Spring 2021

Digital Signal Processing

AUDIO EQUALIZER

Karim Sameh	6169
Ahmad Abdelrahman	6543
Andrew Wahid	6578

Contents (click to follow link to title)

- Copy of the Code 2
- Sample Runs 7
- Analysis of Each Filter 10
- Figures of signals in Time and Frequency Domains 15

➢ Copy of the Code

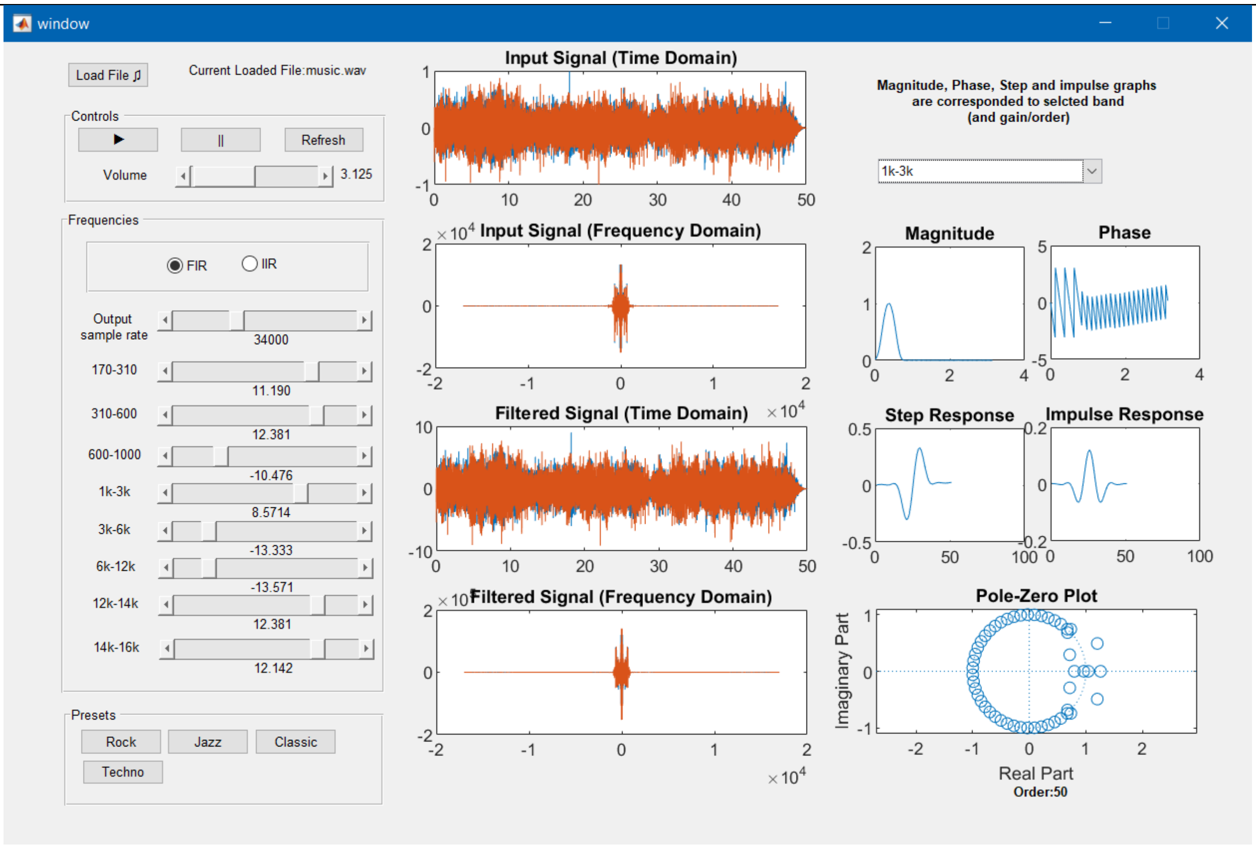
```
function varargout = window(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @window_OpeningFcn, ...
                   'gui_OutputFcn',  @window_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% Initializes Values
function window_OpeningFcn(hObject, eventdata, handles, varargin)
    handles.output = hObject;
    guidata(hObject, handles);
    global volume filterType selectedBand userSampleRate;
    selectedBand = 1;
    volume = 85;
    filterType = 'FIR';
    userSampleRate = 0;
    refreshFrequencyValues(hObject, 0, handles);

function varargout = window_OutputFcn(hObject, ~, handles)
varargout{1} = handles.output;

% Applies FIR filter to given signal/band
function [y, num, den] = applyFIR(wn, bandType, gain, signalY)
    num = fir1(50, wn, bandType);
    den = 1;
    yFiltered = filter(num, 1, signalY);
    y=gain*yFiltered;


% Applies IIR filter to given signal/band
function [y, num, den, N] = applyIIR(wn, bandType, gain, signalY)
    if (strcmp(bandType,'bandpass'))
        [N, ~] = cheb1ord(wn(1),wn(2),1,10); % or use manual Ns
    else
        N = 7;
    end
    [num, den] = cheby1(N, 0.5, wn, bandType);
    yFiltered = filter(num, den, signalY);
    y = gain * yFiltered;

% Calls filter functions for all bands in a loop and forms composite signal
```

```
function applyEqualizer(handles)
    global audioPathName selectedBand fs userSampleRate;
    [handles.y, Fsload] = audioread(audioPathName);
    bandpasses = [170 310 600 1000 3000 6000 12000 14000 16000];
    if (Fsload <= bandpasses(end)*2)
        fs = bandpasses(end)*2 + 2000;
    else
        fs = Fsload;
    end
    if (userSampleRate == 0)
        userSampleRate = fs;
    end
    handles.y = resample(handles.y, fs, Fsload);
    values = getSliderValues(handles);
    totalY = 0;
    for i=1:length(bandpasses)
        currentValue = bandpasses(i);
        currentGain = values(i);
        currentGain = db2mag(currentGain);
        bandType = 'bandpass';
        if (i>1)
            previousValue = bandpasses(i-1)+1;
            wn = [previousValue/(fs/2) currentValue/(fs/2)];
        else
            bandType = 'low';
            wn = currentValue/(fs/2);
        end
        if (get(handles.firRadio, 'Value'))
            [y, num, den] = applyFIR(wn, bandType, currentGain, handles.y);
            N = 50;
        else
            [y, num, den, N] = applyIIR(wn, bandType, currentGain, handles.y);
        end
        if (i == selectedBand)
            plotMagnitudePhase(num, den, handles.magAxes, handles.phaseAxes);
            plotStepImpulse(num, den, handles.impAxes, handles.stepAxes);
            plotPolesZeros(num, den, handles.polesZeros)
            set(handles.orderLabel, 'String', strcat('Order: ', int2str(N)));
        end
        totalY = totalY + y;
    end
    totalY = resample(totalY, userSampleRate, fs);
    plotTD(handles.y, handles.timeDY, 'Input Signal (Time Domain)');
    plotTD(totalY, handles.timeDYF, 'Filtered Signal (Time Domain)');
    %
    plotFD(fs, handles.y, handles.timeFY, 'Input Signal (Frequency Domain)');
    plotFD(fs, totalY, handles.timeFYF, 'Filtered Signal (Frequency Domain)');
    %
    global audioPlayer volume;
    audioPlayer = audioplayer(totalY*(volume/100), fs);


% Loads .wav File into program
function loadFileButton_Callback(hObject, ~, handles)
    global audioPathName userSampleRate;
    userSampleRate = 0;
    [fileName, pathName]=uigetfile({'*.wav'} , 'File Selector');
    audioPathName = strcat(pathName , fileName);
    set(handles.loadedFileLabel, 'String' , strcat('Current Loaded File: ',
fileName));
    applyEqualizer(handles);
    initializeSampleRate(handles);

% Sets output sample frequency as value taken from user into slider values
function initializeSampleRate(handles)
    global userSampleRate fs;
    userSampleRate = fs;
    set(handles.sampleRateSlider, 'min', fs/2);
    set(handles.sampleRateSlider, 'max', fs*2);
    set(handles.sampleRateSlider, 'value', fs);
    set(handles.lv_sampleRate, 'String', userSampleRate);

% Pauses music player
function pauseButton_Callback(hObject, ~, handles)
    global audioPlayer;
    applyEqualizer(handles);
    pause(audioPlayer);

% Resumes music player
function resumeButton_Callback(hObject, ~, handles)
    global audioPlayer;
    applyEqualizer(handles);
    play(audioPlayer);

% Refreshes inputs and continues playback
function refreshButton_Callback(hObject, ~, handles)
    global audioPlayer;
    currentTime = audioPlayer.CurrentSample;
    pause(audioPlayer);
    applyEqualizer(handles);
    play(audioPlayer, ceil(currentTime));

% Changes volume of music player
function volumeSlider_Callback(hObject, eventdata, handles)
    global volume;
    volume = get(handles.volumeSlider, 'value');
    set(handles.volumeLabel, 'String', volume);

% GUI Functions
function slidersCreate(hObject, ~, handles)
    if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor',[.9 .9 .9]);
    end

function x = getSliderValues(handles)
    x = [get(handles.slider_170, 'value') get(handles.slider_310, 'value')
get(handles.slider_600, 'value') get(handles.slider_1k, 'value')
get(handles.slider_3k, 'value') get(handles.slider_6k, 'value')
get(handles.slider_12k, 'value') get(handles.slider_14k, 'value')
get(handles.slider_16k, 'value')];
```

```
function refreshFrequencyValues(hObject, ~, handles)
    sliderValues = getSliderValues(handles);
    set(handles.lv_170, 'String', sliderValues(1));
    set(handles.lv_310, 'String', sliderValues(2));
    set(handles.lv_600, 'String', sliderValues(3));
    set(handles.lv_1k, 'String', sliderValues(4));
    set(handles.lv_3k, 'String', sliderValues(5));
    set(handles.lv_6k, 'String', sliderValues(6));
    set(handles.lv_12k, 'String', sliderValues(7));
    set(handles.lv_14k, 'String', sliderValues(8));
    set(handles.lv_16k, 'String', sliderValues(9));

function loadPreset(handles, v_170, v_310, v_600, v_1k, v_3k, v_6k, v_12k, v_14k,
v_16k)
    set(handles.slider_170, 'value', v_170);
    set(handles.slider_310, 'value', v_310);
    set(handles.slider_600, 'value', v_600);
    set(handles.slider_1k, 'value', v_1k);
    set(handles.slider_3k, 'value', v_3k);
    set(handles.slider_6k, 'value', v_6k);
    set(handles.slider_12k, 'value', v_12k);
    set(handles.slider_14k, 'value', v_14k);
    set(handles.slider_16k, 'value', v_16k);
    refreshFrequencyValues(0, 0, handles);

function rockButton_Callback(hObject, ~, handles)
    loadPreset(handles, 1, 2, 3, 4, 5, 6, 7, 8, 9)

function jazzButton_Callback(hObject, ~, handles)
    loadPreset(handles, -1, -2, -3, -4, -5, -6, -7, -8, -9)

function classicButton_Callback(hObject, ~, handles)
    loadPreset(handles, 5, 8, 1, 3, 6, 6, -20, 20, 15.225)

function techoButton_Callback(hObject, ~, handles)
    loadPreset(handles, 0, 0, 0, 0, 0, 0, 0, 0, 0)

% Plots given signal in time domain
function plotTD(y, axesObject, plotTitle)
    t = linspace(0,50,length(y));
    axes(axesObject);
    plot(t,y); title(plotTitle);

% Plots given signal in frequency domain
function plotFD(Fs, y, axesObject, plotTitle)
    Y = fftshift(fft(y));
    F = linspace(-Fs/2,Fs/2,length(y));
    axes(axesObject);
    plot(F,real(Y)); title(plotTitle);

% Plots magnitude and phase of given signal
function plotMagnitudePhase(numerator, denominator, magAxes, phaseAxes)
    [H,w] = freqz(numerator, denominator);
    magnitude = abs(H);
    phase = angle(H);
    axes(magAxes);
    plot(w,magnitude); title('Magnitude');
    axes(phaseAxes);
    plot(w,phase); title('Phase');

% Plots step and impulse responses of given signal
function plotStepImpulse(numerator, denominator, impAxes, stepAxes)
    axes(impAxes);
    plot(impz(numerator, denominator)); title('Impulse Response');
    axes(stepAxes);
    plot(stepz(numerator, denominator)); title('Step Response');

% Plots poles and zeros of given signal
function plotPolesZeros(numerator, denominator, polesAxes)
    z = roots(numerator);
    p = roots(denominator);
    axes(polesAxes);
    zplane(z,p); title('Pole-Zero Plot');

% GUI Functions
function bandMenu_Callback(hObject, eventdata, handles)
    global selectedBand;
    selectedBand = get(hObject,'Value');
    applyEqualizer(handles);


function bandMenu_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function sampleRateSlider_Callback(hObject, eventdata, handles)
    global userSampleRate;
    userSampleRate = ceil(get(hObject, 'value'));
    set(handles.lv_sampleRate, 'String', userSampleRate);
```

