Fundamentals of Communication systems Project (ECE252s)

TEAM 34

Name	ID	SEC & Major
Mai Mohamed	2200516	1 ECE
Doha Mohamed	2200911	1 CSE
Marium Waleed	2200823	1 ECE
Huda Mohamed	2200168	2 ECE
Menna Hassan	2200677	1 ECE
Menna Ayman	2200236	4 CSE

SUBMITTED TO:

DR. MAZEN ERFAN   **ENG. AHMED AL-SAYED**

DR. ALAA FATHY   **ENG. AHMED KHALED**

ANALOG COMMUNICATION

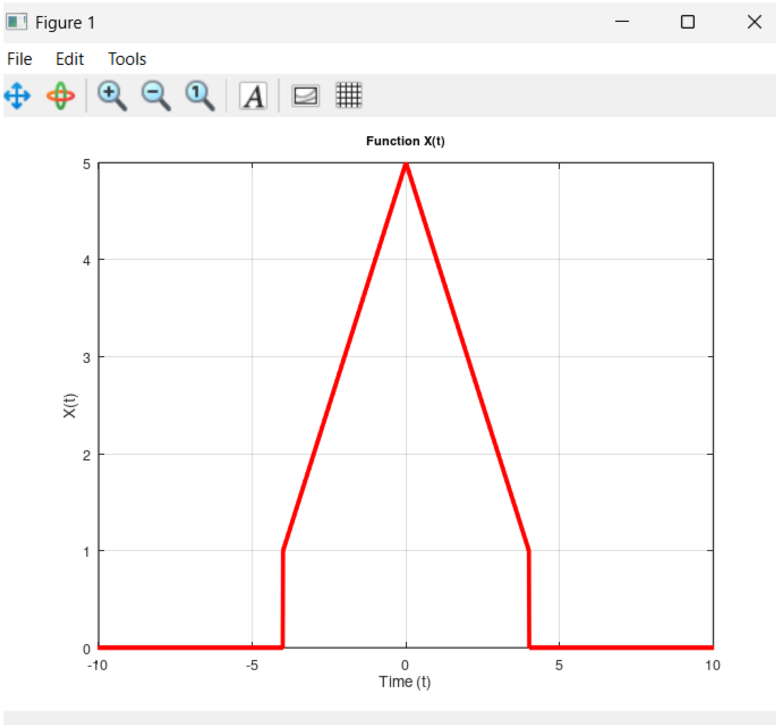
Definitions and parameters

```
clc;
clear;
close all;

%%%%%%%%%%%%%%%%%%%%%%% Time and frequency parameters %%%%%%%%%%%%%%%%%%%%%%%%%
fs = 100;
T = 100;
df = 0.01;

dt = 1/fs;
N = ceil(T/dt);
t = -(N*dt/2) : dt : ((N*dt/2) - dt);
t = t(1:N);

f = -(0.5*fs) : df : (0.5*fs - df);
f = f(1:N);
```

1. Plot x(t)

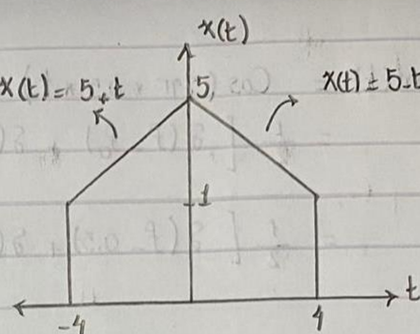
```
%%%%%%%%%%%%%%%%%%%%%%%%% 1- Plot X(t) on octave %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x_time = zeros(size(t));
for i = 1:length(t)
    if t(i) >= -4 && t(i) < 0
        x_time(i) = t(i) + 5;
    elseif t(i) >= 0 && t(i) <= 4
        x_time(i) = -t(i) + 5;
    end
end

figure(1);
plot(t, x_time, 'r', 'LineWidth' , 1.5);
title('Function X(t)');
xlabel('Time (t)');
ylabel('X(t)');
grid on;
xlim([-10 10]);
```

OUTPUT:



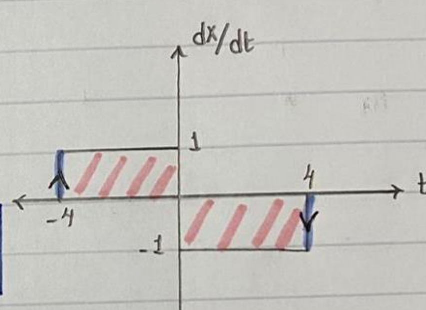
2. Analytical Expression for x(t):

$$FT\left[\frac{dx}{dt}\right] = j2\pi f \, X[f]$$



$$FT\left[\frac{dx}{dt}\right] = e^{j2\pi f(4)} + 4\,\mathrm{Sinc}(4f)\, e^{j2\pi f(2)} - 4\,\mathrm{Sinc}(4f)\, e^{-j2\pi f(2)} - e^{-j2\pi f(4)}$$

$$\therefore \; j2\pi f \, X[f] = 4\,\mathrm{Sinc}(4f)\left[e^{j4\pi f} - e^{-j4\pi f}\right] + \left[e^{j8\pi f} - e^{-j8\pi f}\right]$$



$$X[f] = 4\,\mathrm{Sinc}(4f)\left[\frac{e^{j4\pi f} - e^{-j4\pi f}}{2j \; \pi f}\right] + \left[\frac{e^{j8\pi f} - e^{-j8\pi f}}{2j \; \pi f}\right]$$

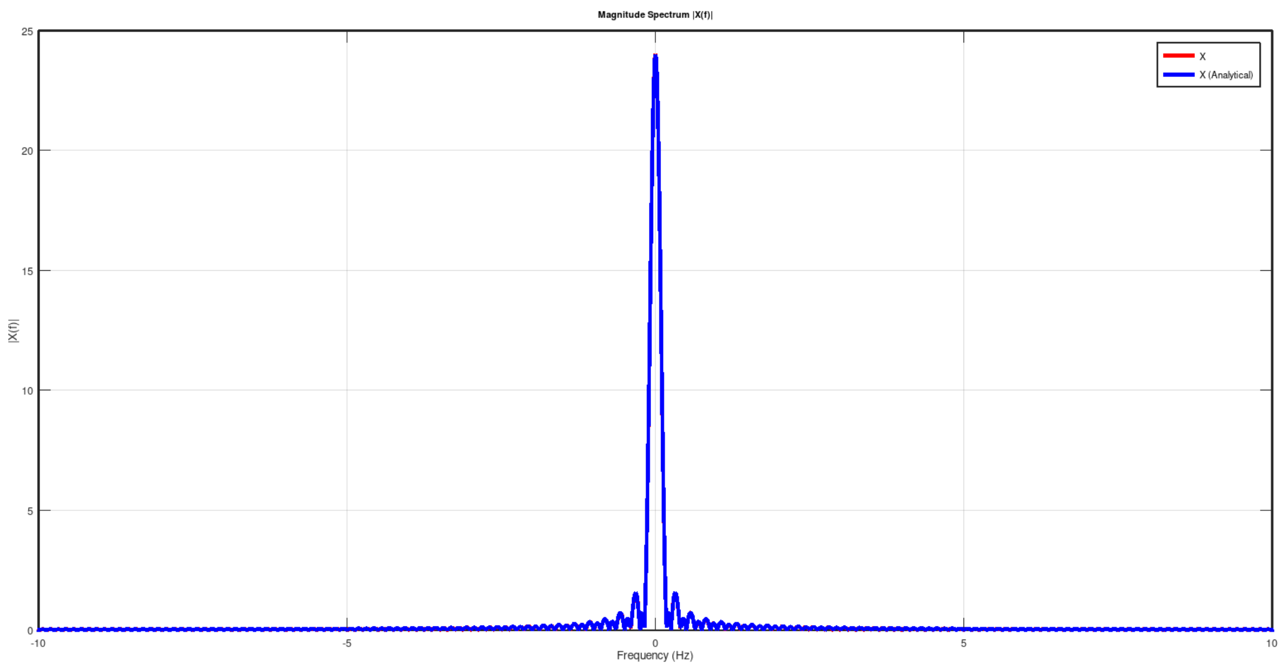
$$X[f] = 4\,\mathrm{Sinc}(4f) \cdot \frac{4\,\mathrm{Sin}(4\pi f)}{4\pi f} + \frac{8\,\mathrm{Sin}(8\pi f)}{8\pi f}$$

$$X[f] = 16\,\mathrm{Sinc}^2(4f) + 8\,\mathrm{Sinc}(8f)$$

3. Fourier Transform of x(t):

```
%%%%%%%%%%%%%%%%%%%%%%%%% 3- Plot X(f) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x_freq = fftshift(fft(x_time) * dt);

figure(2);
plot(f, abs(x_freq), 'r', 'LineWidth', 1.5);
hold on;
plot(f, abs(x_analytical_freq), 'b', 'LineWidth', 1.5);
title('Magnitude Spectrum |X(f)|');
legend('X', 'X (Analytical)');
xlabel('Frequency (Hz)');
ylabel('|X(f)|');
xlim([-10 10]);
grid on;
```

OUTPUT



4. Estimating the Bandwidth:

```
%%%%%%%%%%%%%%%%%%%%%%%%% 4- Estimate Bandwidth of X %%%%%%%%%%%%%%%%%%%%
psd_x = abs(x_freq).^2; % Power Spectral Density

figure(3);
plot(f, psd_x, 'r');
title('Power Spectrum of X(t)');
xlabel('Frequency (Hz)');
ylabel('Power');
grid on;

threshold = 0.05 * max(psd_x);
indices = find(psd_x >= threshold);
BW_x = max(abs(f(indices)));

fprintf('X Estimated Bandwidth = %1.3f Hz\n', BW_x);

% Analytical bandwidth
psd_analytical_x = abs(x_analytical_freq).^2;
threshold = 0.05 * max(psd_analytical_x);
indices = find(psd_analytical_x >= threshold);
BW_analytical_x = max(abs(f(indices)));

fprintf('X Analytical Estimated Bandwidth = %1.3f Hz\n', BW_analytical_x);
```

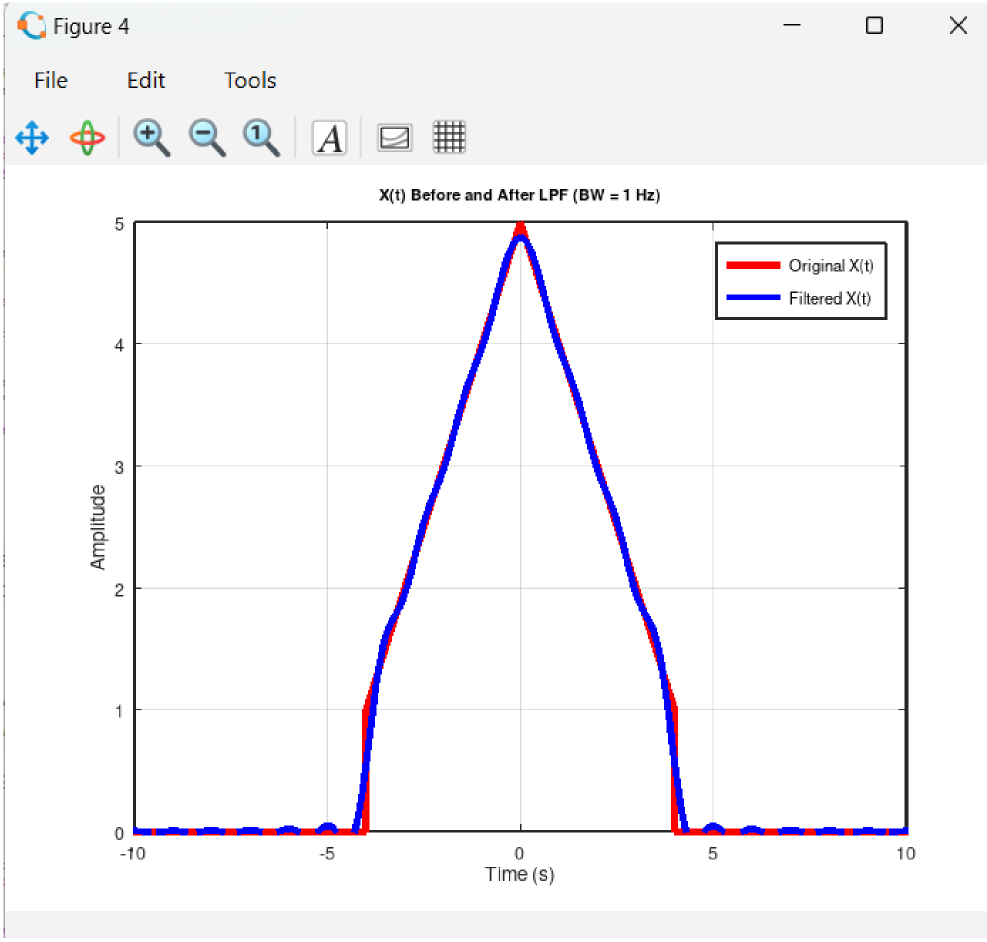
OUTPUT:

```
X Estimated Bandwidth = 0.130 Hz
X Analytical Estimated Bandwidth = 0.130 Hz
```

