MATLAB implementation of a

Super heterodyne receiver

Name: Yousef Elbadry Haroun

Sec: 4

BN: 37

ID: 9220974

1. ***The transmitter***

Discussion

1. Loading the Audio Signals

The audio signals, each representing a unique input source are loaded into the system. These signals serve as the baseband input for the transmitter

Purpose:
Preparing input signals for further processing .

---

2. Resampling to a Common Sampling Frequency

Input signals may have different sampling frequencies .To ensure uniform processing, all signals are resampled to a common sampling frequency (Maximum among them).

Purpose:
Difference in sampling rates could lead to signal distortion or improper alignment during transmission.

---

3. Combining Stereo Channels

The audio signals are in stereo format, containing two channels (left and right). To simplify processing, the two channels of each signal are combined into a single monophonic channel by adding them together.

Purpose:
Reduces complexity of processing and modulation by converting stereo signals into single-channel signals

---

4. Padding Signals

Durations of input signals may vary, resulting in different signal lengths. To ensure uniformity during multiplexing, all signals are padded with zeros to match the length of the longest signal.

Purpose:
Ensures that all signals are fully represented during transmission without misalignment.

---

5. Generating Carrier Signals

Each signal has its own carrier frequency to prevent overlapping.

6. Increasing Sampling Frequency

The sampling frequency of the signals is increased by interpolation to match the requirements of the carrier frequencies. A higher sampling frequency ensures that the modulated signals can properly represent the higher frequency components during modulation.

Purpose:
Interpolation increases the resolution of the signals, allowing them to accurately represent the modulated waveforms and preventing aliasing during modulation

---

7. Modulating Signals

Each signal is modulated onto its respective carrier using Double Sideband Suppressed Carrier (DSB-SC) modulation. This process involves multiplying the input signals with their corresponding carrier waveforms.

Purpose:
This step enables the multiplexing of signals in the frequency domain, ensuring that each signal occupies a unique portion of the spectrum.

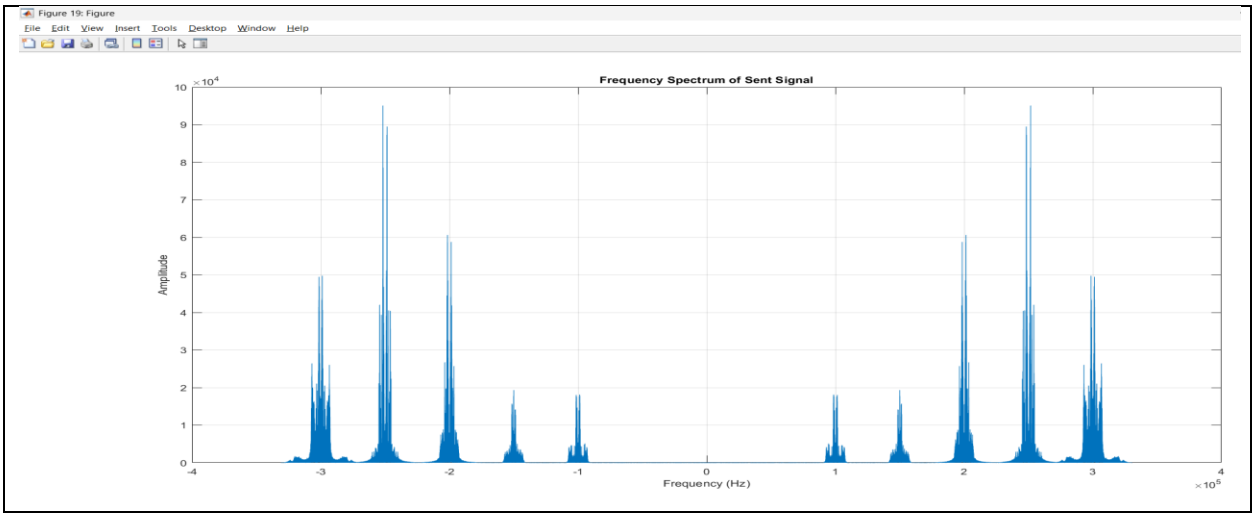
---

8. Combining Modulated Signals

The modulated signals are summed together to form the final FDM signal. This composite signal contains all the input signals, each occupying a distinct frequency band.

Purpose :
This single signal can now be transmitted over the communication channel.

Spectrum of sent signal



2. ***The RF stage***

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

Discussion

Responsible for isolating the desired signal from a composite Frequency Division Multiplexing (FDM) signal using a **(BPF)** to extract a specific modulated signal based on its carrier frequency and bandwidth. Also from its main points is to remove any images.

Tunable Band pass Filter Design with dynamic parametrization

A band pass filter (BPF) is designed for each modulated signal in the FDM system. The filter characteristics are defined by the carrier frequency and the bandwidth of the target signal.

I designed the bandpass filter to be tunable with variable (n) to choose the required signal from n=0 to n=4 ,besides that I put the higher bandwidth signals relative to the higher n because also I made the pass band tunable with the same variable n.

1. **Carrier Frequency :** 100k+(50k*n) which is tunable according to the value of n [0→4]
2. **Bandwidth :** 9000+(1500*n) which also tunable for the required signal .

Filter Design Method

An equiripple FIR filter design is employed for the BPF. This design method offers:

- **Flat Passband Response**: Ensures the desired signal is preserved without distortion.
- **Sharp Transition Band**: Minimizes overlap with adjacent signals.
- **High Stopband Attenuation**: Effectively removes unwanted frequencies

