National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

EE-232 Signals & Systems

Task 1 Report

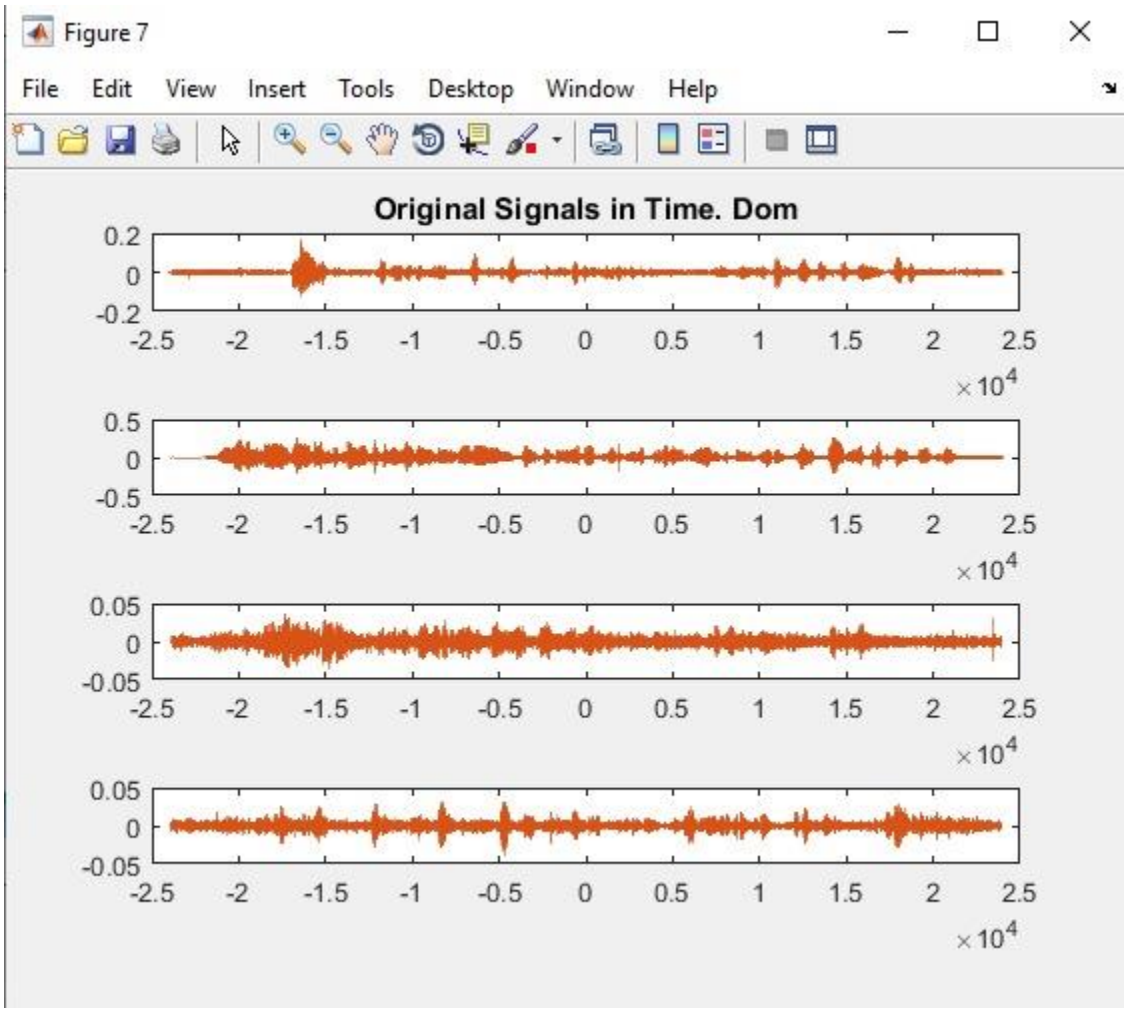
Group Members

1.	**Amur Saqib Pal**	**288334**
2.	**Junaid Ali**	**290927**
3.	**Faiez Kazi**	**295848**
4.	**Sannan Abbasi**	**293786**

Steps Performed in Task 1

1. Time and Frequency Domain Representation of Audio Signals:

Audio signals are read using audioread() and then plotted, first in time domain and then in frequency domain through the application of Fast Fourier Transform.