5. LPF with BW = 1 Hz

```
%%%%%%%%%%%%%%%%%%%%%%%%% 5- Low Pass Filter (BW = 1 Hz) %%%%%%%%%%%%%%%%%%%
LPF_BW1 = 1;
H = abs(f) < LPF_BW1;
x_filtered1 = ifft(ifftshift(H .* x_freq)) / dt;

figure(4);
plot(t, x_time, 'r', 'LineWidth', 1.5);
hold on;
plot(t, real(x_filtered1), 'b', 'LineWidth', 1.5);
title('X(t) Before and After LPF (BW = 1 Hz)');
legend('Original X(t)', 'Filtered X(t)');
xlabel('Time (s)');
ylabel('Amplitude');
xlim([-10 10]);
ylim([0 5]);
grid on;
```

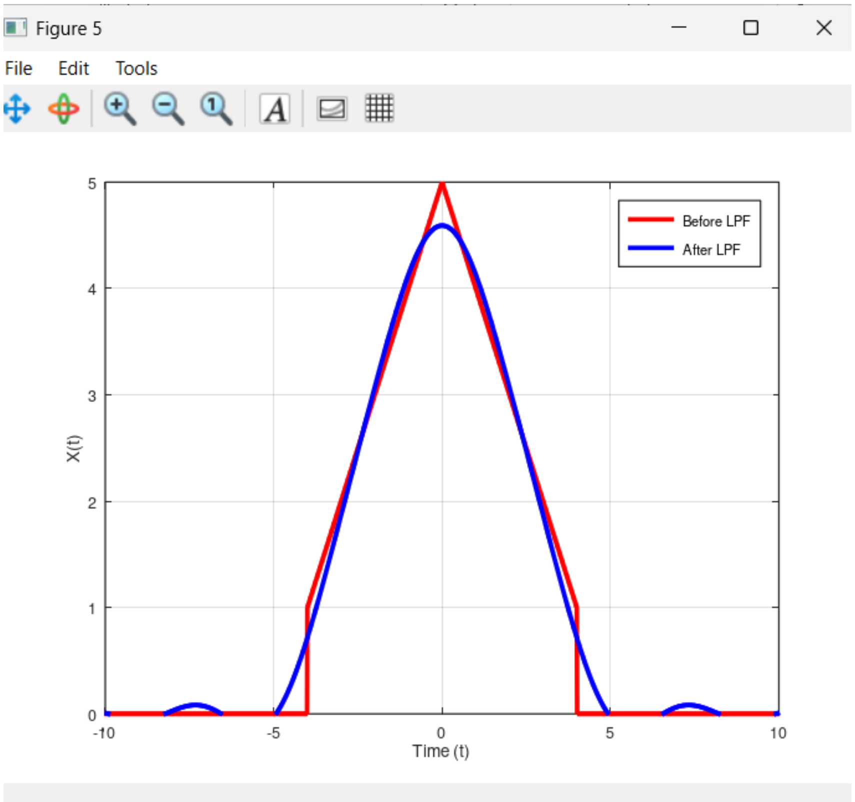
OUTPUT:



6. LPF with BW = 0.3 Hz

```
%%%%%%%%%%%%%%%%%%%%%%%%% 6- Low Pass Filter (BW = 0.3 Hz) %%%%%%%%%%%%%%%%%
LPF_BW2 = 0.3;
H = abs(f) < LPF_BW2;
x_filtered2 = ifft(ifftshift(H .* x_freq)) / dt;

figure(5);
plot(t, x_time, 'r', 'LineWidth', 1.5);
hold on;
plot(t, real(x_filtered2), 'b', 'LineWidth', 1.5);
title('X(t) Before and After LPF (BW = 0.3 Hz)');
legend('Original X(t)', 'Filtered X(t)');
xlabel('Time (s)');
ylabel('Amplitude');
xlim([-10 10]);
ylim([0 5]);
grid on;
```

OUTPUT:

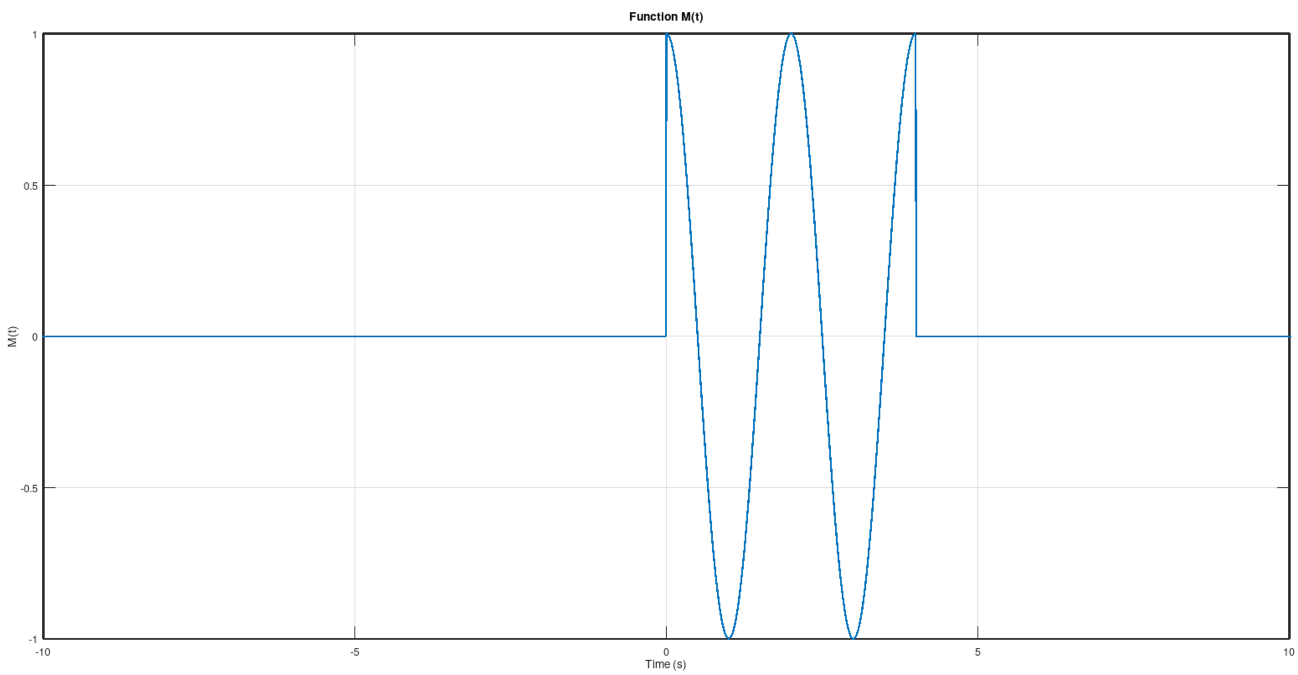


7. m(t) = cos(2 $\pi * 0.5 * t$)   0 < t < 4

```
%%%%%%%%%%%%%%%%%%%%%%%%%% 7- Repeat for m(t) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% a- Define m(t)
m_time = cos(2 * pi * 0.5 * t);
m_time(t > 4) = 0;
m_time(t < 0) = 0;

figure(6);
plot(t, m_time);
title('Function M(t)');
xlabel('Time (s)');
ylabel('M(t)');
grid on;
xlim([-10 10]);

% b- Analytical expression for M(f)
m_analytical_freq = 2 * sinc(4 * (f - 0.5)) .* exp(-1i * 2 * pi * 2 * (f - 0.5))
+ 2 * sinc(4 * (f + 0.5)) .* exp(-1i * 2 * pi * 2 * (f + 0.5));

% c- Plot M(f)
m_freq = fftshift(fft(m_time) * dt);

figure(7);
plot(f, abs(m_freq), 'r', 'LineWidth', 1.5);
hold on;
plot(f, abs(m_analytical_freq), 'b', 'LineWidth', 1.5);
legend('M (Numerical)', 'M (Analytical)');
xlabel('Frequency (Hz)');
ylabel('|M(f)|');
xlim([-10 10]);
grid on;
```

```
% d- Calculate bandwidth of m(t)
psd_m = abs(m_freq).^2;

figure(8);
plot(f, psd_m, 'r');
title('Power Spectrum of M(t)');
xlabel('Frequency (Hz)');
ylabel('Power');
grid on;

threshold = 0.05 * max(psd_m);
indices = find(psd_m >= threshold);
BW_m = max(abs(f(indices)));

fprintf('M Estimated Bandwidth = %1.3f Hz\n', BW_m);

% e- Analytical bandwidth
psd_analytical_m = abs(m_analytical_freq).^2;
threshold = 0.05 * max(psd_analytical_m);
indices = find(psd_analytical_m >= threshold);
BW_analytical_m = max(abs(f(indices)));

fprintf('M Analytical Estimated Bandwidth = %1.3f Hz\n', BW_analytical_m);
```

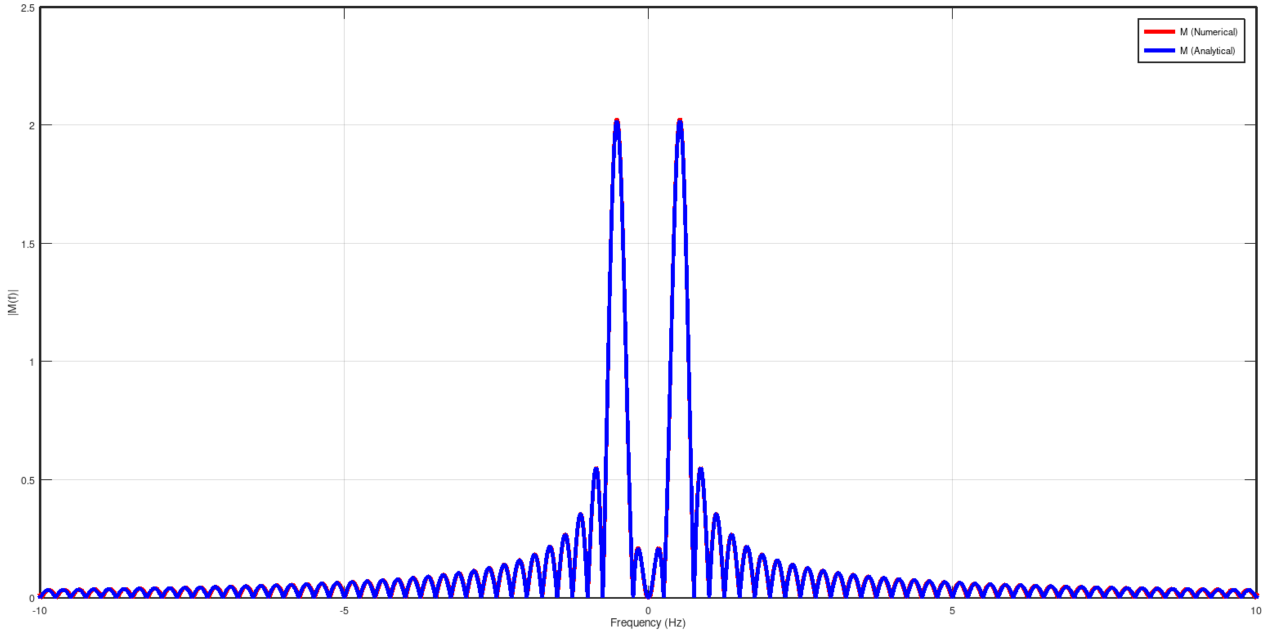
1- m(t):



2- Analytical Expression

m(t) is the product of cos(2 π 0.5t) with Rectangle its width is from t=0 to t=4 and its amplitude 1 m(f)=1 2 (δ(f-0.5)+δ(f+0.5)) convoluted with 4·sinc(4f)·*e*j2π(2f) m(f)= 2·sinc(4(f-0.5))·*e*j2π(2(f−0.5))+2·sinc(4(f+0.5))·*e*j2π(2(f+0.5))

3- M(f)



4- Estimated BW:

```
M Estimated Bandwidth = 0.900 Hz
M Analytical Estimated Bandwidth = 0.910 Hz
```

