DSP Final Project

Made by:

1) Ahmad Said Nouh (7086)

2) Hussein Mourad AbdelHamid (6729)

3) Abdelrahman Elsayed (3953)

Contents

[**1) Code:** 2](#_Toc104778614)

[**1-a) gui.m** 2](#_Toc104778615)

[**1-b) main.m** 5](#_Toc104778616)

[**1-c) custom\_filter.m** 7](#_Toc104778617)

[**1-d) custom\_plot.m** 8](#_Toc104778618)

[**2) Sample runs:** 10](#_Toc104778619)

[**2-a) FIR with file default sample rate:** 10](#_Toc104778620)

[**2-a-i) Analysis of the nine filters:** 10](#_Toc104778621)

[**2-a-ii) All figures in time and frequency domain:** 19](#_Toc104778622)

[**2-b) IIR with file default sample rate:** 24](#_Toc104778623)

[**2-b-i) Analysis of the nine filters:** 24](#_Toc104778624)

[**2-b-ii) All figures in time and frequency domain:** 33](#_Toc104778625)

[**2-c) IIR with file double sample rate:** 38](#_Toc104778626)

[**2-c-i) Final figure in time and frequency domain:** 38](#_Toc104778627)

[**2-d) IIR with file half sample rate:** 39](#_Toc104778628)

[**2-d-i) Final figure in time and frequency domain:** 39](#_Toc104778629)

[**3) Error handling:** 40](#_Toc104778630)

# **1) Code:**

We divided our work into four files to make the work more readable, clean and organized: gui1.m, main.m, custom\_filter.m, and custom\_plot.m. The first file is gui1.m which is the file that we run to start the whole program. This file contains the gui code which we will not display here as it is irrelevant and too long. The only part of this file that we will display will be the callback function of our run button which validates all the gui inputs and displays the appropriate message if something is missing or calls the main function from the main.m file if all inputs are correct. The second file is main.m which contains the main function of our code. In this file, we call the custom\_plot function from custom\_plot.m file to plot the figures in a specific way and call the custom\_filter function from custom\_filter.m file to create specific filters.

## **1-a) gui.m**

% --- Executes on button press in runBtn.

function runBtn\_Callback(hObject, eventdata, handles)

isValidInputs = validateInputs(handles);

if isValidInputs

chkValue = get(handles.fsCheckBox,'Value');

fs = 0;

default = 1;

if chkValue == 1

fs = str2double(get(handles.fs, 'String'));

default = 0;

end

g1 = str2double(get(handles.g1,'String'));

g2 = str2double(get(handles.g2,'String'));

g3 = str2double(get(handles.g3,'String'));

g4 = str2double(get(handles.g4,'String'));

g5 = str2double(get(handles.g5,'String'));

g6 = str2double(get(handles.g6,'String'));

g7 = str2double(get(handles.g7,'String'));

g8 = str2double(get(handles.g8,'String'));

g9 = str2double(get(handles.g9,'String'));

file\_name = get(handles.fileName, 'String');

filter\_type = get(handles.filterType, 'Value');

gains = [g1, g2, g3, g4, g5, g6, g7, g8, g9];

disp('valid')

main(file\_name, default, fs, filter\_type, gains);

end

function isValid = validateInputs(handles)

fileName = get(handles.fileName, 'String');

if isempty(fileName)

errordlg('Must enter file name!!');

isValid = 0;

return

end

selectedIndex = get(handles.filterType, 'Value');

if selectedIndex == 1

errordlg('Must choose filter type!!');

isValid = 0;

return

end

chkValue = get(handles.fsCheckBox,'Value');

if chkValue == 1

fs = get(handles.fs, 'String');

if isempty(fs)

errordlg('Must enter output sampling rate!!');

isValid = 0;

return

elseif isnan(str2double(fs))

errordlg('Output sample rate must be a real number more than 32000 Hz!!');

isValid = 0;

return

elseif str2double(fs) <= 32000

errordlg('Output sample rate must be a real number more than 32000 Hz!!');

isValid = 0;

return

end

end

g1 = get(handles.g1,'String');

if isempty(g1)

errordlg('Must enter gain 1!!');

isValid = 0;

return

elseif isnan(str2double(g1))

errordlg('Gain 1 must be a real number!!');

isValid = 0;

return

end

%

g2 = get(handles.g2,'String');

if isempty(g2)

errordlg('Must enter gain 2!!');

isValid = 0;

return

elseif isnan(str2double(g2))

errordlg('Gain 2 must be a real number!!');

isValid = 0;

return

end

%

g3 = get(handles.g3,'String');

if isempty(g3)

errordlg('Must enter gain 3!!');

isValid = 0;

return

elseif isnan(str2double(g3))

errordlg('Gain 3 must be a real number!!');

isValid = 0;

return

end

%

g4 = get(handles.g4,'String');

if isempty(g4)

errordlg('Must enter gain 4!!');

isValid = 0;

return

elseif isnan(str2double(g4))

errordlg('Gain 4 must be a real number!!');

isValid = 0;

return

end

%

g5 = get(handles.g5,'String');

if isempty(g5)

errordlg('Must enter gain 5!!');

isValid = 0;

return

elseif isnan(str2double(g5))

errordlg('Gain 5 must be a real number!!');

isValid = 0;

return

end

%

g6 = get(handles.g6,'String');

if isempty(g6)

errordlg('Must enter gain 6!!');

isValid = 0;

return

elseif isnan(str2double(g6))

errordlg('Gain 6 must be a real number!!');

isValid = 0;

return

end

%

g7 = get(handles.g7,'String');

if isempty(g7)

errordlg('Must enter gain 7!!');

isValid = 0;

return

elseif isnan(str2double(g7))

errordlg('Gain 7 must be a real number!!');

isValid = 0;

return

end

%

g8 = get(handles.g8,'String');

if isempty(g8)

errordlg('Must enter gain 8!!');

isValid = 0;

return

elseif isnan(str2double(g8))

errordlg('Gain 8 must be a real number!!');

isValid = 0;

return

end

%

g9 = get(handles.g9,'String');

if isempty(g9)

errordlg('Must enter gain 9!!');

isValid = 0;

return

elseif isnan(str2double(g9))

errordlg('Gain 9 must be a real number!!');

isValid = 0;

return

end

isValid = 1;

## **1-b) main.m**

function main(file\_name,default\_fs, fs\_user, filter\_type, gains)

try

[x,fs\_default] = audioread(file\_name);

if default\_fs == 1

fs = fs\_default;

else

fs = fs\_user;

end

catch ex

errordlg('No file with such name exists!!');

return

end

Ns = length(x);

t = 0:1/fs:(Ns-1)/fs;

T = linspace(-fs/2,fs/2,Ns);

wn = 2\*170/fs;

[x1\_filtered, x1\_amp\_filtered] = custom\_filter(x, gains(1), wn, fs, Ns, 'low', 'First', 1, filter\_type);

wn = [340/fs, 620/fs];

[x2\_filtered, x2\_amp\_filtered] = custom\_filter(x, gains(2), wn, fs, Ns, 'bandpass', 'Second', 4, filter\_type);

wn = [620/fs, 1200/fs];

[x3\_filtered, x3\_amp\_filtered] = custom\_filter(x, gains(3), wn, fs, Ns, 'bandpass', 'Third', 7, filter\_type);

wn = [1200/fs, 2000/fs];

[x4\_filtered, x4\_amp\_filtered] = custom\_filter(x, gains(4), wn, fs, Ns, 'bandpass', 'Fourth', 10, filter\_type);

wn = [2000/fs, 6000/fs];

[x5\_filtered, x5\_amp\_filtered] = custom\_filter(x, gains(5), wn, fs, Ns, 'bandpass', 'Fifth', 13, filter\_type);

wn = [6000/fs, 12000/fs];

[x6\_filtered, x6\_amp\_filtered] = custom\_filter(x, gains(6), wn, fs, Ns, 'bandpass', 'Sixth', 16, filter\_type);

wn = [12000/fs, 24000/fs];

[x7\_filtered, x7\_amp\_filtered] = custom\_filter(x, gains(7), wn, fs, Ns, 'bandpass', 'Seventh', 19, filter\_type);

wn = [24000/fs, 28000/fs];

[x8\_filtered, x8\_amp\_filtered] = custom\_filter(x, gains(8), wn, fs, Ns, 'bandpass', 'Eighth', 22, filter\_type);

wn = [28000/fs, 32000/fs];

[x9\_filtered, x9\_amp\_filtered] = custom\_filter(x, gains(9), wn, fs, Ns, 'bandpass', 'Ninth', 25, filter\_type);

custom\_plot(x1\_filtered, x1\_amp\_filtered, t, T, fs, ' after first filter', ' after first filter')

custom\_plot(x2\_filtered, x2\_amp\_filtered, t, T, fs, ' after second filter', ' after second filter')

custom\_plot(x3\_filtered, x3\_amp\_filtered, t, T, fs, ' after third filter', ' after third filter')

custom\_plot(x4\_filtered, x4\_amp\_filtered, t, T, fs, ' after fourth filter', ' after fourth filter')

custom\_plot(x5\_filtered, x5\_amp\_filtered, t, T, fs, ' after fifth filter', ' after fifth filter')

custom\_plot(x6\_filtered, x6\_amp\_filtered, t, T, fs, ' after sixth filter', ' after sixth filter')

custom\_plot(x7\_filtered, x7\_amp\_filtered, t, T, fs, ' after seventh filter', ' after seventh filter')

custom\_plot(x8\_filtered, x8\_amp\_filtered, t, T, fs, ' after eighth filter', ' after eighth filter')

custom\_plot(x9\_filtered, x9\_amp\_filtered, t, T, fs, ' after ninth filter', ' after ninth filter')

xt\_amp\_filtered = x1\_amp\_filtered + x2\_amp\_filtered + x3\_amp\_filtered + x4\_amp\_filtered + x5\_amp\_filtered + x6\_amp\_filtered + x7\_amp\_filtered + x8\_amp\_filtered + x9\_amp\_filtered;

custom\_plot(x, xt\_amp\_filtered, t, T, fs, ' original', ' filtered')

audiowrite('new.wav',xt\_amp\_filtered,fs)

end

## **1-c) custom\_filter.m**

function [x\_filtered, x\_amp\_filtered] = custom\_filter(x,gain, wn, fs, Ns, type, message, i, filter\_type)

gain\_watt = 10.^(gain/20);

if filter\_type == 2

num = fir1(50,wn,type);

den = 1;

[num,den] = eqtflength(num, den);

[z,p,k] = tf2zp(num,den);

[num\_amp,den\_amp] = zp2tf(z,p,gain\_watt\*k);

disp('fir')

else

[num,den] = butter(3, wn, type);

[z,p,k] = butter(3, wn, type);

[num\_amp,den\_amp] = zp2tf(z,p,gain\_watt\*k);

disp('iir')

end

figure(i)

subplot(2,1,1)

freqz(num\_amp,den\_amp)

figure(i+1)

sys = tf(num\_amp,den\_amp);

subplot(2,1,1)

step(sys)

subplot(2,1,2)

impulse(sys)

figure(i+2)

zplane(z,p)

title(strcat(message,' filter zeros and poles'))

x\_filtered = filter(num,den,x);

x\_amp\_filtered = filter(num\_amp,den\_amp,x);

end

## **1-d) custom\_plot.m**

function custom\_plot(x, x\_amp, t, T, fs, message1, message2)

X\_mags = abs(fftshift(fft(x)))/fs;

X\_amp\_mags = abs(fftshift(fft(x\_amp)))/fs;

figure

subplot(2,2,1)

plot(t,x)

title(strcat(strcat('Signal',message1),' time domain no gain'));

xlabel('time(sec)');

ylabel('Amplitude')

subplot(2,2,3)

plot(t,x\_amp)

title(strcat(strcat('Signal',message2),' time domain with gain'));

xlabel('time(sec)');

ylabel('Amplitude')

subplot(2,2,2)

plot(T,X\_mags)

title(strcat(strcat('Signal',message1),' frequency domain no gain'))

xlabel('freq (Hz)')

ylabel('Magnitude')

subplot(2,2,4)

plot(T,X\_amp\_mags)

title(strcat(strcat('Signal',message2),' frequency domain with gain'))

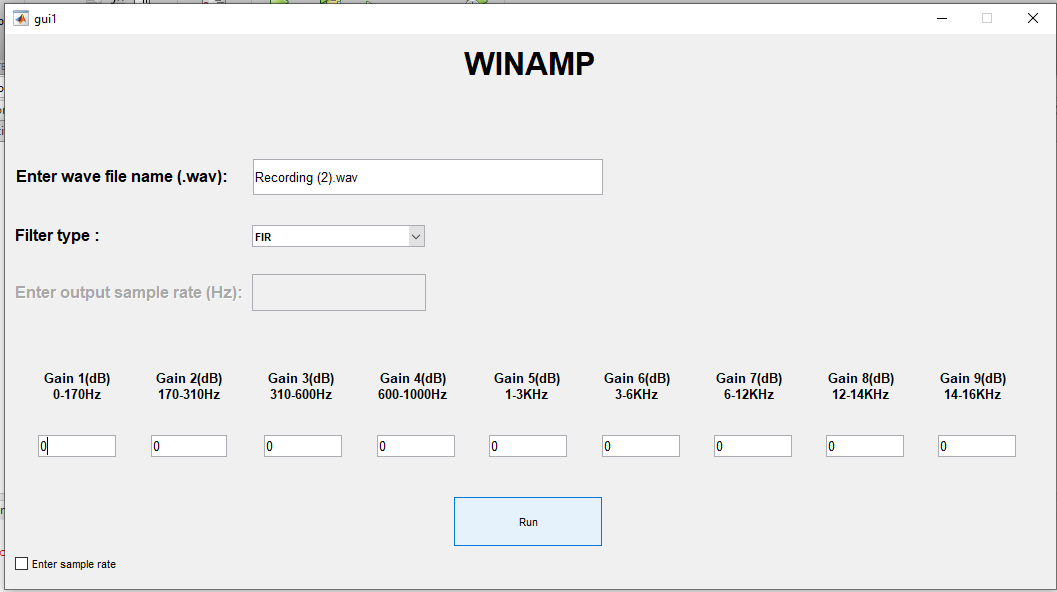
xlabel('freq (Hz)')

ylabel('Magnitude')

