Lab 4

Rajdeep Gill (7934493) & Daniyal Hasnain (7942244)

ECE 4830 B03

March 16, 2025

Contents

1 Problem 1

There is a distinct noise when playing the audio overlapping the tune. We can use a simple mask to eliminate this single sinusoidal tone. First, we plot the FFT before any filtering to detect the noise frequency. Since the audio is stereo, we can work with just the left side (as they are near-identical). The plot below shows the magnitude spectrum before filtering:

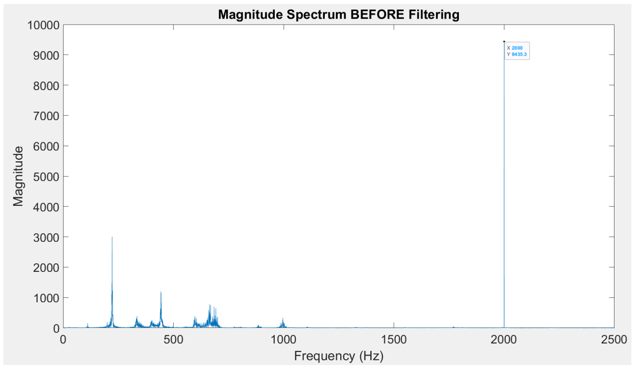

Figure 1: Magnitude spectrum of the given audio's left side before filtering.

A quick scan shows a distinct peak at 2000 Hz—the noise we seek to remove. A straightforward notch filter implemented as a binary mask will do the trick; we set the mask to an array of ones except at the index that corresponds to 2000 Hz, where we set zeroes. Multiplying the data vector with this mask will effectively remove the noise. To get the bin index for the observed noise, we use:

$$\text{Index} = \frac{\text{Noise Frequency} \times N}{F_s} = \frac{2000 \times 132300}{44100} = \mathbf{6000}$$

To account for small numerical errors in rounding and/or leakage, we can set a small margin of error around this index to set to zero (5995-6005 is sufficient). The snippet below shows the filter used in Matlab code:

```matlab
% notch filter
mask = ones(N, 1);
notch_range = 5;  % small margin for errors
% Remove the detected frequency with its margin
mask(index-notch_range:index+notch_range) = 0; % set 5995-6005 to zero
mask(N-(index+notch_range):N-(index-notch_range)) = 0; % symmetric pair
```

Code Snippet 1: Notch Filter

Multiplying the left and right sides by the mask, taking the inverse FFT (ifft), and plotting the combined waveform gave the following wave seen in Figure 2. As we can see, the noise-peak at 2000 Hz is now gone, leaving us with the filtered audio. Listening to the output confirms the result. At this point, we used only the FFT and designed a simple notch filter (binary mask) to eliminate the single sinusoidal tone successfully.

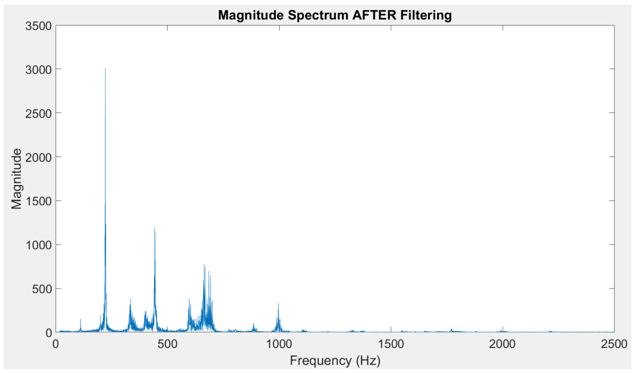

Figure 2: Magnitude spectrum of the given audio's left side after filtering.

2 Problem 2

To remove the water interaction with the hydrophone, we first find the STFT (Short-Time Fourier Transform) of the audio. We can then try to filter out all the noise by using an threshold filter. The STFT plotted in the decibel scale is shown in Figure 3.

