Signals Project

Name	sec	ID
Vivianne Emad	3	9220585
Nardin Milad	4	9220899

Table of figures

I. Image filtering and restoration

A. First of all, we will read our image then extract its three channels which are red, green, blue channels, then create color versions of the individual color channels using concatenation.

Figure 1: extracted color channels

B. For the required operation:

1. For edge detection, to get vertical edge it is required to apply a matrix $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$, to get horizontal edge it is required to apply a matrix $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}$.

It is required to get all edges (upper, down, left, right) so we will add all matrices

$$\text{Edges} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} =$$
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

The resulted filter is convolved with each of the color channel separately using conv2 resulting in an image with edges only.

Figure 2: before and after edge detection

2. Image sharpening is a process where edges are added to the original image, that's what we will do

$$\text{Sharpen} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

The resulted filter is convolved with each of the color channel separately resulting in a sharpened image.

Figure 3: before and after sharpening

3. To blur an image we will create a large matrix with all ones then divide it with the matrix size that's what is called averaging

   Blurring $= \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \div (3 * 3)$, the larger the matrix would be , the more blurred image we will get .so we will use function that generates a matrix of size 5, with all ones .

```
% Define a plurring filter.
plurring = ones(5)/25;
```

   Then we will convolve this filter with each of the color channels separately resulting in a blurred image.

Figure 4: before and after blurring

4. For motion blurred image in horizontal direction, we want a matrix that has row with all ones as if the original image is added to sequence of images each of them is shifted horizontally.

   Motion $= \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$, for more precise results we will use a function that generates a matrix of large size N with all zeros, Then we will use another function to put ones in the middle row. Then convolve the filter with each color channel separately so that the resulted image is horizontally blurred.

```
motion = zeros(25);
motion (12,1:25)=1;
```

Figure 5: horizontally blurred

C. Using Fourier transform

We will perform Fourier transform on each color channel of the original image and the motion blurring filter, then we will multiply the fft2 of filter with each of the fft2 (color channels), which is equivalent to convolution in time domain, then the resulted image will be converted back to time domain then display the real part of it.

To restore the original image, we will divide each of the resulted color channel after applying the filter (the Blurred one) by the Fourier Transform of the filter. to avoid dividing by zero, a small value (eps) is added to the fft2 of filter and this division is equivalent to getting the inverse of filter in time domain, then apply ifft2 to the resulted image then display the real part of it (recombined image).

```
% adding eps to avoid dividing by zero
outri = (ifft2(fft2(outr) ./(ftmotion+eps)));
outgi = (ifft2(fft2(outg) ./(ftmotion+eps)));
outbi = (ifft2(fft2(outb) ./(ftmotion+eps)));
```

Figure 6: original, motional blurred & recombined using Fourier Transform and Fourier inverse

II. Communication system simulation

a. At first we recorded 2 segments of our voice.

We chose high value for sampling frequency fs=40000 because the greater the sampling rate, the more favorable the outcome and high quality.

b. Then we designed a LPF filter to filter each of the signals. We tried more than one filter in order to get the best sound quality. We tried Butterworth and Chebyshev type 2 and Elliptic and we chose Elliptic because it was better. After testing a lot of for cutoff frequencies, we chose the best quality. In the following image you see the details of the filter :

