Filtering Noisy Audio

Signals & Systems Lab Project

Prepared By:

Ali Asghar (21PWCSE2059)

Suleman Shah (21PWCSE1983)

Shahzad Bangash (21PWCSE1980)

Overview:

Introduction:

The project aims to analyze and process a noisy audio signal using filtering techniques to attenuate unwanted noise and improve audio quality. It involves exploring

- frequency characteristics
- designing filters
- analyzing their response
- and evaluating filtered audio outputs.

Problem statement:

This project addresses the issue of noise in audio signals, which diminishes quality and impacts the listening experience. The objective is to use filtering techniques to eliminate or minimize noise and improve the overall audio quality.

Project Source:

The idea of the project is taken from the github account of **alirezajaberirad[1].**

Tools Used:

- **MATLAB[2]:** A software tool widely used for processing and analyzing discrete data, including signal processing applications and computational mathematics.
- **Fourier transform[3]:** Used to decompose data represented as a function of time or space into frequency components.
- **fft function[4][5]:** Implements a fast Fourier transform algorithm in MATLAB, reducing computational cost compared to other direct implementations.
- **conv and filter functions:** Useful tools for modifying the amplitude or phase of input data using a transfer function.

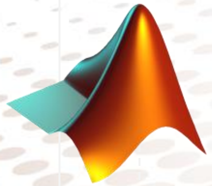


Figure 1,
MATLAB Logo

Methodology:

The project methodology comprises the following steps:

