Cairo University

Faculty of Engineering

Electronics and Electrical Communications Engineering Department

Third Year

Analog Communications

Term Project

MATLAB implementation of a superheterodyne receiver

Student Name: شادي علاء الدين بدير

Sec: ……02…… BN: ……24…….

Student Name: علي محمود علي موسى

Sec: ……02…… BN: ……51…….

Contents

Table of figures

1. The transmitter

This part contains the following tasks

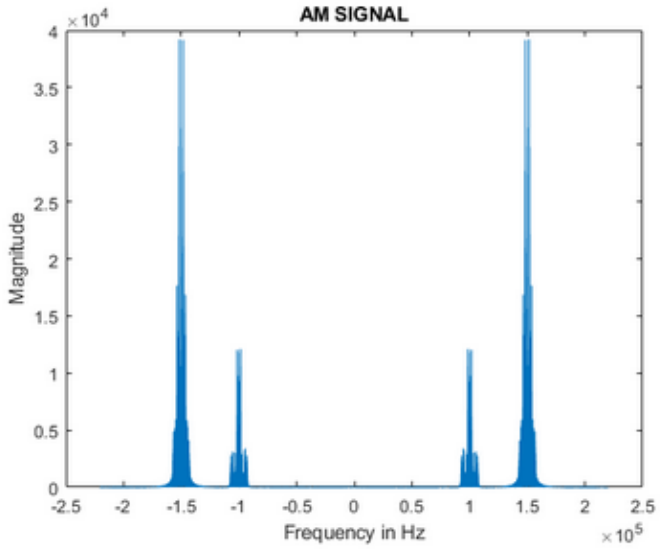
1. Reading monophonic audio signals into MATLAB.
2. Upsampling the audio signals.
3. Modulating the audio signals (each on a separate carrier).
4. Addition of the modulated signals.

Discussion

Generally, a stereo audio signals means that it has two channels a channel for the right speakers and a channel for the left speakers, the first step after reading the stereo is converting it to monophonic this is done by adding both channel into a 1 channel signal. Since our carrier frequency is higher than the sampling frequency, we needed to up-sample the messages. Then each message is then multiplied by a different carrier with dF = 50KHz, then we did FDM by adding both messages into 1 signal.

The figures

Figure 1: The spectrum of the output of the transmitter



2. The RF stage

This part addresses the RF filter and the mixer following it.

Discussion

The received message is a FDM message, it has many channels on which there is unwanted messages that aren't our scope, this stage is simply a filter that selects the desired channel only, rejects all other channels especially the channel that is centered at (desired channel frequency + 2*IF frequency) because this channel will be imaged in IF stage to stand on top of our channel, so this stage is simply IMAGE and INTERFERENCE REJECTION.

The figures

Assume we want to demodulate the first signal (at ω_o).

Figure 2: the output of the RF filter (before the mixer)