➢ Sample Runs

FIR FILTERS



window

Current Loaded File:music.wav

Volume 3.125

● FIR ○ IIR

Output sample rate 34000

170-310 11.190

310-600 12.381

600-1000 -10.476

1k-3k 8.5714

3k-6k -13.333

6k-12k -13.571

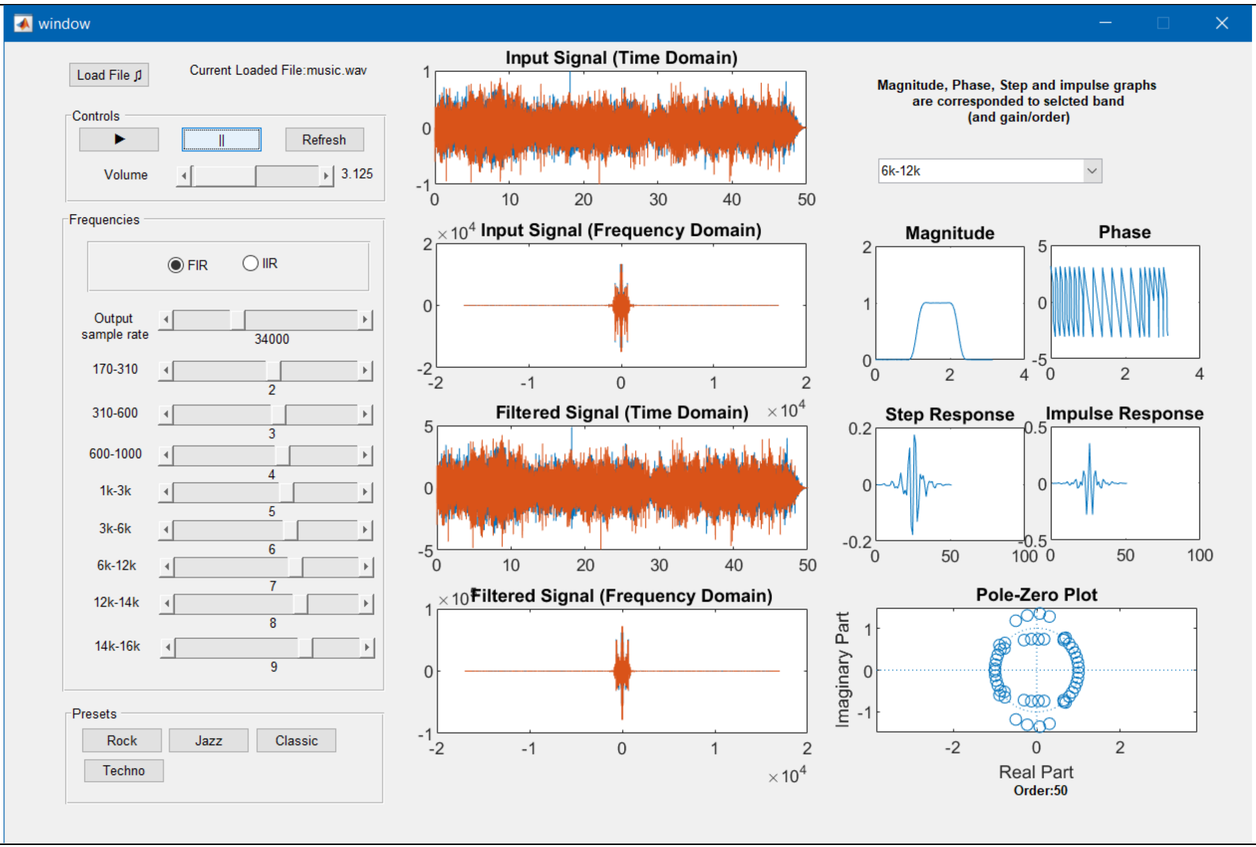
12k-14k 12.381

14k-16k 12.142

Magnitude, Phase, Step and impulse graphs are corresponded to selcted band (and gain/order)

