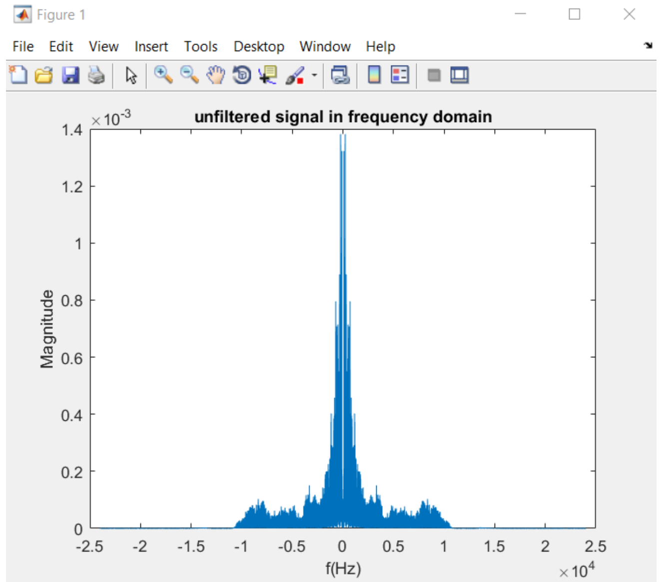
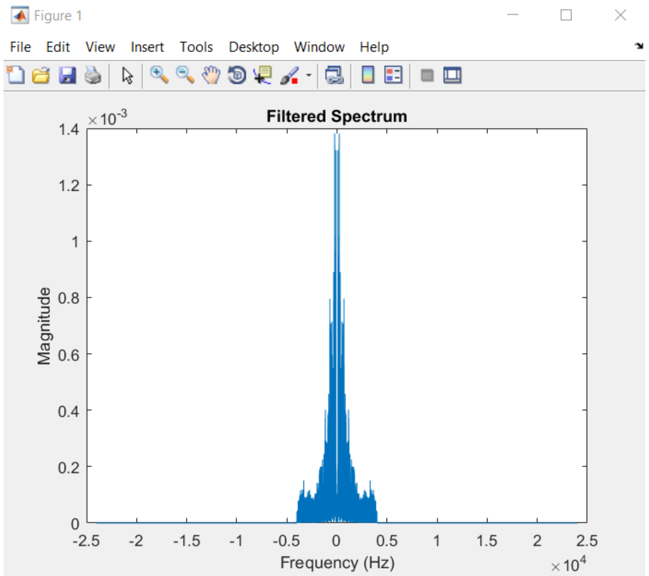
Experiment 1: Double Sideband Modulation

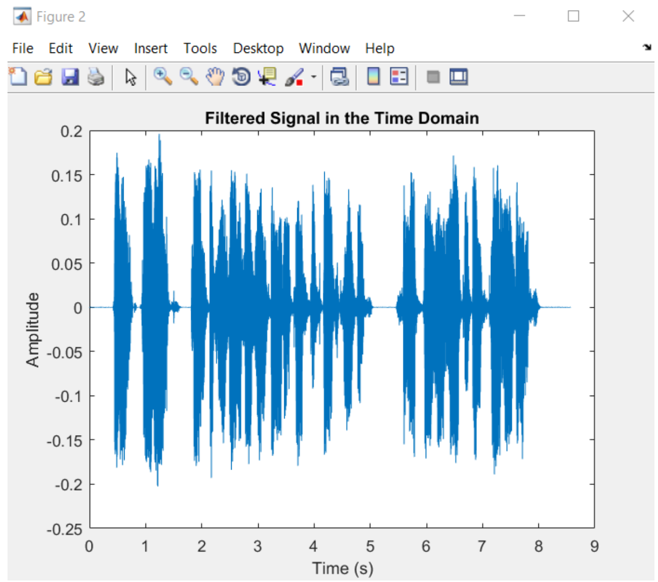
```
%% PART 1
clear all;
%% 1

filename = 'eric.wav';
[audio, fs] = audioread(filename);
sound(audio,fs);

len = length(audio);
audio_freq = fftshift(fft(audio));
f_axis = fs/2*linspace(-1,1,len);

% Plot the spectrum
figure;
plot(f_axis, abs(audio_freq)/len);
xlabel('f(Hz)');
ylabel('Magnitude');
title('unfiltered signal in frequency domain');
```




```
%% 2 and 3

% Filter as 4kHz
BW = 4000;
audio_freq(f_axis >= BW | f_axis <= -BW) = 0;
filtered_signal = ifft(ifftshift(audio_freq));
len = length(filtered_signal);
audio_freq = fftshift(fft(filtered_signal));
f_axis = fs/2*linspace(-1,1,len);

% Plot the filtered spectrum
figure;
plot(f_axis, abs(audio_freq)/len);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Filtered Spectrum');

t1 = linspace(0,len/fs,len);
t1=t1';
% Plot the filtered waveform
figure;
plot(t1, filtered_signal);
xlabel('Time (s)');
ylabel('Amplitude');
title('Filtered Signal in the Time Domain');
```

⇨ Obtain the filtered signal in time domain and frequency domain.



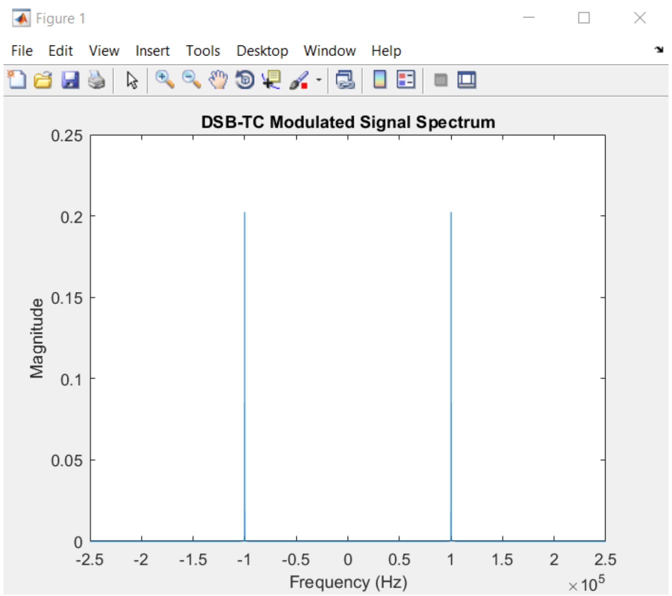


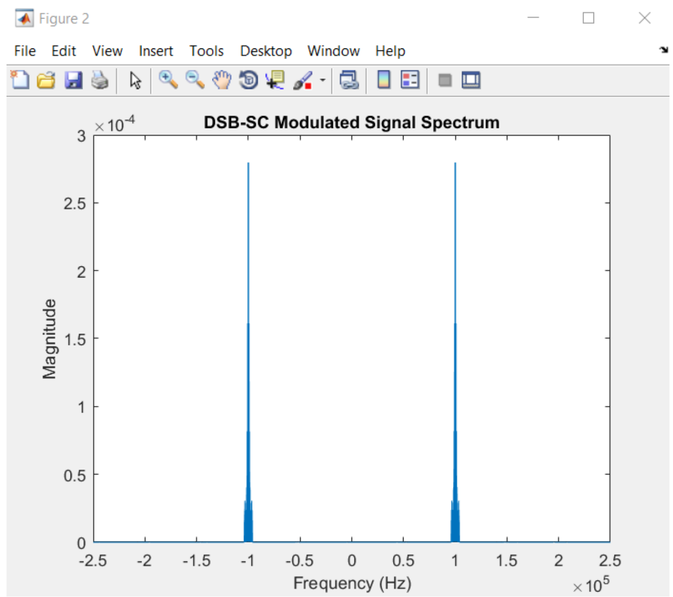

```
%% 4 sound
sound(abs(filtered_signal),fs);
```

```
%% 5

fc = 100000;
m = 0.5;
Am=max(filtered_signal);
Ac = Am/m;

filtered_signal  = resample(filtered_signal,5*fc,fs);
fs=5*fc;
t1=linspace(0,length(filtered_signal)/fs,length(filtered_signal));
t1=t1';
carrier = Ac.*cos(2*pi*fc*t1);

% DSB-TC
dsbtc = carrier.*(1+m*filtered_signal/Am);

% DSB-SC
dsbsc = carrier .* filtered_signal;

spectrum_dsbtc = fftshift(fft(dsbtc));
spectrum_dsbsc = fftshift(fft(dsbsc));

len = length(dsbtc);
f_axis=fs/2*linspace(-1,1,len);
```

```
% Plot the spectrum of the modulated signals
figure;
plot(f_axis, abs(spectrum_dsbtc)/len);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('DSB-TC Modulated Signal Spectrum');

len = length(dsbsc);
f_axis=fs/2*linspace(-1,1,len);

figure;
plot(f_axis, abs(spectrum_dsbsc)/len);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('DSB-SC Modulated Signal Spectrum');
```

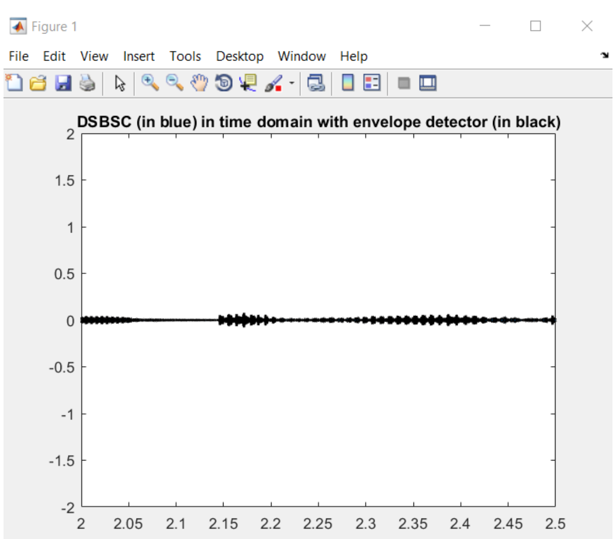
⇨ Modulation of the filtered signal using both DSB-TC and DSB-SC and plot their spectrum.

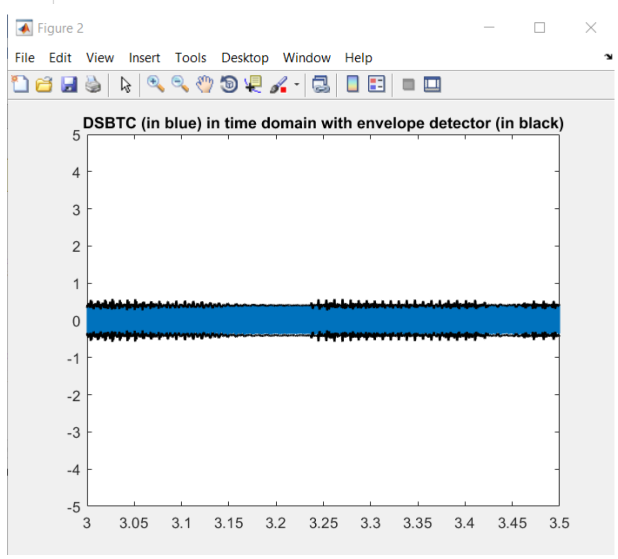





```
%% 6 Envelope

DSBSC_envelope = abs(hilbert(dsbsc));
DSBTC_envelope = abs(hilbert(dsbtc));
%% Plot
figure;
plot(t1,dsbsc);
hold on;
plot(t1,-DSBSC_envelope,'k-',t1, DSBSC_envelope,'-k','Linewidth',1.5); % phase reversal occurs
hold off;
title('DSBSC (in blue) in time domain with envelope detector (in black)');
ylim([-2 2]);
xlim([2 2.5]);

figure;
plot(t1,dsbtc);
hold on;
plot(t1,-DSBTC_envelope,'k',t1,DSBTC_envelope,'-k','Linewidth',1.5); % no phase reversal occurs
title('DSBTC (in blue) in time domain with envelope detector (in black)');
hold off;
ylim([-5 5]);
xlim([3 3.5]);
```