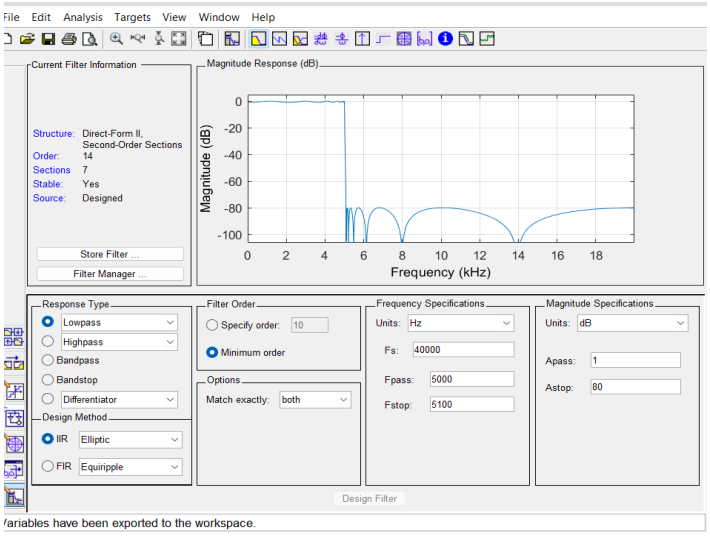

Figure 7: low pass filter_1

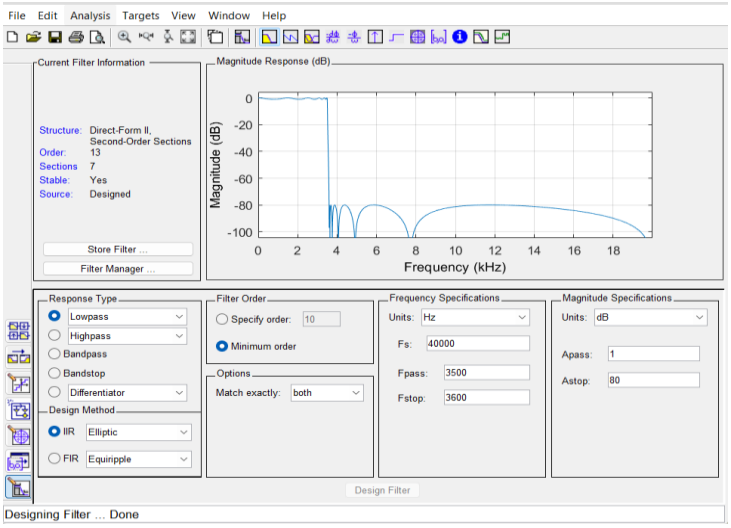

Figure 8: low pass filter_2

Then we made plotting to the frequency response of the best filter to each audio.