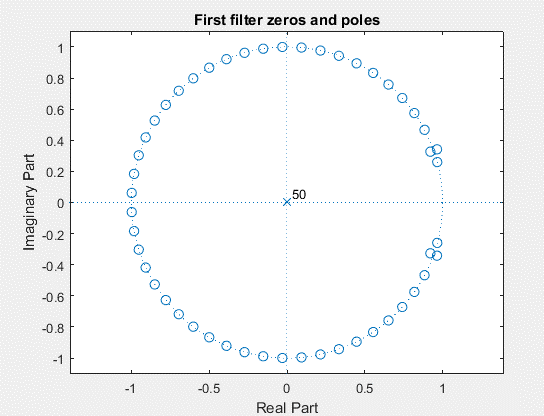
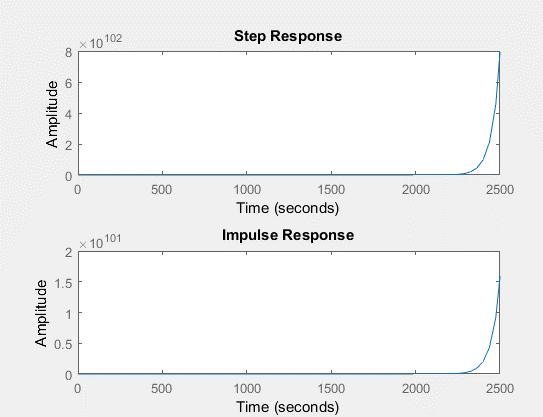
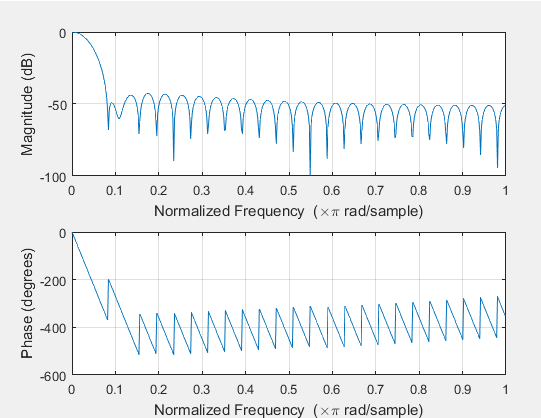
end

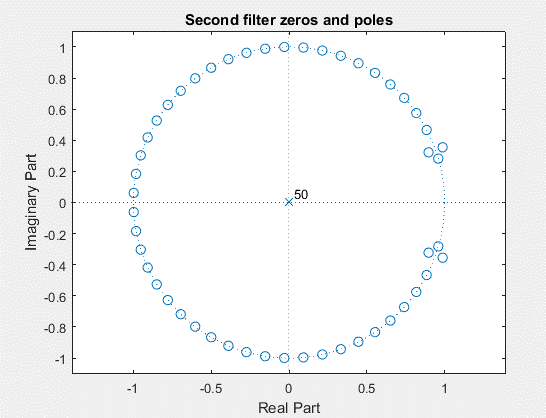
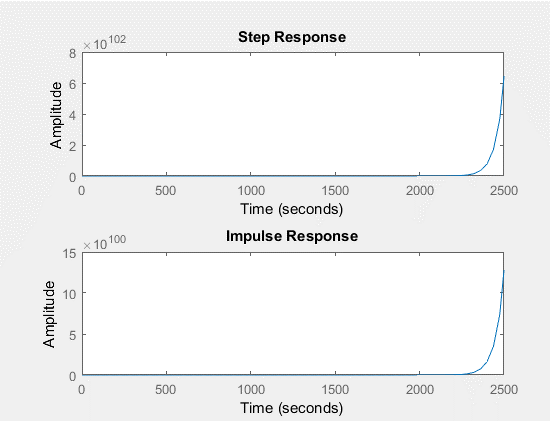
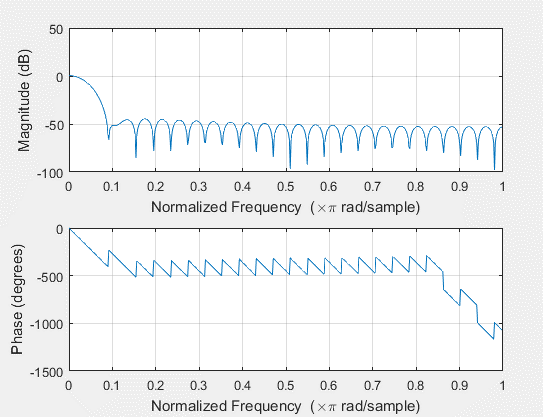
# **2) Sample runs:**

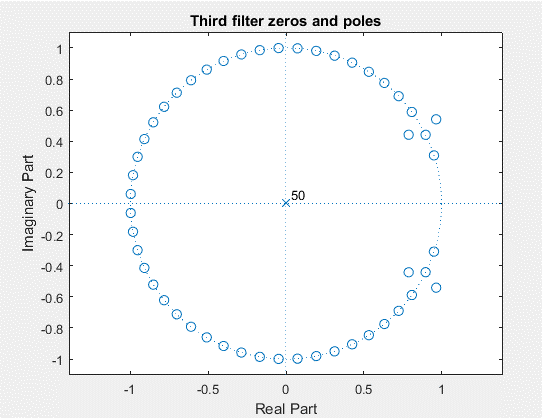
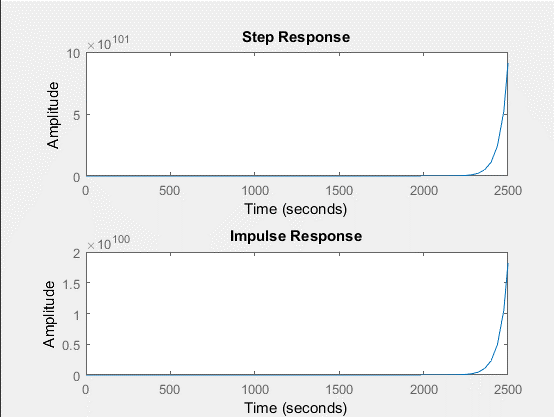
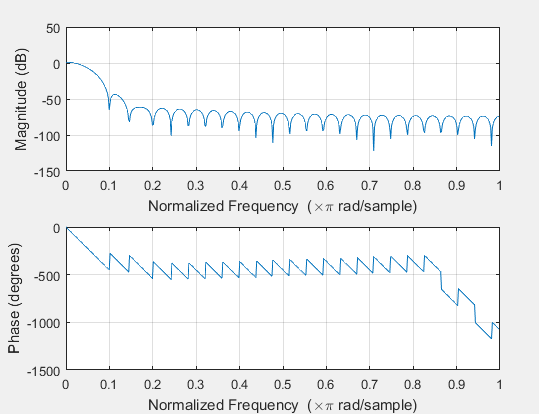
## **2-a) FIR with file default sample rate:**

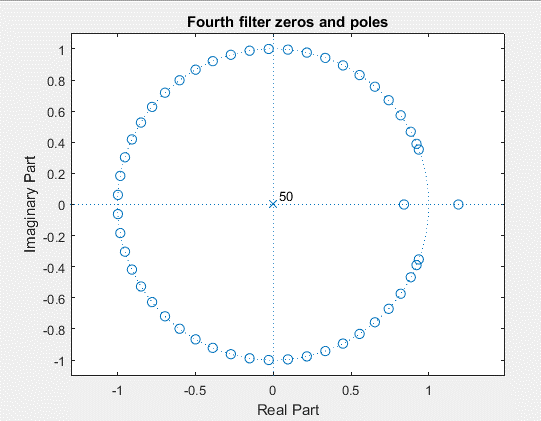
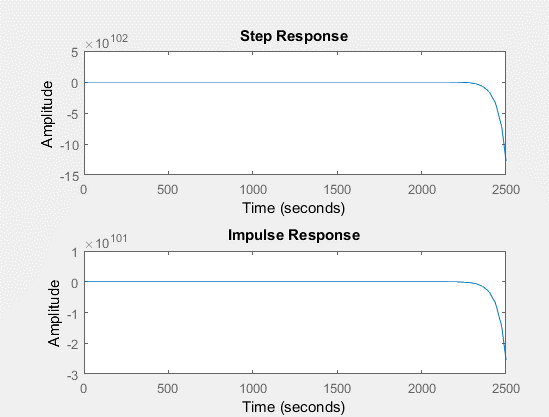
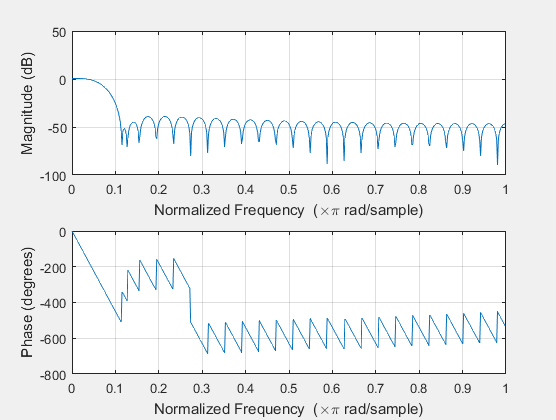


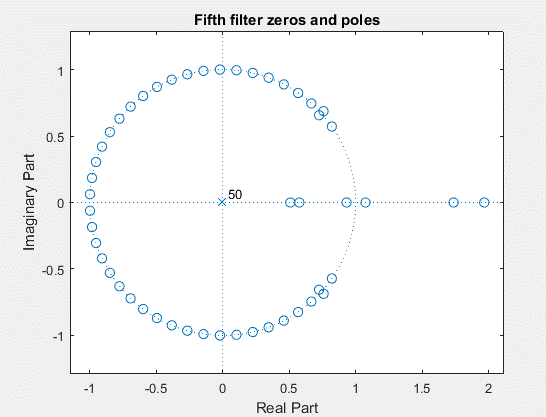
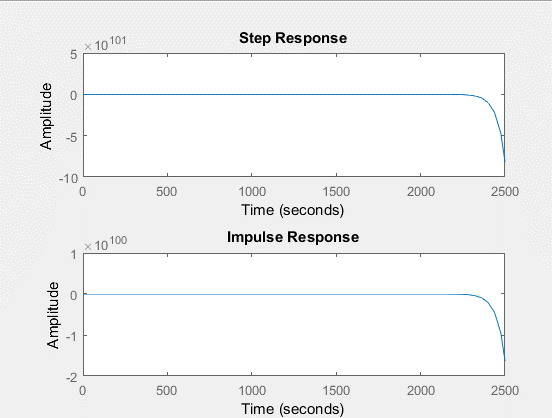
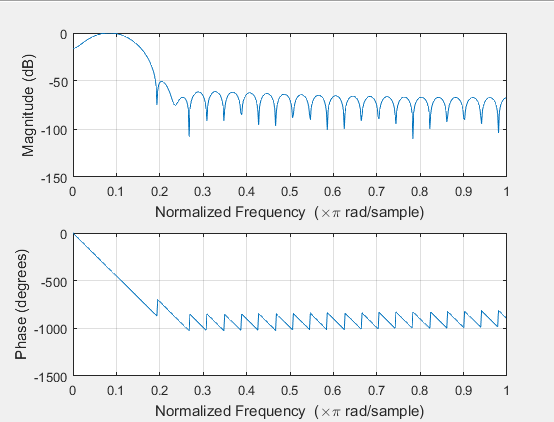
### **2-a-i) Analysis of the nine filters:**

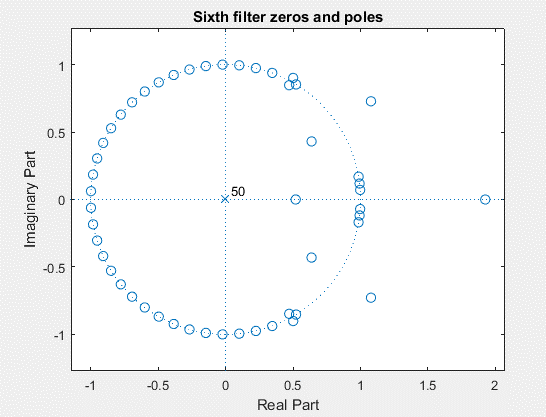
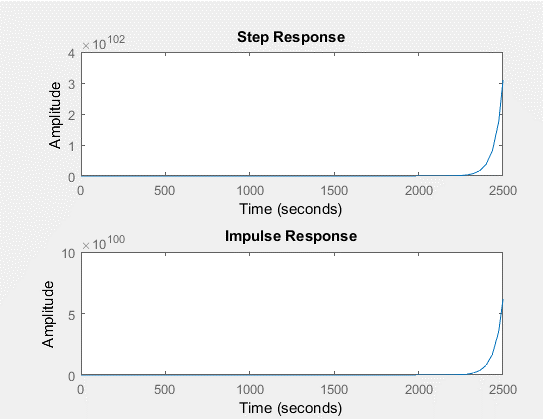
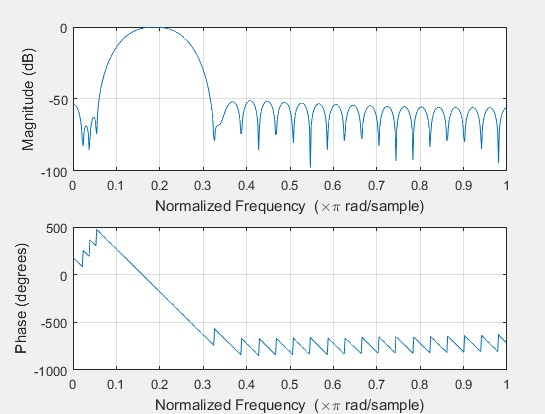


