A cover of a book

Description automatically generated

**Design and Implementation of a 5-band Graphic Equalizer**

**Project Details**

**Part I**

The goal of this project is to design a 10-band graphic equalizer and then to implement it - first employing Simulink to check the design and then designing a GUI. Matlab can be employed to design the required filters and then Simulink can be used to implement the graphic equalizer in real time. Most commercial equalizers use either 1/3 octave or 2/3 octave bandpass filters but to keep this from becoming too large we will employ one octave bandpass filters.

Following are the design specifications for the equalizer:

1. Employing Matlab, design 5 different bandpass filters with center frequencies of 63 Hz, 250 Hz, 1000 Hz, 4000 Hz, and 16000 Hz. These center frequencies correspond to the ISO (International Standards Organization) standard for graphic equalizer center frequencies.

2. The bandwidth of each filter is the frequency difference ∆f = f2 – f1, where f1 and f2 correspond to the frequencies where the gain is 3 dB less than the maximum gain at the center frequency. It also is necessary to choose f1 and f2 such that the center frequency, fc, is equal to the geometric mean of f1 and f2, i.e. fc = (f1f2) 1/2 . We also have to choose the bandwidth of each filter so that we get a flat frequency response when all filter gains are equal and added together.

3. You can use Butterworth filters; however you are free to choose the order of the filters. The Matlab help file for the Butterworth filter is the following: [B,A] = butter(N,Wn) designs an Nth order lowpass digital Butterworth filter and returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The coefficients are listed in descending powers of z. The cutoff frequency Wn must be 0.0 < Wn < 1.0, with 1.0 corresponding to half the sample rate. If Wn is a two-­­element vector, Wn = [W1 W2], butter returns an order 2N bandpass filter with passband W1 < W < W2. [B,A] = butter(N,Wn,'high') designs a highpass filter. [B,A] = butter(N,Wn,'low') designs a lowpass filter. [B,A] = butter(N,Wn,'stop') is a bandstop filter if Wn = [W1 W2].

4. Write a Matlab m-file to compute the set of filter coefficients and plot the combination (sum) of all filter frequency responses. Note that you can use the ‘freqz’ command to easily find the frequency response of a filter defined by the filter coefficient arrays B and A. Your goal is to achieve as flat of a frequency response as you can when all the frequency response of all filters are added ± 1 dB is a good goal. Remember that the center frequency of each filter must be fixed to one of the five values given above and the upper and lower cutoff frequencies f2 & f1 must satisfy fc = (f1f2) 1/2 . Your goal is to find the ∆f value for each filter that achieves a flat frequency response when all filters are combined with equal weights. [Hint: the filters should all be constant Q, where Q = fc / (f2 – f1), so once you find the right value for Q all filters should have the same Q.]

The simplified 3-band graphic equalizer is shown below, where the filters are in parallel and each one is followed by a gain (using the Matlab slider gain block). Your mixer will have 5 filters in parallel.

We would like to be able to adjust the gain of each band by ± 12 dB. Remember that 6dB corresponds to approximately a factor of 2x, so 12 dB is about 4x. So +12 dB is like multiplying by 4 and -12 dB is like multiplying by ¼.Use these values as the limits for the slider gain blocks.

One final note: The “From Multimedia File” and “To Audio Device” blocks can be found in the DSP Toolbox.

**Part II**

Design a GUI with the following provisions

1. A ‘load’ button that can load an audio file of your choice (Suggestion: choose a file with large frequency range e.g a symphony).

2. Display for the input and output signals.

3. Display for the input and output spectrum.

4. Adjustable gain sliders for the filters.

5. A ‘play’ button that can playback the output file.

**Introduction**

The project aims to design and implement a 5-band graphic equalizer in two parts. In Part I, the focus is on crafting five bandpass filters with specific center frequencies, adhering to ISO standards. Matlab and Simulink are employed for design validation and real-time implementation. The use of Butterworth filters with adjustable orders is recommended. The goal is to achieve a flat frequency response when combining all filters with equal weights, ensuring constant Q. Part II involves designing a user-friendly GUI with features such as a 'load' button for audio file selection, input/output signal displays, adjustable gain sliders, and a 'play' button for playback, facilitating intuitive audio customization.

1. Designing of Filters on Matlab

**Objectives**

1. Implementation of those filters using Simulink
2. Creating a GUI for our Audio Equaliser

In this project conducted using MATLAB, we aimed to design a 5-band graphic equalizer. The process involved specifying center frequencies for each band and determining a Q factor. As we have to design a single octave filter so we will be using a Q factor of √2. Utilizing MATLAB's Butterworth filter design, octave bandpass filters were created for each center frequency. The higher frequency is just double the lower frequency in each case. The code then systematically plots the individual frequency responses of each band and their combined response, providing a visual representation of the equalizer's behavior. Importantly, this MATLAB implementation serves as a crucial step before transitioning to Simulink. Once the filters are designed and optimized in MATLAB, they are exported to Simulink for real-time implementation. The individual functions in the code handle the calculation of lower and upper cutoff frequencies, the design of Butterworth bandpass filters, and the plotting of frequency responses. This MATLAB-to-Simulink workflow ensures a seamless integration of the designed filters into a real-time graphic equalizer system.

**Filter Designing using MATLAB**

**Code:**

% Define center frequencies (Hz)

center\_frequencies = [63, 250, 1000, 4000, 16000];

% Define Q factor

Q = sqrt(2);

% Define sampling frequency (Fs)

Fs = 62000;

% Preallocate arrays for filter coefficients

B = cell(1, length(center\_frequencies));

A = cell(1, length(center\_frequencies));

% Design octave bandpass filters for each center frequency

for i = 1:length(center\_frequencies)

fc = center\_frequencies(i);

% Calculate lower and upper cutoff frequencies for octave bandpass filter

f1 = fc / Q;

f2 = 2 \* f1;

% Design Butterworth bandpass filter with Q factor

[B{i}, A{i}] = butter(3, [f1, f2]/(Fs/2), 'bandpass');

end

% Plot individual frequency responses of all bands

figure;

for i = 1:length(center\_frequencies)

[H, F] = freqz(B{i}, A{i}, 1024, Fs);

plot(F, 20\*log10(abs(H)), 'LineWidth', 1.5);

hold on;

end

% Plot the combined frequency response of all filters

combined\_response = zeros(1024, 1);

for i = 1:length(center\_frequencies)

[H, F] = freqz(B{i}, A{i}, 1024, Fs);

combined\_response = combined\_response + abs(H).^2; % Accumulate squared magnitude

end

% Normalize the combined response

combined\_response = 20\*log10(sqrt(combined\_response/length(center\_frequencies)));

% Plot the normalized combined response

plot(F, combined\_response, 'LineWidth', 1.5);

title('Combined Frequency Response of 1-Octave Bandpass Filters');

xlabel('Frequency (Hz)');

ylabel('Gain (dB)');

legend('Band 1', 'Band 2', 'Band 3', 'Band 4', 'Band 5', 'Combined', 'Location', 'northeastoutside');

hold off;

%grid on;

% Display additional filter information

fprintf('Filter Information:\n');

Filter Information:

for i = 1:length(center\_frequencies)

f1 = center\_frequencies(i) / Q;

f2 = 2 \* f1;

fprintf('Band %d - Center Frequency: %d Hz, Lower Frequency: %.2f Hz, Upper Frequency: %.2f Hz, Bandwidth: %.2f Hz\n', i, center\_frequencies(i), f1, f2, f2-f1);

end

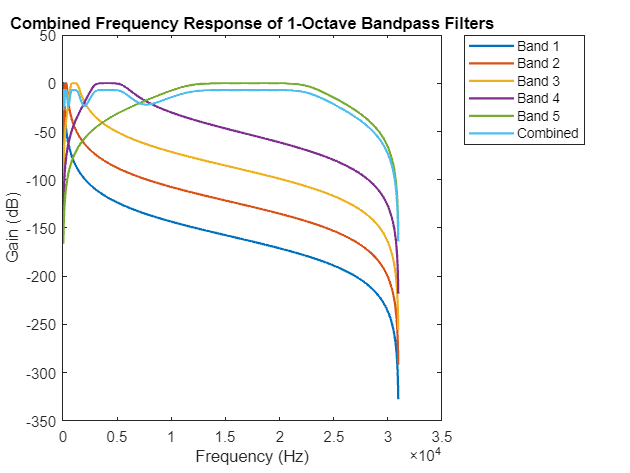
**Output:**

Figure 1: Combined Frequency Response of our 5 Band Filter

**Values of Upper and Lower Frequencies Calculated and Printed by our Code:**

Band 1 - Center Frequency: 63 Hz, Lower Frequency: 44.55 Hz, Upper Frequency: 89.10 Hz, Bandwidth: 44.55 Hz