---

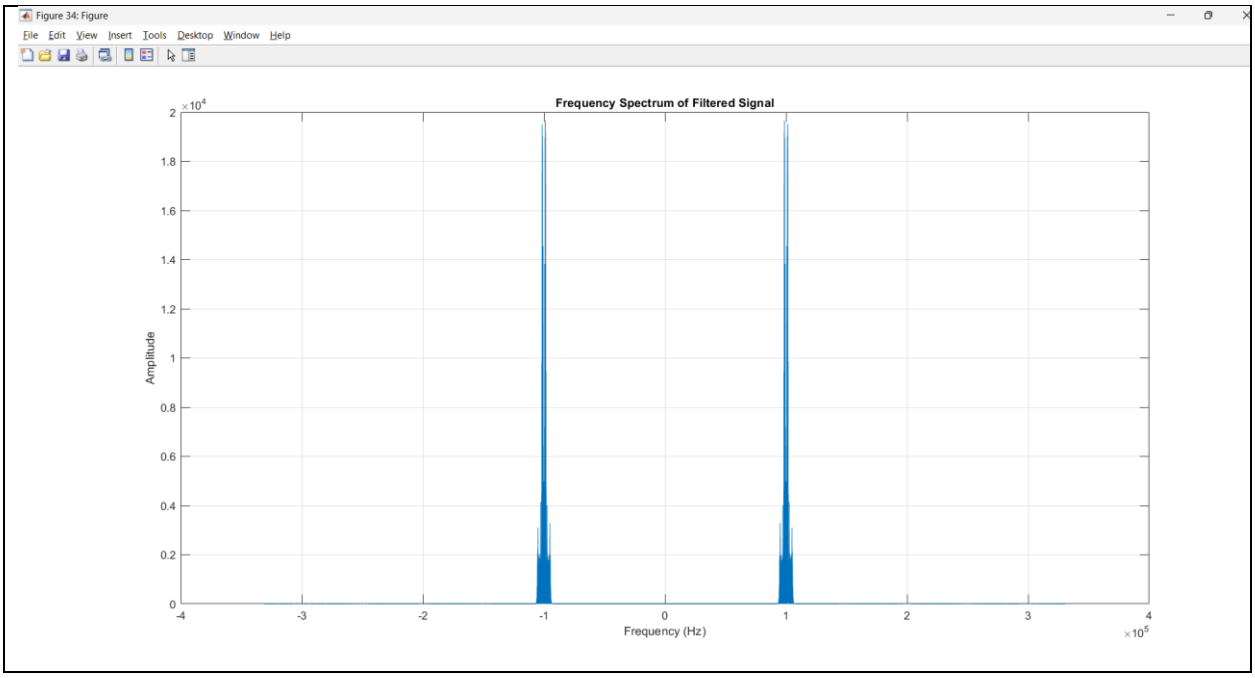
Tunable Mixer to shift signal to Wif (25 KHZ)

This mixing process shifts the modulated signal from its carrier frequency to the desired intermediate frequency Wif, so I designed the mixer tunable with the same variable **n** where it determines the Wosc frequency according to this variable by getting the Wc=100k+(500k*n)

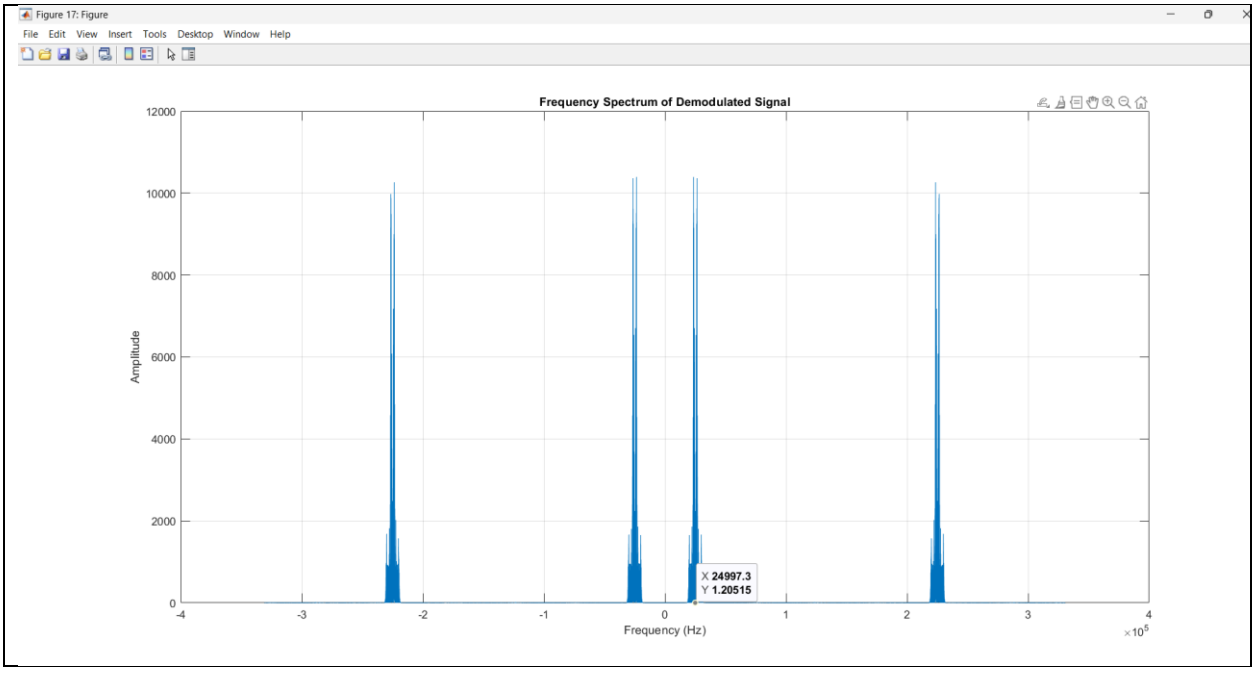
Figures

Assume we want to demodulate the first signal (at 100 KHZ).

The output of the RF filter (before the mixer)



The output of the mixer



3. ***The IF filter stage***

Discussion

IF stage prepares the signal for baseband recovery with minimal distortion and interference. Its main purpose is to isolate signal, Filter it and prepare for demodulation.

Tunable BW Band pass Filter Design for the IF Stage

A band pass filter is used in the IF stage to clean the signal at Wif.

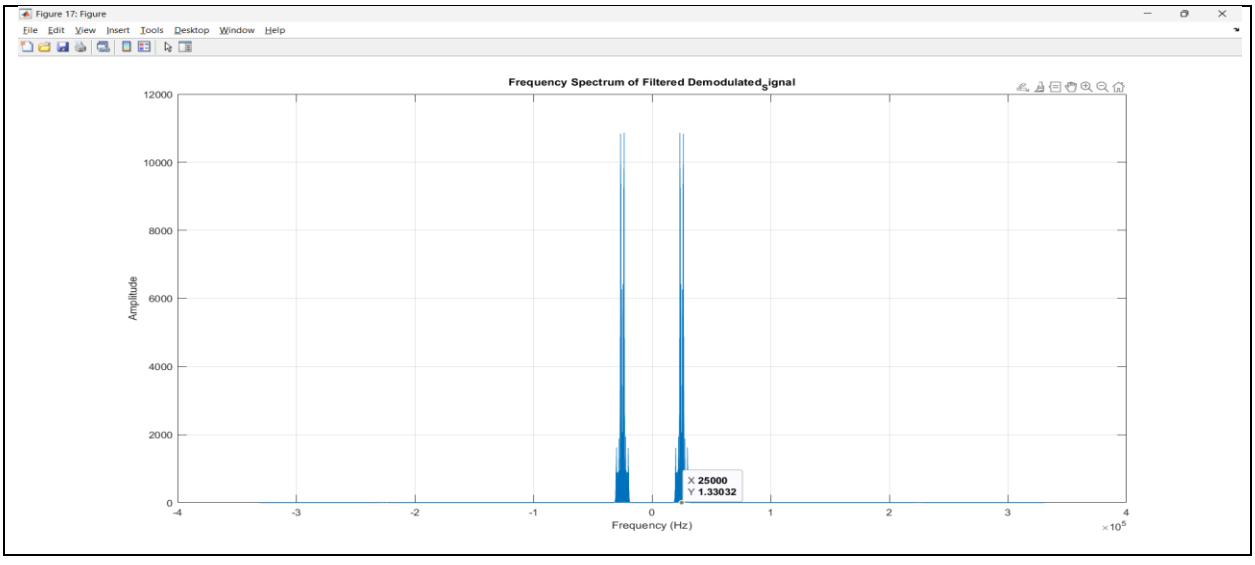
1. **Center Frequency (**Wif)
   - The filter is centered at Wif=25 kHZ corresponding to the intermediate frequency of the desired signal.
2. **Bandwidth**:
   - The filter's bandwidth is tunable by variable n like the previous filter. **9000+(1500*n)**
3. **Stopband Design**:
   - A high stopband attenuation is chosen to minimize the impact of noise and interference.

