İSTANBUL TİCARET ÜNİVERSİTESİ

FACULTY OF ENGINEERING

MECHATRONICS DEPARTMENT

EEE-206 SIGNALS AND SYSTEMS

2024-2025 Autumn Semester

AUDIO FILTER COMPARISON REPORT

Emir Ayar

200030473

December 29, 2024

Lecturer: Prof. Dr. Mehmet SEZGİN

CONTENTS

1. PURPOSE AND SCOPE

The purpose of this project is to design and analyze various digital filters (Lowpass, Highpass, Bandpass, and Bandstop) using MATLAB and apply them to an audio signal. The objective is to understand the impact of filter parameters on audio signals by comparing filtered outputs with the original audio.

The project involves the following:

1. Designing four types of digital filters using MATLAB's Filter Designer.
2. Generating two variations for each filter type by changing key parameters (cutoff frequencies, filter order, etc.).
3. Applying these filters to an audio file and observing the results.
4. Preparing a report that evaluates the filtering results and discusses the impact of filter parameters.

2. STUDIES CARRIED OUT

2.1 Audio File and Filter Design

The audio file used in this project is a recorded `.mp3` file loaded into MATLAB with a sampling frequency of 48 kHz.

The digital filters were designed using MATLAB's `filterDesigner` app. The following filters were created:

1. **Lowpass Filter** (two variations with cutoff frequencies at 3 kHz and 6 kHz).
2. **Highpass Filter** (two variations with cutoff frequencies at 5 kHz and 2 kHz).
3. **Bandpass Filter** (two variations with passbands of 2-8 kHz and 4-10 kHz).
4. **Bandstop Filter** (two variations with stopbands of 3-7 kHz and 1.5-3 kHz).

2.2 MATLAB Implementation

Each filter was saved as a MATLAB function, and the audio signal was filtered using these functions. The filtered audio was played sequentially for evaluation.

The following steps were implemented:

- Load the original audio file.
- Apply each filter (two variations per type) to the audio data using the `filter` function.
- Play the original and filtered audio files sequentially to observe the effects.

3. RESULTS

3.1 Observations on Filtered Audio

- **Lowpass Filter**:
  - Variation 1 (cutoff = 3 kHz): High frequencies were attenuated, producing a muffled sound.
  - Variation 2 (cutoff = 6 kHz): Retained more high frequencies, providing a less muffled result.
- **Highpass Filter**:
  - Variation 1 (cutoff = 5 kHz): Low frequencies were attenuated, producing a sharp sound.
  - Variation 2 (cutoff = 2 kHz): Slightly more low frequencies were retained, creating a more balanced sharpness.
- **Bandpass Filter**:
  - Variation 1 (passband = 2-8 kHz): Emphasized midrange frequencies, giving a focused sound.
  - Variation 2 (passband = 4-10 kHz): A broader passband allowed for slightly richer audio.
- **Bandstop Filter**:
  - Variation 1 (stopband = 3-7 kHz): Removed midrange frequencies, resulting in hollow audio.
  - Variation 2 (stopband = 1.5-3 kHz): A wider stopband produced a more dramatic hollowing effect.

4. EVALUATION OF RESULTS

The filtering results demonstrated the following:

1. The choice of cutoff frequencies and filter order significantly influences the filtered audio's quality and characteristics.
2. Lowpass filters effectively attenuated high frequencies, while highpass filters emphasized sharpness by removing low frequencies.
3. Bandpass filters provided focused output in a specific frequency range, while bandstop filters created hollow-sounding audio by suppressing certain midrange frequencies.
4. The second variations of each filter type offered more relaxed frequency control, demonstrating the impact of wider cutoff ranges.

