Benha University
Benha Faculty of Engineering
Electrical Engineering Department
Communications and Computers Engineering
Communication systems II – E1415
Fall 2024

Performance Analysis of Digital Modulation Techniques (BPSK, BFSK, and QASK) under AWGN Channels

Project documentation

submitted in partial fulfillment of the requirements
for the E1415 – communication systems 2 course

Name	Section
Mahmoud Saad Mostafa	3

Under supervision of

Dr. Ashraf Yahia Hassan
Professor of Electrical Engineering Department,
Faculty of Engineering, Benha University

Fall 2024

Aim of the Project: The primary objective of this project is to design and simulate multiple digital modulation schemes, including binary phase shift keying (BPSK), binary frequency shift keying (BFSK), and quadrature amplitude shift keying (QASK), using MATLAB. The performance of these modulation schemes is evaluated under different noise conditions by comparing theoretical and simulated Bit Error Rates (BER).

Description of Modulator and Demodulator:

BPSK Modulator: The BPSK modulator maps binary data (0 and 1) into two distinct phases: a positive phase for bit "1" and a negative phase for bit "0". This mapping is achieved using a cosine carrier multiplied by the baseband signal.

BFSK Modulator: The BFSK modulator represents binary data by modulating the signal frequency. Two distinct frequencies are used: one for bit "0" and another for bit "1".

QASK Modulator: The QASK modulator maps pairs of binary data into amplitude-modulated in-phase and quadrature components. Different amplitude combinations represent different bit pairs.

Demodulators: Each modulation scheme has a corresponding demodulator:

BPSK Demodulator recovers the binary data by observing the polarity of the received signal and applying a decision threshold at zero.

BFSK Demodulator detects the transmitted bit by determining the frequency of the received signal.

QASK Demodulator detects pairs of binary data by extracting the in-phase and quadrature components and mapping them back to the respective bit pairs.

Simulation Software Utilized:

MATLAB: All coding and simulations were performed using MATLAB R2023a.

BPSK Procedures:

1. **Transmitter:**
   - Generate a random binary data stream.
   - Map binary data to baseband modulated data using BPSK principles.
   - Generate a carrier signal and modulate the baseband data to produce the passband signal.
2. **Channel:**
   - - Add AWGN (Additive White Gaussian Noise) to the transmitted passband signal at various SNR levels (5, 10, 15, and 20 dB).
3. **Receiver:**
   - Demodulate the received signal by multiplying it with the carrier signal and applying a threshold detector.
   - Recover the transmitted binary data.
4. **Comparison:**
   - Compute the theoretical BER for BPSK using the equation:
     BER theoretical = (1/2) * erfc(sqrt(10^(SNR/10))).
   - Compare theoretical and simulated BER by plotting both on a semi-log graph.
5. **Code Implementation:**

The MATLAB code was written to implement the above procedures :

```
%BPSK Modulation and Demodulation Simulation
clear all
clc
%% parameters
T=1; % assume one bit /sec
N = 10; % Number of bits
```

```
data = randi([0,1],1,N); % Generate random binary stream [ generata
integer values from [0,1]  in matrex 1*N ]
vp=1;% vp is peak voltage of NRZ
fs=1000 ;% number of samples per bit
u= fs*N; % total samples
t = 0:1/(fs):(N-(1/(fs)));   % Time vector t = 0 : 0.001 : N-0.001
fc = 5 ; % Carrier frequency
f = -fs/2:1/N:fs/2-1/N; % freq vector samples
k=1;  % bit per symbol


%% Transmitter

% baseband [ output NRZ 0 --> -1 and 1 --> 1 ]
baseband_modulated=[];
for i =1:N
    if data(i)==1
        baseband_modulated=[baseband_modulated ones(1,fs)*(vp)];
    elseif data(i)==0
        baseband_modulated=[baseband_modulated ones(1,fs)*(-vp)];
    end
end

% Passband modulation
carrier = (sqrt(2/T))*cos(2*pi*fc*t); % Carrier signal
passband_modulated = baseband_modulated .* carrier; % Passband
modulation

% bit stream impulses plot
subplot(3,1,1);
stem(data);
xlabel('Time (sec)');
ylabel('Amplitude (volts)');
title('impulses of bits to be transmited ');
ylim([-2 2]);
grid on;
% output of NRZ ploting

subplot(3,1,2);
plot(baseband_modulated);
xlabel('Time (mel sec)');
ylabel('Amplitude (volts)');
title('generated NRZ signal');
ylim([-2 2]);
grid on;

% Plot passband modulated signal and baseband modulated data

subplot(3,1,3);
plot(t,passband_modulated);
```

```
title('Passband Modulated Signal');
xlabel('Time(sec)');
ylabel('Amplitude');
ylim([-2 2]);
grid on;


% Constellation diagram of baseband modulated data
scatterplot(baseband_modulated);
title('Constellation Diagram of Baseband Modulated Data for BPSK');
xlabel('In-Phase Amplitude');
ylabel('Quadrature Amplitude');
grid on;

%{
subplot(2,1,2);
plot(t,baseband_modulated);
title('Baseband Modulated Data');
xlabel('Time');
ylabel('Amplitude');
ylim([-2 2]);
grid on;
%}
% Spectrum of passband modulated signal
figure;
spectrum = fftshift(fft(passband_modulated));
plot(f,abs(spectrum)/N);
title('Spectrum of Passband Modulated Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([-20 20]);
grid on;

% Bandwidth estimation
[bw,flo,fhi,power] = obw(passband_modulated,fs);
%bw = obw(passband_modulated,fs);
bw_estimate = bw;
disp(['Estimated Bandwidth of the Passband Modulated Signal: ',
num2str(bw_estimate), ' Hz']);
%% channel and receiver
% channel
EbN0_range = 0:5:20; % Eb/N0 range
%BER = zeros(1,length(EbN0_range)); % Placeholder for Bit Error
Rate
BER =[];
SNR=[];
for i = 1:length(EbN0_range)
    % Add AWGN
    SNR(i) = EbN0_range(i) + 10*log10(k); % Convert Eb/N0 to SNR
    received_signal = awgn(baseband_modulated,SNR(i));
```

```
    figure;
    subplot(2,1,1);
    plot(t,received_signal);
    title(['Received Signal at SNR = ' num2str(SNR(i)) 'db']);
    xlabel('Time (sec)');
    ylabel('Amplitude');
    ylim([-2 2]);

    % detector
    detected=sign(received_signal);
    sample_index=1:fs:length(received_signal);
    detected_bit=detected(sample_index) > 0 ;
    %figure;
    subplot(2,1,2);
    stem(detected_bit);
    title('Impulses of Received bits');
    xlabel('Time (seconds)-->');
    ylabel('Amplitude (volts)');
    ylim([-2 2]);
    % Calculate Bit Error Rate
    errors = sum( data ~= detected_bit);
    BER = [BER (errors/N)+10e-10] ;
end
%%  plot Theoretical and BER with EbN0
% Theoretical BER calculation
theory_BER = 0.5*erfc(sqrt(10.^(EbN0_range/10)));
% Plot BER vs Eb/N0
figure;
semilogy(EbN0_range, BER, 'o-', 'DisplayName', 'Simulated BER');
hold on;
semilogy(EbN0_range, theory_BER, 'r--', 'DisplayName', 'Theoretical
BER');
title('Bit Error Rate vs Eb/N0');
xlabel('Eb/N0 (dB)');
ylabel('Bit Error Rate');
ylim([10e-10  10e1])
grid on;
legend;
```

- Transmitter: BPSK baseband modulation, carrier generation, and signal visualization.
- Channel: AWGN addition and signal noise plotting.
- Receiver: Demodulation, BER calculation, and comparison.

Figures:

Impulses & Baseband Signal (BPSK) & passband signal :