Filter Design Method

An equiripple FIR filter design is employed for the BPF. This design method offers:

- **Flat Passband Response**: Ensures the desired signal is preserved without distortion.
- **Sharp Transition Band**: Minimizes overlap with adjacent signals.
- **High Stopband Attenuation**: Effectively removes unwanted frequencies

Output of the IF filter



4. The baseband demodulator

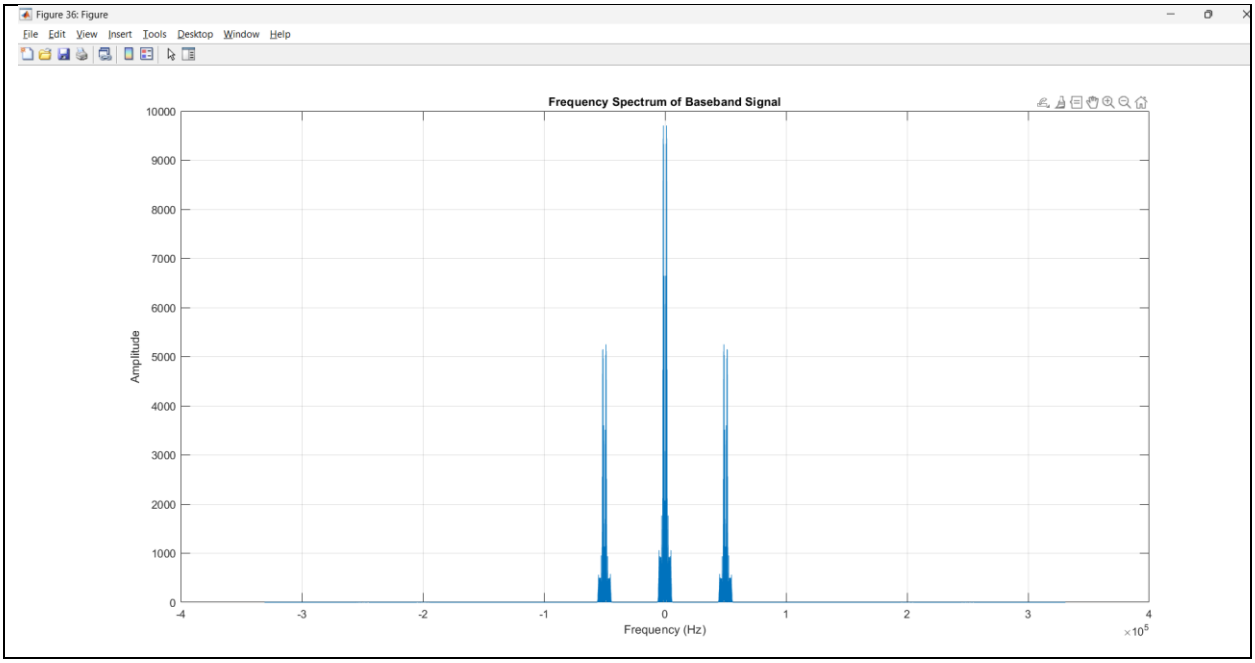
Discussion

Baseband detection is the final stage in the FDM receiver, where the downconverted intermediate frequency (IF) signal is processed to recover the original message signal. This stage involves two main steps: shifting the signal to baseband (0 Hz) and applying a low-pass filter (LPF) to extract the baseband content. Also, the signal is decimated to return it to its original sampling frequency for playback or further processing.

