# Tuning Out The Haters: Noise Cancelling Technology and Fourier Series

Eric Yen, Shreya Nagpal, Susanna Weber

December 11$^{th}$ 2020

## Theory

For our capstone project, we wanted to recreate the active noise cancelling technology used in noise-cancelling headphones. This is distinct from passive noise cancelling, which simply uses material to block out sound, the way a conversation in the next room is muffled. Active noise cancelling uses a negated Fourier series to cancel out sound. The noise is approximated with a Fourier series , which is then inverted, in order to cancel out the noise with destructive interference. The closer the noise itself is to a series of sin and cosine terms, the better it can be cancelled out, and the less noise will come through the headphones. Our goal was to recreate this fitting process, and see how well we could cancel out different types of noise. We then compared this to how well noise cancelling headphones could block out noise.

## Procedure

### Set Up

We used two different types of noise, one at a constant pitch, and one a series of dissonant tones, so with different pitches. This was to test how well our fitting process would work with different sounds. In total, we had three distinct sounds that we played, one at a constant pitch of 220 HZ, one at 880 HZ, and one made up of several dissonant noises.

We had three different set ups that we used for our experiment, and in order to compare our noise cancelling code to the headphones. The two set ups used for comparison were one without any headphones, and one with a pair of noise cancelling headphones around the microphone (see figure). For our third set-up, we used regular (non noise cancelling) headphones. This was to look for any irregularities in the headphones, and to make sure that passive noise cancelling wasn't already blocking out all the noise.

Our set up was very simple, due to only having limited materials at home. For the set up with no headphones, it involved either simply playing audio from a phone placed a set distance from the microphone. In the set up with the headphones and noise cancelling headphones, we placed the microphone inside a paper towel tube, and then attached the headphones around it.

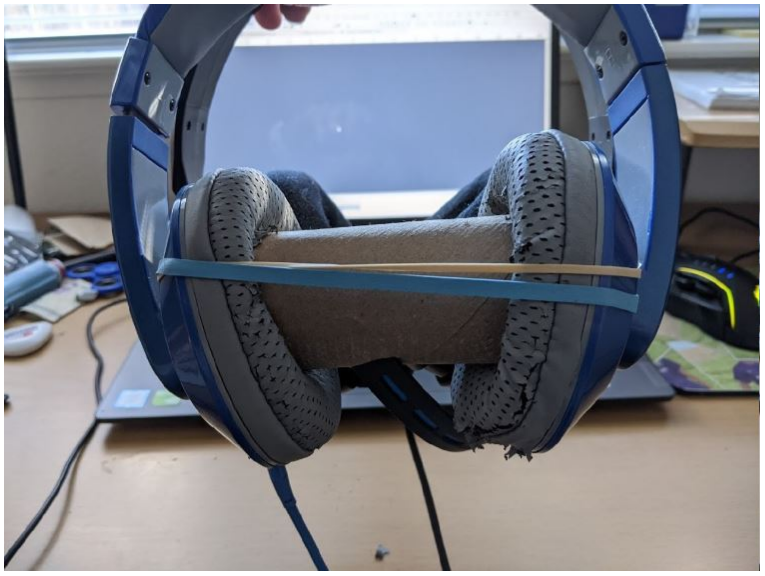


Figure 1: Our setup for the headphones and noise cancelling headphones. The microphone sits in the middle of the paper towel tube.

All data was recorded in Audacity, and each data set was a fraction of a second long, since the software takes data very rapidly.

