INTERNATIONAL STUDENT CHALLENGE PROBLEM IN ACOUSTIC SIGNAL PROCESSING 2023

Jacob Brotman-Krass, Sochima Omenkeukwu, Jenna Rutowski, Augustus Standeven, and Isaac Wang

University of Rochester: AME472 Final

Abstract

The International Student Challenge Problem in Acoustic Signal Processing 2023 involves processing acoustic sensor data from a set of hydrophones. The goal is to extract information about the source from its corresponding acoustical data. In this case, a diver swims at a constant speed and depth while passing the hydrophones which are located equidistant from each other and an equal distance above the seafloor. The main signal processing tasks were to display the spectrogram of the hydrophones, estimate the breathing rate of the diver, estimate when the diver is closest to the middle hydrophone O, estimate the diver's altitude in relation to the hydrophone array, and estimate the diver's swimming speed.

I. Introduction

The three hydrophones, N, O, and P, are located 1m above the seafloor at a depth of 20 m. The hydrophones are each separated by 14m along a straight line. Hydrophone O is located at the center of the array. Each hydrophone is sampled at 250 kHz. The acoustic signature of the diver consists of periodic pulses of white noise which correspond to the diver inhaling.

II. Task 1a

This specific task involved displaying the output spectrogram for the middle hydrophone O and commenting on the spectral properties. A spectrogram is a visual representation of the frequencies which make up a signal as it varies with time. By analyzing the spectrogram of the hydrophone data, it is possible to analyze and identify frequency components present in the signal and observe how they vary with time.

In terms of the scuba diver's acoustic signature, the spectrogram helps to display characteristic frequency components associated with the sound of the diver's breathing and movement. The noise generated by the scuba gear produces a characteristic spectral pattern that is distinguishable from other noise sources.

By analyzing the spectral properties of the acoustic signature, it is possible to identify the sound source, estimate its location, and extract other information about it, such as its movement. In this case, this periodic noise pattern correlates to the breathing rate of the diver.

The code loads an audio file named "SCP23_Hyd O.wav", which contains audio recording of the underwater data. The audio file is read using the built in audioread() function, and the hydrophone O channel data and sample rate are extracted. Next, a spectrogram is generated using the built in spectrogram() function. The spectrogram is a 2D representation of the audio signal's frequency content over time. It is created by dividing the audio signal into small, overlapping time segments and computing the frequency spectrum for each section. The resulting spectrogram has time on the x-axis and frequency on the y-axis, while the color intensity of each point in the plot represents the magnitude of the frequency content at that point in time. The parameters used for generating the spectrogram are a window size of 1024 and a hop size of 512. The hamming window function is used to reduce spectral leakage and smearing in the frequency domain caused by windowing. Finally, the spectrogram is converted to a logarithmic scale in decibels (dB) using the 20*log10 function. The spectrogram is plotted using the imagesc function. The plot is displayed with the title "Spectrogram of Hydrophone O", and the x- and y-axes labeled with "Time [s]" and "Frequency [Hz]," respectively.

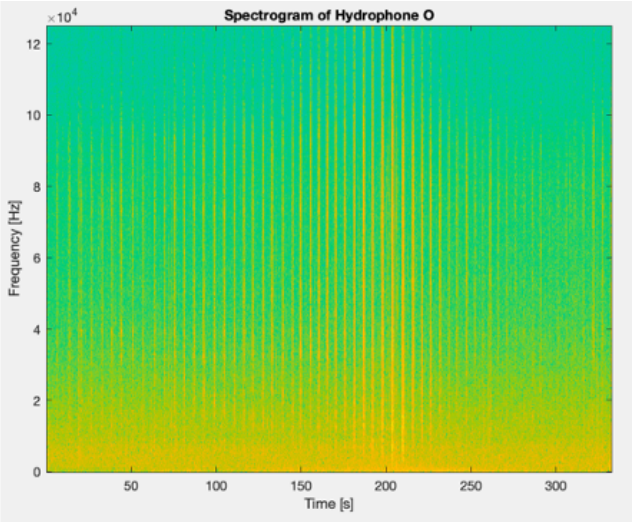

Figure 1: *Spectrogram of Hydrophone O*

Based on the spectrogram of the hydrophone O as seen in Figure 1, we can observe that the scuba diver's acoustic signature has several spectral properties. First, we can see that there are several peaks in the frequency domain that correspond to the various sounds made by the diver. In particular, there is periodic broadband noise corresponding to the diver's breathing rate. This indicates that, barring computational difficulties, a small window size and sizable

overlap will give an optimal depiction of the diver's breathing, which requires little spectral precision but significant time resolution. The intensity of this noise appears to change over time, indicating that the diver seems to swim toward and then away from this particular hydrophone.

III. Task 1b

To estimate the breathing rate of the diver from the spectrogram, we need to look for spectral peaks that correspond to the breathing sounds. First, we can visually review the spectrogram to identify the breathing sounds. From the spectrogram, several vertical bands of energy that correspond to the diver's breathing are observed. To find the breathing rate, it is possible to find the time between these peaks and calculate the corresponding rate in Hz. Based on this visual inspection, it can be estimated the breathing time is about 5 seconds, and therefore the breathing rate is approximately 0.2 Hz.

To confirm this observation through data analysis, one possibility is to use a peak-finding algorithm, such as the findpeaks() function built into MATLAB, to locate the peaks in the spectrogram. It is also possible to implement a Fourier-based method to estimate the breathing rate. This would involve performing a Fourier transform on the audio signal to obtain the power spectrum, and then identifying the frequency band that corresponds to the breathing sounds. In this case, the first method described was employed.

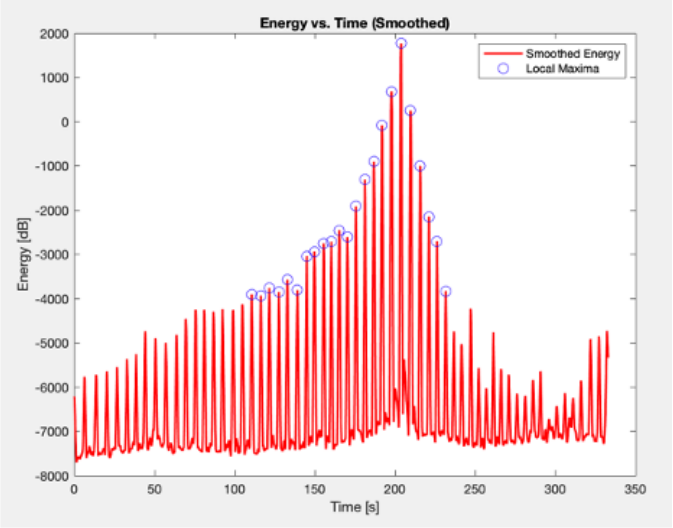

Figure 2: *Finding Maxima*

To find the breathing rate, an audio file was loaded and a spectrogram analysis was performed on the signal for hydrophone O using the built-in spectrogram() function in MATLAB. The energy of the spectrogram was calculated, and a moving average filter was applied to create a resulting energy curve. The smoothed energy curve was used to find local maxima in the energy curve, which are interpreted as peaks in the audio signal. The time intervals between adjacent peaks were calculated using the built-in findpeaks() function and used to estimate the average time interval between peaks. This corresponds to the breathing rate of the diver that produced the audio signal. The code then plotted the smoothed energy curve with the identified local maxima and calculated and displayed the breathing rate in Hz.