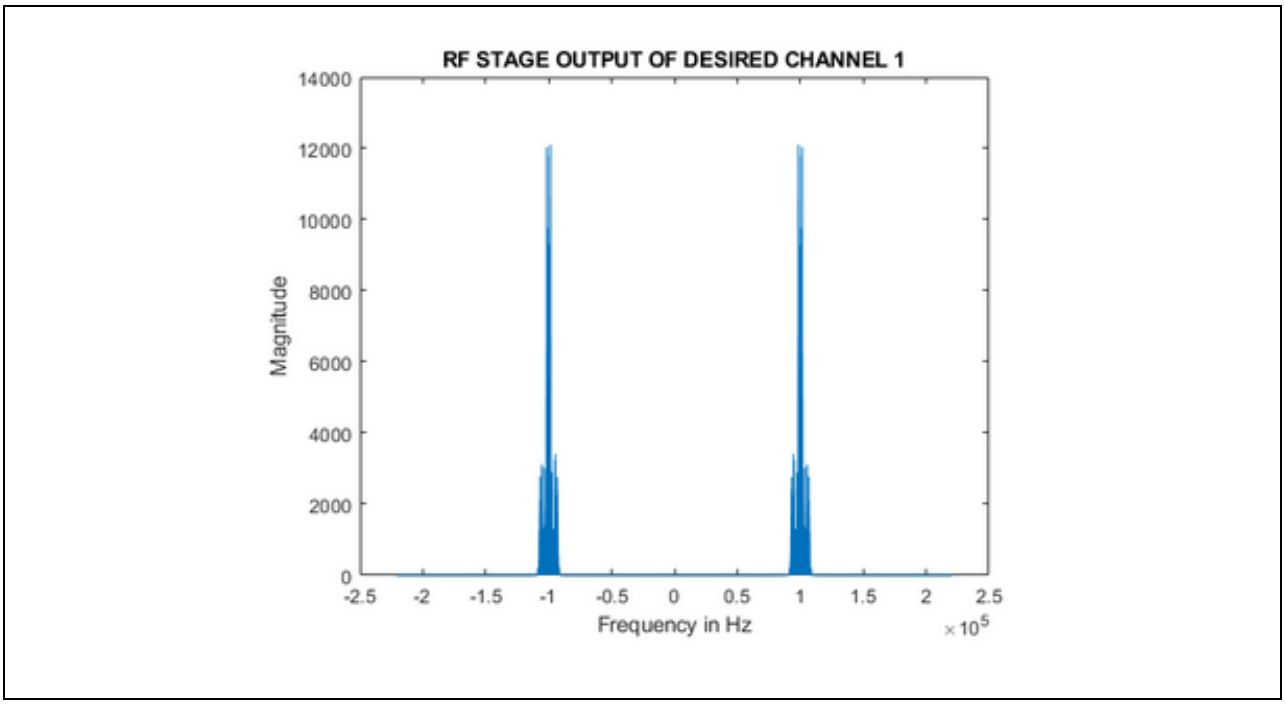
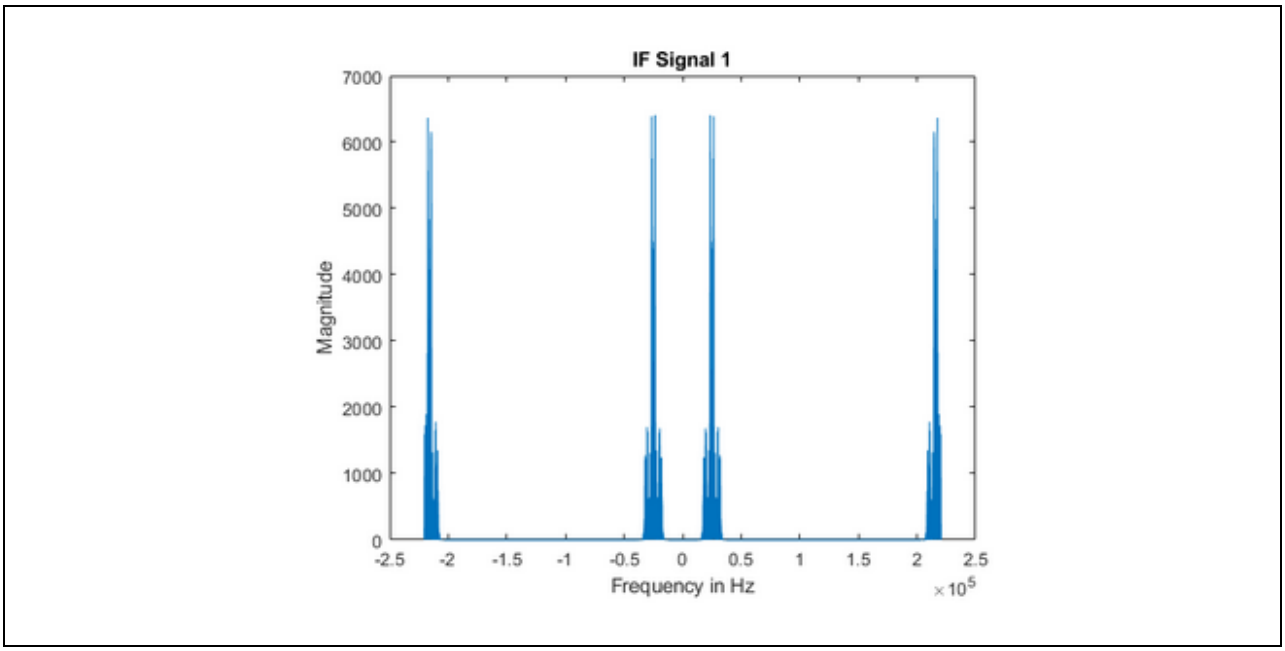