## Code

Our code functions the same way that the technology in noise cancelling headphones functions. For the constant frequency noise used the Python fitting function to fit the noise to an eight term Fourier series in sin cosine form,

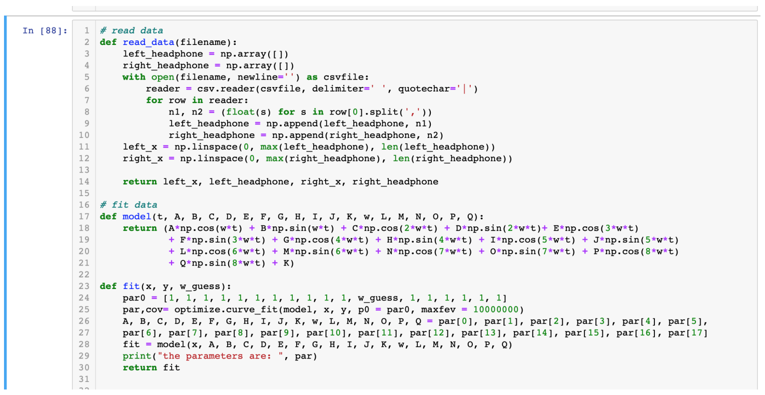
$$f(x) = a_0 + \sum_{n=1}^{N} (a_n * cos(nx\omega) + b_n * sin(nx\omega)), \tag{1}$$

where $N = 8$, and $x$ is the time in seconds, and $\omega$ is the frequency of the signal. Each coefficient term was calculated to 95% confidence bounds.

For the multiple-frequency noise, we used the Matlab fitting function. This is because this proved more difficult to fit than the constant noise, and the Matlab fitter worked much better in this case, because it was better at estimating the fitting coefficients.

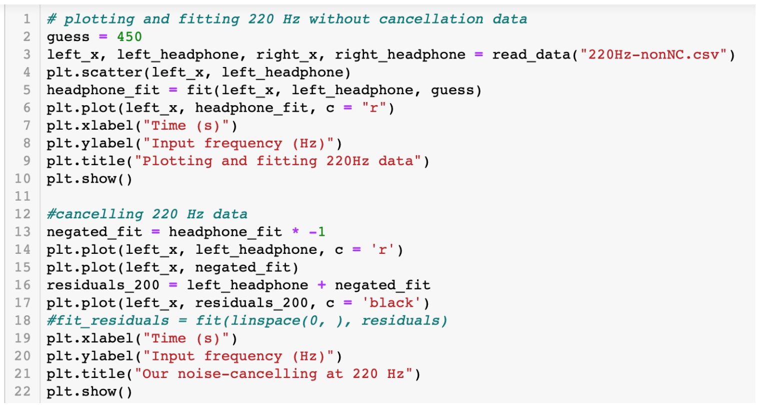
We then negated the fit to find the inverse. We then added this to the original noise, in order to cancel them out. The remaining noise was then treated as the noise that would be heard through our experimental headphones.

Attached are screenshots of the Jupyter notebook code we used to noise cancel the constant tones we played. The full Jupyter Notebook attached at the end of the report. The Matlab code is also attached.



```
In [88]:
# read data
def read_data(filename):
    left_headphone = np.array([])
    right_headphone = np.array([])
    with open(filename, newline='') as csvfile:
        reader = csv.reader(csvfile, delimiter=' ', quotechar='|')
        for row in reader:
            n1, n2 = (float(s) for s in row[0].split(','))
            left_headphone = np.append(left_headphone, n1)
            right_headphone = np.append(right_headphone, n2)
    left_x = np.linspace(0, max(left_headphone), len(left_headphone))
    right_x = np.linspace(0, max(right_headphone), len(right_headphone))

    return left_x, left_headphone, right_x, right_headphone

# fit data
def model(t, A, B, C, D, E, F, G, H, I, J, K, w, L, M, N, O, P, Q):
    return (A*np.cos(w*t) + B*np.sin(w*t) + C*np.cos(2*w*t) + D*np.sin(2*w*t)+ E*np.cos(3*w*t)
            + F*np.sin(3*w*t) + G*np.cos(4*w*t) + H*np.sin(4*w*t) + I*np.cos(5*w*t) + J*np.sin(5*w*t)
            + L*np.cos(6*w*t) + M*np.sin(6*w*t) + N*np.cos(7*w*t) + O*np.sin(7*w*t) + P*np.cos(8*w*t)
            + Q*np.sin(8*w*t) + K)

def fit(x, y, w_guess):
    par0 = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, w_guess, 1, 1, 1, 1, 1, 1]
    par,cov= optimize.curve_fit(model, x, y, p0 = par0, maxfev = 10000000)
    A, B, C, D, E, F, G, H, I, J, K, w, L, M, N, O, P, Q = par[0], par[1], par[2], par[3], par[4], par[5],
    par[6], par[7], par[8], par[9], par[10], par[11], par[12], par[13], par[14], par[15], par[16], par[17]
    fit = model(x, A, B, C, D, E, F, G, H, I, J, K, w, L, M, N, O, P, Q)
    print("the parameters are: ", par)
    return fit

