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: محمد ناصر محمد كمال**

**Sec: 4 BN: 3 ID: 9203349**

**Student Name: هنا إيهاب خاطر يونس**

**Sec: 4 BN: 42 ID: 9203687**

Contents

[1. The transmitter 3](#_Toc122733947)

[Discussion 3](#_Toc122733948)

[The figures 3](#_Toc122733949)

[2. The RF stage 3](#_Toc122733950)

[Discussion 3](#_Toc122733951)

[The figures 4](#_Toc122733952)

[3. The IF stage 5](#_Toc122733953)

[Discussion 5](#_Toc122733954)

[The figures 5](#_Toc122733955)

[4. The baseband demodulator 5](#_Toc122733956)

[Discussion 5](#_Toc122733957)

[The figures 6](#_Toc122733958)

[5. Performance evaluation without the RF stage 7](#_Toc122733959)

[The figures 7](#_Toc122733960)

[6. Comment on the output sound 9](#_Toc122733961)

[7. The code 10](#_Toc122733962)

**Table of figures**

[Figure 1: The spectrum of the output of the transmitter 3](#_Toc122733982)

[Figure 2: the output of the RF filter (before the mixer) 4](#_Toc122733983)

[Figure 3: The output of the mixer 4](#_Toc122733984)

[Figure 4: Output of the IF filter 5](#_Toc122733985)

[Figure 5: Output of the mixer (before the LPF) 6](#_Toc122733986)

[Figure 6: Output of the LPF 6](#_Toc122733987)

[Figure 7: output of the RF mixer (no RF filter) 7](#_Toc122733988)

[Figure 8: Output of the IF filter (no RF filter) 7](#_Toc122733989)

[Figure 9: Output of the IF mixer before the LPF (no RF filter) 8](#_Toc122733990)

[Figure 10: Output of the LPF (no RF filter) 8](#_Toc122733991)

# 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

|  |
| --- |
| First we read 3 audio signals & added their 2 stereo columns to transform them into monophonic signals as we implement monophonic receiver, then we modified their sizes to fit the longest one using padding, after that we  up-sampled the signals to fit the highest sampling rate of their highest carrier (fs>=2\*200 KHz), then we generated 3 carriers, multiplied by 3 signals & adding the 3 results together to construct FDM signal. |

## The figures

Chart, histogram

Description automatically generatedFigure : The spectrum of the output of the transmitter

# The RF stage

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

## Discussion

|  |
| --- |
| As we intend to move signals to intermediate frequency IF, we may suffer the problem of IF image so we use bandpass filter to select the intended received signal & reject the others then multiply the filtered signal by a carrier with frequency = fc + fIF to move the signal to IF. |

## The figures

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

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

Chart

Description automatically generated

A picture containing chart

Description automatically generatedFigure : The output of the mixer

# The IF stage

This part addresses the IF filter.

## Discussion

|  |
| --- |
| After IF mixer stage signal is carried to IF as intended but it also has carried to higher frequency band, so we need a bandpass filter to select the IF signal & reject its higher frequency version, and the importance of carrying the received signal on IF before baseband to get rid of leakage & flicker noise & to improve the filters’ selectivity. |

## The figures

A picture containing chart

Description automatically generatedFigure : Output of the IF filter

# The baseband demodulator

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

## Discussion

|  |
| --- |
| The previous modulated signal needs to be demodulated to return the signal to the baseband, we will multiply it by a carrier with frequency = fIF to perform coherent demodulation then use a lowpass filter to select the baseband signal & reject higher frequencies then multiplied by a gain = 4.5 (due to carriers the signal’s amplitude was reduced 4 times), now if we use the sound command we can successfully listen to the original audio message. |

## The figures

Chart

Description automatically generatedFigure : Output of the mixer (before the LPF)