c. To plot before and after filtering

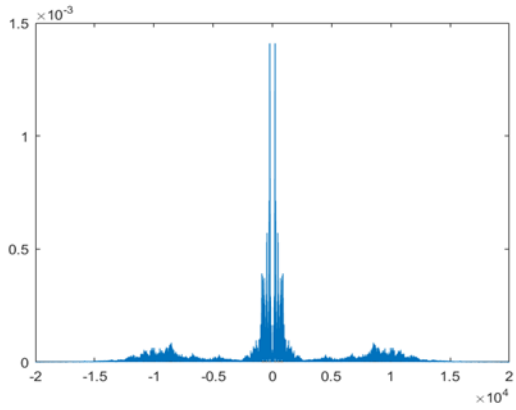

Figure 9: original signal1

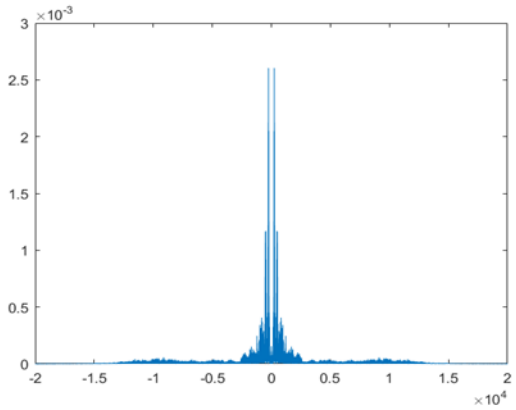

Figure 10: original signal2

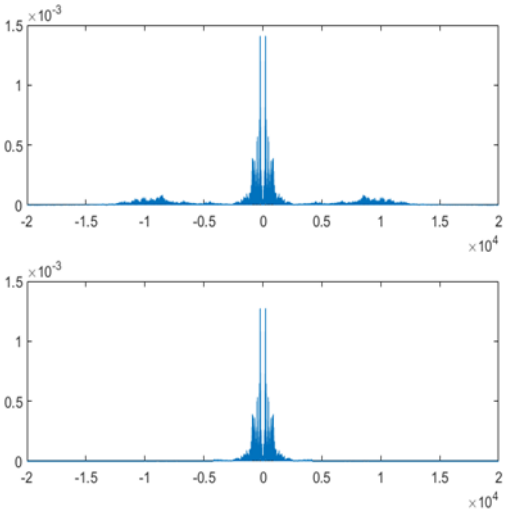

Figure 11: Before & after applying LPF on signal_1

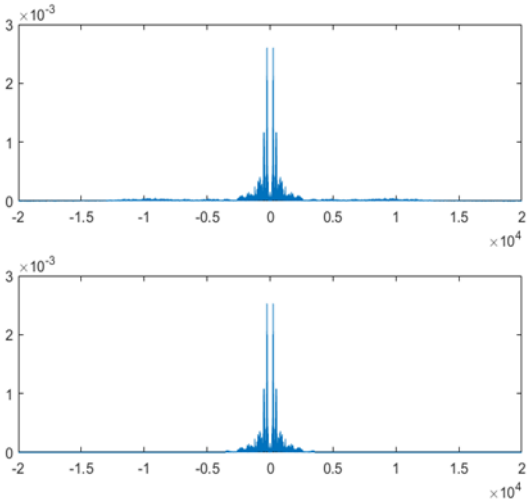

Figure 12: Before & after applying LPF on signal_2

d. Amplitude modulation

We chose carrier frequencies 5000, 15000 because it must be less than half the sampling frequency (40000) and the difference must be big enough so that the signals will not interfere.
Each signal is multiplied with cos (2*pi*fc*t) to be shifted in both directions

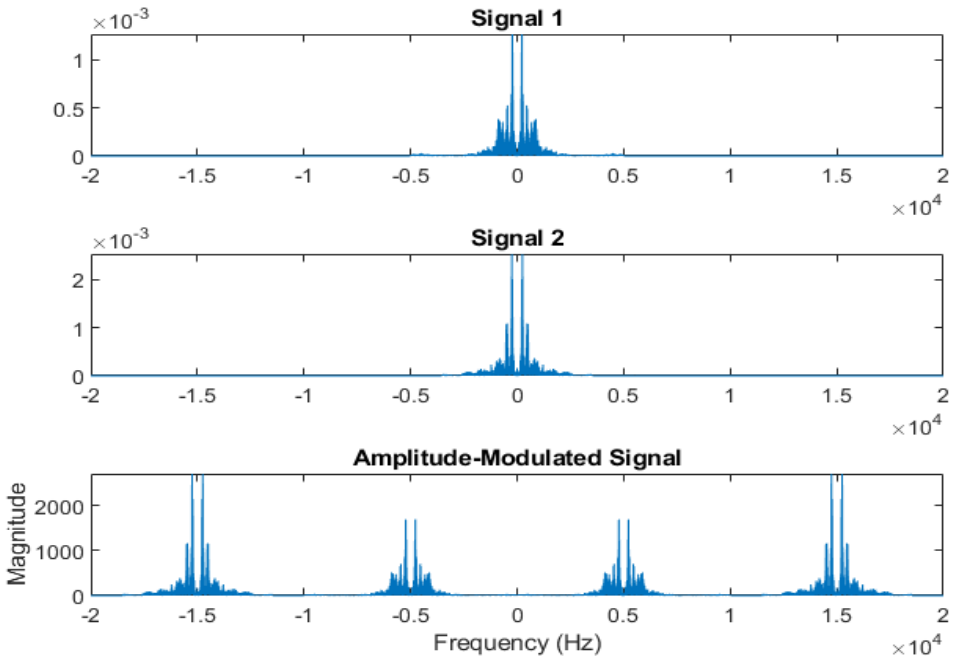

Figure 13: Amplitude Modulation

e. Demodulation

the summation signal is multiplied by cos again to be shifted at 0 and apply low pass filter on it.