```

Figure 2: Section of code showing the Audacity data file being read, and then being fitted.



```python
# plotting and fitting 220 Hz without cancellation data
guess = 450
left_x, left_headphone, right_x, right_headphone = read_data("220Hz-nonNC.csv")
plt.scatter(left_x, left_headphone)
headphone_fit = fit(left_x, left_headphone, guess)
plt.plot(left_x, headphone_fit, c = "r")
plt.xlabel("Time (s)")
plt.ylabel("Input frequency (Hz)")
plt.title("Plotting and fitting 220Hz data")
plt.show()

#cancelling 220 Hz data
negated_fit = headphone_fit * -1
plt.plot(left_x, left_headphone, c = 'r')
plt.plot(left_x, negated_fit)
residuals_200 = left_headphone + negated_fit
plt.plot(left_x, residuals_200, c = 'black')
#fit_residuals = fit(linspace(0, ), residuals)
plt.xlabel("Time (s)")
plt.ylabel("Input frequency (Hz)")
plt.title("Our noise-cancelling at 220 Hz")
plt.show()
```

Figure 3: Plotting the data, fit, and negated fit.

## Comparison

The remaining noise from our code was then compared to the data from the set up with the noise cancelling headphones. Theoretically, if our code worked just as well as the headphones, these two data sets should be the same.

Comparing the amplitude of our two data sets, we can then see how our code measures up to the headphones. More details on how this was done can be found in the analysis section.

## Data

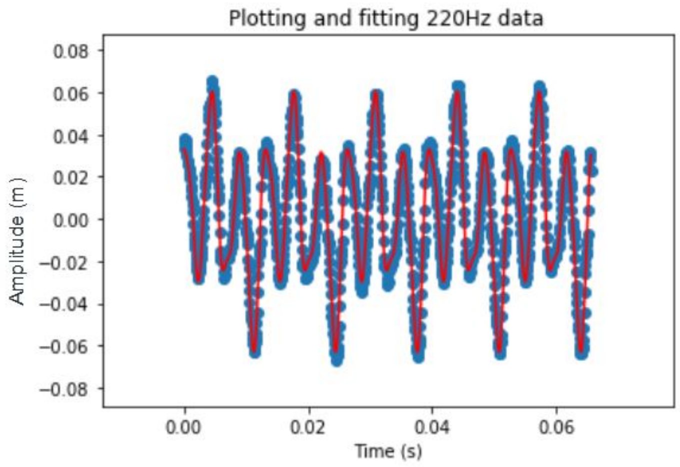
The x axis of the data is graphed in seconds, and the y axis shows noise amplitude, which is distinct from decibels, which are a relative variable. In Audacity, the y axis goes linearly from +1 at the top (the maximum possible loudness for the positive signal) to -1 at the bottom (the maximum for the negative signal). Zero amplitude means silence.

