Final Project - Analogue Communications

Fadi Sarwat Farouk – 7432
Peter Anton Naguib – 7406
Yassin Medhat Tamam – 7574
Zeyad Ahmed Fathalla – 7621
Yamen Mohamed Saad – 7577

Amplitude Modulation (DSB-TC and DSB-SC)

Code

```
clear all;
close all;
[y, fs] = audioread('eric.wav');
Y = fftshift(fft(y));
f = linspace(-fs/2, fs/2, length(Y));

% Plot the spectrum
plot(f, abs(Y)/length(Y));
title('Spectrum of m');
figure();

% Apply filter
bw = 4000;
filt = ones(size(Y));
filt(f > bw|f<-bw) = 0;
y_filter = Y .* filt;
plot(f, abs(y_filter)/length(y_filter));
title('filtered signal spectrum');
% Inverse transform
y_filtered_time = ifft(ifftshift(y_filter));

% Ensure data type and scaling
y_filtered_time = real(y_filtered_time); % Ensure real values
y_filtered_time = double(y_filtered_time); % Convert to double if necessary

%% Normalize if values are outside the range [-1, 1]
max_val = max(abs(y_filtered_time));
if max_val > 1
   y_filtered_time = y_filtered_time / max_val;
end
fc = 100000;
U = 0.5;
Am = max(y_filtered_time);
Ac = Am/U; %modulationindex = Am/Ac
new_fs = 5 * fc;

resampled_signal = resample(y_filtered_time, new_fs, fs);
t1 = linspace(0, length(resampled_signal) / new_fs, length(resampled_signal));
t1 = t1';
carrier = Ac .* cos(2 * pi * fc * t1);
DSB_SC = resampled_signal .* carrier;
```

```
DSB_TC = (1 + U * resampled_signal/Am ) .* carrier;
%% Plot DSB-SC spectrum
figure();
subplot(1,2,1)
DSB_SC_spectrum = fftshift(fft(DSB_SC));
f_DSB_SC =linspace(- new_fs/2,  new_fs/2, length(DSB_SC));
plot(f_DSB_SC, abs(DSB_SC_spectrum)/length(DSB_SC_spectrum));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('DSB-SC Modulated Signal Spectrum');
% Plot DSB-TC spectrum
subplot(1,2,2)
DSB_TC_spectrum = fftshift(fft(DSB_TC));
f_DSB_TC=linspace(-new_fs/2,new_fs/2,length(DSB_TC));
plot(f_DSB_TC, abs(DSB_TC_spectrum)/length(DSB_TC_spectrum));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('DSB-TC Modulated Signal Spectrum');

%% envelop for DSB-SC
envelope_DSB_SC = abs(hilbert(DSB_SC));
figure;
plot(t1, DSB_SC);
hold on;
plot(t1,-envelope_DSB_SC,'r-',t1, envelope_DSB_SC,'-r','Linewidth',1.5);
hold off;
title('DSB_sc with envellope');
ylim([-2 2]);
xlim([2 2.5]);

%envelop for DSB-TC
envelope_DSB_TC = abs(hilbert(DSB_TC));
figure;
plot(t1, DSB_TC);
hold on;
plot(t1, -envelope_DSB_TC ,'r-', t1, envelope_DSB_TC, '-r', 'Linewidth', 1.5);
hold off;
title('DSB_Tc with envellope');
ylim([-2 2]);
xlim([2 2.5]);


demod_DSB_SC=envelope_DSB_SC.*cos(2 * pi * fc * t1);
%% plot demodulation signal
```

```
figure;
subplot(2, 1, 1);
plot(t1, envelope_DSB_TC);
xlabel('Time');
ylabel('Amplitude');
title('Received Signal - DSB-TC');
subplot(2, 1, 2);
plot(t1, demod_DSB_SC);
xlabel('Time');
ylabel('Amplitude');
title('Received Signal - DSB-SC');
% resample the envelope DSB_SC to hear it
recived_sig_DSB_SC = resample(envelope_DSB_SC, fc, fs);
% resample the envelope DSB_TC to hear it
recived_sig_DSB_TC = resample(envelope_DSB_TC, fc, fs);
%% sound the three msgs
sounds={y_filtered_time,envelope_DSB_SC,envelope_DSB_TC};
for i = 1:length(sounds)
    sound(sounds{i}, fs);
    pause(10);
end


%% Coherent detection
% when noise is added to DSB_SC with SNR = 0, 10, and 30
snr_values = [0, 10, 30];
for snr_dB = snr_values
    % Add noise to DSB_SC
    noisy_DSB_SC = awgn(DSB_SC, snr_dB);
    demodulated_noisy = noisy_DSB_SC .* cos(2*pi*fc*t1);
    f = new_fs / 2 * linspace(-1,1,length(DSB_SC));
    % Coherent detection with noise
    demodulated_noisy = double(real(demodulated_noisy));
    demodulated_noisy = fftshift(fft(demodulated_noisy));
    demodulated_noisy(f >= bw | f <= -bw) = 0;
    demodulated_noisy = ifft(ifftshift(demodulated_noisy));
    L = length(demodulated_noisy);
    % Plot received waveform and spectrum for each SNR value
    figure;
    subplot(2, 1, 1);
    plot(t1, demodulated_noisy);
    title(['Received Signal with SNR = ' num2str(snr_dB) ' dB - Time Domain']);
    xlabel('Time (s)');
    ylabel('Amplitude');
```

```
    subplot(2, 1, 2);
    plot(f, abs(fftshift(fft(demodulated_noisy))) / L);
    title(['Spectrum with SNR = ' num2str(snr_dB) ' dB']);
    xlabel('Frequency (Hz)');
    ylabel('Magnitude');

    demodulated_noisy_sound = resample(abs(demodulated_noisy), fs, new_fs);
    sound(abs(demodulated_noisy_sound), fs);
    pause(10);
end
%% coherent detection with frequency error
fc_error=100100;
for snr_dB = snr_values
    % Add noise to DSB_SC
    noisy_DSB_SC = awgn(DSB_SC, snr_dB);
    demodulated_noisy = noisy_DSB_SC .* cos(2*pi*fc_error*t1);
    f = new_fs / 2 * linspace(-1,1,length(DSB_SC));
    % Coherent detection with noise
    demodulated_noisy = double(real(demodulated_noisy));
    demodulated_noisy = fftshift(fft(demodulated_noisy));
    demodulated_noisy(f >= bw | f <= -bw) = 0;
    demodulated_noisy = ifft(ifftshift(demodulated_noisy));
    L = length(demodulated_noisy);
    % Plot received waveform and spectrum for each SNR value
    figure;
    subplot(2, 1, 1);
    plot(t1, demodulated_noisy);
    title(['Received Signal with SNR (FE) = ' num2str(snr_dB) ' dB - Time Domain']);
    xlabel('Time (s)');
    ylabel('Amplitude');

    subplot(2, 1, 2);
    plot(f, abs(fftshift(fft(demodulated_noisy))) / L);
    title(['Spectrum with SNR (FE) = ' num2str(snr_dB) ' dB']);
    xlabel('Frequency (Hz)');
    ylabel('Magnitude');

    demodulated_noisy_sound = resample(abs(demodulated_noisy), fs, new_fs);
    sound(abs(demodulated_noisy_sound), fs);
    pause(10);
end
%% phase error
for snr_dB = snr_values
```

```
    % Add noise to DSB_SC
    noisy_DSB_SC = awgn(DSB_SC, snr_dB);
    demodulated_noisy = noisy_DSB_SC .* cos(2*pi*fc*t1+20);
    f = new_fs / 2 * linspace(-1,1,length(DSB_SC));
    % Coherent detection with noise
    demodulated_noisy = double(real(demodulated_noisy));
    demodulated_noisy = fftshift(fft(demodulated_noisy));
    demodulated_noisy(f >= bw | f <= -bw) = 0;
    demodulated_noisy = ifft(ifftshift(demodulated_noisy));
    L = length(demodulated_noisy);
    % Plot received waveform and spectrum for each SNR value
    figure;
    subplot(2, 1, 1);
    plot(t1, demodulated_noisy);
    title(['Received Signal with SNR (PE) = ' num2str(snr_dB) ' dB - Time Domain']);
    xlabel('Time (s)');
    ylabel('Amplitude');

    subplot(2, 1, 2);
    plot(f, abs(fftshift(fft(demodulated_noisy))) / L);
    title(['Spectrum with SNR (PE) = ' num2str(snr_dB) ' dB']);
    xlabel('Frequency (Hz)');
    ylabel('Magnitude');

    demodulated_noisy_sound = resample(abs(demodulated_noisy), fs, new_fs);
    sound(abs(demodulated_noisy_sound), fs);
    pause(10);
end
```