Figure 3: The output of the mixer



3. ***The IF stage***

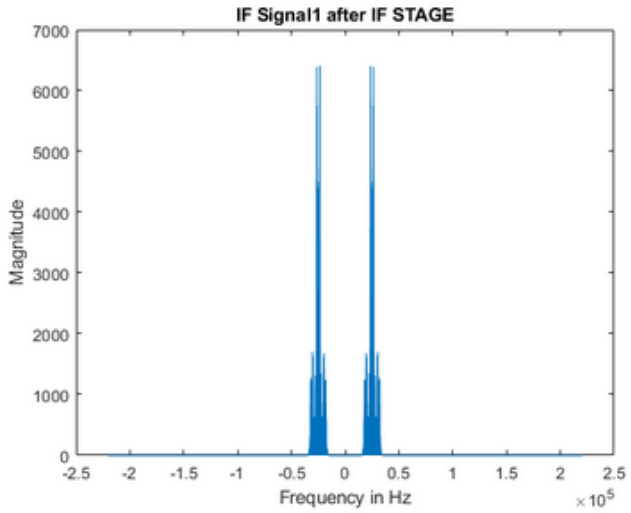
This part addresses the IF filter.

Discussion

After mixing our RF signal with the LO carrier we know for sure that the message at w0 will have components at the following frequencies (wIF, 2w0+wIF), thus a IF filter will do a good job in rejecting the higher frequency components and only focuses on the components at the IF frequency.

The figures

Figure 4: Output of the IF filter



4. *The baseband demodulator*

This part addresses the coherent detector used to demodulate the signal from the IF stage.

Discussion

```
Now we have the message at the IF frequency, to detect the message we need to
down convert it into baseband frequency, this is done by centering it at F =
0Hz, this is done by multiplying the IF signal by a carrier with w = wIF, the
output will have components at the baseband and components at other
frequencies, a simple LPF will reject other frequencies and selects the
Baseband signal.
```

The figures

Figure 5: Output of the mixer (before the LPF)

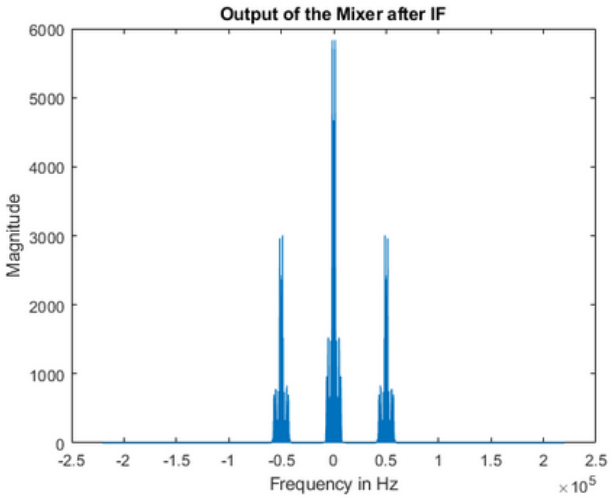
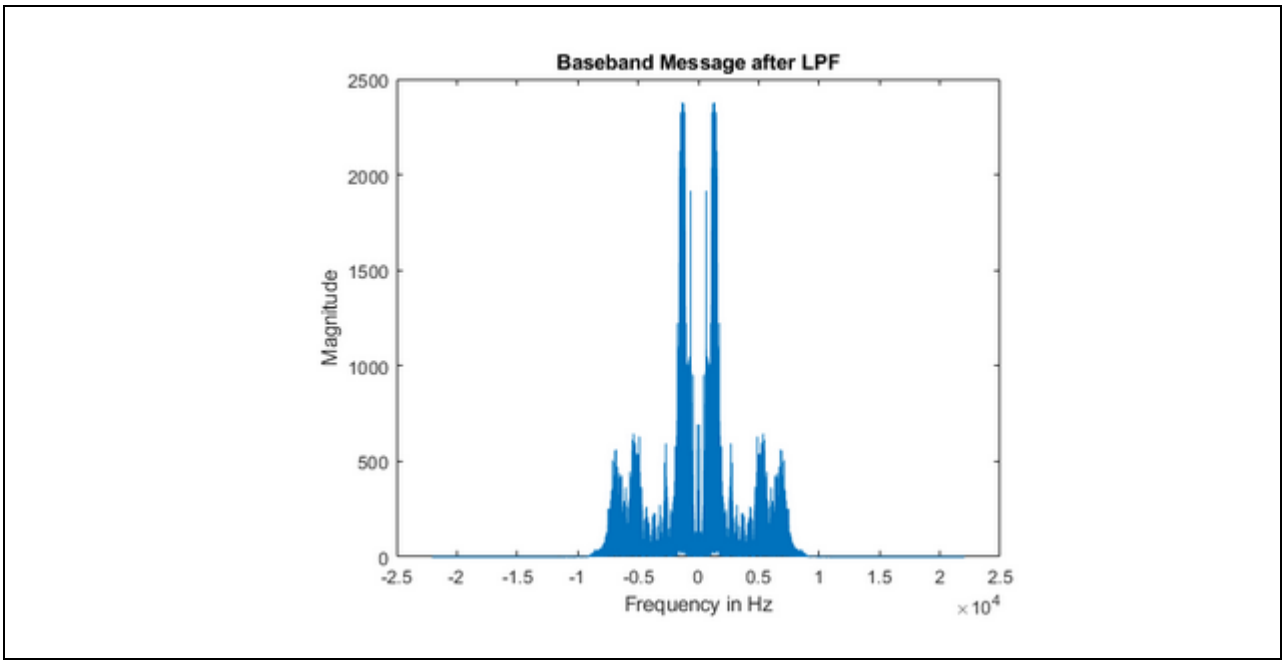