8. FDM Modulation Scheme:

```
%%%%%%%%%%%%%%%%%%%%%%%%% 8- Frequency Division Multiplexing (FDM) %%%%%%%%%%%%%
c1 = cos(2 * pi * 20 * t);
x_modulated = x_filtered1 .* c1;  % s1(t)


%%%%%%%%%%%%%%%%%%%%%%%%% 9- Second carrier and modulated m(t) %%%%%%%%%%%%%%%%%
c2 = cos(2 * pi * 24 * t);
m_modulated = m_time .* c2;  % s2(t)
m_modulated_freq = fftshift(fft(m_modulated) * dt);

%%%%%%%%%%%%%%%%%%%%%%%%% 10- Apply Bandpass Filter for LSB %%%%%%%%%%%%%%%%%%%%
BBF = zeros(size(f));
BBF(f >= 22 & f <= 24) = 1;
BBF(f <= -22 & f >= -24) = 1;

m_modulated_freq2 = m_modulated_freq .* BBF;
m_modulated2 = ifft(ifftshift(m_modulated_freq2)) / dt;
```

9. Lower Side Band

10. The value of C2(t):

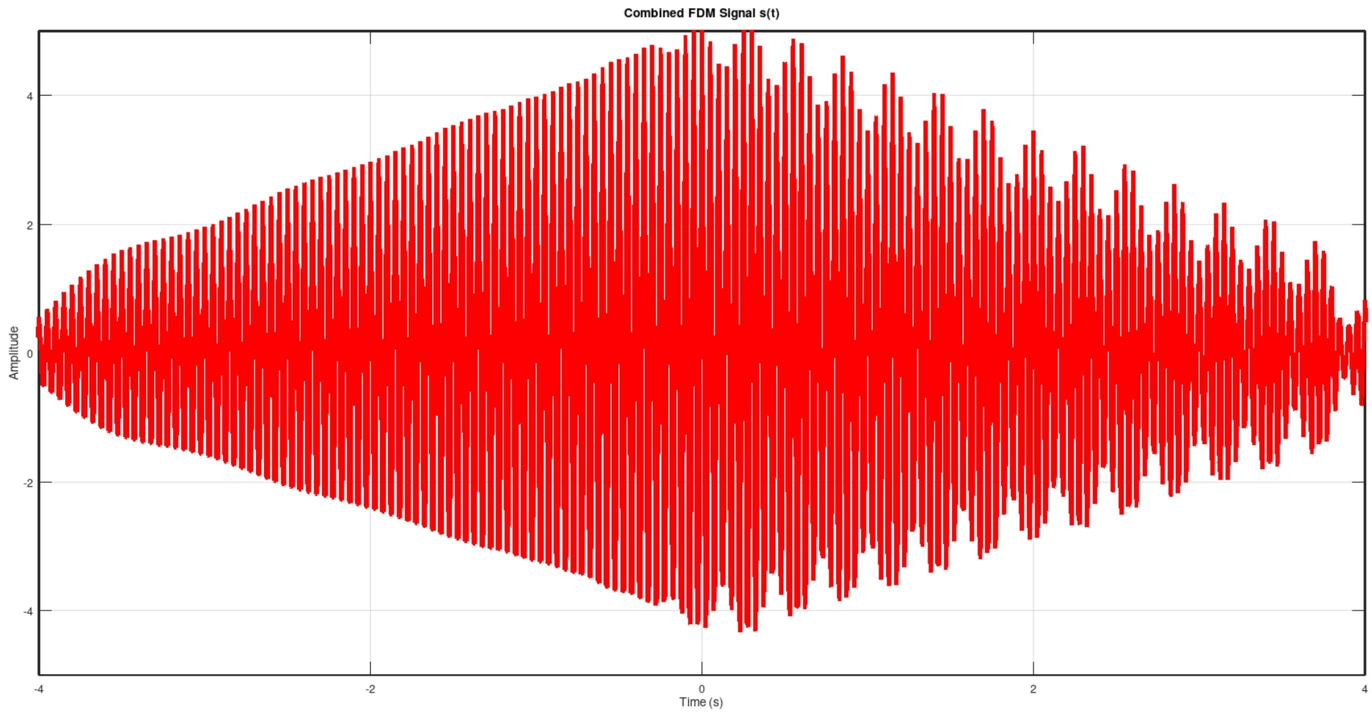
c2 = cos(2*pi*24*t)

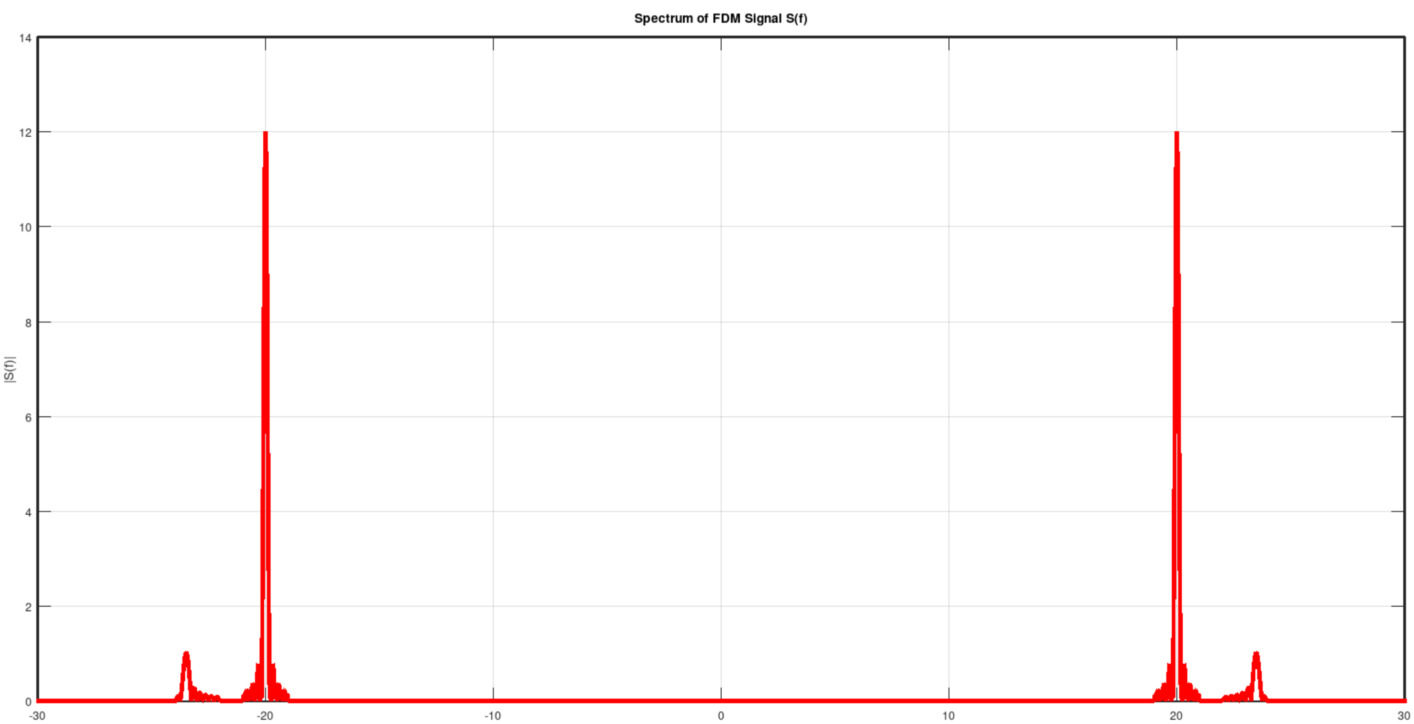
fc2 = fc1 + 2 (guard band) + 2*1 (channels BW)

11. Plot s(t) which is s1(t) + s2(t) then Plot S(f).

```
%%%%%%%%%%%%%%%%%%%%%%%%% 11- Combine and plot FDM signals %%%%%%%%%%%%%%%%%%%%%
s1 = x_modulated;
s2 = m_modulated2;
s_time = s1 + s2;
s_freq = fftshift(fft(s_time) * dt);

figure(12);
plot(t, s_time, 'r', 'LineWidth', 1.5);
title('Combined FDM Signal s(t)');
xlabel('Time (s)');
ylabel('Amplitude');
xlim([-4 4]);
ylim([-5 5]);
grid on;

figure(13);
plot(f, abs(s_freq), 'r', 'LineWidth', 1.5);
title('Spectrum of FDM Signal S(f)');
xlabel('Frequency (Hz)');
ylabel('|S(f)|');
xlim([-30 30]);
grid on;
```

OUTPUT:

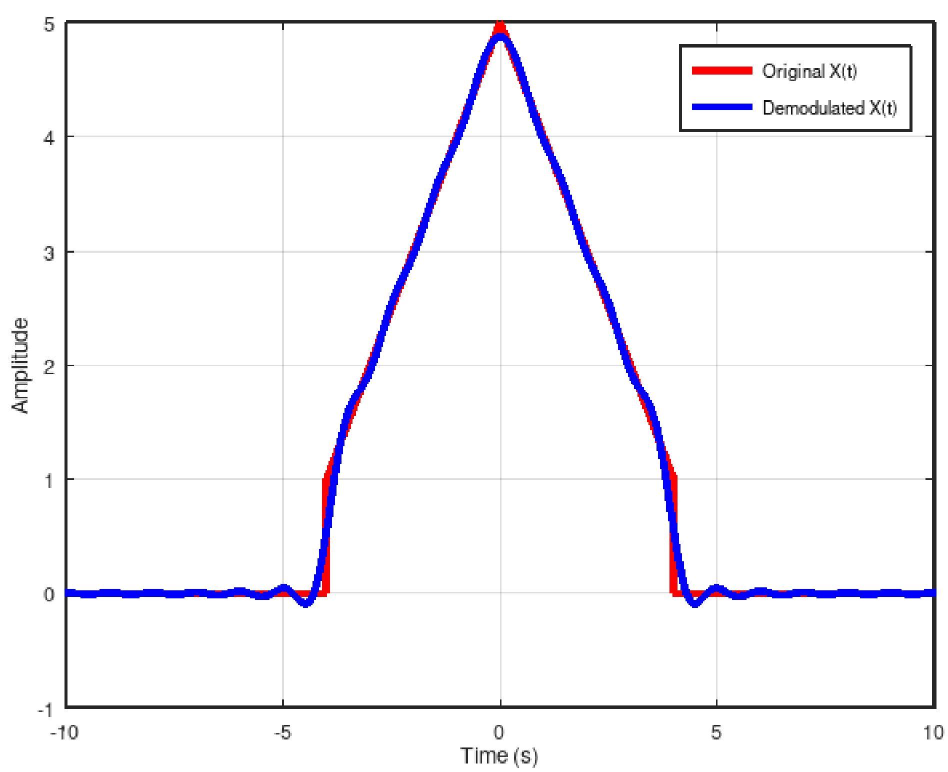

Combined FDM Signal s(t)
Amplitude
Time (s)
4
2
0
-2
-4
-4
-2
0
2
4

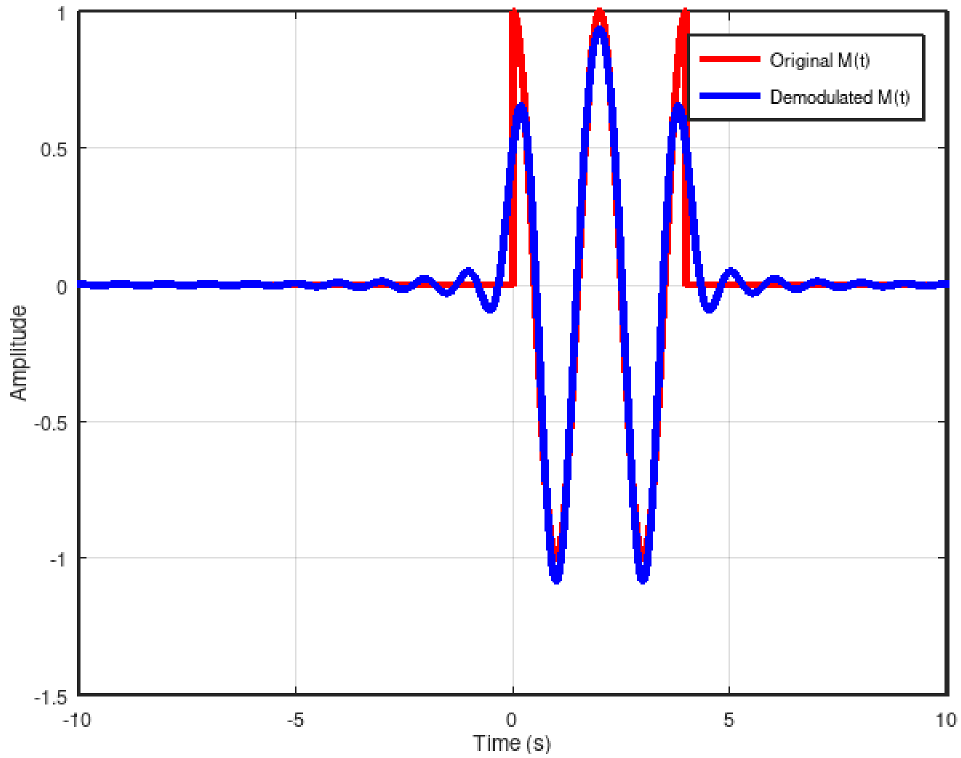

Spectrum of FDM Signal S(f)
|S(f)|
Frequency (Hz)
14
12
10
8
6
4
2
0
-30
-20
-10
0
10
20
30