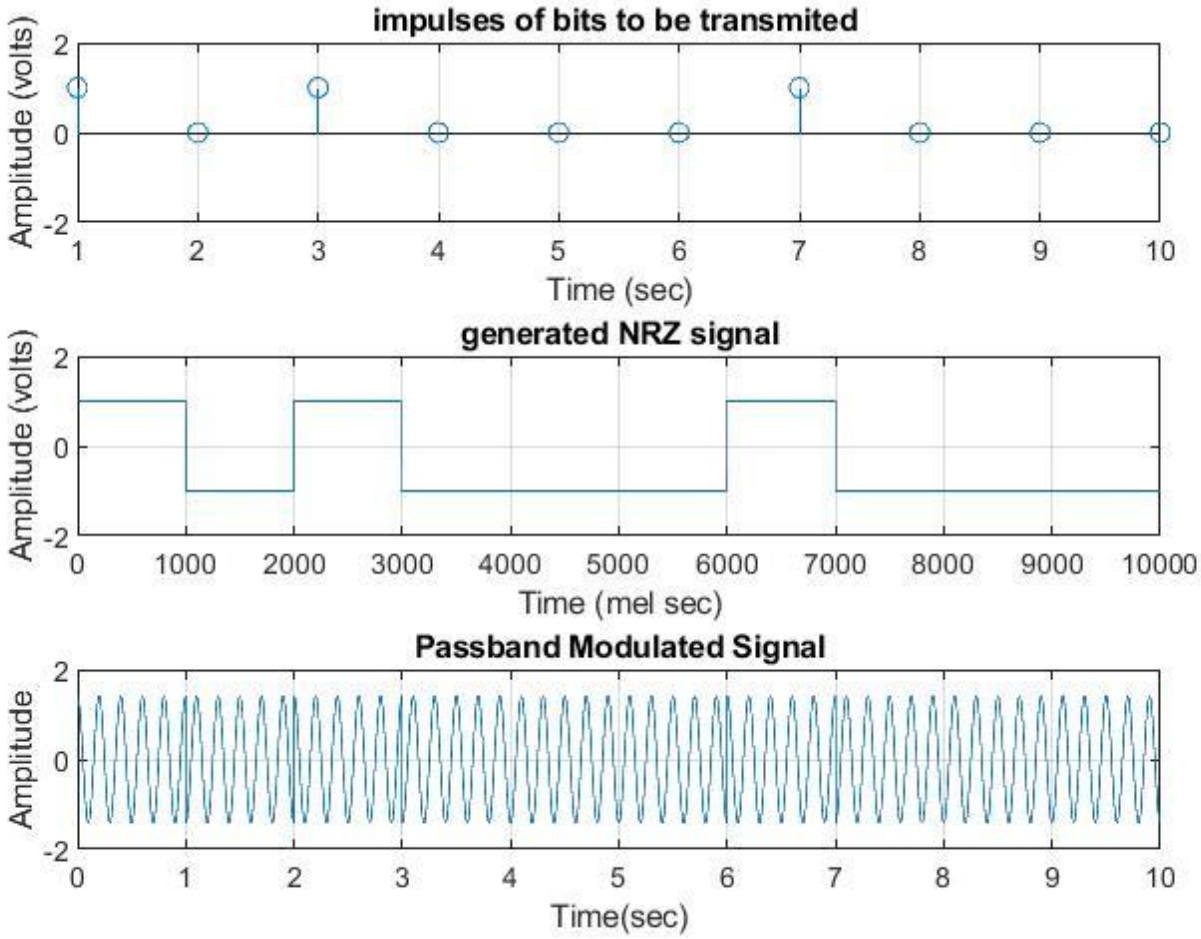


Figure 1: Displays Impulses data and the baseband modulated data and passband signal.

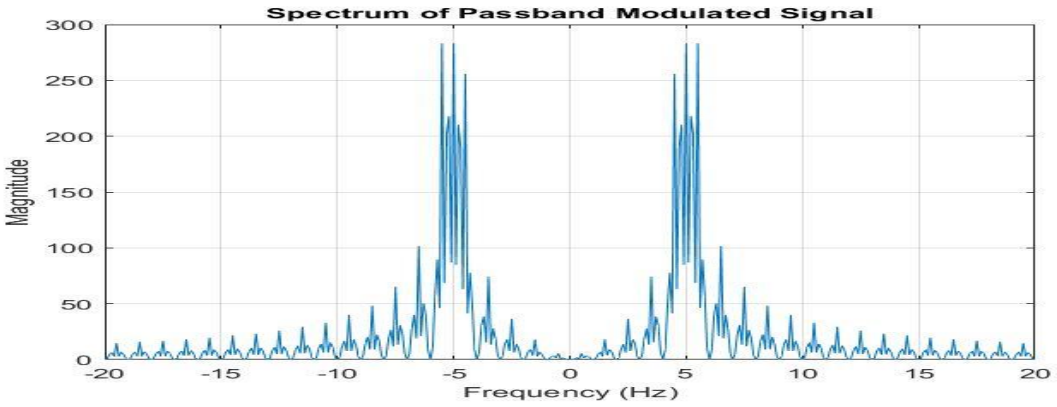


Figure 2: spectrum of passband modulation .

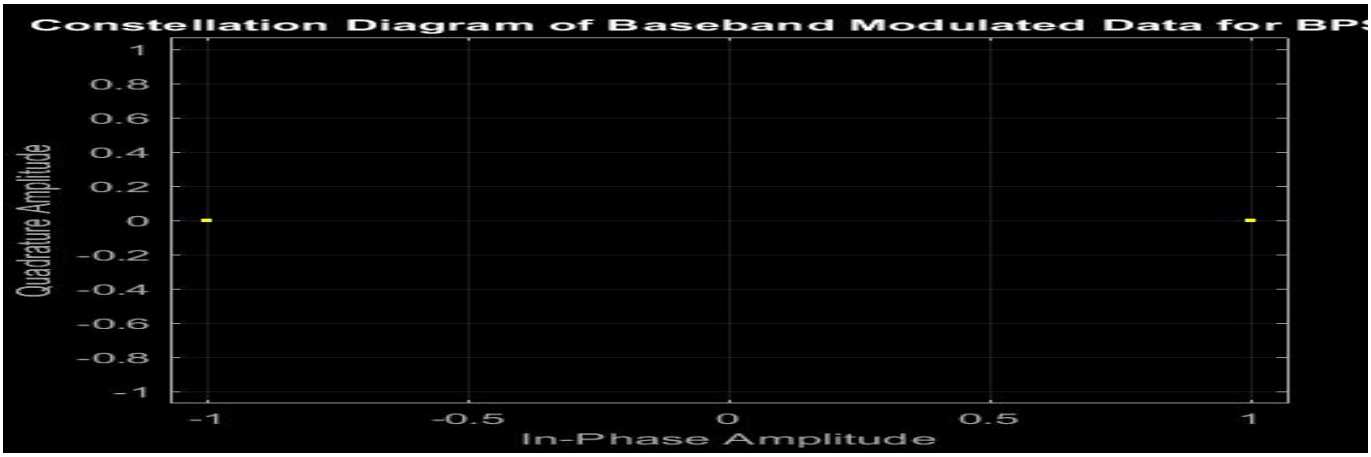
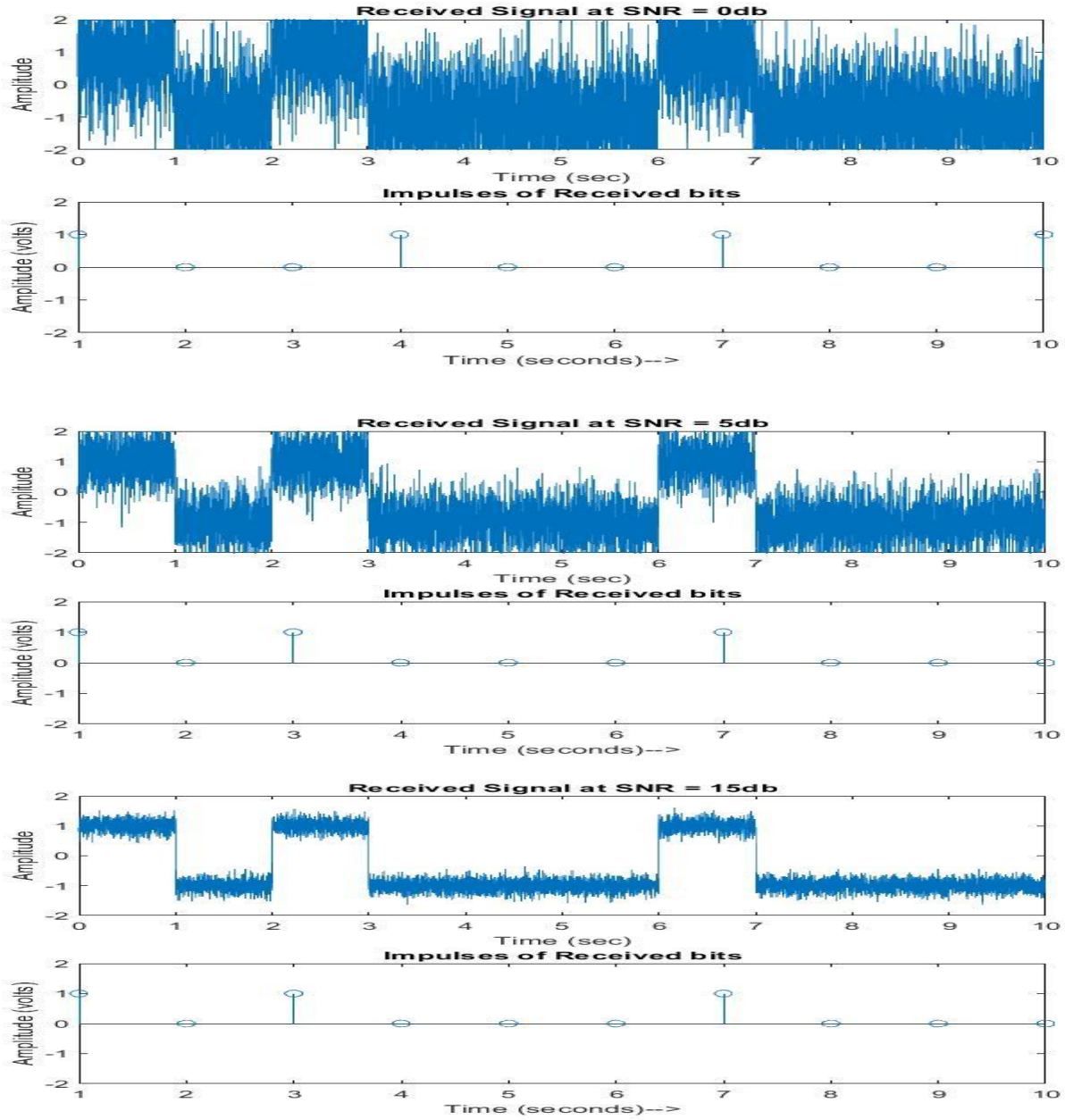


Figure 3: constellation diagram .



