourier transform. And you can just set that [AUDIO OUT] Because the filtering in that case would be knocking down a whole band of [AUDIO OUT] frequencies.

AUDIENCE: [INAUDIBLE]

MICHALE FEE: Well, it's just that with filtfilt, like I said, there are many different ways of doing things. filtfilt won't do that for you. But once you know this stuff that we've been learning, you can go in and do stuff.

You don't have to have some Matlab function to do it. You just know how it all works and you just write a program to do it. OK, that's pretty cool, right? All right-- oh, and here's the band-stop filter. [INAUDIBLE] that lag there stop, OK?

OK, let's keep going. Oh, and there's a tool here that's part of Matlab. It's called a filter visualization tool, FV tool. You just run this and you can select different kinds of filters that have different kinds of roll-off in frequency, that have different properties in the time domain. It's kind of fun to play with.

If you have to do filtering on some signal, just play around with this. Because there are a bunch of different kind of filters that have different weird names like Butterworth and Chebyshev and a bunch of other things that have different properties. But you can actually just play around with this and design your own filter to meet your own [AUDIO OUT]

OK, so I want to end by spending a little bit of time talking about some really cool things about the Fourier transform and talk about the Nyquist Shannon theorem. This is really kind of mind boggling. It's pretty cool.

So all right, so remember that when you take the Fourier transform-- the fast Fourier transform of something-- [INAUDIBLE] take the Fourier transform of something analytically, the Fourier transform is defined continuously. At every value of F, there's a Fourier transform. But when we do fast Fourier transforms, we've discretized time and we've discretized frequency, right?

So when we take the fast Fourier transform, we get answer back where we have a value of the Fourier transform at a bunch of discrete frequencies. So frequency is discretized. And we have frequencies, little samples of the spectrum at different frequencies, that are separated by a little delta f. [INAUDIBLE] What does that mean?

Remember when we were doing a Fourier series? What was it that we had to have to write down a Fourier series where we can write down an approximation to a function as a sum of sine waves and multiples of a common frequency? What was it about the signal in time that allowed us to do that? It's periodic. We could only do that if the signal is periodic.

So when we write down our fast Fourier transform of a signal, it's discretized in time and frequency. What that means is that it's periodic in time. So when we pass a signal that we've sampled of some duration and the fast Fourier transform algorithm passes back a spectrum that's discreted in frequency, what that means is that you can think about that signal as being periodic in time, OK?

Now, when you discretize the signal in time, you've taken samples of that signal in time separated by delta t. What does that tell you about the spectrum? So when we pass the Fourier transform FFT algorithm, a signal that's discretized in time, it passes us back this thing here, right, with positive frequencies in the first half of the vector, the negative frequencies in the second half. It's really a piece-- it's one period of a periodic spectrum. [AUDIO OUT] right?

Mathematically, if our signal is discretized in time, it means the spectrum is periodic. And the FFT algorithm is passing back one period. And then there's a circular shift to get this thing. Does that makes sense?

OK, now, because these are real functions, this piece here is exactly equal to that piece. It's symmetric. The magnitude of the spectrum is symmetric. So what does that mean?

What that means, if our signal has some bandwidth-- if the highest frequency is less than some bandwidth B-- if the sampling rate is high enough, then you can see that the frequency components here don't interact with the frequency components here. You can see that they're separated.

OK, one more thing. The period of a spectrum 1 is over delta t [INAUDIBLE] which is equal to the sampling rate. So when we have a signal that's discretized in time, the spectrum is periodic and there are multiple copies of that spectrum, of this spectrum, at intervals of [AUDIO OUT] rate.

OK, so if the sampling rate is high enough, then the positive frequencies are well separated from the negative frequencies if the sampling rate is higher than twice the bandwidth [AUDIO OUT] If I sample the signal at a slower and slower rate but it's the same signal, you can see at some point that negative frequencies are going to start crashing into the positive frequencies. So you can see that you don't run into this problem as long as the sampling rate is greater than twice the highest frequency [INAUDIBLE]

So what? So who cares? What's so bad about this? Well, it turns out that if you sample at a frequency higher than twice the bandwidth, the highest frequency in the signal, then you can do something really cool. You can perfectly reconstruct the original signal even though you've sampled it only discretely.

Put an arbitrary signal in. You can sample it discretely. And as long as you've sampled it at twice the highest frequency in the original signal, you can perfectly reconstruct the original signal.

Back to this. Here's our discretely sampled signal. There is the spectrum. It's periodic. Let's say that the sampling rate is more than twice the bandwidth. How would I reconstruct the original signal?

But remember that the convolution theorem says that by multiplying the frequency domain, I'm convolving in the time domain, OK? So remember that this piece right here was the spectrum of the original signal, right? As I sampled it in time, I added these [AUDIO OUT] copies at intervals of the sampling rate. If I want to get the original signal back, I can just put a square window around this, keep that, and throw away all the others. [INAUDIBLE]

By sampling regularly, I've just added these other copies. But they're far enough away that I can just throw them off. I can set them to zero. Now, when I put a square window in the frequency domain, what am I doing in the time domain? Multiply by a square window in frequency, what am I doing in time? [INAUDIBLE]

So basically what I do is I take the original signal sampled regularly in time. And I just convolve it with what? What's the Fourier transform of a square pulse? [INAUDIBLE] If I could just convolve the time domain [INAUDIBLE] with a kernel, that's the Fourier transform of that square window. It's just the sync function.

And when I do that, I get back the original function. But it's actually easier to do. Rather than convolving with a sync function, it's easier just to multiply in the frequency domain. So I can basically get back my sampled function at arbitrarily fine [AUDIO OUT]

Here's how you actually do that. That process is called zero-padding. OK, so what you can do is you can take a function, Fourier transform it, get the spectrum. And what the Fourier transform hands us back is just this piece right here [INAUDIBLE] But what I can do is I can just move those other peaks away. So that's what my FFT sends back to me.

Now what am I going to do, I'm just going to push that away and add zeros in the middle. Now, inverse Fourier transform, and [INAUDIBLE] So the sampling rate is just the number of frequency samples I have times delta f. And here I'm just adding a bunch of frequency samples that are zero. And my new delta t is just going to be 1 over that new sampling rate.

Here's an example. This is a little bit of code that does it. Here I've taken a sine wave at 20 hertz.

You can see 50 millisecond spacing sampled four times per cycle. I just run this little zero-padding algorithm. And you can see that it sends me back these red dots. [INAUDIBLE] have more completely reconstructed the sine wave that I sampled. OK, but you can do that with any function as long as the highest frequency in your original signal is less than [AUDIO OUT] half the sampling rate. [INAUDIBLE]

So zero-padding, so what I showed you here is that zero-padding in the frequency domain gives you higher sampling, faster sampling, in the time domain. OK, and you can also do the same thing. You can also zero-pad in the time domain to give finer spacing in the frequency domain. FFT samples will be closer together in the frequency domain.

OK, so here's how you do that. So you take a little piece of data. You multiply it by your DPSS taper. And then just add a bunch of zeros.

And then take the Fourier transform of that longer piece with all those zeros added to it. And when you do that, what you're going to get back is an FFT that has the samples in frequency more finely spaced. Your delta f is going to be smaller.

Now, that doesn't [AUDIO OUT] frequency resolution. There's no magic getting around the minimum time [INAUDIBLE] product. OK? But you have more samples in frequency.

All right, any questions? [INAUDIBLE] We're going to be starting a new topic next time. We're done with spectral analysis.