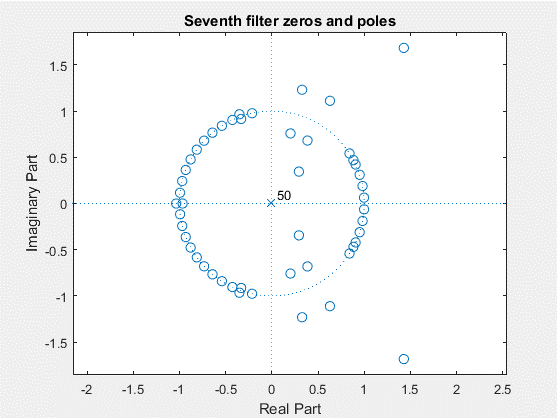
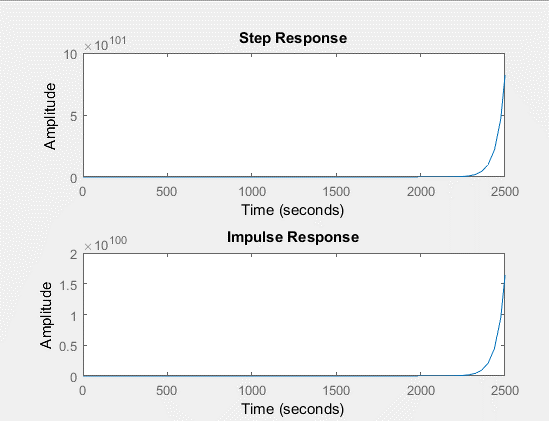
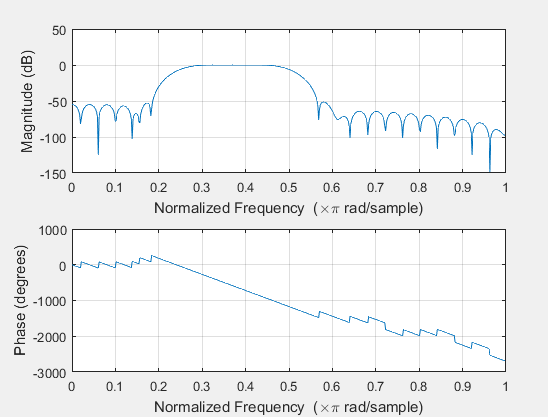


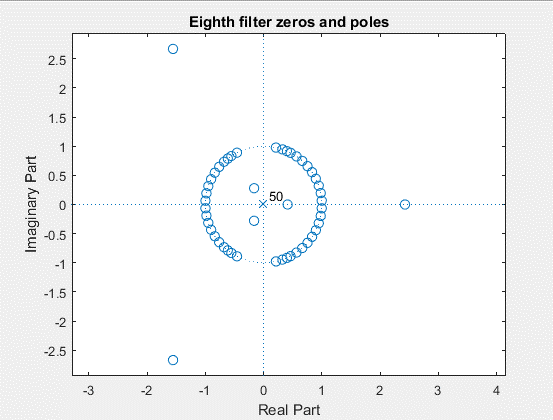
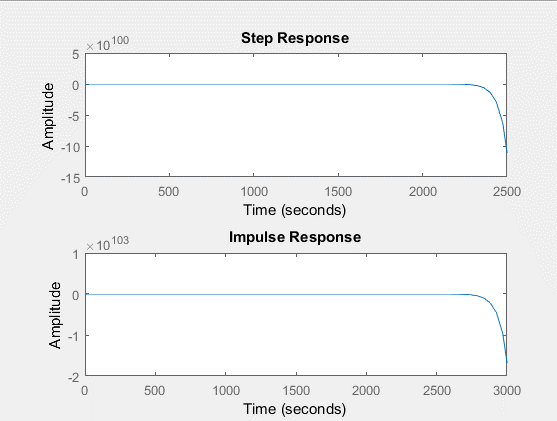
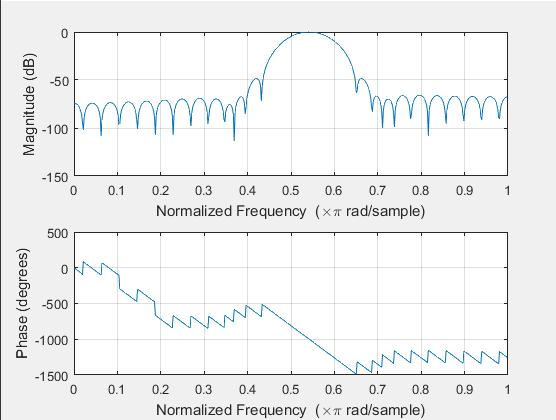


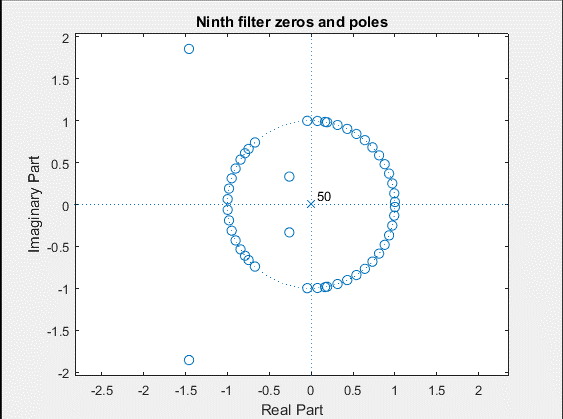
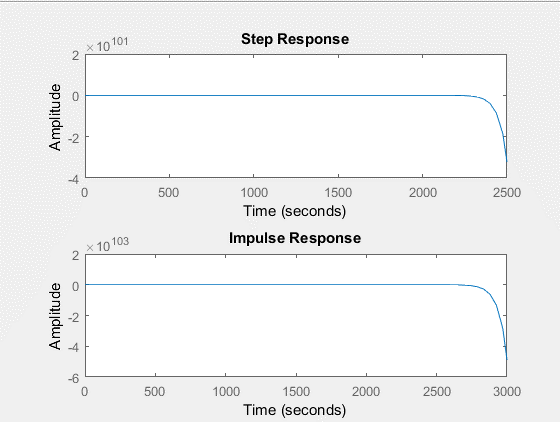
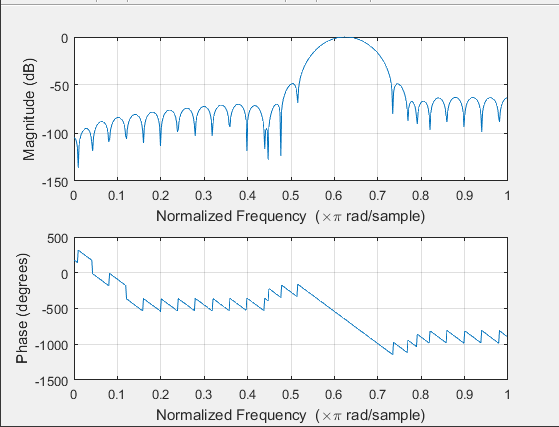