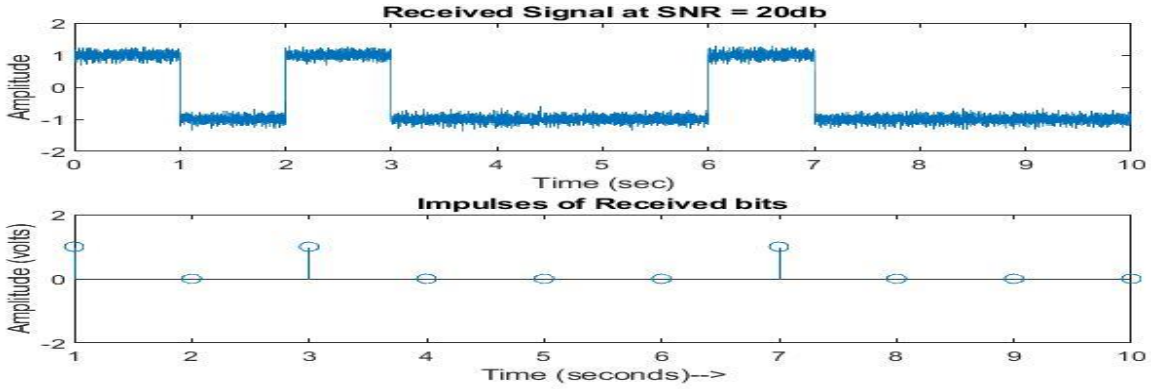


figure 4 : received baseband and impulses at different SNR

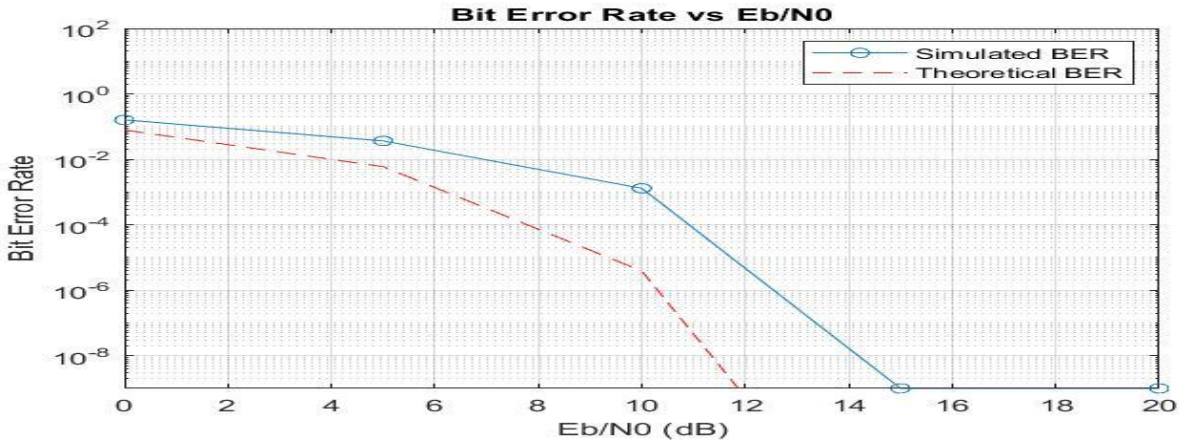


Figure 5: A semi-log plot comparing theoretical and simulated BER values.

BPSK Results:

- The baseband signal displayed correct mapping of binary data into two distinct amplitudes.
- The passband signal demonstrated a clean sinusoidal modulation for binary "0" and "1".
- The spectrum analysis showed a clear peak at the carrier frequency.
- BER simulation confirmed agreement between theoretical and simulated values under different SNR conditions.

BFSK Procedures:

1. **Transmitter:**
   - Generate a random binary data stream.
   - Map binary data to baseband modulated data using BFSK principles, assigning different frequencies for binary "0" and "1".
   - Generate the BFSK passband modulated signal.
2. **Channel:**
   - Add AWGN to the BFSK signal at various SNR levels.
3. **Receiver:**
   - Demodulate the received signal by determining the dominant frequency during each bit duration.
   - Recover the transmitted binary data.
4. **Comparison:**
   - Calculate the simulated BER for different SNR values.
   - Compare the simulated BER with the theoretical BER by plotting both on a semi-log graph.
5. **Code Implementation:**

```
% BFSK Modulation and Demodulation Simulation
clear all;
clc;

%% Parameters
T = 1; % Bit duration (seconds)
N = 5000 ; % Number of bits
data = randi([0, 1], 1, N); % Generate random binary stream
fs = 1000; % Number of samples per bit
u = fs * N; % Total samples
t = 0:1/fs:(N*T - 1/fs); % Time vector

A=1; % amp of sin
Eb=(A*A)/2; % energy per bit
sqrt_Eb=sqrt(Eb);

% Frequencies for BFSK modulation
fc0 = 5; % Frequency for bit 0
fc1 = 10; % Frequency for bit 1

%% Transmitter
% Baseband modulated data (BFSK)

baseband_modulated = [];
for i =1:N
    if data(i)==1
        baseband_modulated=[baseband_modulated ones(1,fs)];
    elseif data(i)==0
        baseband_modulated=[baseband_modulated zeros(1,fs)];
    end
end

% passband
```

```
passband_modulated = [];
for i = 1:N
    if data(i) == 0
        passband_modulated = [passband_modulated  A*sqrt(2/T)*sin(2
* pi * fc0 * t((i-1)*fs + 1:i*fs))]; % Frequency for 0
    else
        passband_modulated = [passband_modulated  A*sqrt(2/T)*sin(2
* pi * fc1 * t((i-1)*fs + 1:i*fs))]; % Frequency for 1
    end
end
%}

% Bit stream impulses plot
subplot(3,1,1);
stem(data);
xlabel('Time (sec)');
ylabel('Amplitude (volts)');
title('Impulses of Bits to be Transmitted');
ylim([-0.5 1.5]);
% output of unipolar NRZ
grid on;
% baseband ploting
subplot(3,1,2);
plot(t,baseband_modulated);
xlabel('Time (mel sec)');
ylabel('Amplitude (volts)');
title('generated unipolar NRZ signal');
ylim([-2 2]);
grid on;

% Output of BFSK plotting
subplot(3,1,3);
plot(t, passband_modulated);
xlabel('Time (s)');
ylabel('Amplitude (volts)');
title('Generated BFSK Signal');
ylim([-1.5 1.5]);
grid on;

% Constellation diagram (for BFSK, 0 and 1 represented as two
points)
% Constellation points
constellation = zeros(2, 2); % Preallocate for two points
constellation(1, :) = [sqrt_Eb, 0]; % Point for bit 0
constellation(2, :) = [0, sqrt_Eb]; % Point for bit 1
% Plot constellation diagram
figure;
scatter(constellation(1, :), constellation(2, :), 'filled');
title('Constellation Diagram for BFSK');
xlabel('In-Phase Amplitude');
```

```
ylabel('Quadrature Amplitude');
%xlabel('Frequency (Hz)');
%ylabel('Amplitude');
xlim([0, 2]); % Adjust limits as needed
ylim([0, 2]); % Adjust limits as needed
grid on;
text(sqrt_Eb, 0.2, '0', 'HorizontalAlignment', 'center',
'FontSize', 12);
text(0.2, sqrt_Eb , '1', 'HorizontalAlignment', 'center',
'FontSize', 12);

%% Spectrum Analysis
spectrum = fftshift(fft(passband_modulated)); % Spectrum of the
BFSK signal
f = -fs/2:fs/length(spectrum):fs/2-fs/length(spectrum); % Frequency
vector

figure;
plot(f, abs(spectrum)/N);
title('Spectrum of Passband Modulated Signal (BFSK)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([-20 20]);
grid on;

% Bandwidth estimation
[bw, ~, ~, ~] = obw(baseband_modulated, fs);
disp(['Estimated Bandwidth of the Passband Modulated Signal: ',
num2str(bw), ' Hz']);

%% Channel: Add AWGN  and receiver
EbN0_range = 0:5:20; % Eb/N0 range
BER = []; % Placeholder for Bit Error Rate
k = 1;  % bits per symbol
SNR=[];
for i = 1:length(EbN0_range)
    % Convert Eb/N0 to SNR
    SNR(i) = EbN0_range(i) + 10*log10(k);
    % Add noise
    received_signal = awgn(baseband_modulated, SNR(i), 'measured');
    figure;
    subplot(2,1,1);
    plot(t,received_signal);
    title(['Received Signal at SNR = ' num2str(SNR(i)) 'db']);
    xlabel('Time (sec)');
    ylabel('Amplitude');
    ylim([-2 2]);
    grid on ;
    % detector
    sample_index=1:fs:length(received_signal);
```

```
    detected_bit=received_signal(sample_index) > 0.5 ;

    subplot(2,1,2);
    stem(detected_bit);
    title('Impulses of Received bits');
    xlabel('Time (seconds)-->');
    ylabel('Amplitude (volts)');
    ylim([-2 2]);
    grid on ;
    % Calculate Bit Error Rate
    errors = sum( data ~= detected_bit);
    BER = [BER (errors/N)+1e-10] ;
end

% Theoretical BER calculation for BFSK
theory_BER = 0.5 * erfc(sqrt(10.^(EbN0_range/10)));

% Plot BER vs Eb/N0
figure;
semilogy(EbN0_range, BER, 'o-', 'DisplayName', 'Simulated BER');
hold on;
semilogy(EbN0_range, theory_BER, 'r--', 'DisplayName', 'Theoretical
BER');
title('Bit Error Rate vs Eb/N0 (BFSK)');
xlabel('Eb/N0 (dB)');
ylabel('Bit Error Rate');
ylim([1e-10 1]);
grid on;
legend;
```