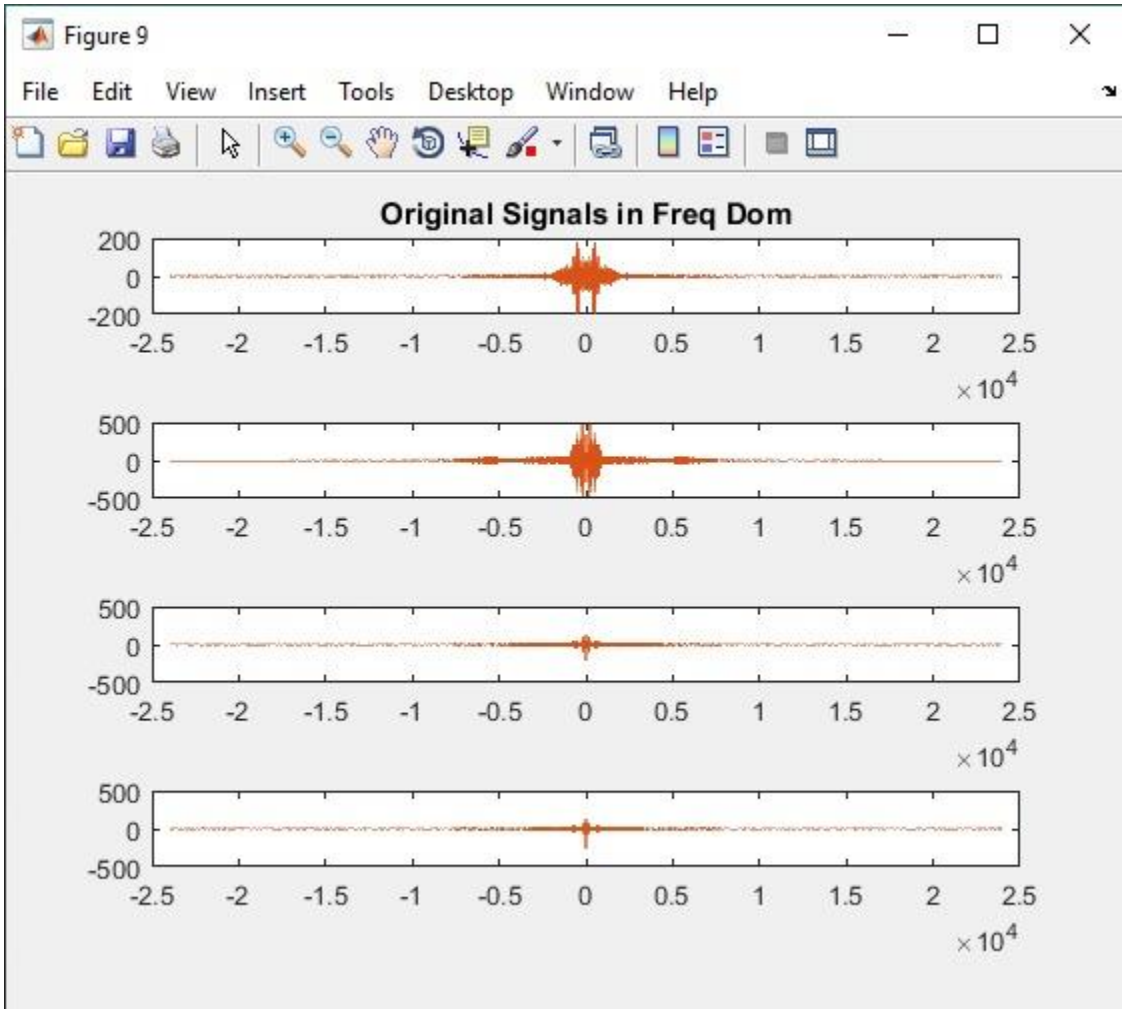
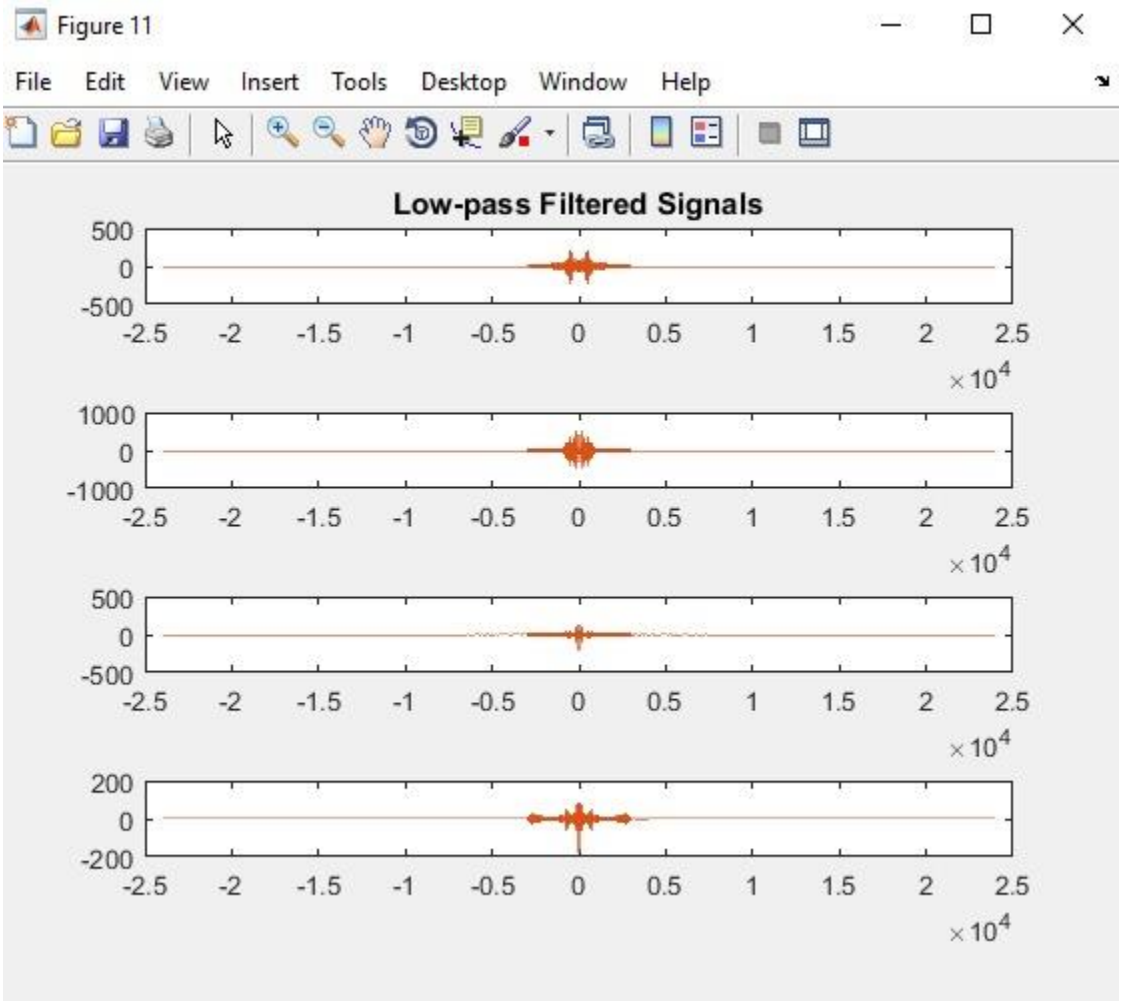


Figure 9
File Edit View Insert Tools Desktop Window Help
Original Signals in Freq Dom
200
0
-200
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5
×10^4
500
0
-500
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5
×10^4
500
0
-500
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5
×10^4
500
0
-500
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5
×10^4