Band 2 - Center Frequency: 250 Hz, Lower Frequency: 176.78 Hz, Upper Frequency: 353.55 Hz, Bandwidth: 176.78 Hz

Band 3 - Center Frequency: 1000 Hz, Lower Frequency: 707.11 Hz, Upper Frequency: 1414.21 Hz, Bandwidth: 707.11 Hz

Band 4 - Center Frequency: 4000 Hz, Lower Frequency: 2828.43 Hz, Upper Frequency: 5656.85 Hz, Bandwidth: 2828.43 Hz

Band 5 - Center Frequency: 16000 Hz, Lower Frequency: 11313.71 Hz, Upper Frequency: 22627.42 Hz, Bandwidth: 11313.71 Hz

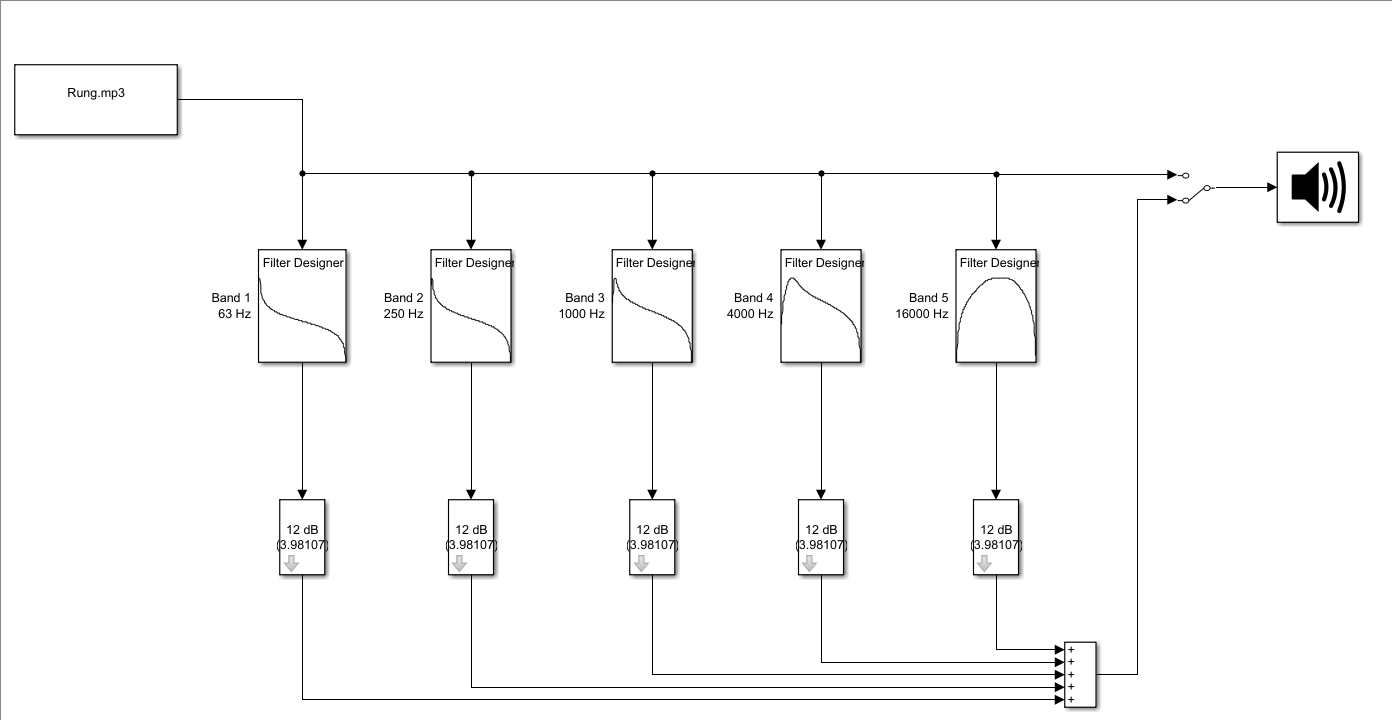
**Simulink Schematic:**

Figure 2: Simulink Schematic of our Audio Equaliser

**Simulink Circuit and Filters**

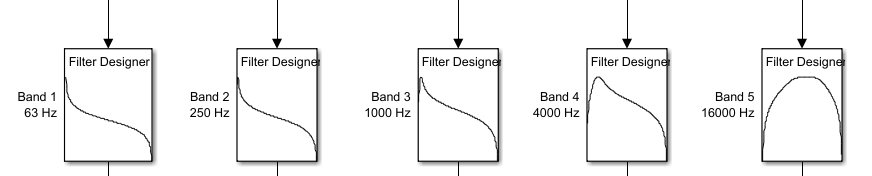
We have used the following blocks in our Simulink Schematic:

1. **From Multimedia File Block:**

The "From Multimedia File" block in Simulink is a vital tool for importing audio data into simulations. It serves as an input interface, facilitating the seamless integration of external audio files into Simulink models. This block is essential for testing and analyzing systems involving audio processing within the Simulink environment, enhancing the versatility of modeling and simulation tasks.A white rectangle with blue text

Description automatically generated It is a part of the DSP Toolbox

1. **Filter Designer Block:**

The "Filter Designer" block in Simulink is a user-friendly interface for creating and customizing digital filters. It enables real-time adjustments of filter specifications, including type, order, and cutoff frequencies. Supporting various filter designs, such as lowpass and highpass, this block is essential for efficient digital filter design within Simulink simulations.

1. **dB Gain Block:**

A white rectangular object with black text and black arrow pointing down

Description automatically generatedThe "dB Gain" block in Simulink is a crucial component for adjusting signal gain in decibels. This block simplifies the process of scaling signal amplitudes, providing an efficient way to apply logarithmic gain adjustments within Simulink simulations.

1. **Add Block:**

The "Add" block in Simulink is a core element for mathematical operations in a model. It enables the summation of multiple input signals, providing a versatile tool for combining and manipulating signals within a simulation. The block enhances the modularity and flexibility of Simulink models, allowing users to perform additive operations and create complex signal processing or control system models efficiently.

1. A diagram of a computer program

   Description automatically generated**Manual Switch Block:**

The "Manual Switch" block in Simulink enables users to manually toggle between multiple input signals, controlling signal flow within a model. This block is valuable for scenarios requiring dynamic signal routing or conditional switching. By interacting with the "Manual Switch," users can efficiently simulate different scenarios and test the impact of various signal pathways in their models, enhancing adaptability and versatility in Simulink simulations.A diagram of a manual switch

Description automatically generated

1. **Audio Device Writer Block:**

The "Audio Device Writer" block in Simulink facilitates audio output in simulation models. It allows users to stream simulated audio signals directly to audio devices for real-time auditory feedback during simulations. By connecting this block to the desired audio source, users can integrate and test audio processing algorithms or control systems with ease. The "Audio Device Writer" enhances Simulink simulations by providing a practical way to evaluate the real-world impact of models through audible output.