12. Create a coherent demodulator for each channel and plot the received messages and the input messages on the same figure.

```
%%%%%%%%%%%%%%%%%%%%%%%%% 12- Coherent Demodulation for x(t) %%%%%%%%%%%%%%%%%%%
x_demodulated= s_time .* cos(2 * pi * 20 * t);
x_demodulated_freq = fftshift(fft(x_demodulated)) * dt;

LPF_BW = 1;
H = abs(f) < LPF_BW;
x_rec = 2 * real(ifft(ifftshift(H .* x_demodulated_freq)) / dt);

figure(14);
plot(t, x_time, 'r', 'LineWidth', 1.5);
hold on;
plot(t, x_rec, 'b', 'LineWidth', 1.5);
legend('Original X(t)', 'Demodulated X(t)');
xlabel('Time (s)');
ylabel('Amplitude');
xlim([-10 10]);
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%% 13- Coherent Demodulation for m(t) %%%%%%%%%%%%%%%%%%%
m_demodulated = s_time .* cos(2 * pi * 24 * t);
m_demodulated_freq = fftshift(fft(m_demodulated)) * dt;

H = abs(f) < LPF_BW;
m_rec = 4 * real(ifft(ifftshift(H .* m_demodulated_freq)) / dt);

figure(15);
plot(t, m_time, 'r', 'LineWidth', 1.5);
hold on;
plot(t, m_rec, 'b', 'LineWidth', 1.5);
legend('Original M(t)', 'Demodulated M(t)');
xlabel('Time (s)');
ylabel('Amplitude');
xlim([-10 10]);
grid on;
```

OUTPUT:





Original M(t)
Demodulated M(t)
Amplitude
Time (s)
1
0.5
0
-0.5
-1
-1.5
-10
-5
0
5
10

DIGITAL COMMUNICATION

Part I

Using Octave simulator, you have to develop a code for line coding. Each group will choose two types of line codes and make a comparison between them. Each group should select one of these line codes: AMI, CMI, and Manchester to be compared with one of these codes: unipolar non-return to zero, and polar non-return to zero. Each group must plot the time and spectral domains of the pulses. The students should select the number of bits to be at least 64. This bit stream should be selected to be random, which means that the type of each bit is randomly selected by the program code to be either '1' or '0'.

Using AMI and Polar -NRZ

CODE:

```
% ------------Parameters---------------%
num_bits = 64;
bit_rate = 100;               % bits per second
Fs = 1000;                    % Sampling frequency
Tb = 1/bit_rate;              % Bit period
t = 0:1/Fs:num_bits*Tb - 1/Fs;
% ----------Random bit stream----------%
bits = randi([0 1], 1, num_bits)
% ---------Polar NRZ Encoding----------%
polar_nrz = zeros(1, length(t));
for i_1 = 1:num_bits
    idx_p = (i_1-1)*Fs*Tb + 1:i_1*Fs*Tb;
    if bits(i_1) == 1
        polar_nrz(idx_p) = 1;
    else
        polar_nrz(idx_p) = -1;
    end
end
% -------------AMI Encoding--------------%
AMI =zeros(1, length(t));
no_of_ones=0;
for i_2 = 1:num_bits
    idx_A = (i_2-1)*Fs*Tb + 1:i_2*Fs*Tb;
    if bits(i_2) == 0
        AMI(idx_A) = 0;
    else
     no_of_ones=no_of_ones+1;
     if(rem(no_of_ones,2)==0)
         AMI(idx_A) = -1;
     else
         AMI(idx_A) = 1;
     end
    end
end
```

```matlab
% -----------Plot  AMI Time Domain-----------%
figure (1);
plot(t, AMI, 'LineWidth', 2,'Color','R');
title('AMI Line Code - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');
ylim([-1.5 1.5]);
grid on;
% -----------Plot Polar-NRZ Time Domain-----------%
figure(2)
plot(t,polar_nrz, 'LineWidth', 2,'color','b');
title('Polar-NRZ Line Code - Time Domain');
xlabel('Time (s)');

ylabel('Amplitude');
ylim([-1.5 1.5]);
grid on;
```

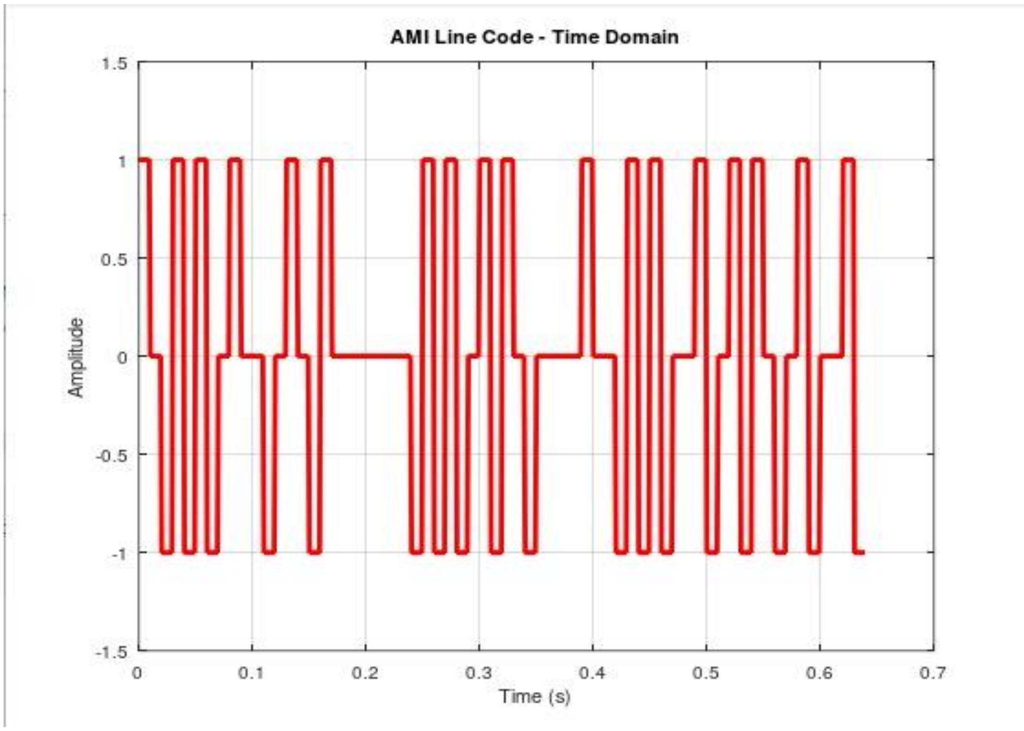
```matlab
% -----------Plot AMI Frequency Domain------------%
N = length(AMI);
f = (-N/2:N/2-1)*(Fs/N);
AMI_fft = fftshift(abs(fft(AMI)/length(AMI)));
figure(3);
plot(f, AMI_fft, 'LineWidth', 1.5);
title('AMI Line Code - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
% ----------Plot Polar-NRZ Frequency Domain--------%
N = length(polar_nrz);
f_2 = (-N/2:N/2-1)*(Fs/N);
polar_nrz_fft = fftshift(abs(fft(polar_nrz)/length(polar_nrz)));
figure(4);
plot(f_2, polar_nrz_fft, 'LineWidth', 1.5);
title('Polar-NRZ Code - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

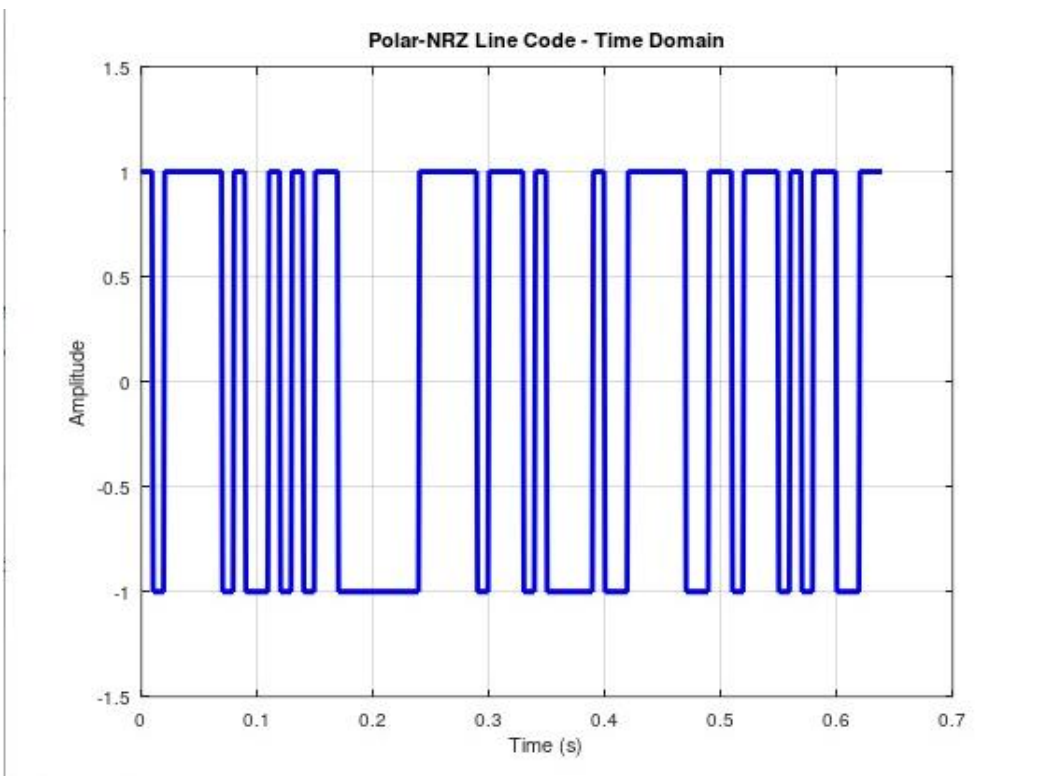
BIT STREAM

1 0 1 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 1 1 0 1 1 1 0 10 0 0 0 1 0 0 1 1 1 1 1 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 1 1

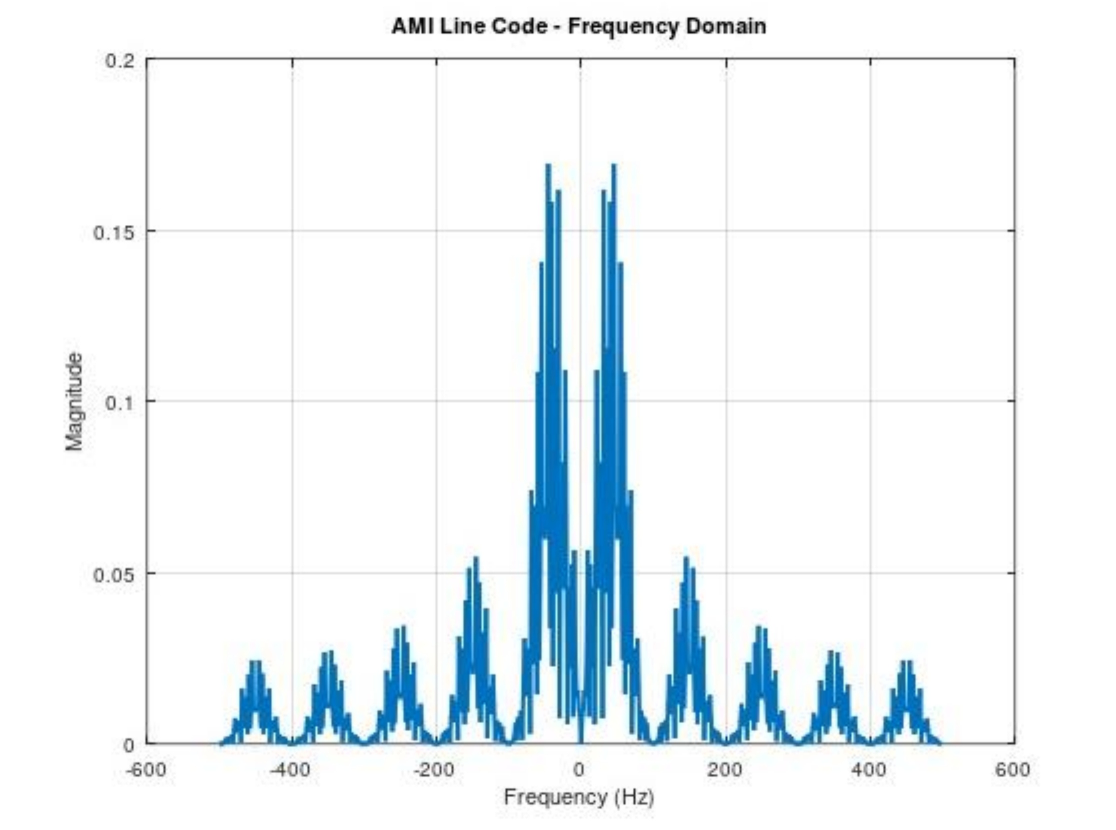
OUTPUTS

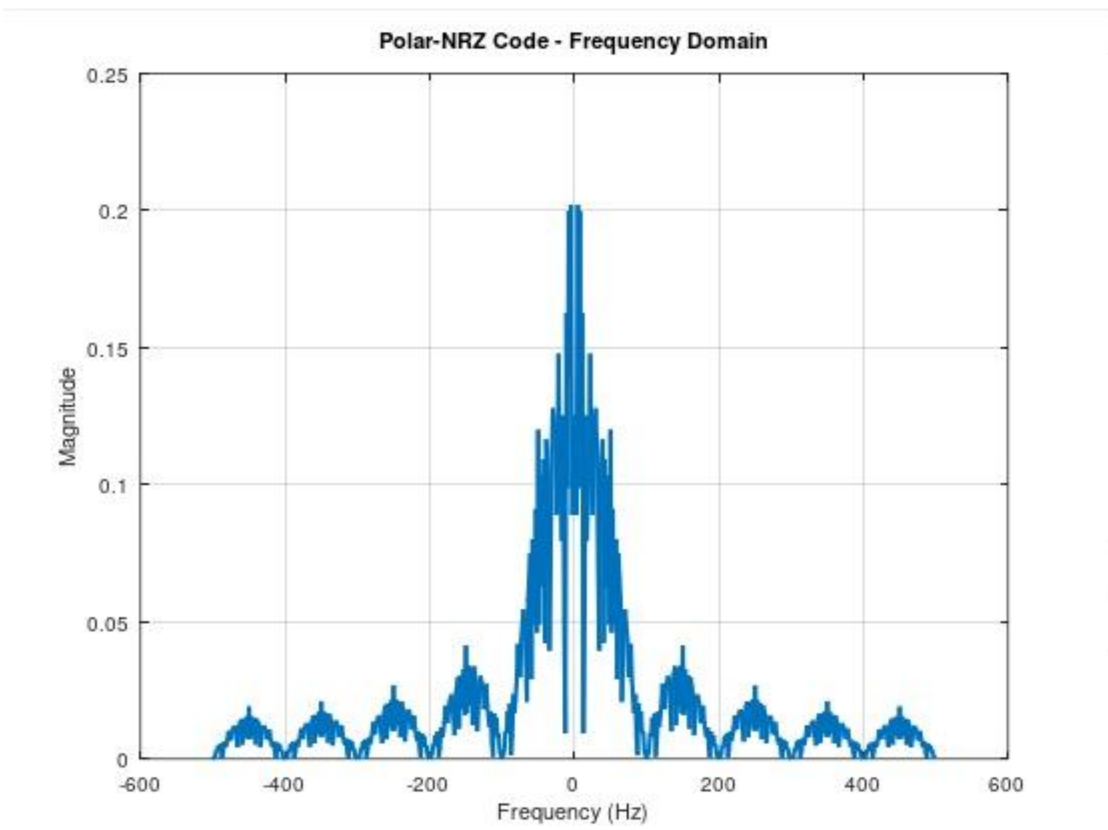
TIME DOMAIN





FREQUENCY DOMAIN





COMPARISON BETWEEN AMI AND POLAR-NRZ

<table>
<tr><th>Line coding / Comparison points</th><th>AMI</th><th>POLAR-NRZ</th></tr>
<tr><td>time domain</td><td>Three-Level Signal:
Levels: +V, 0, and –V.
Binary ‘0’ is encoded as 0 volts.
Binary ‘1’ is encoded alternately as +V and –V
Error Detection Capability</td><td>Signal Levels
Binary ‘1 ‘→ +V
Binary ‘ 0’ → –V
No inherent error detection capability.</td></tr>
<tr><td>frequency domain</td><td>Zero DC Component
Bandwidth =Rb</td><td>May have a significant DC component
Bandwidth =Rb</td></tr>
</table>

ADVATAGES AND DISADVATAGES

- **AMI (BIPOLAR RZ):**
    - ➢ More complex technique than Unipolar NRZ (As it has three voltage levels) (**Cons**).
    - ➢ Error detection (More immune to error) (**Pros**).
    - ➢ Does not have DC power component (**Pros**).
- **POLAR –NRZ**
    - ➢ Simpler implementation compared to AMI (uses only two voltage levels) **(Pros).**
    - ➢ Poor synchronization (No transitions in long runs of same bits) **(Cons)**
    - ➢ May have DC power component (causes issues in AC-coupled systems) **(Cons).**

Part II

Each group develops a program code for the transmitter and coherent receiver of either of ASK, FSK and PSK systems. The baseband data can be the random data explained above. The carrier frequency should be selected to be higher than the bit rate. The student must plot the temporal and spectrum of the signal at outputs of the transmitter and the receiver. In the receiver, each group should select the oscillator phase to be 30°, 60°, and 90°, and comment of the results each group will get.

CODE

