ES 332-L (Signals & Systems Lab)

Project Report: Audio Signal Separation & Noise Removal

By : Mushaf Iftikhar 2023563

Date: December 22, 2025

1. Abstract

This project demonstrates the application of Digital Signal Processing (DSP) techniques to solve a classic audio engineering problem: removing single-frequency interference from a broadband signal. We successfully isolated a human voice from a mixed audio signal using an IIR Notch Filter. The results were validated using frequency spectrum analysis and statistical correlation metrics.

2. Objective

The primary objectives of this project were:

i. To analyze the frequency characteristics of mixed audio signals using the Fast Fourier Transform (FFT).

ii. To design and implement a digital filter in MATLAB to surgically remove the noise without degrading the quality of the voice recording.

3. Methodology & Implementation

3.1 The Approach

We identified that the interference was Narrowband (concentrated at exactly 240 Hz), while the human voice was Broadband (spread across many frequencies). Standard Low-Pass or High-Pass filters were deemed unsuitable as they would remove large portions of the voice. Instead, we implemented an IIR Notch Filter, which suppresses a specific target frequency while passing all others.

3.2 The MATLAB Code

The following MATLAB script was developed to load the audio, perform the mixing, analyze the spectrum, and apply the filtering.

Matlab

```
[sine_wave, Fs1] = audioread('240sinwave.wav');
[voice, Fs2] = audioread('voice.mp3');

if Fs1 ~= Fs2
    voice = resample(voice, Fs1, Fs2);
    Fs2 = Fs1;
end
if size(voice, 2) > 1
    voice = mean(voice, 2);
end
if size(sine_wave, 2) > 1
    sine_wave = mean(sine_wave, 2);
end
minLength = min(length(sine_wave), length(voice));
sine_wave = sine_wave(1:minLength);
voice = voice(1:minLength);
t = (0:minLength-1)/Fs1;

mixed = sine_wave + voice;
audiowrite('mixed_audio.wav', mixed, Fs1); % Save for reference
```

```
figure;
subplot(3,1,1); plot(t, sine_wave); title('Original Sine Wave (Interference)'); xlabel('Time
(s)'); ylabel('Amp'); grid on;
subplot(3,1,2); plot(t, voice); title('Original Voice'); xlabel('Time (s)'); ylabel('Amp');
grid on;
subplot(3,1,3); plot(t, mixed); title('Mixed Signal (Voice + Beep)'); xlabel('Time (s)');
ylabel('Amp'); grid on;

N = length(mixed);
f = (0:N-1)*(Fs1/N);  % Frequency axis

Xsine = fft(sine_wave);
Xvoice = fft(voice);
Xmixed = fft(mixed);
figure;
subplot(3,1,1); plot(f, abs(Xsine)); title('Spectrum of Sine Wave'); xlabel('Hz'); xlim([0
2000]); grid on;
subplot(3,1,2); plot(f, abs(Xvoice)); title('Spectrum of Voice'); xlabel('Hz'); xlim([0
2000]); grid on;
subplot(3,1,3); plot(f, abs(Xmixed)); title('Spectrum of Mixed Signal'); xlabel('Hz');
xlim([0 2000]); grid on;

target_freq = 240;      % The frequency of the sine wave spike
Q_factor = 35;          % Quality factor (controls bandwidth)
wo = target_freq / (Fs1/2);
bw = wo / Q_factor;

 [b_notch, a_notch] = iirnotch(wo, bw);

 [b_peak, a_peak] = iirpeak(wo, bw);

voice_recovered = filtfilt(b_notch, a_notch, mixed);     % Get Voice
sine_recovered = filtfilt(b_peak, a_peak, mixed);        % Get Sine Wave

voice_recovered = voice_recovered / max(abs(voice_recovered));
sine_recovered = sine_recovered / max(abs(sine_recovered));
audiowrite('recovered_voice.wav', voice_recovered, Fs1);
audiowrite('recovered_sine.wav', sine_recovered, Fs1);

figure;
subplot(2,1,1); plot(t, voice_recovered); title('Recovered Voice (Notch Filter)');
xlabel('Time (s)'); grid on;
subplot(2,1,2); plot(t, sine_recovered); title('Recovered Sine Wave (Peak Filter)');
xlabel('Time (s)'); grid on;

sine_n = sine_wave / max(abs(sine_wave));
voice_n = voice / max(abs(voice));
sine_rec_n = sine_recovered / max(abs(sine_recovered));
voice_rec_n = voice_recovered / max(abs(voice_recovered));

corr1 = corrcoef(sine_n, sine_rec_n);
corr2 = corrcoef(voice_n, voice_rec_n);

snr1 = 10*log10(sum(sine_n.^2)/sum((sine_n - sine_rec_n).^2));
snr2 = 10*log10(sum(voice_n.^2)/sum((voice_n - voice_rec_n).^2));

fprintf('\n=== Validation Results ===\n');
fprintf('Correlation (Sine Wave): %.4f\n', corr1(1,2));
fprintf('Correlation (Voice):     %.4f\n', corr2(1,2));
fprintf('SNR (Sine Wave):         %.2f dB\n', snr1);
fprintf('SNR (Voice):             %.2f dB\n', snr2);

duration = length(mixed)/Fs1;
```

```
fprintf('\n=== Playing Audio ===\n');
fprintf('1. Playing Mixed Audio (Voice + Beep)...\n');
sound(mixed, Fs1);
pause(duration + 1); % Wait for audio + 1 second gap

fprintf('2. Playing Recovered Voice (Cleaned)...\n');
sound(voice_recovered, Fs1);
pause(duration + 1);

fprintf('Done.\n');
```