⇨ Apply envelope detector on both modulations and plot them in time domain.





```
%% 7 Resample and sound for DSB-SC

DSBSC_envelope = resample(abs(DSBSC_envelope), fs/5, fs);
sound(abs(DSBSC_envelope), fs/5);
% Observation: not detected very well, distorted sound
% Envelope detector can only be used for DSB-TC
```

```
%% Resample and Sound for DSB-TC

DSBTC_envelope = resample(abs(DSBTC_envelope), fs/5, fs);
sound(abs(DSBTC_envelope), fs/5);
% Observation: more accurately detected, less distortion.
```

```
%% 8
SNR = [0,10,30];
for i=1:length(SNR)

    % generate signal+noise
    dsbsc_noise = awgn(dsbsc, SNR(i));

    % demodulate using coherent detector
    demodulatedsignal = dsbsc_noise.*cos(2*pi*fc*t1);

    clear dsbsc_noise;

    % fourier transform
    demodulatedsignal_in_FD = fftshift(fft(demodulatedsignal));

    % LPF at Fm
    demodulatedsignal_in_FD(f_axis >= BW | f_axis <= -BW) = 0;

    % inverse fourier transform to get demodulated signal in time domain
    demodulatedsignal = ifft(ifftshift(demodulatedsignal_in_FD));

    % plot demodulated signal in time domain
    figure; plot(t1, demodulatedsignal); title([num2str(SNR(i)),' SNR demodulated signal in time domain']);

    % fourier transform
    len = length(demodulatedsignal);
    spectrum_demodulated = fftshift(fft(demodulatedsignal));
    f_axis = fs/2*linspace(-1,1,len);

    % fourier transform
    len = length(demodulatedsignal);
    spectrum_demodulated = fftshift(fft(demodulatedsignal));
    f_axis = fs/2*linspace(-1,1,len);

    % plot demodulated signal in frequency domain
    figure; plot(f_axis, abs(spectrum_demodulated) / len); title([num2str(SNR(i)),' SNR demodulated signal in frequency domain']);

    % resample to sound the demodulated signal
    demodulatedsignal = resample(abs(demodulatedsignal), fs/5, fs);
    sound(abs(demodulatedsignal), fs/5);
    pause(10);
end
clear demodulatedsignal;
clear demodulatedsignal_in_FD;
```

⇨ Observations: As the SNR value increases, the sound of the message becomes purer with less interferences.

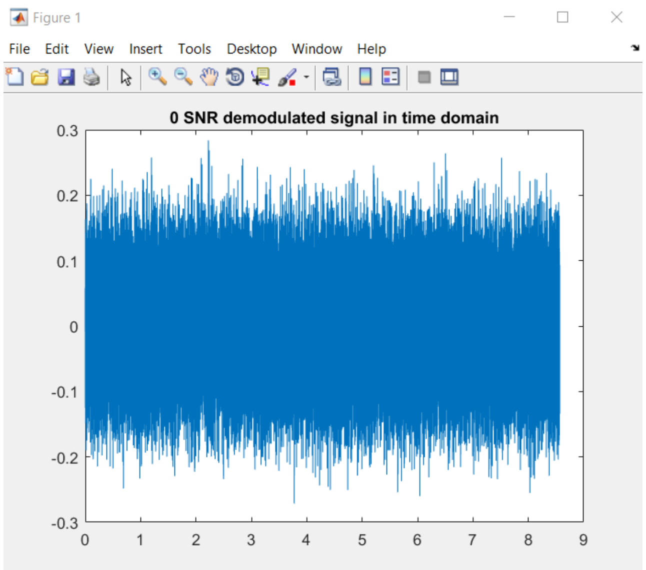


Figure 1
File Edit View Insert Tools Desktop Window Help
0 SNR demodulated signal in time domain

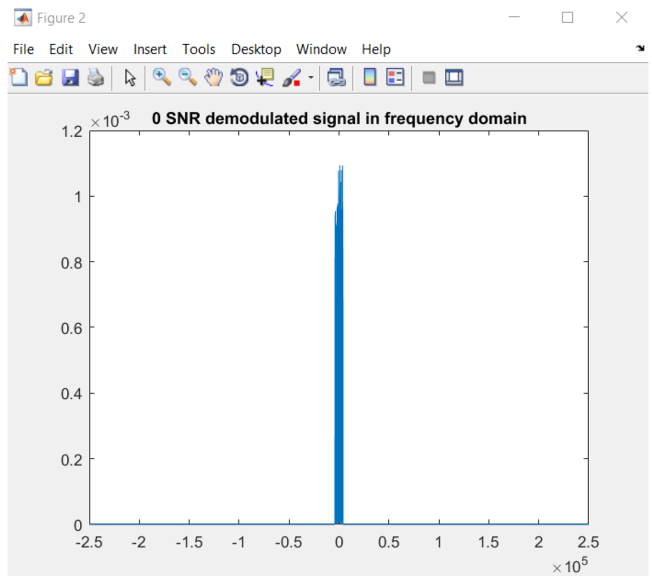


Figure 2
File Edit View Insert Tools Desktop Window Help
0 SNR demodulated signal in frequency domain
$\times 10^{-3}$
$\times 10^{5}$

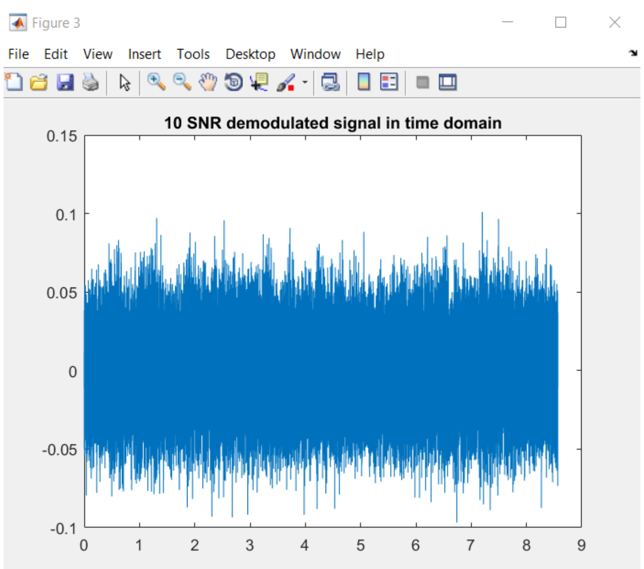


Figure 3
File Edit View Insert Tools Desktop Window Help
10 SNR demodulated signal in time domain

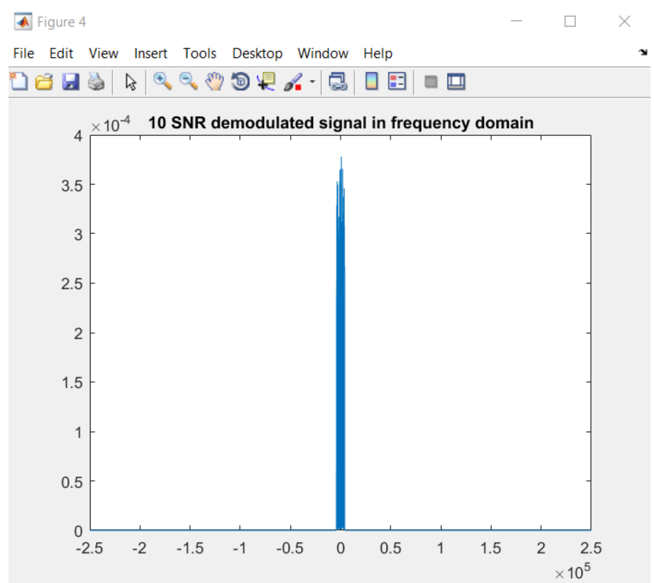


Figure 4
File Edit View Insert Tools Desktop Window Help
10 SNR demodulated signal in frequency domain
$\times 10^{-4}$
$\times 10^{5}$

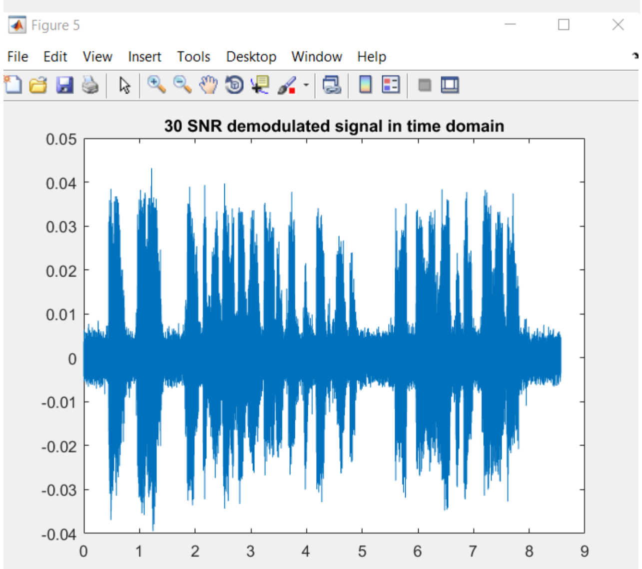


Figure 5
File Edit View Insert Tools Desktop Window Help
30 SNR demodulated signal in time domain

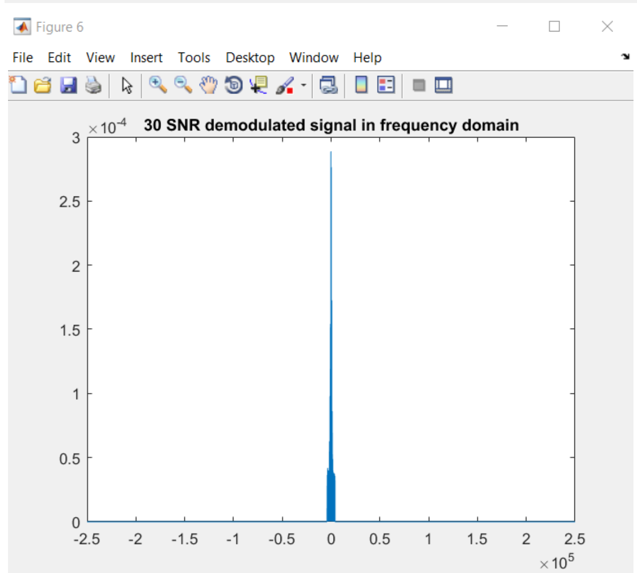
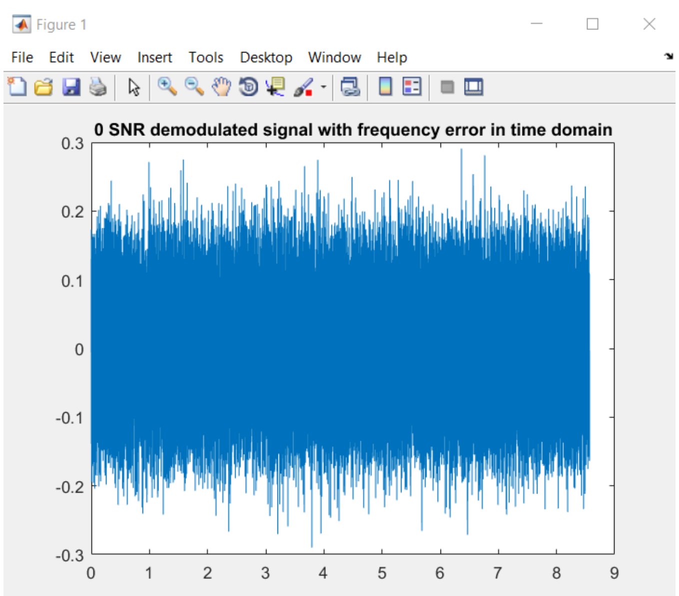
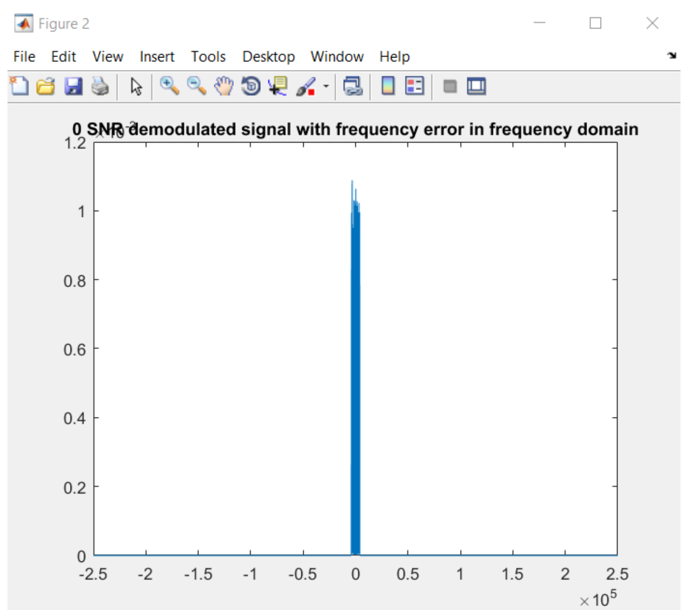