Figures:

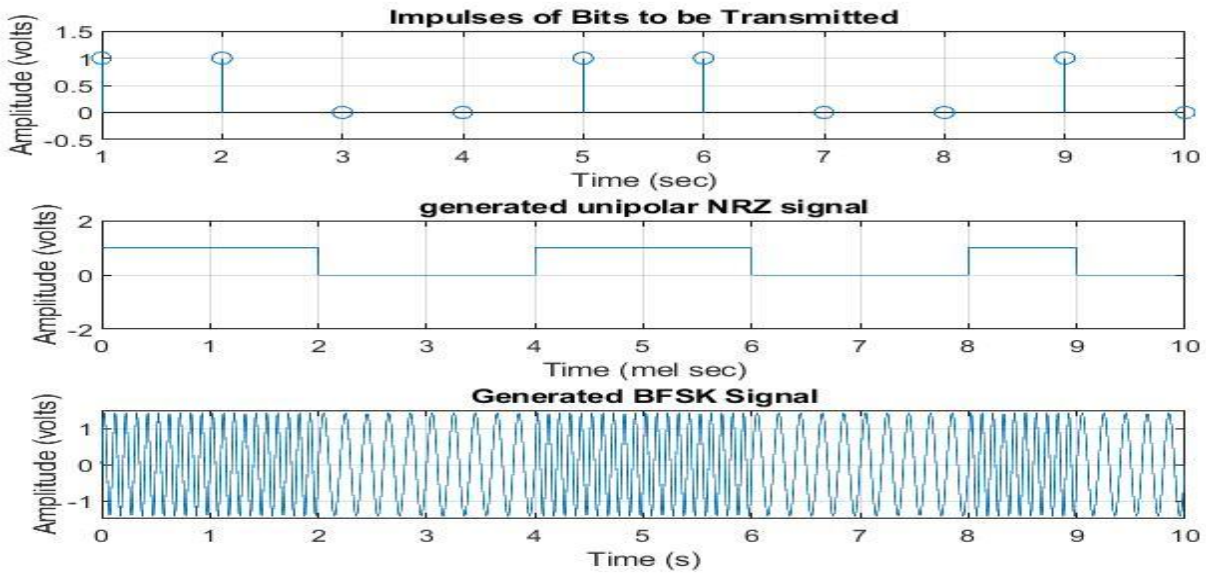


Figure 6: Displays impulses data & baseband NRZ Signal and passband modulation .

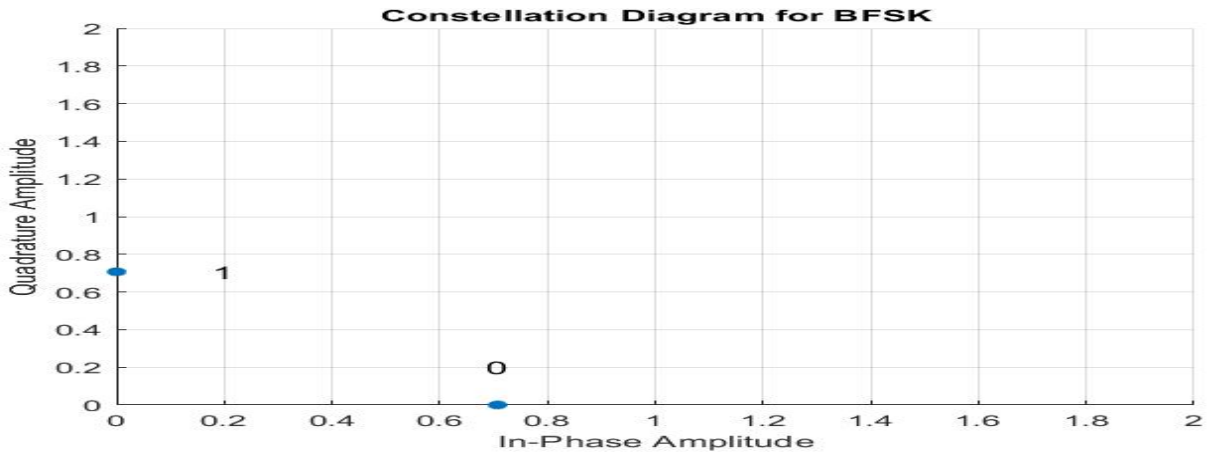


Figure 7: constellation diagram

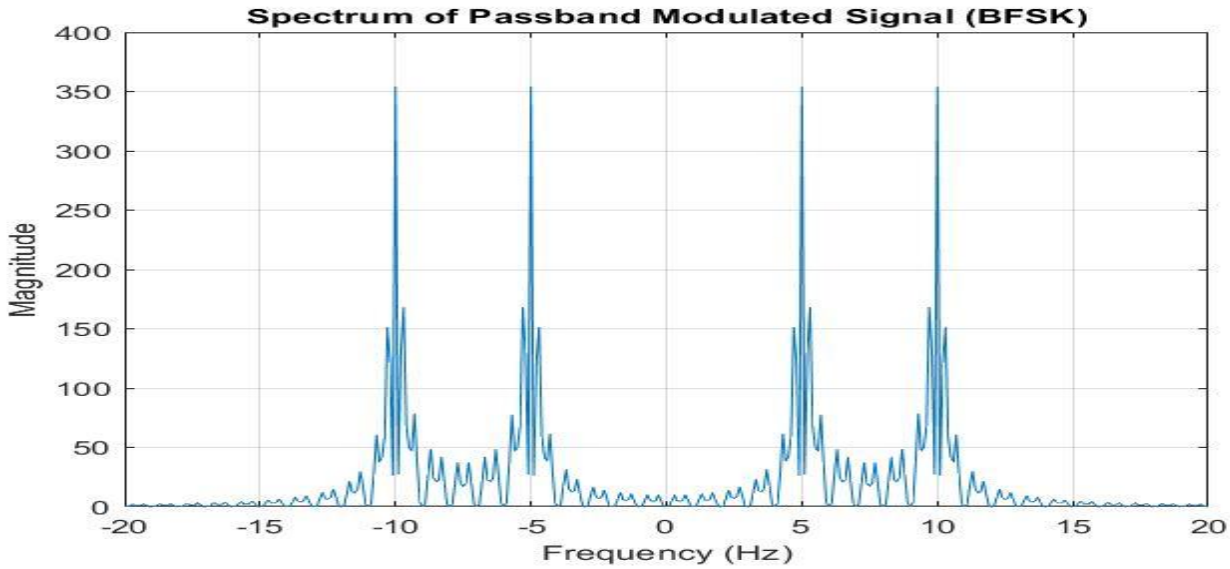
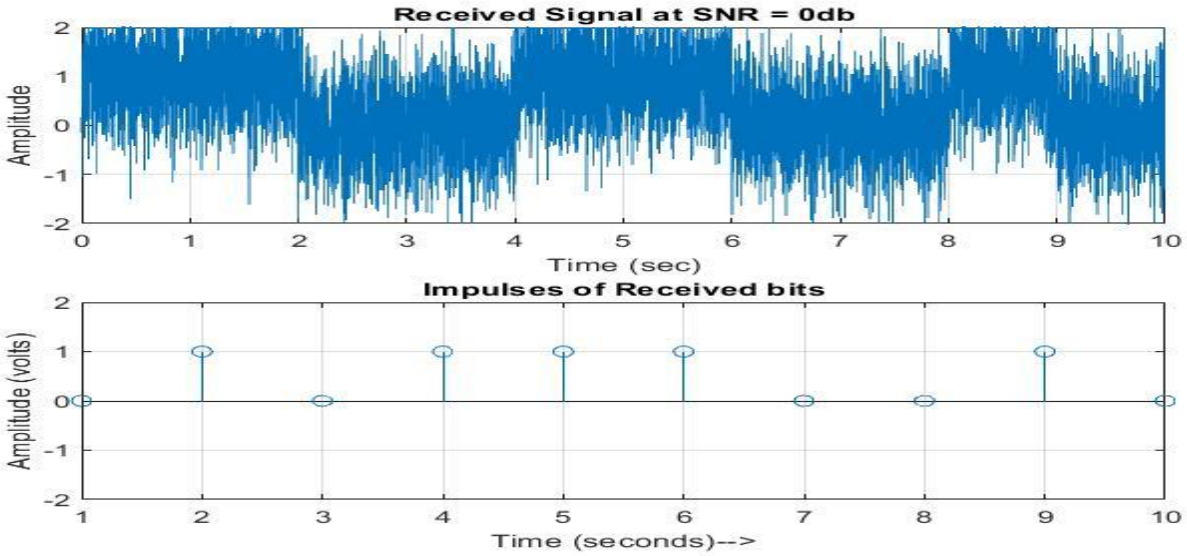
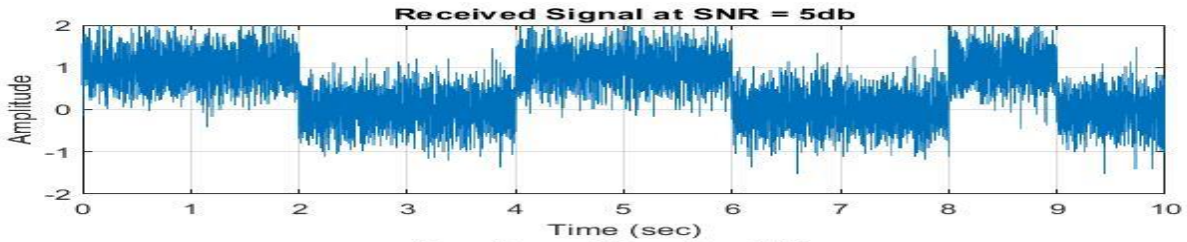
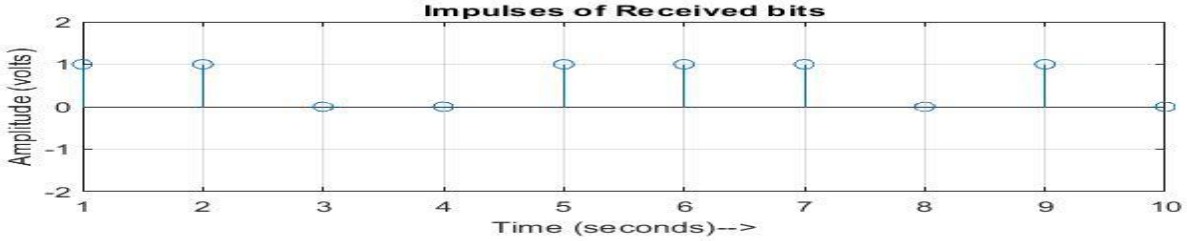
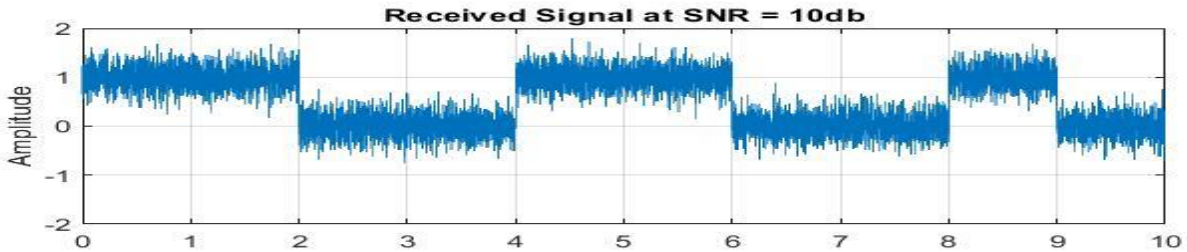