Outputs

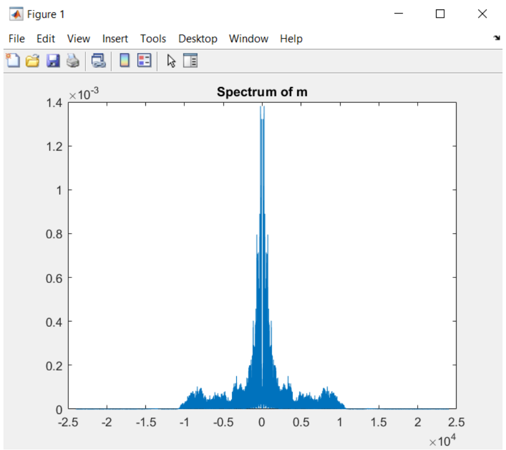


Figure 1
File Edit View Insert Tools Desktop Window Help
Spectrum of m

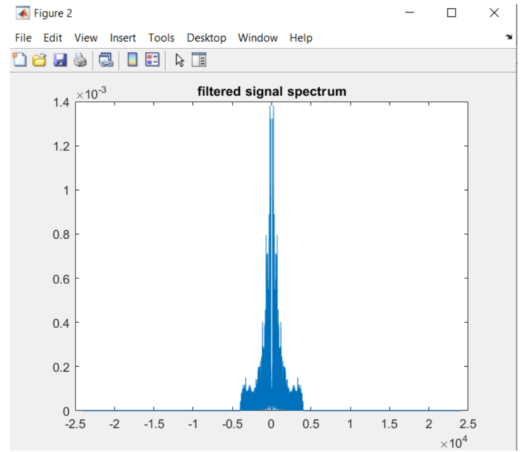


Figure 2
File Edit View Insert Tools Desktop Window Help
filtered signal spectrum

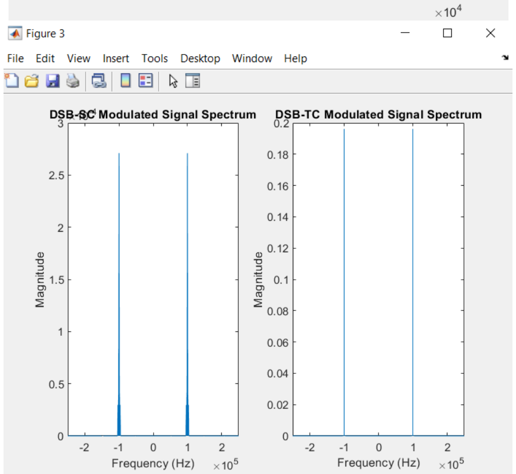


Figure 3
File Edit View Insert Tools Desktop Window Help
DSB-SC Modulated Signal Spectrum
DSB-TC Modulated Signal Spectrum
Magnitude
Frequency (Hz)

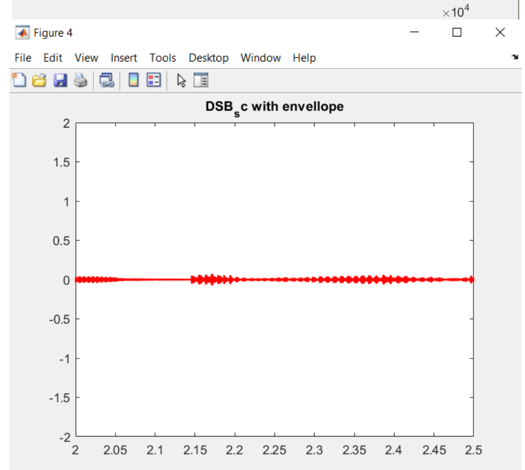


Figure 4
File Edit View Insert Tools Desktop Window Help
DSB$_s$c with envellope

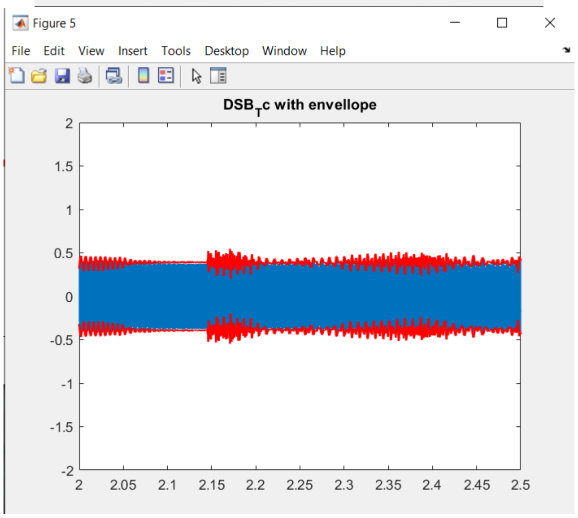


Figure 5
File Edit View Insert Tools Desktop Window Help
DSB$_T$c with envellope

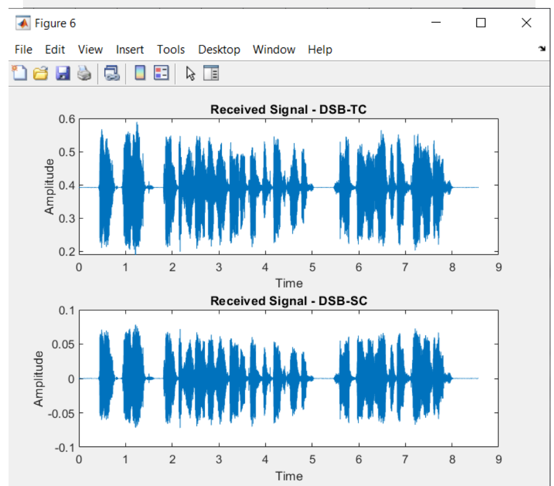


Figure 6
File Edit View Insert Tools Desktop Window Help
Received Signal - DSB-TC
Received Signal - DSB-SC
Amplitude
Time