Doing this, the average time between breaths was found to be 5.416 seconds. The breathing rate of the diver in Hz was then calculated to be 0.18464 Hz.

Another way to find the breath rate was to use the total energy of the signal. The train of thought behind this method was to get the width between each of the high-intensity bands in the spectrogram. The process involved splitting hydrophone O's signal vector into frames, calculating the total signal energy of each frame, and storing each value into a vector. The next step was to get the FFT of the total energy (see Fig. 3) and use Matlab's findpeaks() function to get the heights and locations of the peaks. The breathing rate was found to be 0.1742Hz.

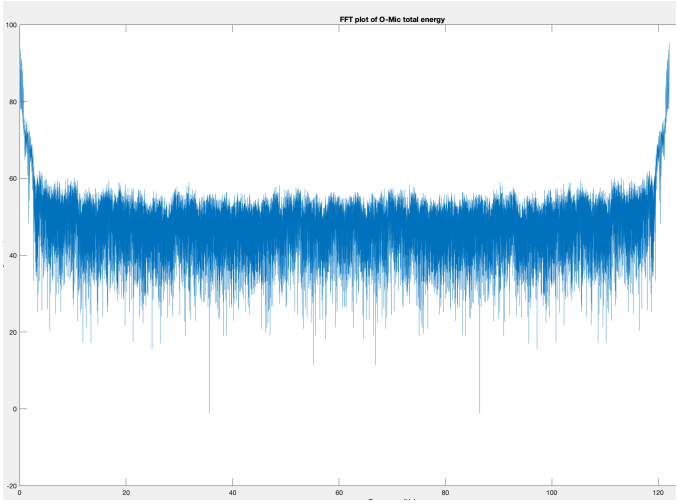

Figure 3: *FFT of the total signal energy of O Mic*

One of the thought processes that has turned out to be a failure is by setting a threshold to eliminate the noise. In this case, the findpeaks( ) function was first applied to locate the peak amplitude values and their corresponding locations. Then, the breath count can be detected by the amount of peaks that are over thresholds, and figure out the breaths per minute from that point. After generating the plot, it turns out that a fixed threshold may not work well in a noisy environment. In underwater recording, there are many variations that can affect the result, such as wind, marine life, wave, etc. In this case, as the diver gets closer to the hydrophone, the amplitude of their breath sounds will increase, but so will the amplitude of the background noise due to the resulting bubbles. If the threshold is set too low, it will result in false positives where the noise is mistaken for breath sounds. On the other hand, if the threshold is set too high, it may miss some of the breath sounds.

Another approach is to use the spectrogram, which has a very clear visualization of each breath. The breaths correspond with a wideband burst of noise that sustains for about a second. After converting the STFT data to decibels, the bins in

each individual frame were summed to get an idea of total spectral intensity. A zoomed-in plot of this is shown below:

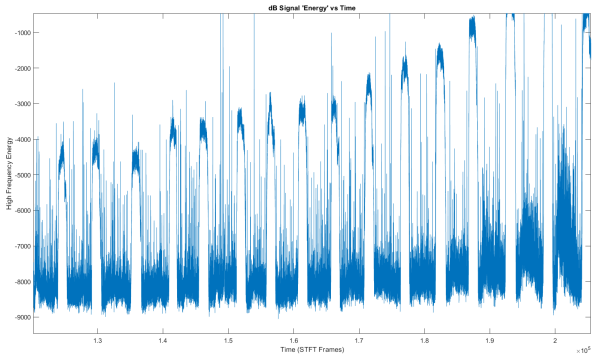

Figure 4: *Spectral 'energy' in dB for each frame*

Clearly, there is a periodicity in this signal, and it appears to correspond to the breathing rate, as confirmed by a listening test. An FFT of the above plot displays a peak of 0.18189 Hz (see Fig. 5), which seems to line up visually with the STFT and audibly with the sound file. Despite 100x zero padding and the use of a Hanning window, there are significant sidelobes. This can likely be attributed to the nonuniformity of the diver's breathing rate; these other periodicities are likely also present in the breathing rate at one time or another.

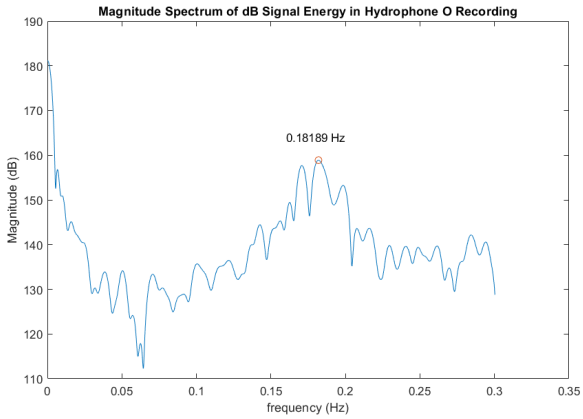

Figure 5: *Magnitude spectrum of the dB signal 'energy' in Hydrophone O*

IV. Task 2a

The code used for this section computes the spectral energy of the audio signal received by each of the hydrophones N, O, and P in the frequency range of interest which is expected to contain the dominant components of the diver's acoustic signature.

First, audio files for three hydrophones are loaded using the audioread() function. Then, the parameters for the spectrogram are set, including the window size and hop size. Spectrograms are then generated for each hydrophone using the spectrogram() function. The spectrograms are then converted to a logarithmic scale using 20*log10(abs(S)). The code then finds the frequency bins that correspond to the frequency range of interest (above 20 kHz and below 125 kHz) and extracts the spectral energy in that range for each hydrophone using sum(). Below 20kHz, there was a lot of noise, and above 125 kHz exceeds the Nyquist limit.

Then, it finds the time index of the peak spectral energy in the spectrogram of hydrophone O, which corresponds to the time when the diver is closest to hydrophone O. The index is converted to the corresponding time in seconds by looking up the time vector T_o. The time index of the peak energy in the spectrogram for hydrophone O is then found using the max() function, and the corresponding time is printed using fprintf(). This time represents the time when the diver is at the closest point of approach to hydrophone O.