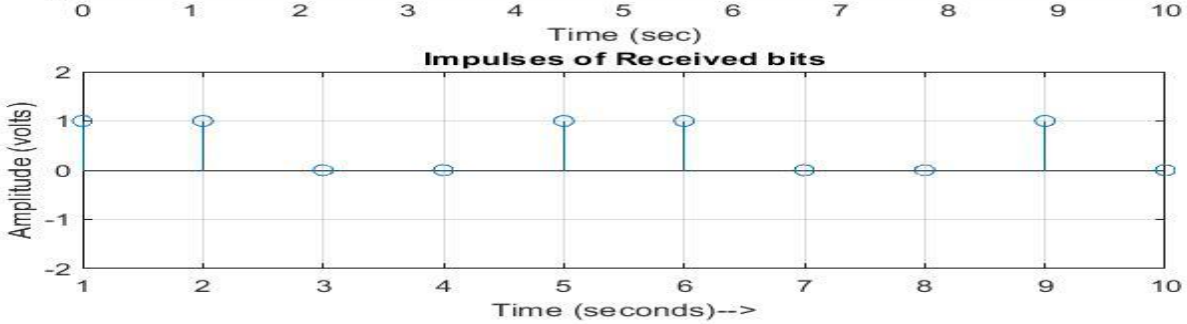
figure 8: spectrum of passband modulation

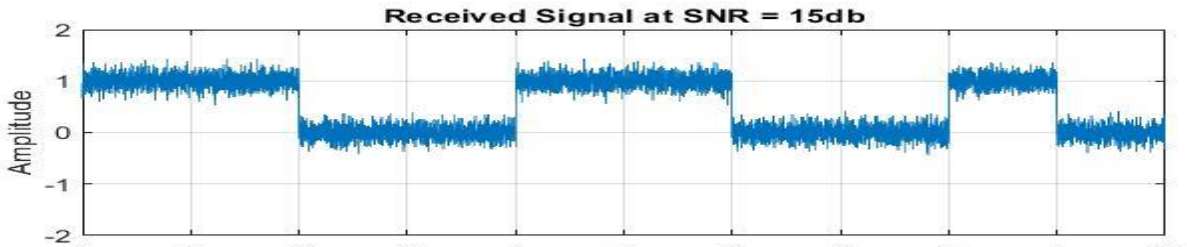


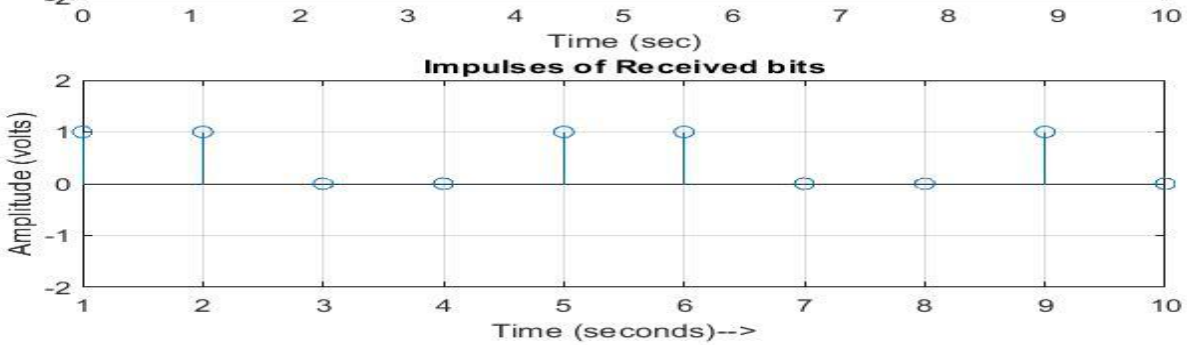

Received Signal at SNR = 5db
Amplitude
Time (sec)


Impulses of Received bits
Amplitude (volts)
Time (seconds)-->


Received Signal at SNR = 10db
Amplitude
Time (sec)


Impulses of Received bits
Amplitude (volts)
Time (seconds)-->


Received Signal at SNR = 15db
Amplitude
Time (sec)