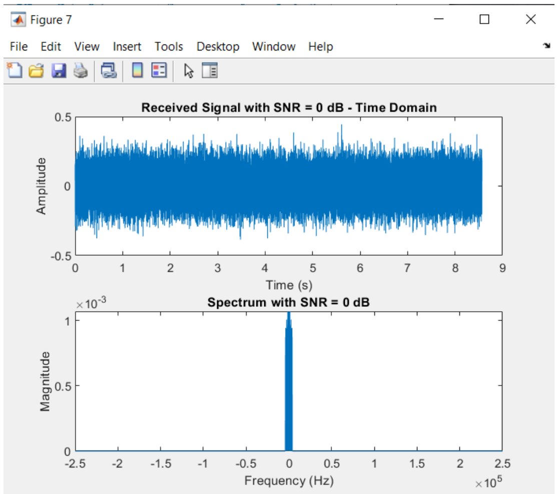


Figure 7
File Edit View Insert Tools Desktop Window Help
Received Signal with SNR = 0 dB - Time Domain
Amplitude
Time (s)
Spectrum with SNR = 0 dB
Magnitude
Frequency (Hz)

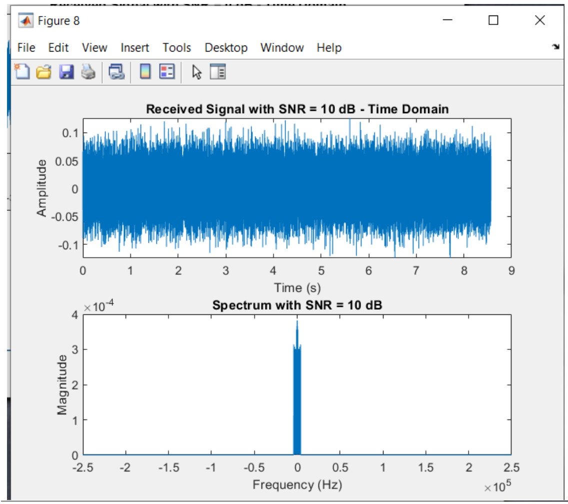


Figure 8
File Edit View Insert Tools Desktop Window Help
Received Signal with SNR = 10 dB - Time Domain
Amplitude
Time (s)
Spectrum with SNR = 10 dB
Magnitude
Frequency (Hz)

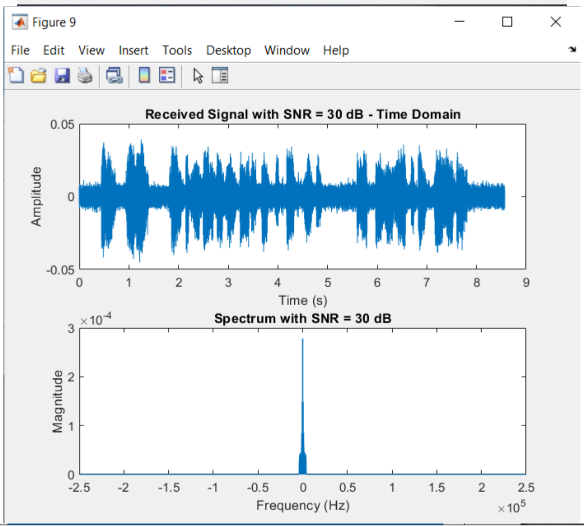


Figure 9
File Edit View Insert Tools Desktop Window Help
Received Signal with SNR = 30 dB - Time Domain
Amplitude
Time (s)
Spectrum with SNR = 30 dB
Magnitude
Frequency (Hz)

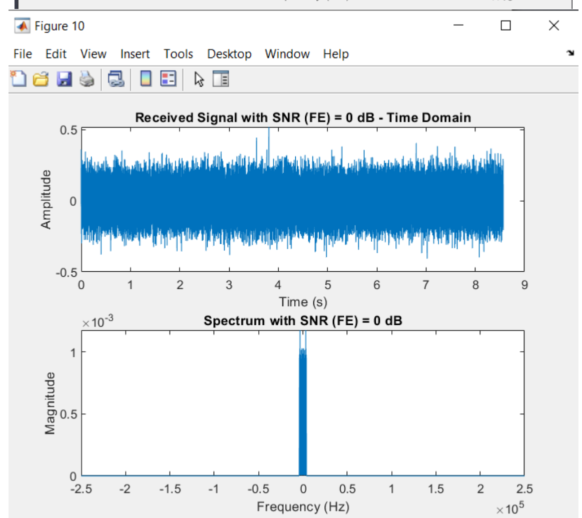


Figure 10
File Edit View Insert Tools Desktop Window Help
Received Signal with SNR (FE) = 0 dB - Time Domain
Amplitude
Time (s)
Spectrum with SNR (FE) = 0 dB
Magnitude
Frequency (Hz)

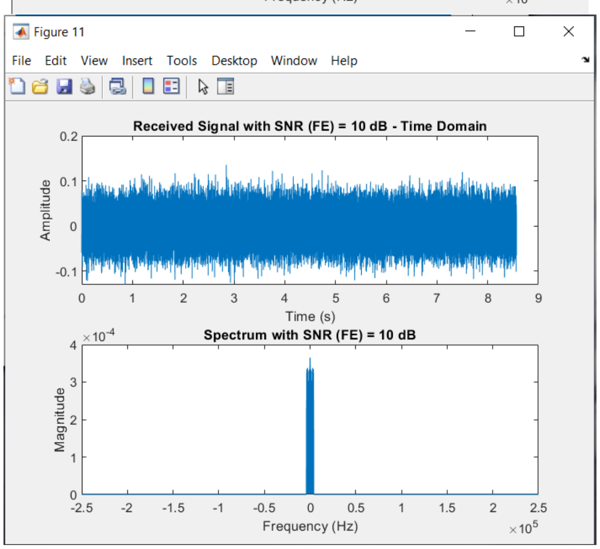


Figure 11
File Edit View Insert Tools Desktop Window Help
Received Signal with SNR (FE) = 10 dB - Time Domain
Amplitude
Time (s)
Spectrum with SNR (FE) = 10 dB
Magnitude
Frequency (Hz)