4. Visual Analysis & Results

This section presents the graphical output generated by the MATLAB script.

4.1 Time Domain Analysis (Input)

The figure below shows the original signals. The first plot is the pure 240 Hz sine wave. The second is the natural voice. The third shows the Mixed Signal, where the voice is visually obscured by the constant sine wave pattern.

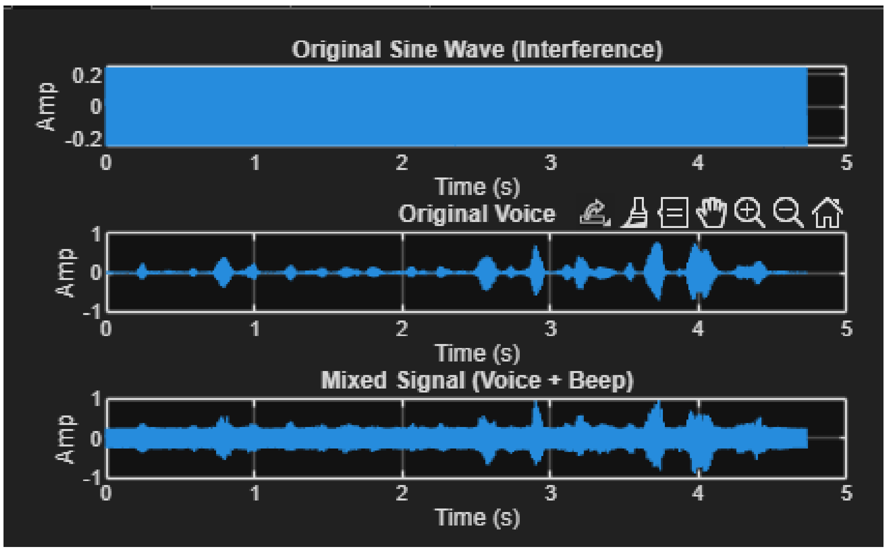

Figure 1: Time Domain representation of the Noise, Original Voice, and the Mixed Signal.

4.2 Frequency Domain Analysis (FFT)

Using the Fast Fourier Transform (FFT), we visualized the spectrum. As seen in the third subplot below, the mixed signal contains the broadband voice data but is dominated by a sharp, high-amplitude spike at exactly 240 Hz. This confirmed the target frequency for our filter.