Figure 6
File Edit View Insert Tools Desktop Window Help
30 SNR demodulated signal in frequency domain
$\times 10^{-4}$
$\times 10^{5}$

```
%% 9 with frequency error

fc = 100100;
for i=1:length(SNR)

    % generate signal+noise
    dsbsc_noise = awgn(dsbsc, SNR(i));

    % demodulate using coherent detector
    demodulatedsignal = dsbsc_noise.*cos(2*pi*fc*t1);

    clear dsbsc_noise;

    % fourier transform
    demodulatedsignal_in_FD = fftshift(fft(demodulatedsignal));

    % LPF at Fm
    demodulatedsignal_in_FD(f_axis >= BW | f_axis <= -BW) = 0;

    % inverse fourier transform to get demodulated signal in time domain
    demodulatedsignal = ifft(ifftshift(demodulatedsignal_in_FD));

    % plot demodulated signal in time domain
    figure; plot(t1, demodulatedsignal); title([num2str(SNR(i)),' SNR demodulated signal with frequency error in time domain']);

    % fourier transform
    len = length(demodulatedsignal);
    spectrum_demodulated = fftshift(fft(demodulatedsignal));
    f_axis = fs/2*linspace(-1,1,len);

    % plot demodulated signal in frequency domain
    figure; plot(f_axis, abs(spectrum_demodulated) / len); title([num2str(SNR(i)),' SNR demodulated signal with frequency error in frequency domain']);
    % resample to sound the demodulated signal
    demodulatedsignal = resample(demodulatedsignal, fs/5,fs);
    sound(abs(demodulatedsignal), fs/5);

    pause(10);

end

clear demodulatedsignal;
clear demodulatedsignal_in_FD;
```





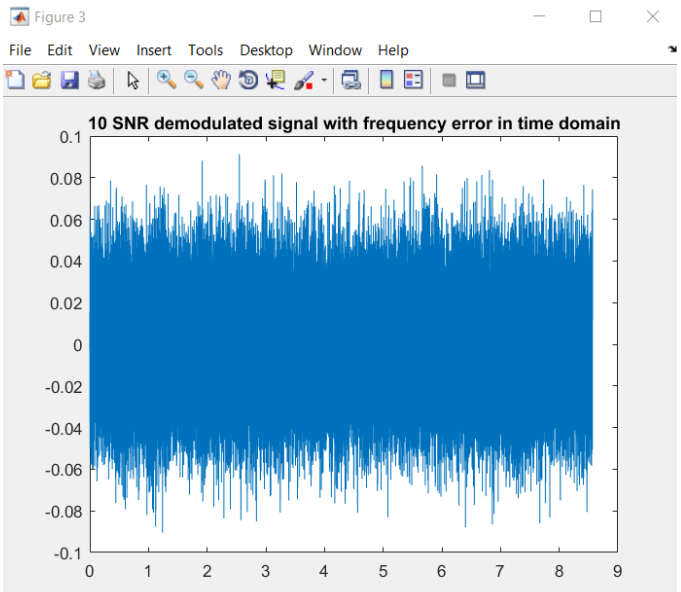


Figure 3
File Edit View Insert Tools Desktop Window Help
10 SNR demodulated signal with frequency error in time domain
0.1 0.08 0.06 0.04 0.02 0 -0.02 -0.04 -0.06 -0.08 -0.1
0 1 2 3 4 5 6 7 8 9

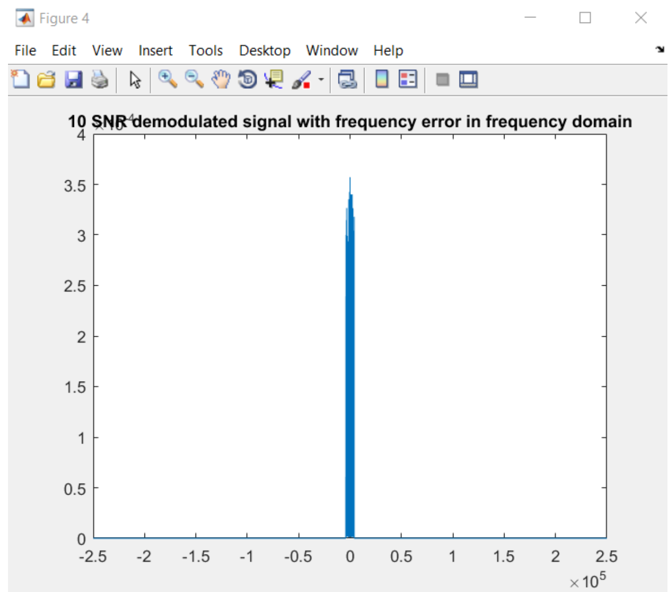


Figure 4
File Edit View Insert Tools Desktop Window Help
10 SNR demodulated signal with frequency error in frequency domain
4 3.5 3 2.5 2 1.5 1 0.5 0
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5
× 10^5

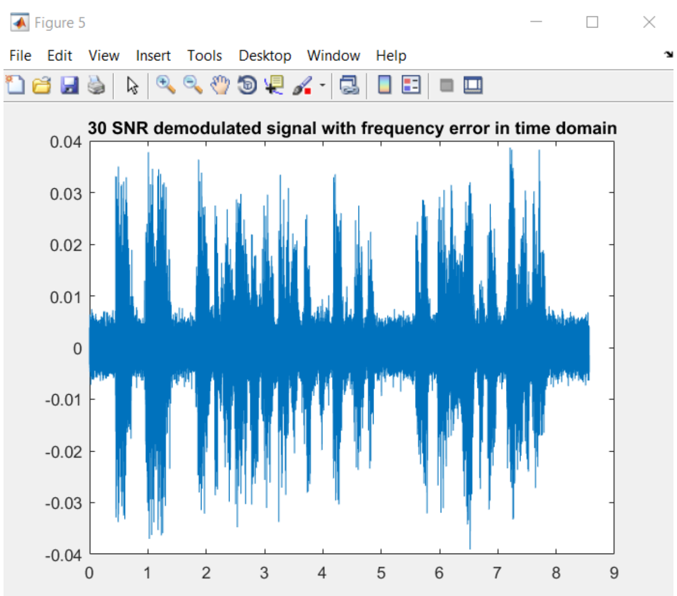


Figure 5
File Edit View Insert Tools Desktop Window Help
30 SNR demodulated signal with frequency error in time domain
0.04 0.03 0.02 0.01 0 -0.01 -0.02 -0.03 -0.04
0 1 2 3 4 5 6 7 8 9

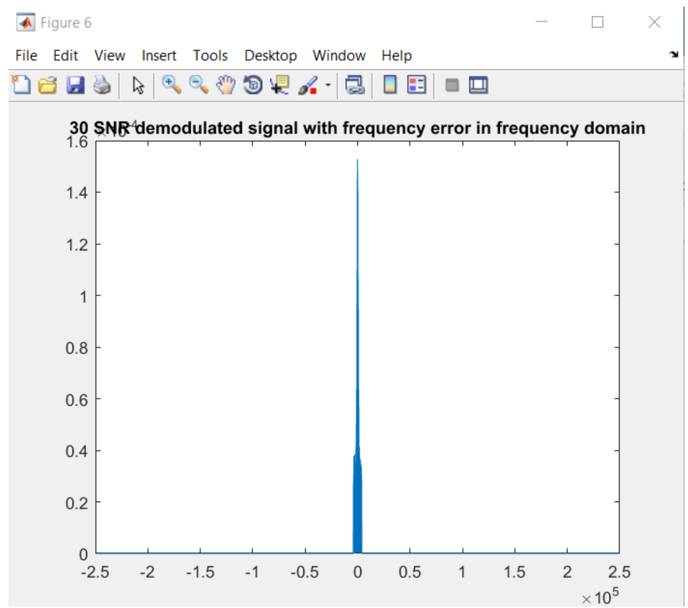
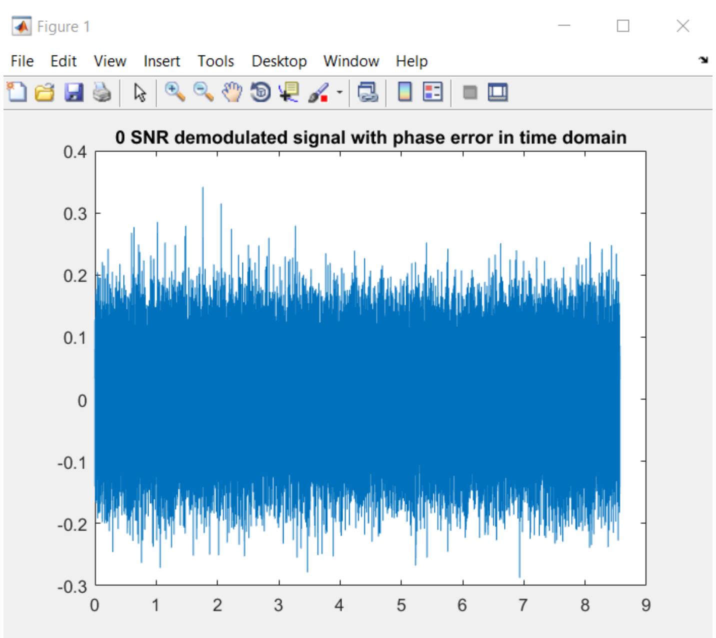
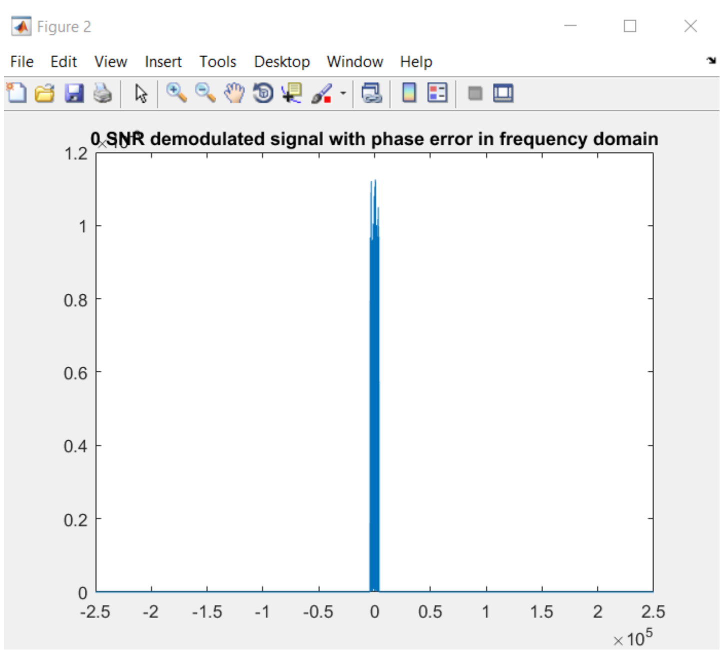