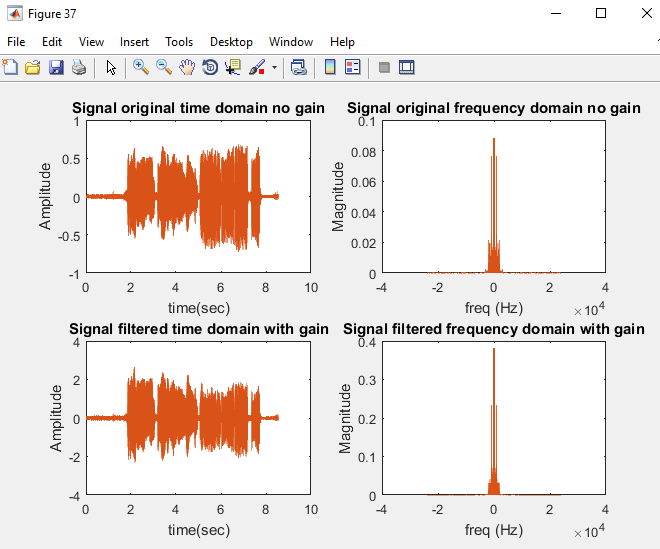
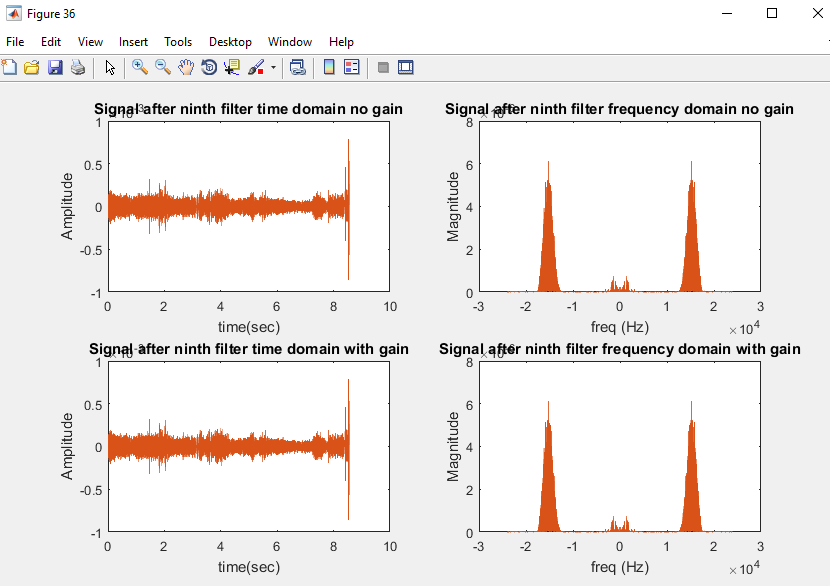
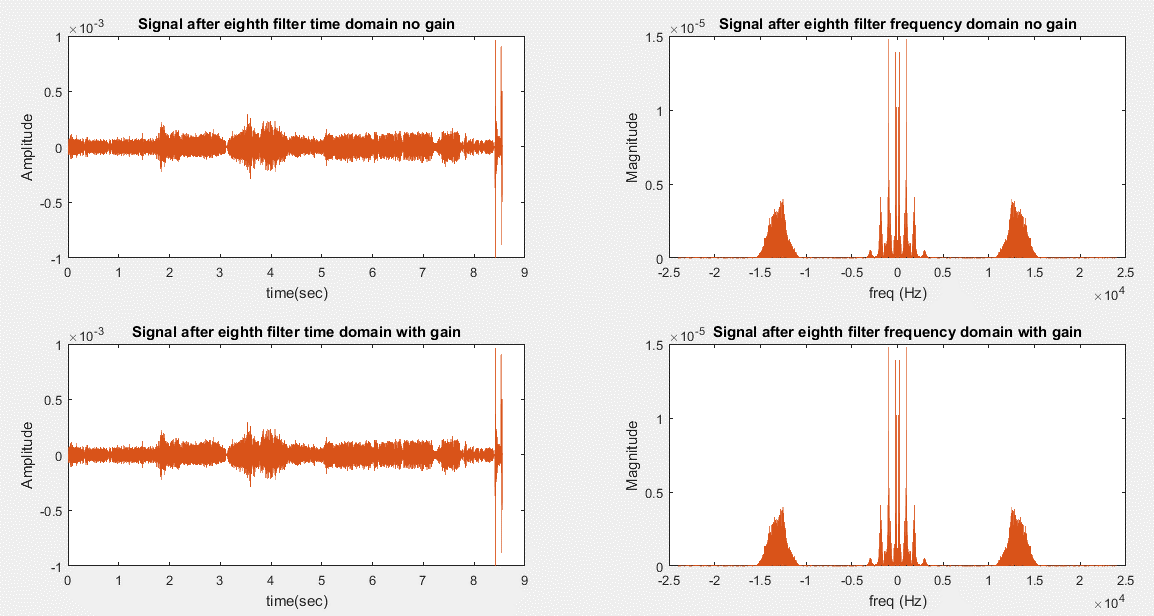
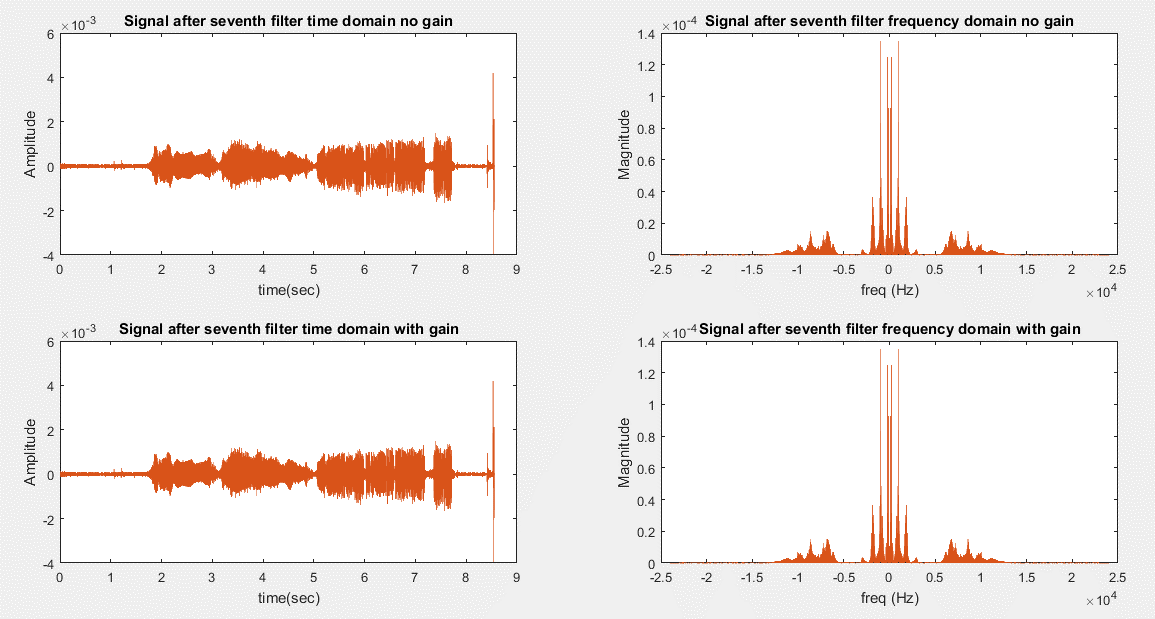
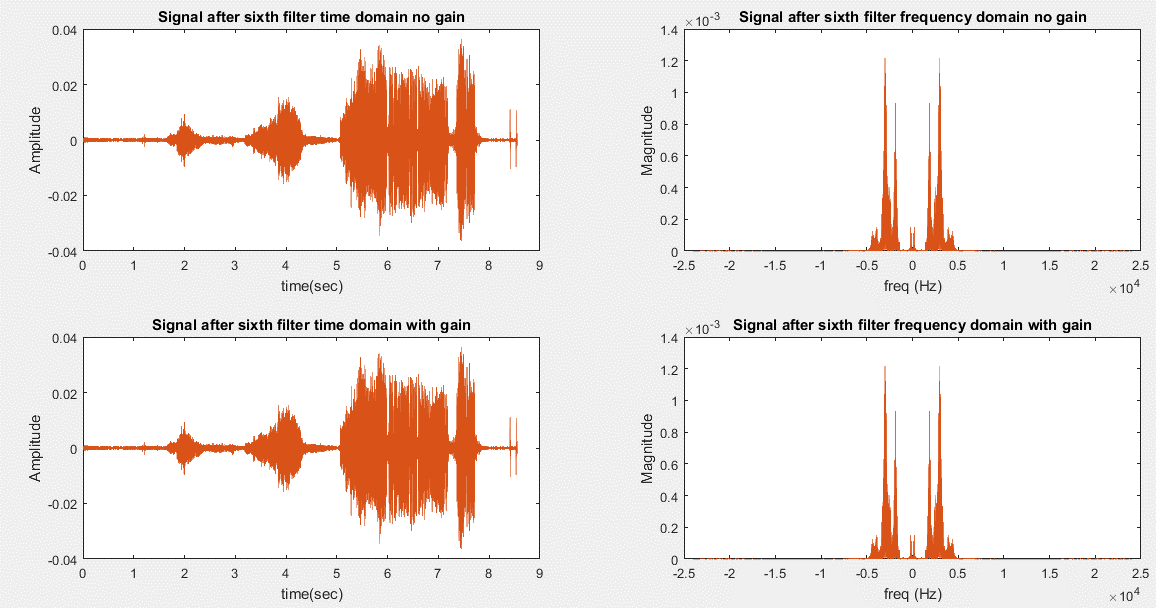
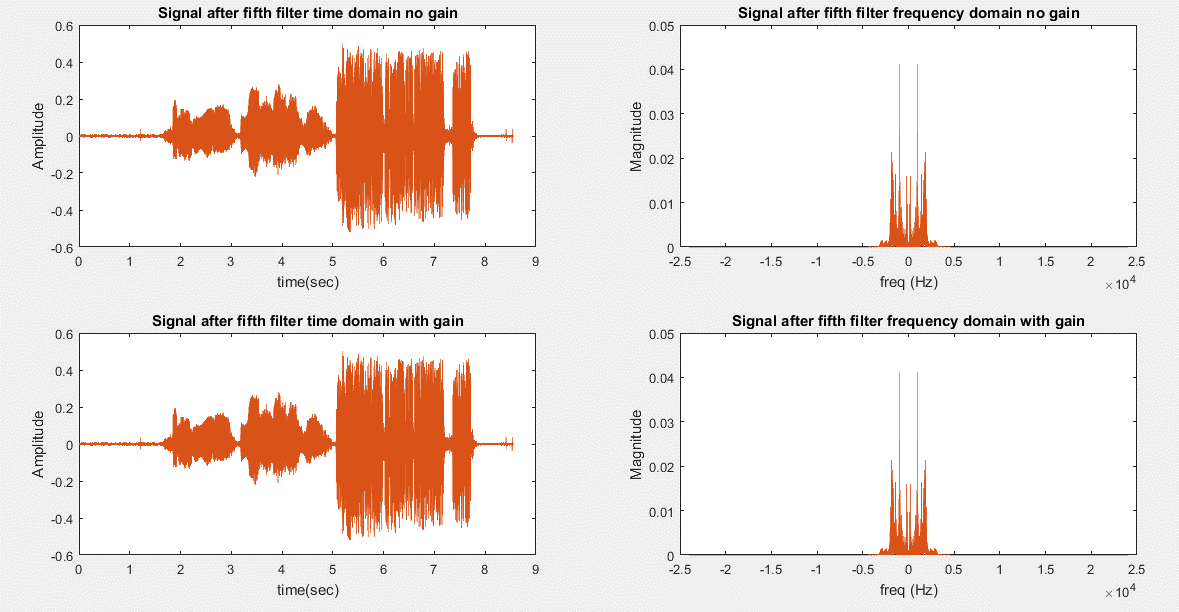
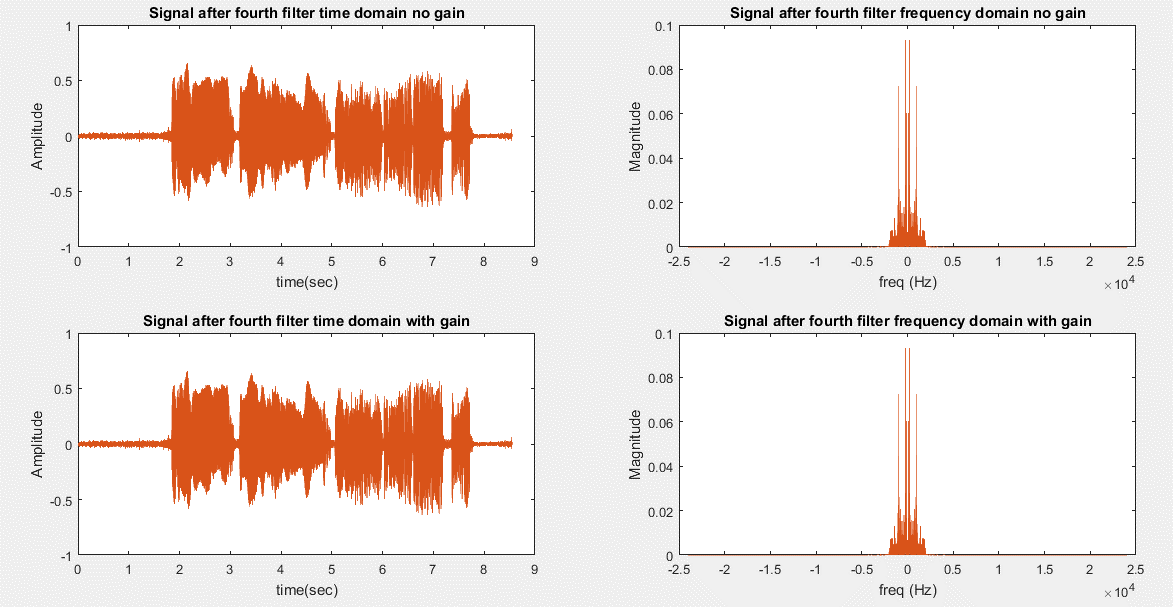
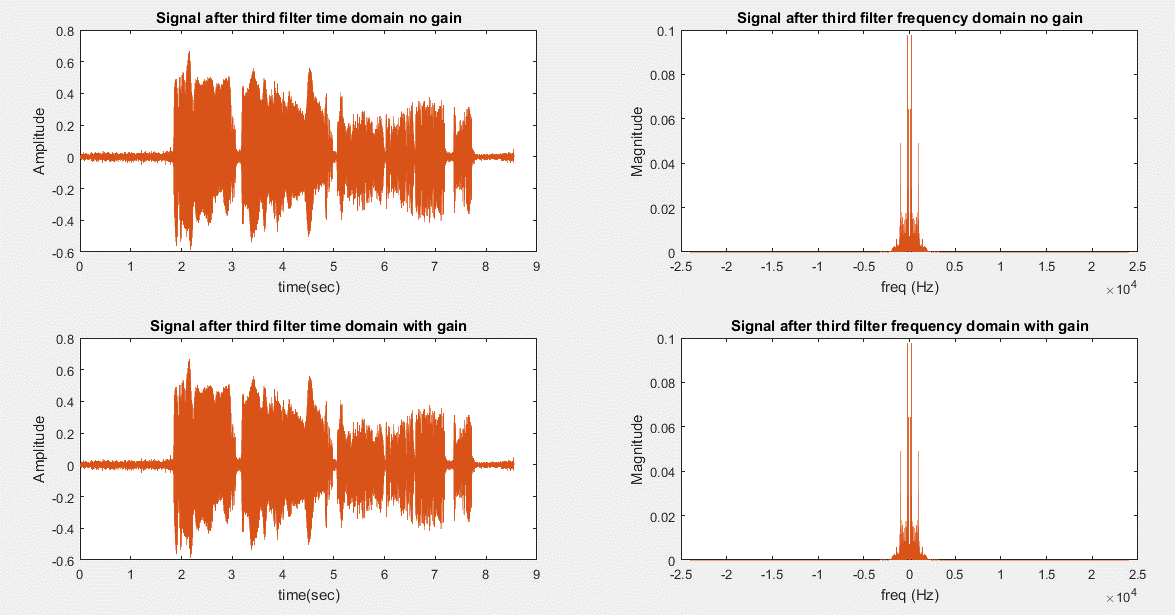
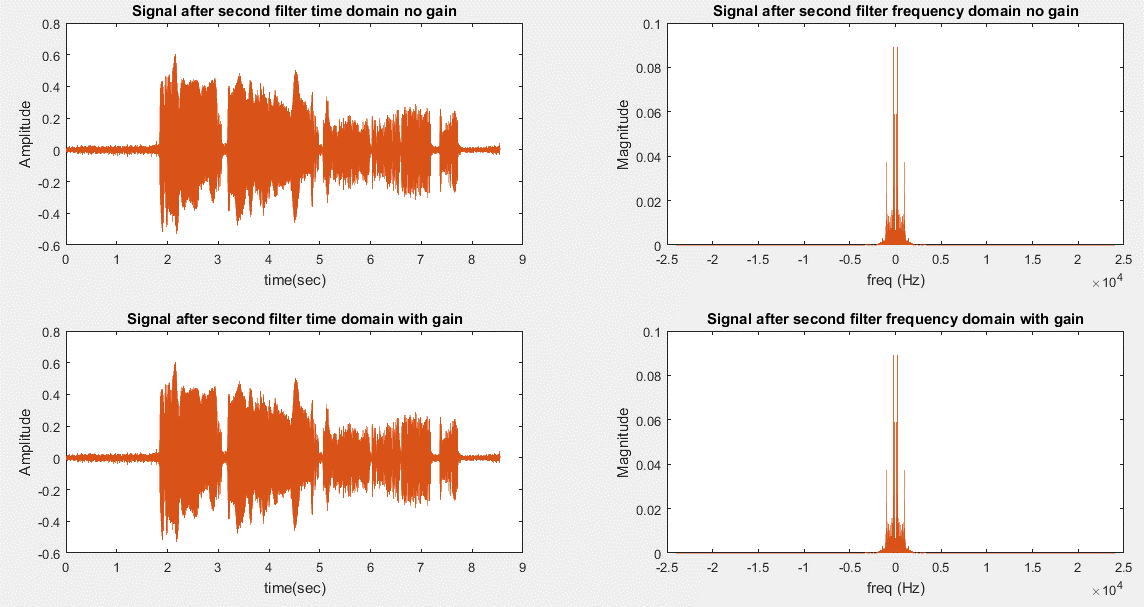
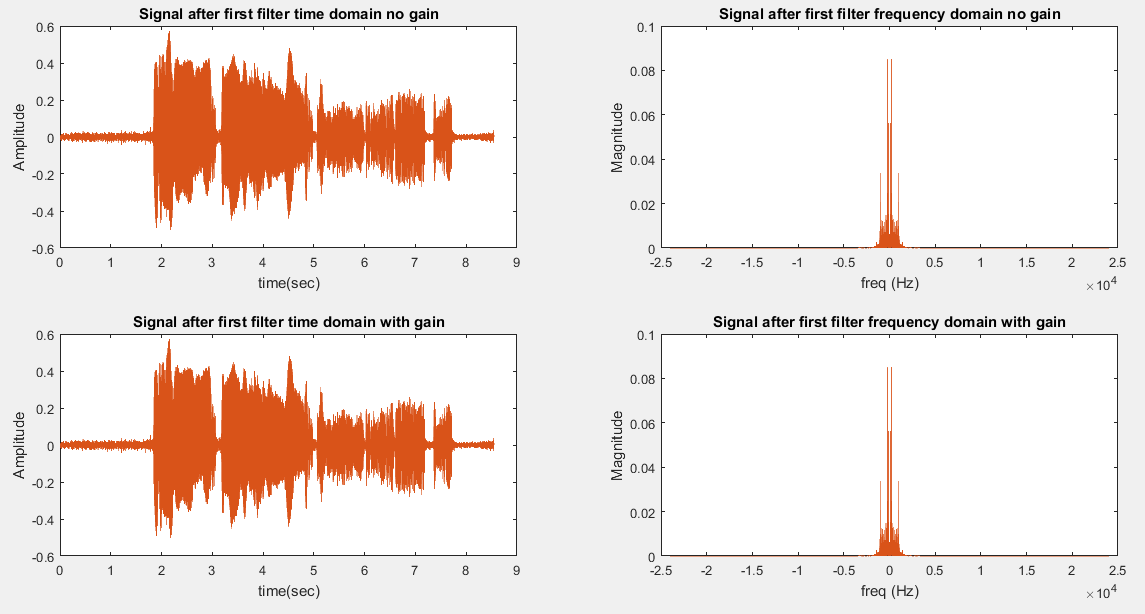