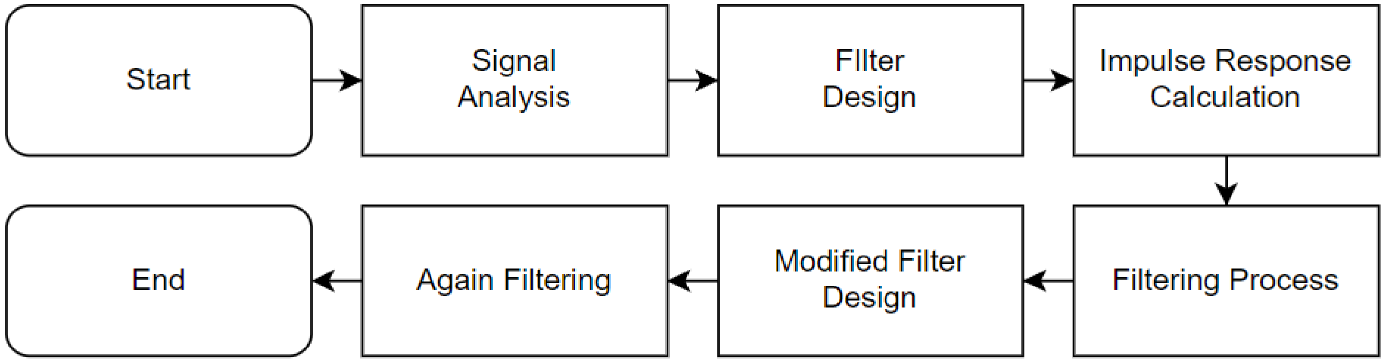
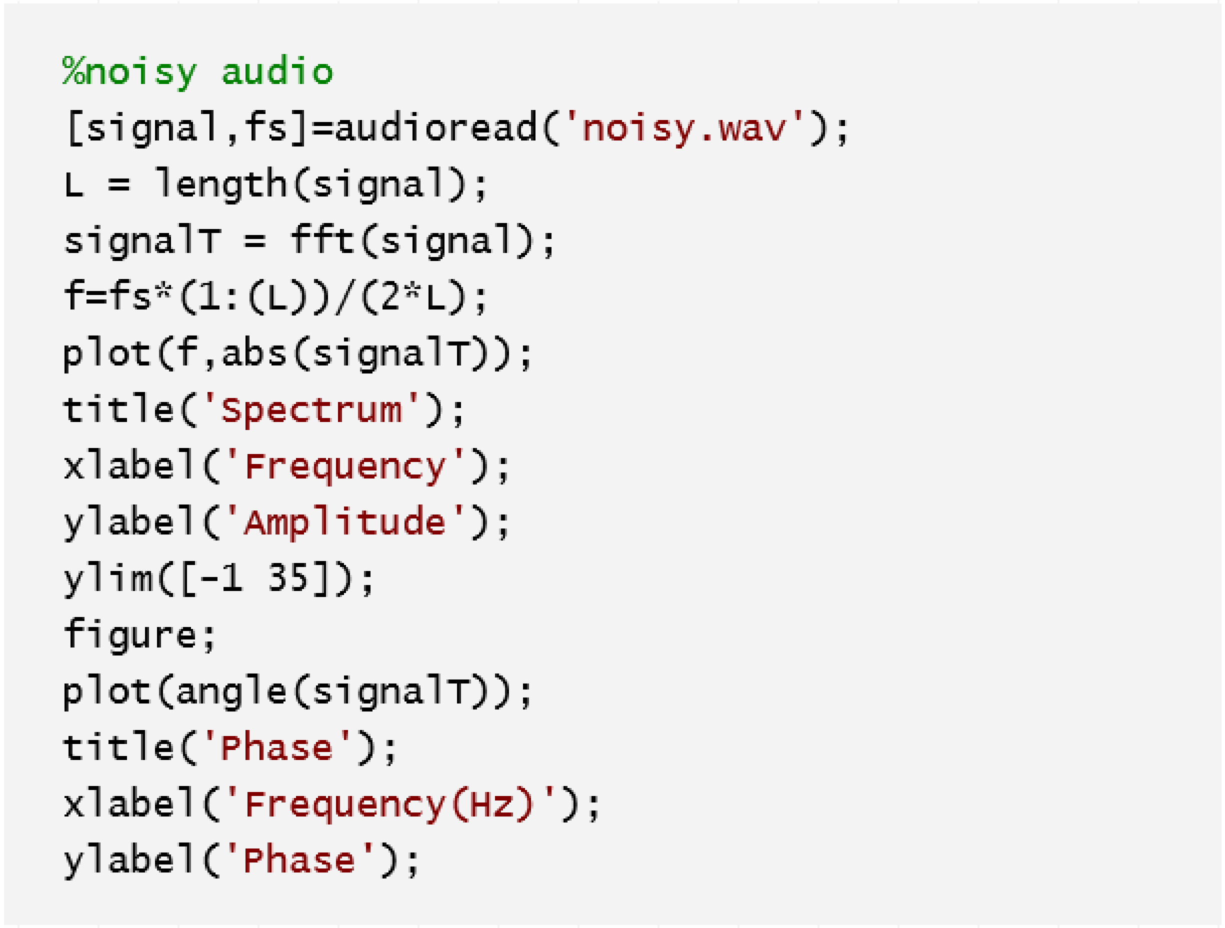


Figure 2, Flowchart of Program

Step 1, Signal Analysis:

- Read the noisy audio signal and extract its sampling frequency.
- Apply the Fast Fourier Transform (FFT) to analyze the frequency content and phase information of the signal.
- Visualize the magnitude spectrum and phase spectrum of the signal.

Step 1, Signal Analysis (Cont.): Code:



```
%noisy audio
[signal,fs]=audioread('noisy.wav');
L = length(signal);
signalT = fft(signal);
f=fs*(1:(L))/(2*L);
plot(f,abs(signalT));
title('Spectrum');
xlabel('Frequency');
ylabel('Amplitude');
ylim([-1 35]);
figure;
plot(angle(signalT));
title('Phase');
xlabel('Frequency(Hz)');
ylabel('Phase');
```

Figure 3.1, Signal Analysis Section Code

Step 1, Signal Analysis (Cont.): Output:

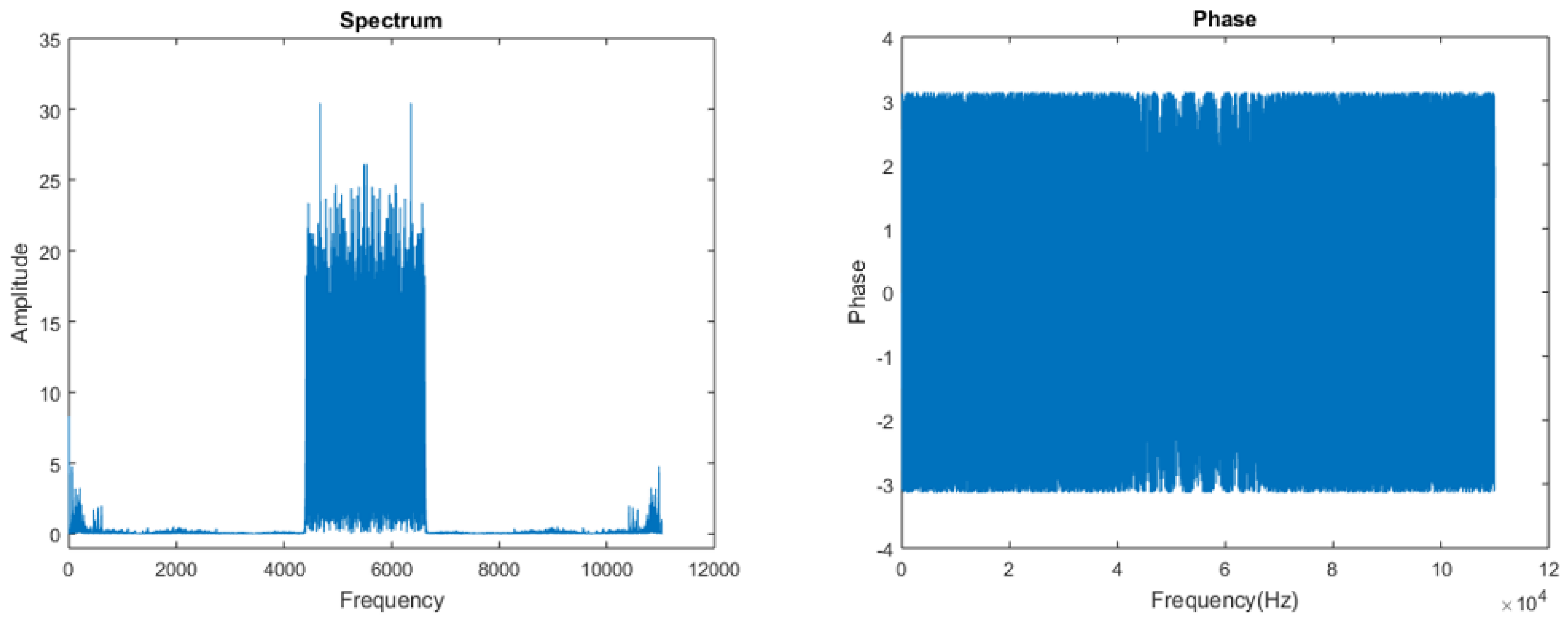
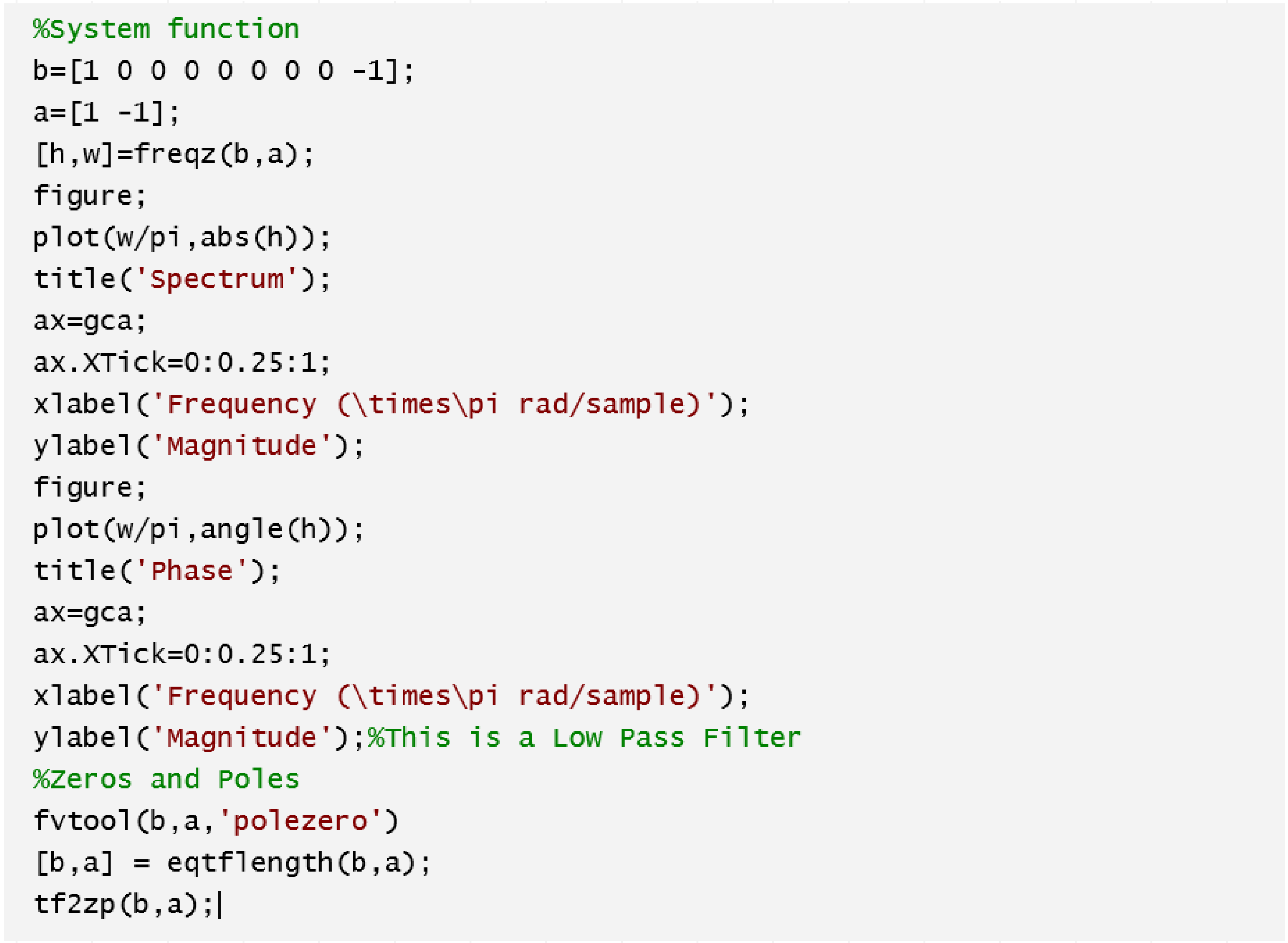