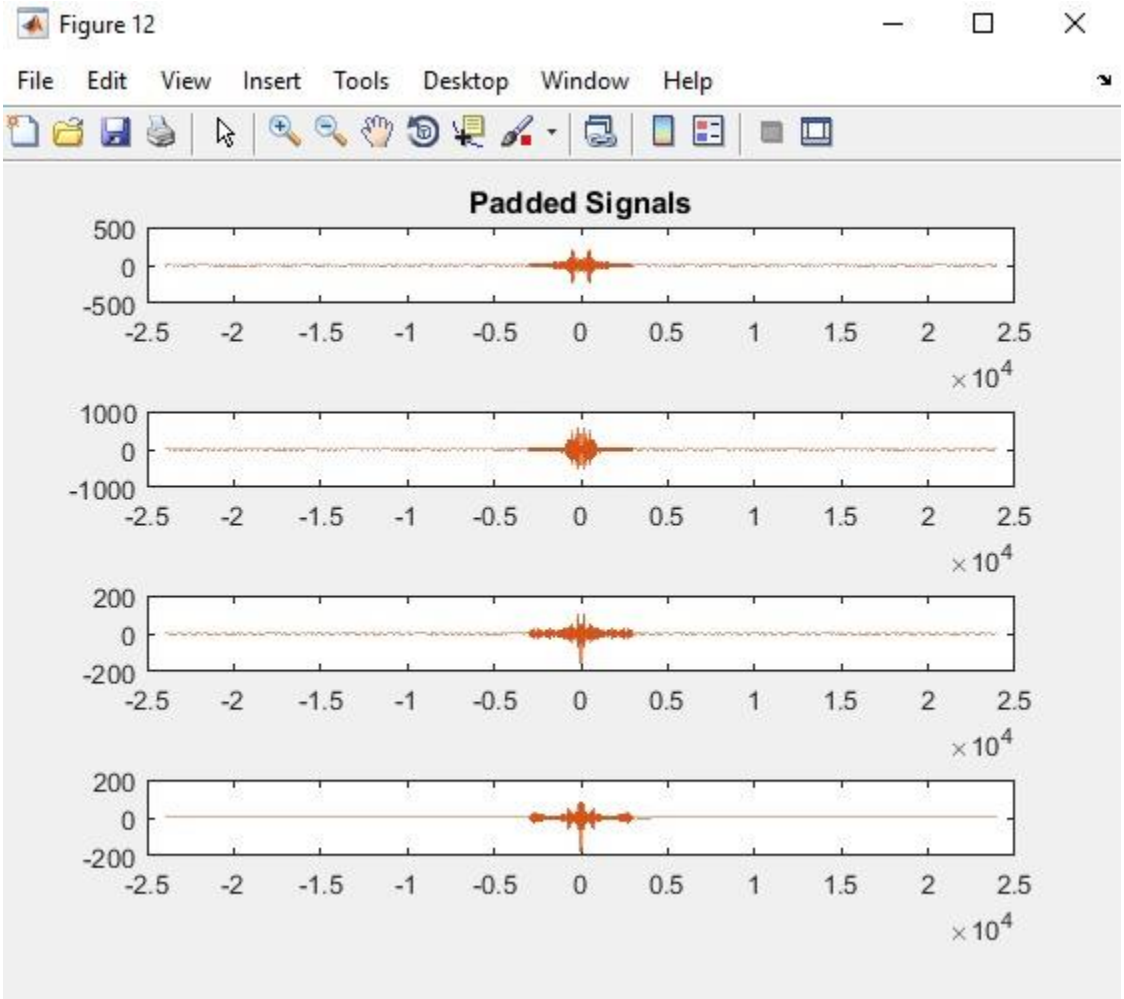
2. Signals filtered through Low-Pass filter:

Next, we designed a low-pass filter of 3000Hz frequency and proceeded by passing the input signals through said filter.



