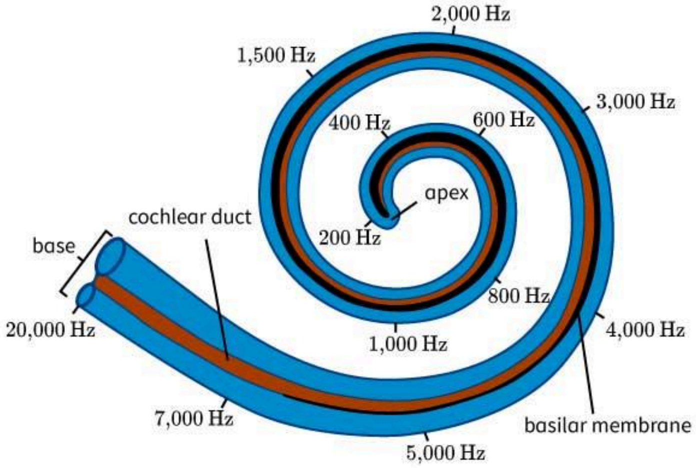
Landon Minkow - Summer at Northwestern University Feinberg School of Medicine
Department of Otolaryngology (ENT)
Supervisors: Dr. Claus-Peter Richter, Dr. Joaquin Cury

Last summer, I worked as a research intern at the Northwestern Feinberg School of Medicine to help build code for a new generation of cochlear implant medical devices. Over three months, I worked closely with my postdoc, Dr. Joaquin Cury, under Dr. Claus-Peter Richter. In an effort to be careful with respect to confidentiality and patents, I will summarize our results and what I learned below. The primary accomplishment was the optimization of the processing time (runtime of the code) from 300 ms to less than 1 ms for the signal processing + FFT.

The Problem

Roughly 1.5 billion people (~20% of the world) suffer from some level of hearing loss, and almost 15% of those with hearing aids still struggle. In contrast to hearing aids, cochlear implants are half implantable and half external devices for those with profound to complete hearing loss. The eligibility for cochlear implant candidacy depends on significant thresholds of sound energy being required at various frequencies for a person to distinguish sounds. As frequencies increase, these thresholds one must exceed to be considered decrease, as it generally becomes harder for the human ear to differentiate between two very high pitches compared to two lower pitches. Still, it is estimated that some 30 million people could benefit from cochlear implants.

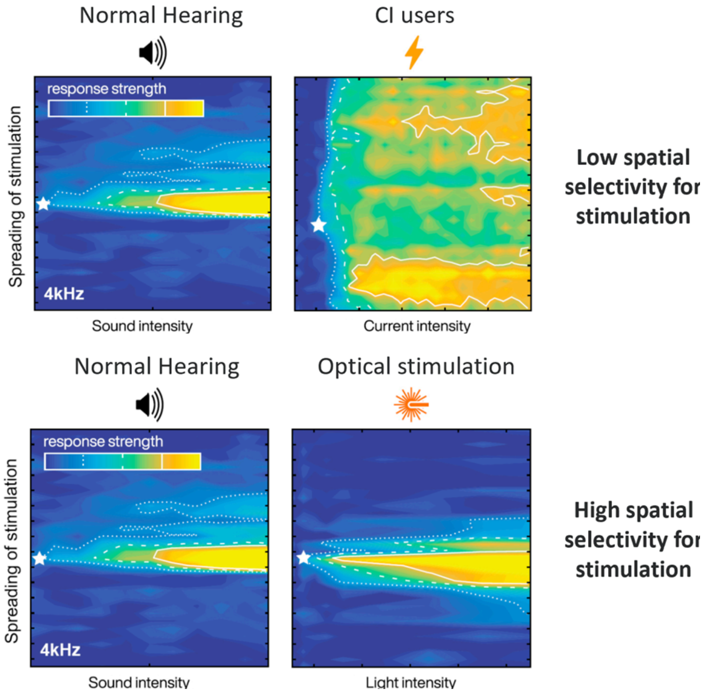
Consider the structure of the cochlea:



The cochlea has what is called a tonotopic structure—in other words, different frequencies of incoming sound activate the basilar membrane (which is lined with hair cells) of the cochlea at different places. Frequency is tied to pitch. Thus, audio signals of lower frequencies will trigger the cochlea towards the narrower apex, while higher frequencies (sounds at higher pitches) activate the cochlea closer to the wider base.