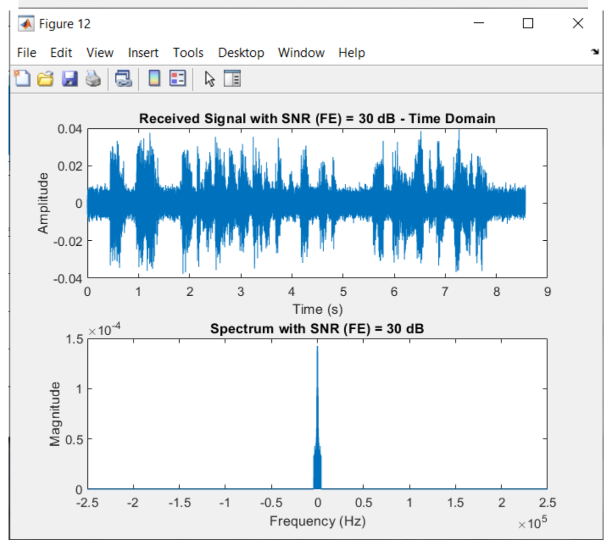
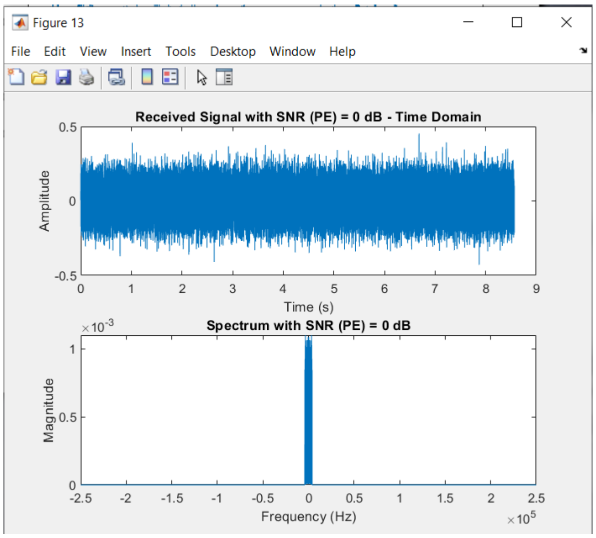
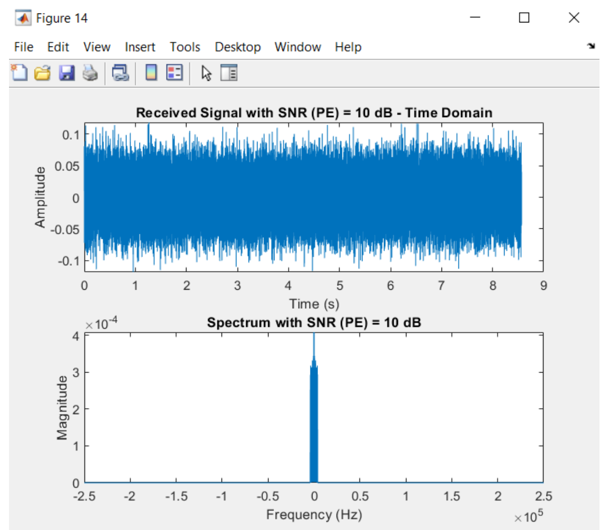
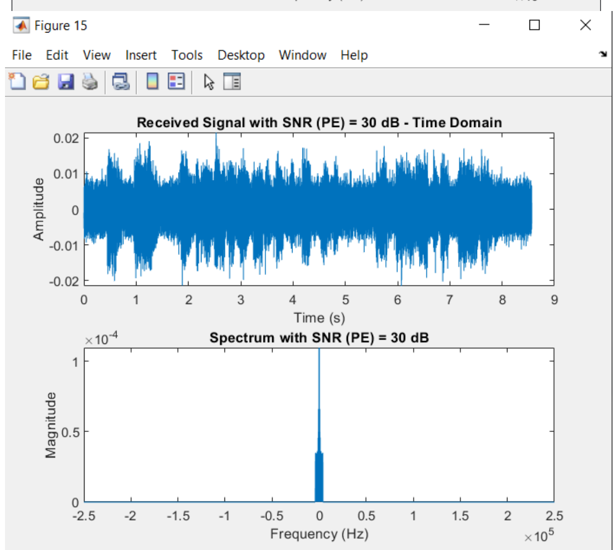


Figure 12
File Edit View Insert Tools Desktop Window Help
Received Signal with SNR (FE) = 30 dB - Time Domain
Amplitude
Time (s)
Spectrum with SNR (FE) = 30 dB
Magnitude
Frequency (Hz)







Questions

1) What observation can you make of this or which type of modulation the envelope detector can be used with?

In the DSB-TC the envelop detection worked and it was clear since there is no phase reversals in the DSB-SC the envelop detection wasn't optimal since the modulation index is critical and the coherent detection for it worked better.

2) Do you have a name for this phenomenon?

Frequency Offset

Single Side Band (SSB)