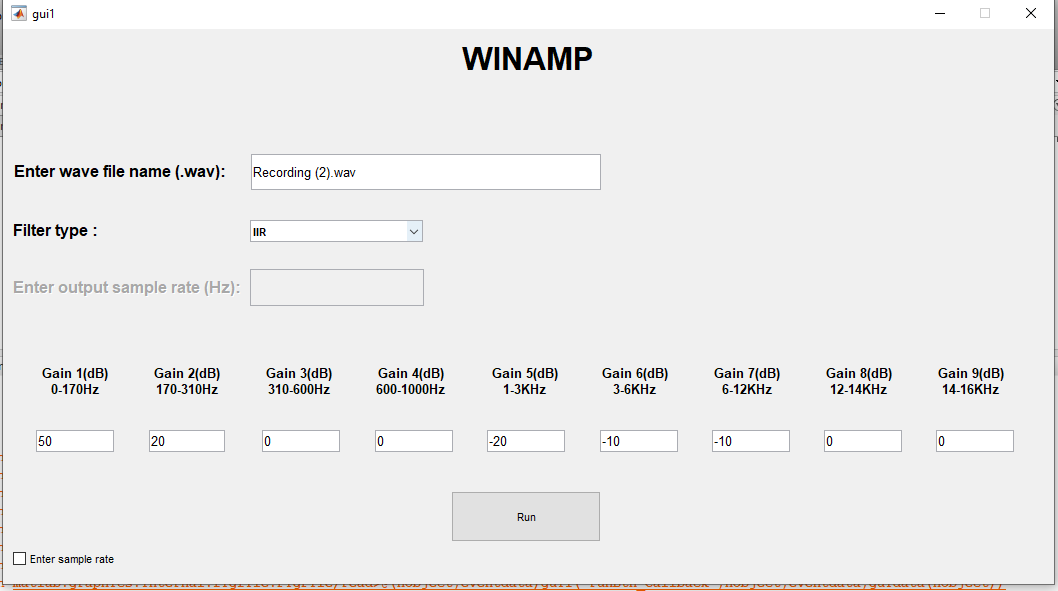




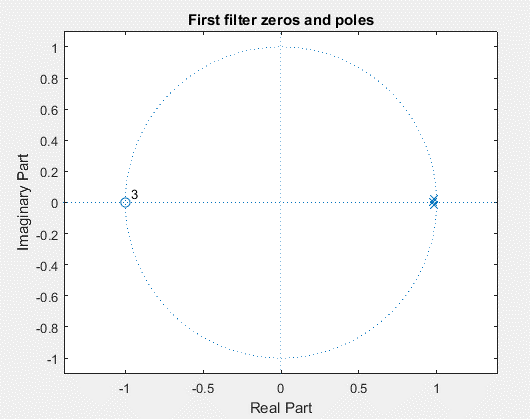
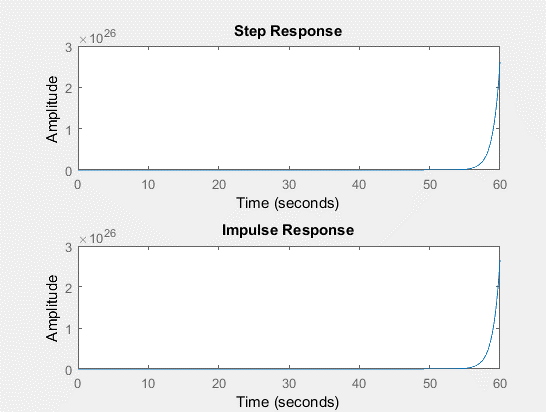
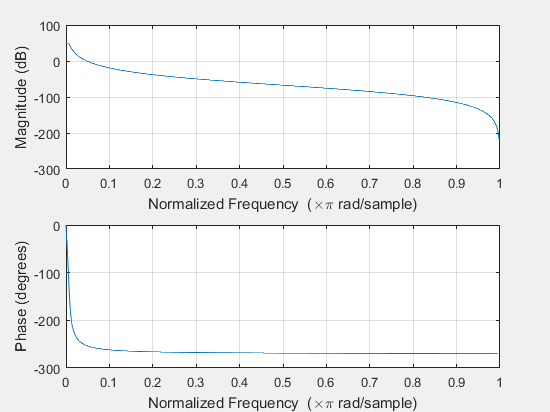
### **2-a-ii) All figures in time and frequency domain:**

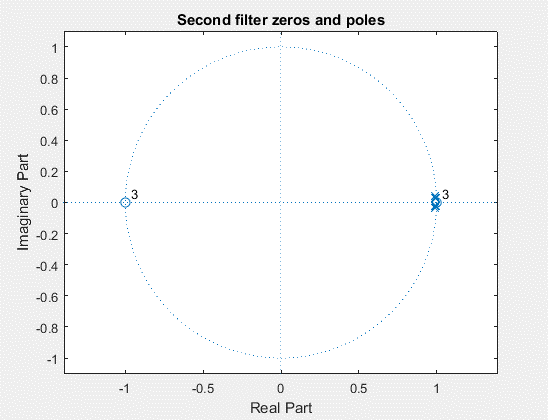
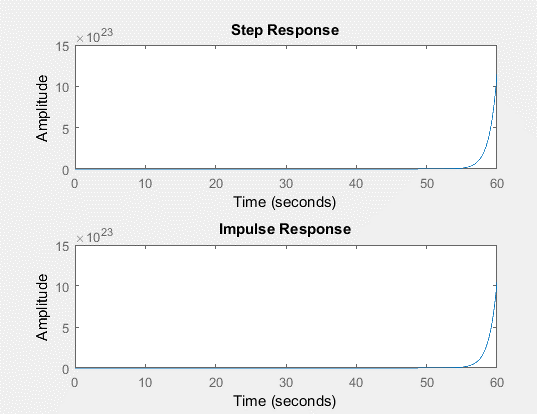
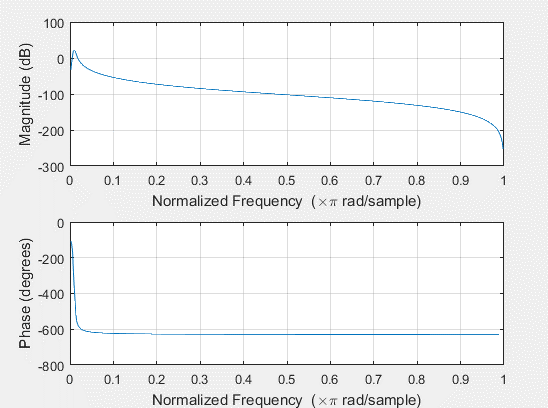


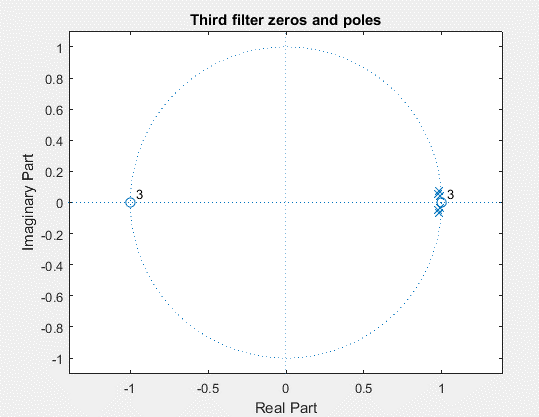
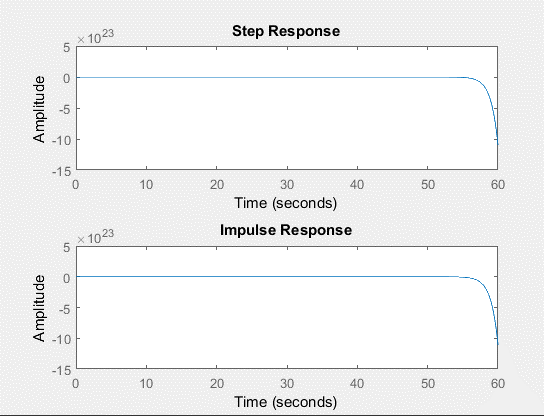
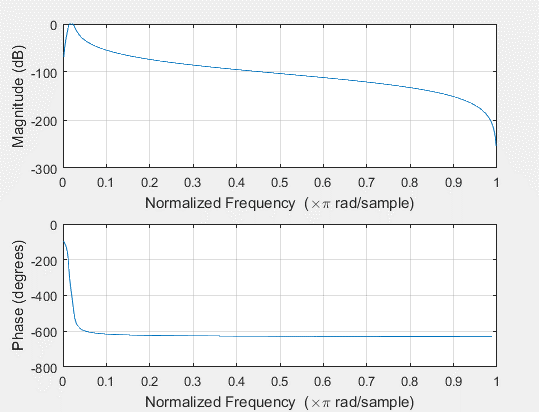
## **2-b) IIR with file default sample rate:**

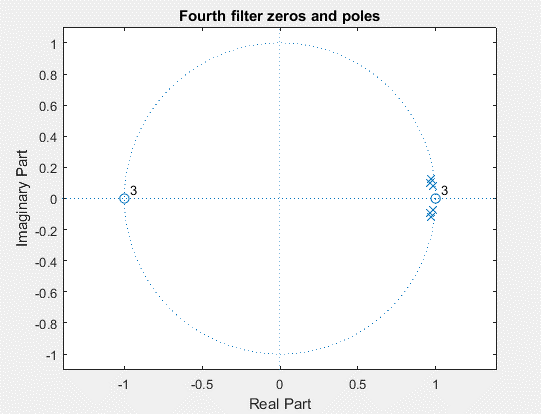
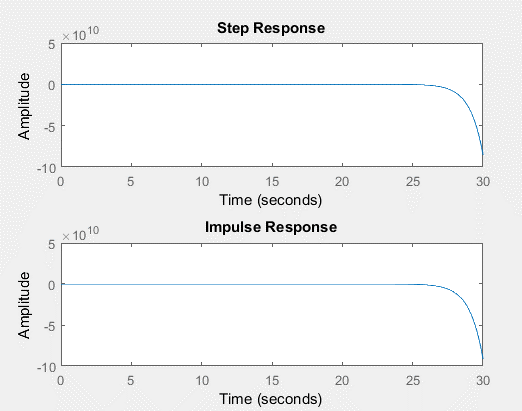
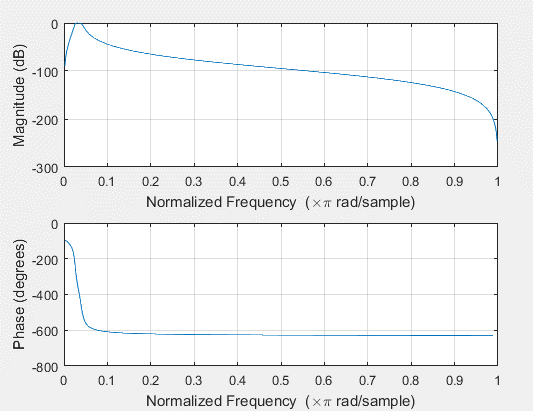


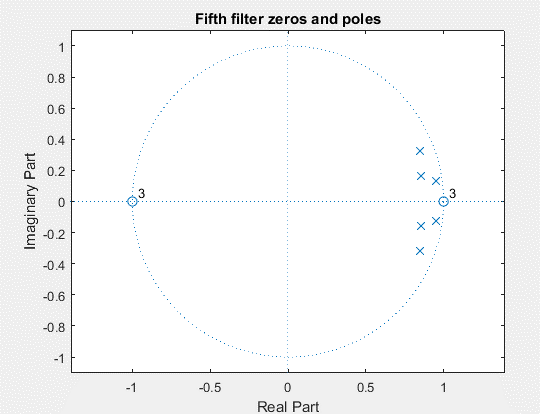
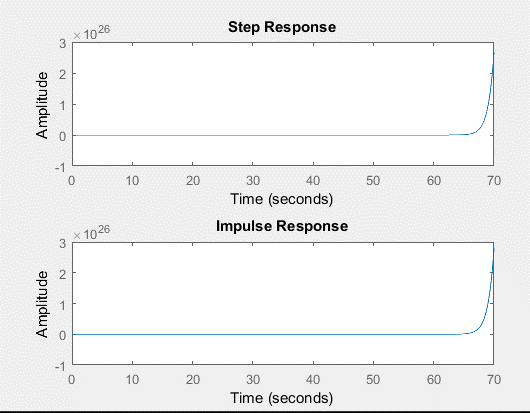
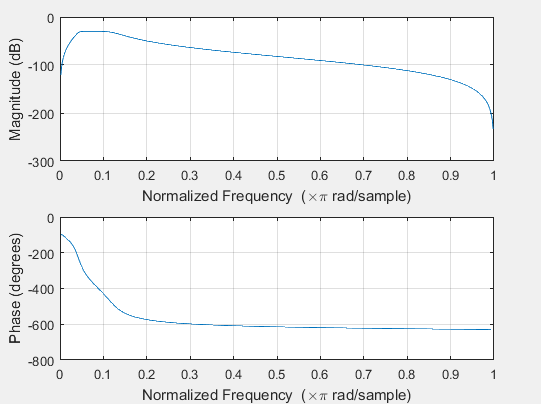
### **2-b-i) Analysis of the nine filters:**