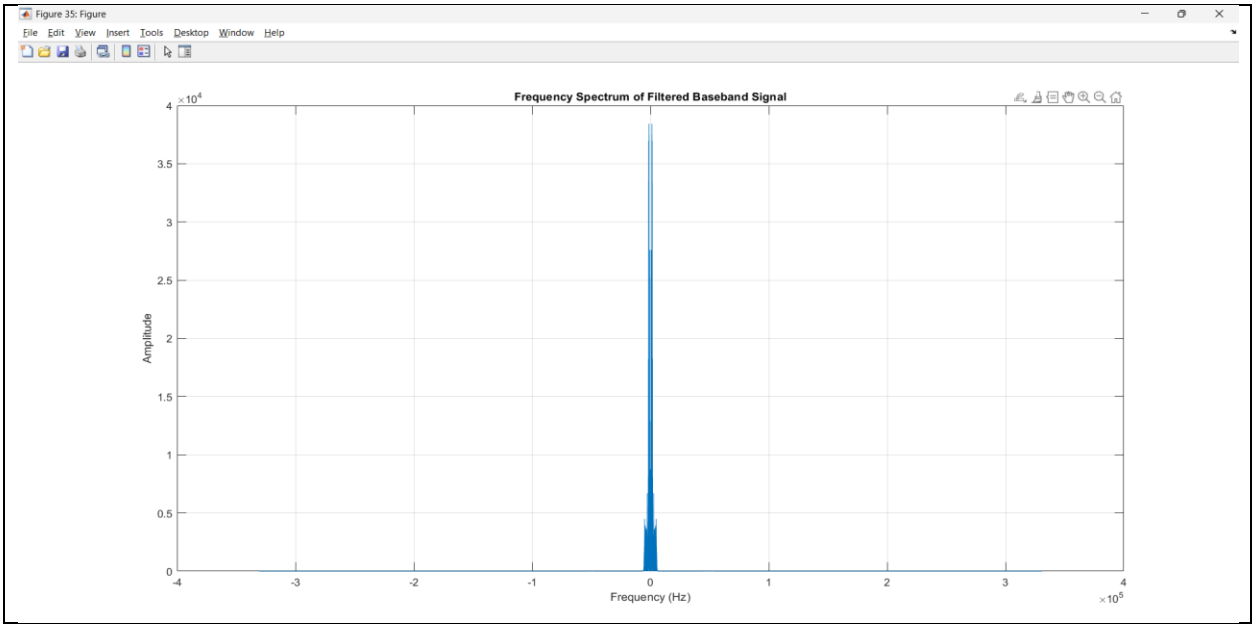
Steps in Baseband Detection

1. **Frequency Down conversion to Baseband**:
    - After the IF stage, the desired signal is centered around Wif=25 KHZ.
    - To recover the baseband signal, the IF signal is mixed with a sinusoidal oscillator of frequency Wif.
    - This multiplication shifts the signal's frequency spectrum from Wif to 0 Hz (baseband).
    - The resulting signal contains both the desired baseband content and higher-frequency components that need to be filtered out.
2. **Low-Pass Filtering (LPF)**:
    - A low-pass filter is applied to remove the high-frequency components.
    - Bandwidth of the LPF is easily estimated because the high signal would be at 50 KHZ so we may design the LPF to be with bandwidth 22 KHZ.
    - The equiripple FIR filter is used .
3. **Signal Scaling**:
    - The filtered signal is scaled to compensate for the effects of modulation and filtering. This ensures that the recovered signal maintains the correct amplitude levels for playback.
4. **Down sampling of the Fs**:
    - During the earlier modulation stages, the signal's sampling frequency was increased significantly by interpolation to meet the requirements of high-frequency carriers.
    - After recovering the baseband signal, it is down sampled to return it to its original sampling frequency, matching the format of the input signal.
    - Involves reducing the sampling rate while preserving the original signal content, allowing the recovered signal to be processed efficiently.
5. **Time Vector Reconstruction**:
    - A new time vector is generated to align with the original sampling frequency. This step ensures that the recovered signal is accurately represented in the time domain.

Output of the mixer (before the LPF)



Output of the LPF



Discussions :-

1- In two or three sentences, discuss the role of the RF, the IF and the baseband detector. Indicate why we need the IF stage?

→Discussed above in their tables.

2- Suppose you want to demodulate the first station (i.e. at $\omega 0$), plot the spectrum of the outputs of the RF, the IF and the baseband stages.

→Plotted above under their tables + all the plots (figures) are at the end of the document.

3-Use the command 'sound' on the demodulated signal and check whether you can successfully listen to the radio station. Please comment about this step in your report

→I used the command with all values of n (all the 5 sounds) and they worked correctly

4- Add "noise" to your signal and then play "sound" the signal. What is the effect of the noise?

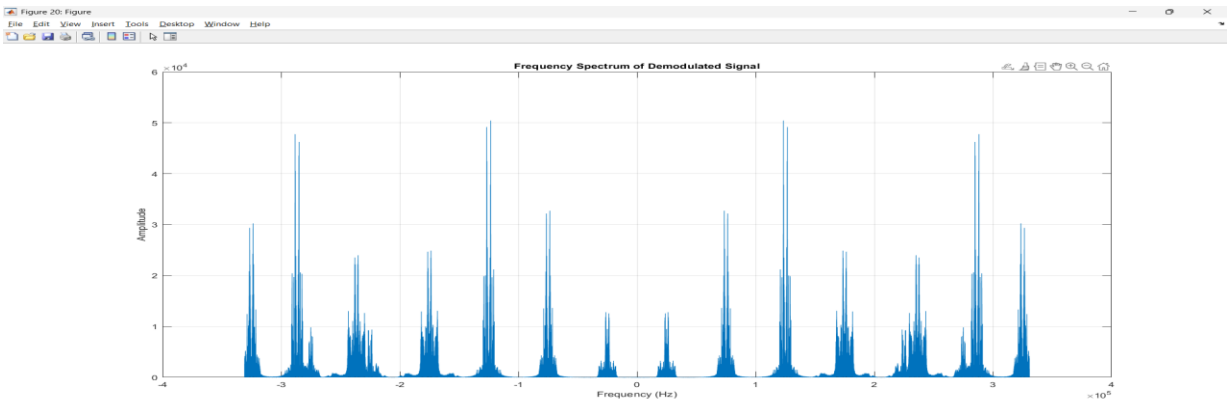
→ I can't hear the sound correctly and hear a sound like I am in a cave. The degradation of signal quality depends on the SNR:

- **High SNR**: Minimal distortion, the signal remains clear.
- **Low SNR**: Severe degradation, the signal becomes difficult to interpret.

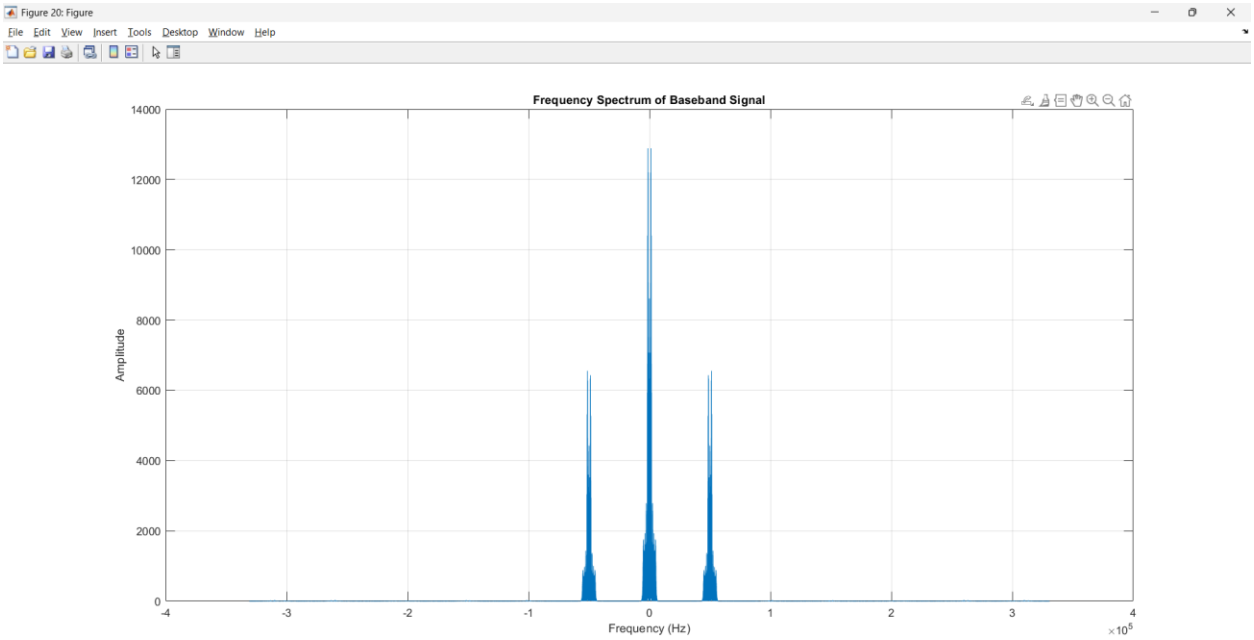
5- Repeat parts 2 and 3 but after removing the RF BPF. That is, the RF stage does not exist, what would happen if you try to demodulate the station at $\omega 0$?