1k-3k

Order:50



window

Current Loaded File:music.wav

Volume 3.125

● FIR ○ IIR

Output sample rate 34000

170-310 2

310-600 3

600-1000 4

1k-3k 5

3k-6k 6

6k-12k 7

12k-14k 8

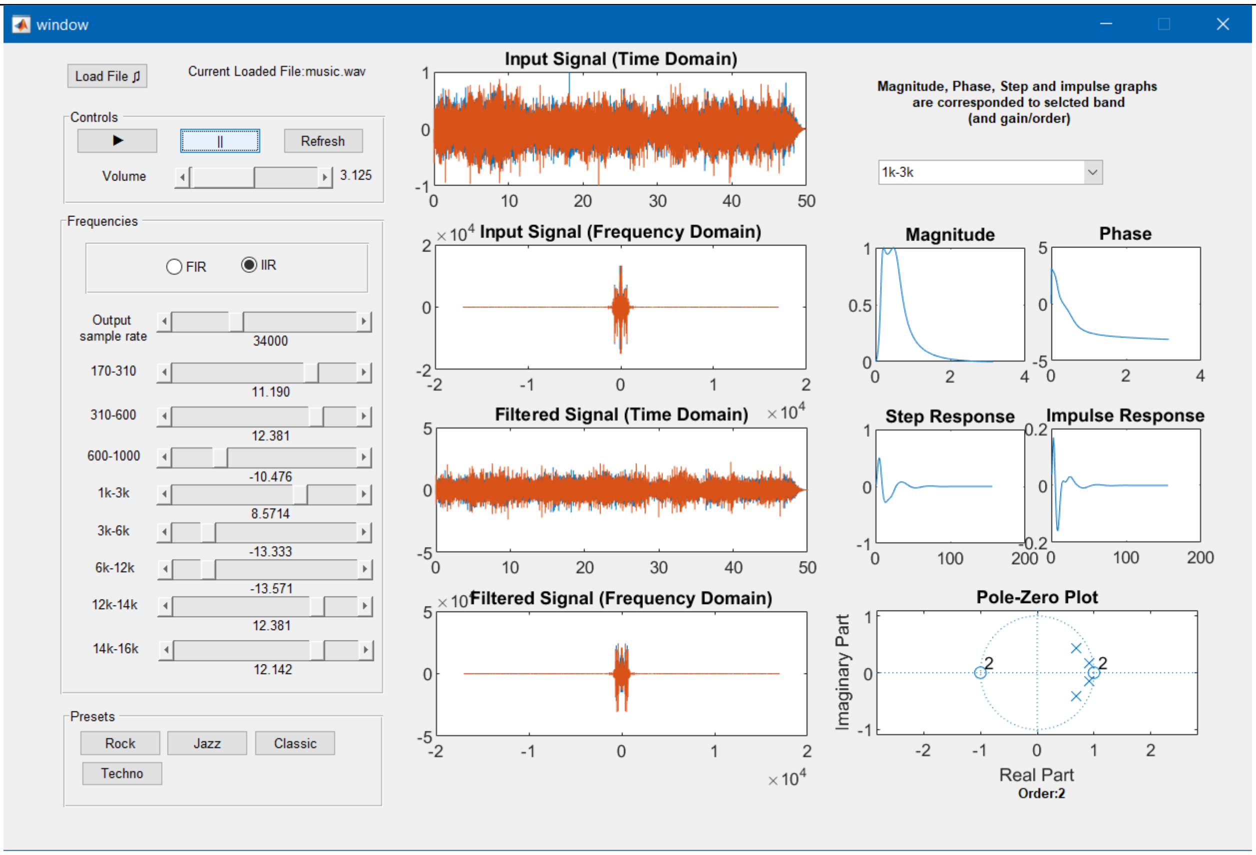
14k-16k 9

Magnitude, Phase, Step and impulse graphs are corresponded to selcted band (and gain/order)

6k-12k

Order:50

IIR FILTERS



Current Loaded File:music.wav

Volume 3.125

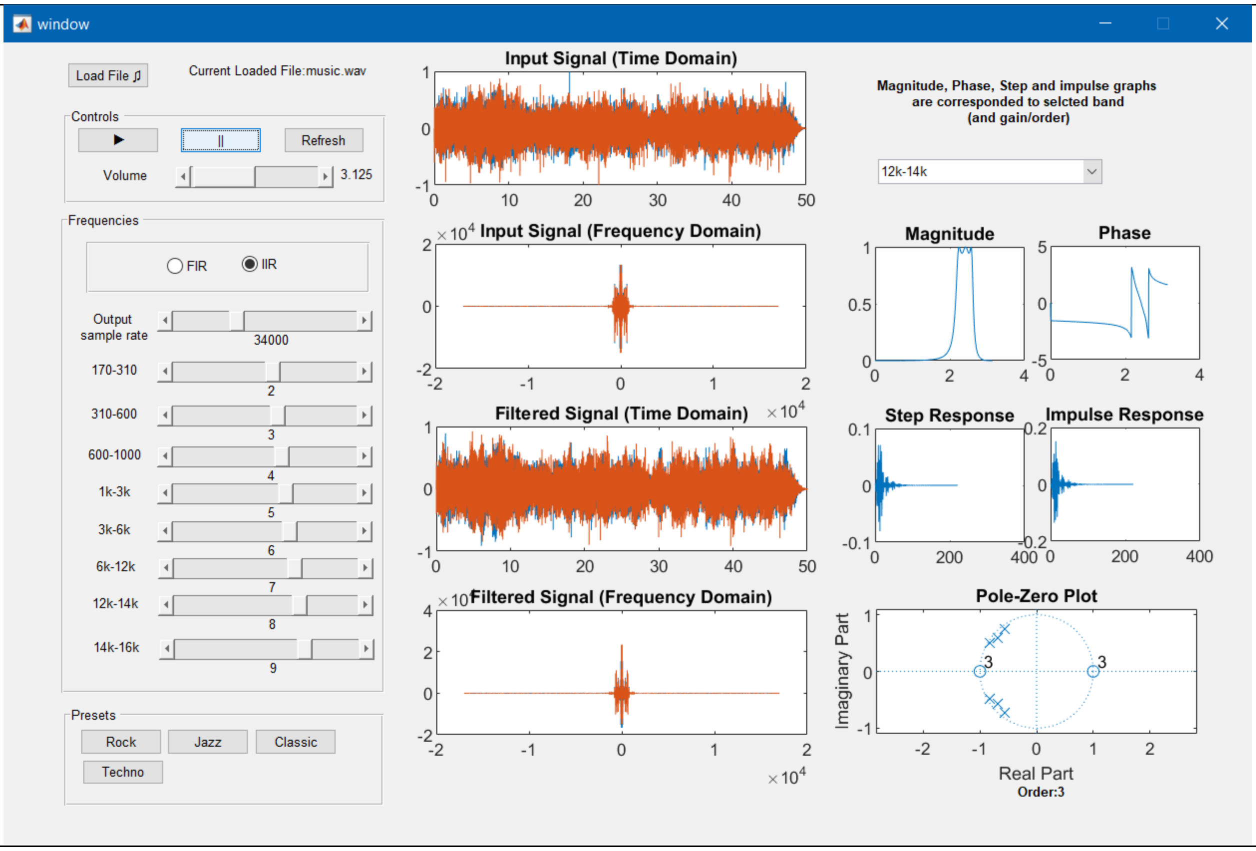
○ FIR ● IIR

Output sample rate	34000
170-310	11.190
310-600	12.381
600-1000	-10.476
1k-3k	8.5714
3k-6k	-13.333
6k-12k	-13.571
12k-14k	12.381
14k-16k	12.142