Figure 6: Output of the LPF



5. *Performance evaluation without the RF stage*

The figures

Figure 7: output of the RF mixer (no RF filter)

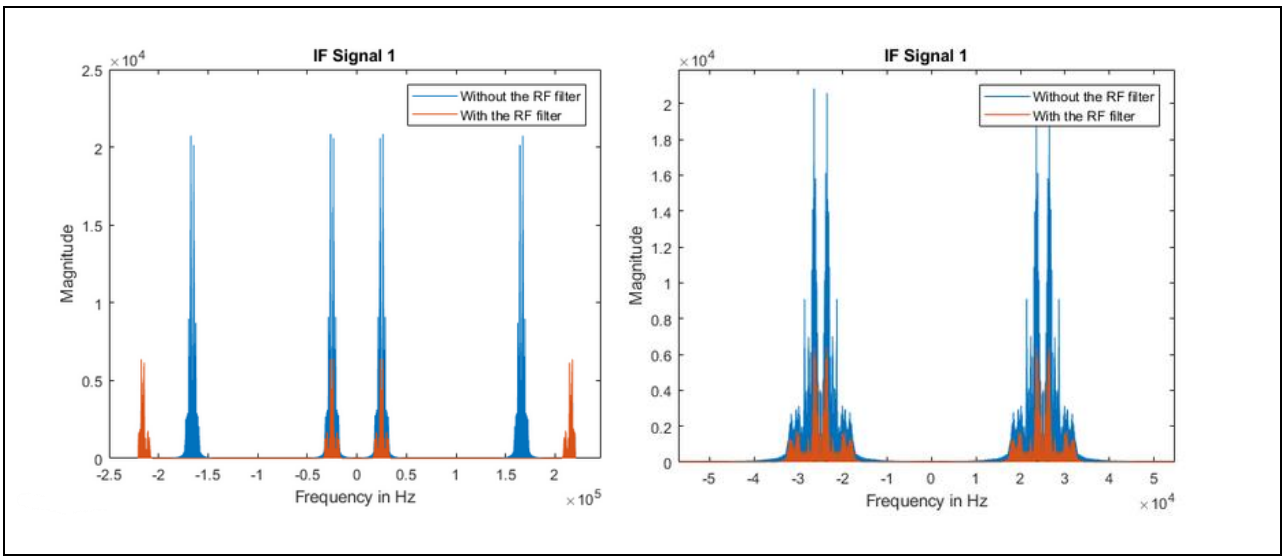


Figure 8: Output of the IF filter (no RF filter)