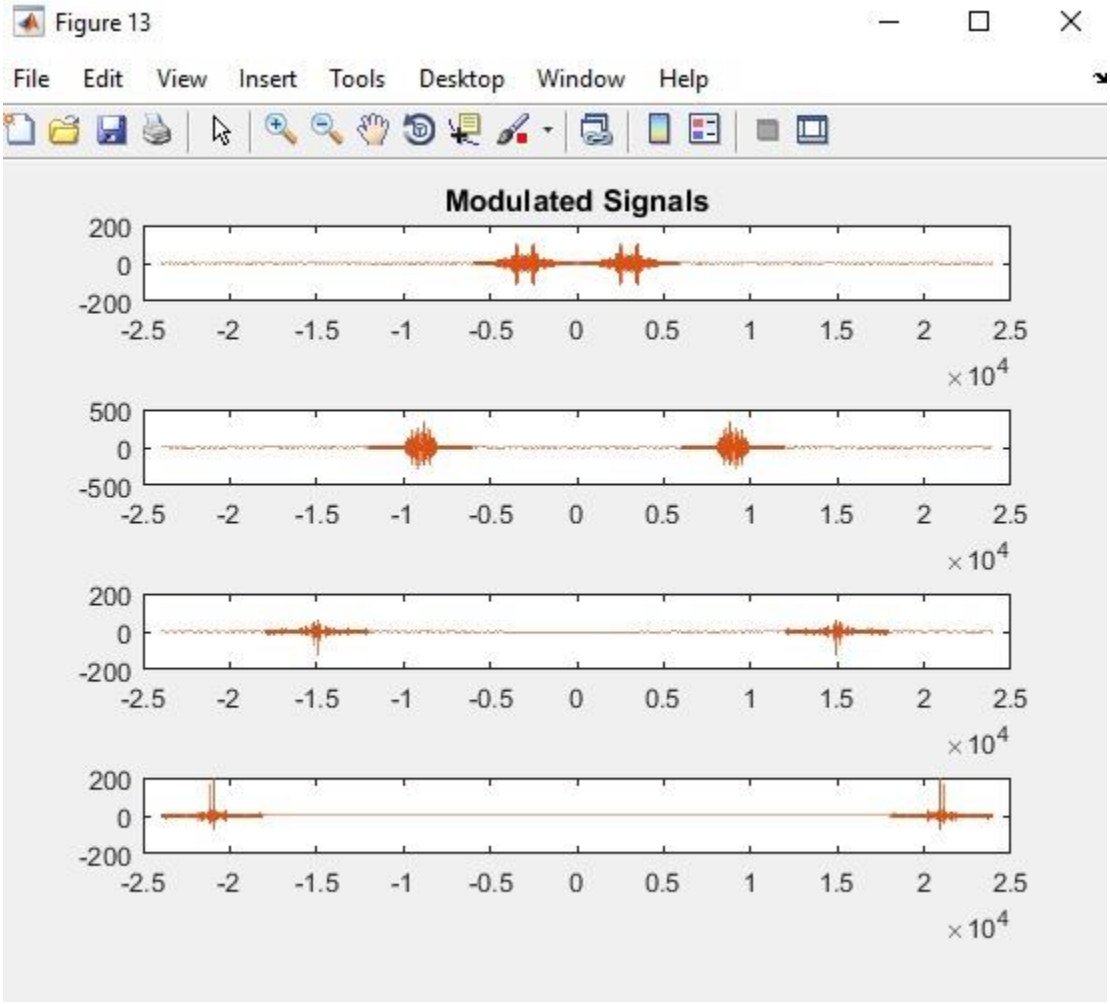
3. Padded Signals

The filtered signals are then post-padded with zeroes to make sure that their vectors are eligible for the next step, namely modulation.



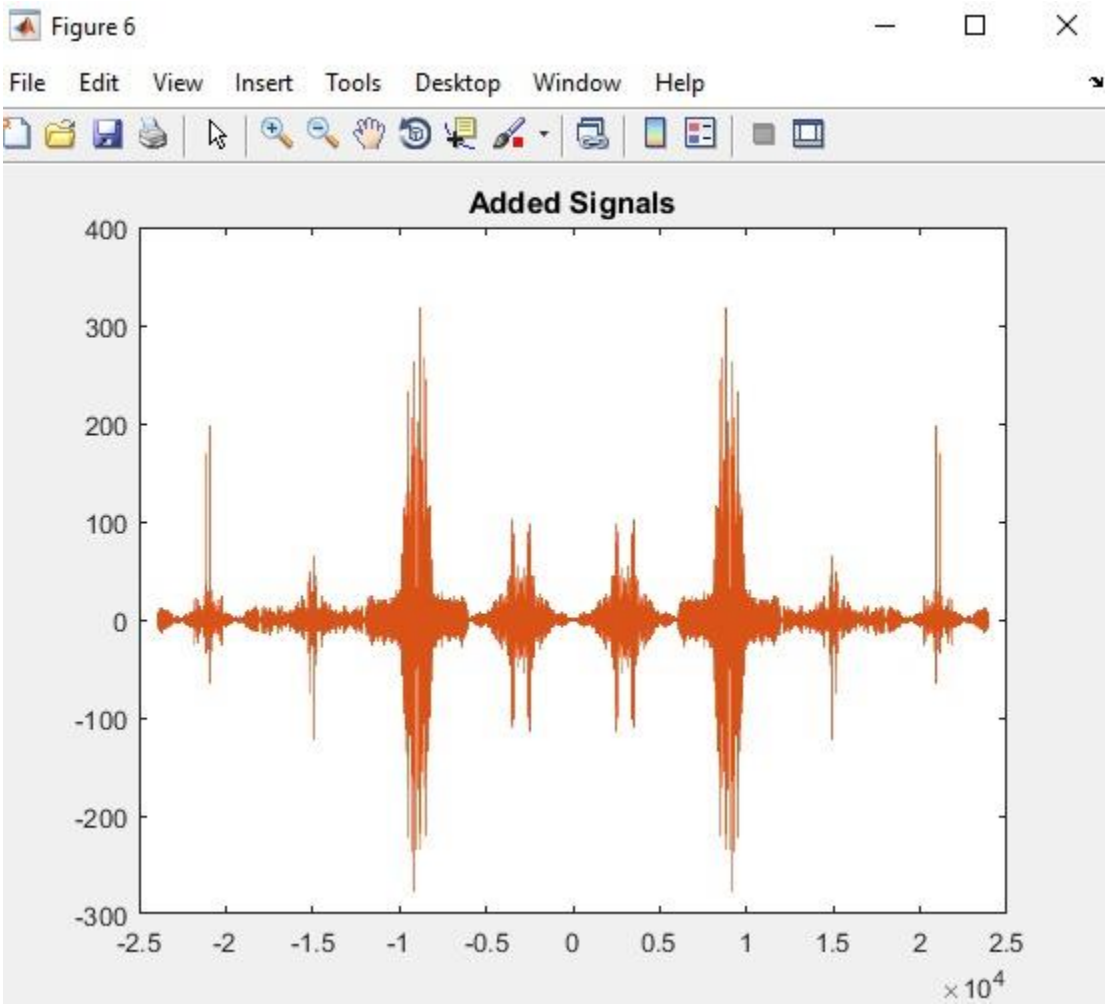
4. Modulating the signals with cosine function:

The padded signals are now to be multiplied by cosines of given frequencies. But first, we must declare a time interval variable for these cosines. It is worth mentioning that this variable must be 2 dimensional and transposed, so we can elementally-multiply them with our signals.