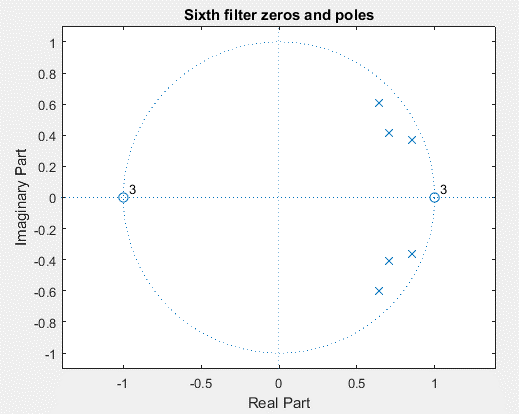
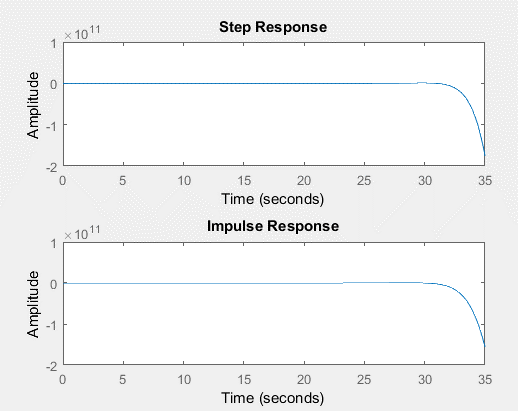
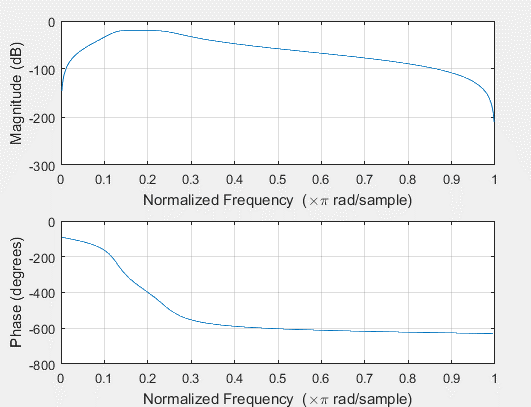


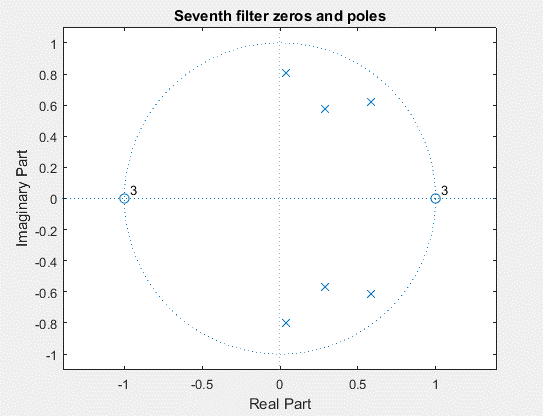
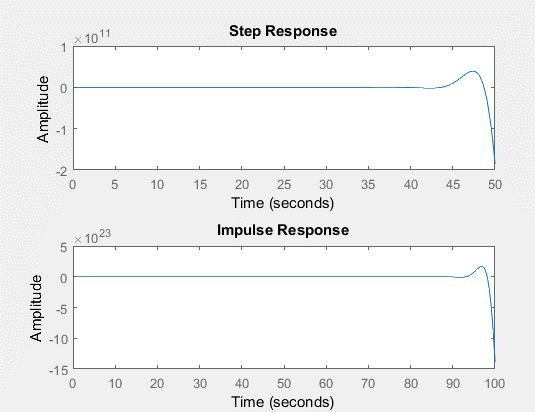
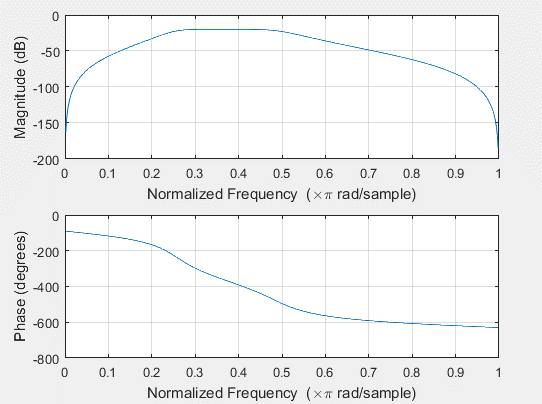


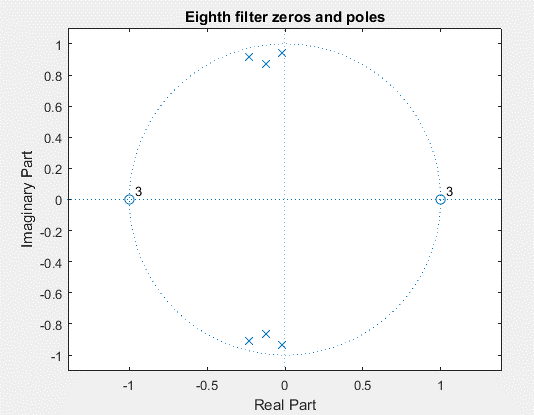
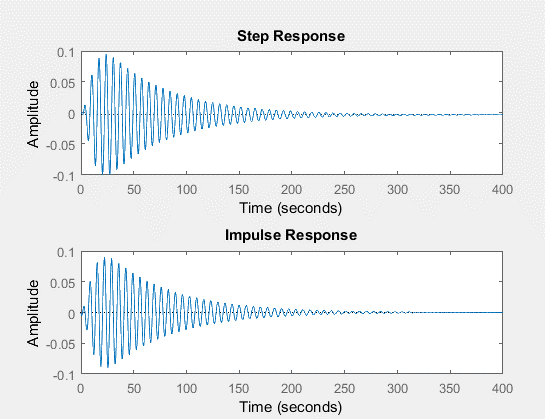
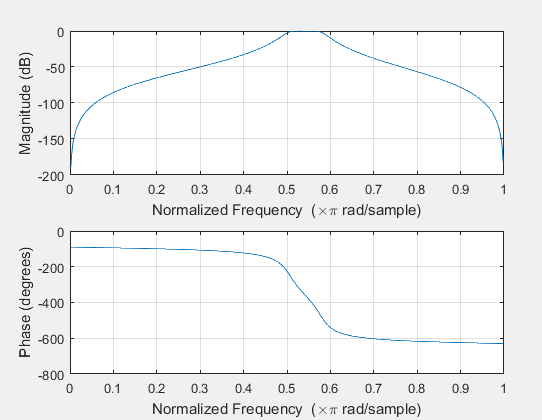


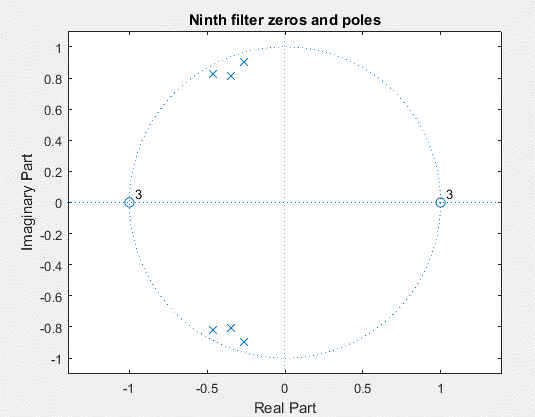
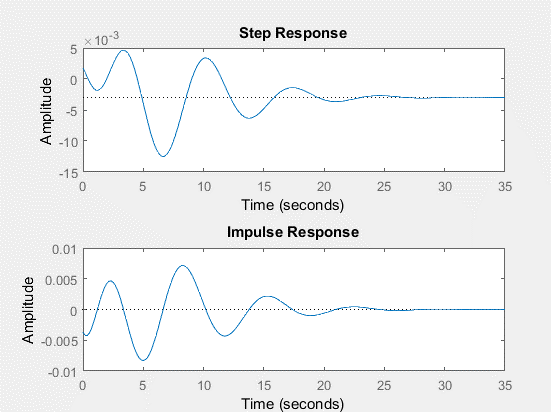
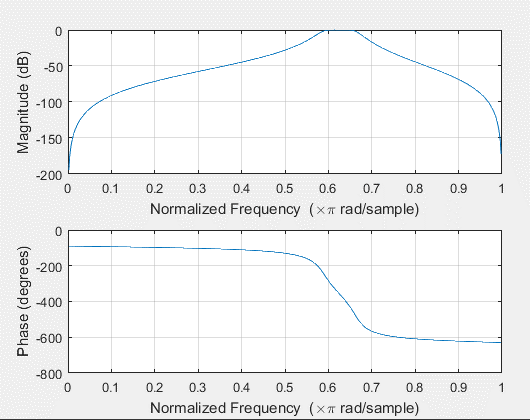




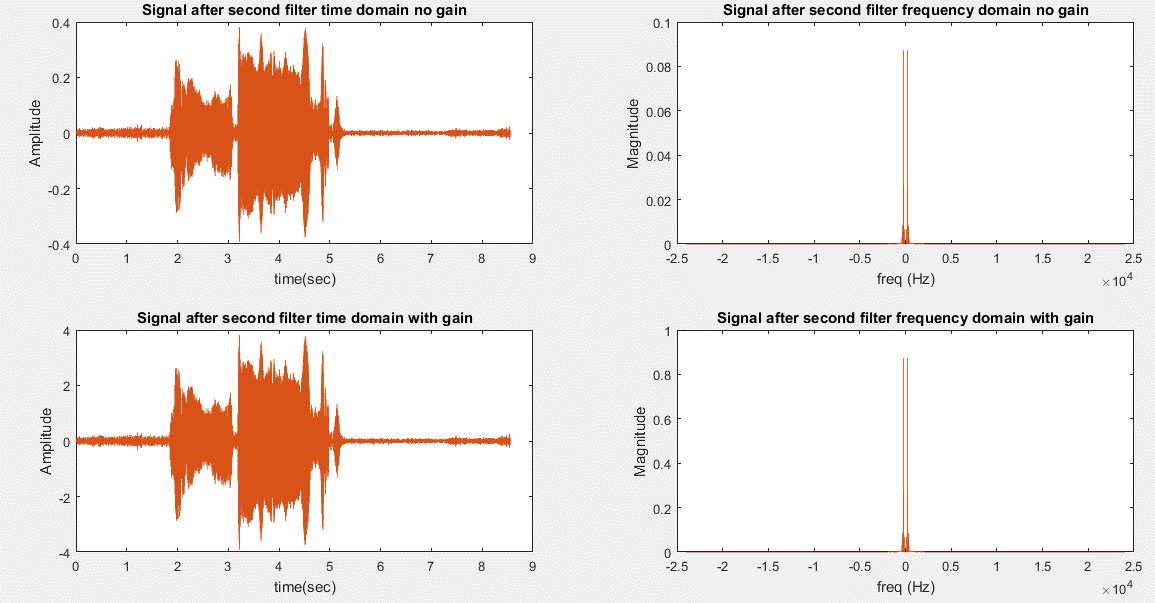
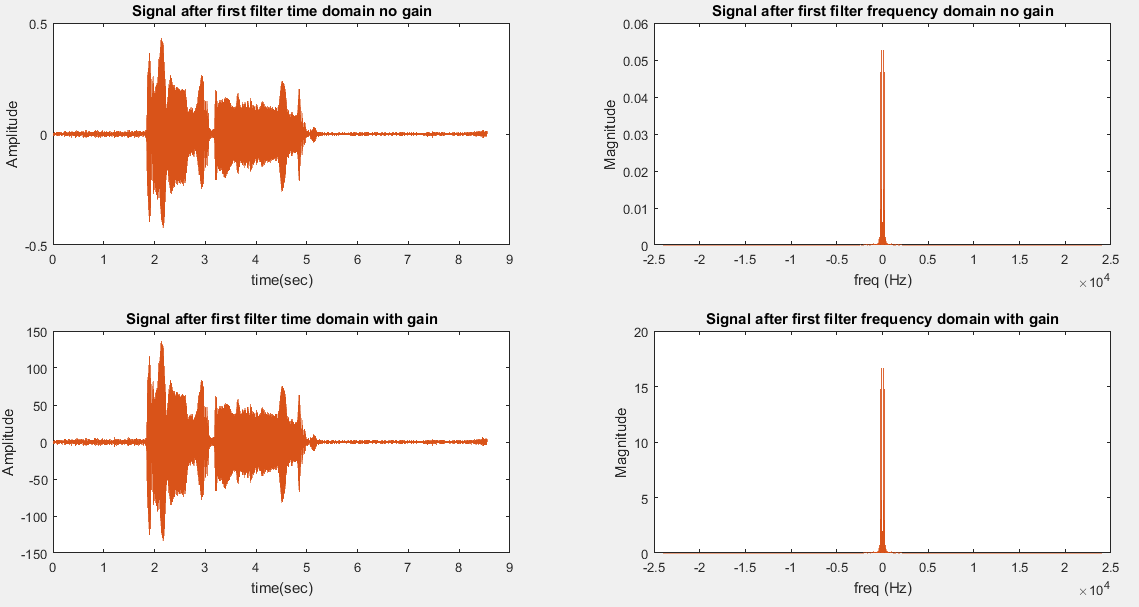