Chart, histogram

Description automatically generatedFigure : Output of the LPF

# Performance evaluation without the RF stage

## The figures

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

Chart, histogram

Description automatically generated

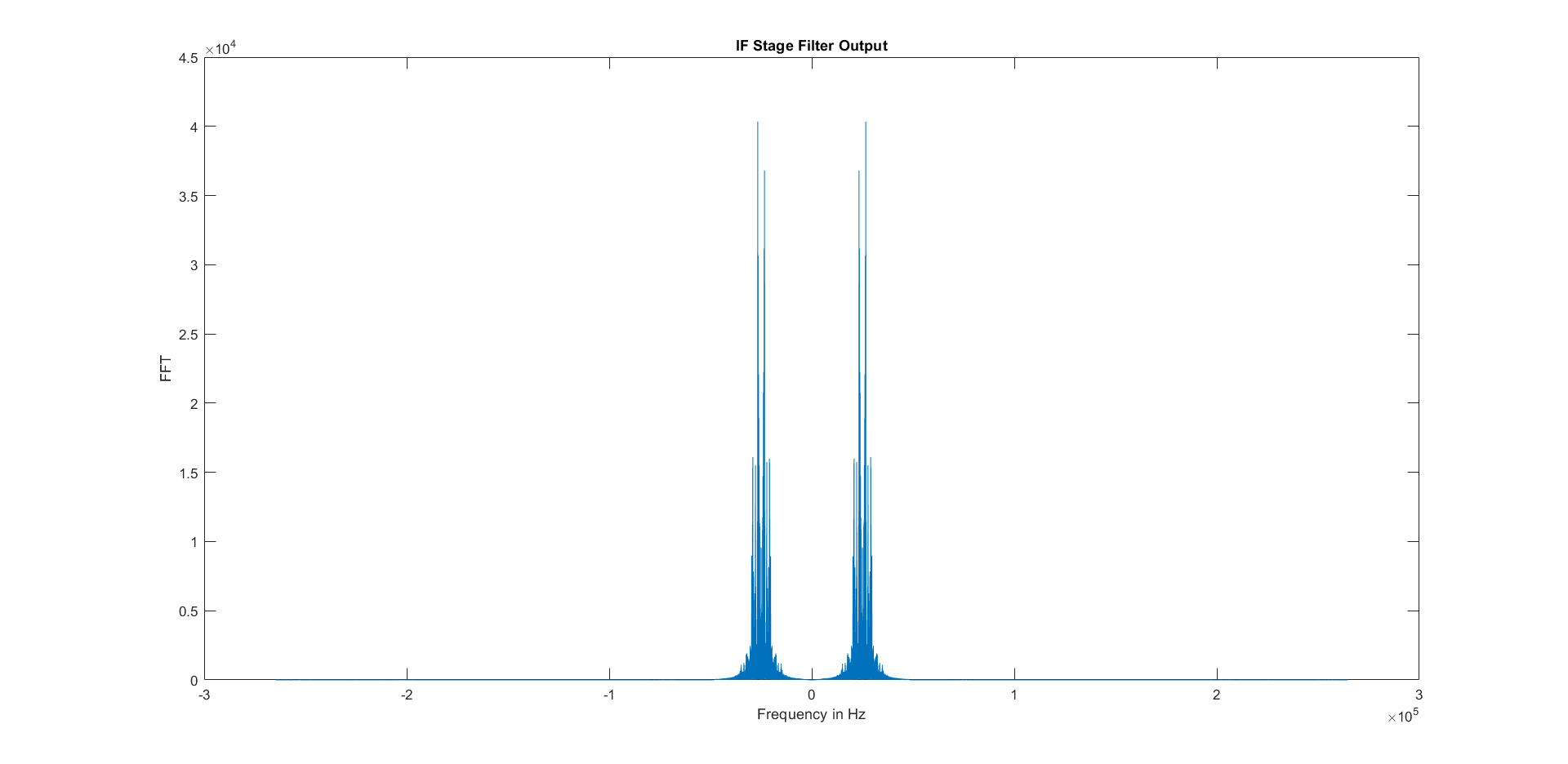
Figure : Output of the IF filter (no RF filter)