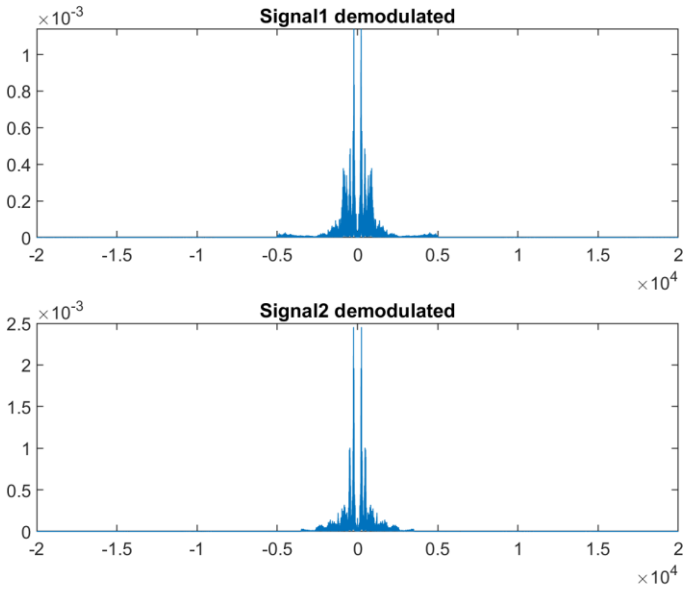

Figure 14: signals 1&2 after demodulation

Block diagram

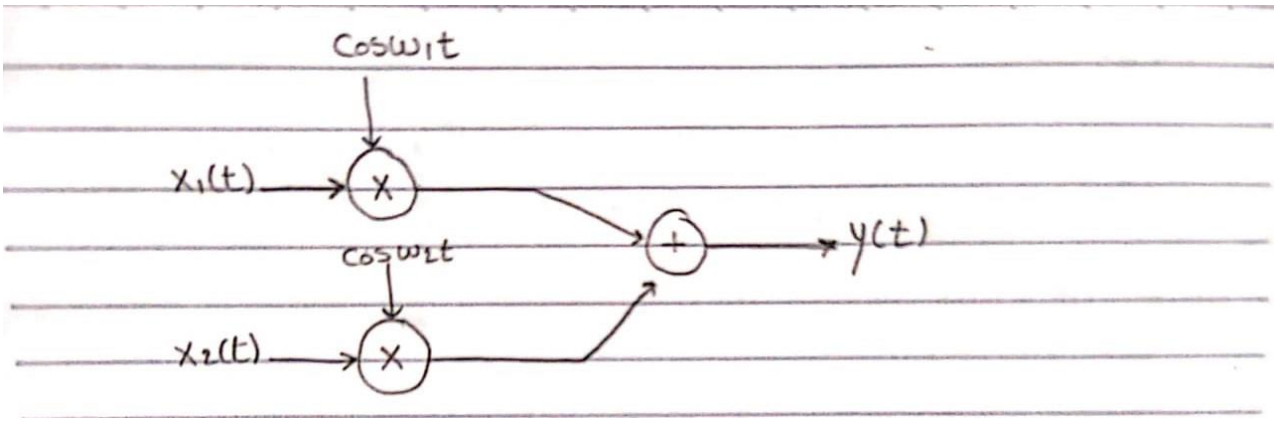

Figure 15: block diagram of transmitter

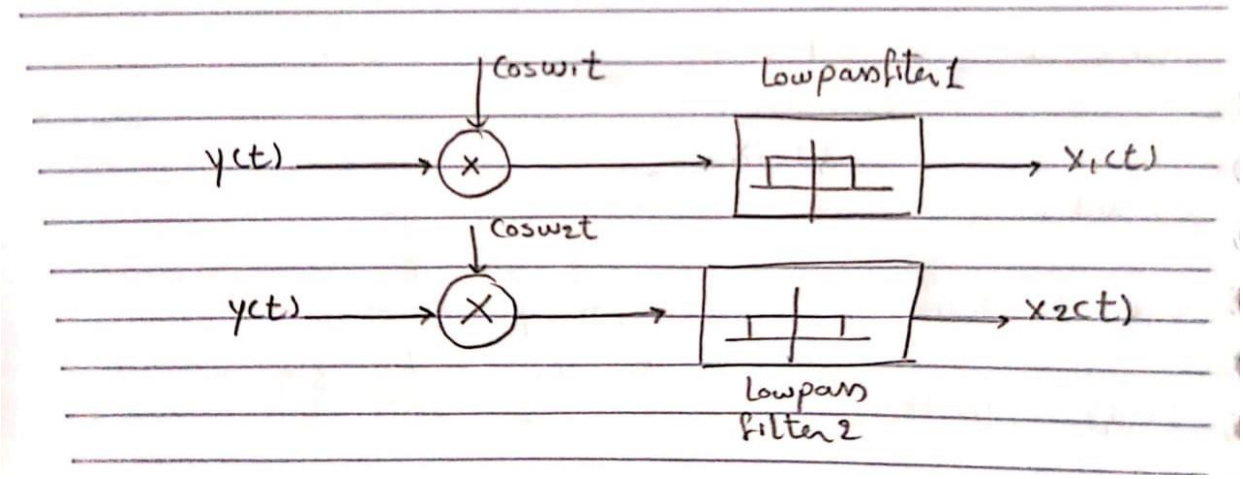

Figure 16: block diagram of receiver

Explanation by equations

reciever :