Magnitude, Phase, Step and impulse graphs are corresponded to selcted band (and gain/order)

1k-3k

Order:2



Current Loaded File:music.wav

Volume 3.125

○ FIR ● IIR

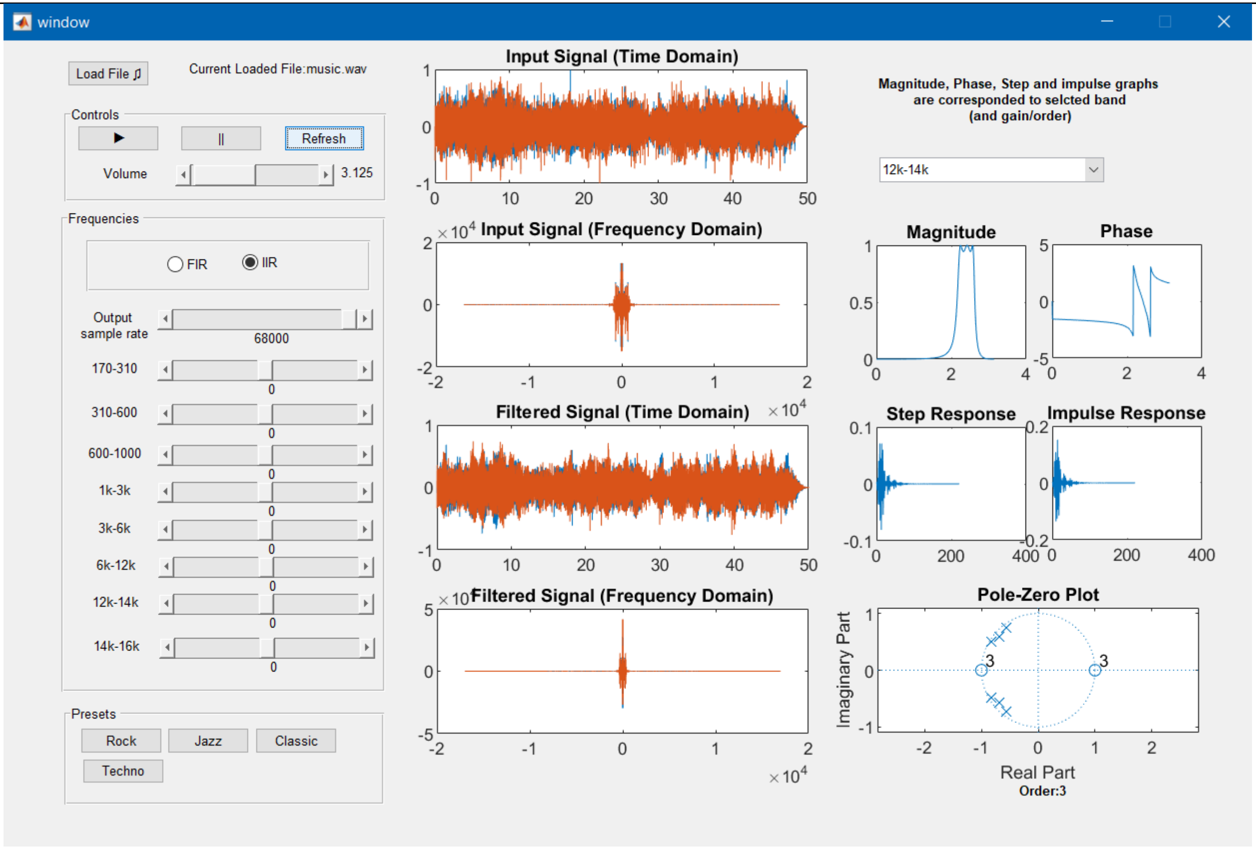
Output sample rate	34000
170-310	2
310-600	3
600-1000	4
1k-3k	5
3k-6k	6
6k-12k	7
12k-14k	8
14k-16k	9

Magnitude, Phase, Step and impulse graphs are corresponded to selcted band (and gain/order)

12k-14k

Order:3

Doubling Output Sample Rate



Current Loaded File:music.wav

Volume 3.125

○ FIR ● IIR

Output sample rate 68000

170-310 0

310-600 0

600-1000 0

1k-3k 0

3k-6k 0

6k-12k 0

12k-14k 0

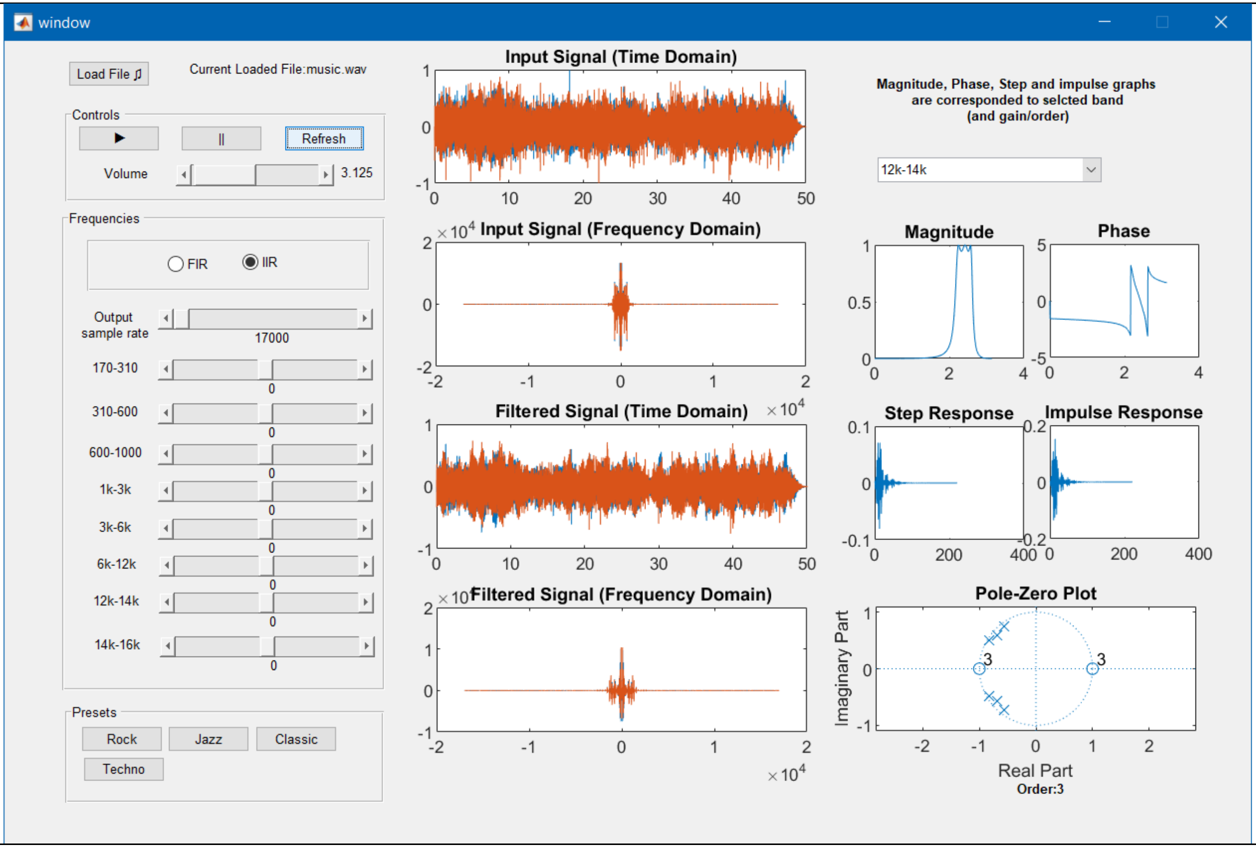
14k-16k 0

Magnitude, Phase, Step and impulse graphs are corresponded to selcted band (and gain/order)