Code

```
%% 1-2. Same as DSB

[y, fs] = audioread('eric.wav');
L = length(y);
Y = fftshift(fft(y));
f = linspace(-fs/2, fs/2, L);

% Plot the spectrum
figure;
subplot(2, 1, 1);
plot(f, abs(Y) / L);
title('Original Spectrum of m');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Apply filter
bw = 4000;
Y(f >= bw | f <= -bw) = 0;

subplot(2, 1, 2);
plot(f, abs(Y) / L);
title('Filtered Spectrum of m');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

%% 3. Filtered signal in time domain (Inverse transform)
y_filtered_time = ifft(ifftshift(Y));

t1 = linspace(0, length(y_filtered_time) / fs, length(y_filtered_time));
t1 = t1';

y_filtered_time = real(double(y_filtered_time));

figure; plot(t1, y_filtered_time);
title('Filtered Signal of m Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

fc = 100000;
U = 0.5;
Am = max(y_filtered_time);
```

```
Ac = Am/U; % modulationindex = Am/Ac
new_fs = 5 * fc;


%% 4. Plot DSB-SC spectrum
message_for_sound = resample(y_filtered_time, new_fs, fs);
% sound(abs(message_for_sound), fs);

message = resample(y_filtered_time, new_fs, fs);
t1 = linspace(0, length(message) / new_fs, length(message));
t1 = t1';

L = length(message);
carrier = Ac .* cos(2*pi*fc*t1);
DSB_SC = message .* carrier;
DSB_SC_spectrum = fftshift(fft(DSB_SC));
f_DSB_SC = new_fs/2 * linspace(-1, 1, length(DSB_SC));

figure;
subplot(2, 1, 1);
plot(f_DSB_SC, abs(DSB_SC_spectrum) / L);
title('DSB-SC Modulated Signal Spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% 5. Obtain SSB_LSB
SSB_LSB = DSB_SC;
L = length(SSB_LSB);
f = new_fs / 2 * linspace(-1, 1, L);
F = fftshift(fft(SSB_LSB));
F(f>=fc | f<=-fc) = 0;
SSB_LSB = ifft(ifftshift(F));

% Plot SSB_LSB spectrum
subplot(2, 1, 2);
plot(f, abs(F) / L);
title('SSB LSB Modulated Signal Spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
%% 6. Coherent detection with no noise interference
demodulated_signal_ideal = SSB_LSB .* cos(2*pi*fc*t1);
demodulated_signal_ideal_sound = resample(abs(demodulated_signal_ideal), fs, new_fs);
sound(abs(demodulated_signal_ideal_sound), fs);
```

```
demodulated_signal_ideal = fftshift(fft(demodulated_signal_ideal));
demodulated_signal_ideal(f >= fc | f <= -fc) = 0;
demodulated_signal_ideal = ifft(ifftshift(demodulated_signal_ideal));

% Plot received waveform and spectrum
subplot(2, 1, 1);
plot(t1, demodulated_signal_ideal);
title('Received Signal (CD) in Time domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Spectrum of received signal
L = length(demodulated_signal_ideal);
Y_demodulated_ideal = fftshift(fft(demodulated_signal_ideal));
f_demodulated_ideal = linspace(-new_fs/2, new_fs/2, L);

subplot(2, 1, 2);
plot(f_demodulated_ideal, abs(Y_demodulated_ideal) / L);
title('Spectrum of Received Signal (CD)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([-10000 10000]);
%% 7. Repeat steps 5 and 6 with Butterworth filter
[b, a] = butter(4, fc / (new_fs / 2), 'low');
SSB_LSB_butter = filtfilt(b, a, demodulated_signal_ideal);
L = length(SSB_LSB_butter);

% Plot the spectrum of SSB-LSB signal using Butterworth filter
Y_SSB_LSB_butter = fftshift(fft(SSB_LSB_butter));
f_SSB_LSB_butter = linspace(-fs/2, fs/2, length(Y_SSB_LSB_butter));

figure;
subplot(3, 1, 1);
plot(f_SSB_LSB_butter, abs(Y_SSB_LSB_butter) / L);
title('Spectrum of SSB-LSB Signal (Butterworth Filter)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Coherent detection with Butterworth filter
demodulated_signal_butter = SSB_LSB_butter .* carrier;
t = linspace(0, length(demodulated_signal_ideal) / fs, length(demodulated_signal_ideal));
% Plot received waveform and spectrum
subplot(3, 1, 2);
plot(t, demodulated_signal_butter);
```

```
title('Received Signal (Coherent Detection with Butterworth) - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Spectrum of received signal with Butterworth filter
Y_demodulated_butter = fftshift(fft(demodulated_signal_butter));
f_demodulated_butter = linspace(-fs/2, fs/2, length(Y_demodulated_butter));

subplot(3, 1, 3);
plot(f_demodulated_butter, abs(Y_demodulated_butter) / L);
title('Spectrum of Received Signal (Coherent Detection with Butterworth)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

%% 8. For the ideal filter case, get the received signal again with noise
% when noise is added to SSB-SC with SNR = 0, 10, and 30
snr_values = [0, 10, 30];
for snr_dB = snr_values
   % Add noise to SSB-SC
   noisy_SSB_SC = awgn(SSB_LSB, snr_dB);
   demodulated_noisy = noisy_SSB_SC .* cos(2*pi*fc*t1);

   % Coherent detection with noise
   demodulated_noisy = double(real(demodulated_noisy));
   demodulated_noisy = fftshift(fft(demodulated_noisy));
   demodulated_noisy(f >= bw | f <= -bw) = 0;
   demodulated_noisy = ifft(ifftshift(demodulated_noisy));

   f = new_fs / 2 * linspace(-1,1,L);
   L = length(demodulated_noisy);

   % Plot received waveform and spectrum for each SNR value
   figure;
   subplot(2, 1, 1);
   plot(t1, demodulated_noisy);
   title(['Received Signal with SNR = ' num2str(snr_dB) ' dB - Time Domain']);
   xlabel('Time (s)');
   ylabel('Amplitude');

   subplot(2, 1, 2);
   plot(f, abs(fftshift(fft(demodulated_noisy))) / L);
   title(['Spectrum with SNR = ' num2str(snr_dB) ' dB']);
   xlabel('Frequency (Hz)');
   ylabel('Magnitude');
```

```
    demodulated_noisy_sound = resample(abs(demodulated_noisy), fs, new_fs);
    sound(abs(demodulated_noisy_sound), fs);
end
%% 9. For the ideal filter case, generate a SSB-TC
SSB_TC = carrier + SSB_LSB;
L = length(SSB_TC);

% Spectrum of received signal
Y_SSB_TC = fftshift(fft(SSB_TC));
f_SSB_TC = linspace(-new_fs/2, new_fs/2, length(Y_SSB_TC));

figure;
plot(f_SSB_TC, abs(Y_SSB_TC) / L);
title('Spectrum of Received SSB-TC Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Envelope detection
envelope_SSB_TC = abs(hilbert(SSB_TC));

% Plot received waveform
figure;
plot(t1, SSB_TC, 'b', t1, -envelope_SSB_TC, 'r', t1, envelope_SSB_TC, 'r', 'Linewidth', 1.5);
title('Received Message with Envelope Detection');
xlabel('Time (s)');
ylabel('Amplitude');
legend('SSB-TC', 'Envelope');
ylim([-5 5]);
xlim([3 3.5]);

envelope_SSB_TC = resample(envelope_SSB_TC, fs, new_fs);
sound(abs(envelope_SSB_TC), fs);
```

Outputs

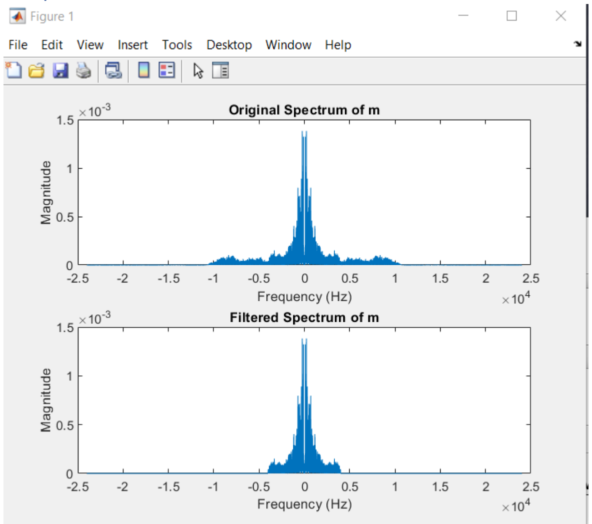


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Spectrum of m
Magnitude
Frequency (Hz)
Filtered Spectrum of m
Magnitude
Frequency (Hz)