Figure 6
File Edit View Insert Tools Desktop Window Help
30 SNR demodulated signal with frequency error in frequency domain
1.6 1.4 1.2 1 0.8 0.6 0.4 0.2 0
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5
× 10^5

```
%% 10 with phase error
fc = 100000;
for i=1:length(SNR)

    % generate signal+noise
    dsbsc_noise = awgn(dsbsc, SNR(i));

    % demodulate using coherent detector
    demodulatedsignal = dsbsc_noise.*cos(2*pi*fc*t1 + pi/9);

    clear dsbsc_noise;

    % fourier transform
    demodulatedsignal_in_FD = fftshift(fft(demodulatedsignal));

    % LPF at Fm
    demodulatedsignal_in_FD(f_axis >= BW | f_axis <= -BW) = 0;

    % inverse fourier transform to get demodulated signal in time domain
    demodulatedsignal = ifft(ifftshift(demodulatedsignal_in_FD));

    % plot demodulated signal in time domain
    figure; plot(t1, demodulatedsignal); title([num2str(SNR(i)),' SNR demodulated signal with  phase error in time domain']);

    % fourier transform
    len = length(demodulatedsignal);
    spectrum_demodulated = fftshift(fft(demodulatedsignal));
    f_axis = fs/2*linspace(-1,1,len);

    % plot demodulated signal in frequency domain
    figure; plot(f_axis, abs(spectrum_demodulated) / len); title([num2str(SNR(i)),' SNR demodulated  signal with phase error in frequency domain']);

    % resample to sound the demodulated signal
    demodulatedsignal = resample(demodulatedsignal, fs/5,fs);
    sound(abs(demodulatedsignal), fs/5);
    pause(10);

end

clear demodulatedsignal;
clear demodulatedsignal_in_FD;
```





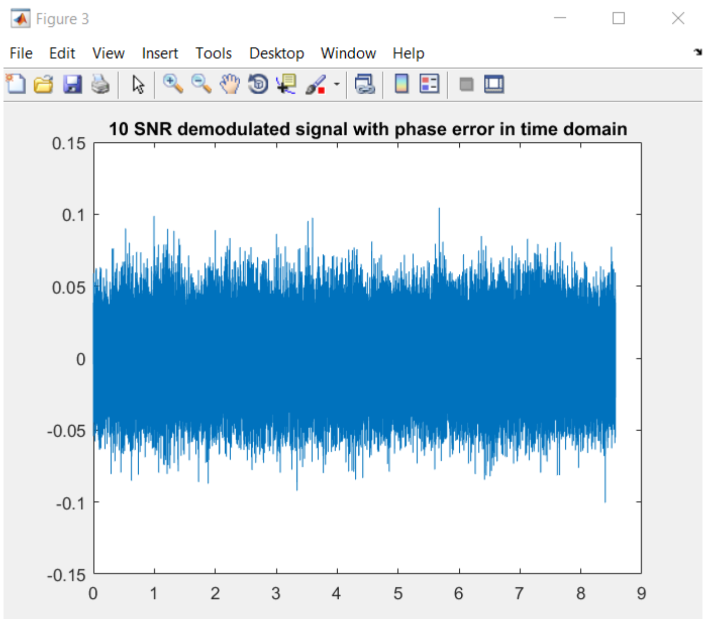


Figure 3
File Edit View Insert Tools Desktop Window Help
10 SNR demodulated signal with phase error in time domain
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
0 1 2 3 4 5 6 7 8 9

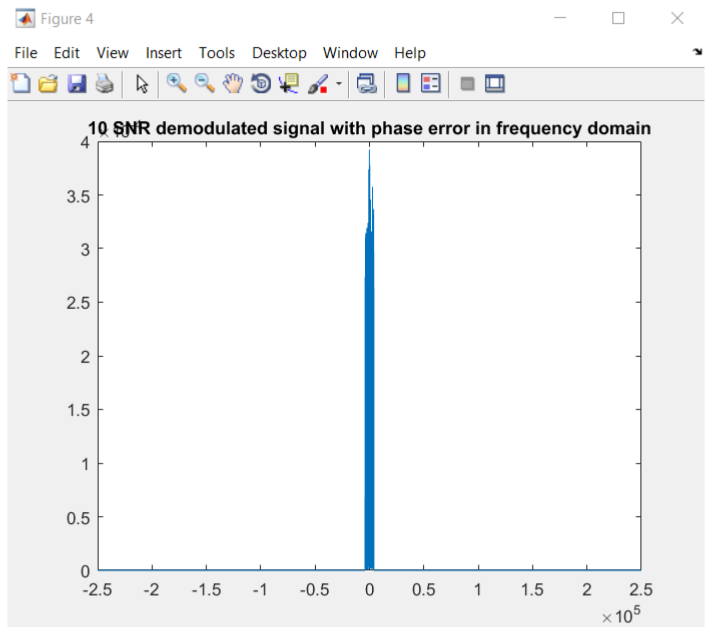


Figure 4
File Edit View Insert Tools Desktop Window Help
10 SNR demodulated signal with phase error in frequency domain
4
3.5
3
2.5
2
1.5
1
0.5
0
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5
×10^5

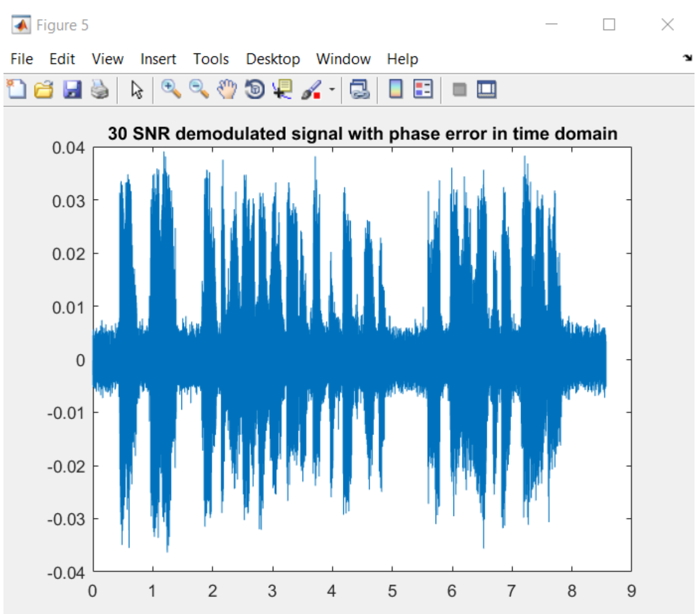


Figure 5
File Edit View Insert Tools Desktop Window Help
30 SNR demodulated signal with phase error in time domain
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
0 1 2 3 4 5 6 7 8 9

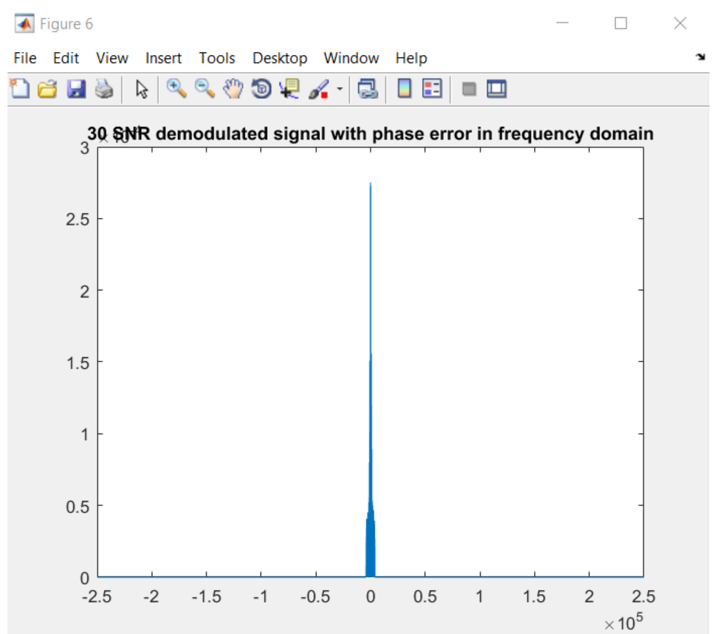
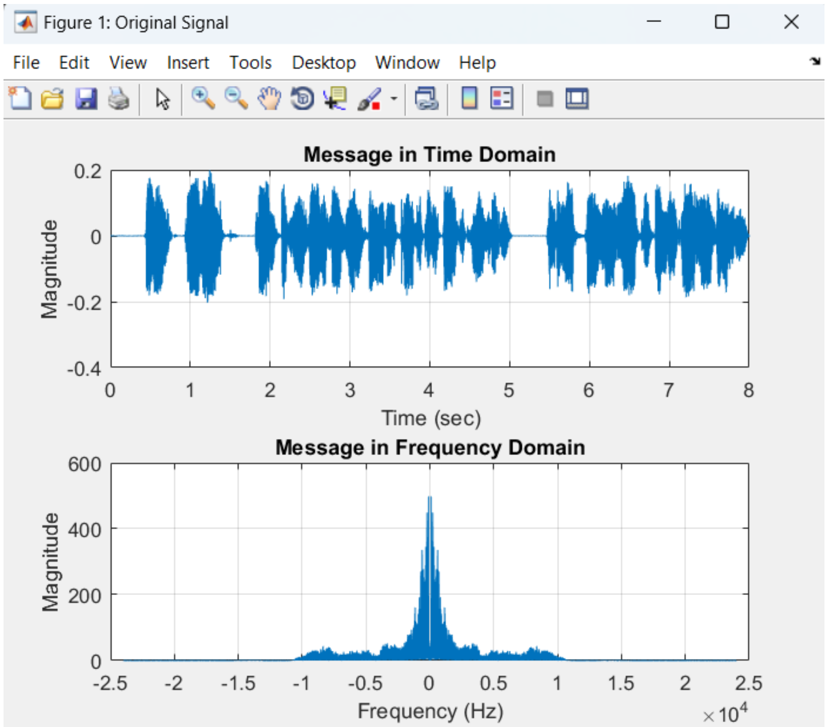


Figure 6
File Edit View Insert Tools Desktop Window Help
30 SNR demodulated signal with phase error in frequency domain
3
2.5
2
1.5
1
0.5
0
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5
×10^5

Experiment 2: Single Sideband Modulation

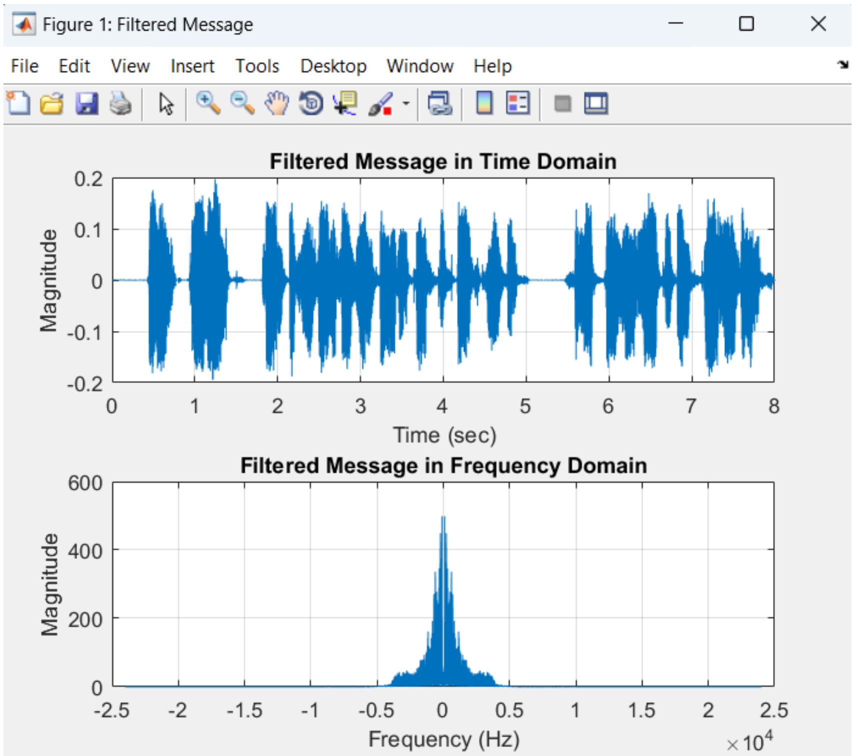
1. Read the attached audio file and plot the waveform and spectrum of the signal.

```
%% 1. Original Message

% Read the Sound File
[voice,fs]=audioread('eric.wav'); % fs = 48kHz
sec = 8;
voice = voice(1:sec*fs,1);
time=linspace(0,sec,sec*fs);

% Plot Original Message in Time Domain
figure('Name','Original Signal','NumberTitle','on');
subplot(2,1,1);
plot(time,voice)
title('Message in Time Domain');
xlabel('Time (sec)');
ylabel('Magnitude');
grid on;

% Compute the spectrum
voice_spectrum=fftshift(fft(voice));
freq=linspace(-fs/2,fs/2,length(voice_spectrum));

% Plot Original Message Spectrum
subplot(2,1,2);
plot(freq,abs(voice_spectrum))
title('Message in Frequency Domain')
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

% Playing Sound
sound (voice,fs);
pause(sec);
```



2. Use an ideal filter to remove all frequencies greater than 4KHz.
3. Obtain the filtered signal in time and frequency domain.