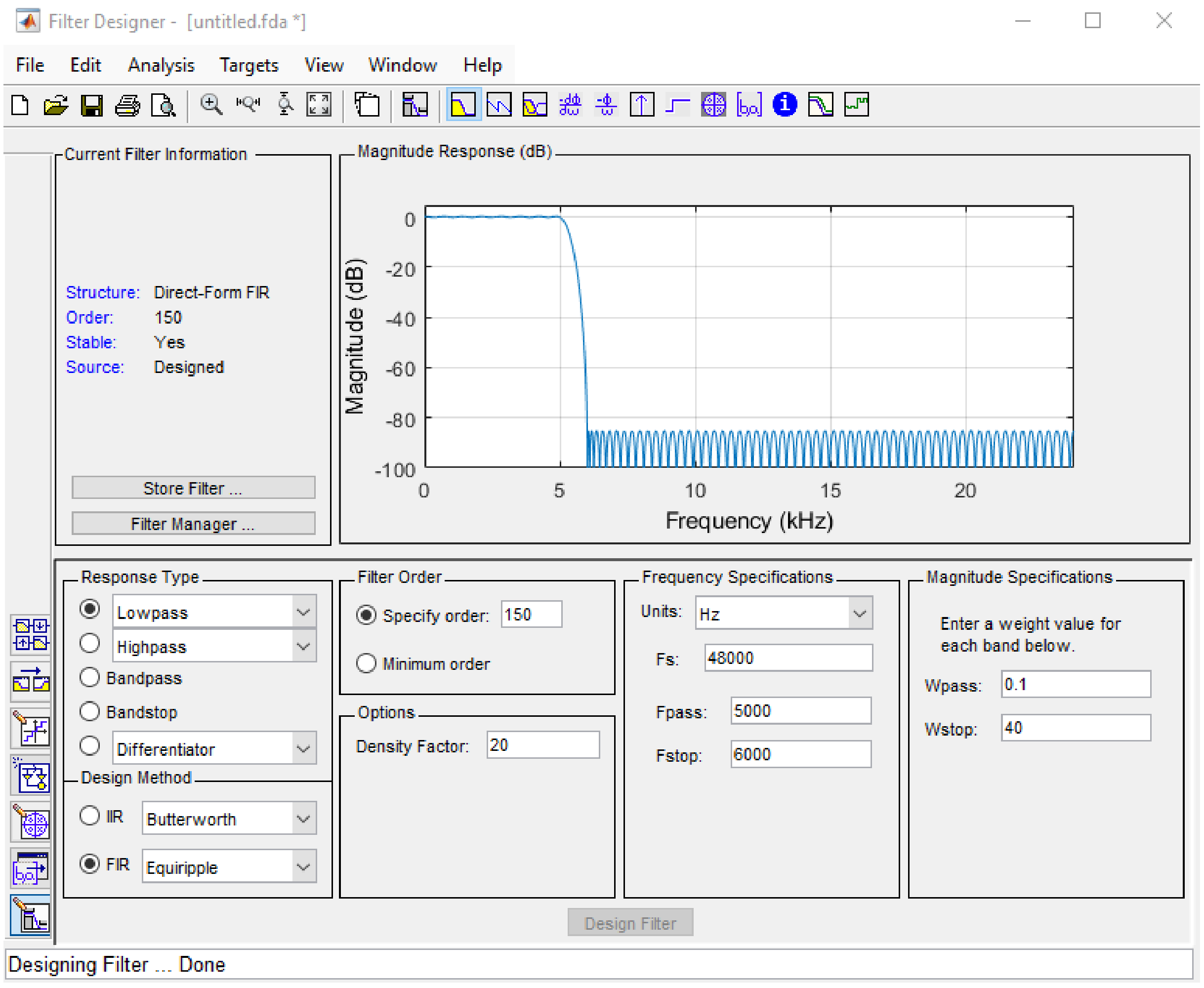
5. REFERENCES

MATLAB Documentation: https://www.mathworks.com/help/signal/ug/introduction-to-filter-designer.html

Oppenheim, A.V., & Schafer, R.W. (2013). Digital Signal Processing

6. APPENDICES

Appendice -1. Filter Designer Interface



Current Filter Information

Structure:	Direct-Form FIR
Order:	150
Stable:	Yes
Source:	Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method: IIR Butterworth [✓ FIR] Equiripple