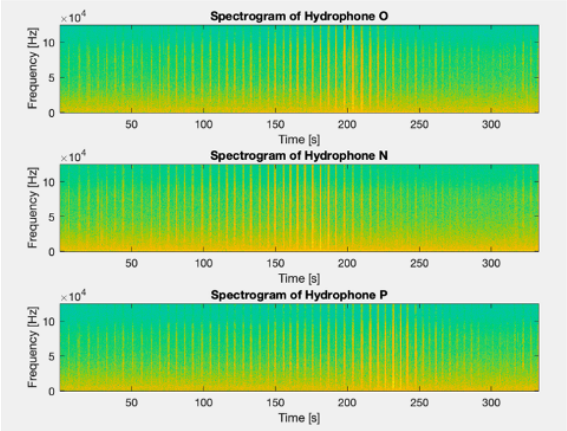

Figure 6: *Spectrograms of Hydrophones O, N, P*

The time of closest point of approach to hydrophone O was found to be 204.04 seconds. This calculation also makes sense visually based on Figure 6 above, as the band of greatest strength for hydrophone O seems to occur around 200s.

Another way to get the closest point of approach was to examine the total signal energy of hydrophone O. The thought process behind this was that at the closest point of approach to the hydrophone, the signal energy will be the highest. This can also be seen in the spectrograms where the points at which the diver is closer to the hydrophones are the brightest points in the plot.

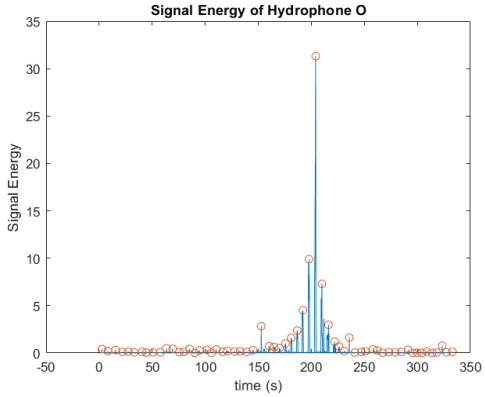

Figure 7: *Signal energy of Hydrophone O with peaks emphasized*

The signal was split into frames and the total signal energy was calculated for each frame and stored in a vector. The findpeaks function was used on the total energy vector to get the peak energy in the signal. The max peak value and the index of that value was derived which was then used in the time vector for the frames. The time at that index was converted to seconds which gave a time of 204.08 seconds.

As the diver's breaths are separated by large blocks of silence, it is fairly likely that the actual time of closest approach is in between breaths. For this purpose, spline interpolation was used to predict the actual peak. Subsequent usage of Matlab's findpeaks() function shows the time of closest approach to Hydrophone O to be 203.83 seconds.

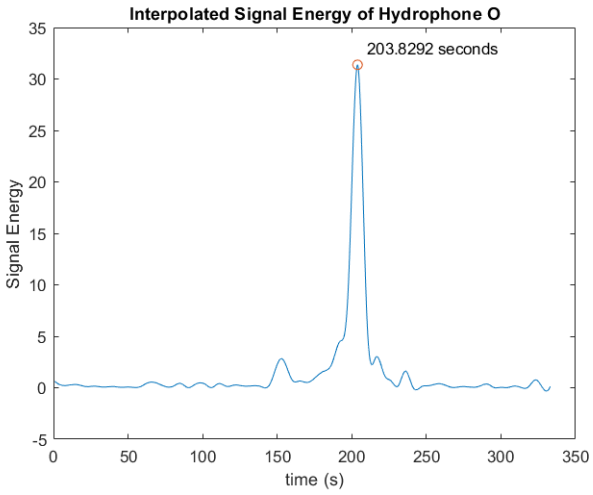

Figure 8: *Interpolated signal energy of Hydrophone O, with peak energy identified at 203.83 s.*

V. Task 2b

Task 2b asks to estimate the diver's altitude above hydrophone O when they are closest to said hydrophone at the time found in task 2a. To estimate the diver's altitude, we need to analyze the time delay between the signals received by each hydrophone. Since the hydrophones are arranged in a straight line, we can find the time delay between signal arrival at adjacent hydrophones and then use Pythagorean Theorem to find the vertical distance between the diver and the hydrophone array.

First, the Pythagorean Theorem was used to put the height component of the diver's position could be put in terms of known variables as seen in Figure 9:

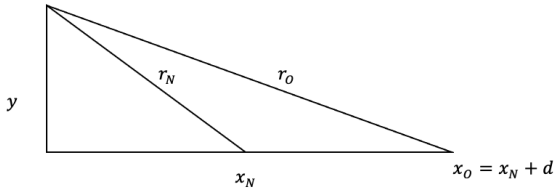

Figure 9a: *Height Diagram*

$$r_N{}^2 = y^2 + x_N{}^2$$

$$r_O{}^2 = y^2 + x_O{}^2 = y^2 + (x_N + d)^2$$

$$r_N = \sqrt{y^2 + x_N{}^2}$$

$$r_O = \sqrt{y^2 + x_N{}^2 + 28x_N + d^2}$$

$$r_O - r_N = c\Delta t = \sqrt{y^2 + x_N{}^2 + 28x_N + d^2} - \sqrt{y^2 + x_N{}^2}$$

$$if\ x_N = 0$$

$$c\Delta t = \sqrt{y^2 + d^2} - y$$

$$(c\Delta t + y)^2 = y^2 + d^2$$

$$c^2\Delta t^2 = y^2 + d^2$$

$$c^2\Delta t^2 + 2c\Delta ty + y^2 = y^2 + d^2$$

$$2c\Delta ty = d^2 - c^2\Delta t^2$$

$$y = \frac{d^2 - c^2\Delta t^2}{2c\Delta t}$$

$$y = \frac{1}{2}\left(\frac{d^2}{c\Delta t} - c\Delta t\right)$$

$$where\ d = 14m, c = 1520\frac{m}{s}, and\ \Delta t\ was\ found\ using\ xcorr()$$

Figure 9b: *Height Derivation*

The time delay between hydrophones N and O and hydrophones O and P can be estimated since the distance between hydrophones and the speed of sound in water are known.

Doing this, the altitude above hydrophone O was found to be 4.31 meters.

For this section of the project we had a lot of difficulty obtaining the time delay between various signal arrivals. The approach our team initially took was to look at the time when the signal was strongest above the O Hydrophone (aka when the diver is directly above the middle hydrophone). At this time (which was approximately 204.08s) the diver breathes and the signal can be seen on all three hydrophones. If we zoom in on this plot with all three hydrophone plots overlaid, we can theoretically choose a noticeable peak or valley in all three signals and discern the time difference between these signals.