```
% ---------------------Parameters-------------------%
bit_rate = 1000;                 % bits per second
fc = 5000;                       % Carrier frequency (Hz)
num_bits = 64;                   % Number of bits
samples_per_bit = 100;           % Oversampling
fs = bit_rate * samples_per_bit; % Sampling frequency
T = 1 / bit_rate;
t = 0:1/fs:num_bits*T - 1/fs;
% -------------------random binary data--------------------%
data_bits = randi([0 1], 1, num_bits)
data_upsampled = repelem(data_bits, samples_per_bit);
%  -------------------ASK Modulation-------------------%
carrier = cos(2*pi*fc*t);
ask_signal = data_upsampled .* carrier;
% -------------Time-Domain Plot (Transmitter Output)-------------%
figure;
plot(t(1:1000), ask_signal(1:1000), 'LineWidth', 1.5); % Range from 1 to 1000 to zoom in
title('ASK Signal at Transmitter (Time Domain)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
% -----------Frequency-Domain Plot (Transmitter Output)------------%
n = length(ask_signal);
df=fs/n;
if (rem(n,2)==0)
     f = (-fs/2):df:(fs/2-df);
else
     f = -(fs/2-df/2):df:(fs/2-df/2);
end
ASK_fft = fftshift(abs(fft(ask_signal))/n);
figure;
plot(f, ASK_fft, 'LineWidth', 1.5);
title('Spectrum of ASK Signal at Transmitter');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 2*fc]);
grid on;
```