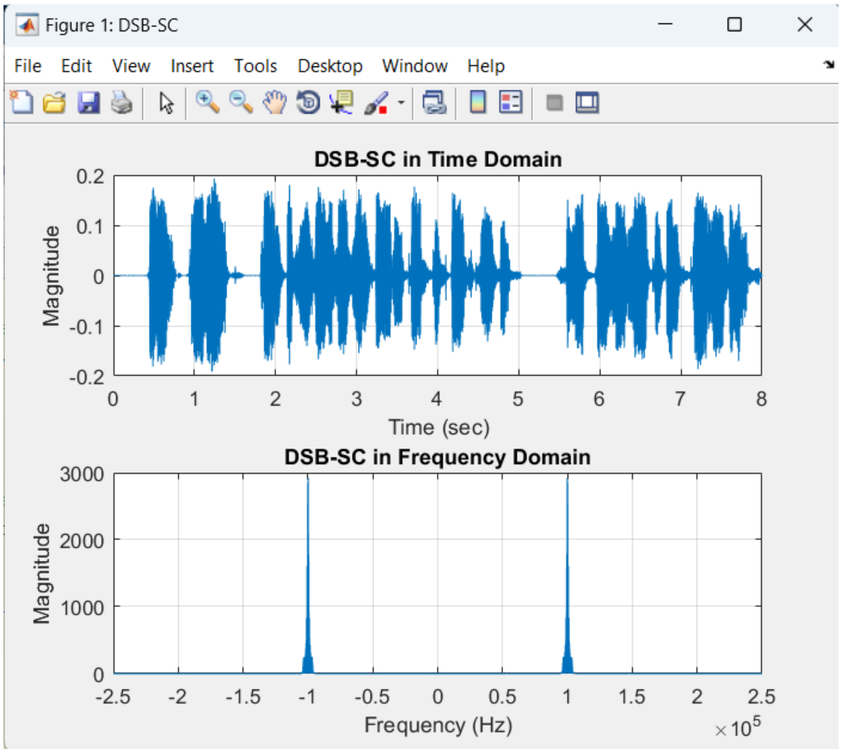
```
%% 2 & 3. Ideal Filter

cutoff_frequency = 4000;
filter_order = 50;

% Signal Filtering
fltr=designfilt('lowpassfir','FilterOrder',filter_order,'CutoffFrequency',cutoff_frequency, 'SampleRate',fs);
filtered_voice=filter(fltr,voice);

% Plot Filtered Message in Time Domain
figure('Name','Filtered Message','NumberTitle','on');
subplot(2,1,1);
plot(time,filtered_voice)
title('Filtered Message in Time Domain');
xlabel('Time (sec)');
ylabel('Magnitude');
grid on;

% Plot Filtered Message Spectrum
subplot(2,1,2);
plot(linspace(-fs/2,fs/2,length(filtered_voice)),abs(fftshift(fft(filtered_voice))));
title('Filtered Message in Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

% Playing Sound
sound (filtered_voice,fs);
pause(sec);
```



Observation: The sound of the message has a small distortion.

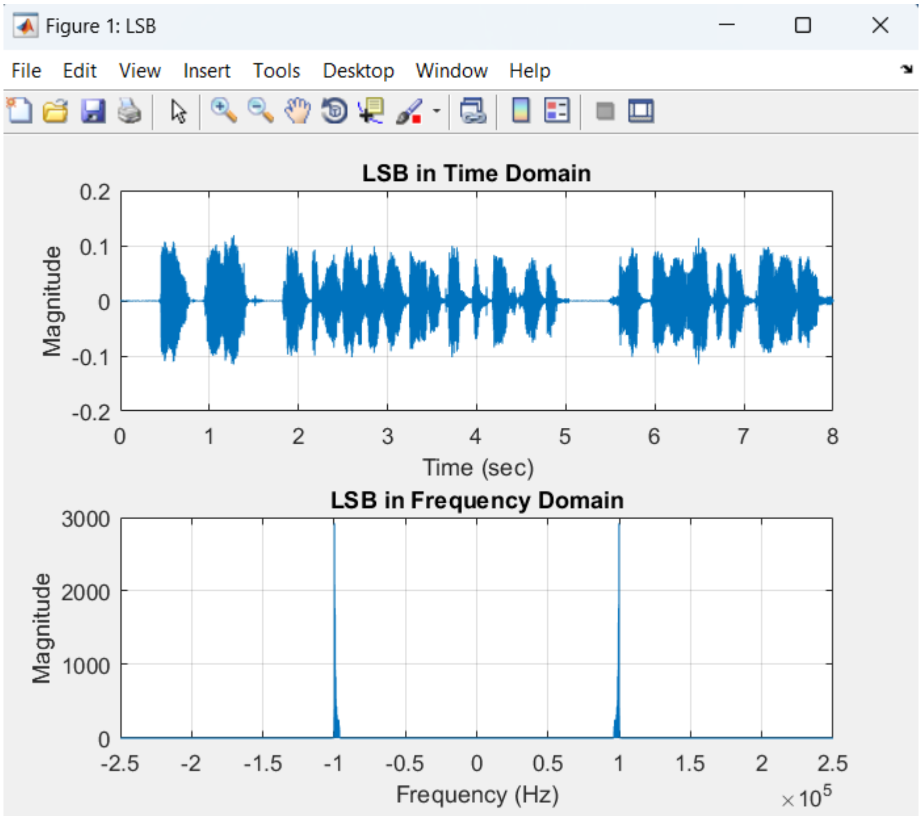
4. Generate a DSB-SC modulated signal and plot its waveform and spectrum.

```
%% 4.DSB-SC

% Upsample
fc = 100000; % Carier Frequency
fs_new = 5*fc ; % Sampling Frequency (new fs =500k)
resampled_voice=resample(filtered_voice,fs_new,fs); % Resmaple by 500k/48k
time=linspace(0,sec,sec*fs_new);

% Message Modulation
Ac = 1;
carrier = Ac .* cos(2*pi*fc*time'); % Carrier Signal
suprsd_carrier = carrier.*resampled_voice; % DSB-SC

% Plot DSB-SC in Time Domain
figure('Name','DSB-SC','NumberTitle','on');
subplot(2,1,1);
plot(time,suprsd_carrier);
title('DSB-SC in Time Domain');
xlabel('Time (sec)');
ylabel('Magnitude');
grid on;

% Plot DSB-SC Spectrum
subplot(2,1,2);
plot(linspace(-fs_new/2,fs_new/2,length(suprsd_carrier)),abs(fftshift(fft(suprsd_carrier))))
title('DSB-SC in Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```



5. Obtain the SSB by filtering out the USB of the DSB-SC using an ideal LPF and plot the waveform and spectrum.

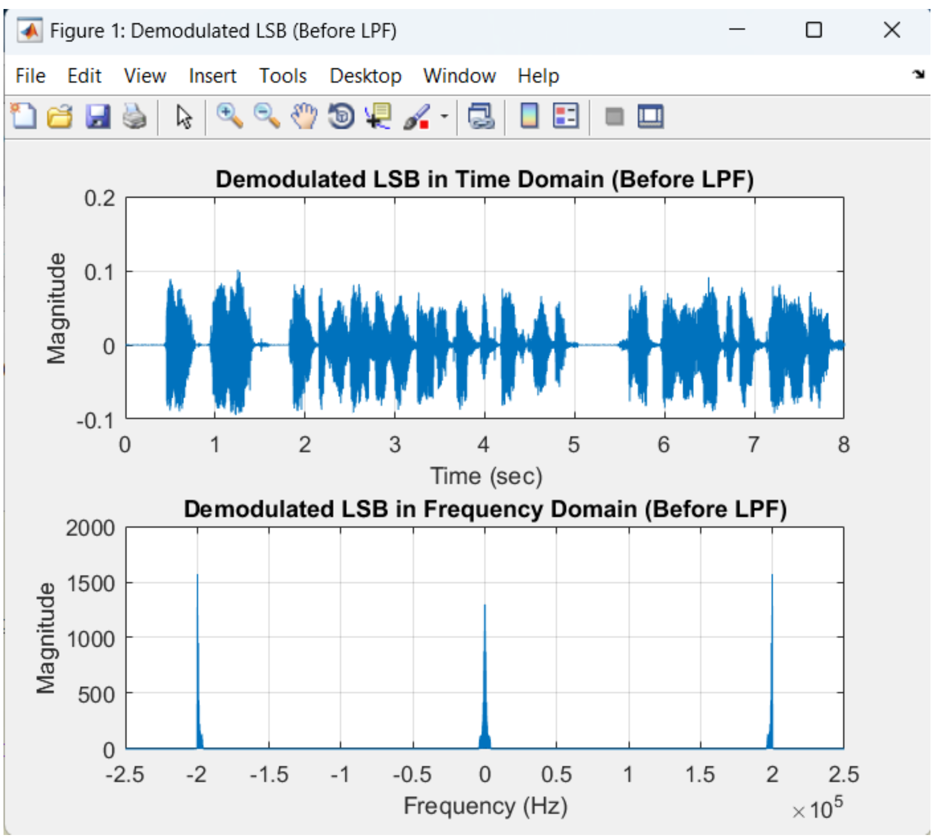
```
%% 5. SSB-SC

% Design an ideal bandpass filter
f_passband = [fc - 4000 , fc]; % Passband frequencies
filter_order = 10000; % Filter order
filter_coeffs_mod = fir1(filter_order, f_passband / (fs_new/2), 'bandpass', hamming(filter_order + 1));

% Apply the filter to the DSB-SC signal
LSB = filter(filter_coeffs_mod, 1, suprsd_carrier);

% Plot LSB in Time Domain
figure('Name','LSB','NumberTitle','on');
subplot(2,1,1);
plot(time',LSB)
title('LSB in Time Domain');
xlabel('Time (sec)');
ylabel('Magnitude');
grid on;

% Plot LSB Spectrum
subplot(2,1,2);
plot(linspace(-fs_new/2,fs_new/2,length(LSB)),abs(fftshift(fft(LSB))))
title('LSB in Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

