Alexandria University
Faculty of Engineering
Department of Electrical Engineering
Communications & Electronics Program

Third Year 2021

Course: Analog Communication

FM Assignment Report

Submitted From: Ismail Mohamed Abdelgeleel Farahat
Sec: 2
ID: 52
Date: 10/06/2021

Content:

Obtained Results and comments

- Condition that will be needed to achieve the NBFM

$$\beta \ll (less\ than\ 1)$$

$$\because \Delta f_{max} = \frac{1}{2\pi}\frac{d\phi_t}{dt}|_{max}$$

$$\therefore\ \beta = \frac{\Delta f_{max}}{f_m} = \frac{1}{2\pi f_m}\frac{d\phi_t}{dt}|_{max}$$

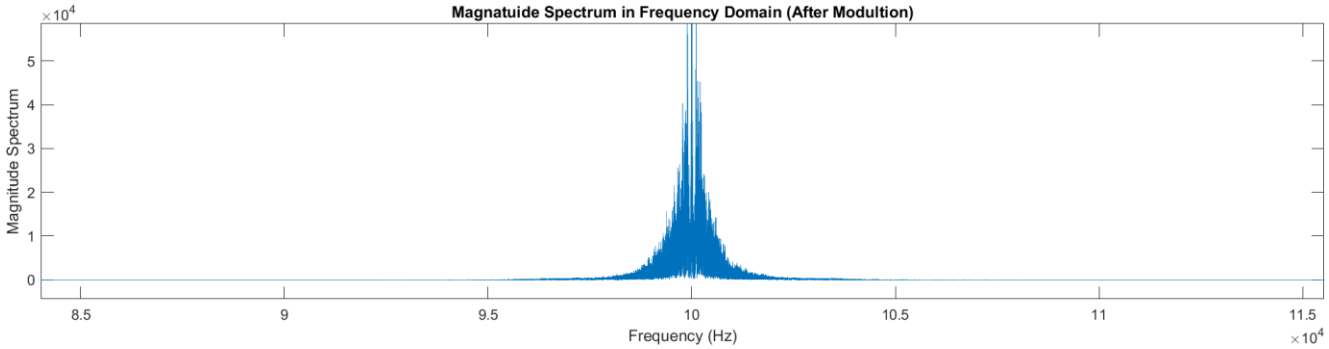
$$\because \phi_t = k_f \int_{-\infty}^{t} m(t)\,dt$$

$$As\ \beta \ll \xrightarrow{yields} \phi_t \ll \xrightarrow{yields} k_f \ll$$

$$\therefore k_f \ll 1, \qquad the\ condition\ for\ NBFM\ modulation$$

As k_f decreases more and more, the modulated signal will be close to be NBFM signal.

- The comment on the frequency spectrum of the modulated signal



Important Note: This figure is **NOT** the entire frequency spectrum, is just zoomed figure for the right part of the spectrum which represents the shifted signal with impulse to get the BW of the signal.

From the figure, we can find that the band width of the modulated signal is equal to almost

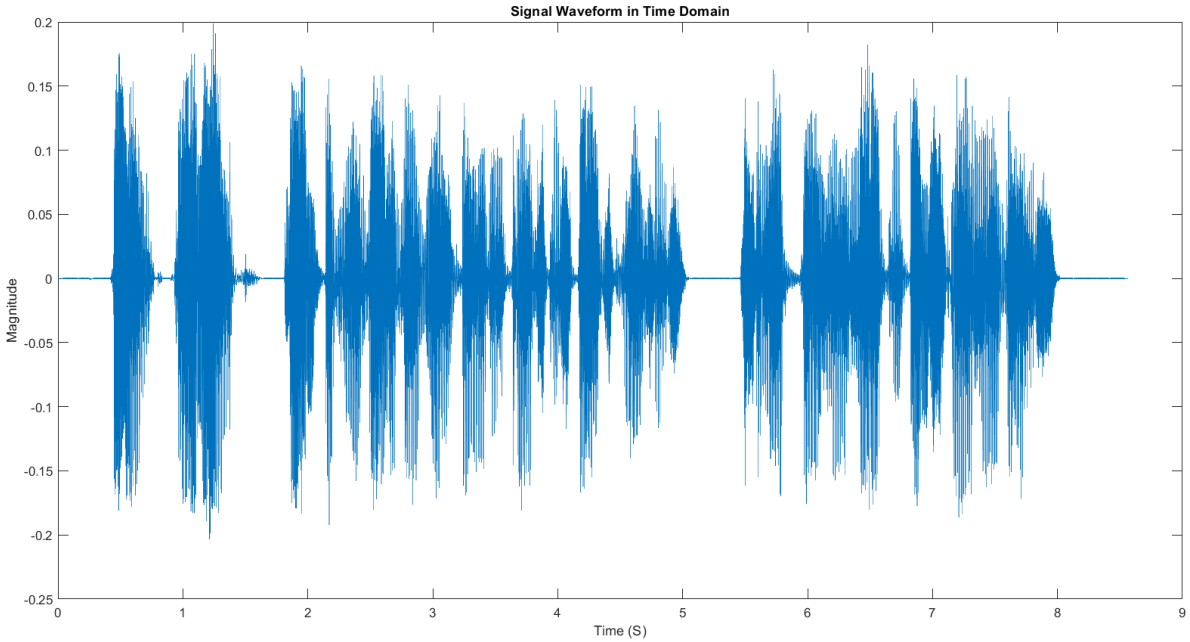
BW = 10.4 -9.6 = 0.8 * 10^4 Hz = **8 k Hz ,** $\mathbf{f_m}$ **= 4 K Hz**

we can conclude that the bandwidth of the signal is almost equal to **2** $\mathbf{f_m}$ which proves that the modulated signal is **NBFM signal**. As we know

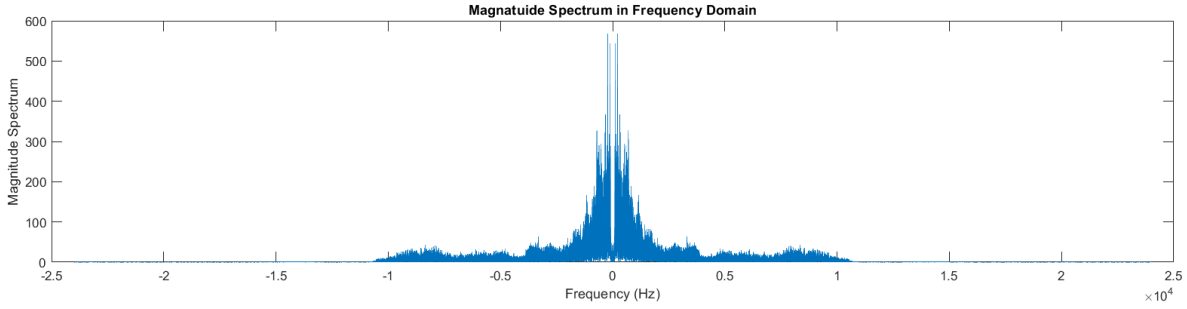
$$BW_{NBFM} = 2f_m(\beta + 1) \cong 2f_m, \qquad when\ \beta \ll 1$$