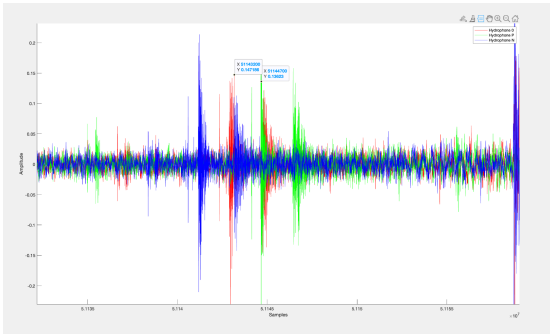

Figure 10: *The Three Hydrophone Plots Overlaid*

Figure 10 displays a noticeable noise spike (not likely a breath) apparent through all three hydrophones. The distance between these spikes appears to be 1500 samples which is about 0.006 seconds.

After further analysis, our team was unsure if these results were a reliable answer for the time delay. This was due to the fact that the noticeable spike appears earlier in time for hydrophone N than hydrophone O. If the noticeable spike was due to the diver (and since we are looking at the time when the diver is directly above hydrophone O and directly inbetween hydrophone N and P) then signals N and P would both be directly after the hydrophone O signal, directly on top of each other. As a result, our team decided to take another approach to find the time delay. We concluded that this spike must be due to some noise farther away, not due to the diver, since it arrives at hydrophone N first.

In order to find the time delay we decided to use the xcorr() function within MatLab. The xcorr function was run on a range of values for each of the hydrophones. The range was about 1 second before the time at each hydrophone and 1 second after the time at the hydrophone converted to samples. The data from xcorr was stored in two vectors c, and lags. The max value of c was generated with its index which was used to get the number of samples at lags(index). The value at lags(index) represents the amount of delay in samples which was then converted to seconds by dividing by the sample rate.

In an attempt to compare the time of arrivals of every breath at the three hydrophones, we crafted an algorithm to analyze the signal spectrum for sustained energy peaks. Due to the high volume of non-breath-related noise, this required the use of a few methods to filter the signal. First, a high-order Butterworth high-pass filter was applied to remove the noisiest frequency bands (0-20 kHz). The resulting spectrogram is shown in Figure 11.

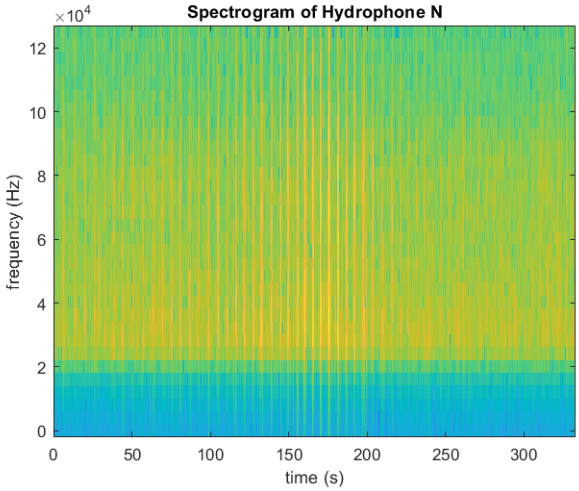

Figure 11: *Spectrogram of Hydrophone N after a 20th-order Butterworth high-pass filter was applied at a cutoff of 20 kHz.*

To extract the clear pattern seen in the spectrogram, a magnitude threshold was set at -39 dB to exclude all low intensity bins. A count of all bins that clear the threshold for each frame is shown below:

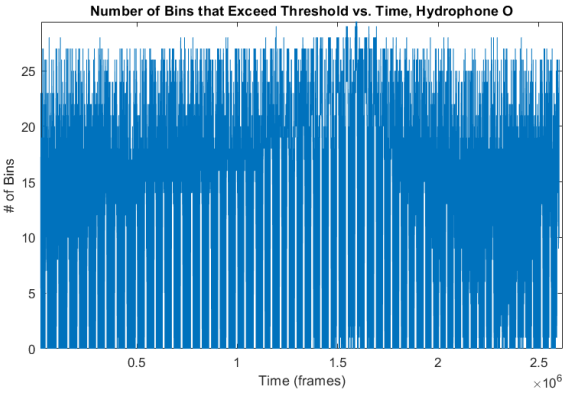

Figure 12: *Plot showing how many bins are greater than the magnitude threshold in each frame. Sustained periods where many bins exceed the threshold correlates well with the breathing rate.*

The noticeable white bands in Figure 12 correspond well to the diver's breaths and appear to show their durations, as well. We devised an algorithm to look ahead/behind each sample and mark spots that begin and end these periods of sustained threshold-clearing. The results, shown in Figure 13, appeared to be promising. However, the level of accuracy needed to properly determine time of arrivals was not achieved with this method. Calculated time delay values gave corresponding heights around 30 m, which was not feasible given the fact that the water was only 20 m deep.

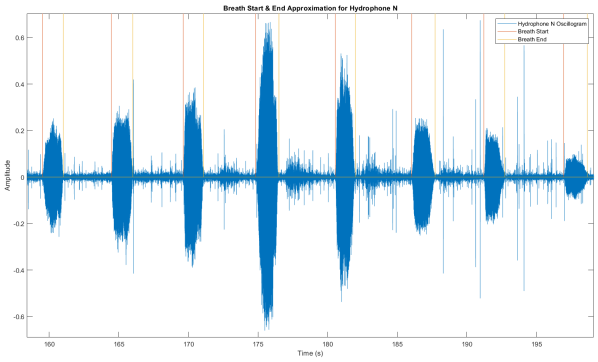

Figure 13: *Amplitude plot of Hydrophone N overlaid with algorithmically derived start and end points of each breath. While initially appearing promising, subsequent height calculation proved this method too inaccurate.*

VI. Task 2c

Task 2c asked to estimate the diver's swimming speed. In this case, the time delays between the hydrophones can still be used. We can also use the time it takes for the diver to move between the hydrophones to estimate their speed. Since we know the spacing of the hydrophones and the time it takes for the diver to move between them, we can calculate the speed of the diver.

To do this, the hydrophone signals were cross correlated to estimate time delays. The time delays between hydrophones were then used in conjunction with the known distances between hydrophones to calculate the speed of the diver and the output was printed. The diver was found to be moving at 0.45 m/s, which is a leisurely pace, but realistic for a slow moving diver.

Another method used to calculate the velocity was a similar approach to that of finding the time at which the diver was closest to each hydrophone. If we know the times at which the diver is closest to each hydrophone, and we know the distance between hydrophones, we can use each of these variables to calculate the velocity of the diver, since $v = d / t$.