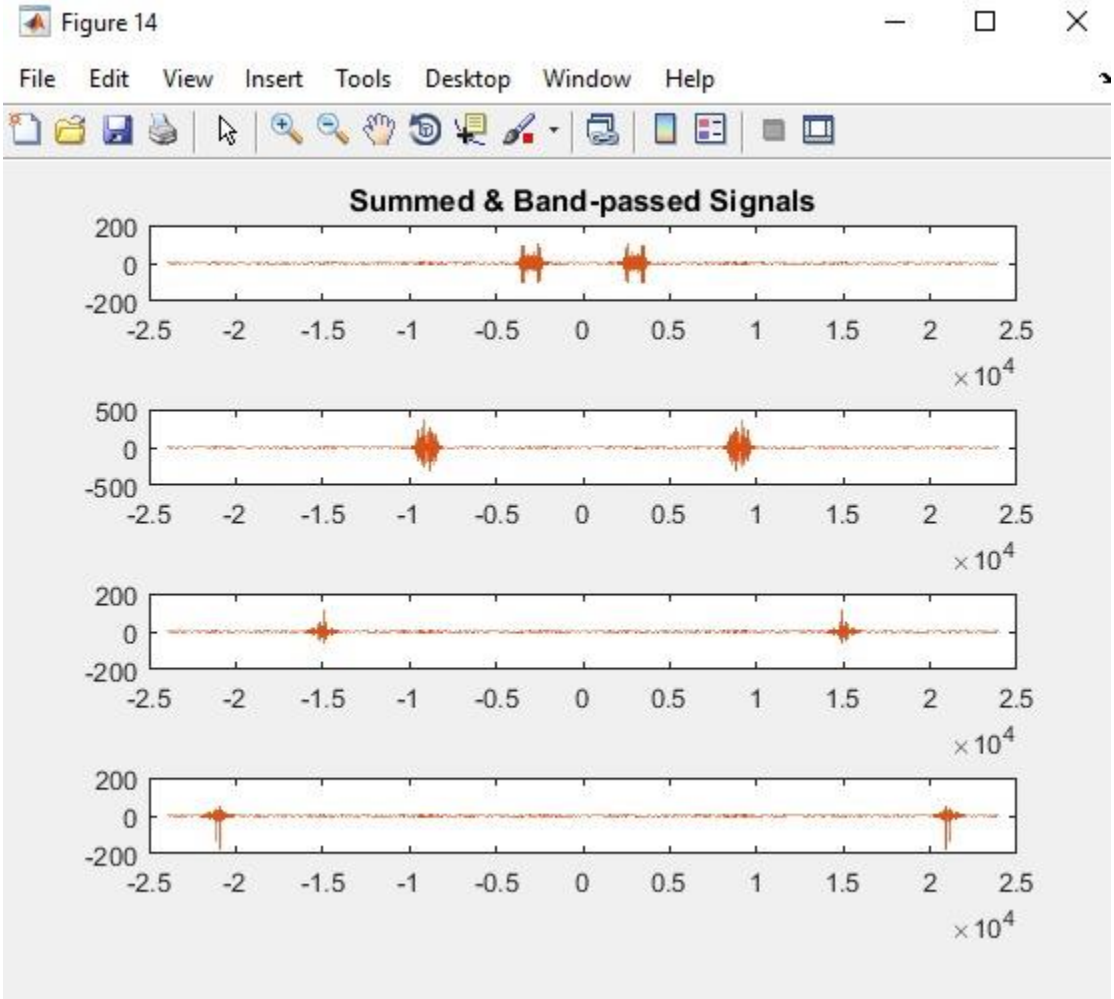
5. Summing the modulated signals:

The modulated signals from the previous steps are now simply added together and displayed.



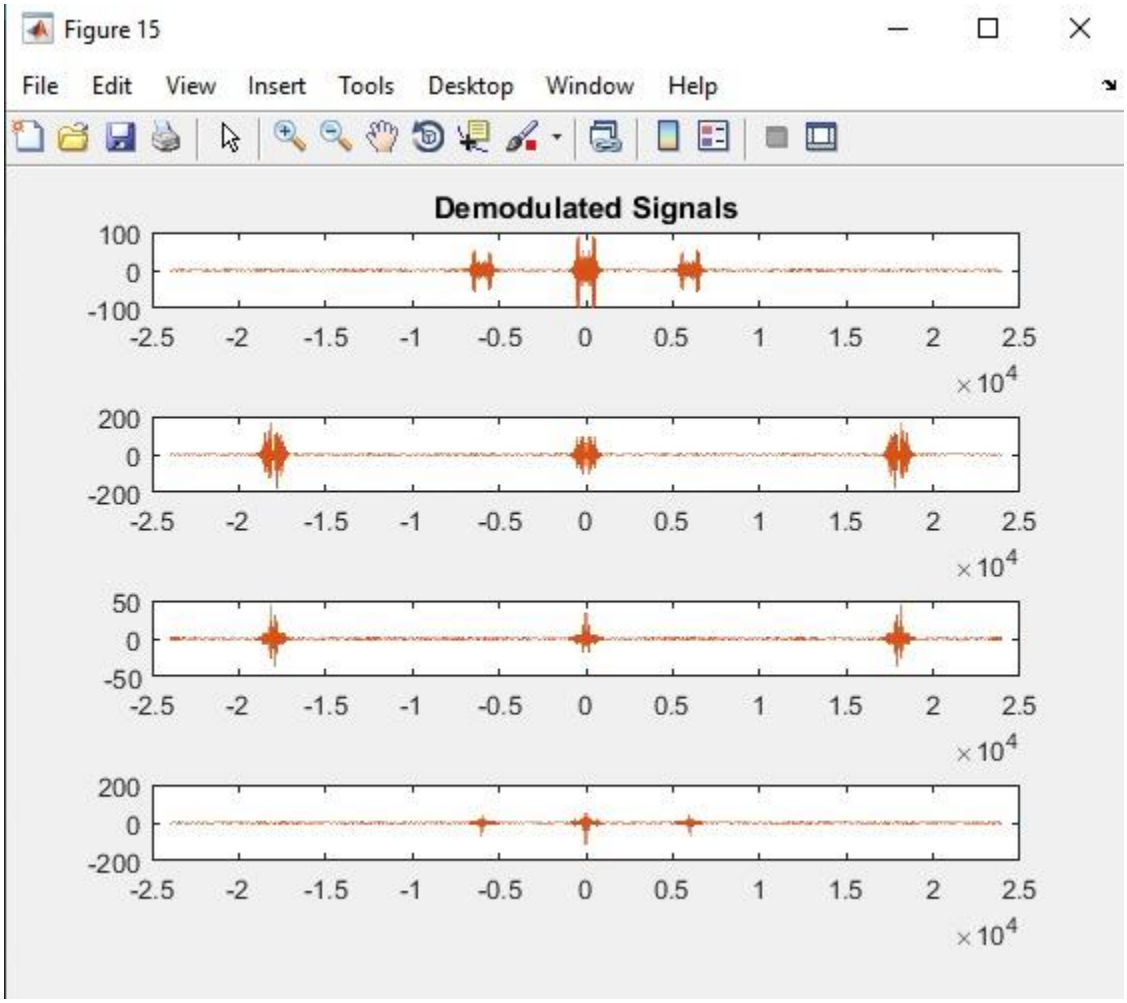
6. Filtered the Modulated Signals and Added Signals through Band-Pass filter:

In this step, we pass the added signal through bandpass filters of different frequencies (given) to retrieve the following signals.