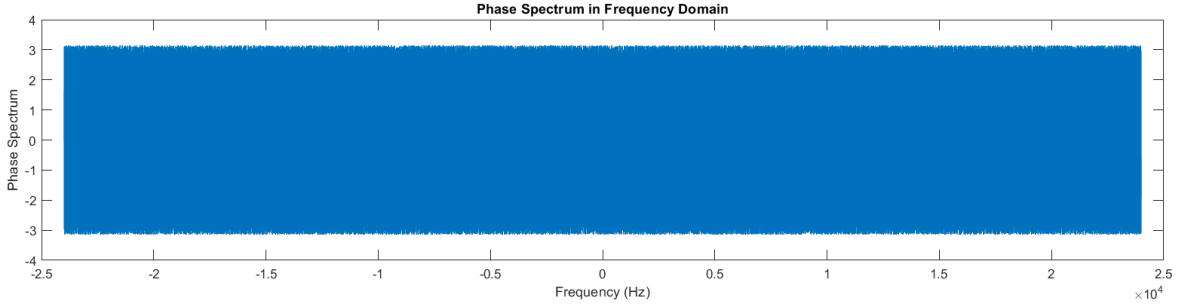
- Plots and Results

1. Audio Reading
   a. Time Domain



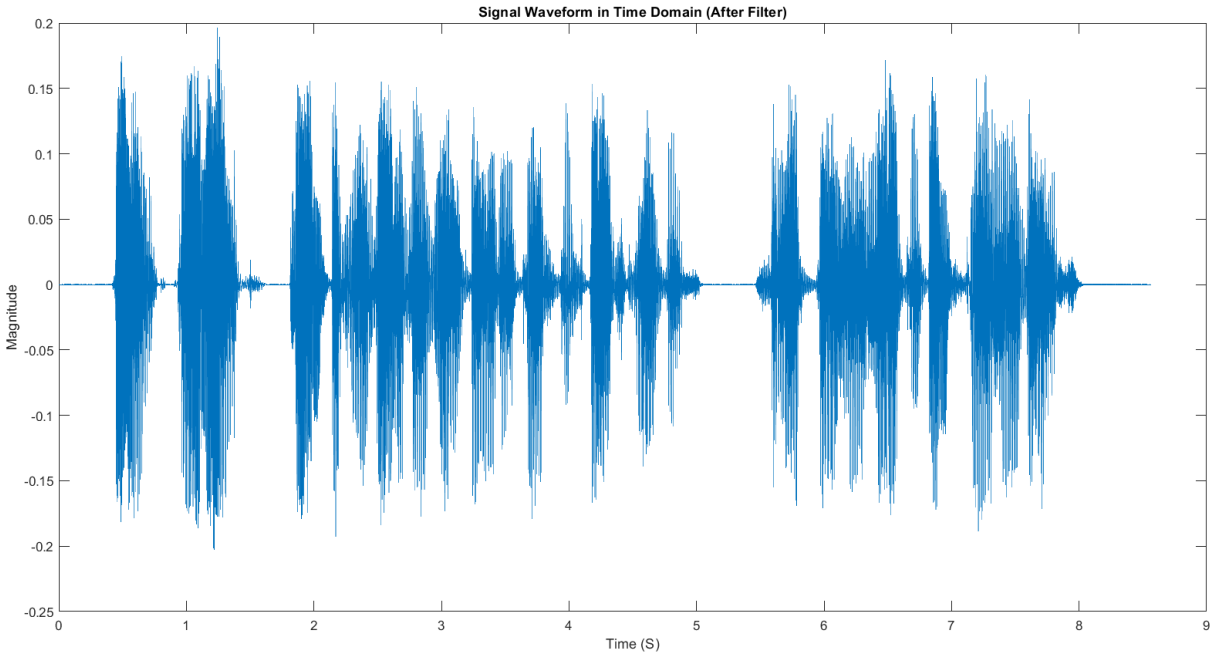
b. Frequency Domain



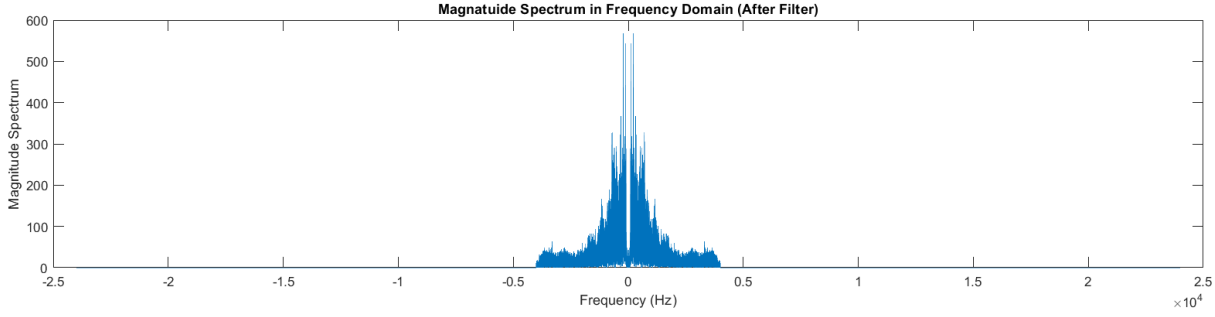


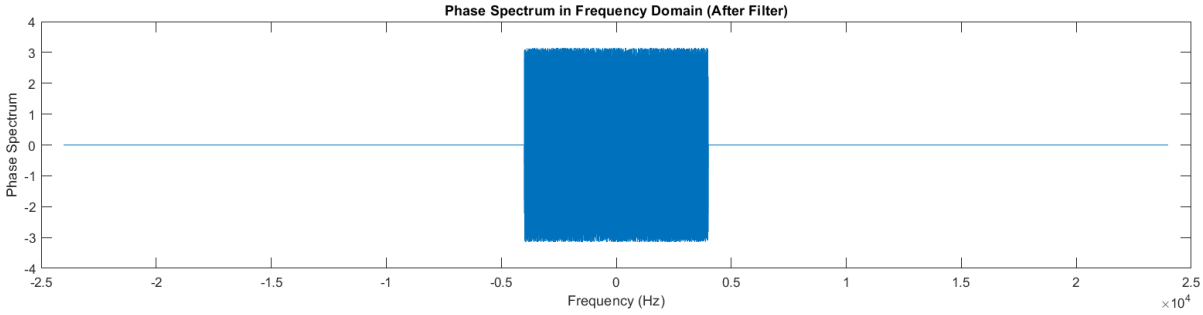
2. Low Pass Filter (LPF)