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

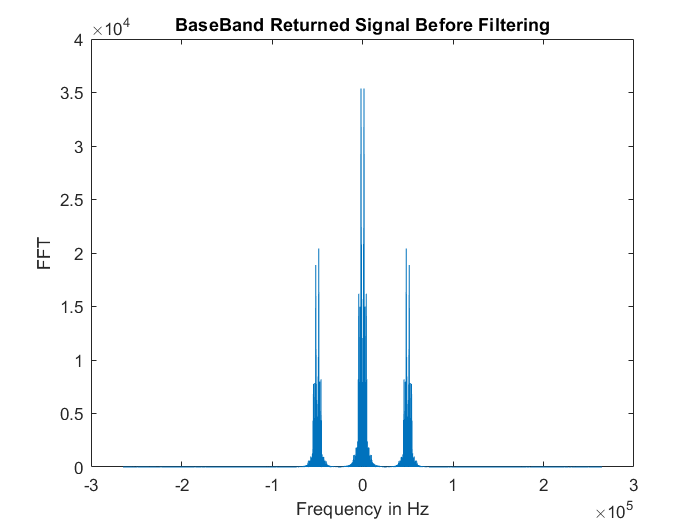
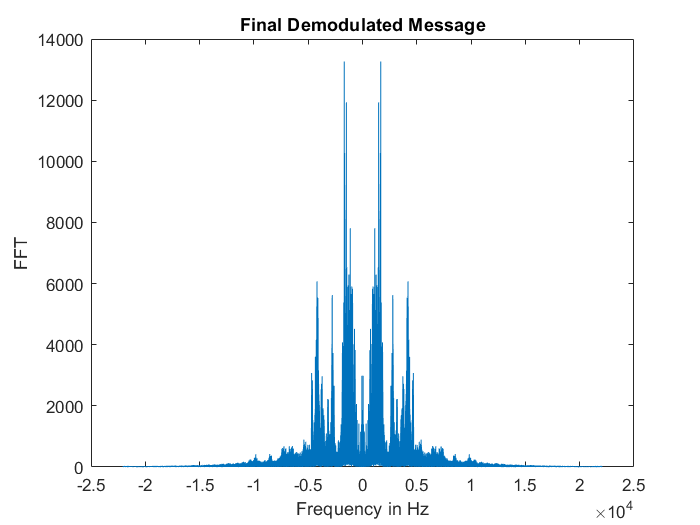


Figure : Output of the LPF (no RF filter)



# Comment on the output sound

|  |
| --- |
| In the existence of RF stage we chose the signal with fc = 100 KHz (Sky News Message) we heard at the receiver this message clearly without any difference between it and the original one, but after removing the RF stage an interference occurred between the required signal and the signal with  fc = 100 + 2\*fIF KHz (150 KHz) it was in our program Quran message which causes IF image to interfere with the required signal and we heard both signals at the receiver’s output. |

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

1. Offset = 0.1 KHz: the sound was a little bit distorted; it is different than the original sound but still recognizable & understood.

2. Offset = 1 KHz: the sound was totally distorted; it is different than the original and neither recognizable nor understood.

This is due to that the signal has not been carried to the fIF exactly but it is shifted so the coherent demodulator with f = 25 KHz has not done its job correctly.