12k-14k

Order:3

Halving Output Sample Rate



Current Loaded File:music.wav

Volume 3.125

○ FIR ● IIR

Output sample rate 17000

170-310 0

310-600 0

600-1000 0

1k-3k 0

3k-6k 0

6k-12k 0

12k-14k 0

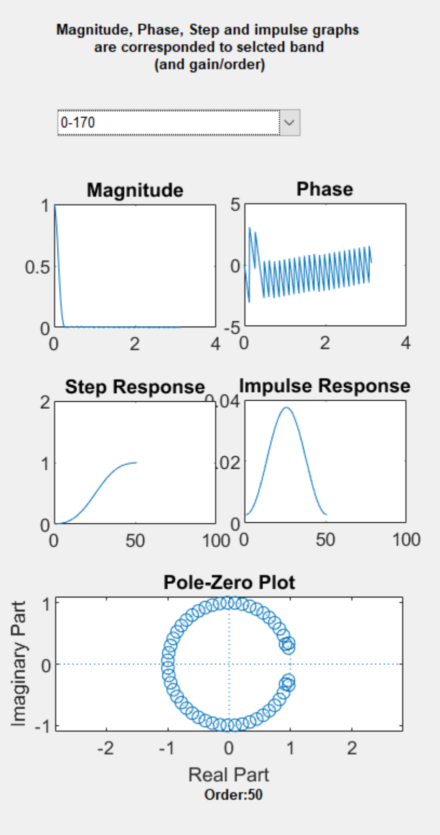
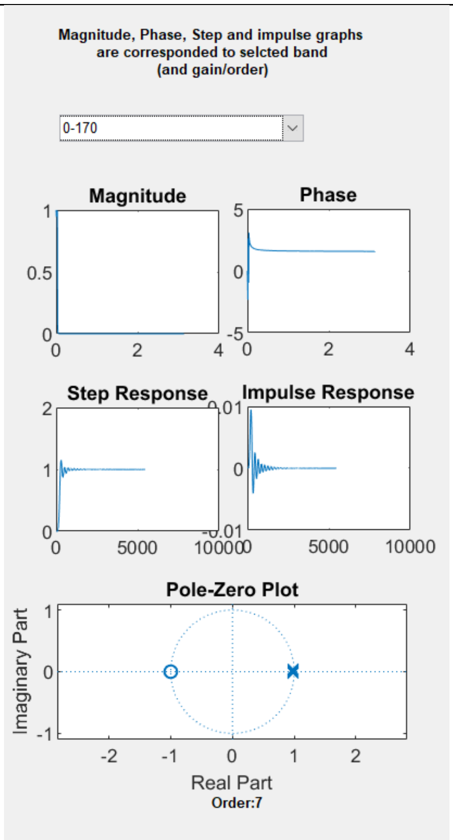
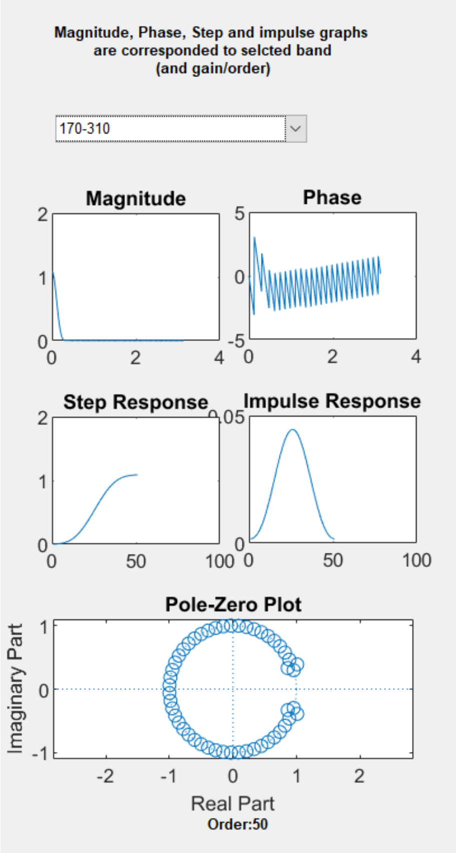
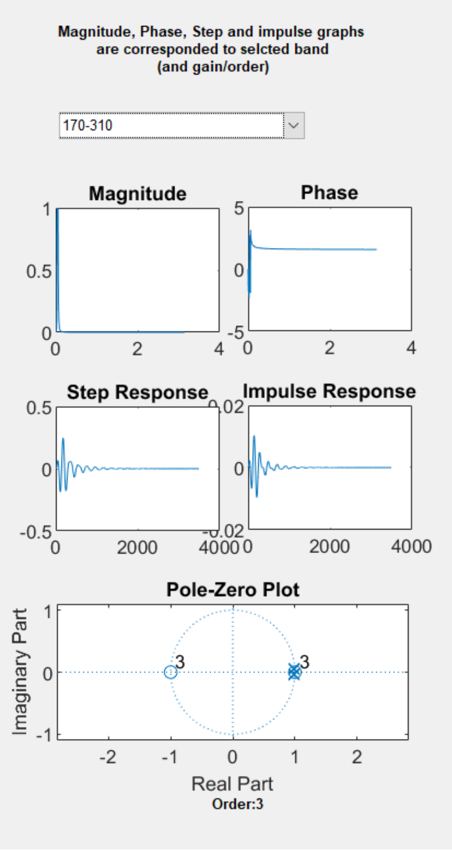
14k-16k 0

Magnitude, Phase, Step and impulse graphs are corresponded to selcted band (and gain/order)

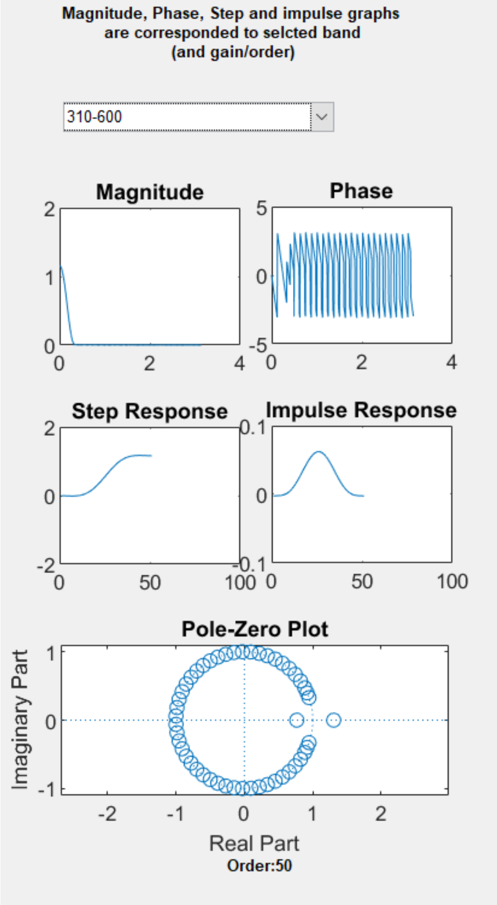
12k-14k

Order:3

➢ Analysis of Each Filter

FIR FILTERS	IIR FILTERS
0 - 170 Hz	
 Magnitude, Phase, Step and impulse graphs are corresponded to selcted band (and gain/order) 0-170 Magnitude Phase Step Response Impulse Response Pole-Zero Plot Imaginary Part Real Part 1 0.5 0 0 2 4 5 0 -5 2 1 0 0 50 100 0.04 0.02 0 -2 -1 0 1 2 Order:50	 Magnitude, Phase, Step and impulse graphs are corresponded to selcted band (and gain/order) 0-170 Magnitude Phase Step Response Impulse Response Pole-Zero Plot Imaginary Part Real Part 1 0.5 0 0 2 4 5 0 -5 2 1 0 0 5000 10000 0.01 0 -0.01 -2 -1 0 1 2 Order:7
170 - 310 Hz	
 Magnitude, Phase, Step and impulse graphs are corresponded to selcted band (and gain/order) 170-310 Magnitude Phase Step Response Impulse Response Pole-Zero Plot Imaginary Part Real Part 2 1 0 0 2 4 5 0 -5 0 50 100 0.05 0 -2 -1 0 1 2 Order:50	 Magnitude, Phase, Step and impulse graphs are corresponded to selcted band (and gain/order) 170-310 Magnitude Phase Step Response Impulse Response Pole-Zero Plot Imaginary Part Real Part 1 0.5 0 0 2 4 5 0 -5 0.5 0 -0.5 0 2000 4000 0.02 0 -0.02 3 -2 -1 0 1 2 Order:3