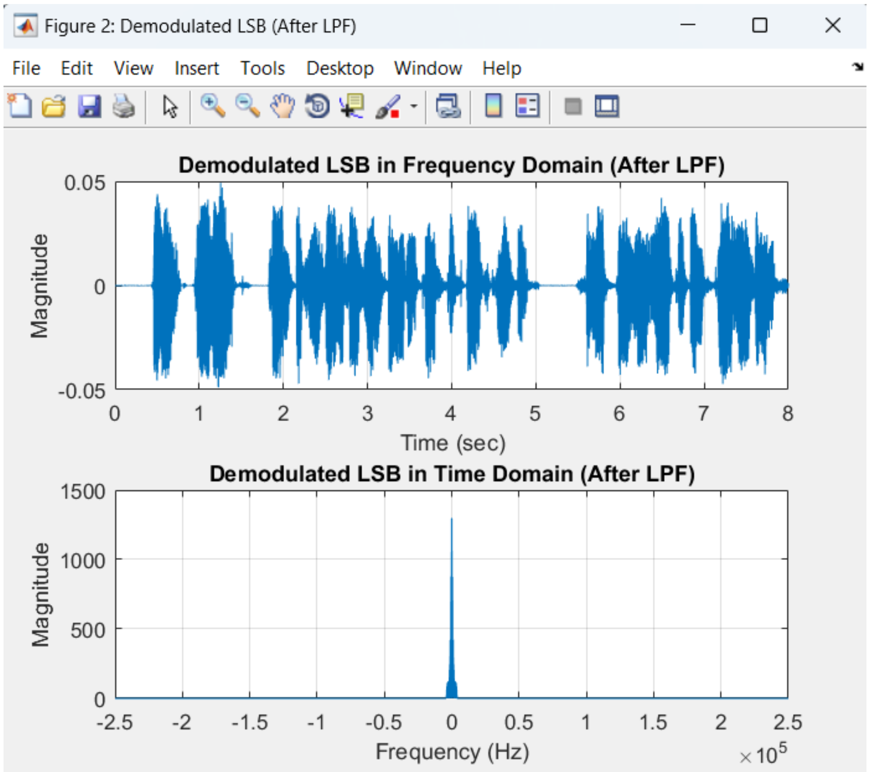


6. Use coherent detection with no noise interference to get the demodulated signal and plot its waveform and spectrum.

```
%% 6. Coherent Detection

% Signal Demodulation
t = linspace(0, length(LSB)/fs_new, length(LSB));
demodulated_signal = LSB .* carrier;

% Plot Demoodulated LSB (before LPF) in Time Domain
figure('Name','Demodulated LSB (Before LPF)','NumberTitle','on');
subplot(2,1,1);
plot(t,demodulated_signal)
title('Demodulated LSB in Time Domain (Before LPF)');
xlabel('Time (sec)');
ylabel('Magnitude');
grid on;

% Plot Demoodulated LSB (before LPF) in Frequency Domain
subplot(2,1,2);
plot(linspace(-fs_new/2,fs_new/2,length(demodulated_signal)),abs(fftshift(fft(demodulated_signal))))
title('Demodulated LSB in Frequency Domain (Before LPF)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

% Design an ideal LPF
f_cutoff = fs; % Cutoff frequency in Hz
filter_order = 1000; % Filter order
filter_coeffs = fir1(filter_order, f_cutoff / (fs_new/2), 'low');
```

```
% Apply LPF on the demodulated signal to remove interferences
filtered_demodulated_signal = filter(filter_coeffs, 1, demodulated_signal);

filtered_t = linspace(0, length(filtered_demodulated_signal)/fs_new, length(filtered_demodulated_signal));
freq_axis_filtered = linspace(-fs_new/2, fs_new/2, length(filtered_demodulated_signal));

% Plot Demoodulated LSB (After LPF) in Time Domain
figure('Name','Demodulated LSB (After LPF)','NumberTitle','on');
subplot(2,1,1);
plot(filtered_t, filtered_demodulated_signal);
title('Demodulated LSB in Frequency Domain (After LPF)');
xlabel('Time (sec)');
ylabel('Magnitude');
grid on;

% Plot Demoodulated LSB (After LPF) in Frequency Domain
subplot(2,1,2);
plot(freq_axis_filtered, abs(fftshift(fft(filtered_demodulated_signal))));
title('Demodulated LSB in Time Domain (After LPF)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

% Playing Sound
r_demodulated_signal=resample(filtered_demodulated_signal,fs,fs_new);
sound(r_demodulated_signal,fs)
pause(sec);
```





7. Repeat steps 5 and 6 using a practical Butterworth filter.

```
%% 7. Butterworth Filter

% Buttterworth Filter to get LSB
[b,a]= butter(4,[((fc-4000)/(fs_new/2)) (fc/(fs_new/2))],'bandpass');
butter_LSB = filter(b, a, suprsd_carrier);

% Plot Butter LSB in Time Domain
figure('Name',' LSB (Butterworth)','NumberTitle','on');
subplot(2,1,1);
plot(time,butter_LSB)
title(' LSB (Butterworth) in Time Domain');
xlabel('Time (sec)');
ylabel('Magnitude');
grid on;

% Plot Butter LSB in Frequency Domain
subplot(2,1,2);
plot(linspace(-fs_new/2,fs_new/2,length(butter_LSB)),abs(fftshift(fft(butter_LSB))))
title(' LSB (Butterworth) in Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

% Coherent Detection
butter_t = linspace(0, length(butter_LSB)/fs_new, length(butter_LSB));
butter_demodulated_signal = butter_LSB .* carrier;
```

```
% Plot Demoodulated butter_LSB (before LPF) in Time Domain
figure('Name','Demodulated LSB (Butterworth) (Before LPF)','NumberTitle','on');
subplot(2,1,1);
plot(butter_t,butter_demodulated_signal)
title('Demodulated LSB (Butterworth) in Time Domain (Before LPF)');
xlabel('Time (sec)');
ylabel('Magnitude');
grid on;

% Plot Demoodulated butter_LSB (before LPF) in Frequency Domain
subplot(2,1,2);
plot(linspace(-fs_new/2,fs_new/2,length(butter_demodulated_signal)),abs(fftshift(fft(butter_demodulated_signal))))
title('Demodulated LSB (Butterworth) in Frequency Domain (Before LPF)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

```
% Apply LPF (in no.6) on the demodulated signal to remove interferences
filtered_butter_demodulated_signal = filter(filter_coeffs, 1, butter_demodulated_signal);

filtered_t = linspace(0, length(filtered_butter_demodulated_signal)/fs_new, length(filtered_butter_demodulated_signal));
freq_axis_filtered = linspace(-fs_new/2, fs_new/2, length(filtered_butter_demodulated_signal));

% Plot Demoodulated LSB (After LPF) in Time Domain
figure('Name','Demodulated LSB (Butterworth) (After LPF)','NumberTitle','on');
subplot(2,1,1);
plot(filtered_t, filtered_butter_demodulated_signal);
title('Demodulated LSB (Butterworth) in Frequency Domain (After LPF)');
xlabel('Time (sec)');
ylabel('Magnitude');
grid on;

% Plot Demoodulated LSB (After LPF) in Frequency Domain
subplot(2,1,2);
plot(freq_axis_filtered, abs(fftshift(fft(filtered_butter_demodulated_signal))));
title('Demodulated LSB (Butterworth) in Time Domain (After LPF)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

% Playing Sound
r_butter_demodulated_signal=resample(filtered_butter_demodulated_signal,fs,fs_new);
sound(r_butter_demodulated_signal,fs)
pause(sec);
```

⇨ Observations: After using the practical Butterworth filter, it appears that the LSB still have a little part of the USB.

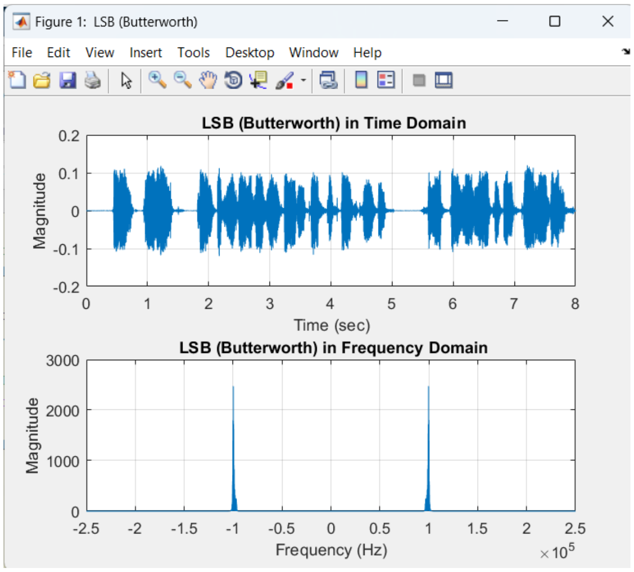


Figure 1: LSB (Butterworth)
File Edit View Insert Tools Desktop Window Help
LSB (Butterworth) in Time Domain
Magnitude
Time (sec)
LSB (Butterworth) in Frequency Domain
Magnitude
Frequency (Hz)
$\times 10^5$

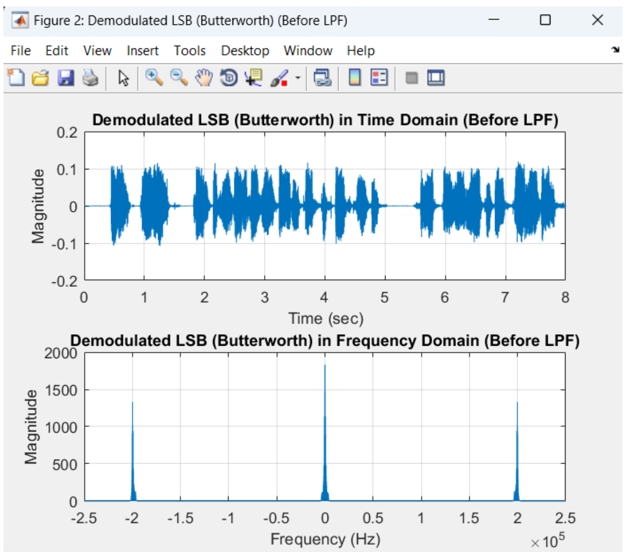


Figure 2: Demodulated LSB (Butterworth) (Before LPF)
File Edit View Insert Tools Desktop Window Help
Demodulated LSB (Butterworth) in Time Domain (Before LPF)
Magnitude
Time (sec)
Demodulated LSB (Butterworth) in Frequency Domain (Before LPF)
Magnitude
Frequency (Hz)
$\times 10^5$

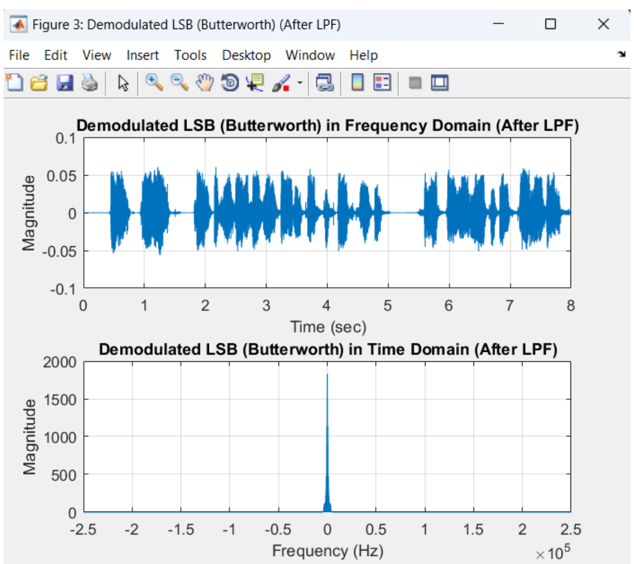


Figure 3: Demodulated LSB (Butterworth) (After LPF)
File Edit View Insert Tools Desktop Window Help
Demodulated LSB (Butterworth) in Frequency Domain (After LPF)
Magnitude
Time (sec)
Demodulated LSB (Butterworth) in Time Domain (After LPF)
Magnitude
Frequency (Hz)
$\times 10^5$

8. For the ideal filter case get the demodulated message again but when white gaussian noise is added to SSB-SC with SNR = 0, 10, 30 and plot its waveform and spectrum in each case.