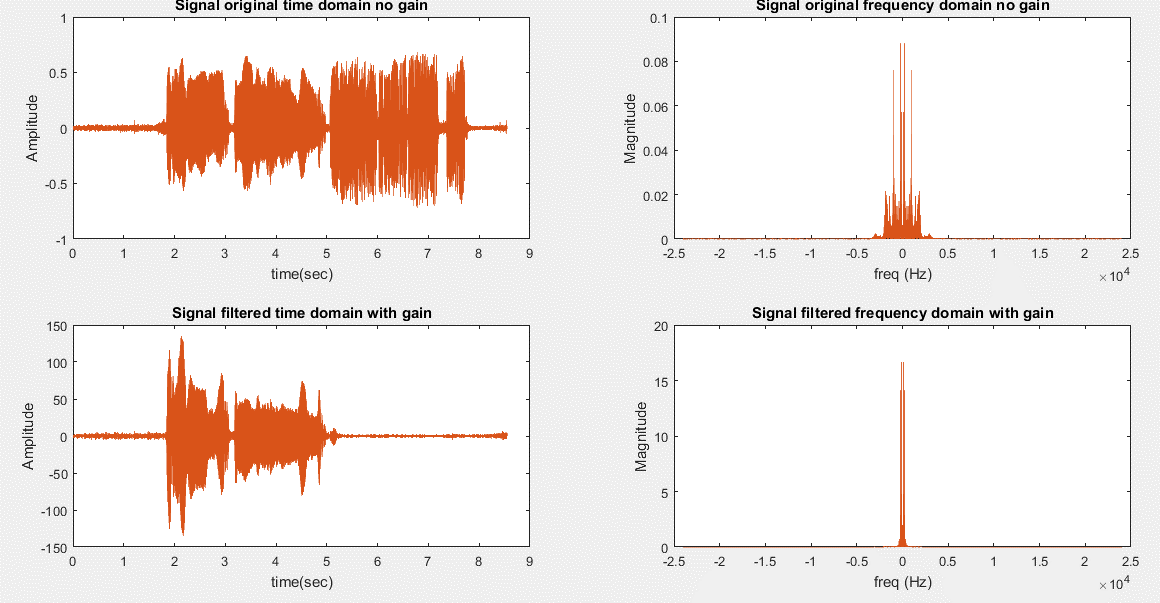
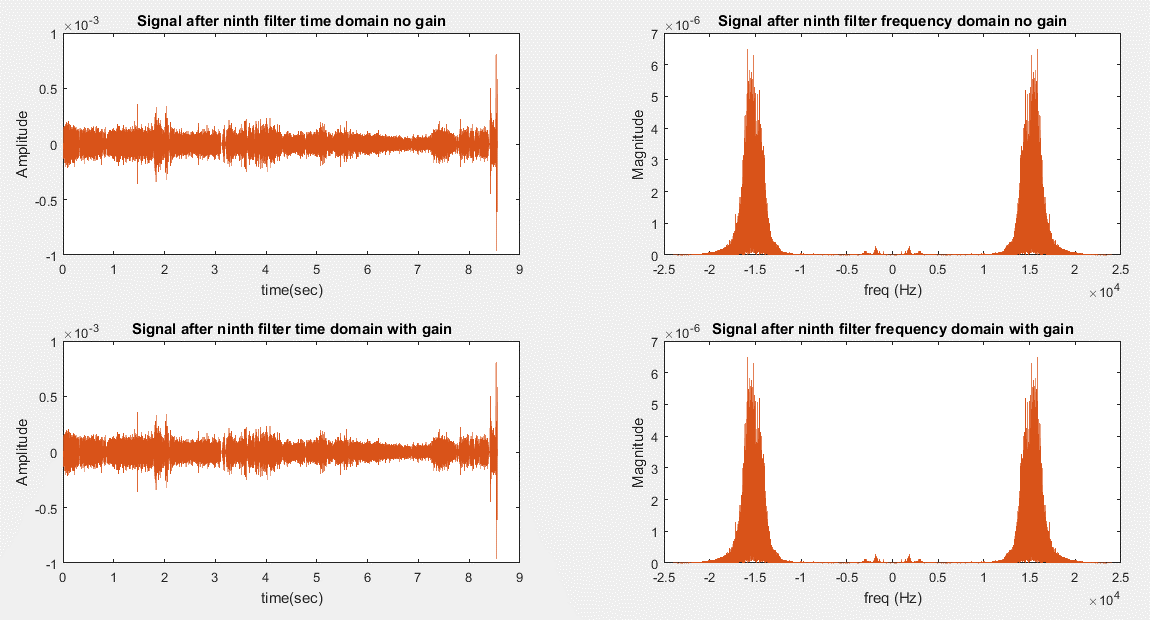
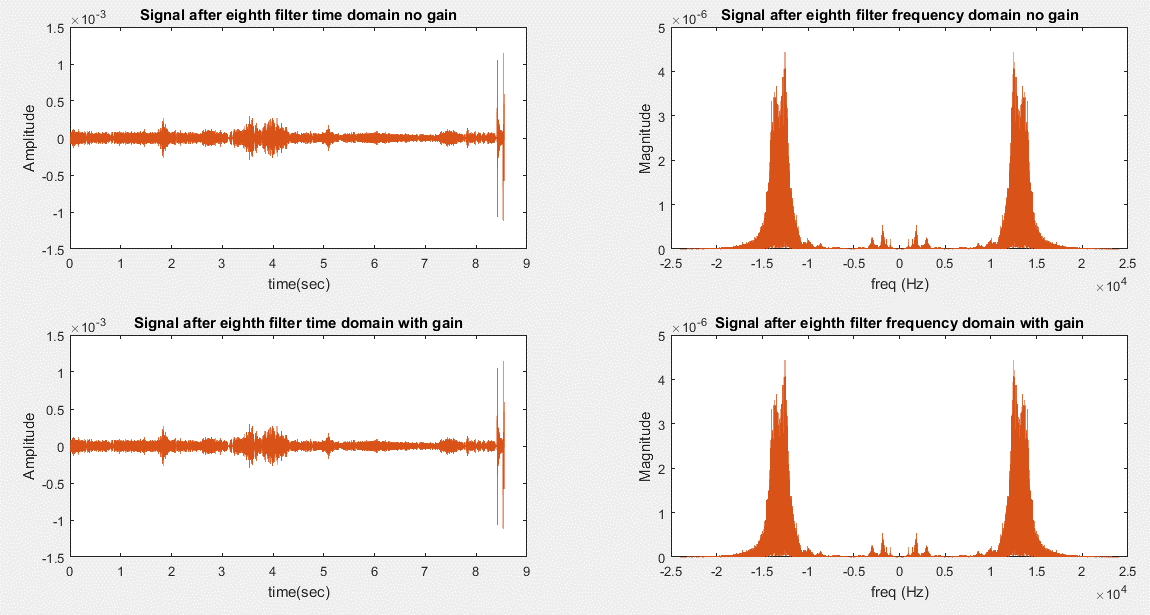
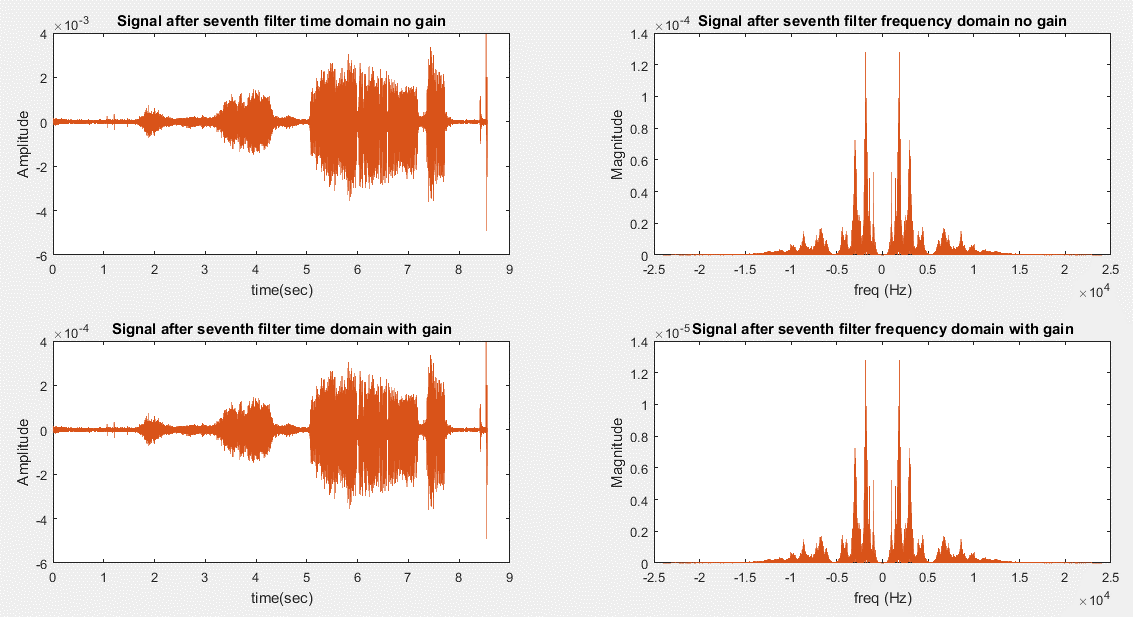
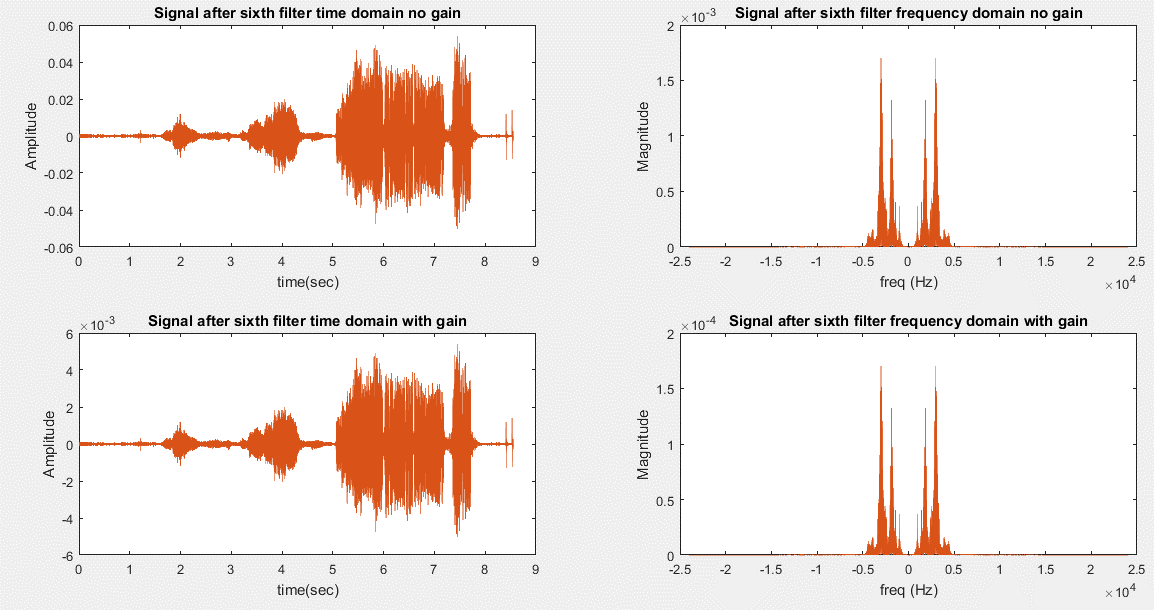
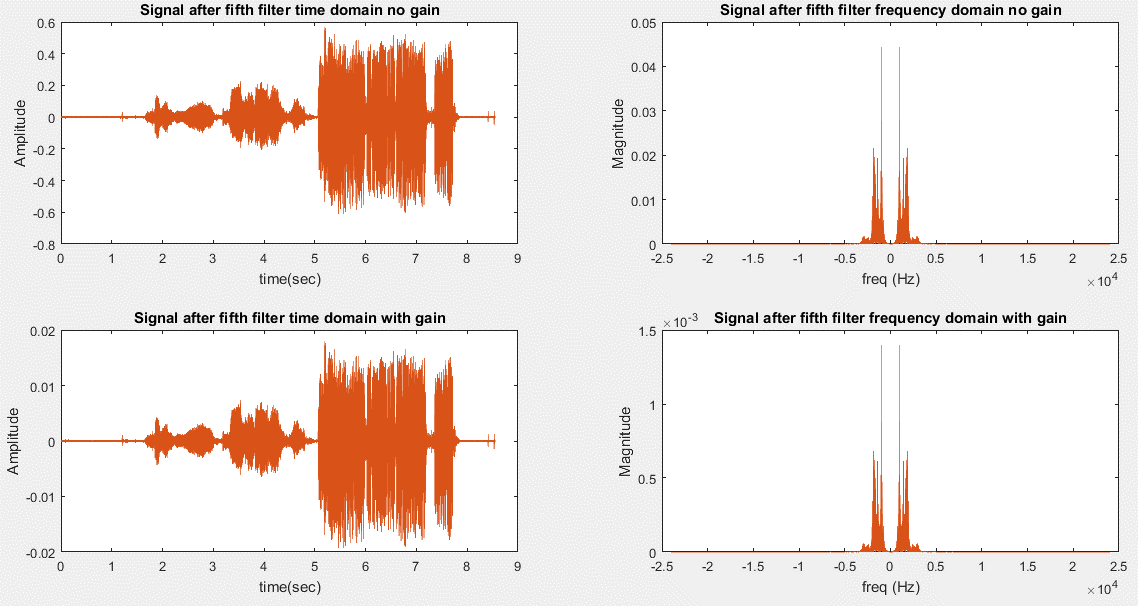
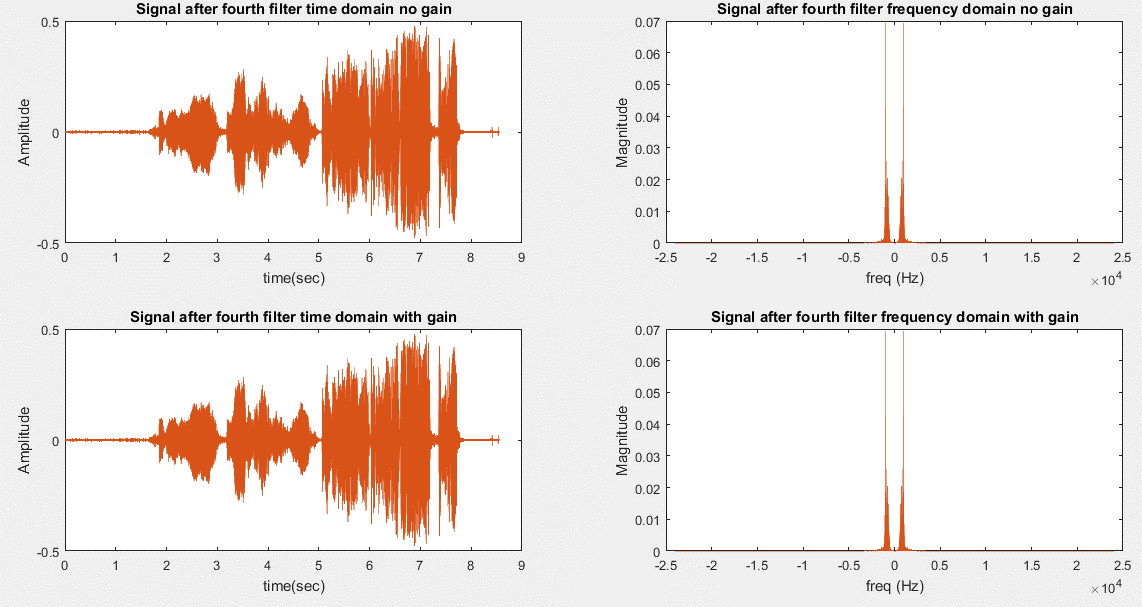
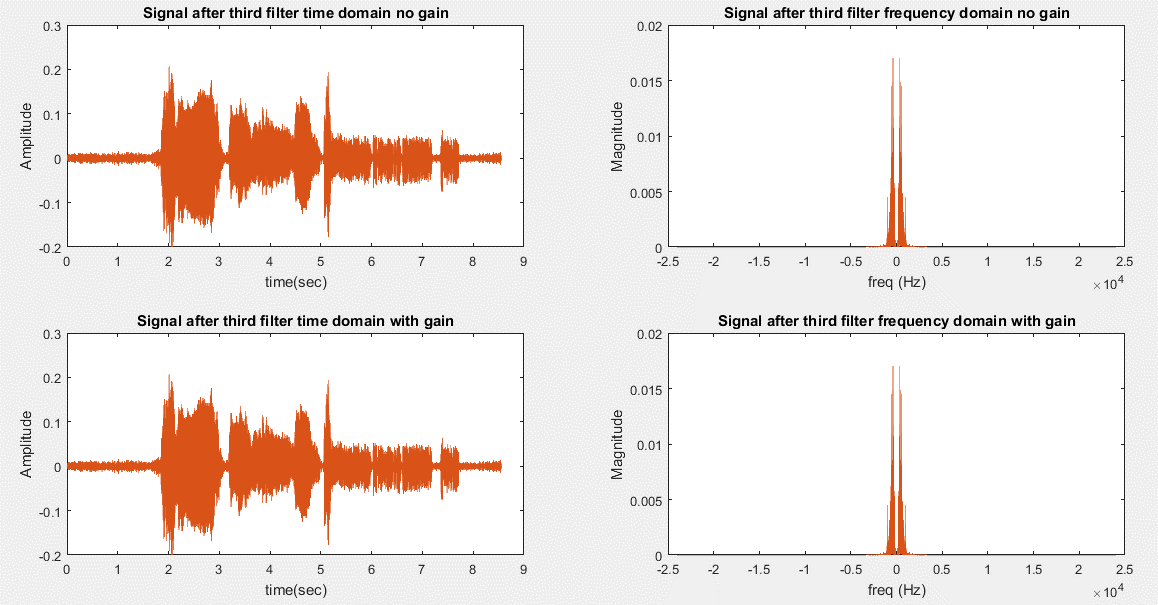