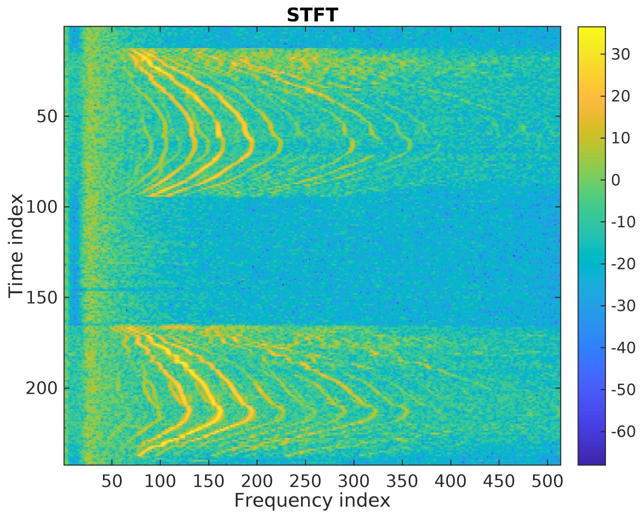

Figure 3: STFT of the given audio before filtering.

We notice that the values below 6 are mostly noise and the most prominent waveforms have a much higher ampltitude. We can use this information to filter out the noise. The code snippet below shows the filtering process:

```
    [audio, fs] = audioread("Globicefalo_clip.wav");
    fft_length = 1024;        % FFT length
    win_length = 512;         % Window length
    time_res = 512;           % Time resolution (hop size)
    X = STFT(audio, fft_length, win_length, time_res);
    thresh = 6;
    X_mag = abs(X);
    X_filtered = X;
    X_filtered(X_mag < thresh) = 0;
```

Code Snippet 2: Threshold Filter

This is a simple binary threshold filter that sets all values below 6 to zero, the plot of the filter is shown in Figure 4.

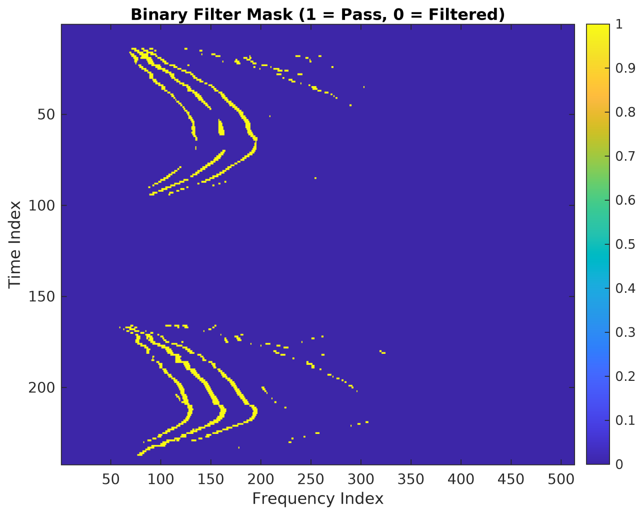

Figure 4: Threshold filter used to filter out the noise.

The STFT of the audio after filtering is shown in Figure 5.

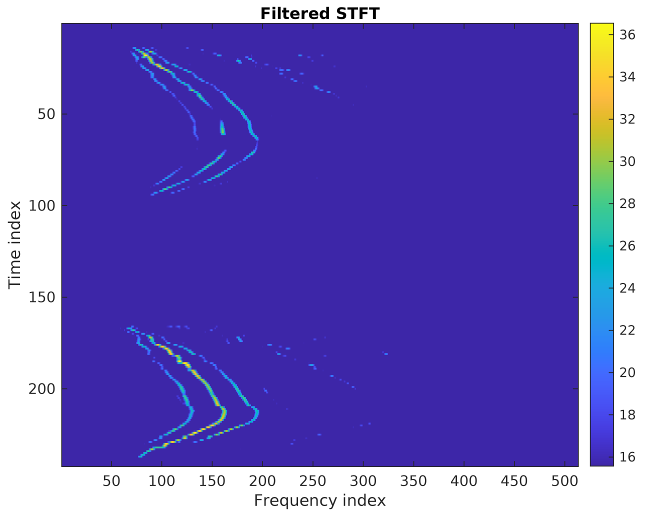

Figure 5: STFT of the given audio after filtering.

Listening to both audios, we see that the filtered audio has no water-interaction noise and only contains the sound of the sea-creature.