Filter Order: [✓ Specify order: 150] Minimum order

Options: Density Factor: 20

Frequency Specifications: Units: Hz; Fs: 48000; Fpass: 5000; Fstop: 6000

Magnitude Specifications: Enter a weight value for each band below. Wpass: 0.1; Wstop: 40

Designing Filter ... Done

Appendice -2. Matlab Code

```
% Load the MP3 file
[audioData, fs] = audioread('my_audio.mp3'); % 'fs' is the sampling frequency
% Original audio
disp('Playing original audio...');
sound(audioData, fs); % Play the original audio
pause(length(audioData) / fs); % Wait for the audio to finish
% Lowpass Filter - Variation 1
disp('Lowpass filter (variation 1) applied and playing...');
lowpass_filter1 = LowpassFilter(); % Load the first lowpass filter
filtered_audio_lowpass1 = filter(lowpass_filter1, audioData); % Apply the filter
sound(filtered_audio_lowpass1, fs); % Play the filtered audio
pause(length(filtered_audio_lowpass1) / fs); % Wait for the audio to finish
% Lowpass Filter - Variation 2
disp('Lowpass filter (variation 2) applied and playing...');
lowpass_filter2 = LowpassFilter2(); % Load the second lowpass filter
filtered_audio_lowpass2 = filter(lowpass_filter2, audioData); % Apply the filter
sound(filtered_audio_lowpass2, fs); % Play the filtered audio
```

```
pause(length(filtered_audio_lowpass2) / fs); % Wait for the audio to finish
% Highpass Filter - Variation 1
disp('Highpass filter (variation 1) applied and playing...');
highpass_filter1 = HighpassFilter(); % Load the first highpass filter
filtered_audio_highpass1 = filter(highpass_filter1, audioData); % Apply the filter
sound(filtered_audio_highpass1, fs); % Play the filtered audio
pause(length(filtered_audio_highpass1) / fs); % Wait for the audio to finish
% Highpass Filter - Variation 2
disp('Highpass filter (variation 2) applied and playing...');
highpass_filter2 = HighpassFilter2(); % Load the second highpass filter
filtered_audio_highpass2 = filter(highpass_filter2, audioData); % Apply the filter
sound(filtered_audio_highpass2, fs); % Play the filtered audio
pause(length(filtered_audio_highpass2) / fs); % Wait for the audio to finish
% Bandstop Filter - Variation 1
disp('Bandstop filter (variation 1) applied and playing...');
bandstop_filter1 = BandstopFilter(); % Load the first bandstop filter
filtered_audio_bandstop1 = filter(bandstop_filter1, audioData); % Apply the filter
sound(filtered_audio_bandstop1, fs); % Play the filtered audio
pause(length(filtered_audio_bandstop1) / fs); % Wait for the audio to finish
% Bandstop Filter - Variation 2
disp('Bandstop filter (variation 2) applied and playing...');
bandstop_filter2 = BandstopFilter2(); % Load the second bandstop filter
filtered_audio_bandstop2 = filter(bandstop_filter2, audioData); % Apply the filter
sound(filtered_audio_bandstop2, fs); % Play the filtered audio
pause(length(filtered_audio_bandstop2) / fs); % Wait for the audio to finish
% Bandpass Filter - Variation 1
disp('Bandpass filter (variation 1) applied and playing...');
bandpass_filter1 = BandpassFilter(); % Load the first bandpass filter
filtered_audio_bandpass1 = filter(bandpass_filter1, audioData); % Apply the filter
sound(filtered_audio_bandpass1, fs); % Play the filtered audio
pause(length(filtered_audio_bandpass1) / fs); % Wait for the audio to finish
% Bandpass Filter - Variation 2
disp('Bandpass filter (variation 2) applied and playing...');
bandpass_filter2 = BandpassFilter2(); % Load the second bandpass filter
filtered_audio_bandpass2 = filter(bandpass_filter2, audioData); % Apply the filter
sound(filtered_audio_bandpass2, fs); % Play the filtered audio
pause(length(filtered_audio_bandpass2) / fs); % Wait for the audio to finish
```