a. Time Domain



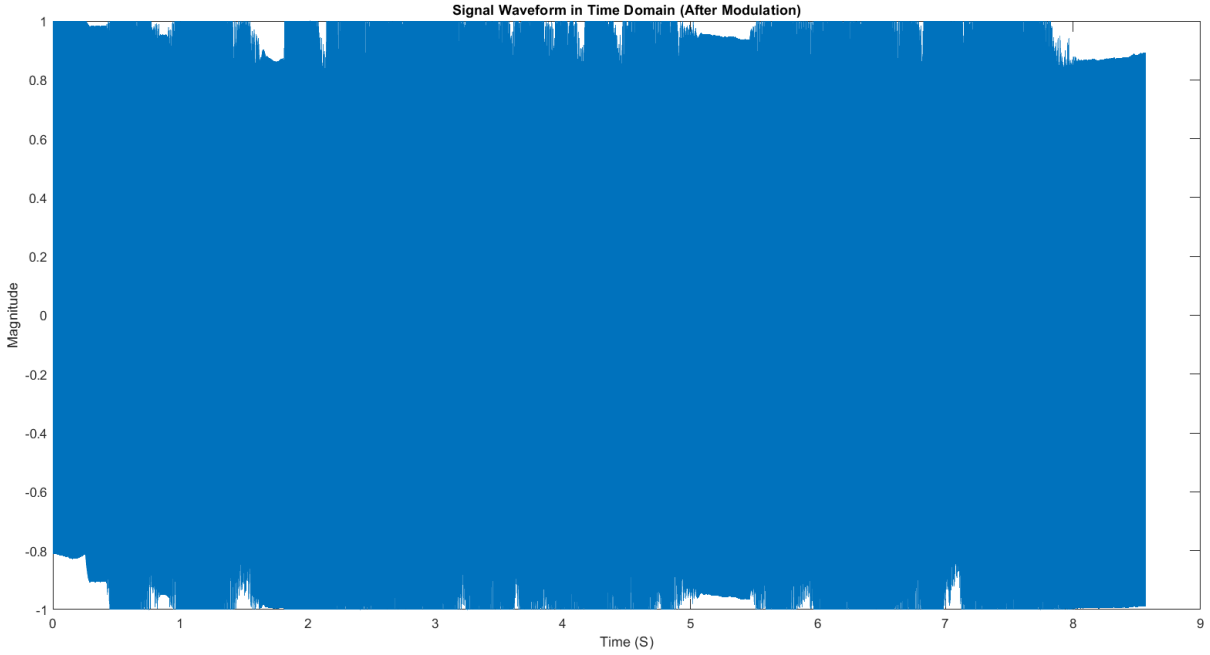
b. Frequency Domain



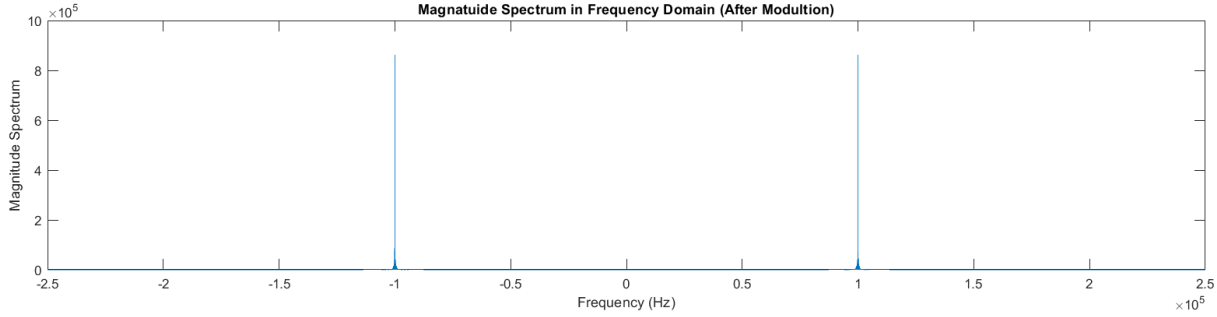


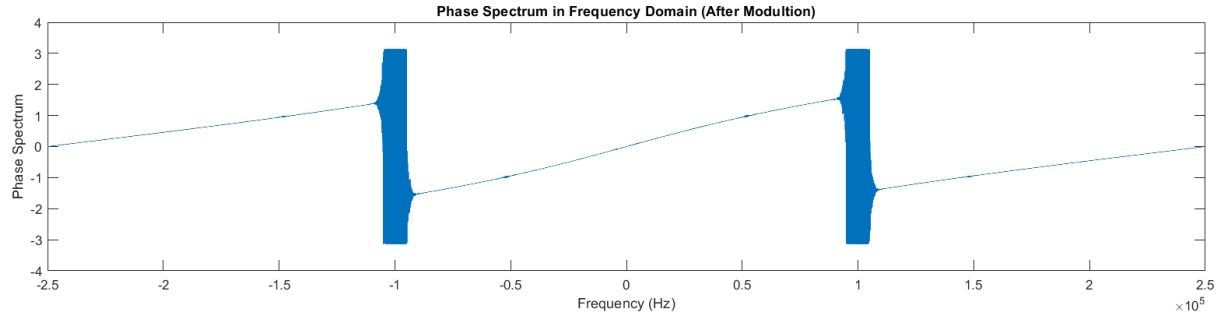
3. NBFM Modulation

a. Time Domain



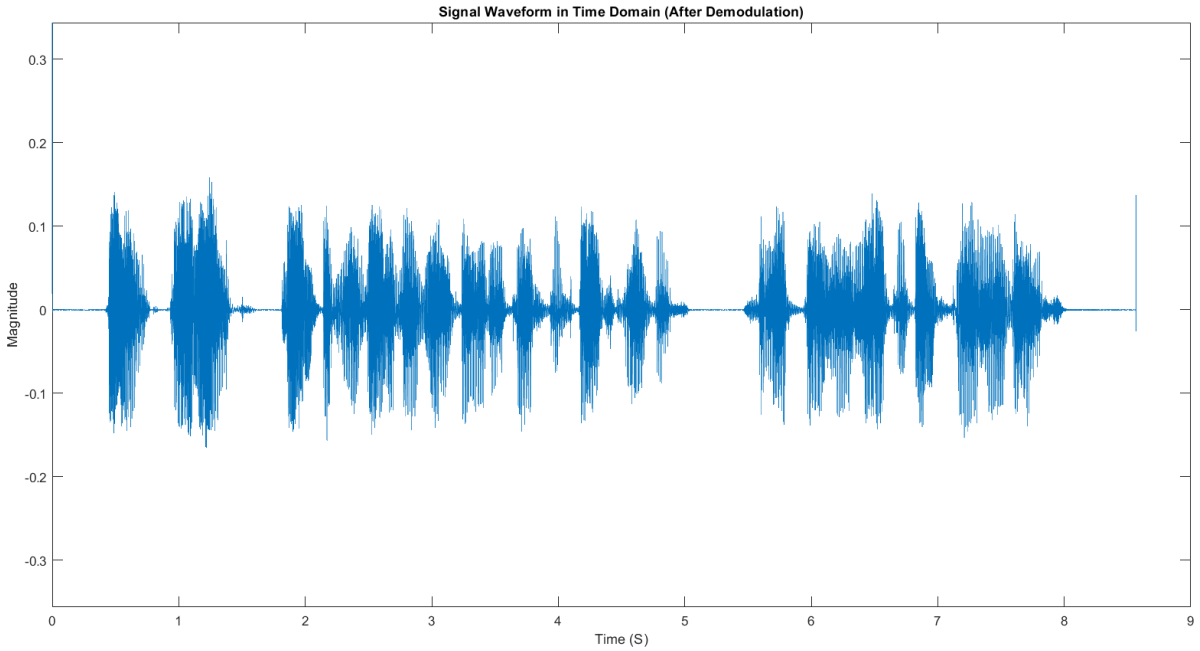
b. Frequency Domain



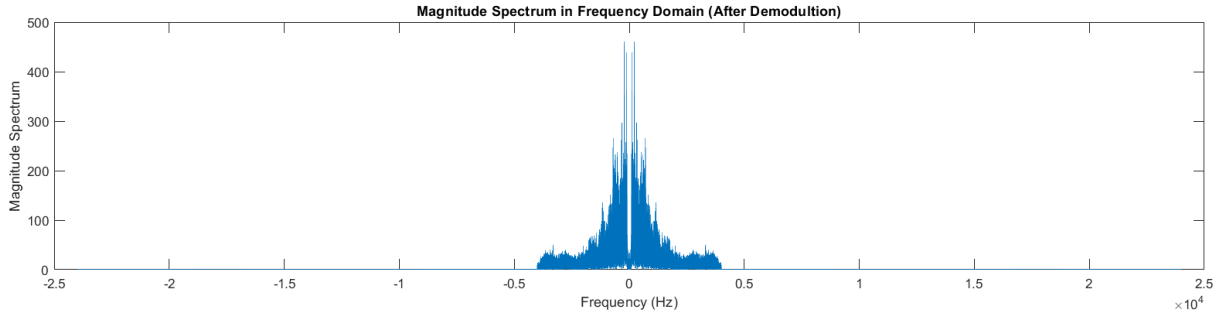


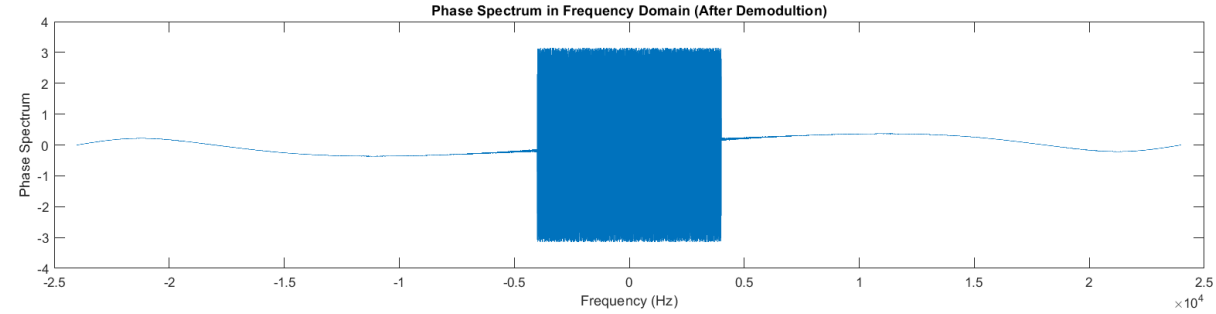
4. Demodulation

a. Time Domain



b. Frequency Domain





Procedures of the Project

1- First, we read the audio file and get its waveform, frequency spectrum for the magnitude and the phase:
   a. We used `audioread()` function to get the audio matrix values
   b. Then applying Fourier transform to get the audio signal in the frequency domain using `fft()` and `fftshift()` functions.
   c. Get the number of audio samples to use to generate the time and frequency vectors which will be used for plotting.
   d. Plot the audio signal in time domain.
   e. Plot the audio signal in frequency domain.
   f. Finally, play the audio to get sure that everything is good using `sound()`.

2- As we know that the bandwidth of the sound is **4K Hz** which means we need to use ideal low pass filter to remove any frequency above these values.
   a. First, we evaluate the number of samples that will get the value of zero in the LPF
   b. Then build LPF by getting vector of zeros then assign the number of samples that above **4K Hz** to zero (we get this number from previous step)
   c. Then multiplying the audio signal in frequency domain by the LPF to get the signal filtered in the frequency domain
   d. Then convert the signal to the time domain to plot the audio signal and play it using `ifft()`, `ifftshift()` and `real()`.
   e. Plot the audio signal in time domain.
   f. Plot the audio signal in frequency domain.
   g. Finally, play the audio to get sure that everything is good using `sound()` function.

3- In this step, we are going to build the modulator that will modulate the signal as FM signal.
   a. First, we are going to initialize the values of carrier frequency (f_c), the new sampling frequency (f_s), the carrier max amplitude (a) and the k_f.