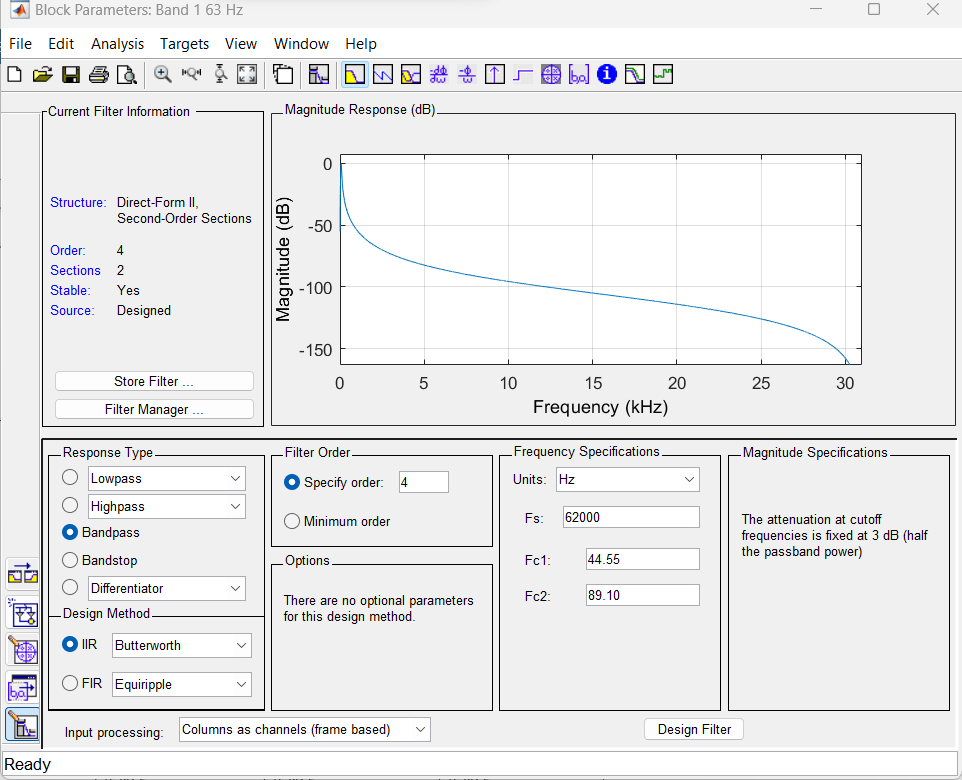
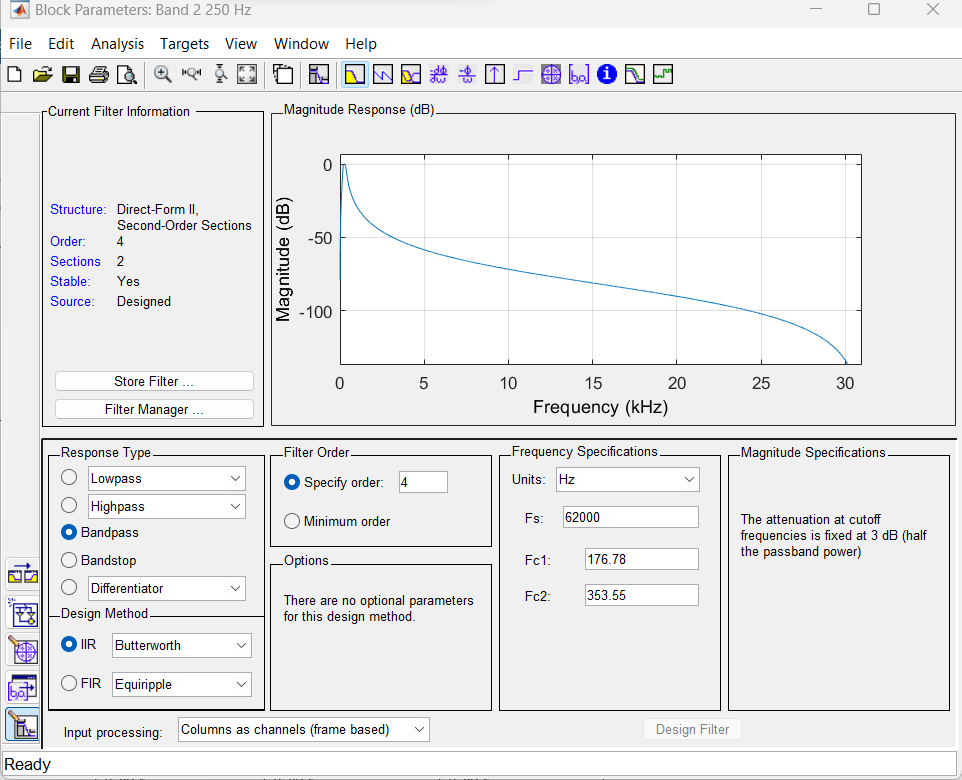
Now let us look at the designed filters of our schematic:

Figure 4: 250 Hz BandPass Filter

Figure 3: 63 Hz BandPass Filter

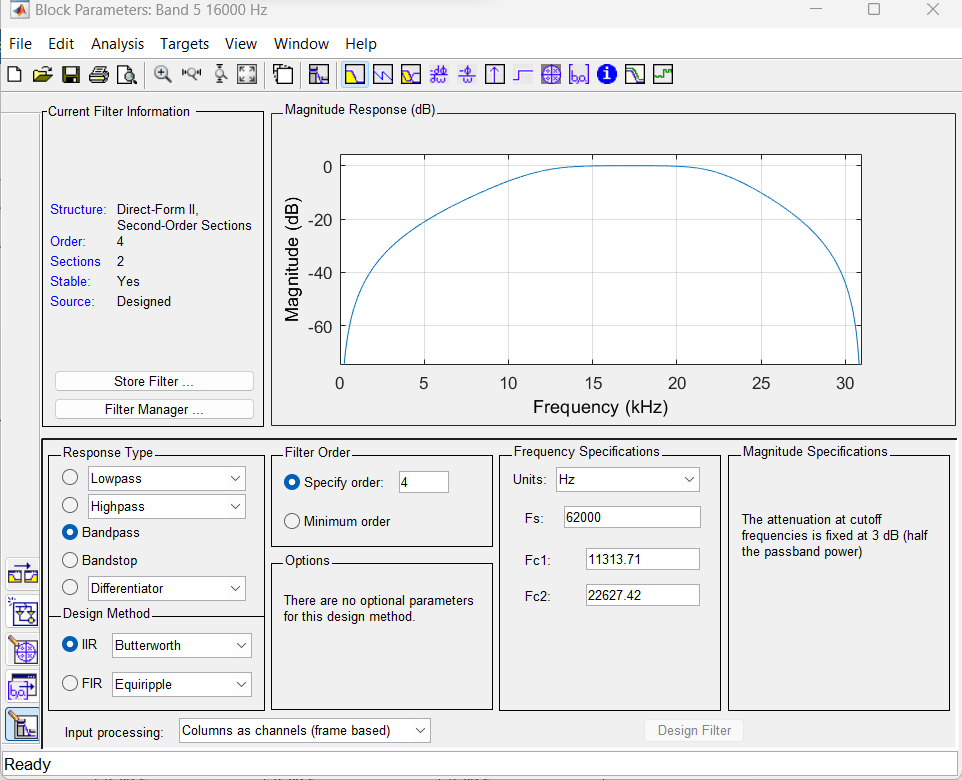
A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure 6: 4000 Hz BandPass Filter

Figure 5: 1000 Hz BandPass Filter

classdef Project\_GUI\_Code < matlab.apps.AppBase

**GUI Code**

Figure 7: 16000 Hz BandPass Filter

% Properties that correspond to app components

properties (Access = public)

UIFigure matlab.ui.Figure

UIAxesBackground matlab.ui.control.UIAxes

TrackDropDown matlab.ui.control.DropDown

SongSelectDropDownLabel matlab.ui.control.Label

PLOTButton matlab.ui.control.Button

RESETButton matlab.ui.control.Button

PLAYButton matlab.ui.control.Button

LOADButton matlab.ui.control.Button

EditField\_5 matlab.ui.control.EditField

Slider\_5 matlab.ui.control.Slider

Band5fc16000HzLabel matlab.ui.control.Label

EditField\_4 matlab.ui.control.EditField

Slider\_4 matlab.ui.control.Slider

Band4fc4000HzLabel matlab.ui.control.Label

EditField\_3 matlab.ui.control.EditField

Slider\_3 matlab.ui.control.Slider

Band3fc1000HzLabel matlab.ui.control.Label

EditField\_2 matlab.ui.control.EditField

Slider\_2 matlab.ui.control.Slider

Band2fc250HzLabel matlab.ui.control.Label

EditField\_1 matlab.ui.control.EditField

Band1fc63HzLabel matlab.ui.control.Label

Slider\_1 matlab.ui.control.Slider

UIAxes5 matlab.ui.control.UIAxes

UIAxes4 matlab.ui.control.UIAxes

UIAxes3 matlab.ui.control.UIAxes

UIAxes2 matlab.ui.control.UIAxes

UIAxes1 matlab.ui.control.UIAxes

% Additional properties for storing data

audioFilePath

audioSignal

sampleRate

fileReader

deviceWriter

end

properties (Access = private)

fg = [63,250,1000,4000,16000];

fk = [63, 70.8, 250, 282, 1000, 1120, 4000, 4470, 16000, 17800];

fs = 44100;

wn = 2\*pi\*[63, 66.9, 70.8, 75, 250, 266, 282, 299, 1000, 1060, 1120, 1180, 4000, 4220, 4470, 4730, 16000, 16800, 17800, 18800];

isStop = 0;

isPlay = 0;

end

% Callbacks that handle component events

methods (Access = private)

function a = den(app,k,Fs)

thetak = 2\*pi\*app.fk(k)/Fs;

if k>1 && k<10

dthetak = (2\*pi\*app.fk(k+1)/Fs-2\*pi\*app.fk(k-1)/Fs)/2;

elseif k == 1

dthetak = 2\*pi\*app.fk(2)/Fs-2\*pi\*app.fk(1)/Fs;

else

dthetak = 2\*pi\*app.fk(10)/Fs-2\*pi\*app.fk(9)/Fs;

end

pk = exp(-dthetak/2);

a = [1 -2\*pk\*cos(thetak) pk^2];

end

function Mrplus = Mrp(app, Fs)