Figure 3.2, Spectrum and Phase Plot of the signal

Step 2, Filter Design:

- Design an initial filter by specifying the filter coefficients.
- Compute the frequency response of the initial filter using the ‘freqz’ function.
- Plot the magnitude and phase spectra of the filter's frequency response.
- Visualize the poles and zeros of the filter using the ‘fvtool’ function.

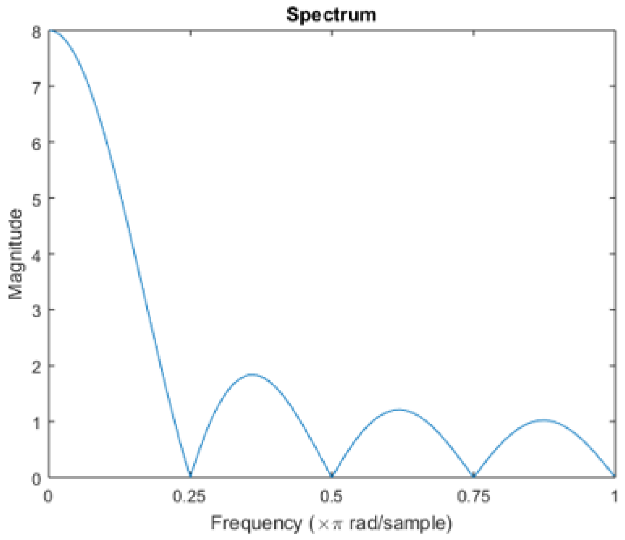
Step 2, Filter Design (Cont.): Code:

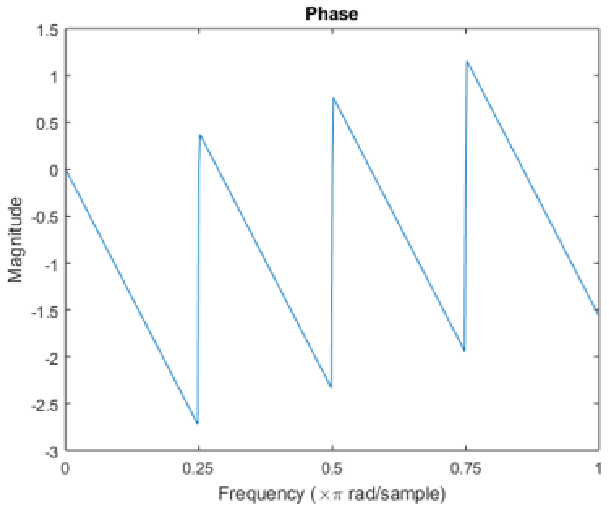


```
%System function
b=[1 0 0 0 0 0 0 0 -1];
a=[1 -1];
[h,w]=freqz(b,a);
figure;
plot(w/pi,abs(h));
title('Spectrum');
ax=gca;
ax.XTick=0:0.25:1;
xlabel('Frequency (\times\pi rad/sample)');
ylabel('Magnitude');
figure;
plot(w/pi,angle(h));
title('Phase');
ax=gca;
ax.XTick=0:0.25:1;
xlabel('Frequency (\times\pi rad/sample)');
ylabel('Magnitude');%This is a Low Pass Filter
%Zeros and Poles
fvtool(b,a,'polezero')
[b,a] = eqtflength(b,a);
tf2zp(b,a);
```

Figure 4, System Design Section Code

Step 2, Filter Design (Cont.): Output:





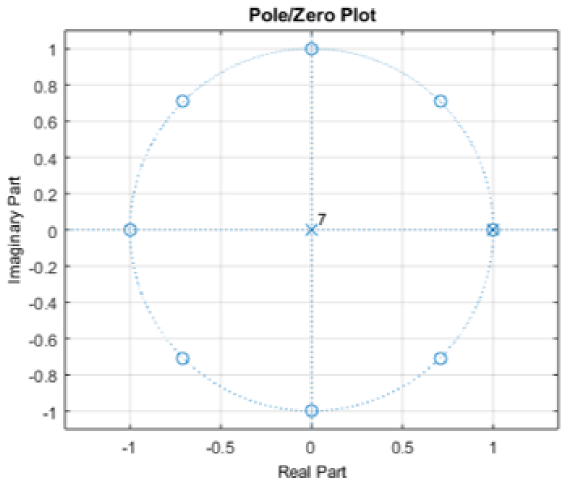


Figure 5, Spectrum, Phase, poles and zeros plot

Step 3,
Impulse Response Calculation:

- Calculate the impulse response of the initial filter using the ‘impz’ function.
- Plot the impulse response to analyze its characteristics and determine its length.