   b. Then, we are going to resample the signal by the new sampling frequency using function `resample()`.

c. Then define variable called phi_t that is equal to:

$$\phi_t = k_f \int_{-\infty}^{t} m(t)\, dt$$

And we are going to use `cumsum()` function to calculate the integral.

d. Then applying the general modulation equation of angle modulation:

$$\begin{aligned} s(t) &= A\cos(\omega_c + \phi_t) \\ &= A\cos(2\pi f_c + \phi_t) \end{aligned}$$

e. Then, we are going to update the samples number, the time vector and the frequency vector.
f. Then, we are going to apply Fourier transform to get the modulated signal in the frequency domain using `fft()` and `fftshift()` functions.
g. Plot the modulated signal in time domain
h. Plot the modulated signal in frequency domain

4- In this step, we are going to demodulate the received signal by following steps and stages:
   a. First, we are going to differentiate the modulated signal using `diff()` function. **NOTE:** we are going to add one at the end of the vector because MATLAB reduce the differentiated vectors by one in size.
   b. Then we are going to use envelope detector to get the magnitude of the modulated signal to recover the signal by using `abs()` and `hilbert()` functions.
   c. Then, the signal will pass by DC block to eliminate the DC component (carrier component) by getting the DC value then subtracting it from the signal by using `mean()` function.
   d. Then, we are going to divide the signal by constant (A * k_f) to get the audio signal with its original values.

e. Then resampled the demodulated signal again but this time for the old value of sampling frequency using function `resample()`.
f. Then, we apply Fourier transform to the signal to get the demodulate signal in the frequency domain using `fft()` and `fftshift()` functions.
g. Then, we are going to update the samples number, the time vector and the frequency vector again for the old sampling frequency.
h. Plot the demodulated signal in the time domain.
i. Plot the demodulated signal in the frequency domain.
j. Finally, play the demodulated audio signal to be sure that the signal is the same original signal that we want to send using `sound()` function.