$m_1(t) = y(t) \times \cos(\omega_1 t)$ → in time

$M_1(\omega) = \frac{1}{2}[Y(\omega+\omega_1) + Y(\omega-\omega_1)]$ → in freq

$m_2(t) = y(t) * \cos(\omega_2 t)$

$M_2(\omega) = \frac{1}{2}[Y(\omega+\omega_2) + Y(\omega-\omega_2)]$

$X_1(\omega) = M_1(\omega)\,H(\omega)$ → lowpassfilter

$x_2(t) = m_1(t) * h_1(t)$

$X_2(\omega) = M_2(\omega)\,H_2(\omega)$

$x_2(t) = m_2(\omega) * h_2(t)$

Scanned with CamScanner

Figure 17: explanation by equations

MATLAB CODES:

I. Image filtering and restoration

A.

```
rgbImage = imread('peppers.png');

% Extract color channels.
redChannel = rgbImage(:,:,1); % Red channel
greenChannel = rgbImage(:,:,2); % Green channel
blueChannel = rgbImage(:,:,3); % Blue channel

% Create an all-black channel.
allBlack = zeros(size(rgbImage, 1), size(rgbImage, 2));

% Create color versions of the individual color channels.
just_red = cat(3, redChannel, allBlack, allBlack);
just_green = cat(3, allBlack, greenChannel, allBlack);
just_blue = cat(3, allBlack, allBlack, blueChannel);
subplot(1, 3, 1); imshow(just_red); title('red');
subplot(1, 3, 2); imshow(just_green); title('green');
subplot(1, 3, 3); imshow(just_blue); title('blue');
```

B.1.

```
rgbImage = imread('peppers.png');
image = rgbImage;

% Extract color channels.
redChannel = rgbImage(:,:,1);
greenChannel = rgbImage(:,:,2);
blueChannel = rgbImage(:,:,3);

% Define an edge enhancement filter.
edges = [0 -1 0; -1 4 -1; 0 -1 0];

% Apply convolution to each channel.
image(:,:,1)= conv2(redChannel, edges,"same");
image(:,:,2)= conv2(greenChannel, edges,"same");
image(:,:,3)= conv2(blueChannel, edges,"same");

% Display the original image and the edge-enhanced image.
subplot(1, 2, 1); imshow(rgbImage); title('Original Image');
subplot(1, 2, 2); imshow(image); title('edged image');
```

B.2.

```
rgbImage = imread('peppers.png');
image = rgbImage;
```

```
% Extract color channels.
redChannel = rgbImage(:,:,1);
greenChannel = rgbImage(:,:,2);
blueChannel = rgbImage(:,:,3);

% Define a sharpening filter.
sharpen = [0 -1 0;-1 5 -1;0 -1 0];

% Apply convolution to each channel.
image(:,:,1)= conv2(redChannel, sharpen,"same");
image(:,:,2)= conv2(greenChannel, sharpen,"same");
image(:,:,3)= conv2(blueChannel, sharpen,"same");

% Display the original image and the edge-enhanced image.
subplot(1, 2, 1); imshow(rgbImage); title('Original Image');
subplot(1, 2, 2); imshow(image); title('sharpened');
```

B.3.

```
rgbImage = imread('peppers.png');
image = rgbImage;
% Extract color channels.
redChannel = rgbImage(:,:,1); % Convert to double for accurate calculations
greenChannel = rgbImage(:,:,2);
blueChannel = rgbImage(:,:,3);

% Define a plurring filter.
plurring = ones(5)/25; % create a matrix with size n with all ones then get the
avg(/n^2)

% Apply convolution to each channel.
image(:,:,1)= conv2(redChannel, plurring,"same");
image(:,:,2)= conv2(greenChannel, plurring,"same");
image(:,:,3)= conv2(blueChannel, plurring,"same");


% Display the plurred image.

imshow(image,[]); title('plurred');
```