M = zeros(20, 21);

M(:, 22) = ones(20, 1);

sqW = Weight(app, app.Slider\_1.Value, app.Slider\_2.Value, app.Slider\_3.Value, app.Slider\_4.Value, app.Slider\_5.Value);

for n = 1:20

for k = 1:10

M(n, 2 \* k - 1) = 1 / (den(app, k, Fs) \* [1; exp(-app.wn(n) / Fs \* 1i); exp(-2 \* app.wn(n) / Fs \* 1i)]);

M(n, 2 \* k) = exp(-app.wn(n) / Fs \* 1i) / (den(app, k, Fs) \* [1; exp(-app.wn(n) / Fs \* 1i); exp(-2 \* app.wn(n) / Fs \* 1i)]);

end

M(n, :) = M(n, :) \* sqW(n);

end

Mr = [real(M); imag(M)];

% Use the pseudo-inverse to improve stability

Mrplus = pinv(transpose(Mr) \* Mr) \* transpose(Mr);

end

%Calculation of the target response vector

function htr = target(app,G1,G2,G3,G4,G5)

y = 10.^(1/20\*pchip([-flip(app.fg) app.fg],[flip([G1,G2,G3,G4,G5]) [G1,G2,G3,G4,G5]],linspace(-app.fs/2,app.fs/2,2^16)))';

phase = unwrap(imag(-hilbert(log(y))));

% Adjust the range based on the actual length of the 'phase' vector

phase\_start = round(32769 \* length(phase) / 2^16);

phase\_end = round(64124 \* length(phase) / 2^16);

phase = phase(phase\_start:phase\_end);

i = 1;

fi = zeros(1,20);

for w = app.wn/(2\*pi)

if round(w\*length(phase)/21100) > length(phase)

fi(i) = phase(length(phase));

else

fi(i) = phase(round(w\*length(phase)/21100));

end

i=i+1;

end

htr = [real(exp(1i\*fi'));imag(exp(1i\*fi'))];

end

%Calculation of the optimal numerator coefficients

function popt = num(~,htr,Mrplus)

popt = Mrplus\*htr;

end

%Filtering process algorithm

function yk = filterNew(app,bWithIndex,xk1,ybuffer,xbuffer,Fs,n)

yk = filter(bWithIndex',den(app,n,Fs),xk1,filtic(bWithIndex',den(app,n,Fs),ybuffer,xbuffer));

end

%Calculation of the Weighting factors

function sqW = Weight(app,G1,G2,G3,G4,G5)

ht = 10.^(1/20\*pchip([app.fg],[G1,G2,G3,G4,G5],app.wn/(2\*pi)))';

sqW1 = zeros(20,1);

for i = 1:20

sqW1(i) = 1/ht(i);

end

sqW = sqW1;

end

% Button pushed function: LOADButton

function LOADButtonPushed(app, ~)

% Set the path to the desired directory

folderPath = 'C:\Users\Azlaan\Music\';

% List MP3 files

mp3Files = struct2cell(dir(fullfile(folderPath, '\*.mp3')));

mp3Files = mp3Files(1, :);

% List WAV files

wavFiles = struct2cell(dir(fullfile(folderPath, '\*.wav')));

wavFiles = wavFiles(1, :);

% Combine MP3 and WAV files into a single list

audioFiles = [mp3Files, wavFiles];

% Update app.TrackDropDown.Items

app.TrackDropDown.Items = audioFiles;

% Check if the list is empty and show a message if true

if isempty(app.TrackDropDown.Items)

uialert(app.UIFigure, ...

['Your current folder does not contain any audio files'], ...

'Info', 'Icon', 'info');

end

end

% Button pushed function: PLOTButton

function PLOTButtonPushed(app, event)

% Get the selected MP3 file name from the dropdown

selectedTrack = app.TrackDropDown.Value;

% Check if a song is selected

if isempty(selectedTrack)

disp('Please select a song from the dropdown.');

return;

end

% Construct the full path to the selected MP3 file

filePath = fullfile('C:\Users\Azlaan\Music\', selectedTrack);

% Read the audio signal from the selected MP3 file

try

[app.audioSignal, app.sampleRate] = audioread(filePath);

catch

disp('Error loading the selected song.');

return;

end

if isempty(app.audioSignal)

disp('Please load a song first.');

return;

end

% Design and apply filters here

center\_frequencies = [63, 250, 1000, 4000, 16000];

Q = sqrt(2);

desired\_order = 3;

Fs = 48000;

% Preallocate arrays for filter coefficients

B = cell(1, length(center\_frequencies));

A = cell(1, length(center\_frequencies));

f1 = [44.55, 176.78, 707.11, 2828.43, 11313.71];

f2 = [89.10, 353.55, 1414.21, 5656.85, 22627.42];

% Design Butterworth filters

for i = 1:length(center\_frequencies)

fc = center\_frequencies(i);

[B{i}, A{i}] = butter(desired\_order, [f1(i), f2(i)]/(Fs/2), 'bandpass');

end

% Apply filters to the input signal and multiply each band by the corresponding gain

filtered\_signals = cell(1, length(center\_frequencies));

combined\_output = zeros(size(app.audioSignal));

for i = 1:length(center\_frequencies)

% Apply filter

filtered\_signals{i} = filter(B{i}, A{i}, app.audioSignal);

% Multiply by gain from the corresponding slider

gain = app.(['Slider\_', num2str(i)]).Value; % Access slider value dynamically

filtered\_signals{i} = filtered\_signals{i} \* 10^(gain/20); % Convert dB to linear scale

% Sum the adjusted signals

combined\_output = combined\_output + filtered\_signals{i};

end

% Plot the original signal

t\_original = (0:length(app.audioSignal)-1) / app.sampleRate;

clf(app.UIAxes1)

plot(app.UIAxes1, t\_original, app.audioSignal);

title(app.UIAxes1, 'Original Signal');

xlabel(app.UIAxes1, 'Time (s)');

ylabel(app.UIAxes1, 'Amplitude');

axis(app.UIAxes1, 'tight');

% Plot the combined adjusted signal on UIAxes2

t\_adjusted = (0:length(combined\_output)-1) / app.sampleRate;

clf(app.UIAxes2)

plot(app.UIAxes2, t\_adjusted, combined\_output);

title(app.UIAxes2, 'Combined Adjusted Signal');

xlabel(app.UIAxes2, 'Time (s)');

ylabel(app.UIAxes2, 'Amplitude');

axis(app.UIAxes2, 'tight');

% Plot the input spectrum on UIAxes3

Audio\_len = length(app.audioSignal);

FFT\_audio = fft(app.audioSignal);

P2\_audio = abs(FFT\_audio / Audio\_len);

P1\_audio = P2\_audio(1:Audio\_len/2+1);

P1\_audio(2:end-1) = 2 \* P1\_audio(2:end-1);

freq\_audio = app.sampleRate \* (0:(Audio\_len/2)) / Audio\_len;

plot(app.UIAxes3, freq\_audio, P1\_audio);

title(app.UIAxes3, 'Input Spectrum');

xlabel(app.UIAxes3, 'Frequency (Hz)');

ylabel(app.UIAxes3, 'Amplitude');

axis(app.UIAxes3, 'tight');

% Plot the output spectrum on UIAxes4

Audio\_len\_combined = length(combined\_output);

FFT\_combined = fft(combined\_output);

P2\_combined = abs(FFT\_combined / Audio\_len\_combined);

P1\_combined = P2\_combined(1:Audio\_len\_combined/2+1);

P1\_combined(2:end-1) = 2 \* P1\_combined(2:end-1);

freq\_combined = app.sampleRate \* (0:(Audio\_len\_combined/2)) / Audio\_len\_combined;

plot(app.UIAxes4, freq\_combined, P1\_combined);

title(app.UIAxes4, 'Output Spectrum');

xlabel(app.UIAxes4, 'Frequency (Hz)');

ylabel(app.UIAxes4, 'Amplitude');

axis(app.UIAxes4, 'tight');

% Plot the characteristic filter response on UIAxes5

combined\_response = zeros(size(app.audioSignal));

for i = 1:length(center\_frequencies)