Cochlear implants are not new devices. The issue our model seeks to resolve is that of spatial selectivity. Current models struggle, especially in noisy environments, because there are not enough separate channels to filter unique and discreet audio signals. The frequency bands in the cochlea are not of equal size. So imagine a patient goes to a concert and listens to a music note at, say 800 Hz. If the associated sound wave passes through the cochlea and activates a frequency band spanning from 500-900 Hz, for example, the part of the cochlea that gets activated might be close, but not precisely, 800 Hz. Consequently, the signal that brain reconstructs could be off compared to the actual sound, and that error can be larger depending on how high or low the original pitch is.

The goal is to *enhance* spatial selectivity. This is essentially asking, “How can we activate a more exact subset of neurons, rather than a broader range?” This is where I learned about optical v. electrical stimulation:



In short, optical stimulation (bottom) has the advantage over traditional electrical stimulation (top) when it comes to spatial selectivity, which is apparent by the far more reduced spread of the signal from the “Normal Hearing” actual sound. The drawback: optical stimulation is less efficient. Imagine little proteins within the cochlear membrane, and having to shine a light in order to excite them. This can require 10x or even 100x more energy than electrical stimulation, but precision still matters. Thus, we have to weigh a trade-off between computational costs and resolution.

My Work

Again, I will not be revealing any code or algorithms exactly, but what follows is a broad overview of the process.

First comes signal processing. The external part of the cochlear implant (the thing you can see on the side of the head) has a microphone and processor to take in the raw audio input, which is the noise around you. The receiver transmits the signals to an electrode placed within the cochlea:

Remember, people with full hearing loss do not naturally experience the electrical process of stimulation of hair cells along the cochlea. Thus, the electrode has contacts to mimic the various frequency bands.

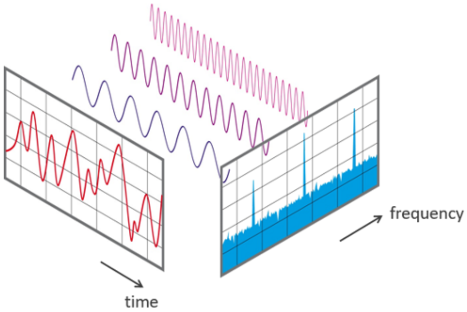
These inputs come in as voltages of variable type int-32, and for the next part, we need variable type floats. I will not get too far into the various parts of the computer here, but Dr. Cury taught me the importance of delegating separate tasks to the **ADC** (analog→digital converter ⇒ so the computer can read the signal), the **DMA** (direct memory access ⇒ to store the value), and the **CPU** (which performs the next algorithm). This way, different parts of the computer perform their own functions, so they can run simultaneously.

But converting from integers to floats takes time, and we need to process the audio signal continuously. So when the empty vector fills up with integers, we cannot wait for the delay in converting each element until the vector is empty again). No spoilers here, but we were able to fill some of the vector elements, and then process those values (into float types) while the rest of the vector filled up. Imagine a continuously-spinning wheel, where input is taken in and then outputted as the next batch is taken in, and so on…

A few steps later…

FFT

And now we get to the **Fast Fourier Transform (FFT)**, our way of teleporting from the time domain to the frequency domain:



There is a lot going on around you right now: the sound of your bedroom fan, the white noise in the air, that person who's typing so loud in the other room.

In short, many different frequencies are at play.