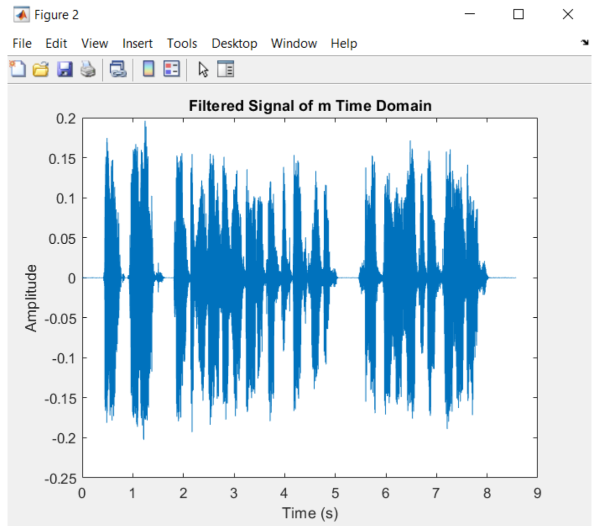


Figure 2
File Edit View Insert Tools Desktop Window Help
Filtered Signal of m Time Domain
Amplitude
Time (s)

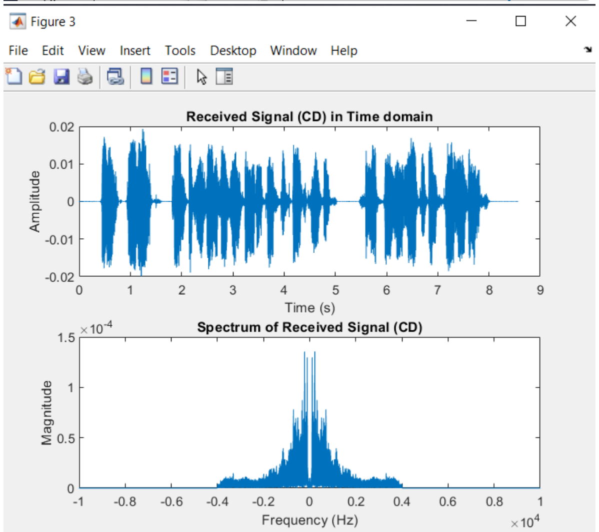


Figure 3
File Edit View Insert Tools Desktop Window Help
Received Signal (CD) in Time domain
Amplitude
Time (s)
Spectrum of Received Signal (CD)
Magnitude
Frequency (Hz)

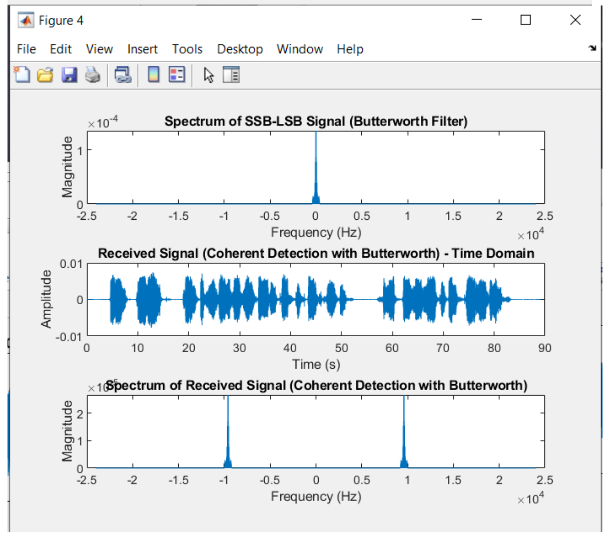


Figure 4
File Edit View Insert Tools Desktop Window Help
Spectrum of SSB-LSB Signal (Butterworth Filter)
Magnitude
Frequency (Hz)
Received Signal (Coherent Detection with Butterworth) - Time Domain
Amplitude
Time (s)
Spectrum of Received Signal (Coherent Detection with Butterworth)
Magnitude
Frequency (Hz)

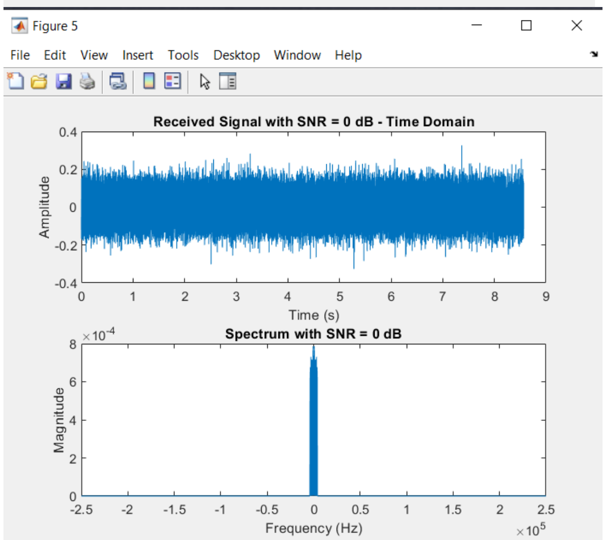


Figure 5
File Edit View Insert Tools Desktop Window Help
Received Signal with SNR = 0 dB - Time Domain
Amplitude
Time (s)
Spectrum with SNR = 0 dB
Magnitude
Frequency (Hz)

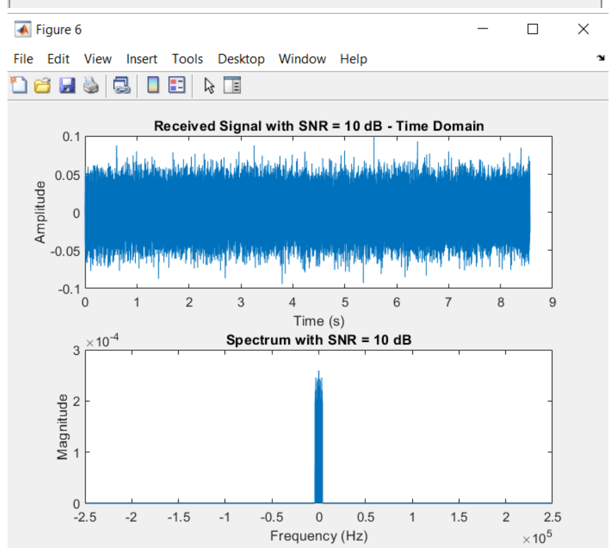


Figure 6
File Edit View Insert Tools Desktop Window Help
Received Signal with SNR = 10 dB - Time Domain
Amplitude
Time (s)
Spectrum with SNR = 10 dB
Magnitude
Frequency (Hz)