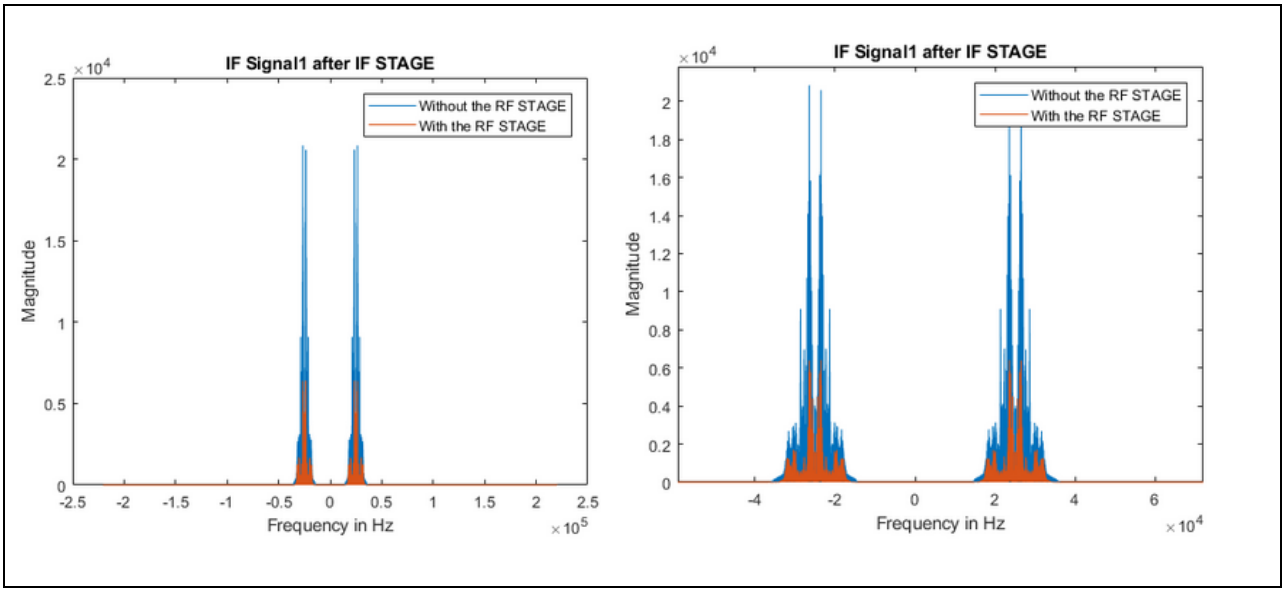


Figure 9: Output of the IF mixer before the LPF (no RF filter)