→I heared 2 sounds instead of one sound and this is because of interference with the adjacent signals

Output of the RF stage



Output of the IF stage



6- What happens (in terms of the spectrum and the sound quality) if the receiver oscillator has frequency offset by 0.2 KHz and 1.2 KHz?

→When offset = 0.2 KHZ : Leads to a minor frequency shift, resulting in minimal degradation of the signal and sound quality.

→When offset = 1.2 KHZ : Leads major frequency shift , resulting distortion in the spectrum and degradation in sound quality.

(But in both cases sound is still hearable)

Loading the audio files

```
% Load the audio files
[x1, f1] = audioread('Short_BBCArabic2.wav');
[x2, f2] = audioread('Short_SkyNewsArabia.wav');
[x3, f3] = audioread('Short_FM9090.wav');
[x4, f4] = audioread('Short_QuranPalestine.wav');
[x5, f5] = audioread('Short_RussianVoice.wav');


% Choose a common sampling frequency (e.g., maximum of all sampling
frequencies)
Fs = max([f1, f2, f3, f4, f5]);


% Resample each signal to the common sampling frequency
x1 = resample(x1, Fs, f1);
x2 = resample(x2, Fs, f2);
x3 = resample(x3, Fs, f3);
x4 = resample(x4, Fs, f4);
x5 = resample(x5, Fs, f5);


% Calculate the duration of the first signal
SignalDuration = length(x1) / Fs;
```

Combine Stereo channels

```
monoSignal1 = x1(:, 1) + x1(:, 2);
monoSignal2 = x2(:, 1) + x2(:, 2);
monoSignal3 = x3(:, 1) + x3(:, 2);
monoSignal4 = x4(:, 1) + x4(:, 2);
monoSignal5 = x5(:, 1) + x5(:, 2);
```

Padding signals

```
% Calculate the maximum length
maxLength = max([length(monoSignal1), length(monoSignal2), length(monoSignal3),
length(monoSignal4), length(monoSignal5)]);


% Pad the signals with zeros
monoSignal1 = [monoSignal1; zeros(maxLength - length(monoSignal1), 1)];
monoSignal2 = [monoSignal2; zeros(maxLength - length(monoSignal2), 1)];
monoSignal3 = [monoSignal3; zeros(maxLength - length(monoSignal3), 1)];
monoSignal4 = [monoSignal4; zeros(maxLength - length(monoSignal4), 1)];
```

```
monoSignal5 = [monoSignal5; zeros(maxLength - length(monoSignal5), 1)];
```

PLOTTING SIGNALS

```
% Compute the FFT and frequency vector for each signal
N = maxLength; % All signals have been padded to the same length
freq = (-N/2:N/2-1) * (Fs / N); % Frequency vector


% Compute the FFT for each mono signal
spectrum1 = fftshift(abs(fft(monoSignal1)));
spectrum2 = fftshift(abs(fft(monoSignal2)));
spectrum3 = fftshift(abs(fft(monoSignal3)));
spectrum4 = fftshift(abs(fft(monoSignal4)));
spectrum5 = fftshift(abs(fft(monoSignal5)));


% Spectrum of Signal 1
figure;
plot(freq, spectrum1, 'b');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Mono Signal 1');
grid on;


% Spectrum of Signal 2
figure;
plot(freq, spectrum2, 'r');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Mono Signal 2');
grid on;


% Spectrum of Signal 3
figure;
plot(freq, spectrum3, 'g');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Mono Signal 3');
grid on;


% Spectrum of Signal 4
figure;
plot(freq, spectrum4, 'm');
xlabel('Frequency (Hz)');
```

```
ylabel('Amplitude');
title('Frequency Spectrum of Mono Signal 4');
grid on;


% Spectrum of Signal 5
figure;
plot(freq, spectrum5, 'k');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Mono Signal 5');
grid on;
```

Amplitude Modulation

```
% Carrier frequencies
fc1 = 100000;
fc2 = 150000;
fc3 = 200000;
fc4 = 250000;
fc5 = 300000;


%The sampling frequency is smaller the the carrier frequency
%Interp multiplies the sampling freq by 15
monoSignal1_HighFs = interp(monoSignal1, 15);
monoSignal2_HighFs = interp(monoSignal2, 15);
monoSignal3_HighFs = interp(monoSignal3, 15);
monoSignal4_HighFs = interp(monoSignal4, 15);
monoSignal5_HighFs = interp(monoSignal5, 15);


% the carrier frequency must be higher than the highest frequency...
% in the baseband signal to ensure proper modulation.
Fs_new=Fs*15;
t= 0 : 1/Fs_new : SignalDuration-(1/Fs_new);


% Generate carriers
carrier1 = cos(2 * pi * fc1 * t');
carrier2 = cos(2 * pi * fc2 * t');
carrier3 = cos(2 * pi * fc3 * t');
carrier4 = cos(2 * pi * fc4 * t');
carrier5 = cos(2 * pi * fc5 * t');
```

```
% DSB-SC modulation
modulatedSignal1 = monoSignal1_HighFs .* carrier1;
modulatedSignal2 = monoSignal2_HighFs .* carrier2;
modulatedSignal3 = monoSignal3_HighFs .* carrier3;
modulatedSignal4 = monoSignal4_HighFs .* carrier4;
modulatedSignal5 = monoSignal5_HighFs .* carrier5;
```

Plot the modulated signals

```
% frequency vector for modulated signals
freq = (-maxLength/2:maxLength/2-1) * (Fs / maxLength);

% Compute the FFT for each modulated signal
spectrumMod1 = fftshift(abs(fft(modulatedSignal1)));
spectrumMod2 = fftshift(abs(fft(modulatedSignal2)));
spectrumMod3 = fftshift(abs(fft(modulatedSignal3)));
spectrumMod4 = fftshift(abs(fft(modulatedSignal4)));
spectrumMod5 = fftshift(abs(fft(modulatedSignal5)));


% Plot each modulated signal's spectrum on a separate figure
% Spectrum of Modulated Signal 1
figure;
plot(freq_modulated, spectrumMod1, 'b');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Modulated Signal 1');
grid on;


% Spectrum of Modulated Signal 2
figure;
plot(freq_modulated, spectrumMod2, 'r');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Modulated Signal 2');
grid on;


% Spectrum of Modulated Signal 3
figure;
plot(freq_modulated, spectrumMod3, 'g');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Modulated Signal 3');
grid on;


% Spectrum of Modulated Signal 4
figure;
```

```
plot(freq_modulated, spectrumMod4, 'm');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Modulated Signal 4');
grid on;


% Spectrum of Modulated Signal 5
figure;
plot(freq_modulated, spectrumMod5, 'k');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Modulated Signal 5');
grid on;
```