The same code from task 2a was used but utilized the datasets from each hydrophone. It was found that the diver was closest to hydrophone N at 175.58 seconds, hydrophone O at 204.04 seconds, and hydrophone P at 238.42 seconds. The distance between each hydrophone is 14 meters, making the distance between the outer hydrophones 28 meters. So, the average velocity of the diver was calculated to be 0.41 m/s between hydrophones O and P, 0.49 m/s between hydrophones N and O, and 0.44 m/s between hydrophones N and P. This is relatively constant, as expected, and averages out to be about 0.45 m/s. This is a plausible velocity for a diver, and about the same as the method which utilized xcorr() previously.

Another method used was to get the time at each mic using the total signal energy method and subtracting the time at hydrophone N from the time at hydrophone P which would give the time difference in the diver's approach between hydrophones N and P. The distance between hydrophone N and P was then divided by the time difference between the two hydrophones to give the speed of the diver which was 0.5m/s.

To confirm the results from the method above, the average of the diver's speed between hydrophones N and O and the speed between hydrophones O and P also gave us 0.5m/s.

VII. Significance

This challenge problem involved the practical implementation of several key signal processing techniques. Take 1a involved displaying an output spectrogram of the hydrophone signal. Spectrograms are very important in analyzing audio signals since they can provide a visual representation of signal changes over both time and frequency by plotting the intensity of the frequency components. Spectrograms are very useful in identifying and analyzing signals and for feature extraction.

Task 1b used a variety of signal processing techniques in tandem. First, energy calculations were utilized. The energy of a signal can provide information about the signal's power, intensity, and loudness. Normalization was then used, which helps when comparing different signals and making them more consistent for processing. A moving average filter was used to smooth out the energy curve and remove noise that may be present. This helps in identifying maxima more accurately, which is helpful for analyzing any audio signal with any sort of noise. Peak detection was utilized, which can provide valuable information about the characteristics of the signal. It can be used to identify transients, periodic sounds, etc. This process helps to simplify complex signals by identifying the most significant parts. It can also extract important features such as amplitude, frequency, or duration. Time interval calculations are critical for any type of rate or frequency analysis.

Task 2a used similar techniques, including spectrogram generation and peak detection. Other important processes for audio included the dB scale conversion, which is common for audio applications since humans hear on a dB scale, as well as spectral energy extraction, which extracts the spectral energy in a frequency range of interest by summing the power spectrogram values across the relevant frequency bins. This is very useful to focus on specific frequency bands in a signal when noise may want to be disregarded.

In task 2b, time delays were calculated and the between hydrophones were used with propagation speed to calculate the distance traveled by a wave. On a conceptual scale, the propagation of sound in a medium was analyzed to determine the location and qualities of a source. The use of cross-correlation to estimate the time delay between signals is a widely used technique in signal processing, particularly in audio processing and speech recognition. The calculation of distance and altitude based on propagation time and applying relevant mathematical concepts is also important for sonar applications.

Lastly, task 2c applied concepts used in part 2a, plus cross-correlation and the relationship between distance and time to calculate velocity. The concepts used in these tasks can be applied to a wide range of signal processing fields, including medical imaging, speech recognition, and machine

health. These are just a few examples of how the signal-processing techniques used in this challenge problem can apply to a wide range of signal-processing applications. Several concepts used throughout this entire semester were conceptually applied in this problem.

VIII. Conclusion

The International Student Challenge Problem in Acoustic Signal Processing 2023 has significant implications in the field of signal processing. This particular problem involves analyzing the acoustic signature of a scuba diver, which consists of periodic broadband emissions. By analyzing the signals received by the three hydrophones arranged in a line array, it is possible to extract certain properties about the diver and environment.

This problem is important because it highlights the potential applications of signal processing in acoustics. Acoustic signal processing can be used to analyze and interpret acoustic signals from a variety of underwater sources, and these same concepts generally apply to acoustics at large as well. The techniques used in this problem can be applied to a range of other underwater or above-water acoustic sensing applications.

Further, this challenge problem provides an opportunity to gain experience in signal processing and to develop approaches for solving practical problems utilizing concepts learned in class. Some important concepts utilized in this problem that apply to a myriad of signal processing applications include the use and interpretation of spectrograms, signal envelopes and filters, power, dB scaling, time of arrival, cross-correlation, Fourier transforms, Doppler effects, and more. Please note full source code is also available as an attached document.

References

[1] Ferguson, B.G., Culver, R.L., & Gemba, K.L. (2023). International Student Challenge Problem in Acoustic Signal Processing 2023. Acoustics Today, 19(1), 52-59.

[2]

[3] The MathWorks Inc. (2021). MATLAB (R2021a) [Computer software]. Natick, MA: The MathWorks Inc.