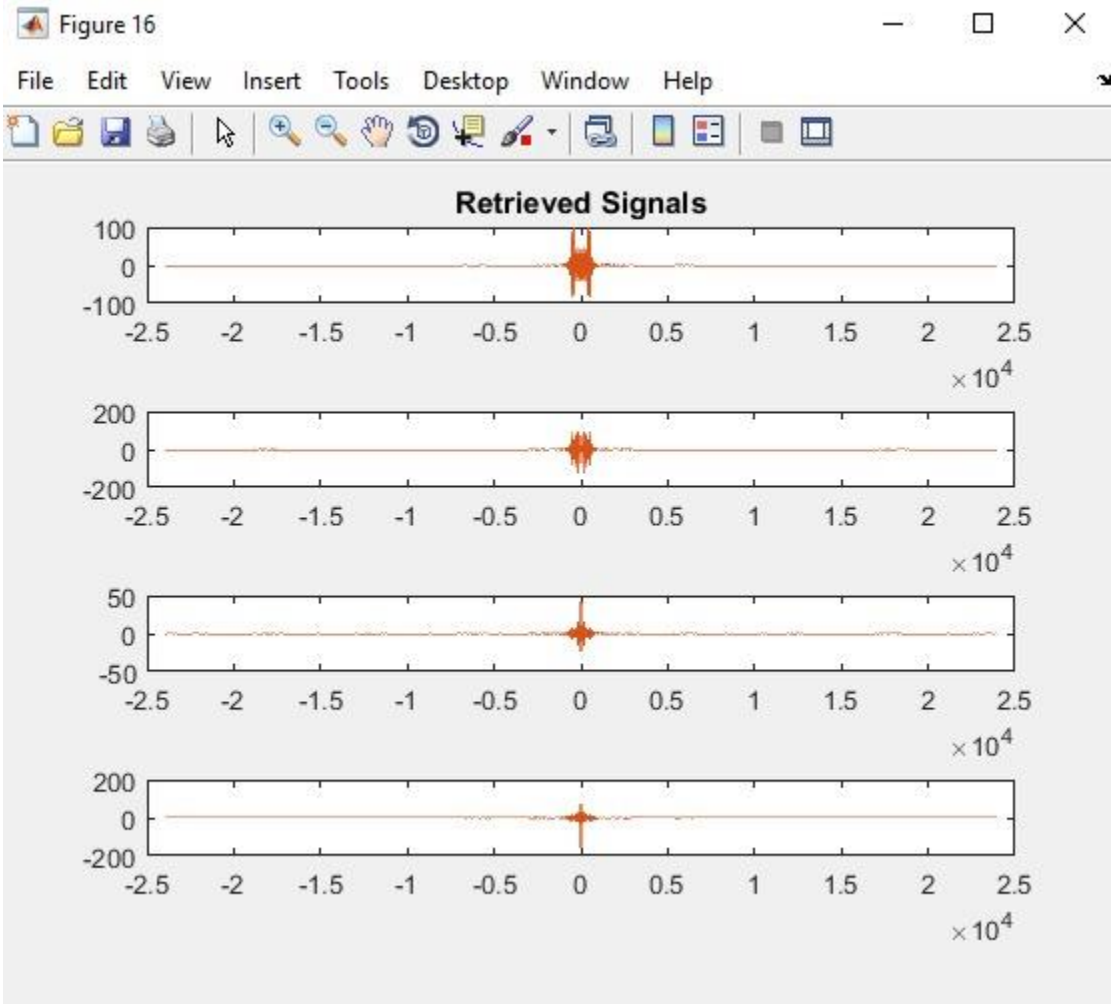
7. Demodulating the Signals:

Here, we simply demodulate the band-pass-filtered signals by multiplying them with their respective cosines again.



8. Filtered the Demodulated Signals through Low-Pass filter:

Lastly, the demodulated signals are passed through a low-pass filter to retrieve the original signals.



Division of Work:

Task 1 of the project was completed by Amur and Sannan. Both worked together in compiling and formatting the code. They both were responsible for completing the Task 1 report as well.