% Calculate the frequency response of the filter

[H, F] = freqz(B{i}, A{i}, length(app.audioSignal), app.sampleRate);

% Multiply by gain from the corresponding slider

gain = app.(['Slider\_', num2str(i)]).Value; % Access slider value dynamically

H = H \* 10^(gain/20); % Convert dB to linear scale

% Sum the adjusted responses

combined\_response = combined\_response + abs(H);

end

% Plot the characteristic filter response on UIAxes5

clf(app.UIAxes5);

plot(app.UIAxes5, F, combined\_response, 'b');

title(app.UIAxes5, 'Characteristic Filter Response');

xlabel(app.UIAxes5, 'Frequency (Hz)');

ylabel(app.UIAxes5, 'Magnitude');

axis(app.UIAxes5, 'tight');

end

% Value changed function: Slider\_1

function Slider\_1ValueChanged(app, event)

% Get the current value of the slider

sliderValue = app.Slider\_1.Value;

% Display the value in the edit field

app.EditField\_1.Value = num2str(sliderValue);

end

% Value changed function: Slider\_2

function Slider\_2ValueChanged(app, event)

% Get the current value of the slider

sliderValue = app.Slider\_2.Value;

% Display the value in the edit field

app.EditField\_2.Value = num2str(sliderValue);

end

% Value changed function: Slider\_3

function Slider\_3ValueChanged(app, event)

% Get the current value of the slider

sliderValue = app.Slider\_3.Value;

% Display the value in the edit field

app.EditField\_3.Value = num2str(sliderValue);

end

% Value changed function: Slider\_4

function Slider\_4ValueChanged(app, event)

% Get the current value of the slider

sliderValue = app.Slider\_4.Value;

% Display the value in the edit field

app.EditField\_4.Value = num2str(sliderValue);

end

% Value changed function: Slider\_5

function Slider\_5ValueChanged(app, event)

% Get the current value of the slider

sliderValue = app.Slider\_5.Value;

% Display the value in the edit field

app.EditField\_5.Value = num2str(sliderValue);

end

% Value changed function: EditField\_1

% Value changed function: EditField\_1

function EditField\_1ValueChanged(app, event)

editfieldvalue = str2double(app.EditField\_1.Value);

% Check if the value is within the slider limits

editfieldvalue = max(min(editfieldvalue, app.Slider\_1.Limits(2)), app.Slider\_1.Limits(1));

% Update the corresponding slider

app.Slider\_1.Value = editfieldvalue;

end

% Value changed function: EditField\_2

function EditField\_2ValueChanged(app, event)

editfieldvalue = str2double(app.EditField\_2.Value);

editfieldvalue = max(min(editfieldvalue, app.Slider\_2.Limits(2)), app.Slider\_2.Limits(1));

app.Slider\_2.Value = editfieldvalue;

end

% Value changed function: EditField\_3

function EditField\_3ValueChanged(app, event)

editfieldvalue = str2double(app.EditField\_3.Value);

editfieldvalue = max(min(editfieldvalue, app.Slider\_3.Limits(2)), app.Slider\_3.Limits(1));

app.Slider\_3.Value = editfieldvalue;

end

% Value changed function: EditField\_4

function EditField\_4ValueChanged(app, event)

editfieldvalue = str2double(app.EditField\_4.Value);

editfieldvalue = max(min(editfieldvalue, app.Slider\_4.Limits(2)), app.Slider\_4.Limits(1));

app.Slider\_4.Value = editfieldvalue;

end

% Value changed function: EditField\_5

function EditField\_5ValueChanged(app, event)

editfieldvalue = str2double(app.EditField\_5.Value);

editfieldvalue = max(min(editfieldvalue, app.Slider\_5.Limits(2)), app.Slider\_5.Limits(1));

app.Slider\_5.Value = editfieldvalue;

end

% Button pushed function: PLAYButton

function PLAYButtonPushed(app, ~)

app.audioSignal = app.TrackDropDown.Value;

% Check if a song is loaded

if isempty(app.audioSignal)

disp('Please load a song first.');

return;

elseif strcmp(app.PLAYButton.Text,'PLAY') && app.isPlay == 0

% Specify the folder path where audio files are stored

folderPath = 'C:\Users\Azlaan\Music\';

% Construct the full path to the selected audio file

selectedFile = fullfile(folderPath, char(app.TrackDropDown.Value));

% Acquiring the audio file

app.fileReader = dsp.AudioFileReader(selectedFile, 'SamplesPerFrame', 1024);

app.deviceWriter = audioDeviceWriter('SampleRate', app.fileReader.SampleRate);

app.PLAYButton.Text = 'PAUSE';

app.isPlay = 1;

app.sampleRate = app.fileReader.SampleRate;

Fs = app.fs;

%Initializing the delays

xbuffer1 = 0;

xbuffer2 = 0;

ybuffer1 = zeros(10,3);

ybuffer2 = zeros(10,3);

S1 = app.Slider\_1.Value;

S2 = app.Slider\_2.Value;

S3 = app.Slider\_3.Value;

S4 = app.Slider\_4.Value;

S5 = app.Slider\_5.Value;

Mrplus = Mrp(app,Fs);

b = num(app,target(app,S1,S2,S3,S4,S5),Mrplus);

dF = Fs/1024;

f = -Fs/2:dF:Fs/2-dF;

i = 0;

%Playback loop

while ~isDone(app.fileReader)

%Acquiring the succeeding frame

xk = app.fileReader();

if length(xk(1,:))~=2

xk = [xk,xk];

xk1 = xk(:,1)';

xk2 = xk(:,2)';

else

xk1 = xk(:,1)';

xk2 = xk(:,2)';

end

if strcmp(app.PLAYButton.Text,'PLAY') == 1

%Pause loop

while strcmp(app.PLAYButton.Text,'PLAY') == 1 && ...

app.isStop == 0

pause(1);

end

end

pause(0);

%Checking if slider configuration changed

if app.Slider\_1.Value ~= S1 || app.Slider\_2.Value ~= S2 || app.Slider\_3.Value ~= S3 || S4 ~= app.Slider\_4.Value || S5 ~= app.Slider\_5.Value

S1 = app.Slider\_1.Value;

S2 = app.Slider\_2.Value;

S3 = app.Slider\_3.Value;

S4 = app.Slider\_4.Value;

S5 = app.Slider\_5.Value;

%Calculating the new filters

Mrplus = Mrp(app,Fs);

b = num(app,target(app,S1,S2,S3,S4,S5), Mrplus);

end

%Filtering process

for n=1:11

if n<11

ykNew1(n,:) = filterNew(app,b((2\*n-1):(2\*n)),xk1,ybuffer1(n,:),xbuffer1,Fs,n);

ykNew2(n,:) = filterNew(app,b((2\*n-1):(2\*n)),xk2,ybuffer2(n,:),xbuffer2,Fs,n);

else

ykNew1(n,:) = xk1\*b(22);

ykNew2(n,:) = xk2\*b(22);

end

end

yk1=0;

yk2=0;

for n=1:11

yk1=yk1 + ykNew1(n,:);

yk2=yk2 + ykNew2(n,:);

end

%Playback of frame

app.deviceWriter([0.25\*yk1',0.25\*yk2']);

%Delay updates

xbuffer1 = flip(xk1(length(xk1)-1:length(xk1)));

xbuffer2 = flip(xk2(length(xk2)-1:length(xk2)));

for n=1:10

ybuffer1(n,:)=flip(ykNew1(n,...

(length(ykNew1(n,:))-2):(length(ykNew1(n,:)))));

ybuffer2(n,:)=flip(ykNew2(n,...