```
% ---------Coherent Receiver: Demodulation with Phase Offsets-------------%
phases = [30, 60, 90];
for i = 1:length(phases)
    phase_deg = phases(i);
    phase_rad = deg2rad(phase_deg);
    % receiver_carrier with phase offset
    receiver_carrier = cos(2*pi*fc*t + phase_rad);
    % Coherent detection
    received = ask_signal .* receiver_carrier;
    % Integrate and decision
    demod_bits = zeros(1, num_bits);
    for i = 1:num_bits
        idx_start = (i-1)*samples_per_bit + 1;
        idx_end = i*samples_per_bit;
        segment = received(idx_start:idx_end);
        avg_val = mean(segment);
        demod_bits(i) = avg_val > 0.15;  % simple thresholding
    end
    % Plot demodulated signal (partial for visualization)
    figure;
    plot(t(1:1000), received(1:1000), 'LineWidth', 1.5);
    title(['Demodulated Signal (Phase = ', num2str(phase_deg), '°)']);
    xlabel('Time (s)');
    ylabel('Amplitude');
    grid on;
```

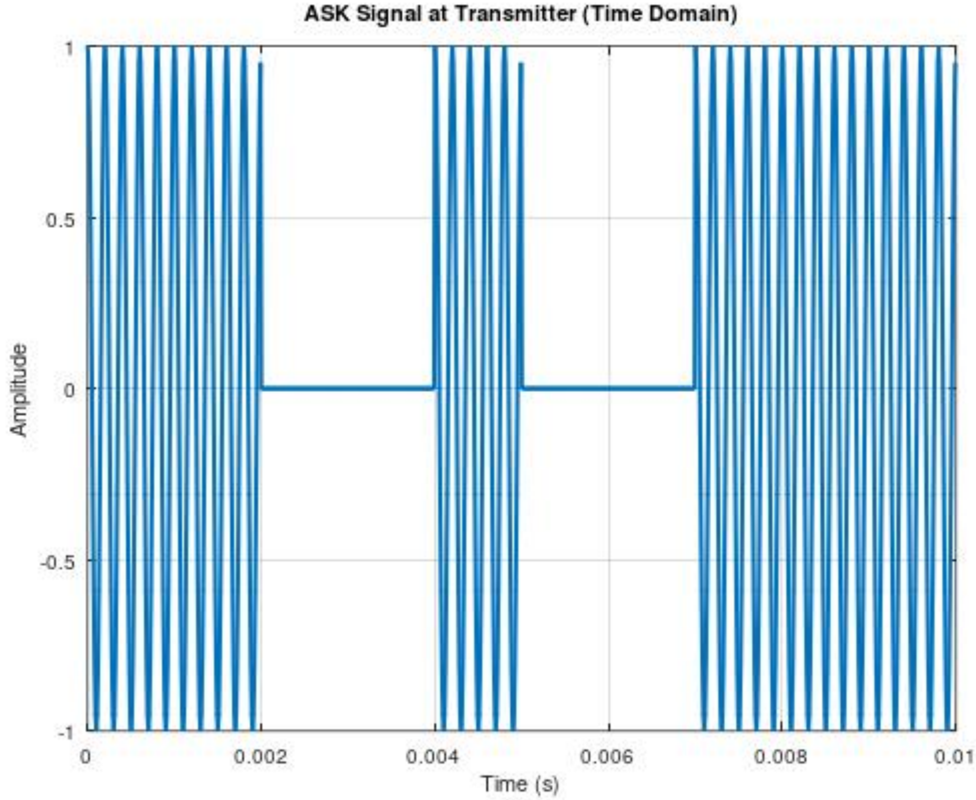
```
    % -----------Frequency-Domain Plot (Receiver Output)------------%
    received_fft = fftshift(abs(fft(received)) / n);
    figure;
    plot(f, received_fft, 'LineWidth', 1.5);