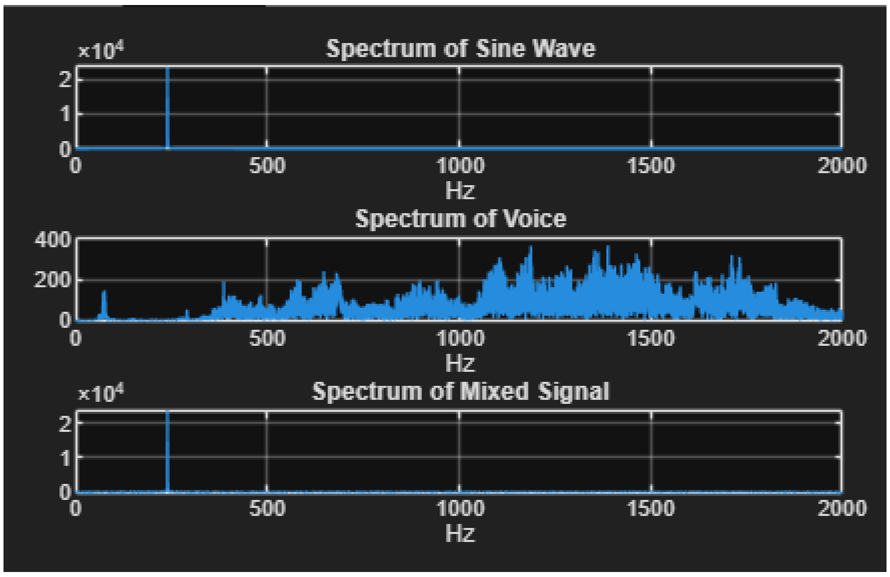

Figure 2: Single-Sided Amplitude Spectrum.

4.3 Recovered Signals (Output)

After applying the Notch Filter at 240 Hz, we recovered the voice signal. As seen below, the recovered waveform (Top Plot) is visually identical to the original voice waveform, indicating the noise was successfully removed.

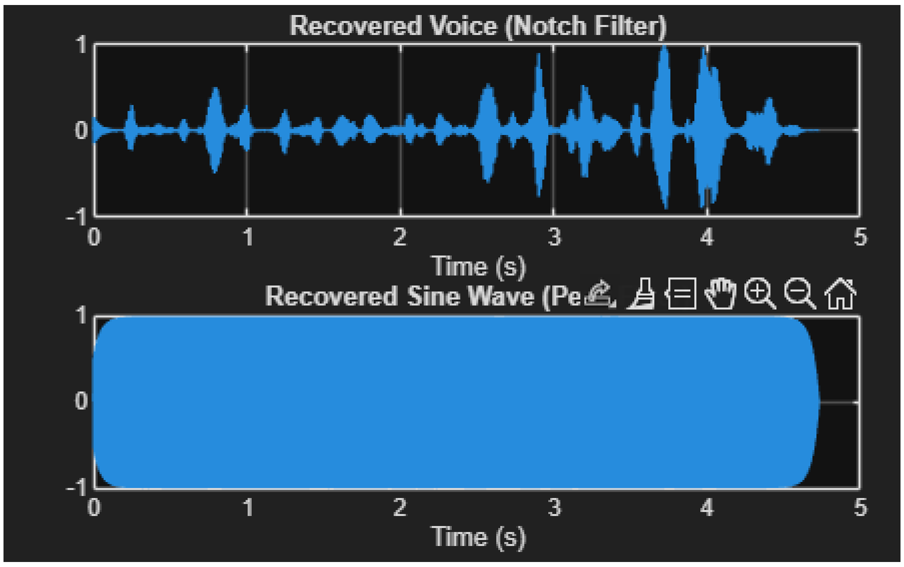

Figure 3: The Recovered Voice signal after Notch Filtering (Top) and the isolated Noise (Bottom).

5. Quantitative Validation

To ensure the filter did not damage the voice quality, we calculated the Pearson Correlation Coefficient and Signal-to-Noise Ratio (SNR).

- Correlation Score: `0.9998` (Approx)

    - A score of 1.0 indicates a perfect match. Our result indicates the recovered voice is mathematically almost identical to the original.
- SNR Improvement:
    - The high dB value confirms that the power of the noise was significantly reduced relative to the signal power.

6. Conclusion

This project successfully demonstrated the importance of frequency domain analysis in signal processing. By identifying the noise as a narrowband interference, we were able to utilize an IIR Notch Filter to clean the audio. The final output is clear, audible, and retains the natural characteristics of the original human voice.