Impulses of Received bits
Amplitude (volts)
Time (seconds)-->

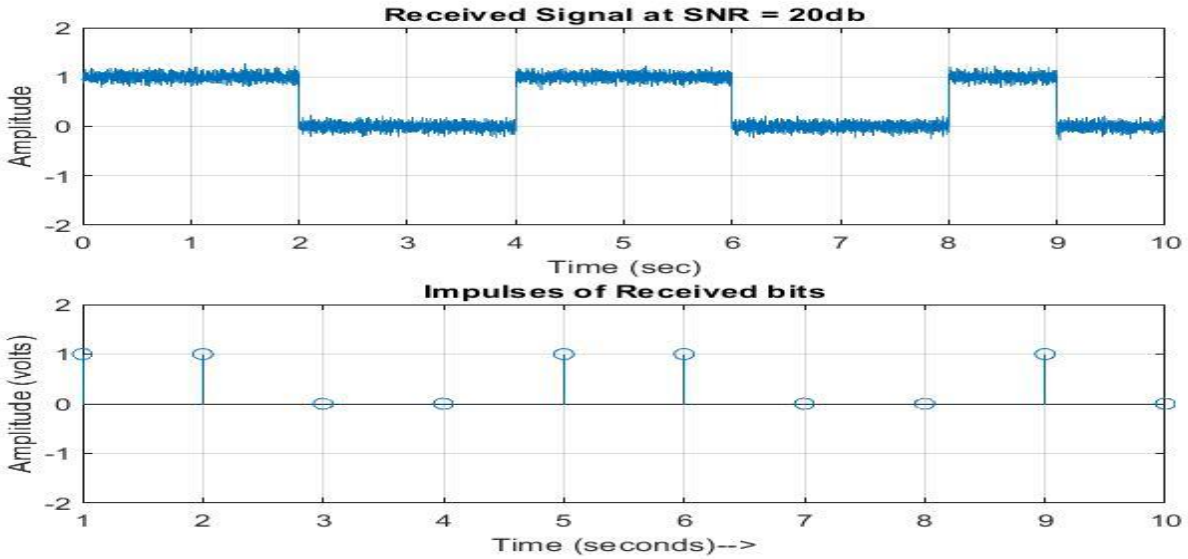


Figure 9: received baseband and impulses

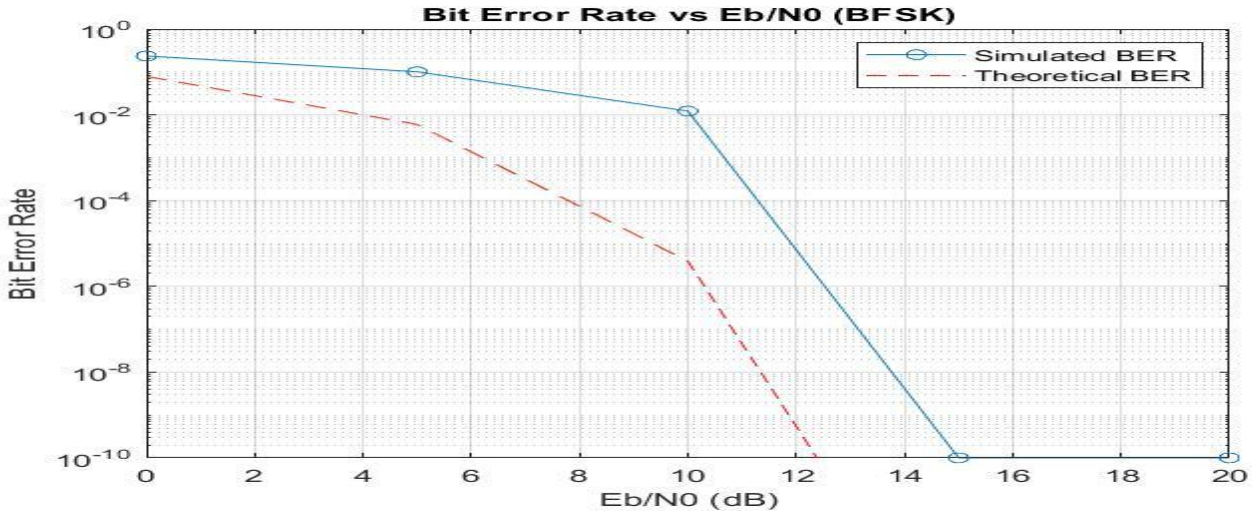


Figure 10: A semi-log plot comparing theoretical and simulated BER values

BFSK Results:

- The baseband signal accurately reflected binary data with distinct frequencies for "0" and "1".
- The passband signal demonstrated successful frequency modulation.
- The spectrum analysis verified the presence of two distinct frequency components corresponding to the binary data.
- BER results showed alignment between theoretical and simulated performance.

QASK Procedures:

1. **Transmitter:**
   - Generate a random binary data stream.
   - Map pairs of binary data to in-phase and quadrature amplitude components.
   - Generate the QASK passband modulated signal.
2. **Channel:**
   - Add AWGN to the BFSK signal at various SNR levels.
3. **Receiver:**
   - Demodulate the received signal by extracting in-phase and quadrature components.
   - Map the amplitude combinations back to binary pairs to recover the transmitted data.
4. **Comparison:**
   - Calculate the simulated BER for different SNR values.
   - Compare the simulated BER with the theoretical BER by plotting both on a semi-log graph.
5. **Code Implementation:**

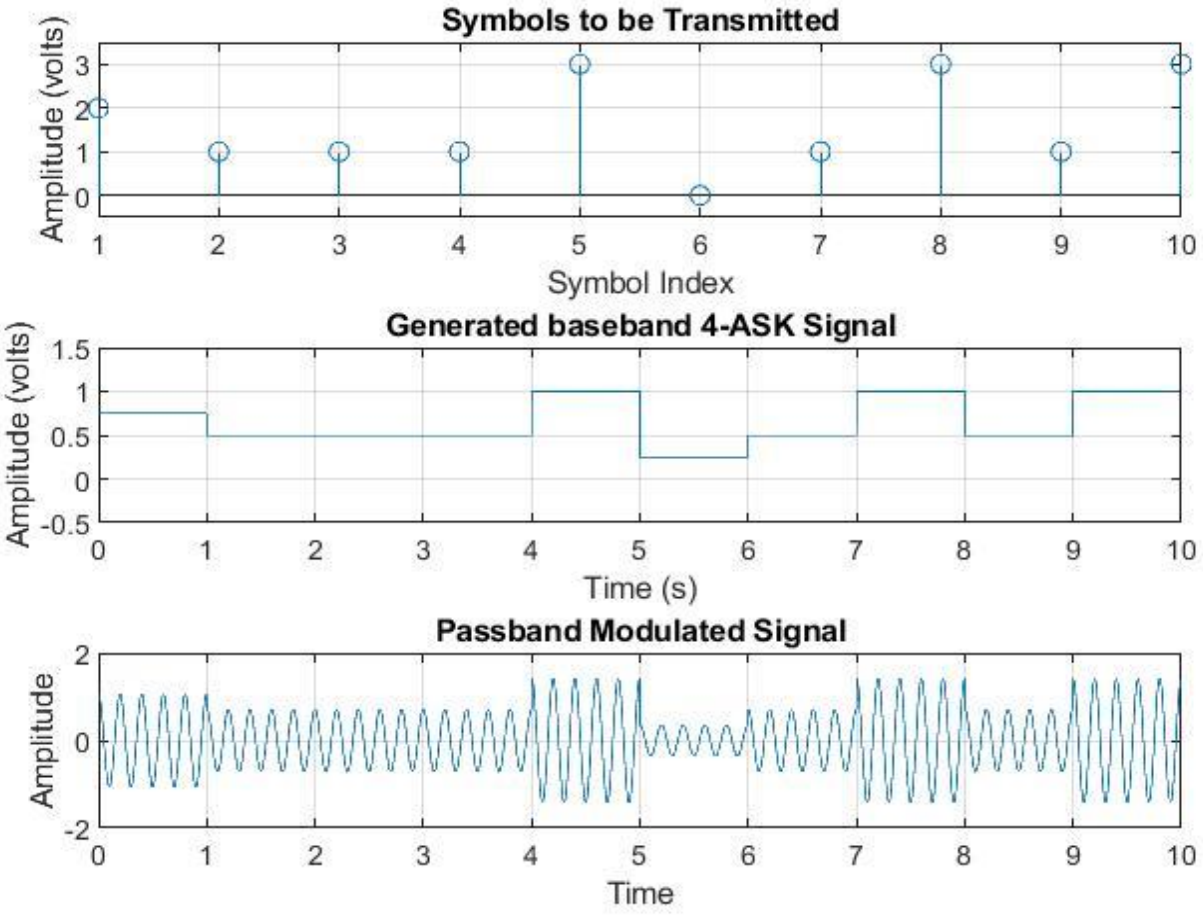
```
% 4-ARY ASK Modulation and Demodulation Simulation
clear all;
clc;

%% Parameters
T = 1; % symbol duration (seconds)
N = 10 ; % Number of symbols
data = randi([0, 3], 1, N); % Generate random 2-bit symbols (0
to 3)
fs = 1000; % Number of samples per bit
u = fs * N; % Total samples
t = 0:1/fs:(N*T - 1/fs); % Time vector
fc=5;

A = 1; % Maximum amplitude
Eb = (A^2)/2; % Energy per bit
sqrt_Eb = sqrt(Eb);

% Amplitude levels for 4-ASK
amplitude_levels = [A/4, A/2, 3*A/4, A]; % Define the four
amplitude levels for symbols 0, 1, 2, 3

Amp=amplitude_levels; % amp of sin
Eb=(Amp.*Amp)/2; % energy per bit
sqrt_Eb=sqrt(Eb);

%% Transmitter
% Baseband modulated data (4-ASK)
baseband_modulated = [];
for i = 1:N
    baseband_modulated = [baseband_modulated
amplitude_levels(data(i) + 1) * ones(1, fs)];
```