Plot Sent signal

```
Sent_Signal=modulatedSignal1+modulatedSignal2+modulatedSignal3+modulatedSignal4
+modulatedSignal5;


% Compute the spectrum of the Sent signal
N = length(Sent_Signal);
freq = (-N/2:N/2-1) * (Fs_new / N);
spectrumsent = fftshift(abs(fft(Sent_Signal)));


% Plot the Sent signal in the frequency domain
figure;
plot(freq, spectrumsent);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Sent Signal');
grid on;
```

BPF 1

```
% parameters
%n=0 for first signal & n=1 for second signal and so on

n = 0;
fc = 100000+(50000*n);
% i put signal 1 has smallest bw then signal 2 and so on
% so that also bw can be controlled using n like the fc
bw = 9000+(1500*n);


% pass edges
```

```
F_pass1 = fc - bw/2;
F_pass2 = fc + bw/2;


% Stopband edges
F_stop1 = F_pass1 - bw/4;
F_stop2 = F_pass2 + bw/4;


% Create the filter specification object
bpSpec = fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2', F_stop1, F_pass1,
F_pass2, F_stop2, 70, 1, 70, Fs_new);


BandPassFilter = design(bpSpec, 'equiripple'); % FIR filter using equiripple
design


%fvtool(BandPassFilter);
```

Plot Filtered signal

```
Filtered_Signal1 = filter(BandPassFilter, Sent_Signal);
% Compute the spectrum of the filtered signal
N = length(Filtered_Signal1);
freq = (-N/2:N/2-1) * (Fs_new / N);
spectrumFiltered1 = fftshift(abs(fft(Filtered_Signal1)));


% Plot the filtered signal in the frequency domain
figure;
plot(freq, spectrumFiltered1);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Filtered Signal');
grid on;
```

Oscillator to shift to Wif

```
Wif = 25000;
Wosc1 = Wif + ((n*50000) + 100000 );
carrier_osc1 = cos(2*pi*Wosc1 * t');


Demodulated_Signal = Filtered_Signal1 .* carrier_osc1;


% Compute the FFT of the demodulated signal
N = length(Demodulated_Signal);
```

```
freq = (-N/2:N/2-1) * (Fs_new / N);
spectrumDemodulated = fftshift(abs(fft(Demodulated_Signal)));


% Plot the demodulated signal in the frequency domain
figure;
plot(freq, spectrumDemodulated);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Demodulated Signal');
grid on;
```

BPF 2

```
% Parameters for the BPF
bandwidth = 9000+(1500*n);

% Define the passband edges
F_pass1 = Wif - bandwidth/2;
F_pass2 = Wif + bandwidth/2;


% Stopband edges
F_stop1 = F_pass1 - bandwidth/4;
F_stop2 = F_pass2 + bandwidth/4;


% Design the filter specification object
bp2Spec = fdesign.bandpass('Fst1,Fp1,Fp2,Fst2,Ast1,Ap,Ast2',F_stop1, F_pass1,
F_pass2, F_stop2,60, 1, 60, Fs_new);


BandPassFilter2 = design(bp2Spec, 'equiripple'); %FIR bandpass filter using the
equiripple method


% fvtool(BandPassFilter2);
```

Plot Filtered signal

```
Filtered_Signal2 = filter(BandPassFilter2, Demodulated_Signal);


% Compute the spectrum of the Filtered Demodulated_Signal
N = length(Filtered_Signal2);
freq = (-N/2:N/2-1) * (Fs_new / N);
spectrumFiltered2 = fftshift(abs(fft(Filtered_Signal2)));


% Plot the Filtered Demodulated_Signal in the frequency domain
```

```
figure;
plot(freq, spectrumFiltered2);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Filtered Demodulated_Signal');
grid on;
```

Oscillator to shift from Wif to baseband

```
% Oscillator to shift from Wif to 0 Hz
carrier_osc2 = cos(2 * pi * Wif * t');


% Downconvert the filtered signal to baseband
Baseband_Signal = Filtered_Signal2 .* carrier_osc2;


% Compute the FFT of the baseband signal
N = length(Baseband_Signal);
freq = (-N/2:N/2-1) * (Fs_new / N);
spectrumBaseband = fftshift(abs(fft(Baseband_Signal)));


% Plot the baseband signal in the frequency domain
figure;
plot(freq, spectrumBaseband);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Baseband Signal');
grid on;
```

LPF at Baseband

```
F_pass = 22000;
F_stop = 22000 + 3000;


% Design the LPF
lpSpec = fdesign.lowpass('Fp,Fst,Ap,Ast', F_pass, F_stop, 1, 70, Fs_new);
LPF = design(lpSpec, 'equiripple'); % FIR low-pass filter


%fvtool(LPF);
```

Baseband filtered signal & return original fs

```
Filtered_Baseband_Signal = filter(LPF, Baseband_Signal);
Filtered_Baseband_Signal = Filtered_Baseband_Signal * 4;
```

```
% Compute the FFT of the Filtered_Baseband_Signal
N = length(Filtered_Baseband_Signal);
freq = (-N/2:N/2-1) * (Fs_new / N);
spectrumFilteredBaseband = fftshift(abs(fft(Filtered_Baseband_Signal)));


% Plot the frequency spectrum
figure;
plot(freq, spectrumFilteredBaseband);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Spectrum of Filtered Baseband Signal');
grid on;




% Decimate the signal by a factor of 10 to return to the original Fs
Returned_Signal = downsample(Filtered_Baseband_Signal, 15);


% Time vector for the original sampling frequency
t_original = (0:length(Returned_Signal)-1) / Fs;


% Plot the returned signal in the time domain
figure;
plot(t_original, Returned_Signal);
xlabel('Time (s)');
ylabel('Amplitude');
title('Time Domain Representation of Returned Signal (Original Fs)');
grid on;


% Play the signal at the original Fs
sound(Returned_Signal, Fs);
```