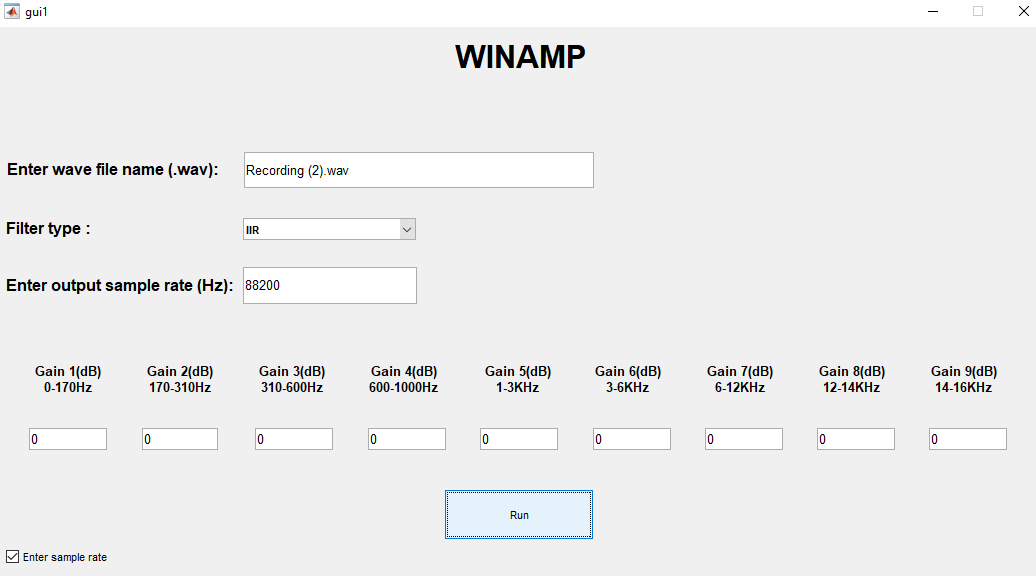


### **2-b-ii) All figures in time and frequency domain:**

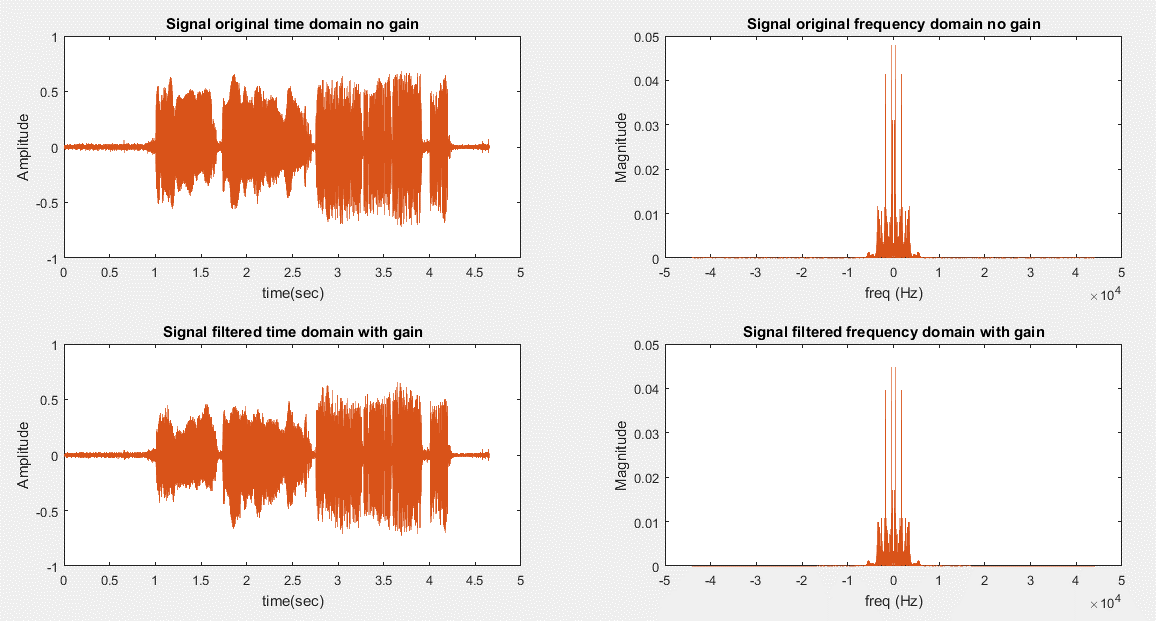




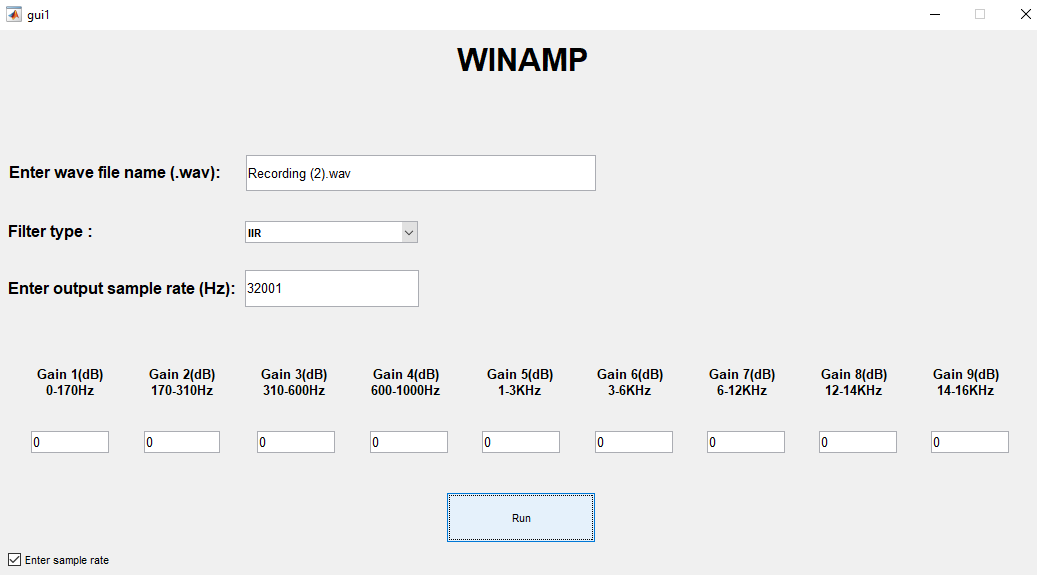
## **2-c) IIR with file double sample rate:**



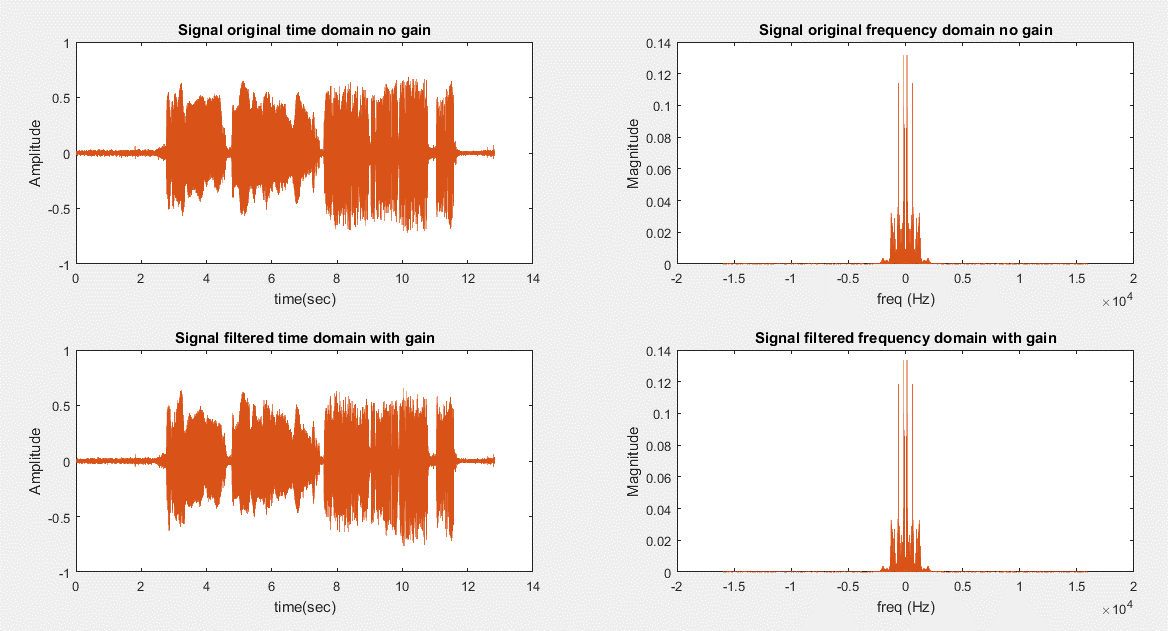
### **2-c-i) Final figure in time and frequency domain:**



## **2-d) IIR with file half sample rate:**



### **2-d-i) Final figure in time and frequency domain:**



# **3) Error handling:**