Code

Source

```
clc
clear all
cd 'C:\Users\ALTAMASH.2712080\Desktop\abc';
% ******************************Audio Signals fed into
matlab**********************************************************************
***********************************************
[sig1,fs1] = audioread('C:\Users\ALTAMASH.2712080\Desktop\abc\amur.m4a');
[sig2,fs2] = audioread('C:\Users\ALTAMASH.2712080\Desktop\abc\sannan.mp4');
[sig3,fs3] = audioread('C:\Users\ALTAMASH.2712080\Desktop\abc\junaid.m4a');
[sig4,fs4] = audioread('C:\Users\ALTAMASH.2712080\Desktop\abc\faiez.m4a');

% ******************************Converting Signals into Frequency
Domain************************
[sig1_freq, L1] = freqDom(sig1,fs1);
[sig2_freq, L2] = freqDom(sig2,fs2);
[sig3_freq, L3] = freqDom(sig3,fs3);
[sig4_freq, L4] = freqDom(sig4,fs4);

% ******************************Plotting of Signals in Freq and Time
Domains*******************
plotSignals(sig1,sig2,sig3,sig4,L1,L2,L3,L4,'Original Signals in Time. Dom');
plotSignals(sig1_freq,sig2_freq,sig3_freq,sig4_freq,L1,L2,L3,L4,'Original
Signals in Freq Dom');

% ******************************Calling the Low Pass filter
function*****************************
LPF = filter2();

% ******************************Filtering the Audion
Signals***********************************
```

```
sig1_filtered = filter(LPF,sig1);
sig2_filtered = filter(LPF,sig2);
sig3_filtered = filter(LPF,sig3);
sig4_filtered = filter(LPF,sig4);

% ******************************Converting the Filtered Signals into freq
domain******************
[sig1_freq, L1] = freqDom(sig1_filtered,fs1);
[sig2_freq, L2] = freqDom(sig2_filtered,fs2);
[sig3_freq, L3] = freqDom(sig3_filtered,fs3);
[sig4_freq, L4] = freqDom(sig4_filtered,fs4);

% ******************************Plotting of Filtered Signals in freq
domain***********************************
plotSignals(sig1_freq,sig2_freq,sig3_freq,sig4_freq,L1,L2,L3,L4,'Low-pass
Filtered Signals');

% ******************************Finding the largest
signal*********************************
S1 = length(sig1_filtered);
S2 = length(sig2_filtered);
S3 = length(sig3_filtered);
S4 = length(sig4_filtered);
Lmax = max([S1, S2, S3, S4]);

% ******************************Padding the Filtered
Signals*********************************
sig1_padded = padarray(sig1_filtered, Lmax-S1, 0, 'post');
sig2_padded = padarray(sig2_filtered, Lmax-S2, 0, 'post');
sig3_padded = padarray(sig3_filtered, Lmax-S3, 0, 'post');
sig4_padded = padarray(sig4_filtered, Lmax-S4, 0, 'post');
[temp, LmaxVec] = freqDom(sig1_padded, fs1); %getting a vector for Lmax
```

```
% ******************************Converting the Padded Signals into freq
domain*****************
[sig1_freq, L1] = freqDom(sig1_padded,fs1);
[sig2_freq, L2] = freqDom(sig2_padded,fs2);
[sig3_freq, L3] = freqDom(sig3_padded,fs3);
[sig4_freq, L4] = freqDom(sig4_padded,fs4);

% ******************************Plotting the Padded Signals in freq
domain*****************
plotSignals(sig1_freq,sig2_freq,sig3_freq,sig4_freq,LmaxVec,LmaxVec,LmaxVec,L
maxVec,'Padded Signals');

% ******************************Taking transpose so that matrices are ready
for elemental
multiplication*****************************************************************
*****
t1 = transpose([linspace(0,Lmax/fs1,Lmax) ; linspace(0,Lmax/fs1,Lmax)]);
t2 = transpose([linspace(0,Lmax/fs2,Lmax) ; linspace(0,Lmax/fs2,Lmax)]);
t3 = transpose([linspace(0,Lmax/fs3,Lmax) ; linspace(0,Lmax/fs3,Lmax)]);
t4 = transpose([linspace(0,Lmax/fs4,Lmax) ; linspace(0,Lmax/fs4,Lmax)]);

% ******************************Modulating the Signals with cosine of
assigned frequencies**********************************
fm1 = 3000;
fm2 = 9000;
fm3 = 15000;
fm4 = 21000;
mod1 = cos(2*pi*fm1*t1);
mod2 = cos(2*pi*fm2*t2);
mod3 = cos(2*pi*fm3*t3);
mod4 = cos(2*pi*fm4*t4);
modded1 = sig1_padded.*mod1;
modded2 = sig2_padded.*mod2;
```

```
modded3 = sig3_padded.*mod3;
modded4 = sig4_padded.*mod4;

% ******************************Converting Modulated Signals into freq
domain*******************************
[sig1_freq, L1] = freqDom(modded1,fs1);
[sig2_freq, L2] = freqDom(modded2,fs2);
[sig3_freq, L3] = freqDom(modded3,fs3);
[sig4_freq, L4] = freqDom(modded4,fs4);

% ******************************Plotting Modulated Signals in freq
domain************************
plotSignals(sig1_freq,sig2_freq,sig3_freq,sig4_freq,L1,L2,L3,L4,'Modulated
Signals');

% ******************************Adding then Converting and Plotting the
Modulated Signals in freq domain*****************************
added = modded1 + modded2 + modded3 + modded4;
sig_addedfreq = freqDom(added,fs1);
figure;
plot(LmaxVec,sig_addedfreq);
title('Added Signals');

% ******************************Calling Band-Pass filter
functions*******************************
BPF1 = filter3();
BPF2 = filter4();
BPF3 = filter5();
BPF4 = filter6();

% ******************************Filtering the Summed Signal through Band-
Pass filter*****************************************
added_filtered1 = filter(BPF1,added);
```

```
added_filtered2 = filter(BPF2,added);
added_filtered3 = filter(BPF3,added);
added_filtered4 = filter(BPF4,added);
%
% ******************************Converting the Filtered Signals into freq
domain*********************************************
[sig1_freq, L1] = freqDom(added_filtered1,fs1);
[sig2_freq, L2] = freqDom(added_filtered2,fs2);
[sig3_freq, L3] = freqDom(added_filtered3,fs3);
[sig4_freq, L4] = freqDom(added_filtered4,fs4);

% ******************************Plotting Filtered Signals in freq
domain*****************************************************
plotSignals(sig1_freq,sig2_freq,sig3_freq,sig4_freq,L1,L2,L3,L4,'Summed &
Band-passed Signals');

% ******************************Demodulating the
Signals*************************************************************************
**
demodded1 = added_filtered1.*mod1;
demodded2 = added_filtered2.*mod2;
demodded3 = added_filtered3.*mod3;
demodded4 = added_filtered4.*mod4;

% ******************************Converting the Demodualted Signals in freq
domain*****************************************************
[sig1_freq, L1] = freqDom(demodded1,fs1);
[sig2_freq, L2] = freqDom(demodded2,fs2);
[sig3_freq, L3] = freqDom(demodded3,fs3);
[sig4_freq, L4] = freqDom(demodded4,fs4);

% ******************************Plotting Demodulated Signals in freq
domain*************************************************************
```

```
plotSignals(sig1_freq,sig2_freq,sig3_freq,sig4_freq,L1,L2,L3,L4,'Demodulated
Signals');

% ******************************Filtering the Demodulated Signals through
low pass filter*************************************************
demoddedf1 = filter(LPF, demodded1);
demoddedf2 = filter(LPF, demodded2);
demoddedf3 = filter(LPF, demodded3);
demoddedf4 = filter(LPF, demodded4);

% ******************************Converting Filtered Signals into freq
domain*********************************************************
[sig1_freq, L1] = freqDom(demoddedf1,fs1);
[sig2_freq, L2] = freqDom(demoddedf2,fs2);
[sig3_freq, L3] = freqDom(demoddedf3,fs3);
[sig4_freq, L4] = freqDom(demoddedf4,fs4);

% ******************************Plotting the Filtered Signals in freq
domain*********************************************************
plotSignals(sig1_freq,sig2_freq,sig3_freq,sig4_freq,L1,L2,L3,L4,'Retrieved
Signals');

% ******************************Playing the Finally Retrieved
Audios*********************************************************************
audiowrite('C:\Users\ALTAMASH.2712080\Desktop\abc\amurRegen.m4a',demoddedf1,f
s1);
audiowrite('C:\Users\ALTAMASH.2712080\Desktop\abc\sannanRegen.mp4',demoddedf2
,fs2);
audiowrite('C:\Users\ALTAMASH.2712080\Desktop\abc\junaidRegen.m4a',demoddedf3
,fs3);
audiowrite('C:\Users\ALTAMASH.2712080\Desktop\abc\faiezRegen.m4a',demoddedf4,
fs4);
```