The cochlear implant's microphone takes in this raw signal, which is really the composition of all these frequencies together. Refer to the above frequency v. time graph on the left of the image. In layman's terms, the FFT takes that "everything-signal" and extracts the individual frequency components—the 500 Hz from the fan, the 200 Hz from the white noise, etc…

Now, view the right side of the above image, which shows a frequency chart. There are very low levels detected of each frequency except for the peaks at Frequencies a, b, and c. These would correspond to that fan, white noise, and the person typing. Pretty cool right?!

There are several parameters I examined related to the FFT. Here's where some real thinking is required:

Frequency Resolution and Sampling

Keep in mind that our goal is precision and spatial selectivity. Better precision means we can reconstruct the true audio input more accurately.

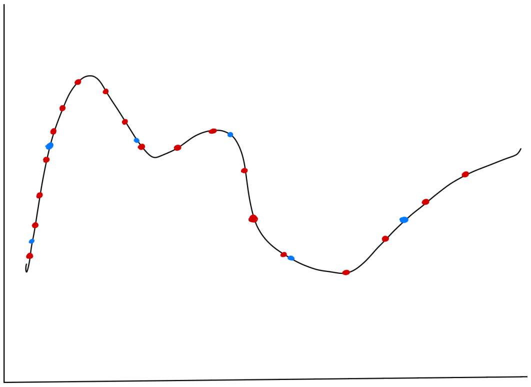
The **frequency resolution** behind the FFT is one to describe this. The frequency resolution=Sampling Rate/Buffer Size (buffer size is our sampling size):

$$\Delta f = fs/N$$

Clearly, increasing our buffer size will decrease the magnitude of the frequency resolution. *Note:* A smaller resolution implies more precision. Why? Because frequency resolution is really a *change in frequency*, so at smaller resolutions, we see smaller changes in frequency.

Remember those wide frequency bands from earlier? Wouldn't it be better if our FFT checked for the presence of all frequencies spaced apart by 50 Hz rather than by 150 Hz. That way, if an audio signal is truly 80 Hz will fall somewhere between 50-100 Hz, which is far more precise than if it was simply categorized within 50-200 Hz.

So how do we pick our **buffer/sample size**? The traditional choices are powers of 2—32, 64, 128, 256, 512, 1024, or 2048 samples. Remember that vector from before? The one we had to fill and then process continuously? The size of that vector is the length of the FFT (the buffer size).



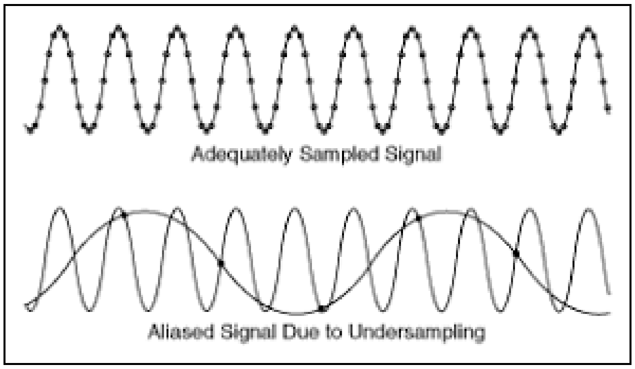
So if we sample at the red points, we are taking more samples in a given window of time. Because they are closer together, this would give us a more accurate reconstruction of the sound signal, whereas if we sample at the blue, we take in data less frequently and thus leave larger gaps that may be filled in incorrectly.

BUT, sampling more frequently (at the red points) requires more computing power, and we don't want to require a large battery (that has to be charged by the patient every few hours) or a bulky cochlear implant to be worn.

So what's the optimal FFT length?

This is where I did a lot of data analysis, testing how changes in sampling size or sampling frequency affected the runtime. That data is not shared here, but generally, 2048 samples will run too slowly, and only 32 samples will lead to decreased accuracy in rebuilding the signal (though it will run much quicker). So really, it boils down to finding a compromise that balances processing time and accuracy in recreating the signal.