[4] MathWorks. xcorr. Retrieved April 23, 2023, from https://www.mathworks.com/help/matlab/ref/xcorr.html

```
%% AME472 Final Project- Challenge Problem 2023

clear all;
close all;

%% Task 1a

% Load audio file
[y, fs] = audioread('SCP23_Hyd 0.wav');

% % Extract the first 30 seconds of the audio signal to speed up run time
% y_short = y(180:180+(fs*60), :);

% Or run full wav file
y_short = y;

% Write the signal to a new wave file
audiowrite('filename_shortened.wav', y_short, fs);

% Extract hydrophone 0 channel
hydrophone_0 = y_short(:, 1);

% Set parameters for spectrogram
window_size = 1024;
hop_size = 512;

% Generate spectrogram
[S, F, T] = spectrogram(hydrophone_0, hamming(window_size), window_size-hop_size,↙
window_size, fs, 'yaxis');

% Convert power spectrogram to dB scale
S_dB = 10*log10(abs(S));

% Plot spectrogram
imagesc(T, F, S_dB);
axis xy;
title('Spectrogram of Hydrophone 0')
xlabel('Time [s]');
ylabel('Frequency [Hz]');

%% Part 1b- Method 1

% Calculate the energy of the signal in each frame
energy = sum(abs(S).^2, 1);

% Normalize energy values to the range [0, 1]
energy_norm = energy ./ max(energy);

% Compute energy of the spectrogram along the freq axis
E = sum(S_dB, 1);

% Define window size for the moving average filter
win_size = 500;

% Apply filter to the energy curve
E_smooth = movmean(E, win_size);

% Find local maxima in the smoothed energy curve
```

```
[peak_values, peak_times] = findpeaks(E_smooth, T, 'MinPeakDistance', 1, 'MinPeakHeight',↙
-4000);

% Calculate time intervals between adjacent peaks
time_intervals = diff(peak_times);

% Calculate average time interval between peaks based on local maxima
avg_time_interval = mean(time_intervals);

% Display the average time interval
disp(['Average time interval between peaks: ' num2str(avg_time_interval) ' seconds']);

% Plot the smoothed energy curve with local maxima
plot(T, E_smooth, 'r', 'LineWidth', 1.5);
hold on;
plot(peak_times, peak_values, 'bo', 'MarkerSize', 8);
title('Energy vs. Time (Smoothed)');
xlabel('Time [s]');
ylabel('Energy [dB]');
legend('Smoothed Energy', 'Local Maxima');

% Calculate breathing rate in Hz using time interval
breathing_Rate = 1 / avg_time_interval;

% Display the average time interval
disp(['Breathing Freq: ' num2str(breathing_Rate) ' Hz']);

%% Task 1b- Method 2

[O_Mic,Fs2]= audioread('SCP23_Hyd O.wav');
framelen = 2048;
window = hamming(framelen);
overlap = 0;
t_frame2 = framelen/Fs2;
f_res2 = Fs2/framelen;
ST_O = spectrogram(O_Mic,window,overlap);
[nBins2,nFr2] = size(ST_O);

total_energy_O = zeros(1,nFr2);
for n = 1:nFr2
   start = (n*framelen) - (framelen-1);
   stop = n*framelen;
   frame_data = O_Mic(start:stop);
   tot_e = sum(frame_data.^2);
   total_energy_O(n) = tot_e;
end

te_fft = fft(total_energy_O);
Fs_new = Fs2/framelen;
freqs = (1:length(total_energy_O))*(Fs_new/nFr2);

[pks,locs,w,p] = findpeaks(20*log10(abs(te_fft)));
[pk,ind] = max(pks);
in = locs(ind);
breath_rate = freqs(in);

% Display the average time interval
disp(['Breathing Freq: ' num2str(breath_rate) ' Hz']);
```

```
%% Task 1b- Method 3

[Hyd_O, fs] = audioread("SCP23_Hyd O.wav");
%audioinfo("SCP23_Hyd O.wav")
% STFT Parameters
framelen = 512;                                %window length
mywindow = hanning(framelen, 'periodic');   %window for DFTs
overlap = 1/2*framelen;                      %frame overlap (0-1)
zp = 1;                                        %zero padding
% take stft of input
raw_stft = stft(Hyd_O, Window=mywindow, OverlapLength=overlap, FFTLength=zp*framelen);  %↙
calculate stft
one_sided_stft = raw_stft(end/2+1:end, :);      %only include positive frequencies
mag_stft = abs(one_sided_stft);       %take magnitude and log scale it
dB_stft = 20*log10(mag_stft);
t_clip = (1:length(Hyd_O))/fs;
f_clip = 0:fs/(zp*framelen):fs/2;
imagesc(t_clip, f_clip, dB_stft)
set(gca,'YDir','normal')
xlabel("time (s)")
ylabel("frequency (Hz)")
title("Spectrogram of Hydrophone O")
% Calculate and Plot "Signal Energy" as it Changes Through Time
number_frames = size(dB_stft, 2);
signal_energy = zeros(1, number_frames);
% From the STFT, it is clear that the breathing rate corresponds with a
% burst of broad band noise
% This is a coarse way to extract this trend, which also is affected by how close or far↙
the diver is:
for frame = 1:number_frames
   % sum over each frame and add up the magnitudes of the upper half of
   % frequencies (bin 128 - 256)
   signal_energy(frame) = sum(dB_stft(:, frame));
end
plot(signal_energy)
ylabel("High Frequency Energy")
xlabel("Time (STFT Frames)")
title("dB Signal 'Energy' vs Time")
% Run FFT of this signal to identify the periodicity at < 1 Hz
% all breathing rate parameters given _b or _breath
fs_b = fs*number_frames/length(Hyd_O);  % This sampling rate correspon
win_b = hanning(number_frames)';
windowed_breathing = win_b.*signal_energy;
zp_b = 100;
Nfft_b = zp_b*length(signal_energy);
breathing_fft = 20*log10(abs(fft(windowed_breathing, Nfft_b)));
f_breath = linspace(0, fs_b, length(breathing_fft));
plot(f_breath, breathing_fft)
xlabel("frequency (Hz)")
ylabel("Magnitude (dB)")
% We really only care about the super low frequency periodicity in this
% signal.
plot(f_breath(1:100*zp_b), breathing_fft(1:100*zp_b))
xlabel("frequency (Hz)")
ylabel("Magnitude (dB)")
title("Magnitude Spectrum of dB Signal Energy in Hydrophone O Recording")
hold on
```

```
[peaks, locs] = findpeaks(breathing_fft(1:100*zp_b));
[biggest_peak, biggest_peak_location] = max(peaks);
breathing_rate = f_breath(locs(biggest_peak_location));
scatter(f_breath(locs(biggest_peak_location)), biggest_peak)