### 220 Hz

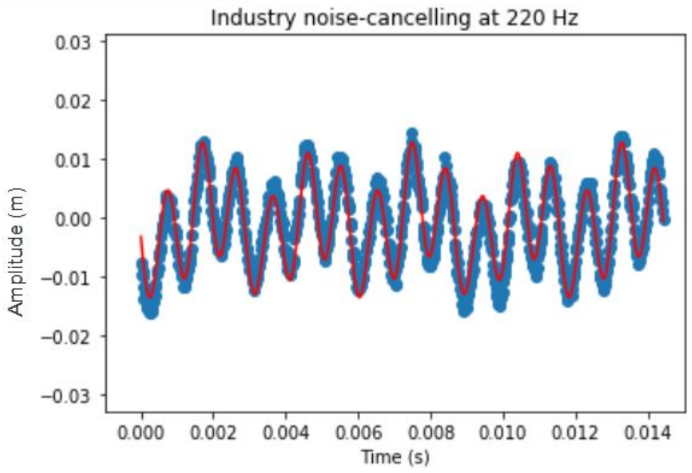
#### No Headphones

The data shown here is for a trial with no headphones and a consistent sound at 220 Hz. The first graph includes the raw data and our Fourier fit, while the second graph includes the raw data, negated fit, and the remaining noise after our active noise cancelling.

By inspection, the data appears fairly periodic and the leftover noise seems to have very small amplitudes compared to the original noise. This smaller residual amplitude correlates to a far lower overall volume. All of our fit coefficients were found with 95% confidence bounds.



(a) 220 Hz data (blue) and Fourier series fit (red). label 1



(b) 220 Hz noise heard through NC headphones (blue) and fit (red) label 2

Figure 4: 220 Hz data, without headphones (left) and with noise cancelling headphones (red). Both sets of data are fit with an eight term Fourier series.

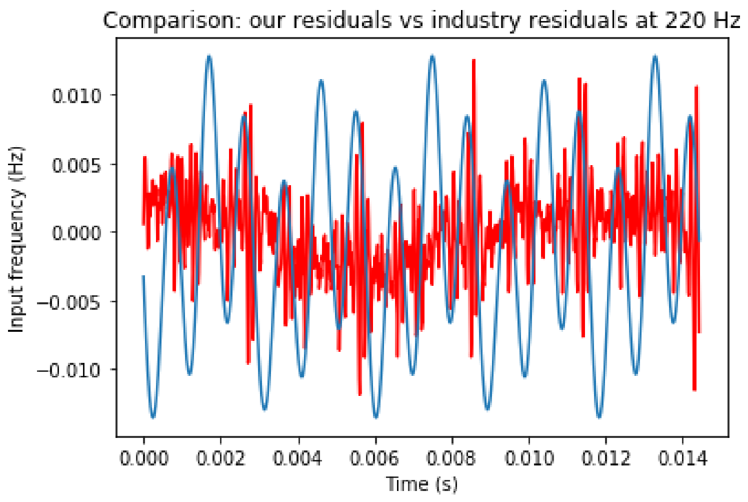


Figure 5: Comparing the remaining noise after our fit in Python (red), and the noise heard through the noise cancelling headphones (blue) for the 220 Hz data.

### Regular Headphones

This ended up just being a test case to ensure that there are no irregularities or unexpected effects in the date. We expected our graph to be very similar to that with no headphones, though likely with some smaller amplitudes due to headphone material doing some passive noise-cancelling. It looks like the net volume is lower, as expected, but there are no other obvious effects. Since there are no new apparent results, we don't consider this case for future cases.