Figures

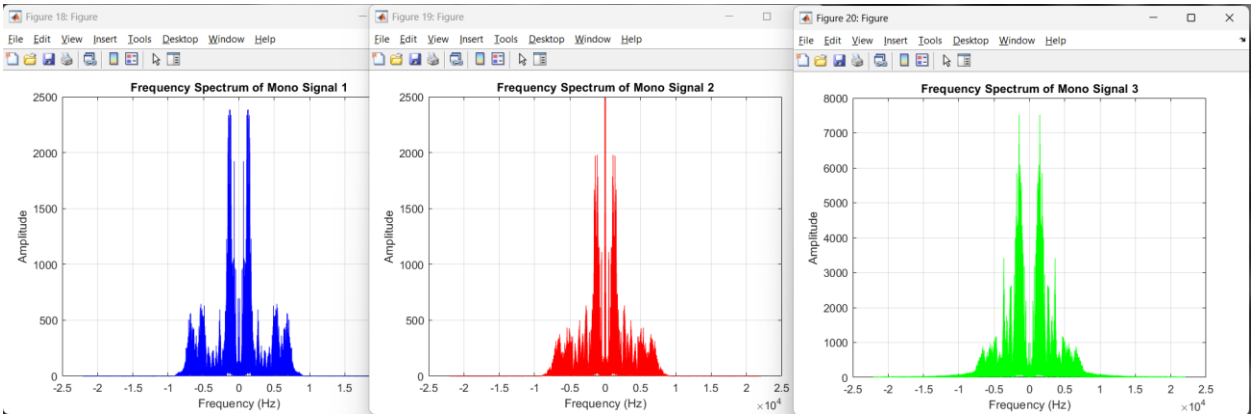


Figure 18: Figure
File Edit View Insert Tools Desktop Window Help
Frequency Spectrum of Mono Signal 1
Amplitude
Frequency (Hz)
Figure 19: Figure
File Edit View Insert Tools Desktop Window Help
Frequency Spectrum of Mono Signal 2
Amplitude
Frequency (Hz)
Figure 20: Figure
File Edit View Insert Tools Desktop Window Help
Frequency Spectrum of Mono Signal 3
Amplitude
Frequency (Hz)

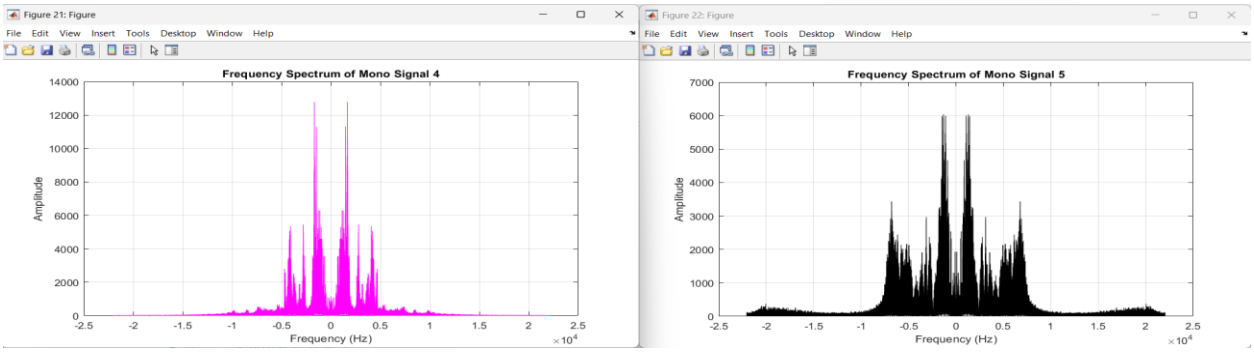


Figure 21: Figure
File Edit View Insert Tools Desktop Window Help
Frequency Spectrum of Mono Signal 4
Amplitude
Frequency (Hz)
Figure 22: Figure
File Edit View Insert Tools Desktop Window Help
Frequency Spectrum of Mono Signal 5
Amplitude
Frequency (Hz)

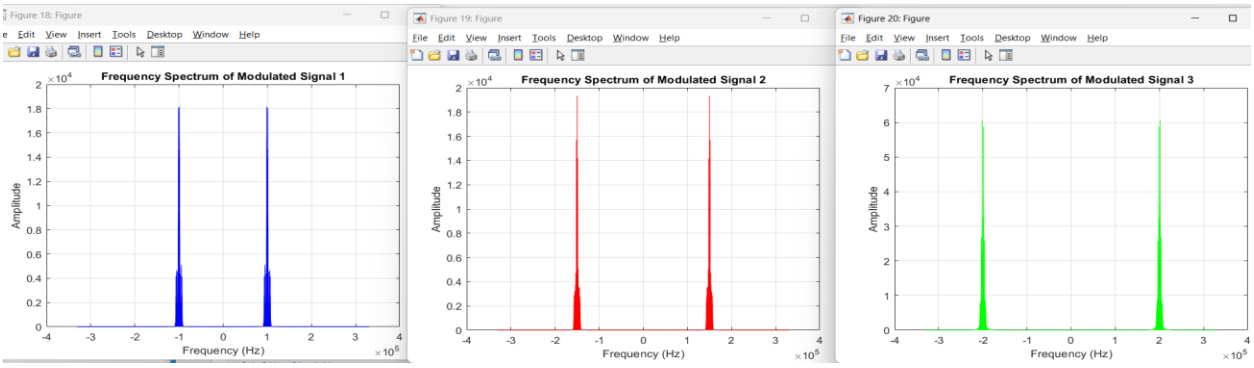


Figure 18: Figure
Frequency Spectrum of Modulated Signal 1
Amplitude
Frequency (Hz)
Figure 19: Figure
Frequency Spectrum of Modulated Signal 2
Amplitude
Frequency (Hz)
Figure 20: Figure
Frequency Spectrum of Modulated Signal 3
Amplitude
Frequency (Hz)

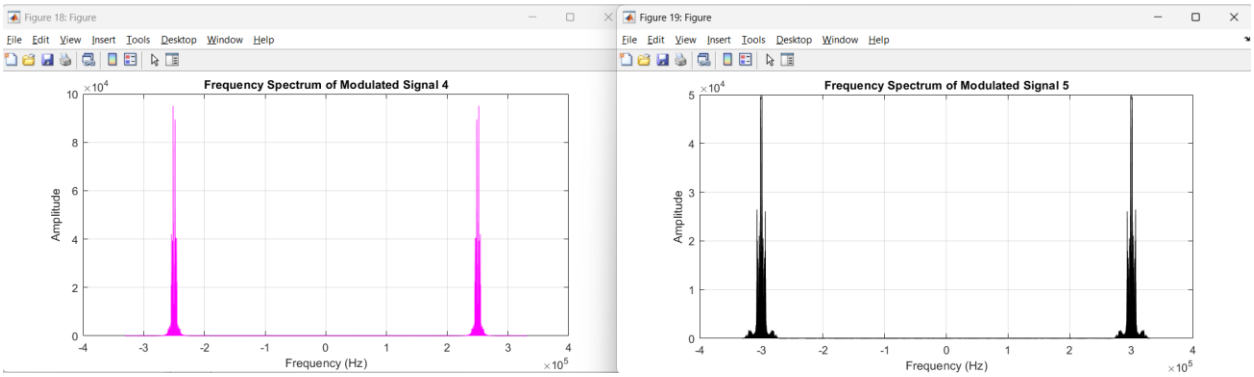


Figure 18: Figure
File Edit View Insert Tools Desktop Window Help
Frequency Spectrum of Modulated Signal 4
Amplitude
Frequency (Hz)
Figure 19: Figure
File Edit View Insert Tools Desktop Window Help
Frequency Spectrum of Modulated Signal 5
Amplitude
Frequency (Hz)