```
%% 8. Ideal Filter Case with Noise

% SNR values to simulate
snr_values = [0, 10, 30];

% Plot demodulated waveform and spectrum for each SNR
for snr_value = snr_values
    % Add noise to the recieved LSB
    noisy_LSB = awgn(LSB, snr_value);

    % Time vector of noisy signal
    t_noise = linspace(0, length(noisy_LSB)/fs_new, length(noisy_LSB));
    local_carrier = Ac .* cos(2*pi*fc*t_noise'); % Carrier Signal

    % Coherent detection
    noisy_demodulated_signal = noisy_LSB .* local_carrier;

    % Apply LPF (in no.6) on the demodulated signal to remove interferences
    filtered_noisy_demodulated_signal = filter(filter_coeffs, 1, noisy_demodulated_signal);

    % Plot received demodulated noisy signal in Time Domain
    figure('Name', ['Demodulated Noisy Signal (SNR = ' num2str(snr_value) ' dB)'], 'NumberTitle', 'on');
    subplot(2, 1, 1);
    plot(t_noise, filtered_noisy_demodulated_signal);
    title(['Demodulated Noisy Signal in Time Domain (SNR = ' num2str(snr_value) ' dB)']);
    xlabel('Time (sec)');
    ylabel('Magnitude');
    grid on;

    % Plot received demodulated noisy signal in Frequency Domain
    subplot(2, 1, 2);
    freq_axis = linspace(-fs_new/2, fs_new/2, length(filtered_noisy_demodulated_signal));
    plot(freq_axis, abs(fftshift(fft(filtered_noisy_demodulated_signal))));
    title(['Demodulated Noisy Signal in Frequency Domain (SNR = ' num2str(snr_value) ' dB)']);
    xlabel('Frequency (Hz)');
    ylabel('Magnitude');
    grid on;

    % Play back the demodulated signal
    r_noisy_demodulated_signal=resample(filtered_noisy_demodulated_signal,fs,fs_new);
    sound(r_noisy_demodulated_signal, fs);
    pause(sec);

end
```

⇨ Observations: As the SNR value increases, the sound of the message becomes purer with less interferences.

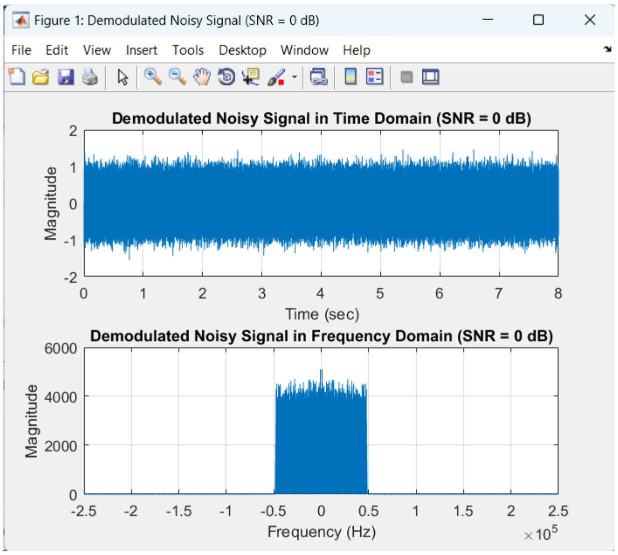


Figure 1: Demodulated Noisy Signal (SNR = 0 dB)
File Edit View Insert Tools Desktop Window Help
Demodulated Noisy Signal in Time Domain (SNR = 0 dB)
Magnitude
Time (sec)
Demodulated Noisy Signal in Frequency Domain (SNR = 0 dB)
Magnitude
Frequency (Hz)
×10^5

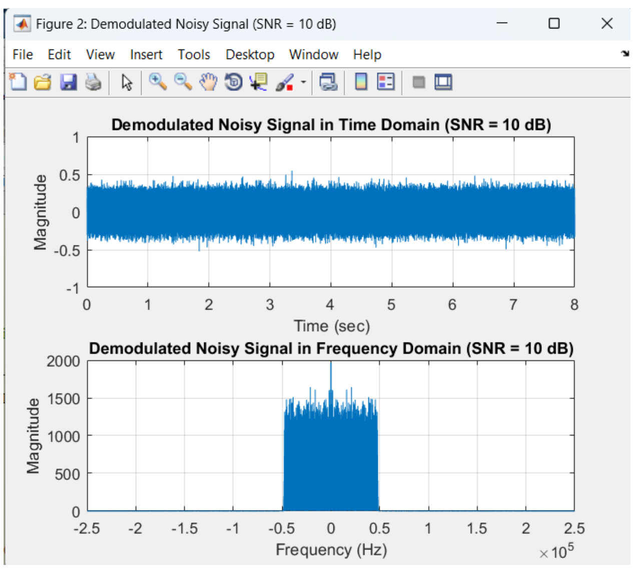


Figure 2: Demodulated Noisy Signal (SNR = 10 dB)
File Edit View Insert Tools Desktop Window Help
Demodulated Noisy Signal in Time Domain (SNR = 10 dB)
Magnitude
Time (sec)
Demodulated Noisy Signal in Frequency Domain (SNR = 10 dB)
Magnitude
Frequency (Hz)
×10^5

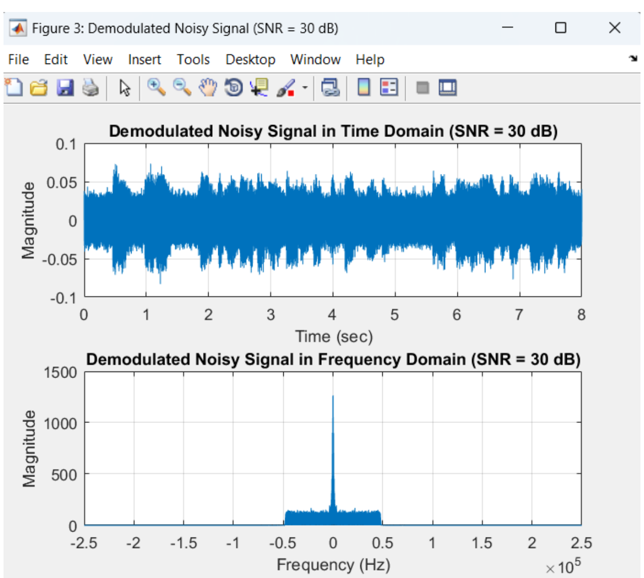


Figure 3: Demodulated Noisy Signal (SNR = 30 dB)
File Edit View Insert Tools Desktop Window Help
Demodulated Noisy Signal in Time Domain (SNR = 30 dB)
Magnitude
Time (sec)
Demodulated Noisy Signal in Frequency Domain (SNR = 30 dB)
Magnitude
Frequency (Hz)
×10^5

9. For the ideal filter case, generate an SSB-TC, use envelope detector to demodulate the message and plot its waveform and spectrum.

```
%% 9. SSB-TC

% DC_bias
dc_bias = 2 .* max(resampled_voice);
trnsm_carrier=(dc_bias + resampled_voice) .* carrier; % DSB-TC

% Apply the bandpass filter (in no. 5) to the DSB-TC signal
LSB_tc = filter(filter_coeffs_mod, 1, trnsm_carrier); % SSB-TC

% Plot LSB in Time Domain
figure('Name','LSB-TC','NumberTitle','on');
subplot(2,1,1);
plot(time',LSB_tc)
title('LSB-TC in Time Domain');
xlabel('Time (sec)');
ylabel('Magnitude');
grid on;

% Plot LSB Spectrum
subplot(2,1,2);
plot(linspace(-fs_new/2,fs_new/2,length(LSB_tc)),abs(fftshift(fft(LSB_tc))))
title('LSB-TC in Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

% Envelope Detector
envelope = abs(hilbert(LSB_tc));

% Plot the Demodulated Signal in Time Domain
figure('Name', 'Envelope (SSB-TC)', 'NumberTitle', 'on');
subplot(2,1,1);
plot(time, envelope);
title('Envelope (SSB-TC) in Time Domain');

% Plot the Demodulated Signal in Frequency Domain
subplot(2,1,2);
plot(linspace(-fs_new/2,fs_new/2,length(envelope)),abs(fftshift(fft(envelope))))
title('Envelop (SSB-TC) in Frequency Domain');