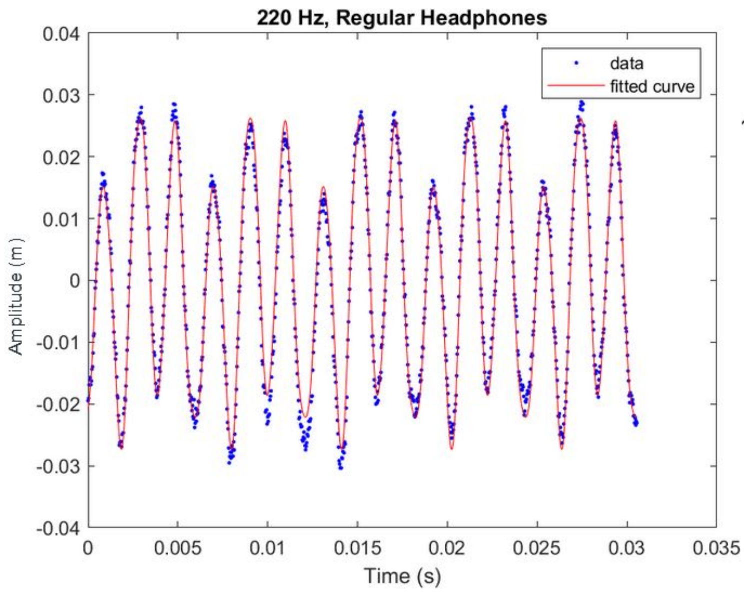


Figure 6: Data points for 220 Hz tone (blue) and Fourier series fit (red). We used the regular headphones to ensure there were no issues with our set up, such as the passive noise cancelling blocking out all the noise, or other irregularities.

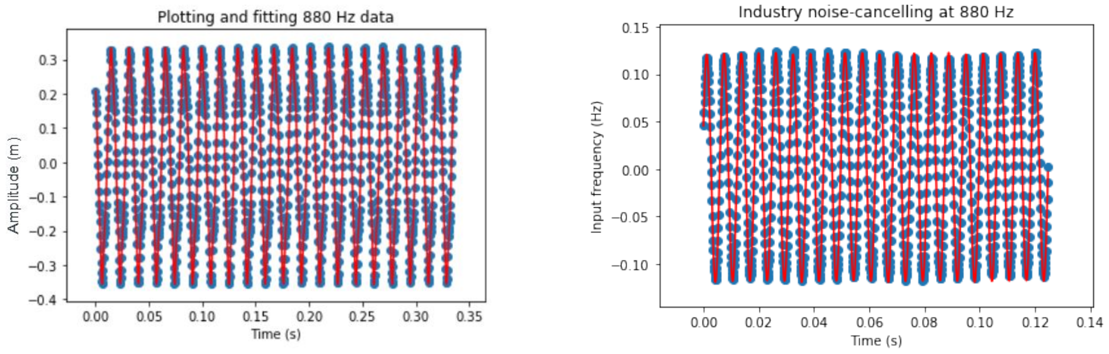
### Noise-cancelling Headphones

The following images are the results from repeating the process with noise-cancelling headphones at 220 Hz. The left graph plots the sound that we are able to hear through the noise-cancelling headphones, which we can infer to be the leftover noise after the headphones perform their active noise-cancellation procedure. In the second graph, we compare the remaining noise after our fit in Python (red) and the noise heard through the noise cancelling headphones (blue) for the 220 Hz data. Our residuals are a bit noisier, but they are roughly the same amplitude, indicating that our fit was roughly on par with the noise cancelling headphones.

## 880 Hz

The data shown here is for a trial with no headphones and a consistent sound at 880 Hz. The first graph includes the raw data and our Fourier fit, while the second graph includes the raw data, negated fit, and the remaining noise after our active noise cancelling.

By inspection, the data once again appears fairly periodic and the leftover noise seems to have very small amplitudes compared to the original noise again. The residual noise actually appears to be smaller than that for our 220 Hz graph at first glance, but note that the y-axis is at a different scale and the magnitudes of the residual sounds are actually just about the same.



(a) 880 Hz data (blue) and Fourier series fit (red)

(b) 880 Hz noise heard through NC headphones (blue) and fit (red)

Figure 7: 880 Hz data, without headphones (left) and with noise cancelling headphones (red). Both sets of data are fit with an eight term Fourier series. The fit in both cases resembles the data very closely, and the higher frequency is clear in the data.