    title(['Spectrum of Received Signal - Phase = ', num2str(phase_deg), '°']);
    xlabel('Frequency (Hz)');
    ylabel('Magnitude');
    grid on;
  end
```

BIT STREAM

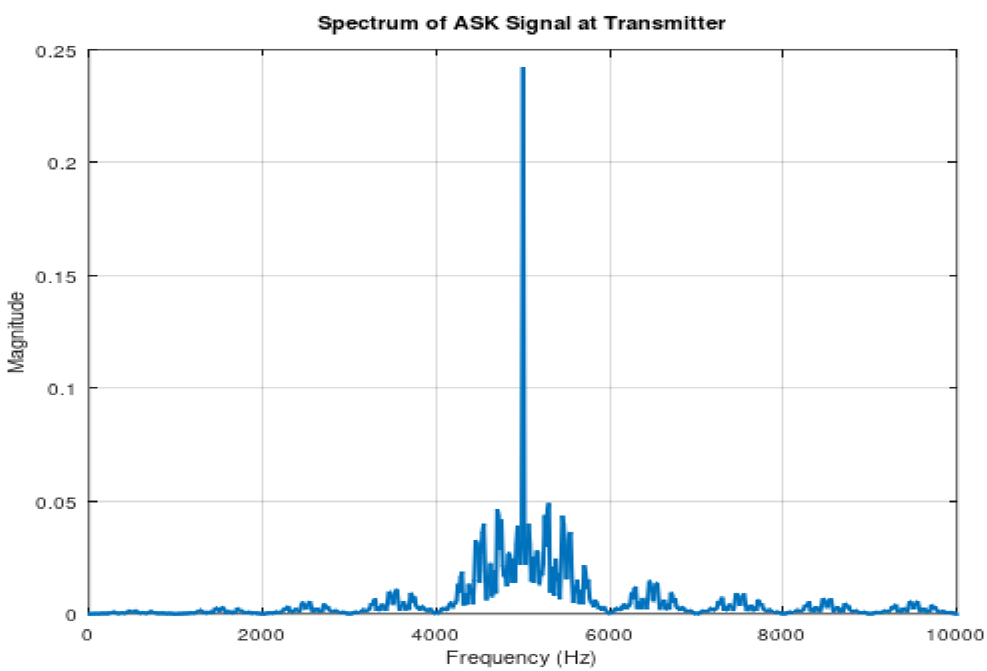
1 1 0 0 1 0 0 1 1 1 0 1 0 1 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 0 10 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0 0 0 1 1 1 0 1 1 0 1 1 1 0

OUTPUTS

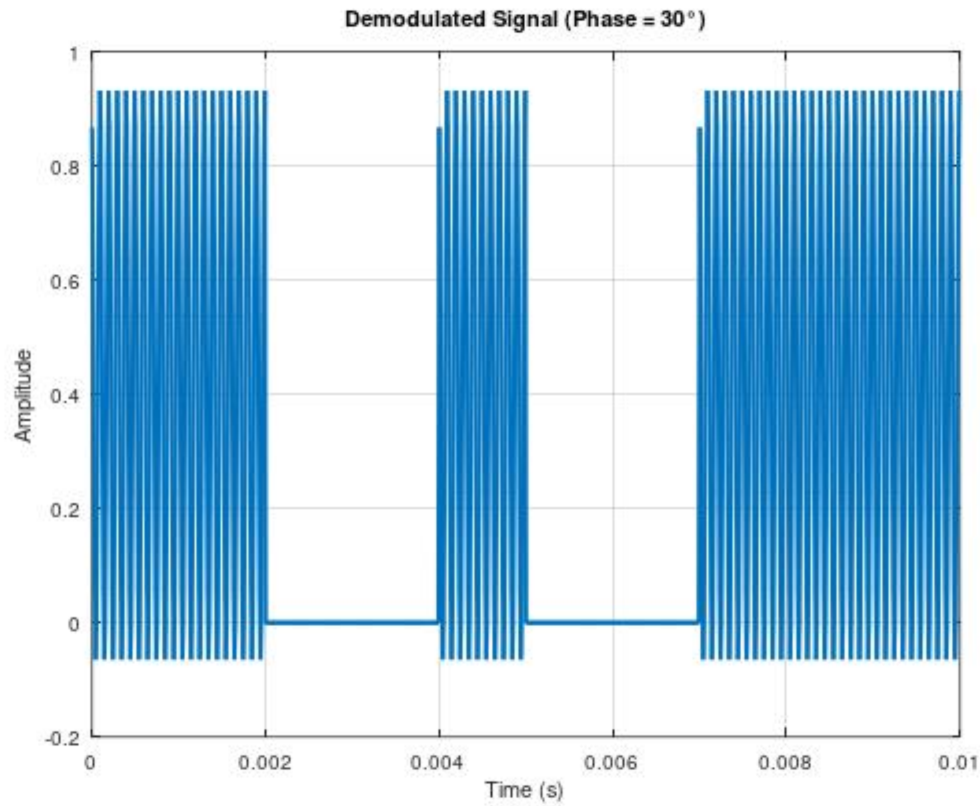
TIME DOMAIN TX

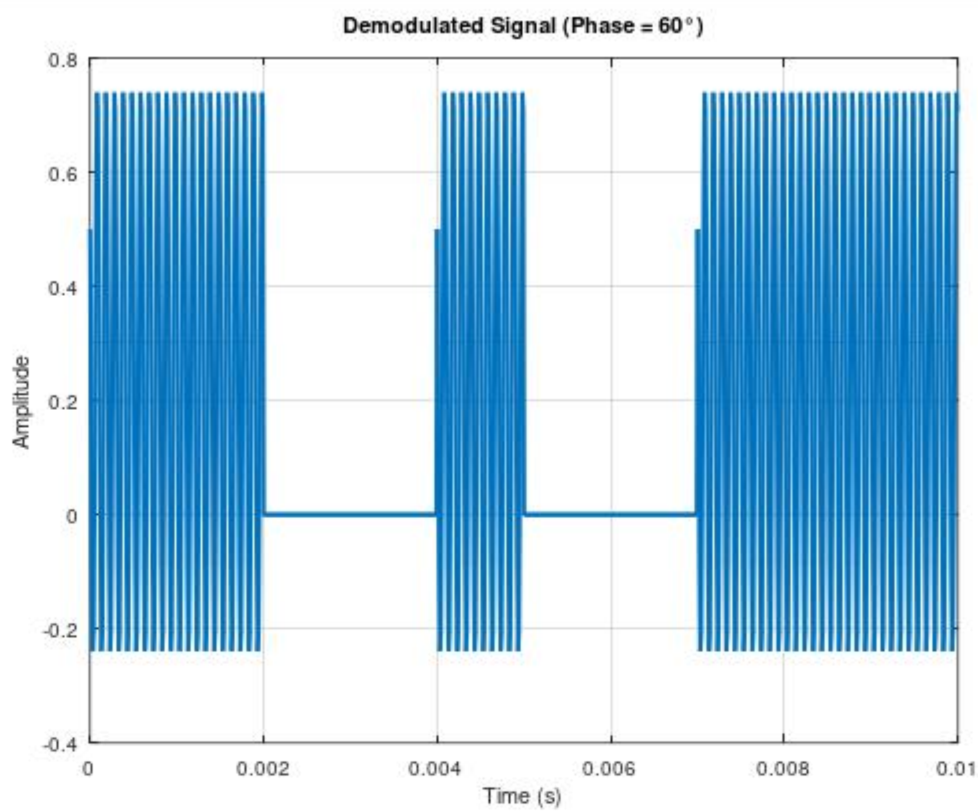


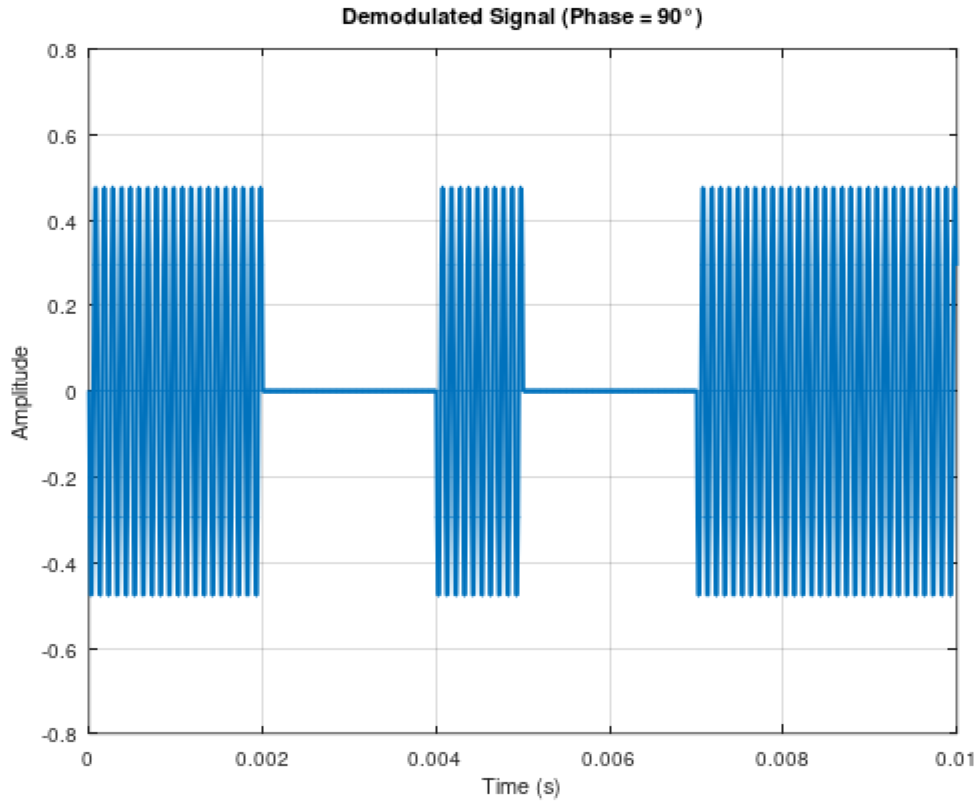
FREQUENCY DOMAIN TX



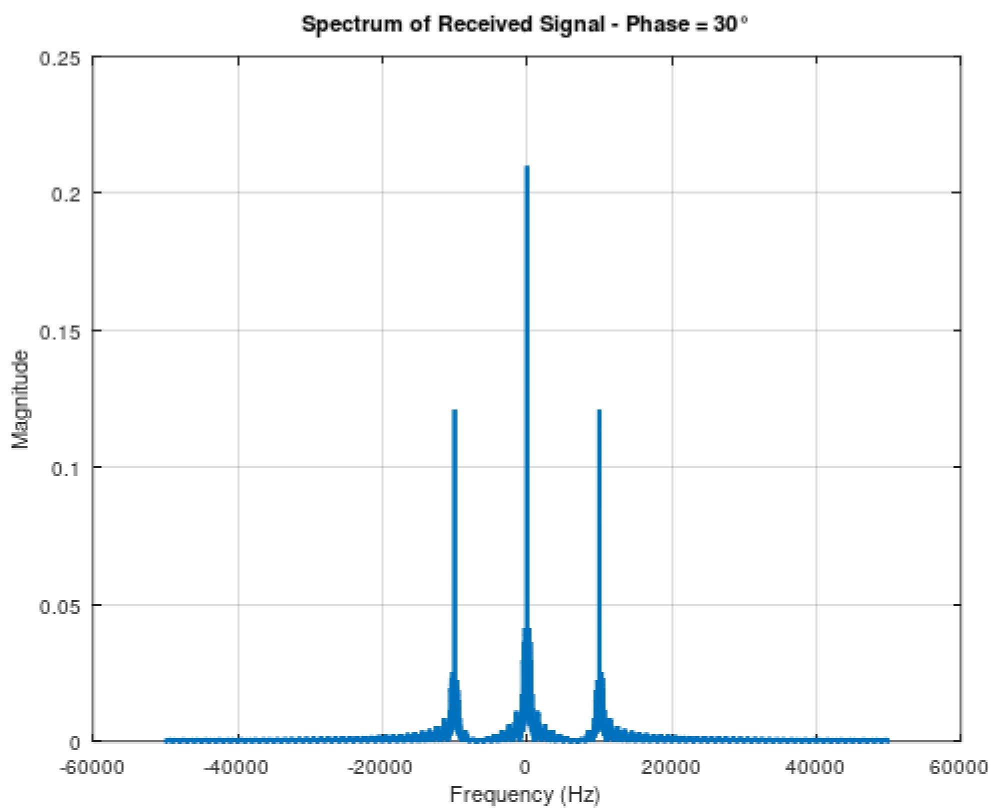
TIME DOMAIN RX

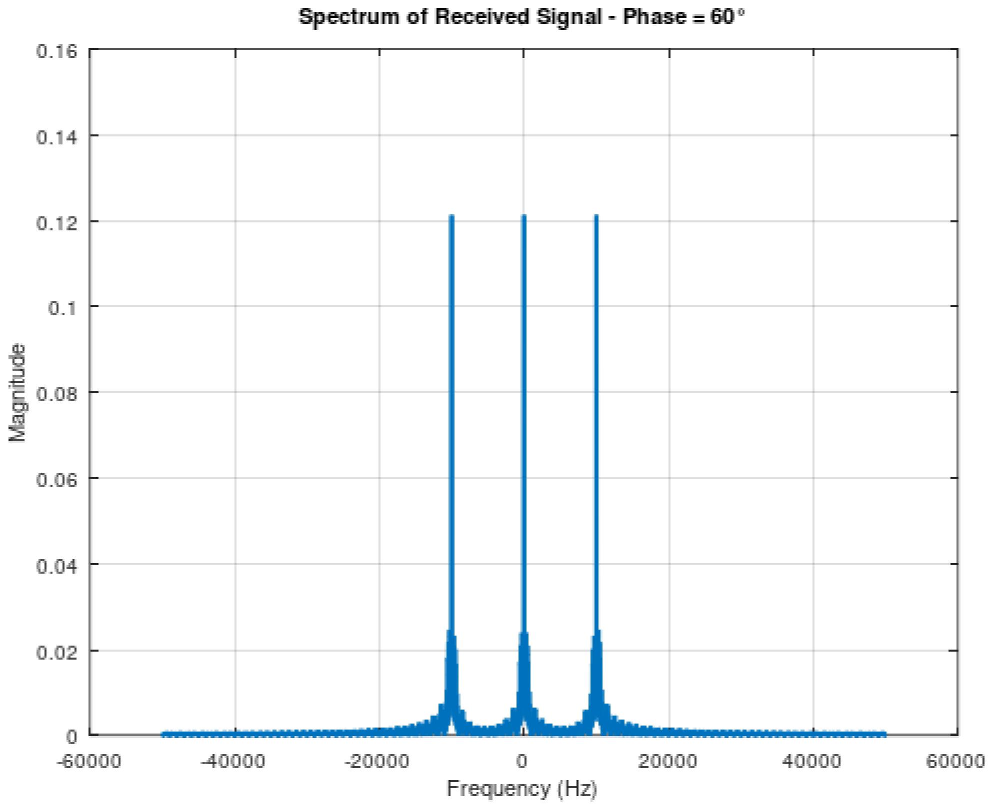




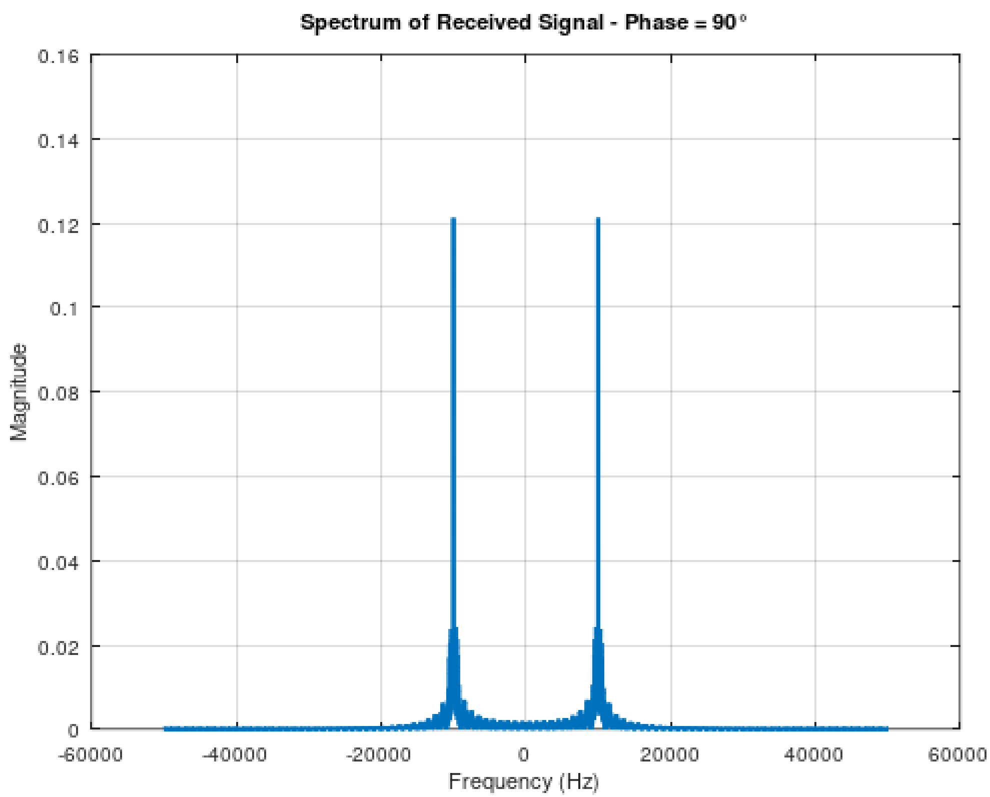


FREQUENCY DOMAIN RX





Spectrum of Received Signal - Phase = 60°
Magnitude
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-60000
-40000
-20000
0
20000
40000
60000
Frequency (Hz)



Spectrum of Received Signal - Phase = 90°
Magnitude
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-60000
-40000
-20000
0
20000
40000
60000
Frequency (Hz)

NOTE

At line 17 it contain plot range, we made it to zoom in in the figure, if we remove it