% Playing Sound
envelope = resample(envelope,fs,fs_new);
sound(envelope,fs)
pause(sec);
```

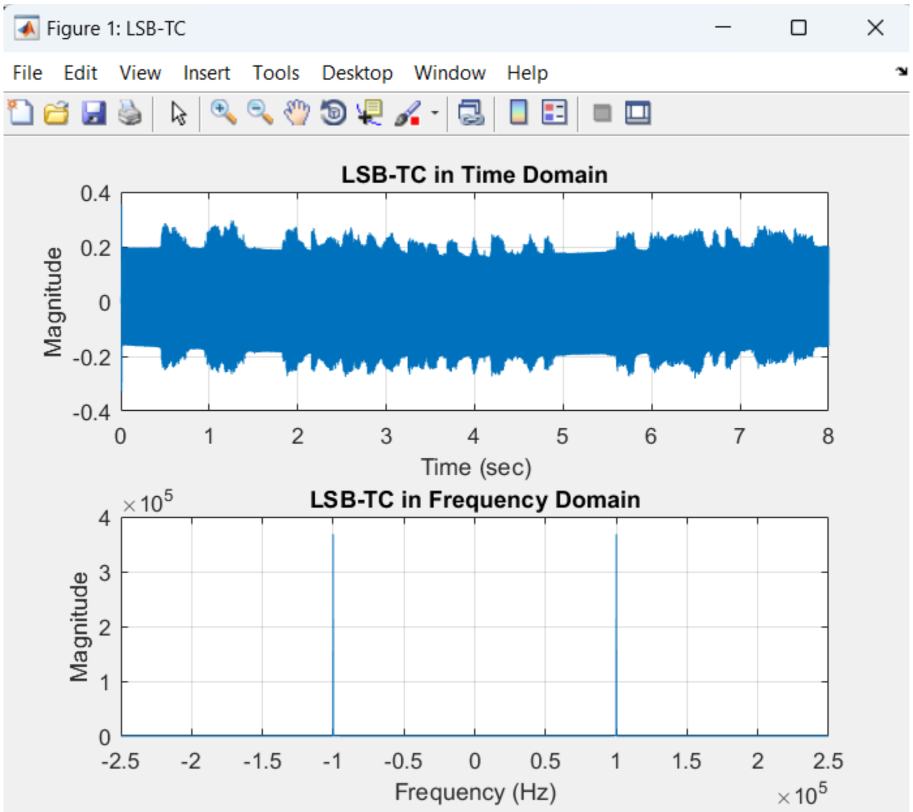


Figure 1: LSB-TC
File Edit View Insert Tools Desktop Window Help
LSB-TC in Time Domain
Magnitude
Time (sec)
LSB-TC in Frequency Domain
Magnitude
Frequency (Hz)
$\times 10^5$

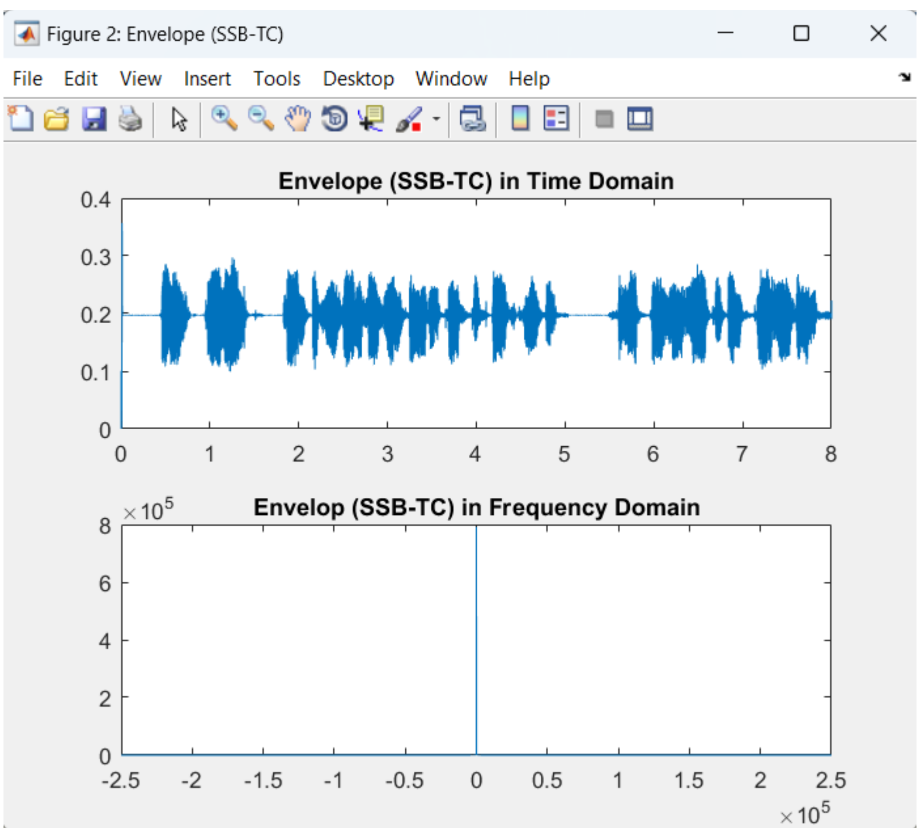
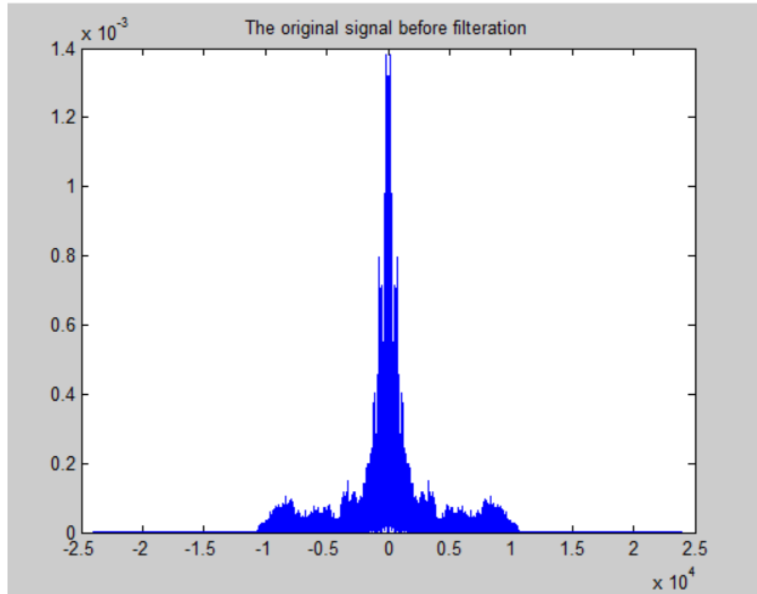


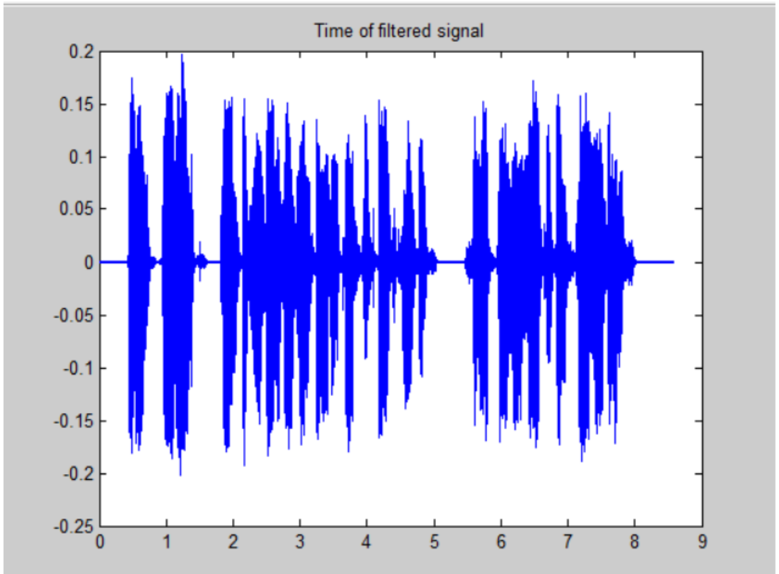
Figure 2: Envelope (SSB-TC)
File Edit View Insert Tools Desktop Window Help
Envelope (SSB-TC) in Time Domain
Envelop (SSB-TC) in Frequency Domain
$\times 10^5$

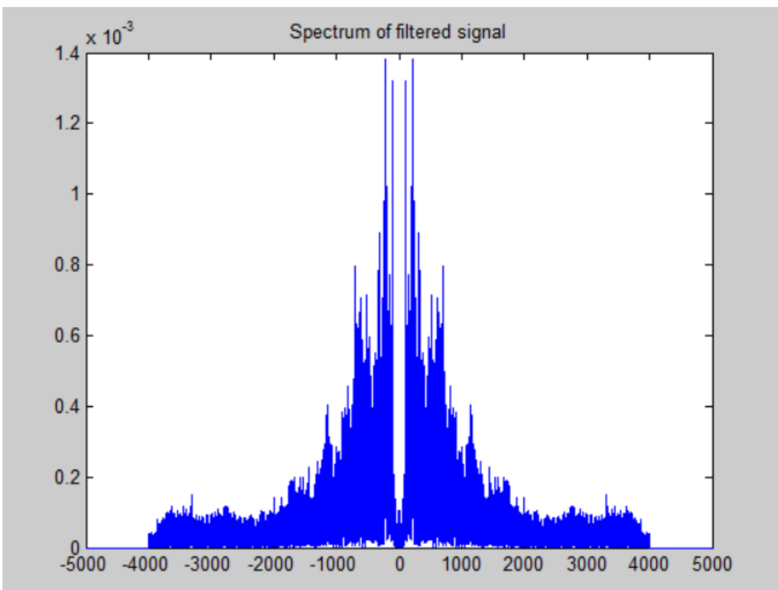
Experiment 3: Frequency Modulation

```
%Read audio file
[S, Fs] = audioread('eric.wav');
%sound(abs(S),Fs);

%Find the spectrum
%Fourier transform
L = length(S);
F = fftshift(fft(S));
f = Fs/2*linspace(-1,1,L);

%Plotting the spectrum
figure; plot(f,abs(F)/L);
title('The original signal before
filteration'); %abs(F)/L ->
spectrum magnitude
```

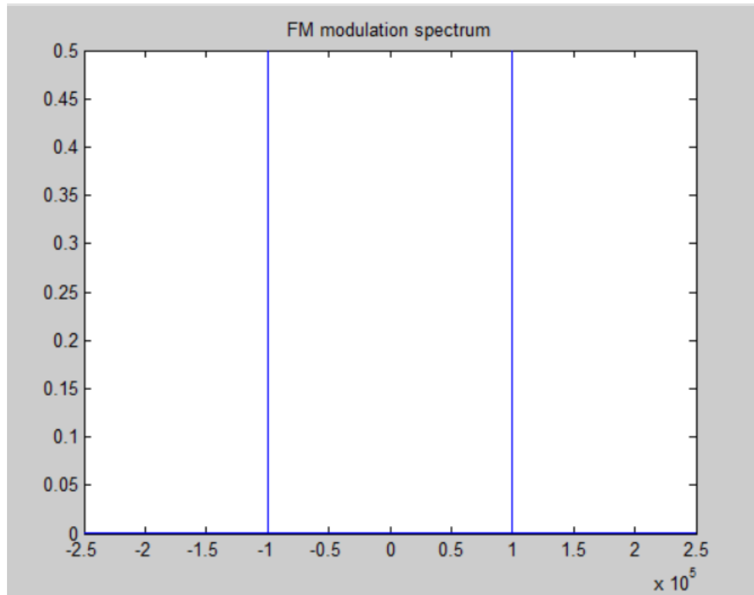



```
%Filtering and plot
cutoff_frequency = 4000;
F(f>=cutoff_frequency|f<=-cutoff_frequency) = 0;
figure; plot(f,abs(F)/L); xlim([-5000,5000]); title('Spectrum of filtered
signal');

%In time domain
X = ifft(ifftshift(F));
%sound(abs(X),Fs);

%calculate time
tStart = 0;
tEnd = tStart + length(X) / Fs;
t=linspace(tStart,tEnd,length(X));
t=t';
figure; plot(t,X); title('Time of filtered signal');
```


Time of filtered signal
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
1
2
3
4
5
6
7
8
9

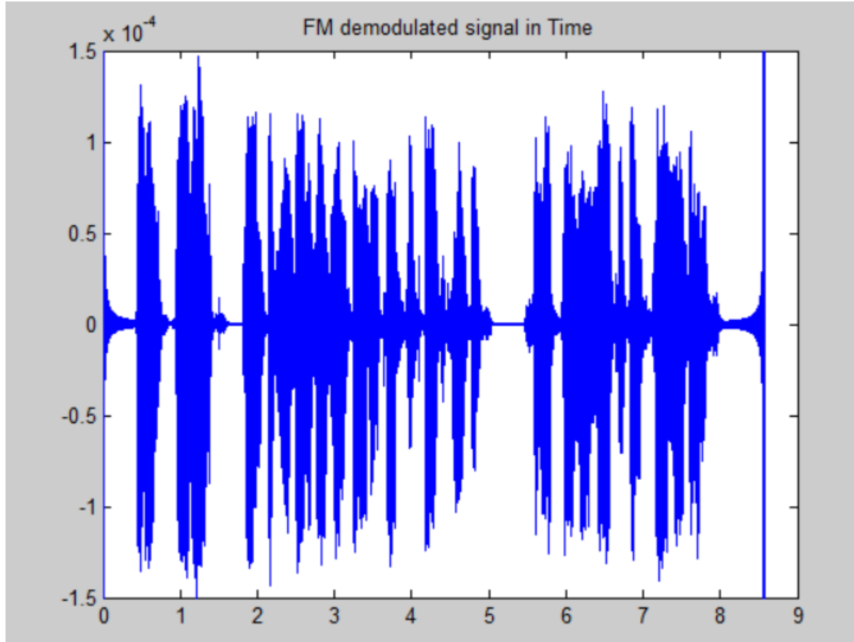

Spectrum of filtered signal
x 10^-3
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-5000
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000
5000

```
%constants
fc=100000;
Ac=1;
omega_c=2*pi*fc;
%Get maximum deviation of the integrated(X) signal from its mean.
max_dev=max(abs(cumsum(X)));
%normalization of the deviation
norm=2*pi*max_dev./Fs;
k_fm=0.2/norm;

%resampling
X=resample(X,5*fc,Fs);
Fs=5*fc;

tStart = 0;
tEnd = tStart + length(X) / Fs;
t=linspace(tStart,tEnd,length(X));
t=t';

%FM modulation
X=Ac*cos(omega_c*t +
2*pi*k_fm*cumsum(X)./Fs);

%Fourier transform
L = length(X);
F = fftshift(fft(X));
f = Fs/2*linspace(-1,1,L);
```




```
%Plotting the spectrum
figure; plot(f,abs(F)/L); title('FM modulation spectrum');

%descriminator
dy=diff(X); %calculating difference between X samples
dy=[0;dy]; %Making sure to let the length be same as the original X

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

%plotting for FM demod. signal in time
figure; plot(t,envelopeFM); title('FM demodulated signal in Time');
ylim([-0.00015 0.00015]);
```

```
% resample to hear the
signal
envelopeFM =
resample(envelopeFM, Fs/5,
Fs);
%sound(100.*abs(envelopeFM),
Fs/5);
sound(1000.*abs(envelopeFM),
Fs/5);
```



- ⇨ The resulting spectrum is the same as DSB-TC.
- ⇨ The condition we needed to achieve NBFM is beta << 1.

Conclusion:

Double Sideband modulation is the easiest and most direct type of analog modulation.

In SSB modulation, the bandwidth required for bandpass transmission is equal to the bandwidth of that of the baseband, so it is more efficient than DSB modulation.

Frequency modulation's SNR is better than SNR of the Amplitude modulation.

Envelope detector is a good method of detection for DSB-TC but not for DSB-SC.

When the SNR increases the quality of the sound increases.

Coherent detection is suitable for any type of modulation.