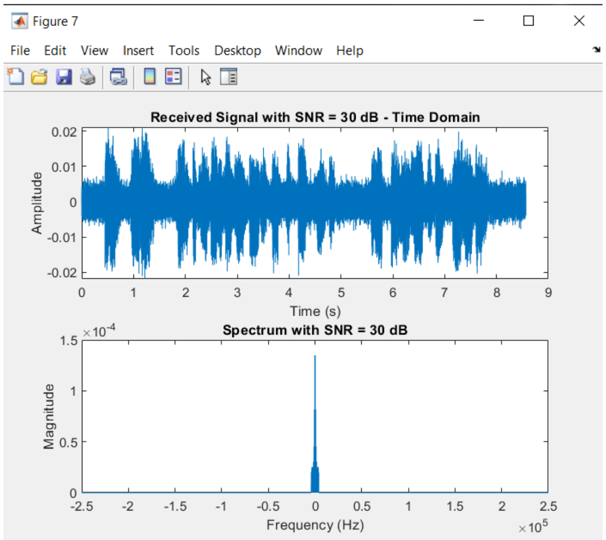


Figure 7
File Edit View Insert Tools Desktop Window Help
Received Signal with SNR = 30 dB - Time Domain
Amplitude
Time (s)
Spectrum with SNR = 30 dB
Magnitude
Frequency (Hz)

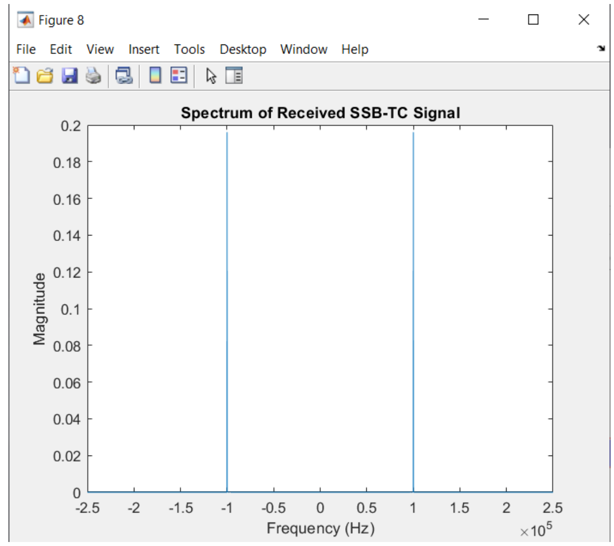


Figure 8
File Edit View Insert Tools Desktop Window Help
Spectrum of Received SSB-TC Signal
Magnitude
Frequency (Hz)

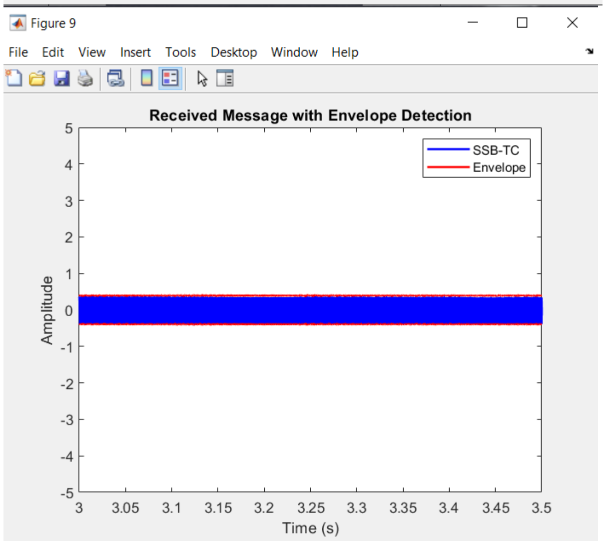


Figure 9
File Edit View Insert Tools Desktop Window Help
Received Message with Envelope Detection
SSB-TC
Envelope
Amplitude
Time (s)

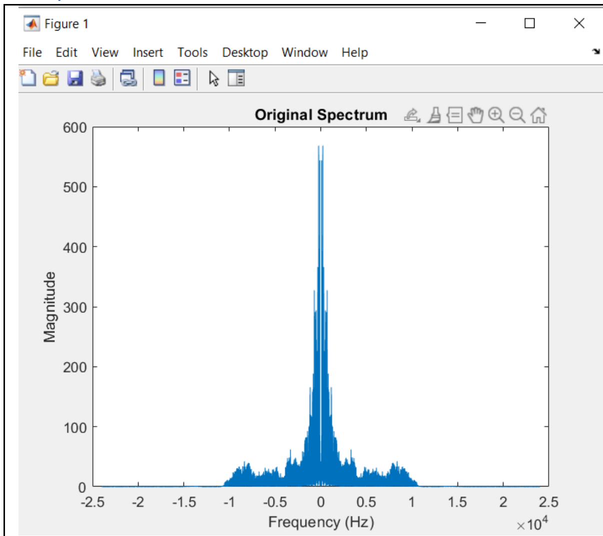
Frequency Modulation (FM)