```
17  plot(t(1:1000), ask_signal(1:1000), 'LineWidth', 1.5); % Range from 1 to 1000 to zoom in
```

This is the new output:

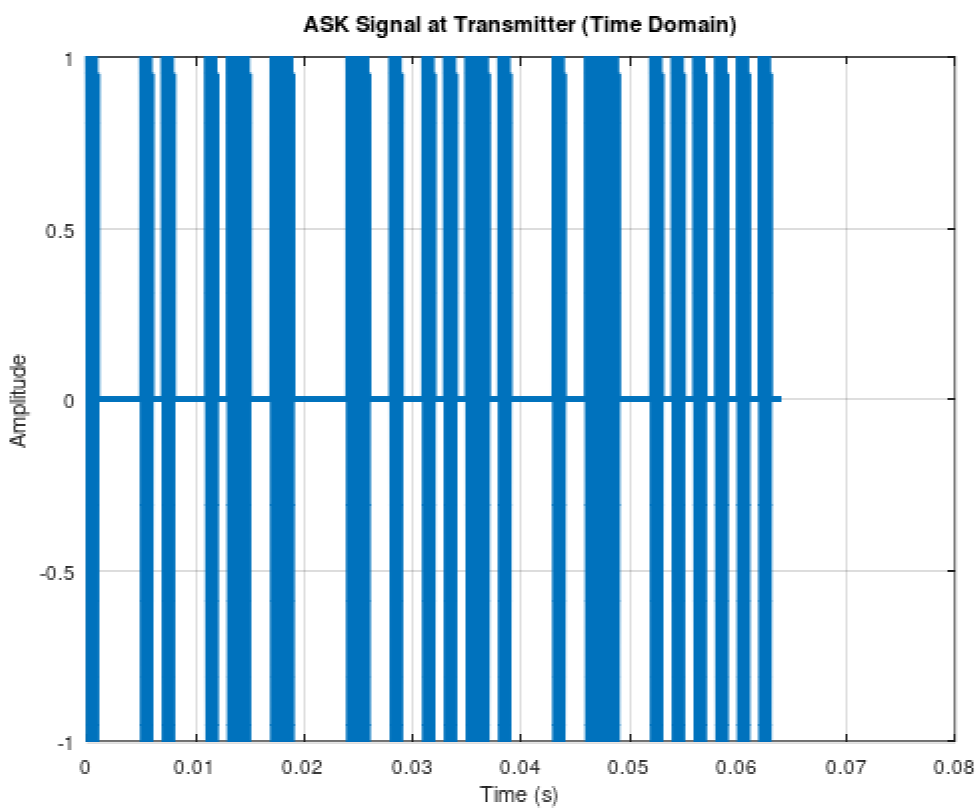
```
17  plot(t, ask_signal, 'LineWidth', 1.5);
```

THE NEW REGENERATED RANDOM BITSTREAM

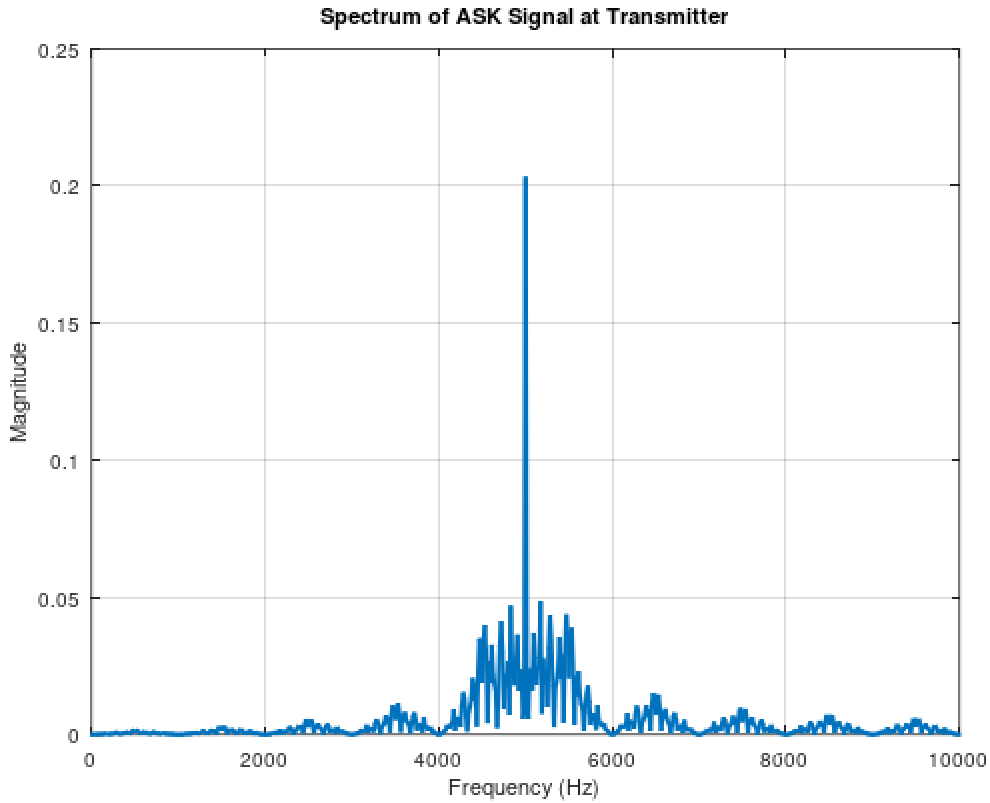
1 0 0 0 0 1 0 1 0 0 0 1 0 1 1 0 0 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 01 1 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0

THE NEW OUTPUTS

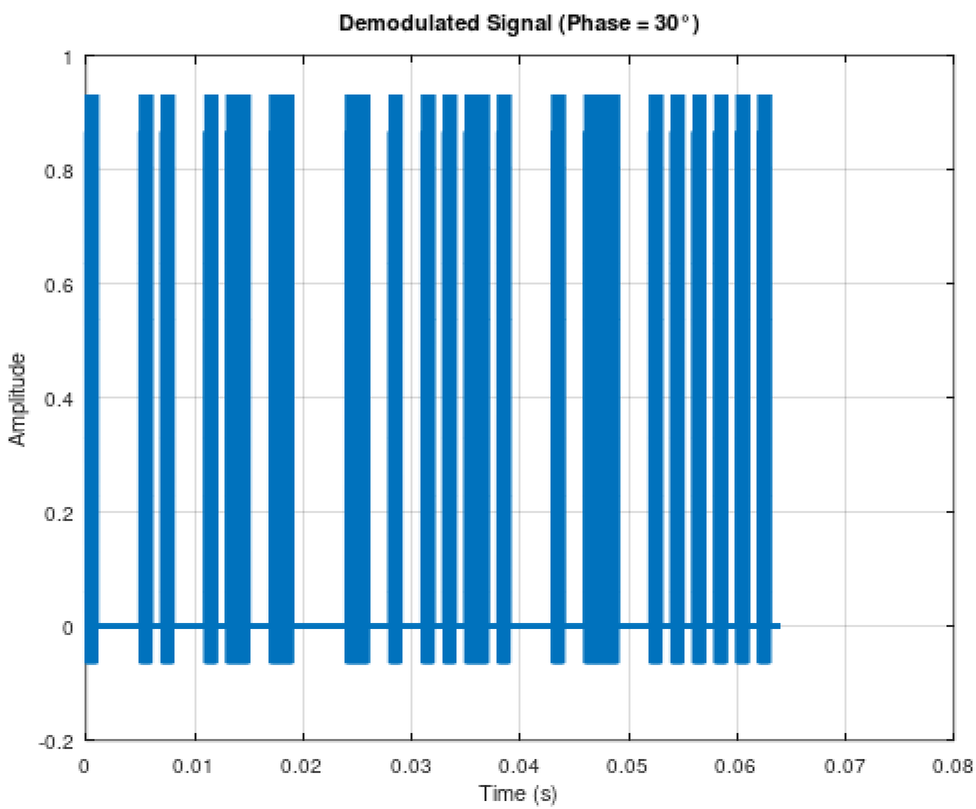
TIME DOMAIN TX

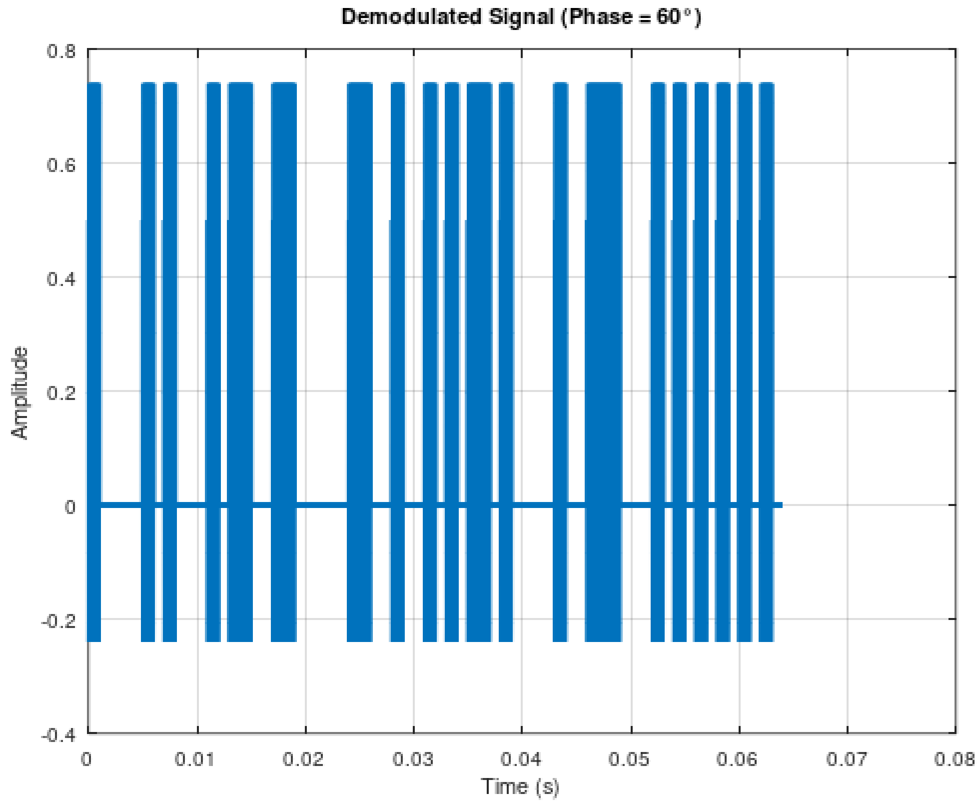


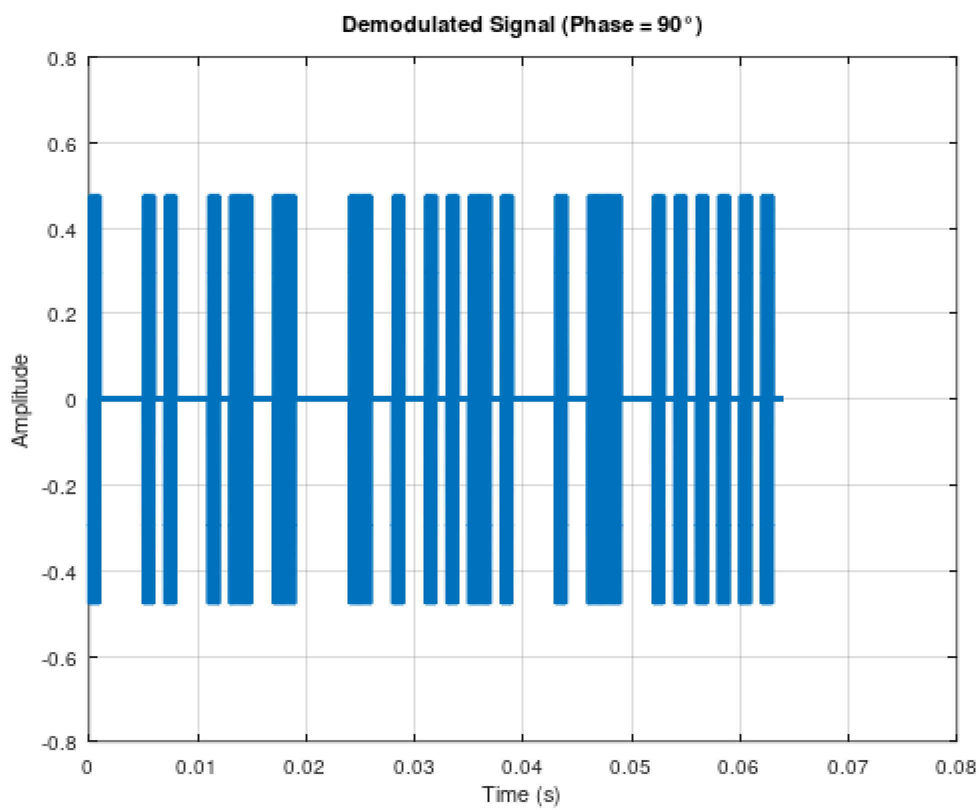
FREQUENCY DOMAIN TX



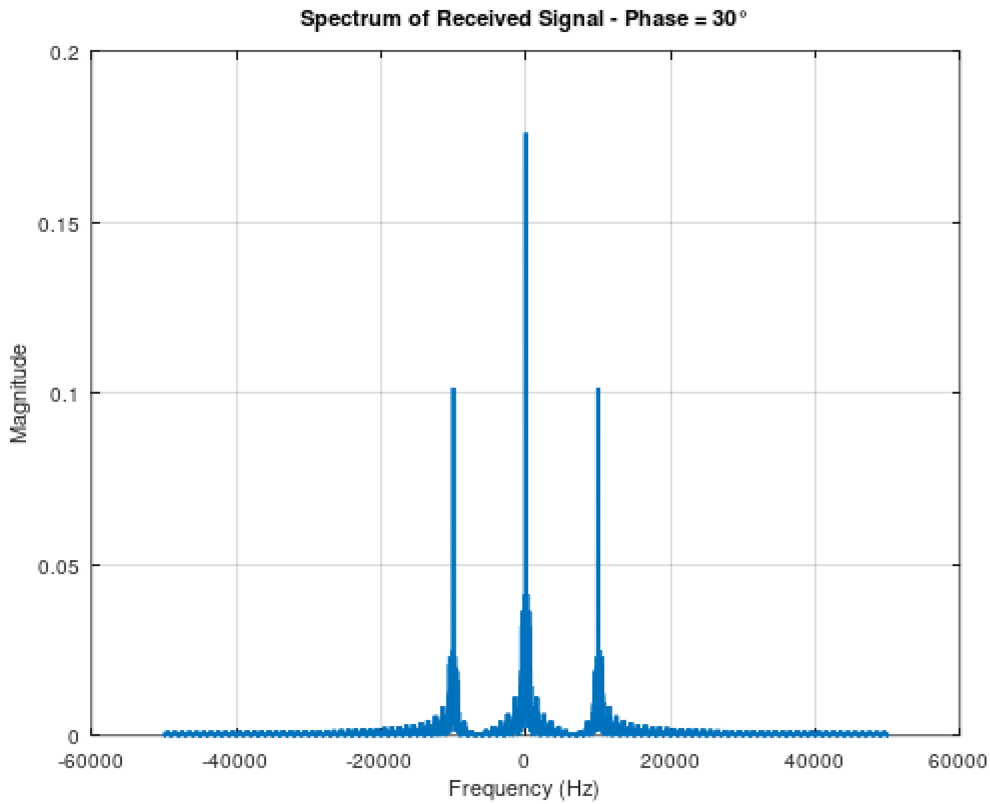
TIME DOMAIN RX

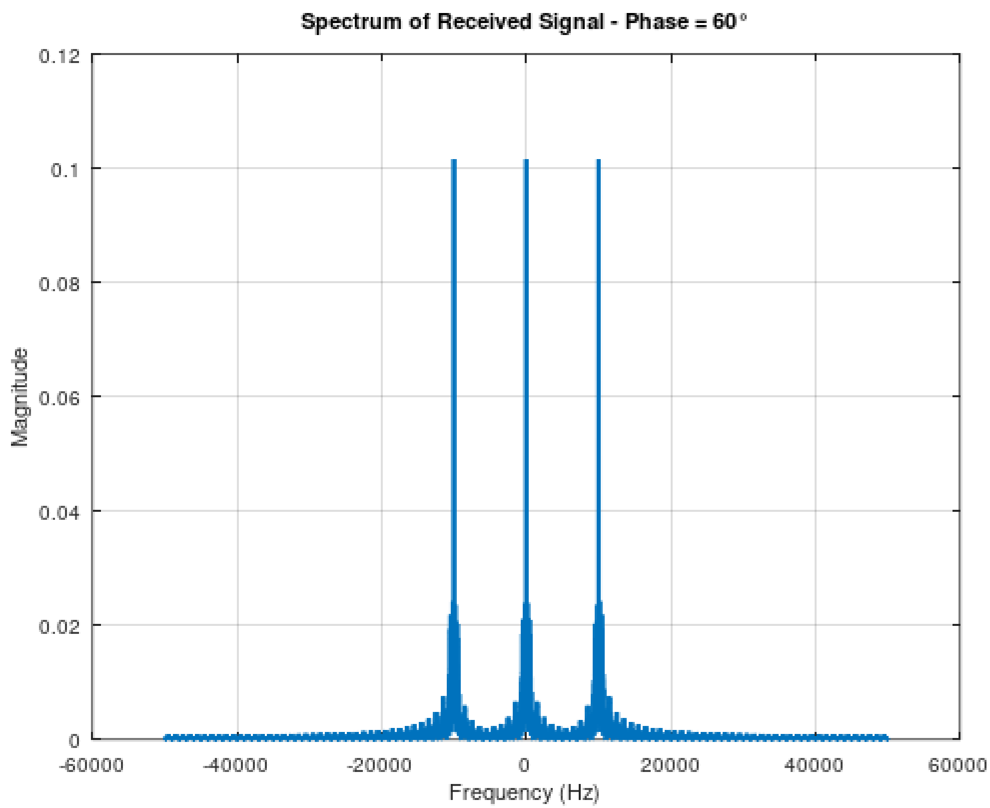


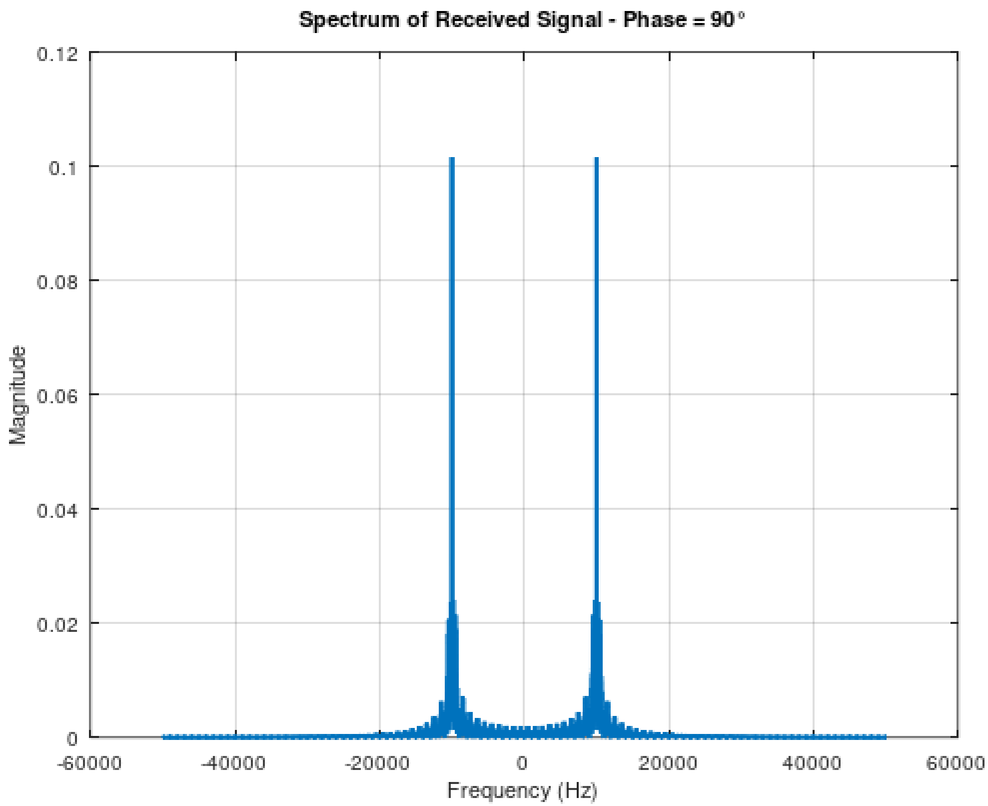

Demodulated Signal (Phase = 60°)
Amplitude
Time (s)
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08


Demodulated Signal (Phase = 90°)
Amplitude
Time (s)
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08

FREQUENCY DOMAIN RX







COMMENTS

• 90-Degree Phase Offset (ϕ = 90°):

- ➢ The demodulated signal amplitude is significantly reduced, approaching zero.
- ➢ Maximum distortion occurs, making data recovery virtually impossible.

Conclusion:

- ➢ Phase offsets between the transmitter and receiver carrier signals directly affect the performance of coherent ASK demodulation.
- ➢ Increasing phase offsets lead to signal attenuation and increased distortion, leading to errors in data recovery.