# The code

|  |
| --- |
| %% The Transmitter  % Read Audio Signals  [sky\_news\_msg,sampling\_rate] = audioread('Short\_SkyNewsArabia.wav');  [quran\_msg,sampling\_rate] = audioread('Short\_QuranPalestine.wav');  [russian\_msg,sampling\_rate] = audioread('Short\_RussianVoice.wav');    % Making Signals Monophonic  sky\_news\_msg = sky\_news\_msg(:,1) + sky\_news\_msg(:,2);  quran\_msg = quran\_msg(:,1) + quran\_msg(:,2);  russian\_msg = russian\_msg(:,1) + russian\_msg(:,2);    % Padding Short Signals With Size Of Longest Signal  s1 = size(sky\_news\_msg);  s1 = s1(1);  s2 = size(quran\_msg);  s2 = s2(1);  s3 = size(russian\_msg);  s3 = s3(1);  N = max([s1 s2 s3]);  sky\_news\_msg = [sky\_news\_msg; zeros(N-s1,1)];  quran\_msg = [quran\_msg; zeros(N-s2,1)];  russian\_msg = [russian\_msg; zeros(N-s3,1)];  clear s1 s2 s3;    % BaseBand Signals Fourier Transform  messages = [sky\_news\_msg, quran\_msg, russian\_msg];  clear sky\_news\_msg quran\_msg russian\_msg;  k = -N/2 :(N/2)-1;  figure('Name','Base-Band Signals','NumberTitle','off');  subplot(1,3,1);  plot(k\*sampling\_rate/N,abs(fftshift(fft(messages(:,1)))));  title('Sky News Message Spectrum');  xlabel('Frequency in Hz');  ylabel('FFT');  subplot(1,3,2);  plot(k\*sampling\_rate/N,abs(fftshift(fft(messages(:,2)))));  title('Quran Message Spectrum');  xlabel('Frequency in Hz');  ylabel('FFT');  subplot(1,3,3);  plot(k\*sampling\_rate/N,abs(fftshift(fft(messages(:,3)))));  title('Russian Voice Message Spectrum');  xlabel('Frequency in Hz');  ylabel('FFT');    % Upsampling The Audio Signals To Fit The Carriers Sampling Rates  messages\_interp = [interp(messages(:,1),12), interp(messages(:,2),12)...  ,interp(messages(:,3),12)];  clear messages;  fs = 12\*sampling\_rate;  num\_samples = 12\*N;    % Generating The Carriers For The Signals  t = 0:1/fs:16.76190476;  sky\_news = cos(2\*pi\*100\*1000\*t)';  quran = cos(2\*pi\*150\*1000\*t)';  russian = cos(2\*pi\*200\*1000\*t)';  carriers = [sky\_news quran russian];  clear sky\_news quran russian ans;    % Perform The Modulation Process For Each Audio Signal  modulated\_msgs = messages\_interp .\* carriers;  clear messages\_interp carriers;    % Create The Frequency Division Multiplexed Signal By Addition Of The Modulated Signals  FDM = modulated\_msgs(:,1) + modulated\_msgs(:,2) + modulated\_msgs(:,3);  clear modulated\_msgs;    % Plot The Frequency Division Multiplexed Spectrum  k2 = -num\_samples/2 : num\_samples/2 - 1;  figure('Name','Frequency Division Multiplexed Signals','NumberTitle','off');  plot(k2\*fs/num\_samples,abs(fftshift(fft(FDM))));  title('Frequency Division Multiplexing Spectrum');  xlabel('Frequency in Hz');  ylabel('FFT');        %% The Receiver  % Choose The Required Audio Signals  disp (" ");  disp ("The Provided Channels: ");  disp ("1. Sky News On Carrier: 100 KHz");  disp ("2. Quran On Carrier: 150 KHz");  disp ("3. Russian Voice On Carrier: 200 KHz");  signal\_req = input ("Please Select The Desired Channel Carrier in KHz: ");  signal\_req = 1000 \* signal\_req;    % The RF Stage  % The BPF BandWidth Will Always BE 44 KHZ To Be Valid For Any RF Voice  F\_stop1 = signal\_req - 24000;  F\_pass1 = signal\_req - 22000;  F\_pass2 = signal\_req + 22000;  F\_stop2 = signal\_req + 24000;  A\_stop1 = 80;  A\_pass = 0.001;  A\_stop2 = 80;  RF\_BandPassSpecs = fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2',...  