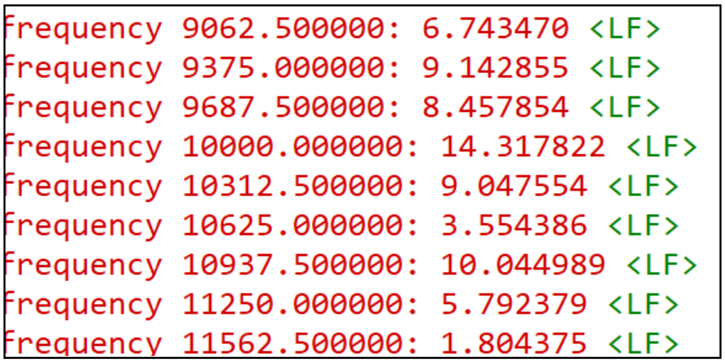
Another concern, and a pretty interesting phenomenon, is **aliasing**. To keep it short, the Nyquist Frequency indicates the sampling rate (Fs) must be at least 2x the maximum frequency in the original signal, or else we risk some distortion. A cool example can be found here (notice if the camera is not taking snapshots quickly enough, you lose some accuracy in capturing the movement of the wheel).


Adequately Sampled Signal
Aliased Signal Due to Undersampling

Testing

I got to use a lot of really cool devices during my time at the lab. I connected wires to the microphone, got to speak into it and test different sounds, and use a Baby Sound Card. Lots of new hardware!

Additionally, I learned about the STLink, the UART Interface display, and voltage conversion from a 9V battery to a lower energy that could be better handled by the computer.

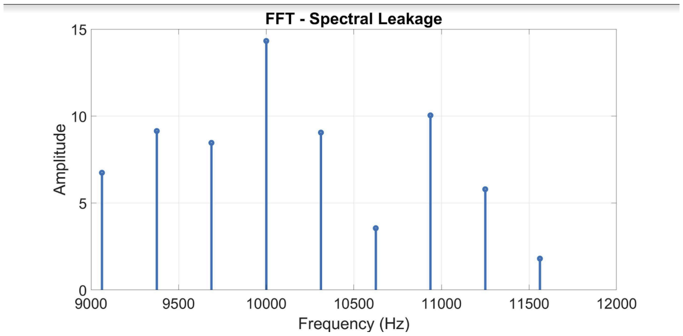


```
frequency 9062.500000: 6.743470 <LF>
frequency 9375.000000: 9.142855 <LF>
frequency 9687.500000: 8.457854 <LF>
frequency 10000.000000: 14.317822 <LF>
frequency 10312.500000: 9.047554 <LF>
frequency 10625.000000: 3.554386 <LF>
frequency 10937.500000: 10.044989 <LF>
frequency 11250.000000: 5.792379 <LF>
frequency 11562.500000: 1.804375 <LF>
```

Above is a sample output (using the Docklight software) of the FFT. Our frequency resolution is 312.5Hz (notice how we have outputs for all frequencies spaced apart by 312.5Hz), and this is the result of a pure tone 10,000 Hz signal I played from my computer.

The outputs are the relative energy levels picked up by the microphone. As expected, a significantly higher concentration of energy falls right at 10,000 Hz. But this is why frequency resolution matters! If we had a poorer resolution, that energy may have just fallen into a wide frequency bin anywhere from 9000-15,000 Hz.

While most of the energy is distributed at 10,000 Hz (14.317822), what about the relatively large energy levels at 10,312.5 Hz, or 9375 Hz? This is where I learned about **spectral leakage**, which is where energy spills over into neighboring frequency bands.



Recap

This is about where my time ended, though the next steps in the coding process would involve sending electrical pulses through the auditory nerve, telling the brain which frequencies (and how much of each relative to the others) were detected.

I used the following tools in my work:

- MATLAB
- C
- Microphone
- STM 32 IDE (to interact with physical microboard)

Throughout my research, I played the role of biologist, software engineer, data scientist, and doctor—the last of which when I presented my work to my fellow ENT lab members.

I'm fortunate to have learned so much at Northwestern, especially given the impact these advances will have on patients, which I hope to witness during the testing phases. Additionally, being surrounded by MD and PhD students kept me engaged in a new, but very ambitious environment.