The Code

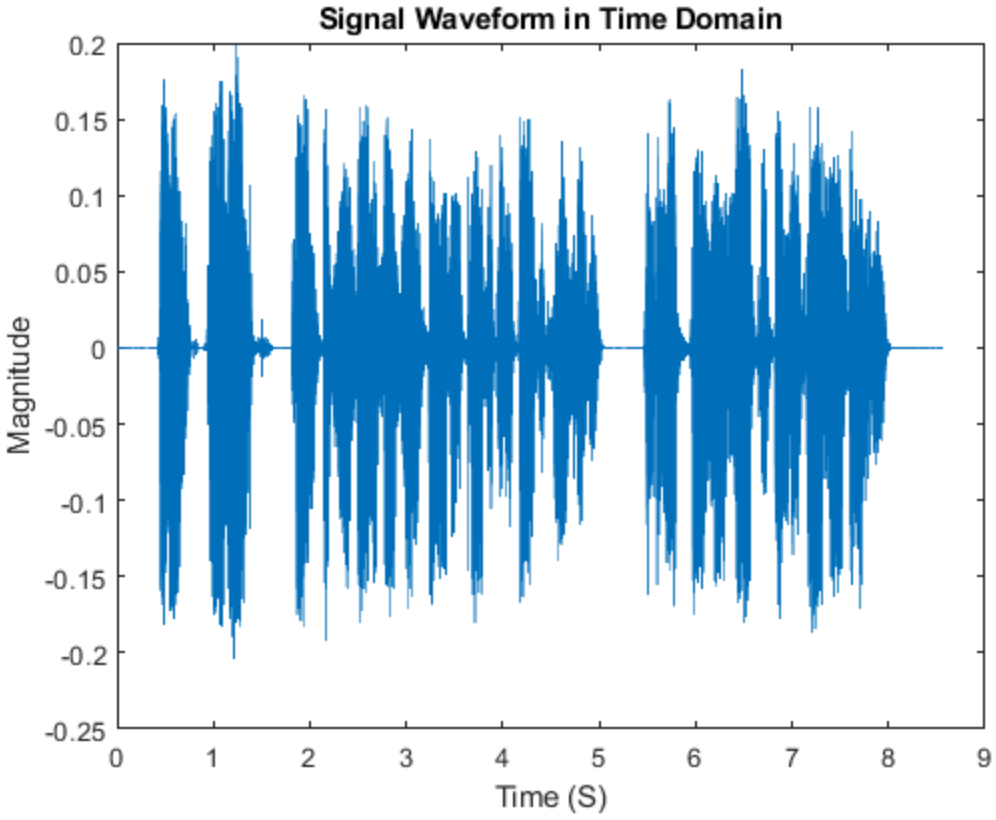
```
%%%%%%%%%%%%%%%%%%%% FM COMMNUNICATION SYSTEM %%%%%%%%%%%%%%%%%%%%
%%%%
clear,clc,close all
%%%%%%%%%%%%%%%%%  Audio Reading  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf("===============  Audio Reading  ================== \n");
% Read The Audio
[y, Fs] = audioread('eric.wav');    % Fs: Sampling Frequency
Y = fftshift(fft(y));               % Fourier Transform (Frequency
 Domain)

% necessary values to plot
Ns = size(y,1);                     % Samples Number
t = linspace(0, Ns/Fs, Ns);         % Generate Time, end time = Ns/Fs
Pvec = linspace(-Fs/2,Fs/2,Ns);     % Frequency values vector on x-
axis

% Plot the audio in Time Domain
plot(t,y);
xlabel('Time (S)');
ylabel('Magnitude');
title("Signal Waveform in Time Domain");

% Plot the audio in Frequency Domain
figure;
subplot(2,1,1);
plot(Pvec, abs(Y))                  % Plot the Magnatuide
xlabel('Frequency (Hz)');
ylabel('Magnitude Spectrum');
title("Magnatuide Spectrum in Frequency Domain");
subplot(2,1,2);
plot(Pvec, angle(Y))                % Plot the Phase
xlabel('Frequency (Hz)');
ylabel('Phase Spectrum');
title("Phase Spectrum in Frequency Domain");

% Play The Audio
fprintf("AUDIO PLAYING ON: Audio Reading \n");
sound(y,Fs);
pause(Ns/Fs) % pausing to have some time between playing audios
fprintf("AUDIO PLAYING OFF: Audio Reading \n");

%%%%%%%%%%%%%%%%%  LOW PASS Filter  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf("=================  LOW PASS FILTER  ====================
 \n");
signal_time = y;
signal_frequency = Y;

% band width of the voice (fm)
low_filter_BW = 4000;      % fm

% number of samples that will be filtered from both sides of the
 signal
```

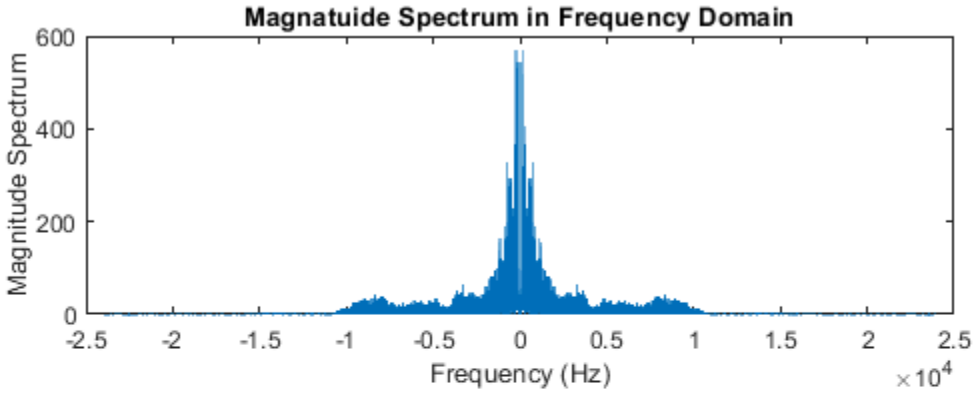
```
n_filter = floor((Ns/Fs) * (Fs/2 - low_filter_BW));    % 4000 Hz
% LOW PASS Filter
lowpass_filter = ones(Ns,1);
lowpass_filter([(1:n_filter) (end-n_filter+1:end)]) = 0;
% to filter the signal to pass only low frequencies (>4000 Hz)
filtered_signal_freq = lowpass_filter .* signal_frequency;
% signal in time domian after passing low pass filter
filtered_signal_time = real(ifft(ifftshift(filtered_signal_freq)));

% plot the filtered signal in time domain
figure;
plot(t, filtered_signal_time)
xlabel('Time (S)');
ylabel('Magnitude');
title("Signal Waveform in Time Domain (After Filter)");

% plot the filtered signal in frequency domain
figure;
subplot(2,1,1);
plot(Pvec, abs(filtered_signal_freq))  % plot the frequency magnatuide
xlabel('Frequency (Hz)');
ylabel('Magnitude Spectrum');
title("Magnatuide Spectrum in Frequency Domain (After Filter)");
subplot(2,1,2);
plot(Pvec, angle(filtered_signal_freq))    % Plot the Phase
xlabel('Frequency (Hz)');
ylabel('Phase Spectrum');
title("Phase Spectrum in Frequency Domain (After Filter)");

% Play the audio after Filter
fprintf("AUDIO PLAYING ON: Signal After Low Filter \n");
sound(filtered_signal_time,Fs)
pause(Ns/Fs) % pausing to have some time between playing audios
fprintf("AUDIO PLAYING OFF: Signal After Low Filter \n");

%%%%%%%%%%%%%%%%%  NBFM Modulation  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf("===============  NBFM Modulation  ================== \n");

Fc = 100000;          % Carrier Frequency
Fs_FM = 5 * Fc;       % Sampling Frequency for FM modluation signal
A = 1;                % Carrier Amplitude
Kf = 2*pi*0.01;       % Frequency Modulation Constant

% resampling the signal with the new sampling frequenccy
resampled_signal_time = resample(filtered_signal_time,Fs_FM,Fs);

% update Ns, t, Pvec to plot the signal in time and frequecy domains
Ns_FM = size(resampled_signal_time,1);      % Samples Number
Pvec_FM = linspace(-Fs_FM/2,Fs_FM/2,Ns_FM); % Generate Frequency
 Vector
t_FM = linspace(0, Ns_FM/Fs_FM, Ns_FM);     % Generate Time Vector

% NBFM Modulation in time domain
phi_t = Kf * cumsum(resampled_signal_time);
```