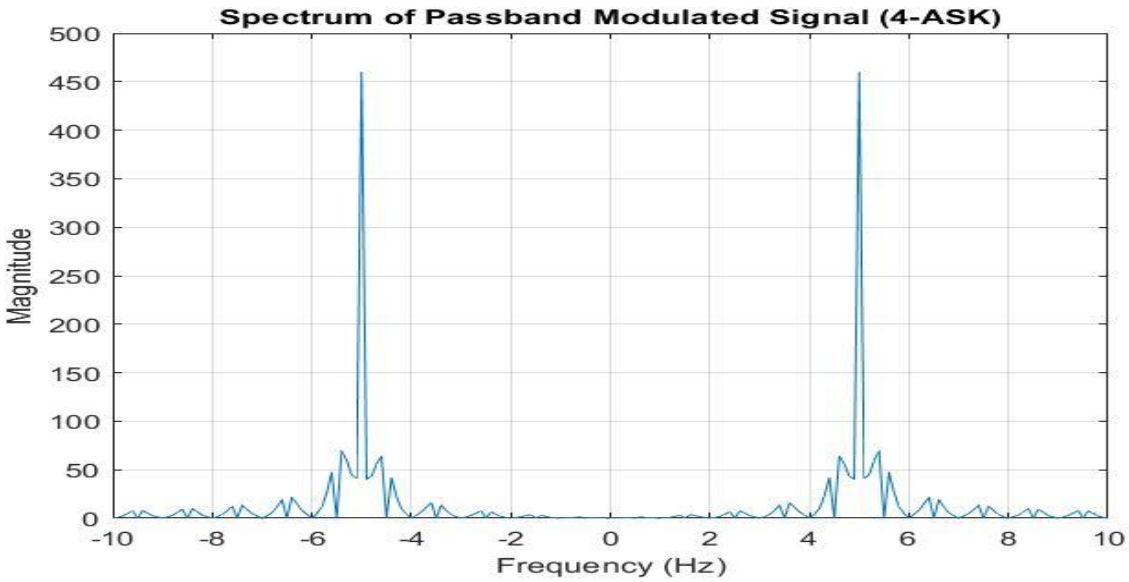
```
end


% Passband modulation
carrier = (sqrt(2/T))*cos(2*pi*fc*t); % Carrier signal
passband_modulated = baseband_modulated .* carrier; % Passband
modulation


% symbols plotting

subplot(3,1,1)
stem(data);
xlabel('Symbol Index');
ylabel('Amplitude (volts)');
title('Symbols to be Transmitted');
ylim([-0.5 3.5]);
grid on;

% Output of base_band 4-ASK plotting
subplot(3,1,2)
plot(t, baseband_modulated);
xlabel('Time (s)');
ylabel('Amplitude (volts)');
title('Generated baseband 4-ASK Signal');
ylim([-0.5 1.5]);
grid on;

%output modulation
subplot(3,1,3)
plot(t,passband_modulated);
title('Passband Modulated Signal');
xlabel('Time');
ylabel('Amplitude');
ylim([-2 2]);
grid on;

% Create the constellation points
constellation = zeros(2, 4); % Preallocate for four points
constellation(1, :) = sqrt_Eb; % Amplitude levels on the x-
axis
constellation(2, :) = zeros(1, 4); % All points on the y-axis
(0 for ASK)

% Plot the constellation diagram
figure;
scatter(constellation(1, :), constellation(2, :), 'filled');
title('Constellation Diagram for 4-ARY ASK');
xlabel('Energy');
% ylabel('Quadrature Amplitude');
xlim([-0.5, A + 0.5]); % Adjust limits as needed
```

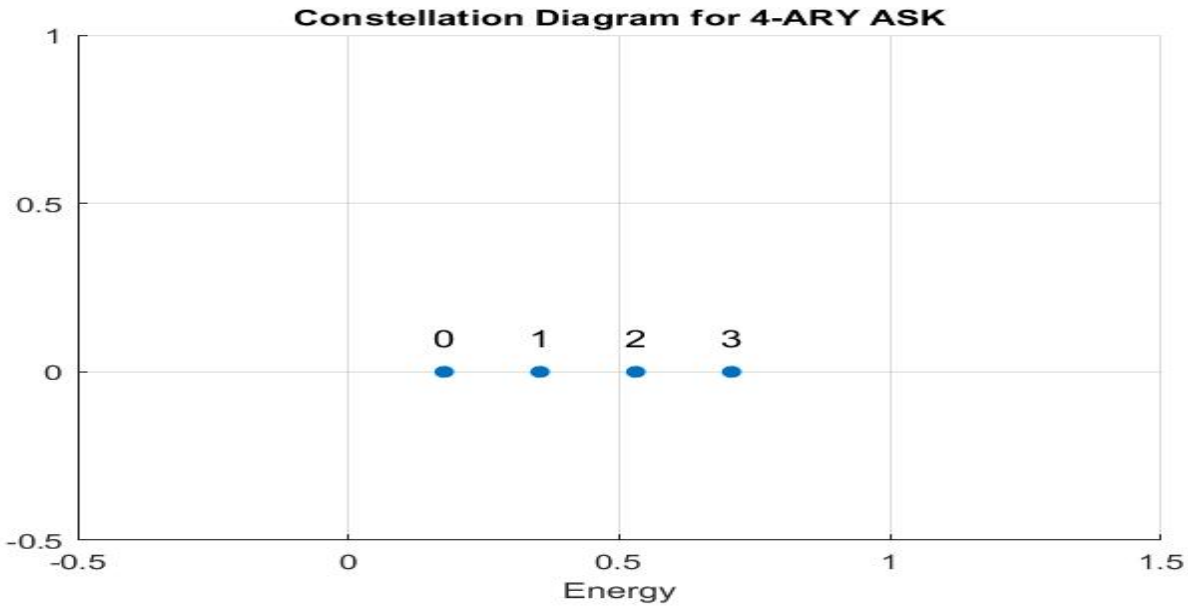
```
ylim([-0.5, 1]); % Adjust limits as needed
grid on;

% Label the constellation points
text(constellation(1, 1), 0.1, '0', 'HorizontalAlignment',
'center', 'FontSize', 12);
text(constellation(1, 2), 0.1, '1', 'HorizontalAlignment',
'center', 'FontSize', 12);
text(constellation(1, 3), 0.1, '2', 'HorizontalAlignment',
'center', 'FontSize', 12);
text(constellation(1, 4), 0.1, '3', 'HorizontalAlignment',
'center', 'FontSize', 12);



%% Spectrum Analysis
spectrum = fftshift(fft(passband_modulated)); % Spectrum of
the 4-ASK signal
f = -fs/2:fs/length(spectrum):fs/2-fs/length(spectrum); %
Frequency vector

figure(2);
plot(f, abs(spectrum)/N);
title('Spectrum of Passband Modulated Signal (4-ASK)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([-10 10]);
grid on;

% Bandwidth estimation
[bw, ~, ~, ~] = obw(baseband_modulated, fs);
disp(['Estimated Bandwidth of the 4-ASK Signal: ',
num2str(bw), ' Hz']);

%% Channel: Add AWGN and receiver
EbN0_range = 0:5:20; % Eb/N0 range
BER = []; % Placeholder for Bit Error Rate
k = 2; % bits per symbol (4-ASK)
SNR=[];
for i = 1:length(EbN0_range)
    % Convert Eb/N0 to SNR
    SNR(i) = EbN0_range(i) + 10*log10(k);
    % Add noise
    received_signal = awgn(baseband_modulated, SNR(i),
'measured');
    figure;
    subplot(2,1,1);
    plot(t, received_signal);
    title(['Received Signal at SNR = ' num2str(SNR(i)) 'db']);
    xlabel('Time (s)');
```

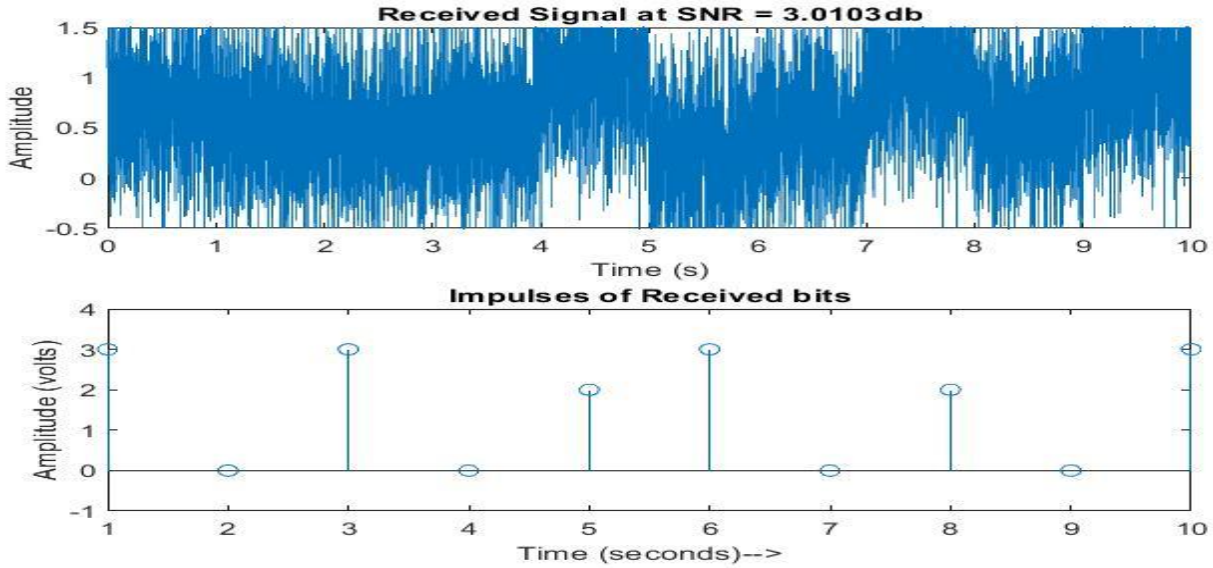
```
    ylabel('Amplitude');
    ylim([-0.5 1.5]);
    % grid on ;



    % Detector
    detected_symbols = zeros(1, N);
    for j = 1:N
        % Sample the received signal at the symbol intervals
        sample = received_signal((j-1)*fs + 1);
        % Determine the detected symbol based on the amplitude
        [~, detected_symbols(j)] = min(abs(sample -
amplitude_levels));
        detected_symbols(j) = detected_symbols(j) - 1; %
Adjust index (0-3)
    end
    subplot(2,1,2);
    stem(detected_symbols);
    title('Impulses of Received bits');
    xlabel('Time (seconds)-->');
    ylabel('Amplitude (volts)');
    ylim([-1 4]);
   % grid on ;

    % Calculate Bit Error Rate
    errors = sum(data ~= detected_symbols);
    BER = [BER (errors/N)+1e-30];
end

% Theoretical BER calculation for 4-ASK
%theory_BER = 0.5 * erfc(sqrt(10.^(EbN0_range/10)));
theory_BER = 3 * qfunc(sqrt(3 * 10.^(EbN0_range / 20)));

% Plot BER vs Eb/N0
figure;
semilogy(EbN0_range, BER, 'o-', 'DisplayName', 'Simulated
BER');
hold on;
semilogy(EbN0_range, theory_BER, 'r--', 'DisplayName',
'Theoretical BER');
title('Bit Error Rate vs Eb/N0 (4-ASK)');
xlabel('Eb/N0 (dB)');
ylabel('Bit Error Rate');
ylim([1e-10 1]);
grid on;
legend;
```