(length(ykNew2(n,:))-2):(length(ykNew2(n,:)))));

end

if app.isStop == 1

release(app.fileReader);

release(app.deviceWriter);

app.PLAYButton.Text = 'PLAY';

app.isPlay = 0;

app.isStop = 0;

end

i = i+1;

end

release(app.fileReader);

release(app.deviceWriter);

app.PLAYButton.Text = 'PLAY';

app.isPlay = 0;

elseif strcmp(app.PLAYButton.Text,'PLAY') && ...

app.isPlay == 1

app.PLAYButton.Text = 'PAUSE';

elseif app.isStop == 1

release(app.fileReader);

release(app.deviceWriter);

app.PLAYButton.Text = 'PLAY';

app.isPlay = 0;

app.isStop = 0;

else

app.PLAYButton.Text = 'PLAY';

end

end

% Button pushed function: RESETButton

function RESETButtonPushed(app, event)

% Stop the song if it is currently playing

if app.isPlay

app.isStop = 1;

end

% Clear stored data

app.TrackDropDown.Value = {}; % Clear the selected audio file path

app.audioSignal = [];

app.sampleRate = [];

release(app.fileReader);

release(app.deviceWriter);

% Reset sliders and edit fields

for i = 1:5

app.(sprintf('Slider\_%d', i)).Value = 0;

app.(sprintf('EditField\_%d', i)).Value = num2str(0);

% Clear both plot and axes

plot(app.(sprintf('UIAxes%d', i)), NaN, NaN);

cla(app.(sprintf('UIAxes%d', i)));

end

end

end

% Component initialization

methods (Access = private)

% Create UIFigure and components

function createComponents(app)

% Create UIFigure and hide until all components are created

app.UIFigure = uifigure('Visible', 'off');

app.UIFigure.Position = [100 100 1204 553];

app.UIFigure.Name = '5 Band Audio Equaliser';

% Load background image

backgroundImage = imread('D:\NUST EME\5th Semester\Signals\_Systems\Lab\Project\Background.jpg');

% Set the background image

app.UIFigure.Color = 'none'; % Make the figure background transparent

app.UIFigure.Position = [100 100 1204 553];

% Create an axes to display the background image

app.UIAxesBackground = uiaxes(app.UIFigure, 'Visible', 'off');

app.UIAxesBackground.PlotBoxAspectRatio = [size(backgroundImage, 2) size(backgroundImage, 1) 1];

app.UIAxesBackground.Position = [-100 -180 1500 820];

imshow(backgroundImage, 'Parent', app.UIAxesBackground);

% Create UIAxes

app.UIAxes1 = uiaxes(app.UIFigure);

title(app.UIAxes1, 'Input Signal')

xlabel(app.UIAxes1, 'X')

ylabel(app.UIAxes1, 'Y')

zlabel(app.UIAxes1, 'Z')

app.UIAxes1.AmbientLightColor = [0 0 0];

app.UIAxes1.Box = 'on';

app.UIAxes1.XGrid = 'on';

app.UIAxes1.XMinorGrid = 'on';

app.UIAxes1.YGrid = 'on';

app.UIAxes1.YMinorGrid = 'on';

app.UIAxes1.Position = [26 299 300 197];

% Create UIAxes2

app.UIAxes2 = uiaxes(app.UIFigure);

title(app.UIAxes2, 'Output Signal')

xlabel(app.UIAxes2, 'X')

ylabel(app.UIAxes2, 'Y')

zlabel(app.UIAxes2, 'Z')

app.UIAxes2.Box = 'on';

app.UIAxes2.XGrid = 'on';

app.UIAxes2.XMinorGrid = 'on';

app.UIAxes2.YGrid = 'on';

app.UIAxes2.YMinorGrid = 'on';

app.UIAxes2.Position = [349 299 292 197];

% Create UIAxes3

app.UIAxes3 = uiaxes(app.UIFigure);

title(app.UIAxes3, 'Input Spectrum')

xlabel(app.UIAxes3, 'X')

ylabel(app.UIAxes3, 'Y')

zlabel(app.UIAxes3, 'Z')

app.UIAxes3.PlotBoxAspectRatio = [2.69491525423729 1 1];

app.UIAxes3.Box = 'on';

app.UIAxes3.XGrid = 'on';

app.UIAxes3.XMinorGrid = 'on';

app.UIAxes3.YGrid = 'on';

app.UIAxes3.YMinorGrid = 'on';

app.UIAxes3.Position = [20 6 313 265];

% Create UIAxes4

app.UIAxes4 = uiaxes(app.UIFigure);

title(app.UIAxes4, 'Output Spectrum')

xlabel(app.UIAxes4, 'X')

ylabel(app.UIAxes4, 'Y')

zlabel(app.UIAxes4, 'Z')

app.UIAxes4.PlotBoxAspectRatio = [2.69491525423729 1 1];

app.UIAxes4.Box = 'on';

app.UIAxes4.XGrid = 'on';

app.UIAxes4.XMinorGrid = 'on';

app.UIAxes4.YGrid = 'on';

app.UIAxes4.YMinorGrid = 'on';

app.UIAxes4.Position = [359 1 292 276];

% Create UIAxes5

app.UIAxes5 = uiaxes(app.UIFigure);

title(app.UIAxes5, 'Characteristic Frequency')

xlabel(app.UIAxes5, 'X')

ylabel(app.UIAxes5, 'Y')

zlabel(app.UIAxes5, 'Z')

app.UIAxes5.Box = 'on';

app.UIAxes5.XGrid = 'on';

app.UIAxes5.XMinorGrid = 'on';

app.UIAxes5.YGrid = 'on';

app.UIAxes5.YMinorGrid = 'on';

app.UIAxes5.Position = [666 64 365 189];

% Create Slider\_1

app.Slider\_1 = uislider(app.UIFigure);

app.Slider\_1.Limits = [-12 12];

app.Slider\_1.MajorTicks = [-12 -9 -6 -3 0 3 6 9 12];

app.Slider\_1.MajorTickLabels = {'-12', '-9', '-6', '-3', '0', '3', '6', '9', '12'};

app.Slider\_1.Orientation = 'vertical';

app.Slider\_1.ValueChangedFcn = createCallbackFcn(app, @Slider\_1ValueChanged, true);

app.Slider\_1.MinorTicks = [-12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12];

app.Slider\_1.Position = [776 358 3 150];

app.Slider\_1.Value = 0; % Set the default value to 0

% Create Band1fc63HzLabel

app.Band1fc63HzLabel = uilabel(app.UIFigure);

app.Band1fc63HzLabel.HorizontalAlignment = 'center';

app.Band1fc63HzLabel.Position = [762 279 59 30];

app.Band1fc63HzLabel.Text = {'Band 1 '; 'fc = 63 Hz'};

% Create EditField

app.EditField\_1 = uieditfield(app.UIFigure, 'text');

app.EditField\_1.Position = [762 321 55 22];

app.EditField\_1.ValueChangedFcn = createCallbackFcn(app, @EditField\_1ValueChanged, true);

app.EditField\_1.Value = '0'; % Set the default value to '0'

% Create Band2fc250HzLabel

app.Band2fc250HzLabel = uilabel(app.UIFigure);

app.Band2fc250HzLabel.HorizontalAlignment = 'center';

app.Band2fc250HzLabel.Position = [842 280 66 30];

app.Band2fc250HzLabel.Text = {'Band 2 '; 'fc = 250 Hz'};

% Create Slider\_2

app.Slider\_2 = uislider(app.UIFigure);

app.Slider\_2.Limits = [-12 12];

app.Slider\_2.MajorTicks = [-12 -9 -6 -3 0 3 6 9 12];

app.Slider\_2.MajorTickLabels = {'-12', '-9', '-6', '-3', '0', '3', '6', '9', '12'};

app.Slider\_2.Orientation = 'vertical';

app.Slider\_2.ValueChangedFcn = createCallbackFcn(app, @Slider\_2ValueChanged, true);

app.Slider\_2.MinorTicks = [-12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12];

app.Slider\_2.Position = [857 358 3 150];

app.Slider\_2.Value = 0; % Set the default value to 0

% Create EditField\_2

app.EditField\_2 = uieditfield(app.UIFigure, 'text');

app.EditField\_2.Position = [848 321 55 22];

app.EditField\_2.ValueChangedFcn = createCallbackFcn(app, @EditField\_2ValueChanged, true);

app.EditField\_2.Value = '0'; % Set the default value to '0'

% Create Band3fc1000HzLabel

app.Band3fc1000HzLabel = uilabel(app.UIFigure);

app.Band3fc1000HzLabel.HorizontalAlignment = 'center';

app.Band3fc1000HzLabel.Position = [925 280 73 30];

app.Band3fc1000HzLabel.Text = {'Band 3 '; 'fc = 1000 Hz'};

% Create Slider\_3

app.Slider\_3 = uislider(app.UIFigure);

app.Slider\_3.Limits = [-12 12];

app.Slider\_3.MajorTicks = [-12 -9 -6 -3 0 3 6 9 12];

app.Slider\_3.MajorTickLabels = {'-12', '-9', '-6', '-3', '0', '3', '6', '9', '12'};

app.Slider\_3.Orientation = 'vertical';

app.Slider\_3.ValueChangedFcn = createCallbackFcn(app, @Slider\_3ValueChanged, true);

app.Slider\_3.MinorTicks = [-12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12];

app.Slider\_3.Position = [941 358 3 150];

app.Slider\_3.Value = 0; % Set the default value to 0

% Create EditField\_3

app.EditField\_3 = uieditfield(app.UIFigure, 'text');

app.EditField\_3.Position = [932 322 55 22];

app.EditField\_3.ValueChangedFcn = createCallbackFcn(app, @EditField\_3ValueChanged, true);

app.EditField\_3.Value = '0'; % Set the default value to '0'