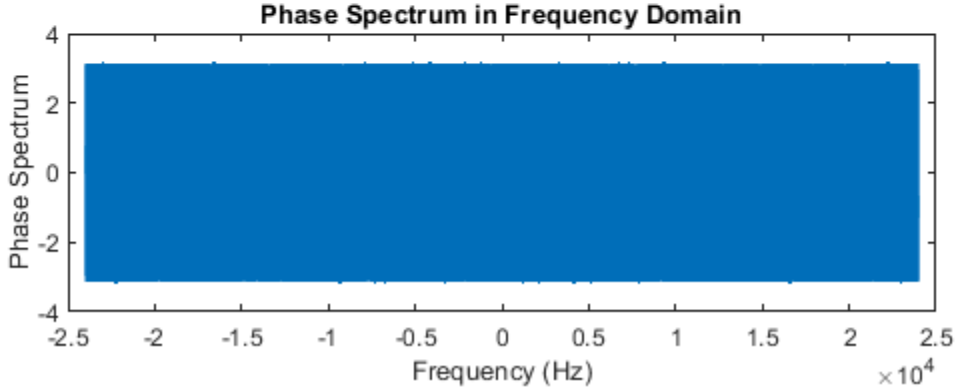
```
modulated_signal_time = A * cos(2*pi*Fc * t_FM' + phi_t);
% Modulation in frequency domain
modulated_signal_freq = fftshift(fft(modulated_signal_time));

% plot the modulated signal in time domain
figure;
plot(t_FM, modulated_signal_time)
xlabel('Time (S)');
ylabel('Magnitude');
title("Signal Waveform in Time Domain (After Modulation)");

% plot the modulated signal in frequency domain
figure;
subplot(2,1,1);
plot(Pvec_FM, abs(modulated_signal_freq))
xlabel('Frequency (Hz)');
ylabel('Magnitude Spectrum');
title("Magnatuide Spectrum in Frequency Domain (After Modultion)");
subplot(2,1,2);
plot(Pvec_FM, angle(modulated_signal_freq))        % Plot the Phase
xlabel('Frequency (Hz)');
ylabel('Phase Spectrum');
title("Phase Spectrum in Frequency Domain (After Modultion)");

%%%%%%%%%%%%%%%%%  NBFM Demodulation  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf("===============  NBFM Demodulation  ================= \n");

% Differentiator Stage
demodulated_signal_time_diff = [diff(modulated_signal_time); 1];
% Envolpe Detector
demodulated_signal_time_ED =
 abs(hilbert(demodulated_signal_time_diff));
% DC Block, AC mean = 0 so DC value is the mean of the signal.
DC_value = mean(demodulated_signal_time_ED);
demodulated_signal_time_dcblock = demodulated_signal_time_ED -
 DC_value;
% to get the original signal, we need to divide the signal by
 constants
demodulated_signal_time = demodulated_signal_time_dcblock / (A * Kf);
% to get the original signal in its orginal sampling frequecy
demodulated_signal_time = resample(demodulated_signal_time,Fs,Fs_FM);

% the demodulated signal in the frequency domain
demodulated_signal_freq = fftshift(fft(demodulated_signal_time));

% update Ns, t, Pvec
Ns = size(demodulated_signal_time,1);     % Samples Number
t = linspace(0, Ns/Fs, Ns);               % Generate Time Vector
Pvec = linspace(-Fs/2,Fs/2,Ns);           % Frequency values Vector

% plot the demodulated signal in time domain
figure;
plot(t, demodulated_signal_time)
xlabel('Time (S)');
```