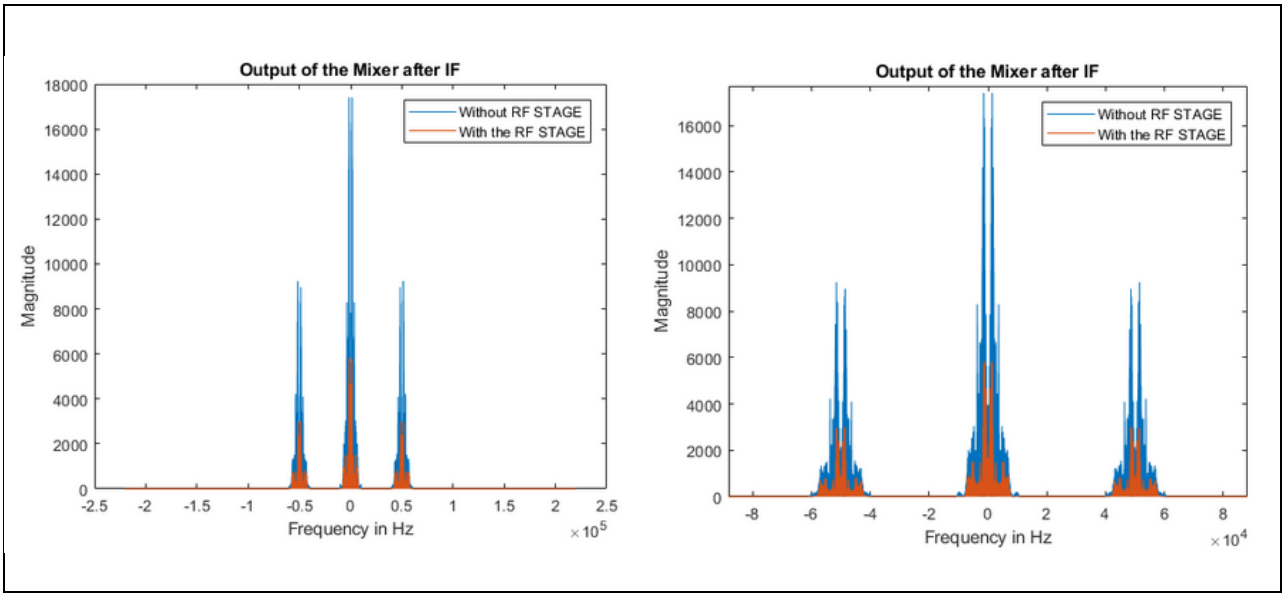
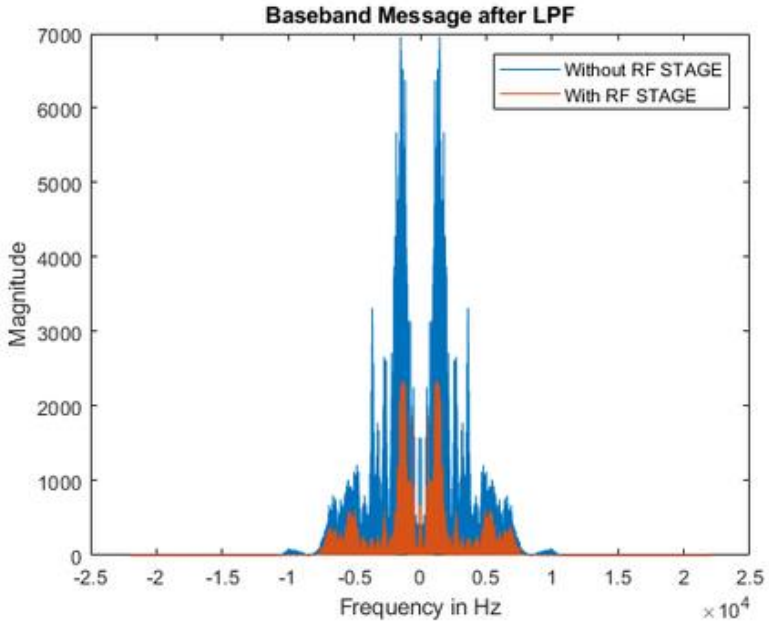


Figure 10: Ouptut of the LPF (no RF filter)



6. Comment on the output sound

With the existence of the RF STAGE, managed to completely reject interferers and images, thus the output sound was identical to the message that was carried on channel w0. While in the absence of the RF STAGE, the message that was on channel w0 is hugely interfered by the image, the image was itself the message on the 2nd channel (f0+50K) = (f0 + 2fIF). The sound was an overlap of the two messages on each other as expected.

What happens (in terms of spectrum and the sound quality) if the receiver oscillator has frequency offset by 0.1 KHz and 1 KHz

In terms of the spectrum there was a little bit of some frequency components missing in 0.1K offset, and the frequency components missing in 1K offset became huge and the spectrum was greatly distorted.in terms of sound quality as offset increased the sound was greatly distorted and at 1K offset the message was nearly unrecognizable.