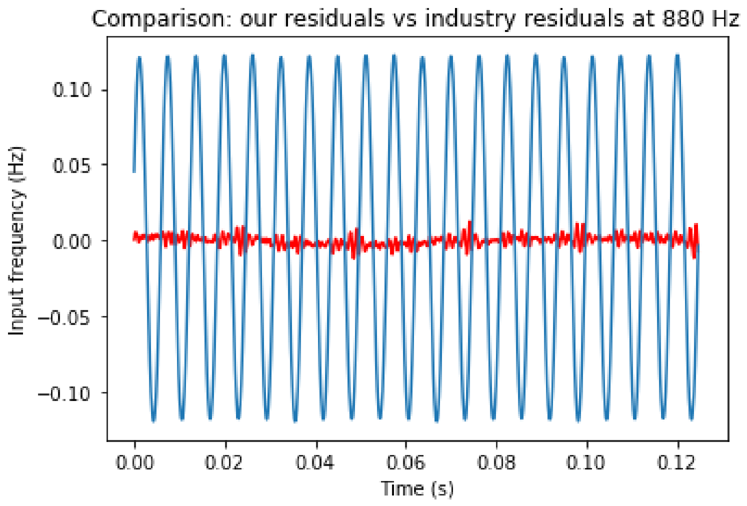


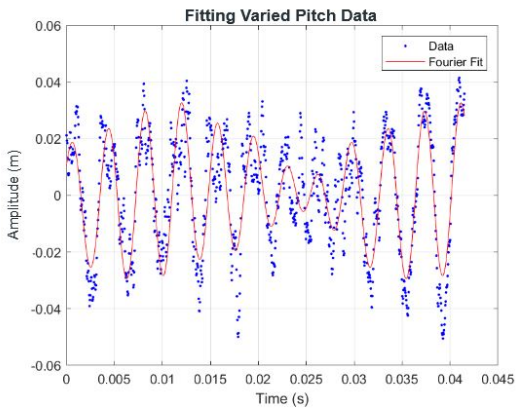
Figure 8: Comparing the remaining noise after our fit in Python (red), and the noise heard through the noise cancelling headphones (blue) for the 880 Hz data. Interestingly, our residuals have a far smaller amplitude than the noise coming through the headphones. We discuss potential reasons for this in the main text.

Interestingly, our residual noise seems to have smaller amplitudes than that of the noise-cancelling headphones. Possible reasons why may be because industry headphones don't try to get to 0 dB, only below noticeable volume levels.

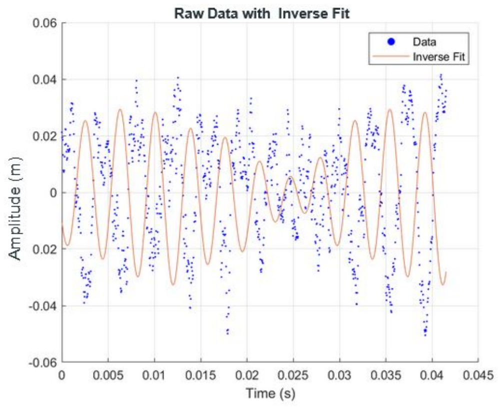
## Variable Pitch

When we repeated the same procedure for sounds of varying pitches, we found it to be a lot noisier and (unsurprisingly) harder to fit. Graphs for this data were produced in MatLab instead of Python.

The left graph below is the raw data and our fit for it. The right graph is the amalgamation of our fit, negated fit, and the residuals. The fit still isn't very good and seems to even add noise in some cases. The leftover volume very obviously varies a lot more than for constant pitches.

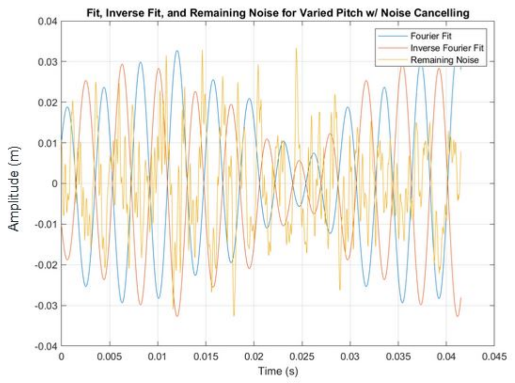


(a) The data for varying pitch noise (blue), and the Fourier series fit (red). Note how much less sinusoidal than the constant pitch noise the data is.