Step 3 Impulse Response Calculation (Cont.): Code and Output:

```
%Impulse response
figure;
[impResp,t] = impz(b,a);
stem(t,impResp);
title('Impulse Response')
```

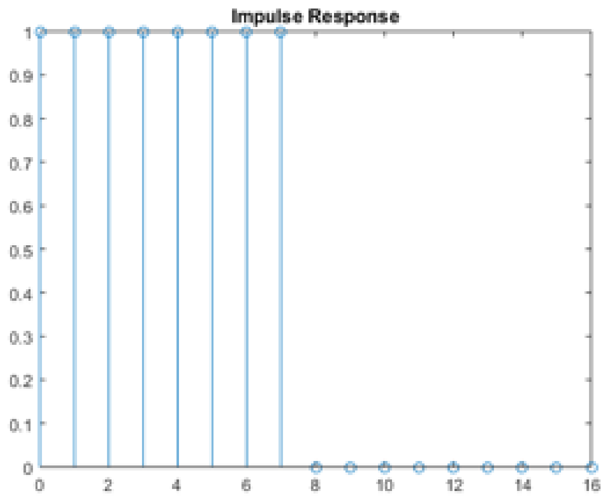
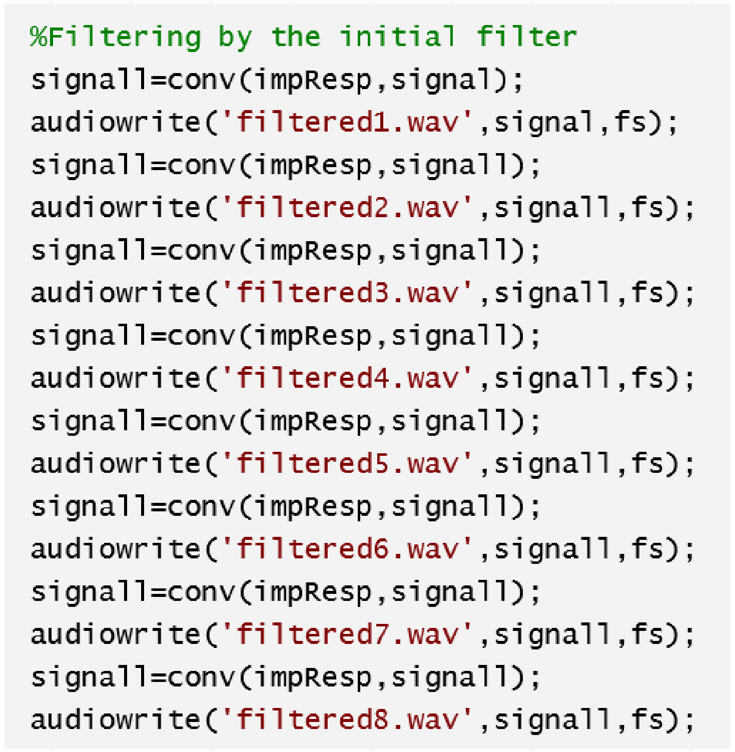


Figure 6, Impulse response of the system

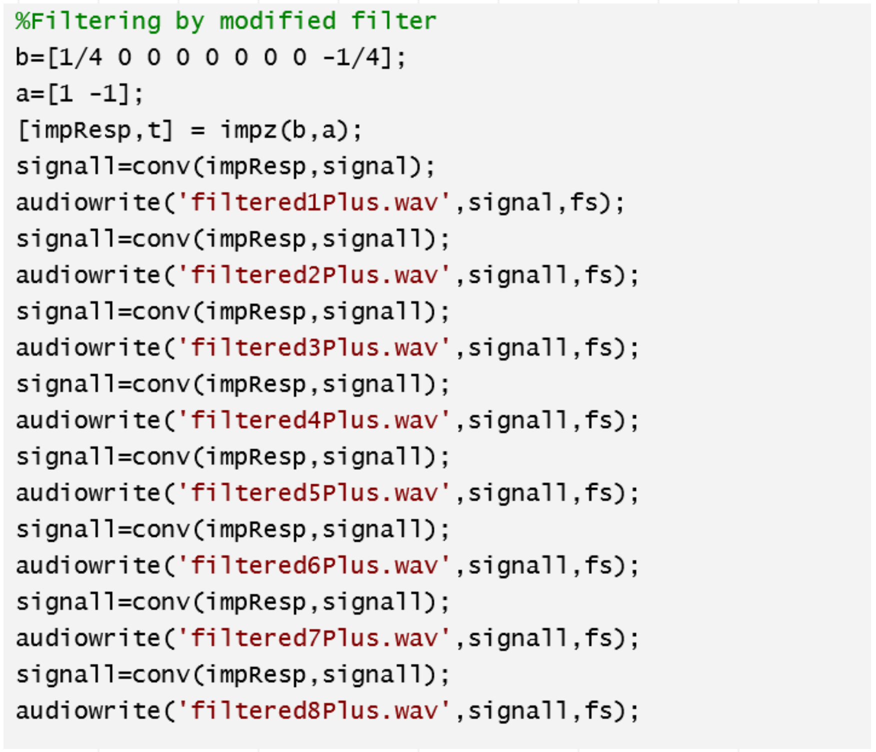
Step 4, Filtering Process:

- Calculate the impulse response of the initial filter using the ‘impz’ function.
- Plot the impulse response to analyze its characteristics and determine its length.

Step 4, Filtering Process (Cont.): Code:



```
%Filtering by the initial filter
signall=conv(impResp,signal);
audiowrite('filtered1.wav',signal,fs);
signall=conv(impResp,signall);
audiowrite('filtered2.wav',signall,fs);
signall=conv(impResp,signall);
audiowrite('filtered3.wav',signall,fs);
signall=conv(impResp,signall);
audiowrite('filtered4.wav',signall,fs);
signall=conv(impResp,signall);
audiowrite('filtered5.wav',signall,fs);
signall=conv(impResp,signall);
audiowrite('filtered6.wav',signall,fs);
signall=conv(impResp,signall);
audiowrite('filtered7.wav',signall,fs);
signall=conv(impResp,signall);
audiowrite('filtered8.wav',signall,fs);
```



```
%Filtering by modified filter
b=[1/4 0 0 0 0 0 0 0 -1/4];
a=[1 -1];
[impResp,t] = impz(b,a);
signall=conv(impResp,signal);
audiowrite('filtered1Plus.wav',signal,fs);
signall=conv(impResp,signall);
audiowrite('filtered2Plus.wav',signall,fs);
signall=conv(impResp,signall);
audiowrite('filtered3Plus.wav',signall,fs);
signall=conv(impResp,signall);
audiowrite('filtered4Plus.wav',signall,fs);
signall=conv(impResp,signall);
audiowrite('filtered5Plus.wav',signall,fs);
signall=conv(impResp,signall);
audiowrite('filtered6Plus.wav',signall,fs);
signall=conv(impResp,signall);
audiowrite('filtered7Plus.wav',signall,fs);
signall=conv(impResp,signall);
audiowrite('filtered8Plus.wav',signall,fs);
```

Figure 7, Filtering Process Section Code

Step 5, Results:

The project yielded the following results:

- Frequency spectra and phase spectra of the original noisy audio signal.
- Frequency response of the initial filter, including magnitude and phase spectra.
- Impulse response of the initial and modified filters.
- Filtered audio signals at different stages of filtering.

Conclusion:

In conclusion, this project effectively utilized filtering techniques to reduce noise and enhance the audio quality of a noisy signal. The analysis of frequency spectra, filter characteristics, and impulse responses provided valuable insights, highlighting the significance of audio filtering techniques in improving audio signals.

References:

1. Alirezajaberirad, Filtering Noisy Audio, https://github.com/alirezajaberirad/Signals-and-Systems/tree/main/CA2%20-%20Filtering%20Noisy%20Audio

2. MATLAB, https://www.mathworks.com/products/matlab.html

3. MathWorks, Fourier Analysis and Filtering, https://in.mathworks.com/help/matlab/fourier-analysis-and-filtering.html?s_tid=CRUX_lftnav

4. Fast Fourier Transform, https://en.wikipedia.org/wiki/Fast_Fourier_transform#:~:text=A%20fast%20Fourier%20transform%20(FFT,frequency%20domain%20and%20vice%20versa.

5. MathWorks, Fast Fourier Transform (FTT), https://in.mathworks.com/help/matlab/math/fourier-transforms.html

THANK YOU