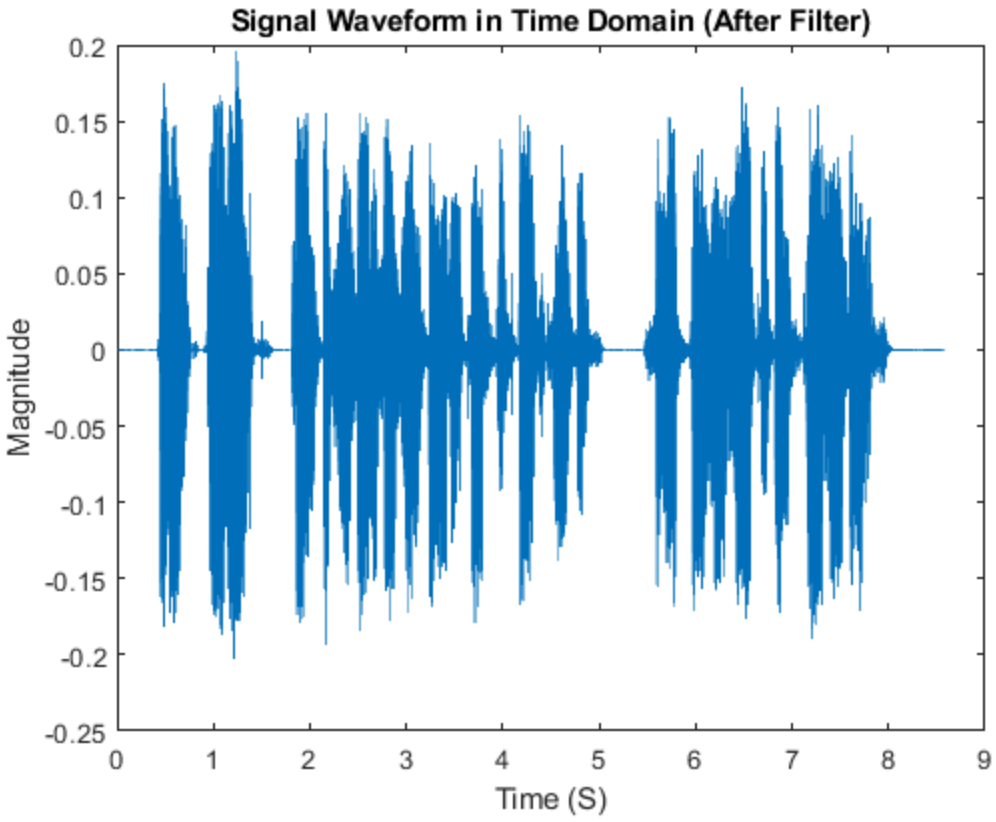
```
ylabel('Magnitude');
title("Signal Waveform in Time Domain (After Demodulation)");

% plot the demodulated signal in frequency domain
figure;
subplot(2,1,1);
plot(Pvec, abs(demodulated_signal_freq))  % plot the signal magnatuide
xlabel('Frequency (Hz)');
ylabel('Magnitude Spectrum');
title("Magnitude Spectrum in Frequency Domain (After Demodultion)");
subplot(2,1,2);
plot(Pvec, angle(demodulated_signal_freq)) % Plot the Phase
xlabel('Frequency (Hz)');
ylabel('Phase Spectrum');
title("Phase Spectrum in Frequency Domain (After Demodultion)");

% Play the audio after Demodulation
fprintf("AUDIO PLAYING ON: Signal After Demodulation \n");
sound(demodulated_signal_time,Fs)
pause(Ns/Fs) % pausing to have some time between playing audios
fprintf("AUDIO PLAYING OFF: Signal After Demodulation \n");

%%%%%%%%%%%%%%%%%%%%%%%%%%%%  THE END  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

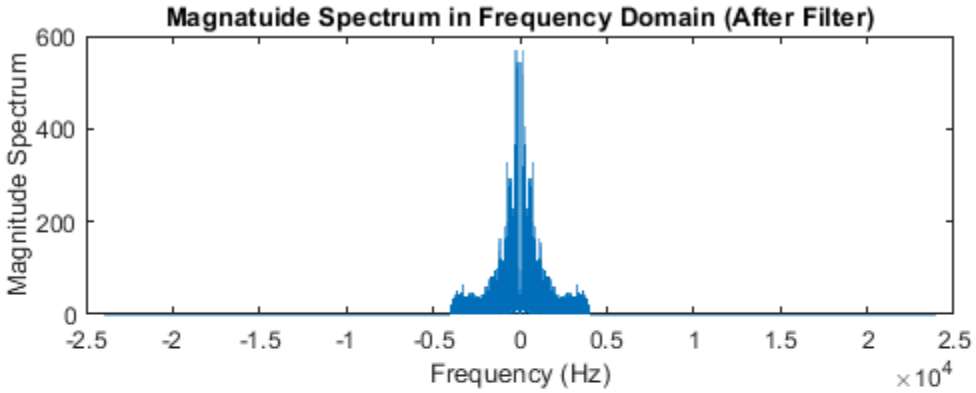
```
===============  Audio Reading  ===================
AUDIO PLAYING ON: Audio Reading
AUDIO PLAYING OFF: Audio Reading
=================  LOW PASS FILTER  ====================
AUDIO PLAYING ON: Signal After Low Filter
AUDIO PLAYING OFF: Signal After Low Filter
===============  NBFM Modulation  ===================
===============  NBFM Demodulation  =================
AUDIO PLAYING ON: Signal After Demodulation
AUDIO PLAYING OFF: Signal After Demodulation
```

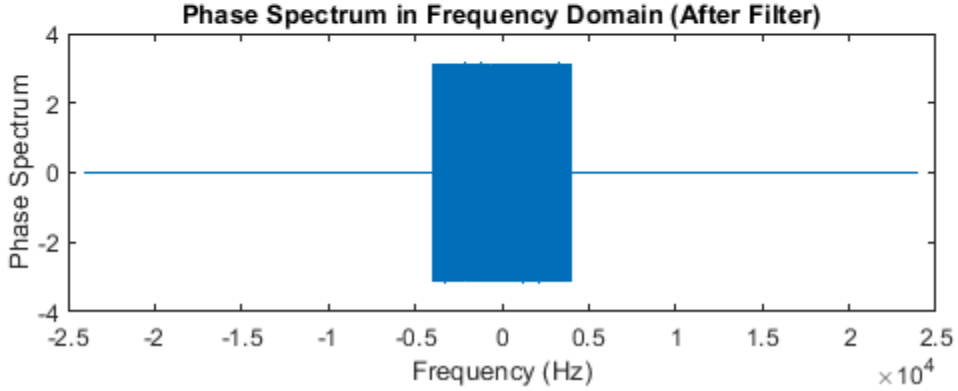

Signal Waveform in Time Domain
Magnitude
Time (S)
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
1
2
3
4
5
6
7
8
9

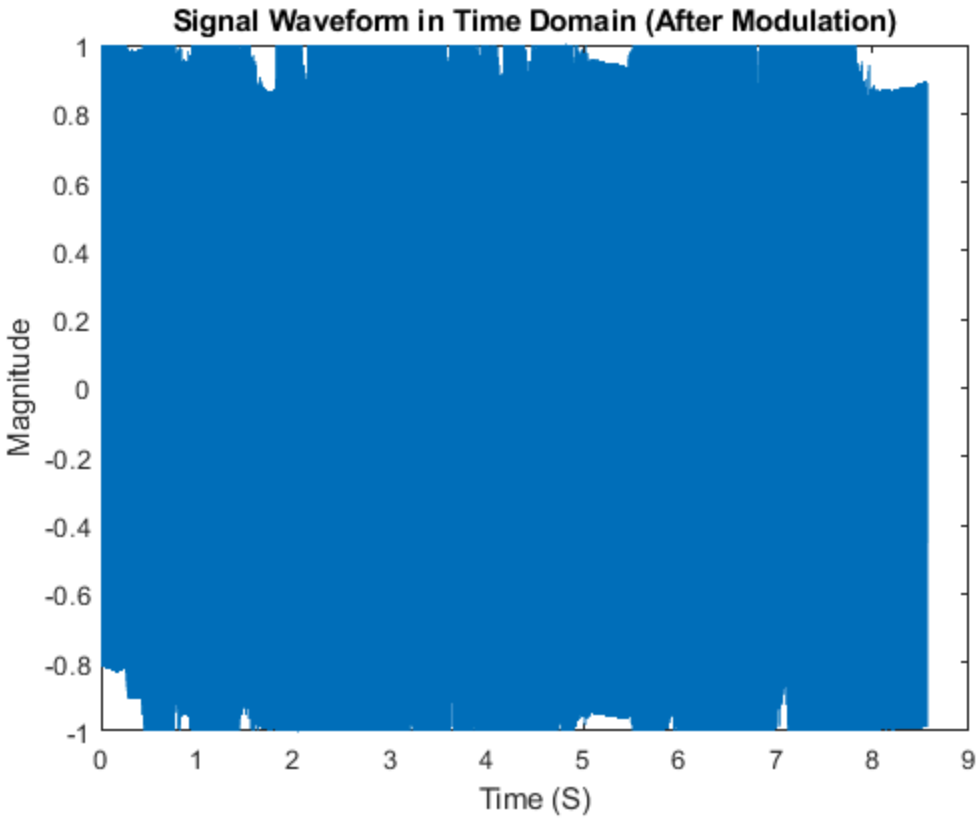