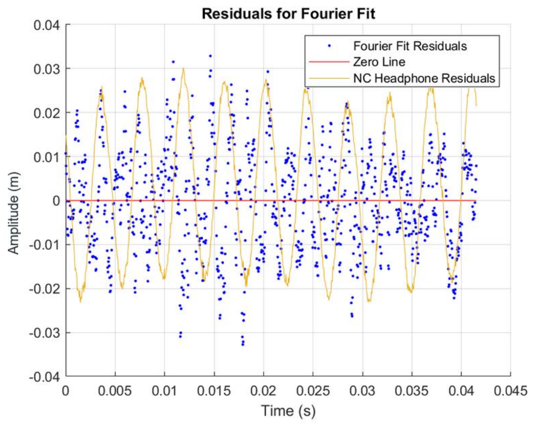


(b) The varying pitch data (blue), Fourier series fit (red), and inverse fit (orange). Because the data is so noisy, the fit and inverse fit is much less accurate.

Figure 9: Varying pitch data and fit (left), and data, fit, and inverse fit (right). Due to the varying pitches, this data is far less sinusoidal, and thus harder to fit.



(a) Varying pitch fit and inverse fit (blue and orange, respectively), and the remaining noise after adding them (yellow). This fit is far less successful than the previous ones due to irregularity of the data.



(b) The remaining noise through the headphones (yellow) and the remaining noise after the data is cancelled by the Python code (blue). One can see in the frequency and amplitude that neither the headphones or our code are as successful as in the previous experiments with the constant pitch tones.

Figure 10: The remaining noise and fits from our code (left), and the comparison between our code's remaining noise and the noise coming through the headphones. Both our fitting method and the headphones themselves struggle with this set of data.

Once again, our leftover noise has approximately the same magnitude as that from the noise-cancelling headphones. This does seem to be consistent with experience of wearing noise-cancelling headphones - variable sounds are a lot harder to cancel out than more consistent sounds.

## Analysis

Here, we numerically compare the success of our fits, and then compare these to the noise cancelling headphones. Our first metric. Results can be found in the table below.

First, we compare sum of squared estimate of errors of each fit. The SSE is the sum of the squares of residuals. Thus, it measures the discrepancy between the data and the Fourier fit. It is given by the equation

$$SSE = \sum_{i=1}^{N} (y_i - f(x_i))^2, \tag{2}$$

where $y_i$ is the value of the $i^{th}$ data point, $f(x_i)$ is the predicted value, and N is the total number of data points. We found the lowest value for our data taken at a frequency of 220 Hz, indicating that this was our best fit. As we expect from our qualitative observations, the SSE for the varying fit is a factor of 10 larger than for the constant pitch fits.

We then compared our fits using the root-mean-square error (RMSE), which measures the standard deviation of the residuals. This is given by the formula

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - f(x_i))^2}{N}}, \tag{3}$$

where once again $y_i$ is the value of the $i^{th}$ data point, $f(x_i)$ is the predicted value, and N is the total number of data points. Here, we once again find very low RMSE values for our constant frequency fits, indicating that they are fairly accurate, and a much larger value for the varied frequency fit.

Finally, we compare the average amplitude (”loudness”) of our residual noise, and converted it to decibels. To do this, we simply averaged our data points after subtracting the negated fit (such as the red data points in Figure 8), and then converted this result to decibels using the formula

$$dec = 20 * \log \frac{a}{a_r}, \tag{4}$$

where log is the base 10 logarithm, $a$ is the amplitude, and $a_r$ is the reference amplitude. Average amplitudes and decibel values are listed in the table with uncertainties.

| Frequency | SSE | RMSE | Average Amplitude | Average Decibel Value |
|---|---|---|---|---|
| 220 HZ | 0.01117 | 0.003372 | 0.0026 ± 0.0005 | 46.12 ± 3.37 dB |
| 880 Hz | 0.02107 | 0.004632 | 0.0037 ± 0.0002 | 57.24 ± 5.10 dB |
| Varying | 0.1309 | 0.01154 | 0.0903 ± 0.0009 | 46.71 ± 10.28 dB |

We can then compare the last two columns to the average volume that makes it through the noise cancelling headphones, to see how our code compares to the functioning of the noise cancelling headphones. We also compare the maximum values. Values are given in amplitude, and each table shows results for one of the three different sounds played.