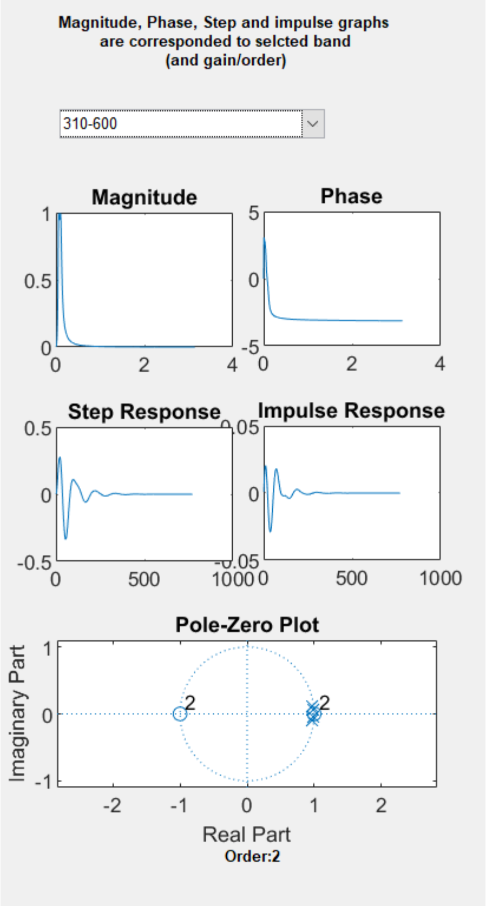
310 - 600 Hz



Magnitude, Phase, Step and impulse graphs
are corresponded to selcted band
(and gain/order)

310-600

Order:50

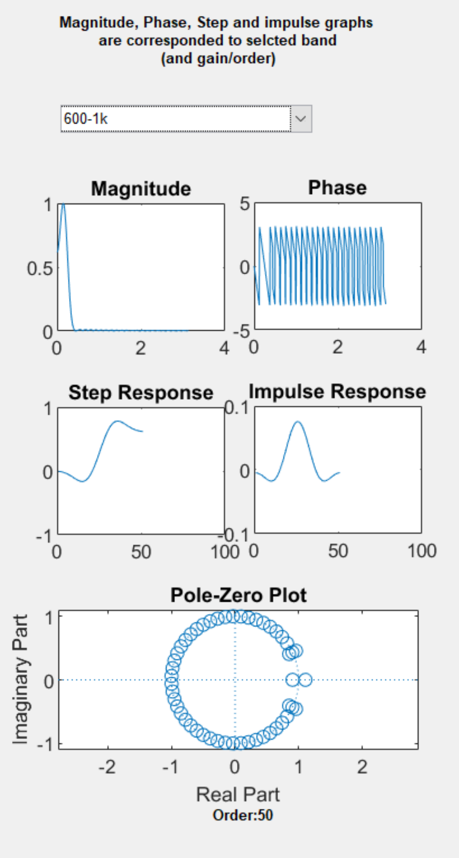


Magnitude, Phase, Step and impulse graphs
are corresponded to selcted band
(and gain/order)

310-600

Order:2

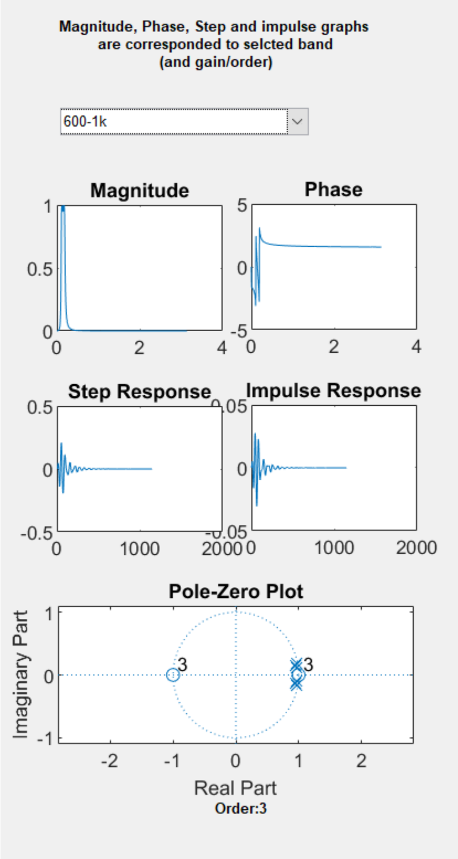
600 - 1000 Hz



Magnitude, Phase, Step and impulse graphs
are corresponded to selcted band
(and gain/order)

600-1k

Order:50

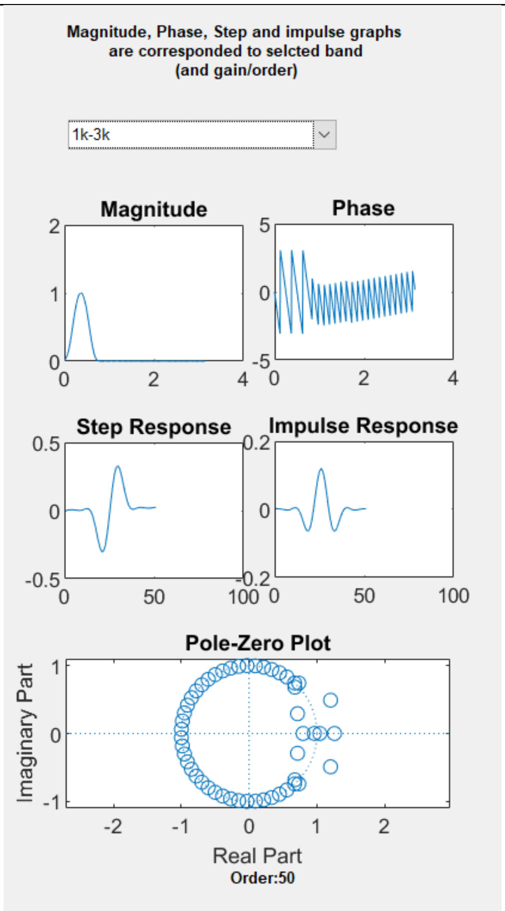


Magnitude, Phase, Step and impulse graphs
are corresponded to selcted band
(and gain/order)

600-1k

Order:3

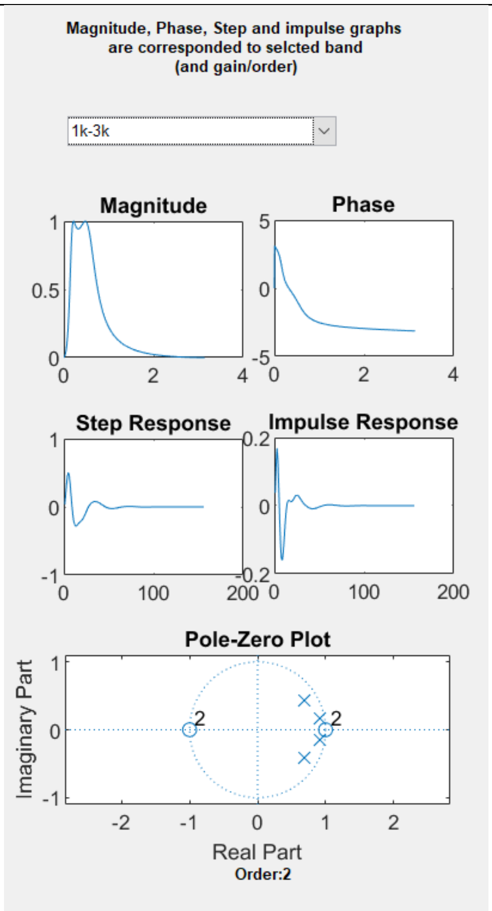
1 - 3 KHz



Magnitude, Phase, Step and impulse graphs
are corresponded to selcted band
(and gain/order)