7. The code

```
%.....................................Pre-Processing Stage.............................%

loc = 'FilePath'; % defining the location of the files %

[Signal1,Fs1] = audioread(fullfile(loc,'Short_BBCArabic2.wav')); % reading the first signal %
[Signal2,Fs2] = audioread(fullfile(loc,'Short_FM9090.wav')); % reading the second signal %

L1 = length(Signal1); % L1 is length of first signal %
L2 = length(Signal2); % L2 is length of second signal %

if L1 > L2
```

```
L = L1;
else
L = L2; % L is the maximum length %
end

Signal1 = padarray(Signal1,L-L1,0,'post'); % making both signals have the same Maximum length %
Signal2 = padarray(Signal2,L-L2,0,'post');

left1 = Signal1(:,1);
right1 = Signal1(:,2);
left2 = Signal2(:,1);
right2 = Signal2(:,2);

Signal1 = left1 + right1; % making the first signal One channel only %
Signal2 = left2 + right2; % making the second signal One channel only %

Signal1 = interp(Signal1,10); % increasing the number of samples of the first signal to follow
Nyquist %
Signal2 = interp(Signal2,10); % increasing the number of samples of the second signal to follow
Nyquist %

% changing the Sampling rate accordingly %
Fs1 = 10*Fs1;
Fs2 = 10*Fs2;

Ts1 = 1/Fs1;
Ts2 = 1/Fs2;

L=length(Signal1);

f1=linspace(-Fs1/2,Fs1/2,L); % defining the x axis %

%.................................Modulation Stage (Transmitter).............................%

t1 = 0:Ts1:(L-1)*Ts1
carrier1 = cos(2*pi*100*1000*t1); % the first modulating signal %
carrier1_transpose=transpose(carrier1); % making the carrier column instead of row %
clear carrier1 % clearing the carrier for memory effeciency %

t2 = 0:Ts2:(L-1)*Ts2
carrier2 = cos(2*pi*150*1000*t2); % the second modulating signal %
carrier2_transpose=transpose(carrier2); % making the carrier column instead of row %
clear carrier2 % clearing the carrier for memory effeciency %

Signal1 = Signal1.*carrier1_transpose; % Modulation of the first message %
Signal2 = Signal2.*carrier2_transpose; % Modulation of the second message %

AM = Signal1 + Signal2; % Multiplexing the two modulated messages %

figure
plot(f1,fftshift(abs(fft(AM))));
xlabel('Frequency in Hz')
ylabel('Magnitude')
title('AM SIGNAL')
```

```
%.............................Bandpass Filter Design of the RF STAGE.....................%

BP1 = fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2',80000/(Fs1/2), 90000/(Fs1/2), 110000/(Fs1/2),
120000/(Fs1/2), 70, 1, 70);
BP2 = fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2',120000/(Fs1/2), 130000/(Fs1/2),
170000/(Fs1/2), 180000/(Fs1/2), 70, 1, 70);

Filter1 = design(BP1,'butter'); % designing a butterworth BandPass filter with attenuation of 70dB to
choose Signal 1 %
Filter2 = design(BP2,'butter'); % designing a butterworth BandPass filter with attenuation of 70dB to
choose Signal 2 %

Filtered1 = filter(Filter1,AM); % Applying the filter on the AM Signal, Filtered1 should be the
Modulated message 1 %
Filtered2 = filter(Filter2,AM); % Applying the filter on the AM Signal, Filtered2 should be the
Modulated message 2 %

figure
plot(f1,fftshift(abs(fft(Filtered1))));
xlabel('Frequency in Hz')
ylabel('Magnitude')
title('RF STAGE OUTPUT OF DESIRED CHANNEL 1')

figure
plot(f1,fftshift(abs(fft(Filtered2))));
xlabel('Frequency in Hz')
ylabel('Magnitude')
title('RF STAGE OUTPUT OF DESIRED CHANNEL 2')

%.......................................... IF STAGE.....................................%

carrierlo1 = cos(2*pi*125000*t1); % the local oscillator carrier for signal one with wLO = wCarrier +
wIF, wIF = 25kHz*2pi %
carrierlo1_transpose = transpose(carrierlo1); % making the carrier a column instead of a row to match
the signal %
clear carrierlo1
carrierlo2 = cos(2*pi*175000*t2); % the local oscillator carrier for signal two with wLO = wCarrier +
wIF, wIF = 25kHz*2pi %
carrierlo2_transpose = transpose(carrierlo2); % making the carrier a column instead of a row to match