F\_stop1, F\_pass1, F\_pass2, F\_stop2, A\_stop1, A\_pass,A\_stop2, fs);  RF\_BandPass\_filter = design(RF\_BandPassSpecs, 'equiripple');  RF\_stage\_out = filter(RF\_BandPass\_filter, FDM);    % Plot The RF Stage Filtered Output  figure('Name','RF Stage Filter Output','NumberTitle','off');  plot(k2\*fs/num\_samples,abs(fftshift(fft(RF\_stage\_out))));  title('RF Stage Filter Output');  xlabel('Frequency in Hz');  ylabel('FFT');    % IF Oscillator  IF\_freq = 25000;  IF\_carrier = cos(2\*pi\*t\* (IF\_freq + signal\_req))';  IF\_stage\_input = RF\_stage\_out .\* IF\_carrier;    % Plot The IF Oscilltor Output  figure('Name','RF Mixer Output','NumberTitle','off');  plot(k2\*fs/num\_samples,abs(fftshift(fft(IF\_stage\_input))));  title('RF Mixer Output');  xlabel('Frequency in Hz');  ylabel('FFT');    % The BPF PassBand Will Always BE 44 KHZ To Be Valid For Any IF Voice  F\_stop1 = IF\_freq - 24000;  F\_pass1 = IF\_freq - 22000;  F\_pass2 = IF\_freq + 22000;  F\_stop2 = IF\_freq + 24000;  A\_stop1 = 80;  A\_pass = 0.001;  A\_stop2 = 80;  IF\_BandPassSpecs = fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2',...  F\_stop1, F\_pass1, F\_pass2, F\_stop2, A\_stop1, A\_pass,A\_stop2, fs);  IF\_BandPass\_filter = design(IF\_BandPassSpecs, 'equiripple');  IF\_stage\_out = filter(IF\_BandPass\_filter, IF\_stage\_input);    % Plot The IF Stage Filtered Output  figure('Name','IF Stage Filter Output','NumberTitle','off');  plot(k2\*fs/num\_samples,abs(fftshift(fft(IF\_stage\_out))));  title('IF Stage Filter Output');  xlabel('Frequency in Hz');  ylabel('FFT');    % The BaseBand Detection  % Generate The Demodulation Oscillator With IF Frequency 25 KHz  baseband\_carrier = cos(2\*pi\*IF\_freq\*t)';  baseband\_signal = IF\_stage\_out .\* baseband\_carrier;    % Plot The BaseBand Stage Before Filtering  figure('Name','BaseBand Demodulation','NumberTitle','off');  plot(k2\*fs/num\_samples,abs(fftshift(fft(baseband\_signal))));  title('BaseBand Returned Signal Before Filtering');  xlabel('Frequency in Hz');  ylabel('FFT');    % The LPF To Fully Demodulate The Signal  F\_pass = 22000;  F\_stop = 24000;  A\_pass = 0.001;  A\_stop = 80;  LowPassSpecs = fdesign.lowpass('Fp,Fst,Ap,Ast',...  F\_pass, F\_stop, A\_pass, A\_stop, fs);  LowPass\_filter = design(LowPassSpecs, 'equiripple');  demodulated\_message = filter(LowPass\_filter, baseband\_signal);  demodulated\_message = 4.5 \* demodulated\_message;  demodulated\_message = downsample(demodulated\_message, 12);  clear F\_pass F\_stop F\_pass1 F\_pass2 F\_stop1 F\_stop2 A\_pass A\_stop A\_stop1 A\_stop2;  clear IF\_carrier baseband\_carrier;    % Plot The Final BaseBand Demodulated Message  figure('Name','Final Demodulated Message','NumberTitle','off');  plot(k\*sampling\_rate/N, abs(fftshift(fft(demodulated\_message))));  title('Final Demodulated Message');  xlabel('Frequency in Hz');  ylabel('FFT');    % Listen To The Demodulated Audio  sound (demodulated\_message, sampling\_rate); |