| 220 Hz | NC Headphones | Our Set Up |
|---|---|---|
| Mean | 0.0063 ± 0.0008 | 0.0026 ± 0.0005 |
| Standard Deviation | 0.0074 ± 0.0003 | 0.0033 ± 0.0005 |
| Max | 0.0144 ± 0.0001 | 0.0125 ± 0.0001 |

| 880 Hz | NC Headphones | Our Set Up |
|---|---|---|
| Mean | 0.0317 ± 0.0009 | 0.0037 ± 0.0002 |
| Standard Deviation | 0.0849 ± 0.0008 | 0.0046 ± 0.0005 |
| Max | 0.1248 ± 0.0001 | 0.0118 ± 0.0001 |

| Varied Pitch | NC Headphones | Our Set Up |
|---|---|---|
| Mean | 0.0093 ± 0.0004 | 0.0670± 0.0009 |
| Standard Deviation | 0.0114 ± 0.0007 | 0.0837 ± 0.0003 |
| Max | 0.0415 ± 0.0001 | 0.2233 ± 0.0001 |

## Conclusion

Our analysis indicates that we were able to cancel noise at a similar level to noise cancelling headphones, both for constant tones and varied pitches (though neither the code nor the headphones were very good at blocking the latter). We found the constant tones very easy to fit with an eight term Fourier series, and were able to cancel them to inaudible levels. The varied pitch tone was much more difficult to fit and cancel, and as a result produced more error than monotone audio.

An important distinction to make is that between the measurement units of dB SPL (Sound Pressure Level) and db FL (Full Scale). Units of db SPL correspond with the ”loudness” of sound when traveling through a medium, whereas db FL is used strictly to measure ”loudness” in a digital context, such as in Audacity. Since we know that Audacity measures sound in dB FL, we are able to find its reference amplitude of 0.0001 to use in our calculations using equation (4). However, just knowing the decibel levels isn't enough to tell us whether sound is audible or not, as human ears become less sensitive to sound as it gets farther away from approximately 1 kHz. This is the theory behind why humans can't hear dog whistles even though the decibel level is comparable to other sounds humans can hear. Though we calculated decibel values equatable to the hum of a refrigerator (40 dB) or a nearby conversation (60 dB), further analysis of our software is needed to determine whether this noise is indeed clearly audible.

This lines up with what we would expect, due to the far less sinusoidal shape of the varied pitch data, so it would be far harder to approximate with a series of sin and cosines. Our data also lines up with what one would expect from the experience of wearing noise cancelling headphones, which are generally much better at cancelling out fairly constant pitch and volume noises, like the sound of a car engine, than irregular sounds, like a conversation between two people.

Our comparisons between our remaining noise and the noise coming through the NC headphones show similar results. The average values for the 220 Hz data show that the average, standard deviation, and maximum of the NC headphone data and our data are very similar. For the 880 Hz data, the max and mean of the NC headphones are larger than for our data, which we also saw by inspection of our graph. Both of these results indicate that our fits were successful, and our noise cancelling was roughly on par with that of the headphones.

Our results for the varied pitch data also indicate our less successful fit, showing that the average volume and maximum volume for our set up are higher than for the noise cancelling headphones. Though neither fit was very good at cancelling out the varied pitch noise, we can observe by inspection as well that the noise cancelling headphones were able to block more audio in this set up.

Possible sources of error are our very short acquisition time, so even though we have a large number of data points due to Audacity's sample rate, they span less than a second. Especially for the varied pitch data, this could have affected our results. We also could have run more trials for each different sound, and played different kinds of noises, but were constrained by time. Other sources of error of course include the headphones not sufficiently insulating the microphone, and thus our comparison data for the headphones being louder than it should be.

This experiment could be extended and made closer to the way that noise cancelling headphones actually function by writing a program to take data in real time, fit it, and transform it. This is possible to do with an Arduino, and a connected microphone and laptop. We weren't able to complete this as part of our set up because none of us owned an Arduino kit.