Magnatuide Spectrum in Frequency Domain
Magnitude Spectrum
Frequency (Hz)
600
400
200
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^4

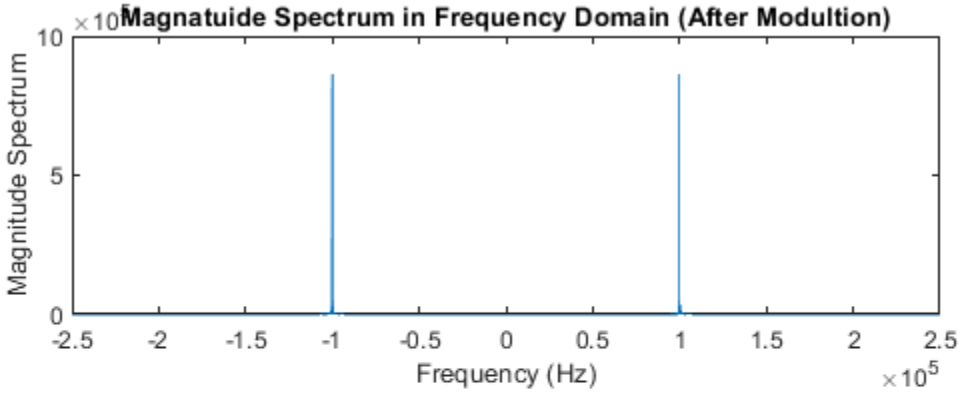

Phase Spectrum in Frequency Domain
Phase Spectrum
Frequency (Hz)
4
2
0
-2
-4
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^4

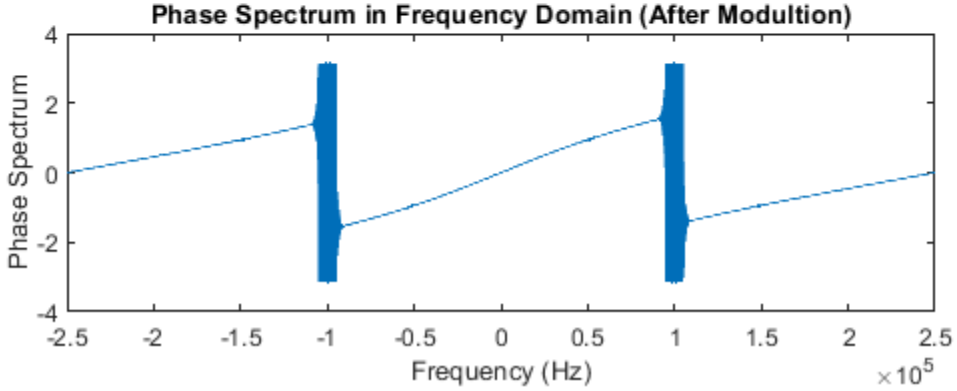

Signal Waveform in Time Domain (After Filter)
Magnitude
Time (S)
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
1
2
3
4
5
6
7
8
9

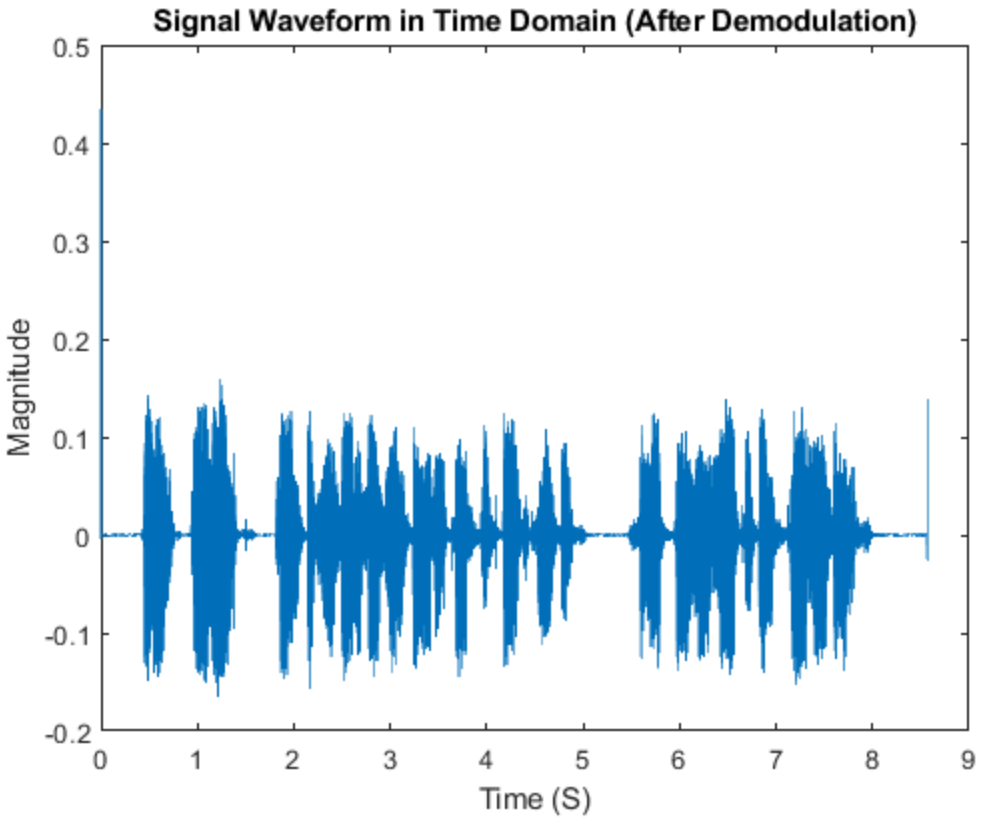

Magnatuide Spectrum in Frequency Domain (After Filter)
Magnitude Spectrum
Frequency (Hz)
600
400
200
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^4

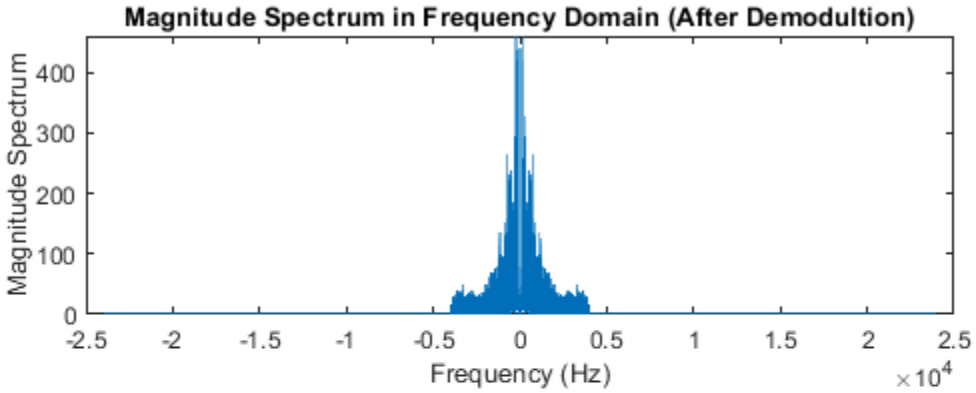

Phase Spectrum in Frequency Domain (After Filter)
Phase Spectrum
Frequency (Hz)
4
2
0
-2
-4
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^4

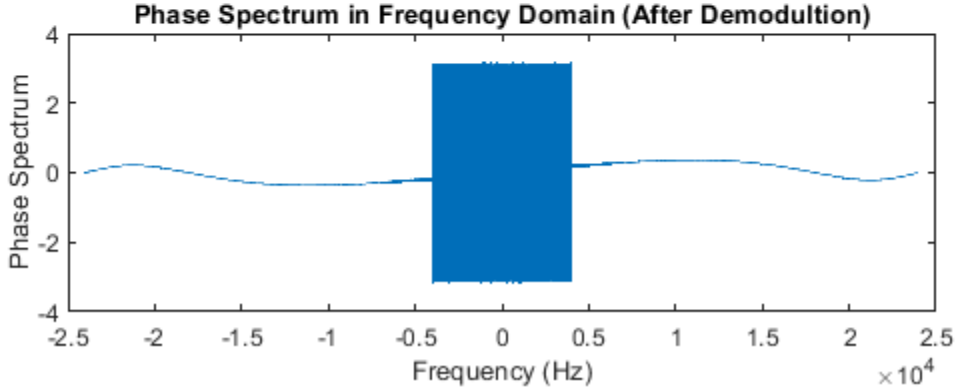

Signal Waveform in Time Domain (After Modulation)
Magnitude
Time (S)


Magnatuide Spectrum in Frequency Domain (After Modultion)
Magnitude Spectrum
Frequency (Hz)


Phase Spectrum in Frequency Domain (After Modultion)
Phase Spectrum
Frequency (Hz)


Signal Waveform in Time Domain (After Demodulation)
Magnitude
Time (S)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
0
1
2
3
4
5
6
7
8
9


Magnitude Spectrum in Frequency Domain (After Demodultion)
Magnitude Spectrum
Frequency (Hz)
400
300
200
100
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^4


Phase Spectrum in Frequency Domain (After Demodultion)
Phase Spectrum
Frequency (Hz)
4
2
0
-2
-4
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^4

Published with MATLAB® R2019a