Supplementary Functions:

1. freqDom

```
% This function converts time domain signals into freq domain signals



function [y,n] = freqDom(sig,fs)
L = length(sig);
y = fft(sig,L);
y = fftshift(y);
n = ((-L/2:L/2-1)*(fs/L));
end
```

2. plotSignals

```
function [] = plotSignals(y1,y2,y3,y4,L1,L2,L3,L4,titleStr)
figure;
subplot(411);
plot(L1,y1);
title(titleStr);
subplot(412);
plot(L2,y2);
subplot(413);
plot(L3,y3);
subplot(414);
plot(L4,y4);
end
```

3. filter1

```
function Hd = filter1
%FILTER1 Returns a discrete-time filter object.

% MATLAB Code
```

```
% Generated by MATLAB(R) 9.2 and the DSP System Toolbox 9.4.
% Generated on: 20-May-2021 20:51:26

% Equiripple Lowpass filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 48000;  % Sampling Frequency

N     = 10;    % Order
Fpass = 0;     % Passband Frequency
Fstop = 3000;  % Stopband Frequency
Wpass = 1;     % Passband Weight
Wstop = 1;     % Stopband Weight
dens  = 20;    % Density Factor

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, [0 Fpass Fstop Fs/2]/(Fs/2), [1 1 0 0], [Wpass Wstop], ...
           {dens});
Hd = dfilt.dffir(b);

% [EOF]
```

4. filter2

```
function Hd = filter2
%FILTER2 Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.2 and the Signal Processing Toolbox 7.4.
% Generated on: 21-May-2021 17:57:04

% Equiripple Lowpass filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 48000;  % Sampling Frequency

Fpass = 2900;               % Passband Frequency
Fstop = 3020;            % Stopband Frequency
Dpass = 0.057501127785;  % Passband Ripple
```

```
Dstop = 0.0001;          % Stopband Attenuation
dens  = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fpass, Fstop]/(Fs/2), [1 0], [Dpass, Dstop]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

% [EOF]
```

5. filter3

```
function Hd = filter3
%FILTER3 Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.2 and the Signal Processing Toolbox 7.4.
% Generated on: 25-May-2021 13:10:25

% Equiripple Bandpass filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 48000;  % Sampling Frequency

Fstop1 = 2000;            % First Stopband Frequency
Fpass1 = 2500;            % First Passband Frequency
Fpass2 = 3500;            % Second Passband Frequency
Fstop2 = 4000;            % Second Stopband Frequency
Dstop1 = 0.001;           % First Stopband Attenuation
Dpass  = 0.057501127785;  % Passband Ripple
Dstop2 = 0.001;           % Second Stopband Attenuation
dens   = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fstop1 Fpass1 Fpass2 Fstop2]/(Fs/2), [0 1 ...
                          0], [Dstop1 Dpass Dstop2]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

% [EOF]
```

6. filter4

```
function Hd = filter4
%FILTER4 Returns a discrete-time filter object.
```

```
% MATLAB Code
% Generated by MATLAB(R) 9.2 and the Signal Processing Toolbox 7.4.
% Generated on: 25-May-2021 15:08:58

% Equiripple Bandpass filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 48000;  % Sampling Frequency

Fstop1 = 8000;            % First Stopband Frequency
Fpass1 = 8500;            % First Passband Frequency
Fpass2 = 9500;            % Second Passband Frequency
Fstop2 = 10000;           % Second Stopband Frequency
Dstop1 = 0.001;           % First Stopband Attenuation
Dpass  = 0.057501127785;  % Passband Ripple
Dstop2 = 0.001;           % Second Stopband Attenuation
dens   = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fstop1 Fpass1 Fpass2 Fstop2]/(Fs/2), [0 1 ...
                          0], [Dstop1 Dpass Dstop2]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

% [EOF]
```

7. filter5

```
function Hd = filter5
%FILTER5 Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.2 and the Signal Processing Toolbox 7.4.
% Generated on: 25-May-2021 15:13:52

% Equiripple Bandpass filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 48000;  % Sampling Frequency

Fstop1 = 14000;           % First Stopband Frequency
Fpass1 = 14500;           % First Passband Frequency
Fpass2 = 15500;           % Second Passband Frequency
Fstop2 = 16000;           % Second Stopband Frequency
Dstop1 = 0.001;           % First Stopband Attenuation
```

```
Dpass  = 0.057501127785;  % Passband Ripple
Dstop2 = 0.001;           % Second Stopband Attenuation
dens   = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fstop1 Fpass1 Fpass2 Fstop2]/(Fs/2), [0 1 ...
                          0], [Dstop1 Dpass Dstop2]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

% [EOF]
```

8. filter6

```
function Hd = filter6
%FILTER6 Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.2 and the Signal Processing Toolbox 7.4.
% Generated on: 25-May-2021 15:14:39

% Equiripple Bandpass filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 48000;  % Sampling Frequency

Fstop1 = 20000;           % First Stopband Frequency
Fpass1 = 20500;           % First Passband Frequency
Fpass2 = 21500;           % Second Passband Frequency
Fstop2 = 22000;           % Second Stopband Frequency
Dstop1 = 0.001;           % First Stopband Attenuation
Dpass  = 0.057501127785;  % Passband Ripple
Dstop2 = 0.001;           % Second Stopband Attenuation
dens   = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fstop1 Fpass1 Fpass2 Fstop2]/(Fs/2), [0 1 ...
                          0], [Dstop1 Dpass Dstop2]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

% [EOF]
```