% Create Band4fc4000HzLabel

app.Band4fc4000HzLabel = uilabel(app.UIFigure);

app.Band4fc4000HzLabel.HorizontalAlignment = 'center';

app.Band4fc4000HzLabel.Position = [1017 278 69 30];

app.Band4fc4000HzLabel.Text = {'Band 4 '; 'fc = 4000Hz'};

% Create Slider\_4

app.Slider\_4 = uislider(app.UIFigure);

app.Slider\_4.Limits = [-12 12];

app.Slider\_4.MajorTicks = [-12 -9 -6 -3 0 3 6 9 12];

app.Slider\_4.MajorTickLabels = {'-12', '-9', '-6', '-3', '0', '3', '6', '9', '12'};

app.Slider\_4.Orientation = 'vertical';

app.Slider\_4.ValueChangedFcn = createCallbackFcn(app, @Slider\_4ValueChanged, true);

app.Slider\_4.MinorTicks = [-12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12];

app.Slider\_4.Position = [1031 359 3 150];

app.Slider\_4.Value = 0; % Set the default value to 0

% Create EditField\_4

app.EditField\_4 = uieditfield(app.UIFigure, 'text');

app.EditField\_4.Position = [1022 324 55 22];

app.EditField\_4.ValueChangedFcn = createCallbackFcn(app, @EditField\_4ValueChanged, true);

app.EditField\_4.Value = '0'; % Set the default value to '0'

% Create Band5fc16000HzLabel

app.Band5fc16000HzLabel = uilabel(app.UIFigure);

app.Band5fc16000HzLabel.HorizontalAlignment = 'center';

app.Band5fc16000HzLabel.Position = [1102 280 79 30];

app.Band5fc16000HzLabel.Text = {'Band 5 '; 'fc = 16000 Hz'};

% Create Slider\_5

app.Slider\_5 = uislider(app.UIFigure);

app.Slider\_5.Limits = [-12 12];

app.Slider\_5.MajorTicks = [-12 -9 -6 -3 0 3 6 9 12];

app.Slider\_5.MajorTickLabels = {'-12', '-9', '-6', '-3', '0', '3', '6', '9', '12'};

app.Slider\_5.Orientation = 'vertical';

app.Slider\_5.ValueChangedFcn = createCallbackFcn(app, @Slider\_5ValueChanged, true);

app.Slider\_5.MinorTicks = [-12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12];

app.Slider\_5.Position = [1127 358 3 150];

app.Slider\_5.Value = 0; % Set the default value to 0

% Create EditField\_5

app.EditField\_5 = uieditfield(app.UIFigure, 'text');

app.EditField\_5.Position = [1114 322 55 22];

app.EditField\_5.ValueChangedFcn = createCallbackFcn(app, @EditField\_5ValueChanged, true);

app.EditField\_5.Value = '0'; % Set the default value to '0'

% Create LOADButton

app.LOADButton = uibutton(app.UIFigure, 'push');

app.LOADButton.ButtonPushedFcn = createCallbackFcn(app, @LOADButtonPushed, true);

app.LOADButton.BackgroundColor = [0 0.4471 0.7412];

app.LOADButton.FontColor = [1 1 1];

app.LOADButton.Position = [667 10 107 41];

app.LOADButton.Text = 'LOAD';

% Create PLAYButton

app.PLAYButton = uibutton(app.UIFigure, 'push');

app.PLAYButton.ButtonPushedFcn = createCallbackFcn(app, @PLAYButtonPushed, true);

app.PLAYButton.BackgroundColor = [0 0.4471 0.7412];

app.PLAYButton.FontColor = [1 1 1];

app.PLAYButton.Position = [787 10 92 41];

app.PLAYButton.Text = 'PLAY';

% Create RESETButton

app.RESETButton = uibutton(app.UIFigure, 'push');

app.RESETButton.ButtonPushedFcn = createCallbackFcn(app, @RESETButtonPushed, true);

app.RESETButton.BackgroundColor = [0 0.4471 0.7412];

app.RESETButton.FontColor = [1 1 1];

app.RESETButton.Position = [888 10 107 41];

app.RESETButton.Text = 'RESET';

% Create PLOTButton

app.PLOTButton = uibutton(app.UIFigure, 'push');

app.PLOTButton.ButtonPushedFcn = createCallbackFcn(app, @PLOTButtonPushed, true);

app.PLOTButton.BackgroundColor = [0 0.4471 0.7412];

app.PLOTButton.FontColor = [1 1 1];

app.PLOTButton.Position = [1002 10 107 41];

app.PLOTButton.Text = 'PLOT';

% Create SongSelectDropDownLabel

app.SongSelectDropDownLabel = uilabel(app.UIFigure);

app.SongSelectDropDownLabel.HorizontalAlignment = 'right';

app.SongSelectDropDownLabel.Position = [1025 208 70 22];

app.SongSelectDropDownLabel.Text = 'Song Select';

% Create SongSelectDropDown

app.TrackDropDown = uidropdown(app.UIFigure);

app.TrackDropDown.Position = [1110 206 94 26];

% Show the figure after all components are created

app.UIFigure.Visible = 'on';

end

end

% App creation and deletion

methods (Access = public)

% Construct app

function app = Project\_GUI\_Code

% Close all existing instances of the app

existingApps = findall(0, 'Type', 'figure', 'Name', '5 Band Audio Equaliser');

delete(existingApps);

% Create UIFigure and components

createComponents(app)

% Register the app with App Designer

registerApp(app, app.UIFigure)

if nargout == 0

clear app

end

end

% Code that executes before app deletion

function delete(app)

% Delete UIFigure when app is deleted

delete(app.UIFigure)

end

end

end

**Code Explanation:**

The code reads an audio file, extracts the audio data, and plots the waveform and spectrum of the audio signal. Here's a breakdown:

**1. Loading Audio File:**

- The user selects an audio file through a dialog box.

- The code then reads this audio file and extracts the audio data.

**2. Plotting Audio Waveform:**

- The audio waveform is the graphical representation of how the audio signal varies over time.

- A figure (a window for displaying plots) is created to show the waveform plot.

- The audio signal is plotted on this figure, providing a visual representation of the sound wave.

**3. Plotting Audio Spectrum:**

- The audio spectrum represents how much of the audio signal exists at different frequencies.

- Another figure is created to display the spectrum plot.

- The Fast Fourier Transform (FFT) is applied to the audio signal to convert it from the time domain to the frequency domain, and the resulting spectrum is plotted.

The code is responsible for creating a graphical user interface (GUI) for controlling a 5-band audio equalizer. The equalizer allows the user to adjust different frequency bands of the audio signal. Let's break it down:

**4. Overall GUI Design:**

- A GUI figure is created with various components, including sliders, buttons, and dropdowns.

- The figure has background images, axes for plotting signals and spectra, sliders for each frequency band, and buttons for actions like loading, playing, resetting, and plotting.

**5. Frequency Bands:**

- The equalizer is divided into 5 frequency bands, each centered around a specific frequency.

- For each band, there is a slider to adjust the gain (volume) and an edit field to display and manually set the gain value.

**6. Plotting Signals and Spectra:**

- The GUI has axes for plotting the input and output audio signals and their respective spectra.

- This helps visualize the effect of equalization on the audio signal.

**7. Buttons for Interaction:**

- There are buttons for loading an audio file, playing it, resetting the equalizer settings, and plotting the audio signals and spectra.

**8. Dropdown for Song Selection:**

- A dropdown menu is provided to select different audio tracks for processing.

**In-Depth Explanation:**

A screenshot of a computer

Description automatically generatedThe combined code serves as an interactive tool for both analyzing and modifying audio signals. Users can load different audio tracks, visualize their waveforms and spectra, and interactively adjust the equalizer settings to observe the impact on the audio output. This GUI provides a user-friendly interface for experimenting with audio processing, making it suitable for educational purposes, research, or practical applications in audio engineering.