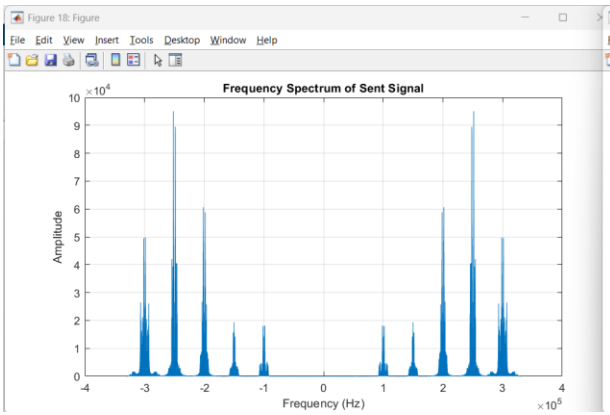


Figure 18: Figure
File Edit View Insert Tools Desktop Window Help
Frequency Spectrum of Sent Signal
Amplitude
Frequency (Hz)

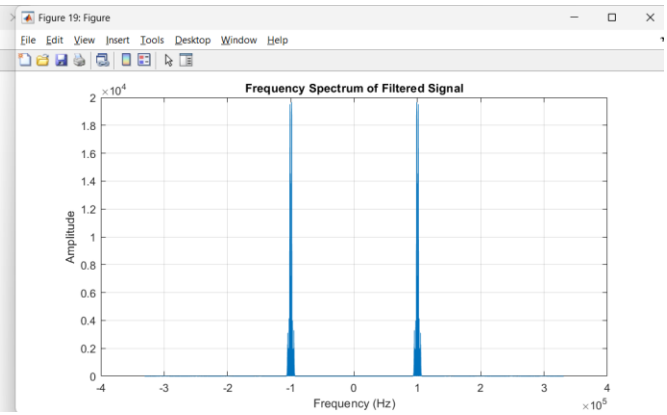


Figure 19: Figure
File Edit View Insert Tools Desktop Window Help
Frequency Spectrum of Filtered Signal
Amplitude
Frequency (Hz)

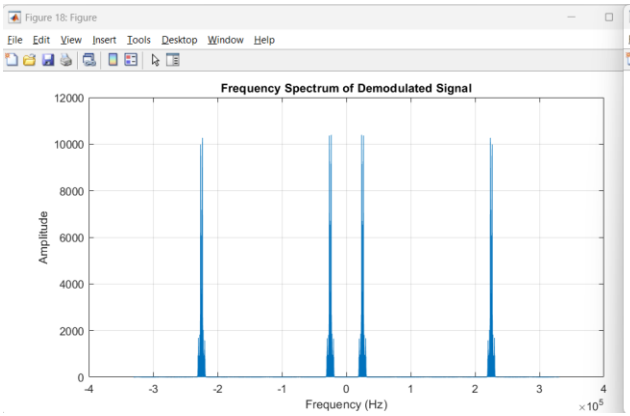


Figure 18: Figure
File Edit View Insert Tools Desktop Window Help
Frequency Spectrum of Demodulated Signal
Amplitude
Frequency (Hz)

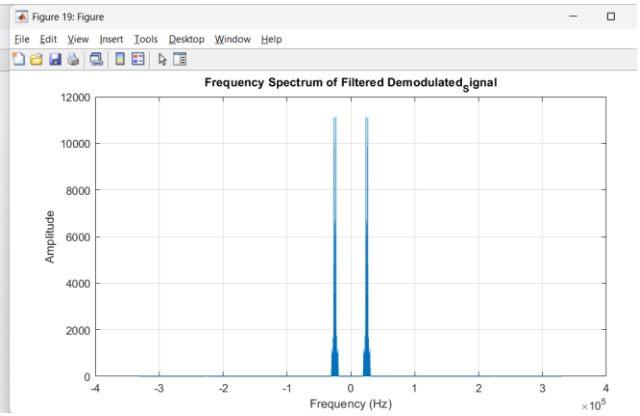


Figure 19: Figure
File Edit View Insert Tools Desktop Window Help
Frequency Spectrum of Filtered Demodulated Signal
Amplitude
Frequency (Hz)

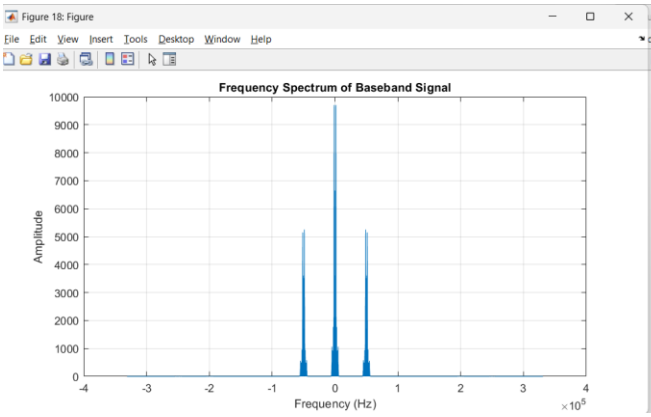
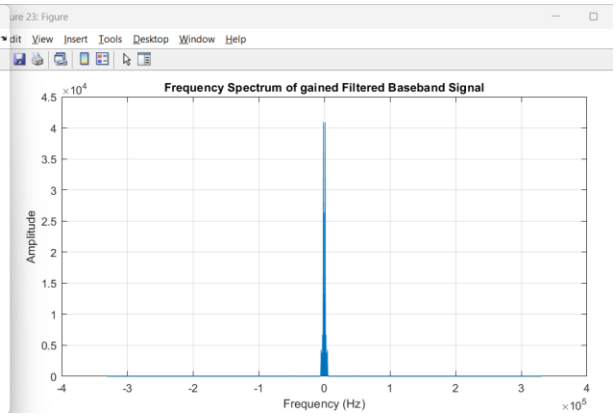


Figure 18: Figure
File Edit View Insert Tools Desktop Window Help
Frequency Spectrum of Baseband Signal
Amplitude
Frequency (Hz)


Frequency Spectrum of gained Filtered Baseband Signal
Amplitude
Frequency (Hz)