disp("The breathing rate of the diver is " + breathing_rate + " Hz")
text(.16, 164, breathing_rate + " Hz")

%% Part 2a- Method 1

% Load audio files for hydrophones O, N, and P
[y_o, fs] = audioread('SCP23_Hyd O.wav');
[y_n, ~] = audioread('SCP23_ Hyd N.wav');
[y_p, ~] = audioread('SCP23_Hyd P.wav');

% Set parameters for spectrogram
window_size = 1024;
hop_size = 512;

% Generate spectrograms for hydrophones O, N, and P
[S_o, F_o, T_o] = spectrogram(y_o, hamming(window_size), window_size-hop_size,↙
window_size, fs, 'yaxis');
[S_n, ~, T_n] = spectrogram(y_n, hamming(window_size), window_size-hop_size, window_size,↙
fs, 'yaxis');
[S_p, ~, T_p] = spectrogram(y_p, hamming(window_size), window_size-hop_size, window_size,↙
fs, 'yaxis');

% Convert power spectrograms to dB scale
S_dB_o = 10*log10(abs(S_o));
S_dB_n = 10*log10(abs(S_n));
S_dB_p = 10*log10(abs(S_p));

% Find the frequency bins that correspond to the freq range of interest
f_bins = find(F_o >= 20000 & F_o <= 125000);

% Extract the spectral energy in the range of interest for each hydrophone
E_o = sum(S_dB_o(f_bins, :), 1);
E_n = sum(S_dB_n(f_bins, :), 1);
E_p = sum(S_dB_p(f_bins, :), 1);

% Find the time index of the peak energy in the spectrogram for hydrophone O
[~, idx] = max(E_o);
t_closest = T_o(idx);

% Print the time when the diver is at the closest point of approach to the middle↙
hydrophone O
fprintf('Time of closest point of approach to hydrophone O: %.2f seconds\n', t_closest);

%% Task 2a- Method 2

[O_Mic,Fs2]= audioread('SCP23_Hyd O.wav');
framelen = 2048;
window = hamming(framelen);
overlap = 0;
t_frame2 = framelen/Fs2;
f_res2 = Fs2/framelen;
ST_O = spectrogram(O_Mic,window,overlap);
[nBins2,nFr2] = size(ST_O);
```

```
total_energy_O = zeros(1,nFr2);
for n = 1:nFr2
   start = (n*framelen) - (framelen-1);
   stop = n*framelen;
   frame_data = O_Mic(start:stop);
   tot_e = sum(frame_data.^2);
   total_energy_O(n) = tot_e;
end

te_fft = fft(total_energy_O);
Fs_new = Fs2/framelen;
freqs = (1:length(total_energy_O))*(Fs_new/nFr2);

ST_O_dB = 20*log10(abs(ST_O));
[nBins2,nFr2] = size(ST_O);
f_bins_loc2 = (0:nBins2-1)*f_res2;
frame_center2 = (t_frame2/2) + (1:nFr2);

[pks,locs,w,p] = findpeaks(20*log10(abs(te_fft)));
[pk,ind] = max(pks);
in = locs(ind);
breath_rate = freqs(in);

[pks_O,locs_O,w_O,p_O] = findpeaks(total_energy_O);
[max_p_O,ind_O] = max(p_O);
[max_pk_O,ind_pk_O] = max(pks_O);
peak_time_O = frame_center2(ind_O);
time_at_mic_O = (locs_O(ind_O)*framelen)/Fs2;

% Print the time when the diver is at the closest point of approach to the middle↙
hydrophone O
fprintf('Time of closest point of approach to hydrophone O: %.2f seconds\n',↙
time_at_mic_O);

%% Task 2b

% Finding difference in TOA between hydrophones visually
[audio0, Fs] = audioread('SCP23_Hyd O.wav');
audio_smaller0 = audio0(50000000:52000000);
[audioP, Fs] = audioread('SCP23_Hyd P.wav');
audio_smallerP = audioP(50000000:52000000);
[audioN, Fs] = audioread('SCP23_ Hyd N.wav');
audio_smallerN = audioN(50000000:52000000);
hold on
plot((50000000:52000000), audio_smaller0, 'r')
plot((50000000:52000000), audio_smallerP, 'g')
plot((50000000:52000000), audio_smallerN, 'b')
legend('Hydrophone 0','Hydrophone P', 'Hydrophone N')
title('Hydrophone Data')
xlabel('Samples')
ylabel('Amplitude')

% Using xcorr to find time delay between hydrophones
time_idx = floor(204.08*Fs);
[c,lags] = xcorr(audio0(time_idx-Fs:time_idx+Fs),audioP(time_idx-Fs:time_idx+Fs));
[max_c,ind_c] = max(c);
delay = lags(ind_c);
delay_t = delay/Fs;
```

```
delta_t = abs(delay_t);

d = 14; % Inter-element spacing of hydrophones (m)
c = 1520; % Speed of sound in water (m/s)

% Use derived equation with knowns and difference in TOA
h_diver = (1/2)*((d^2/(c * delta_t)) - (c * delta_t));

% Display the estimated diver's altitude
fprintf('Diver''s altitude above Hydrophone O: %.2f meters\n', h_diver);

%% Part 2c- Method 1

% Distance between each hydrophone (m)
d = 14;

% Load audio files for hydrophones O, N, and P
[y_o, fs] = audioread('SCP23_Hyd O.wav');
[y_n, ~] = audioread('SCP23_ Hyd N.wav');
[y_p, ~] = audioread('SCP23_Hyd P.wav');

% Set parameters for spectrogram
window_size = 1024;
hop_size = 512;

% Generate spectrograms for hydrophones O, N, and P
[S_o, F_o, T_o] = spectrogram(y_o, hamming(window_size), window_size-hop_size,↙
window_size, fs, 'yaxis');
[S_n, ~, T_n] = spectrogram(y_n, hamming(window_size), window_size-hop_size, window_size,↙
fs, 'yaxis');
[S_p, ~, T_p] = spectrogram(y_p, hamming(window_size), window_size-hop_size, window_size,↙
fs, 'yaxis');

% Convert power spectrograms to dB scale
S_dB_o = 10*log10(abs(S_o));
S_dB_n = 10*log10(abs(S_n));
S_dB_p = 10*log10(abs(S_p));

% Find the frequency bins that correspond to the freq range of interest
f_bins = find(F_o >= 40000 & F_o <= 125000);

% Extract the spectral energy in the range of interest for each hydrophone
E_o = sum(S_dB_o(f_bins, :), 1);
E_n = sum(S_dB_n(f_bins, :), 1);
E_p = sum(S_dB_p(f_bins, :), 1);

% Find the time index of the peak energy in the spectrogram for hydrophone O
[~, idxo] = max(E_o);

t_closest_o = T_o(idxo);

% Find the time index of the peak energy in the spectrogram for hyd
% P
[~, idxp] = max(E_p);
t_closest_p = T_p(idxp);

% Print the time when the diver is at the closest point of approach to hydrophone N
fprintf('Time of closest point of approach to hydrophone N: %.2f seconds\n',↙
```

```
t_closest_o);

% Print the time when the diver is at the closest point of approach tohydrophone P
fprintf('Time of closest point of approach to hydrophone P: %.2f seconds\n',↙
t_closest_p);

t_diff_op = t_closest_o - t_closest_p;

v_op = d / abs(t_diff_op);