B.4.

```
rgbImage = imread('peppers.png');

% Extract color channels.
redChannel = rgbImage(:,:,1); % Red channel
greenChannel = rgbImage(:,:,2); % Green channel
blueChannel = rgbImage(:,:,3); % Blue channel

%creating horizontal blurring filter
motion = zeros(25);
motion (12,1:25)=1;
r= conv2(motion,redChannel);
```

```
g=conv2(motion,greenChannel);
b=conv2(motion,blueChannel);
%crop the image
rr=r(25:size(rgbImage, 1), 25:size(rgbImage, 2));
gg=g(25:size(rgbImage, 1), 25:size(rgbImage, 2));
bb=b(25:size(rgbImage, 1), 25:size(rgbImage, 2));
%concatenate
blurred=cat(3,rr,gg,bb);
bnormalised= mat2gray(blurred); %normalize the image
imshow(bnormalised);
```

C.

```
% Read the image

rgbImage = imread('peppers.png');

% Extract color channels
redChannel = rgbImage(:,:,1);
greenChannel = rgbImage(:,:,2);
blueChannel = rgbImage(:,:,3);


% Create a motion matrix
motion = zeros(size(rgbImage,1),size(rgbImage,2));
motion(12,1:25) = 1;

% Perform FFT
ftmotion = fft2(motion);
ftimager = fft2(redChannel);
ftimageg = fft2(greenChannel);
ftimageb = fft2(blueChannel);

% Multiply the FFTs
outr = (ifft2(ftmotion .* ftimager));
outg = (ifft2(ftmotion .* ftimageg));
outb = (ifft2(ftmotion .* ftimageb));

%to crop the image
rr=outr(25:size(rgbImage, 1), 25:size(rgbImage, 2));
gg=outg(25:size(rgbImage, 1), 25:size(rgbImage, 2));
bb=outb(25:size(rgbImage, 1), 25:size(rgbImage, 2));

% Combine color channels
outtot = cat(3, rr, gg, bb);
moved =mat2gray(outtot);


% adding eps to avoid dividing by zero
outri = (ifft2(fft2(outr) ./(ftmotion+eps)));
outgi = (ifft2(fft2(outg) ./(ftmotion+eps)));
outbi = (ifft2(fft2(outb) ./(ftmotion+eps)));

% Combine color channels
```

```
outtoti = cat(3, outri, outgi, outbi);
recombined = mat2gray(outtoti);

% Display the original image and the result
subplot(1, 3, 1), imshow(rgbImage);  title('Original');
subplot(1, 3, 2), imshow(real(moved),[]);  title('motional blurred');
subplot(1, 3, 3), imshow (real(recombined),[]);  title('recombined');
```

II. Communication system simulation

a.

```
%record first audio
fs=40000;
M = audiorecorder(fs, 16, 1);
disp("Start speaking");
recordblocking(M, 10);
disp("End of recording");
play(M);
audioData = getaudiodata(M);
audiowrite('input1.wav', audioData,fs);
%record second audio
B = audiorecorder(40000, 16, 1);
disp("Start speaking");
recordblocking(B, 10);
disp("End of recording");
play(B);
audioData = getaudiodata(B);
audiowrite('input2.wav', audioData,40000);
```

---

c.

```
%plot the signal1 before and after applying the filter
fs=40000;
[x1,fs1]=audioread('input1.wav');
 y1=filter(low1,x1);

N1=length(x1);
 X1=fft(x1,N1);
 f1=(-N1/2:(N1/2)-1)*fs1/N1;
subplot(2,1,1); plot(f1,abs(fftshift(X1))/N1);

N2=length(y1);
 X2=fft(y1,N1);
 f2=(-N2/2:(N2/2)-1)*fs1/N2;
subplot(2,1,2); plot(f2,abs(fftshift(X2))/N2);


%plot the signal2 before and after applying the filter
fs=40000;
[x3,fs3]=audioread('input2.wav');
 y2=filter(low1,x3);

N3=length(x3);
 X3=fft(x3,N3);
 f3=(-N3/2:(N3/2)-1)*fs3/N3;
```

```
subplot(2,1,1); plot(f3,abs(fftshift(X3))/N3);

N4=length(y2);
 X4=fft(y2,N4);
 f4=(-N4/2:(N4/2)-1)*fs3/N4;
subplot(2,1,2); plot(f2,abs(fftshift(X4))/N4);
```