1k-3k

Order:50

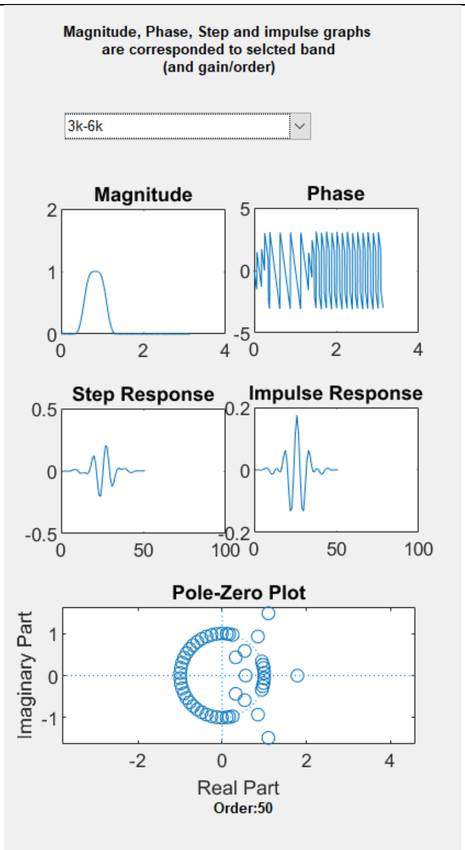


Magnitude, Phase, Step and impulse graphs
are corresponded to selcted band
(and gain/order)

1k-3k

Order:2

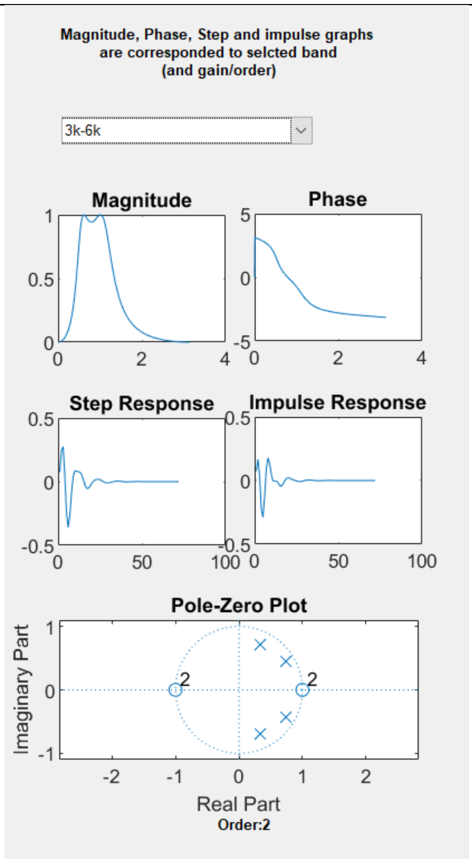
3 - 6 KHz



Magnitude, Phase, Step and impulse graphs
are corresponded to selcted band
(and gain/order)

3k-6k

Order:50

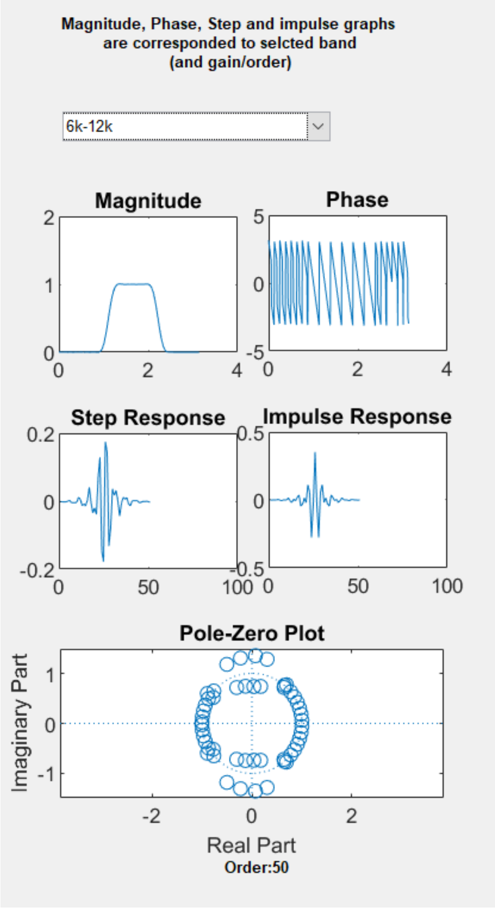


Magnitude, Phase, Step and impulse graphs
are corresponded to selcted band
(and gain/order)

3k-6k

Order:2

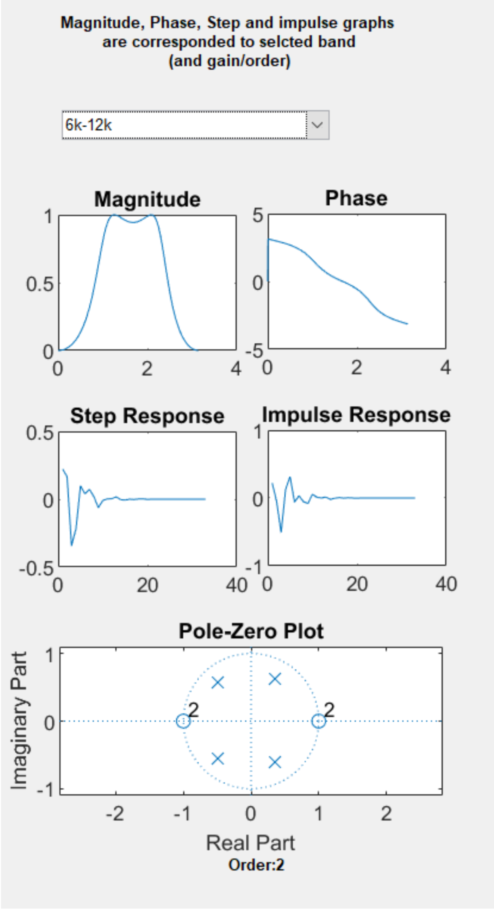
6 - 12 KHz



Magnitude, Phase, Step and impulse graphs
are corresponded to selcted band
(and gain/order)

6k-12k

Order:50

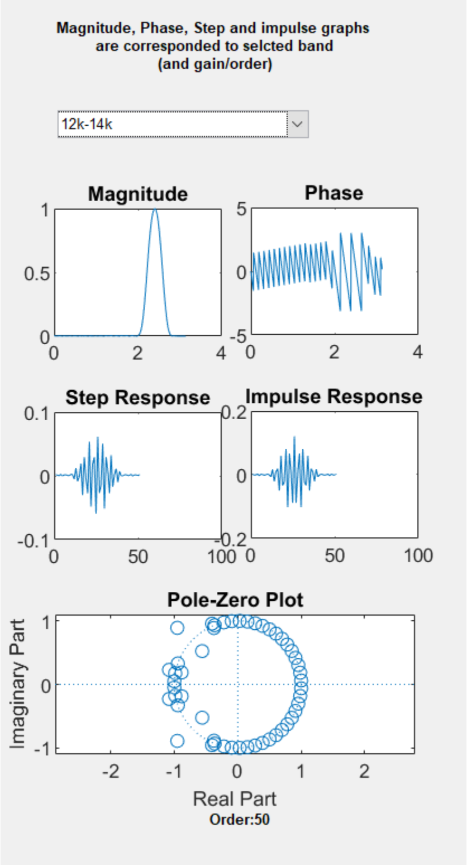


Magnitude, Phase, Step and impulse graphs
are corresponded to selcted band
(and gain/order)

6k-12k

Order:2

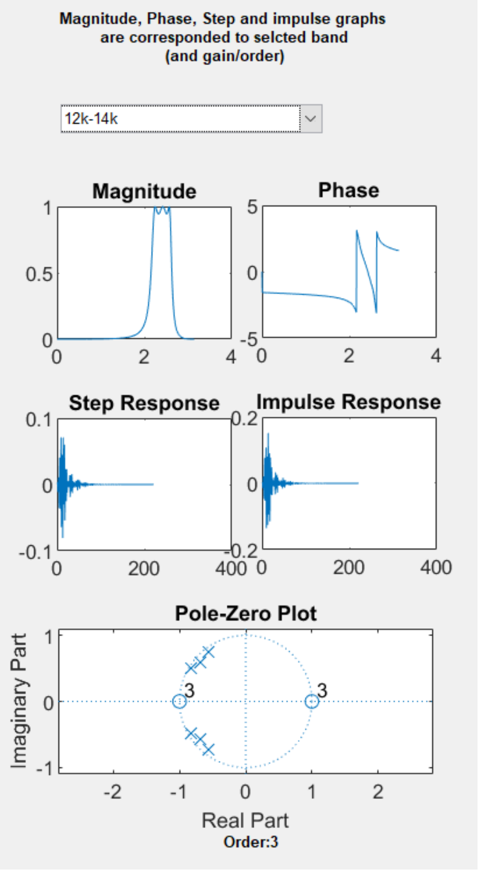
12 - 14 KHz



Magnitude, Phase, Step and impulse graphs
are corresponded to selcted band
(and gain/order)