the signal %
clear carrierlo2

AMIF1 = Filtered1.*carrierlo1_transpose; % Mixing the output of the RFStage1 with the LO carrier to
produce IF message %
AMIF2 = Filtered2.*carrierlo2_transpose; % Mixing the output of the RFStage2 with the LO carrier to
produce IF message %

figure
plot(f1,fftshift(abs(fft(AMIF1))));
xlabel('Frequency in Hz')
ylabel('Magnitude')
title('IF Signal 1')

figure
plot(f1,fftshift(abs(fft(AMIF2))));
```

```
xlabel('Frequency in Hz')
ylabel('Magnitude')
title('IF Signal 2')

%.............................Bandpass Filter Design of the IF STAGE.....................%

BP11 = fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2',10000/(Fs1/2), 15000/(Fs1/2), 35000/(Fs1/2),
40000/(Fs1/2), 70, 1, 70);
BP22 = fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2',100/(Fs1/2), 7000/(Fs1/2), 43000/(Fs1/2),
50000/(Fs1/2), 70, 1, 70);

Filter11 = design(BP11,'butter'); % designing a butterworth BandPass filter with attenuation of 70dB
to choose AM IF1 %
Filter22 = design(BP22,'butter'); % designing a butterworth BandPass filter with attenuation of 70dB
to choose AM IF2 %

Filtered11 = filter(Filter11,AMIF1); % Applying the Bandpass filter on the first IF %
Filtered22 = filter(Filter22,AMIF2); % Applying the Bandpass filter on the second IF %

figure
plot(f1,fftshift(abs(fft(Filtered11))));
xlabel('Frequency in Hz')
ylabel('Magnitude')
title('IF Signal1 after IF STAGE')

figure
plot(f1,fftshift(abs(fft(Filtered22))));
xlabel('Frequency in Hz')
ylabel('Magnitude')
title('IF Signal2 after IF STAGE')

%.............................Base Band Detection Stage .............................%

carrierBB1 = cos(2*pi*25000*t1); % creating the Baseband returning carrier of frequency = wIF %
carrierBB1_transpose = transpose(carrierBB1); % making the carrier a column instead of a row to match
the signal %
clear carrierBB1 % clearing the row carrier for memory effeciency %

carrierBB2 = cos(2*pi*25000*t2); % creating the Baseband returning carrier of frequency = wIF %
carrierBB2_transpose = transpose(carrierBB2); % making the carrier a column instead of a row to match
the signal %
clear carrierBB2 % clearing the row carrier for memory effeciency %

AMBB1 = Filtered11.*carrierBB1_transpose; % Mixing %
AMBB2 = Filtered22.*carrierBB2_transpose; % Mixing %

figure
plot(f1,fftshift(abs(fft(AMBB1))));
xlabel('Frequency in Hz')
ylabel('Magnitude')
title('Output of the Mixer after IF')

figure
plot(f1,fftshift(abs(fft(AMBB2))));
xlabel('Frequency in Hz')
```

```
ylabel('Magnitude')
title('Output of the Mixer after IF')

%.................................Low Pass Filter Design Stage.................................%

LPF1 = fdesign.lowpass('Fp,Fst,Ap,Ast',10000/(Fs1/2),15000/(Fs1/2),1,100);
LPF2 = fdesign.lowpass('Fp,Fst,Ap,Ast',15000/(Fs1/2),20000/(Fs1/2),1,100);

LPFilter1 = design(LPF1,'butter'); % Designing the LOW PASS FILTER that chooses the Baseband 1 Signal
%
LPFilter2 = design(LPF2,'butter'); % Designing the LOW PASS FILTER that chooses the Baseband 2 Signal
%

RAWSignal1 = filter(LPFilter1,AMBB1); % Applying the LPF1 %
RAWSignal2 = filter(LPFilter2,AMBB2); % Applying the LPF2 %

%.................................Post-Processing Stage.....................................%

Message1 = 4*downsample(RAWSignal1,10); % down sampling the Signal1 to remove the effect of
interpolation %
Message2 = 4*downsample(RAWSignal2,10); % down sampling the Signal2 to remove the effect of
interpolation %

Fs1 = Fs1/10;
L = length(Message1);
f2 = linspace(-Fs1/2, Fs1/2 ,L);

figure
plot(f2,fftshift(abs(fft(Message1))));
xlabel('Frequency in Hz')
ylabel('Magnitude')
title('Baseband Message1 after LPF')

figure
plot(f2,fftshift(abs(fft(Message1))));
xlabel('Frequency in Hz')
ylabel('Magnitude')
title('Baseband Message2 after LPF')

%sound(Message1,Fs1)%
%sound(Message2,Fs1)%
```