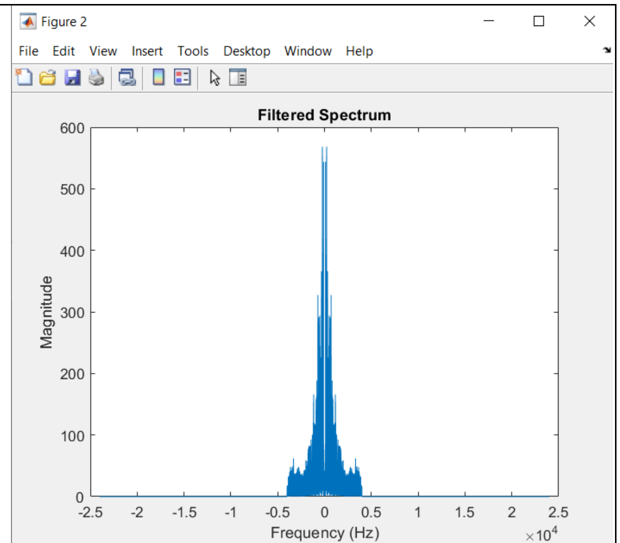
Code

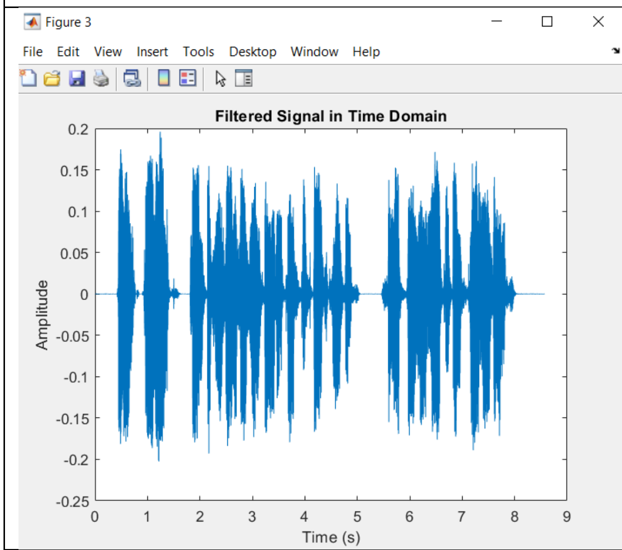
```
% 1. Read the audio file and find the spectrum
[audio, Fs] = audioread('eric.wav');
N = length(audio);
t = (0:N-1) / Fs;

% Calculate the spectrum
audio_spectrum_shifted = fftshift(fft(audio));

% Plot the spectrum
figure;
plot(linspace(-Fs/2, Fs/2, N), abs(audio_spectrum_shifted));
title('Original Spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% 2. Ideal filter to remove frequencies above 4 KHz
cutoff_frequency = 4000;
filter = (abs(linspace(-Fs/2, Fs/2, N)) <= cutoff_frequency);
audio_spectrum_filtered = audio_spectrum_shifted .* filter';

% 3. Obtain filtered signal in time and frequency domain
audio_filtered =real(ifft(ifftshift(audio_spectrum_filtered)));

% Plot the filtered spectrum
figure;
plot(linspace(-Fs/2, Fs/2, N), abs(audio_spectrum_filtered));
title('Filtered Spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Plot the filtered signal in time domain
figure;
plot(t, audio_filtered);
title('Filtered Signal in Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% 4. Sound the filtered audio signal
sound(abs(audio_filtered), Fs);

% 5. Generate NBFM signal
kf = 0.2/(2*pi*max(abs(cumsum(audio_filtered))./Fs));
Ac = 1;
Fc = 100000; % Carrier frequency
resampled_audio = resample(audio_filtered, 5*Fc, Fs);
```

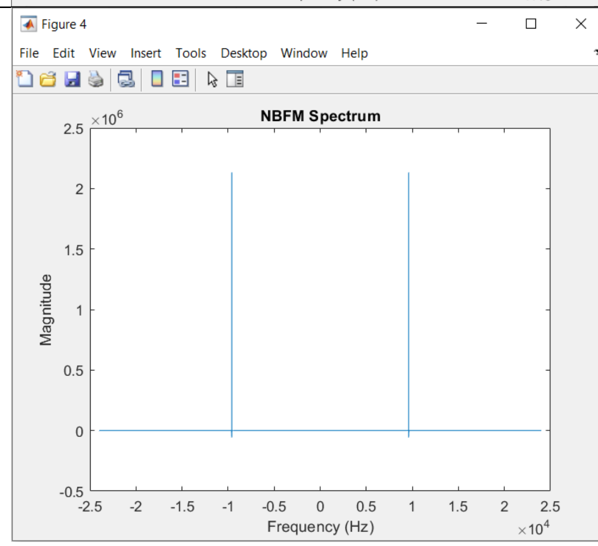
```
Fs_nbfm = 5 * Fc; % Sampling frequency for NBFM

%calculate time vector
tstart = 0;
tend = tstart + length(resampled_audio) / Fs_nbfm;
t1 = linspace(tstart, tend, length(resampled_audio));
t1 = t1';

%FM modulated signal
NBFM_signal = Ac * cos(2*pi*Fc*t1 + 2*pi*kf*cumsum(resampled_audio)./Fs_nbfm);

% Plot the NBFM spectrum
L = length(NBFM_signal);
NBFM_spectrum_shifted = real(fftshift(fft(NBFM_signal)));
f = Fs/2*linspace(-1,1,L);
figure;
plot(f, NBFM_spectrum_shifted)
title('NBFM Spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% 6. Condition for NBFM: Frequency deviation (delta_f) should be much smaller than the message
bandwidth (B)
B = cutoff_frequency;
delta_f = 75; % Frequency deviation
condition = delta_f < B;
disp(['Condition for NBFM: ', num2str(condition)]);

% 7. Demodulate NBFM signal
% Discriminator
dy = diff(NBFM_signal);
dy = [0; dy];

% envelope detector
demodulated_NBFM = abs(hilbert(dy)) -  mean(abs(hilbert(dy)));

% Plot the time-domain signal
figure;
plot(t1,demodulated_NBFM);
title('Demodulated NBFM Time-Domain Signal');
xlabel('Time (s)');
ylabel('Amplitude');
ylim([-2*10^-4 2*10^-4]);
```

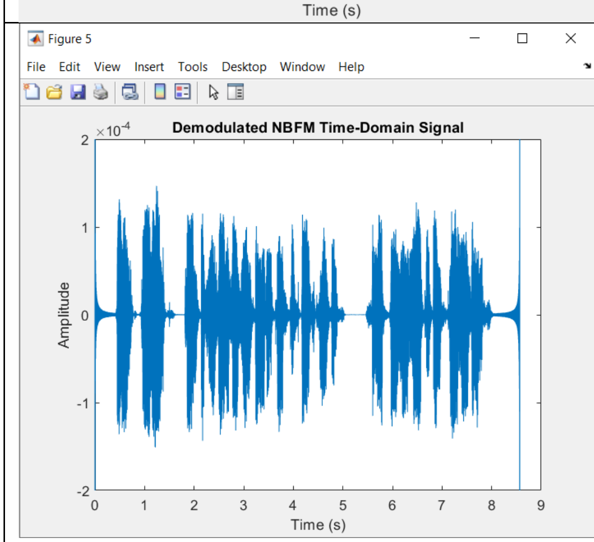
Outputs

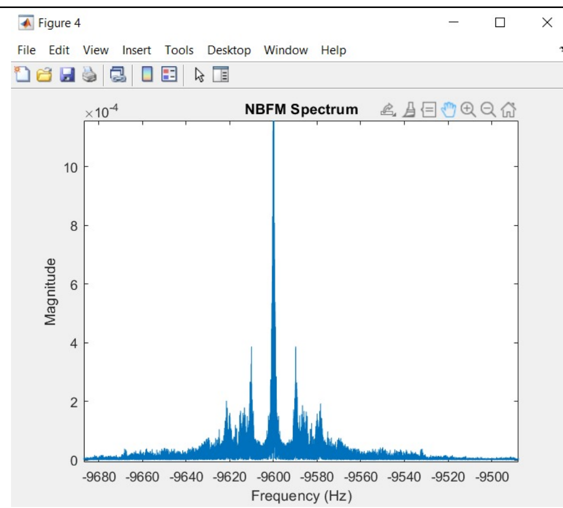












NBFM Zoomed in

Questions

1) What can you make out of the resulting plot?

We realized that the Bandwidth=2fm since beta is very small (condition for NBFM).

2) what is the condition we needed to achieve NBFM?

To acheive NBFM we need to satisfy the condition in this equation BW= 2(beta+1) * Fm that beta <= 0.2