12k-14k

Order:50

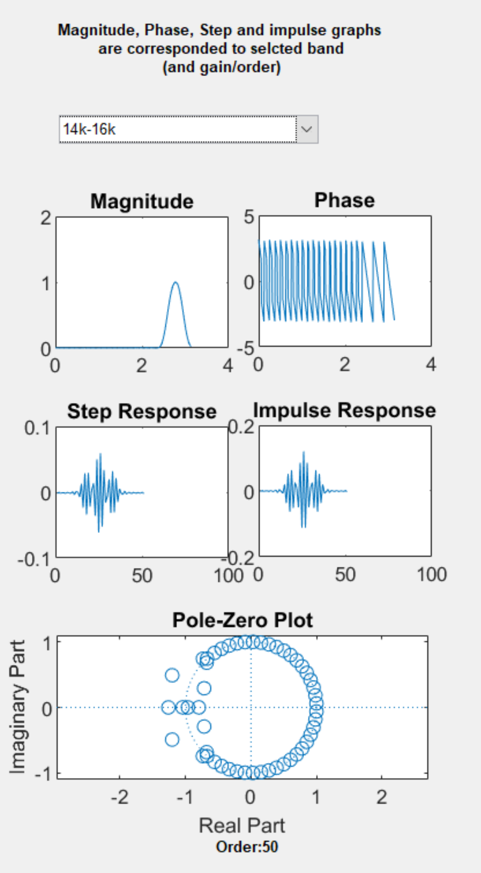


Magnitude, Phase, Step and impulse graphs
are corresponded to selcted band
(and gain/order)

12k-14k

Order:3

<table>
<tr><th colspan="2">14 - 16 KHz</th></tr>
<tr>
<td>

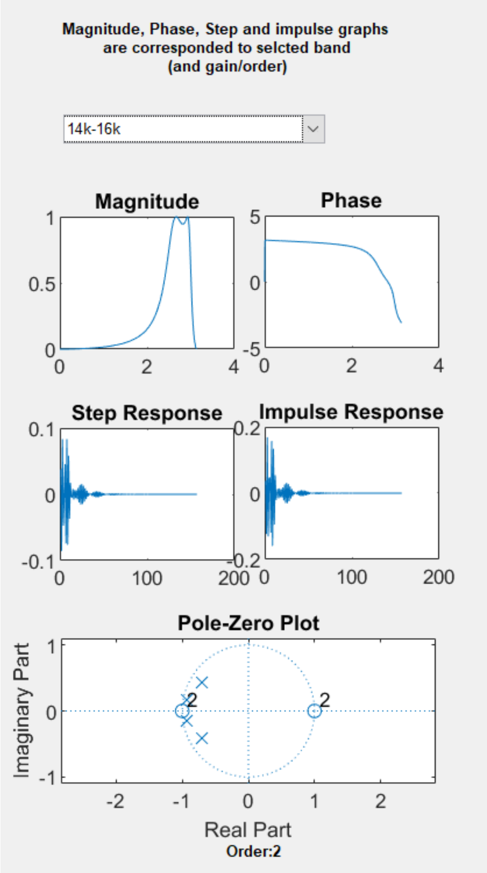


Magnitude, Phase, Step and impulse graphs are corresponded to selcted band (and gain/order)

14k-16k

Order:50

</td>
<td>



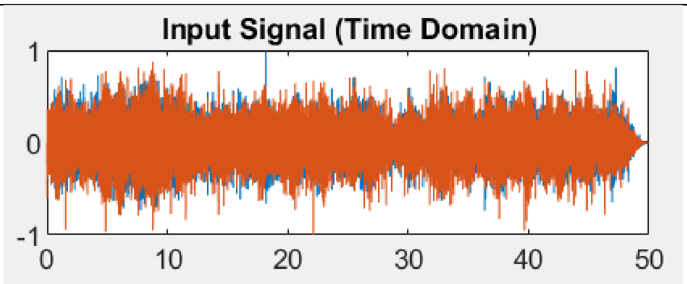
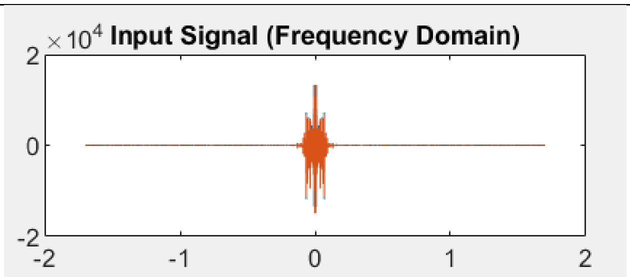
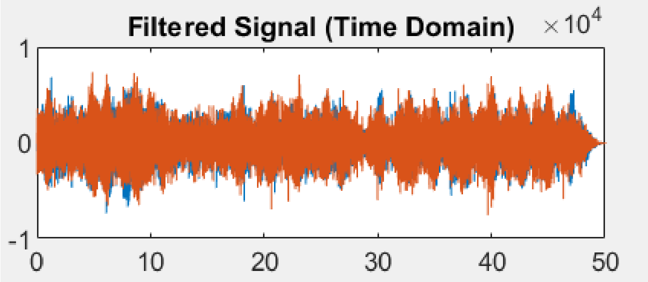
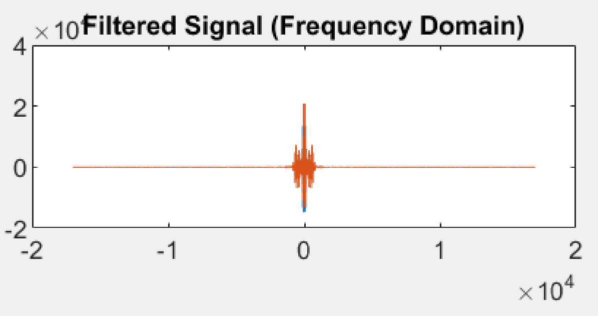
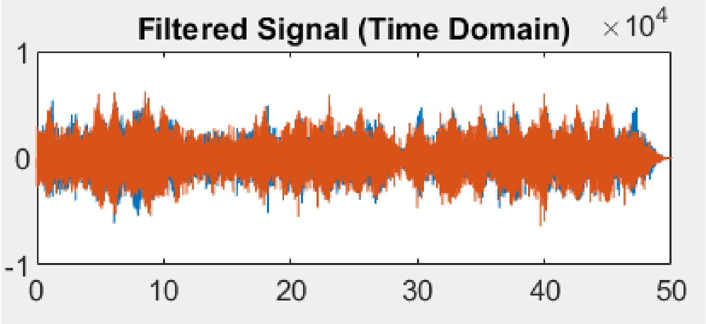
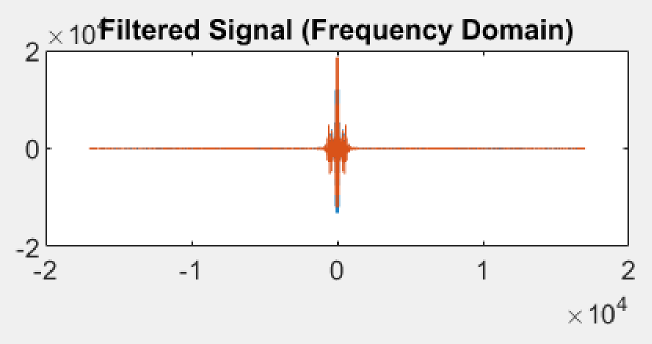
Magnitude, Phase, Step and impulse graphs are corresponded to selcted band (and gain/order)

14k-16k

Order:2

</td>
</tr>
</table>

➢ Figures of signals in Time and Frequency Domains

Time Domain	Frequency Domain
Input Signal	
	
Output Signal (No gain)	
	
Output Signal (With some gain)	
 Filtered Signal (Time Domain) ×10⁴ 1 0 -1 0 10 20 30 40 50	 ×10 Filtered Signal (Frequency Domain) 2 0 -2 -2 -1 0 1 2 ×10⁴