---

d.

```
%Amplitude modulation
fs = 40000;
[y1, fs1] = audioread('input1.wav');
x1= filter(low1,y1);
N1=length(x1);
 X1=fft(x1,N1);
 f1=(-N1/2:(N1/2)-1)*fs1/N1;
subplot(3,1,1); plot(f1,abs(fftshift(X1))/N1); title('Signal 1');


[y2, fs2] = audioread('input2.wav');
x2= filter(low2,y2);
N2=length(x2);
X2=fft(x2,N2);
 f2=(-N2/2:(N2/2)-1)*fs2/N2;
subplot(3,1,2); plot(f2,abs(fftshift(X2))/N2); title('Signal 2');


Fcarrier1 =5000;
Fcarrier2 = 15000;
% Normalize the signals to ensure they have the same amplitude
x1_normalized = x1 / max(abs(x1));
x2_normalized = x2 / max(abs(x2));
t1 = 0:1/fs1:(length(x1_normalized)-1)/fs1;
t2 = 0:1/fs2:(length(x2_normalized)-1)/fs2;
carrier_wave1 = cos(2*pi*Fcarrier1*t1(1:length(x1_normalized)));
carrier_wave2 = cos(2*pi*Fcarrier2*t2(1:length(x2_normalized)));
modulated_signal1 = x1_normalized .* carrier_wave1';
modulated_signal2 = x2_normalized .* carrier_wave2';
sum_modulated = modulated_signal1 + modulated_signal2;

% Perform the Fourier transform on the sum of modulated signals
fft_sum_modulated = fftshift(fft(sum_modulated));

 %Frequency axis for the FFT
N = length(sum_modulated);
f = (-fs1/2):(fs1/N):(fs1/2-fs1/N);

 %Plot the frequency response
subplot(3,1,3); plot(f, abs(fft_sum_modulated));
title('Amplitude-Modulated Signal');
```

---

e.

```
%demodulation
demodulated_signal1 = 2*max(abs(x1))* sum_modulated.* carrier_wave1';
demodulated_signal2 = 2*max(abs(x2))*sum_modulated.* carrier_wave2';
```

```
out1_filtered = filter(low1,demodulated_signal1);
audiowrite('output1.wav', out1_filtered, fs1);
out2_filtered = filter(low2, demodulated_signal2);
audiowrite('output2.wav', out2_filtered, fs2);
disp('Playing demodulated signals....');
m1_demodulatedandfilt = audioplayer(out1_filtered, fs1);
play(m1_demodulatedandfilt);
pause(10);
m2_demodulatedandfilt = audioplayer(out2_filtered, fs2);
play(m2_demodulatedandfilt);

N2=length(out1_filtered);
X2=fft(out1_filtered,N2);

f2=(-N2/2:(N2/2)-1)*fs2/N2;

N3=length(out2_filtered);
 X3=fft(out2_filtered,N3);

 f3=(-N2/2:(N2/2)-1)*fs2/N2;
 figure;
subplot(2,1,1); plot(f2,abs(fftshift(X2)/N2)); title('Signal1 demodulated');
subplot(2,1,2); plot(f3,abs(fftshift(X3)/N3)); title('Signal2 demodulated');
```

References:

- How do I split a color image into its 3 RGB channels? - MATLAB Answers - MATLAB Central (mathworks.com)
- https://ch.mathworks.com/matlabcentral/answers/436566-edge-detection-in-matrix-form
- https://ch.mathworks.com/matlabcentral/answers/1921595-how-to-sharpen-an-image-sharpening-filter-kernel
- https://ch.mathworks.com/matlabcentral/answers/450356-how-can-blur-an-image
- https://youtu.be/8Oi7bfE3Cvg?si=SMu-DiIgLUEB2Usw
- https://youtu.be/dEG_hv5E8VQ?si=XINEx9gTsHcLvD18
- https://youtu.be/aJI_icFbs-0?si=Jq8ONkPQlRTQt1EP
- https://www.mathworks.com/help/signal/ref/modulate.html