Figure 8: GUI Interface

**GUI Interface Images**

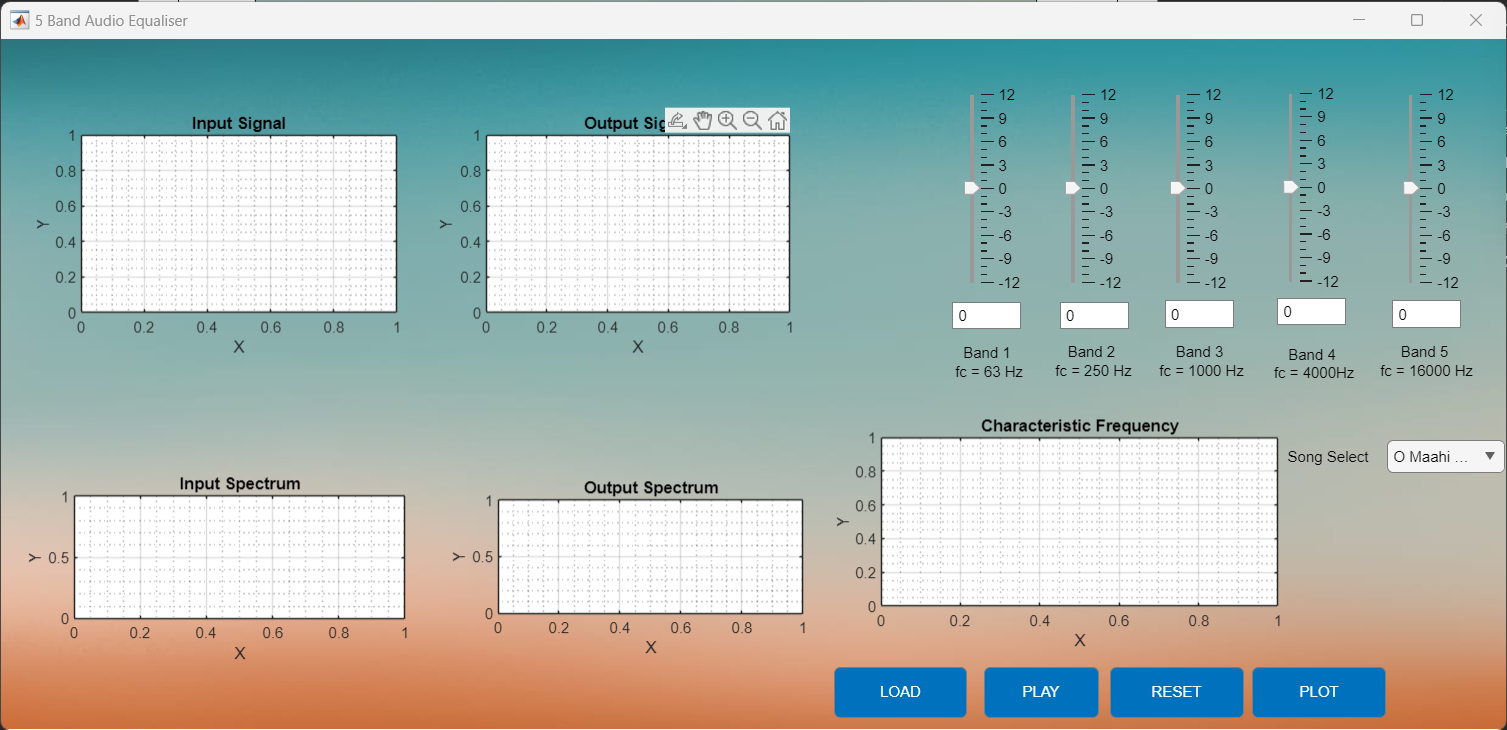
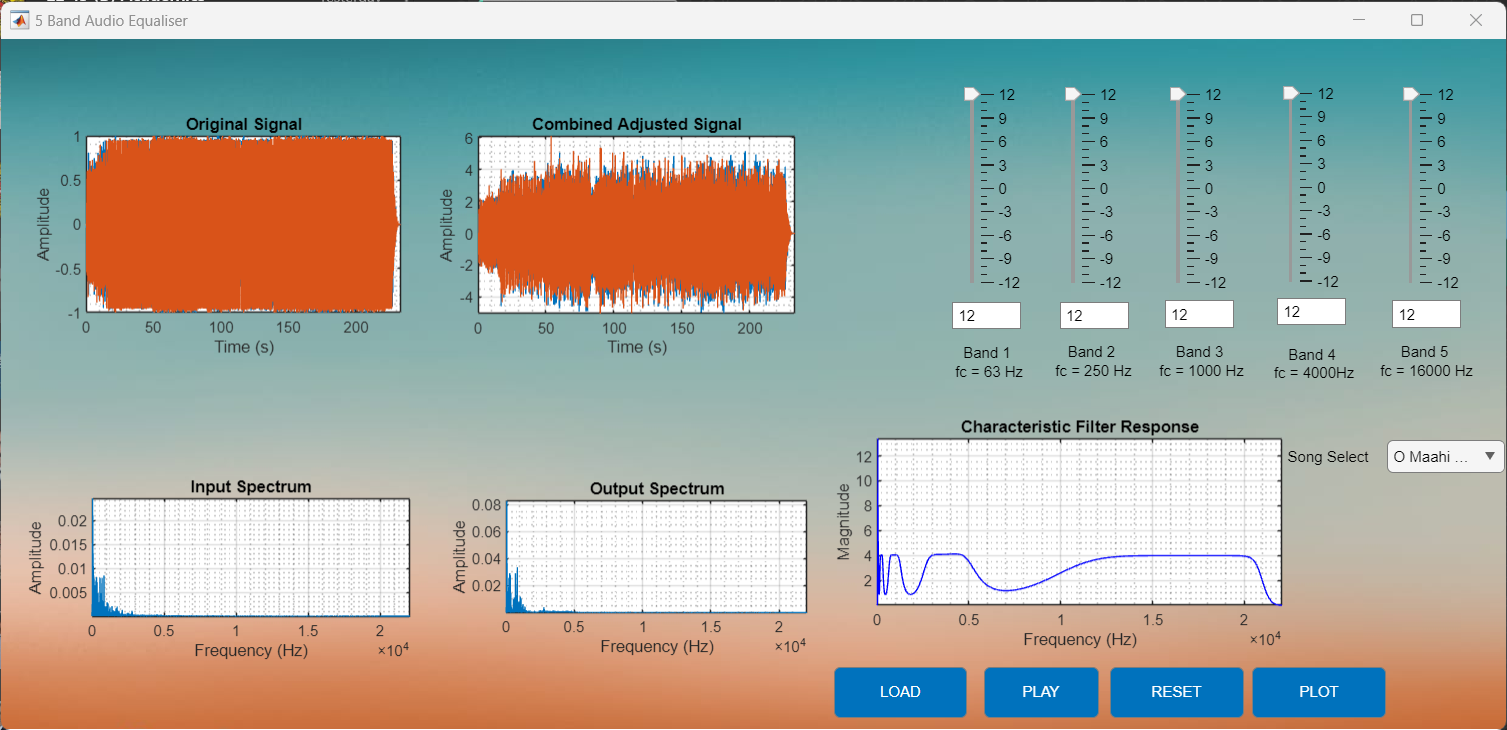


Figure 9: GUI after the Song has been loaded by pressing 'LOAD' button.

A screenshot of a computer

Description automatically generated

Figure 10: Signal Plotted without any gain

A screenshot of a computer

Description automatically generated

Figure 12: Signal plotted with full positive gain

Figure 11: Signal plotted with full negative gain

In summary, this project merged theoretical concepts from our "Signals and Systems" course with practical applications. The first part focused on analyzing audio signals, providing insights into temporal and frequency characteristics. The second part introduced a user-friendly 5-band audio equalizer through a graphical interface, showcasing real-time signal manipulation.

**Conclusion**

This project bridged theory and practice, emphasizing the application of signal processing techniques like Fourier analysis and equalization. The graphical interface highlighted the importance of user-friendly tools in signal processing. The hands-on experience reinforced MATLAB programming and signal processing skills while deepening our understanding of audio quality's dependence on signal manipulation.

Overall, this project was a valuable exercise, enriching our grasp of signals and systems in the context of audio engineering.

1. <https://www.diva-portal.org/smash/get/diva2:1334188/FULLTEXT02.pdf>
2. <https://www.mathworks.com/matlabcentral/fileexchange/23982-digital-audio-equalizer>
3. <https://www.youtube.com/watch?v=Z7urwX82Z8g>
4. <https://www.mathworks.com/help/audio/ug/equalization.html>
5. <https://www.mathworks.com/help/audio/ug/graphic-equalization.html>
6. <https://www.mathworks.com/help/audio/audio-processing-algorithm-design.html>
7. <https://github.com/splAcharya/AudioEqualizerMatlab_Simulink>
8. <https://github.com/MonicaSaid/audio-equalizer>
9. <https://github.com/mohamedmashaal/Digital-Audio-Equalizer/tree/master>

**References**