% Print the velocity of the diver
fprintf('Velocity: %.2f m/s \n', v_op);

% Repeat using Hyd N to check for consistency

%% Task 2c- Method 2

% Load audio files for hydrophones O, N, and P
[N_Mic, ~] = audioread('SCP23_ Hyd N.wav');
[O_Mic, Fs1] = audioread('SCP23_Hyd O.wav');
[P_Mic, ~] = audioread('SCP23_Hyd P.wav');

framelen = 2048;
window = hamming(framelen);
overlap = 0;

ST_O = spectrogram(O_Mic,window,overlap);
ST_P = stft(P_Mic,'Window',window','OverlapLength',overlap);
ST_N = stft(N_Mic,'Window',window','OverlapLength',overlap);

f_res1 = Fs1/framelen;
f_res2 = Fs1/framelen;
f_res3 = Fs1/framelen;

t_frame1 = framelen/Fs1;
t_frame2 = framelen/Fs1;
t_frame3 = framelen/Fs1;

ST_N_dB = 20*log10(abs(ST_N));
[nBins,nFr] = size(ST_N);
f_bins_loc = (0:nBins-1)*f_res2;
frame_center1 = (t_frame1/2) + (1:nFr);

ST_P_dB = 20*log10(abs(ST_P));
[nBins3,nFr3] = size(ST_P);
f_bins_loc3 = (0:nBins3-1)*f_res3;
frame_center3 = (t_frame3/2) + (1:nFr3);

total_energy_N = zeros(1,nFr);
for n = 1:nFr
   start = (n*framelen) - (framelen-1);
   stop = n*framelen;
   frame_data = N_Mic(start:stop);
   tot_e = sum(frame_data.^2);
   total_energy_N(n) = tot_e;
end

total_energy_P = zeros(1,nFr3);
```

```
for n = 1:nFr3
   start = (n*framelen) - (framelen-1);
   stop = n*framelen;
   frame_data = P_Mic(start:stop);
   tot_e = sum(frame_data.^2);
   total_energy_P(n) = tot_e;
end

[pks_N,locs_N,w_N,p_N] = findpeaks(total_energy_N);
[max_p_N,ind_N] = max(p_N);
peak_time_N = frame_center1(ind_N);
time_at_mic_N = (locs_N(ind_N)*framelen)/Fs1;

[pks_P,locs_P,w_P,p_P] = findpeaks(total_energy_P);
[max_p_P,ind_P] = max(p_P);
peak_time_P = frame_center1(ind_P);
time_at_mic_P = (locs_P(ind_P)*framelen)/Fs1;

diver_speed = (2*14)/(time_at_mic_P-time_at_mic_N);
divers_NO = 14/(time_at_mic_O-time_at_mic_N);
divers_OP = 14/(time_at_mic_P-time_at_mic_O);
avg_speed = (divers_NO+divers_OP)/2;

% Print the velocity of the diver
fprintf('Velocity: %.2f m/s \n', avg_speed);

%% Task 2a and c- Method 3

clear
close all
[Hyd_N, fs] = audioread("SCP23_ Hyd N.wav");
Hyd_O = audioread("SCP23_Hyd O.wav");
Hyd_P = audioread("SCP23_Hyd P.wav");
% Calculate signal energy above 20kHz
f_hp = 70000; % cutoff at 20kHz
[b_hp, a_hp] = butter(2, 2*f_hp/fs, 'high');
Hyd_N_hi = filter(b_hp, a_hp, Hyd_N);
Hyd_O_hi = filter(b_hp, a_hp, Hyd_O);
Hyd_P_hi = filter(b_hp, a_hp, Hyd_P);
% Set up frames for energy calculation
framelen = 2500;
hop = framelen/10;
nframes = floor(length(Hyd_O)/hop);
% Initialize energy vector, indices
N_energy = zeros(1, nframes);
O_energy = zeros(1, nframes);
P_energy = zeros(1, nframes);
start_samp = 1;
end_samp = framelen;
% Loop through each frame and calculate energy
for frame = 1:nframes-10
   N_energy(frame) = sum(Hyd_N_hi(start_samp:end_samp).^2);
   O_energy(frame) = sum(Hyd_O_hi(start_samp:end_samp).^2);
   P_energy(frame) = sum(Hyd_P_hi(start_samp:end_samp).^2);
   start_samp = start_samp + hop;
   end_samp = end_samp + hop;
end
% Plots
```

```
% Turn frames into time (set to be center of frame)
t = (1:nframes)*hop/fs - framelen/(fs*2);
plot(t,O_energy)
hold on
% Find peaks in energy
[peaks_O, locs_O] = findpeaks(O_energy, 'MinPeakDistance', 4/hop*fs);
scatter(t(locs_O), peaks_O)
xlabel("time (s)")
ylabel("Signal Energy")
title("Signal Energy of Hydrophone O")
hold off
pause
% Set up time interpolation vector
time_interp = 1:.0001:max(t);
% Interpolate from peaks
peak_interp_O = interp1(t(locs_O), peaks_O, time_interp, 'spline');
plot(time_interp, peak_interp_O)
hold on
% Find peak of interpolated peak function (I know that's confusing)
[max_peak_O, loc_peak_O] = findpeaks(peak_interp_O, 'MinPeakProminence', 12);
scatter(time_interp(loc_peak_O), max_peak_O)
xlabel("time (s)")
ylabel("Signal Energy")
title("Interpolated Signal Energy of Hydrophone O")
hold off
O_time_peak = time_interp(loc_peak_O);
disp("The time nearest to O is: " + O_time_peak)
pause
plot(t,N_energy)
hold on
% Find peaks in energy
[peaks_N, locs_N] = findpeaks(N_energy, 'MinPeakDistance', 4/hop*fs);
scatter(t(locs_N), peaks_N)
xlabel("time (s)")
ylabel("Signal Energy")
title("Signal Energy of Hydrophone N")
hold off
pause
% Interpolate from peaks
peak_interp_N = interp1(t(locs_N), peaks_N, time_interp, 'spline');
plot(time_interp, peak_interp_N)
hold on
% Find peak of interpolated peak function (I know that's confusing)
[max_peak_N, loc_peak_N] = findpeaks(peak_interp_N, 'MinPeakProminence', 15);
scatter(time_interp(loc_peak_N), max_peak_N)
xlabel("time (s)")
ylabel("Signal Energy")
title("Interpolated Signal Energy of Hydrophone N")
hold off
N_time_peak = time_interp(loc_peak_N);
disp("The time nearest to N is: " + N_time_peak)
pause
plot(t,P_energy)
hold on
% Find peaks in energy
[peaks_P, locs_P] = findpeaks(P_energy, 'MinPeakDistance', 4/hop*fs);
scatter(t(locs_P), peaks_P)
xlabel("time (s)")
```

```
ylabel("Signal Energy")
title("Signal Energy of Hydrophone P")
hold off
pause
% Interpolate from peaks
peak_interp_P = interp1(t(locs_P), peaks_P, time_interp, 'spline');
plot(time_interp, peak_interp_P)
hold on
% Find peak of interpolated peak function (I know that's confusing)
[max_peak_P, loc_peak_P] = findpeaks(peak_interp_P, 'MinPeakProminence', 12);
scatter(time_interp(loc_peak_P), max_peak_P)
xlabel("time (s)")
ylabel("Signal Energy")
title("Interpolated Signal Energy of Hydrophone P")
hold off
P_time_peak = time_interp(loc_peak_P);
disp("The time nearest to P is: " + P_time_peak)
% Part 2c Velocity Calculation
d_NO = 14;
d_OP = 14;
d_NP = 28;
v_NO = d_NO/(O_time_peak - N_time_peak);
v_OP = d_OP/(P_time_peak - O_time_peak);
v_NP = d_NP/(P_time_peak - N_time_peak);
```