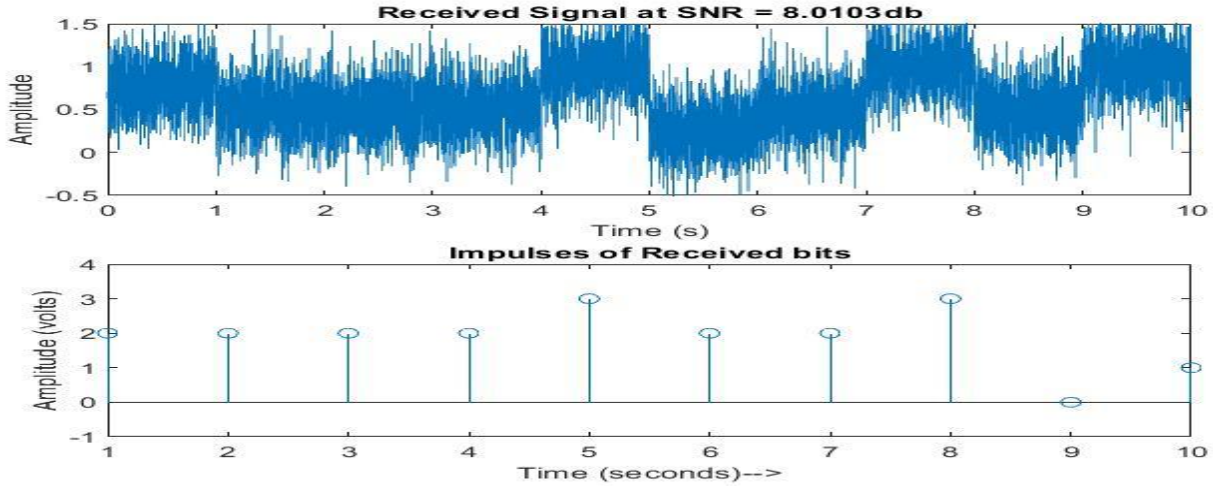
Figures:

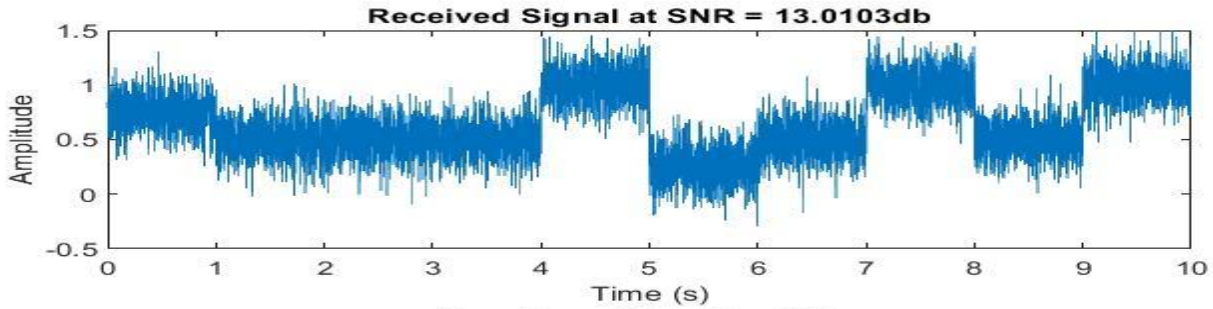

Symbols to be Transmitted
Amplitude (volts)
Symbol Index
Generated baseband 4-ASK Signal
Amplitude (volts)
Time (s)
Passband Modulated Signal
Amplitude
Time

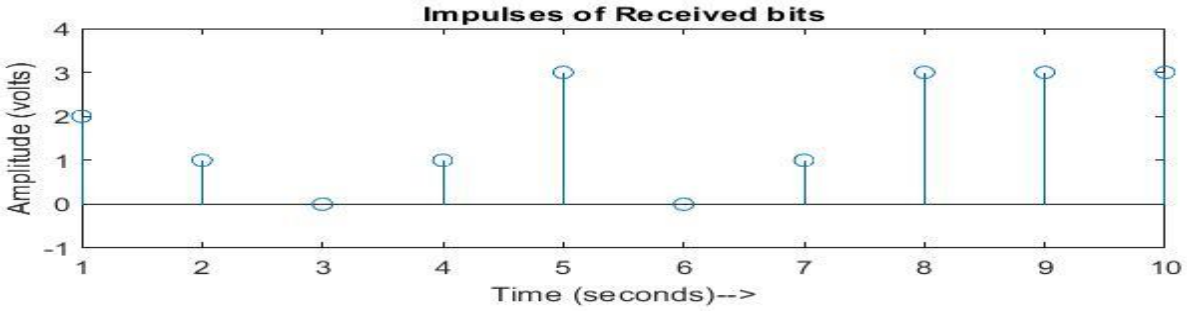

Spectrum of Passband Modulated Signal (4-ASK)
Magnitude
Frequency (Hz)

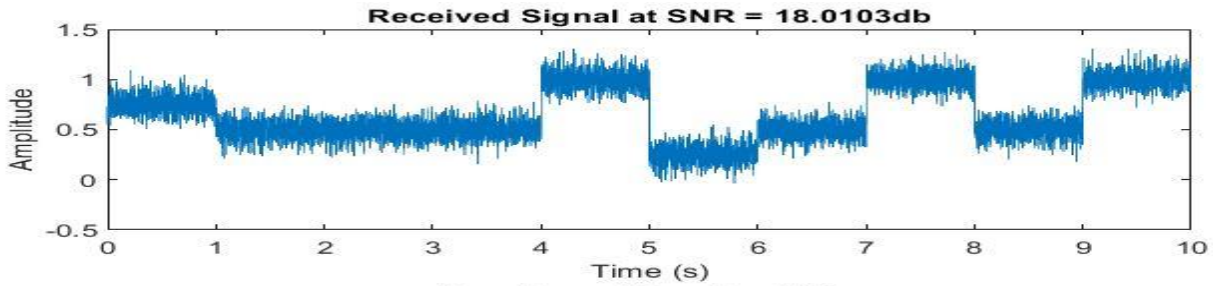

Constellation Diagram for 4-ARY ASK
1
0.5
0
-0.5
0
1
2
3
-0.5
0
0.5
1
1.5
Energy

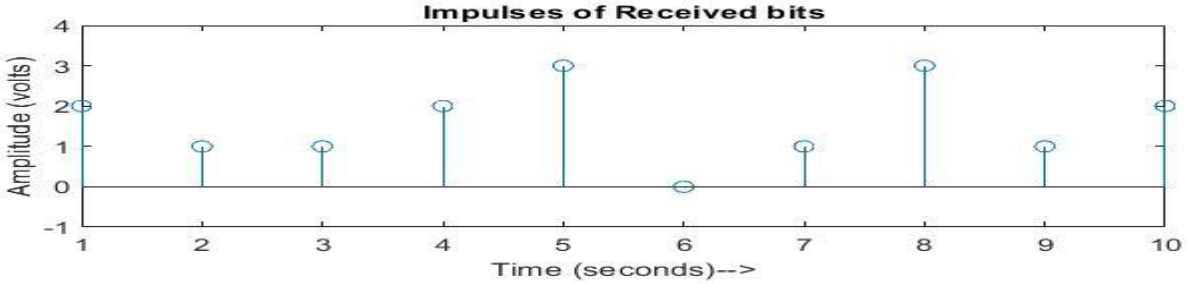

Received Signal at SNR = 3.0103db
Amplitude
Time (s)
Impulses of Received bits
Amplitude (volts)
Time (seconds)-->

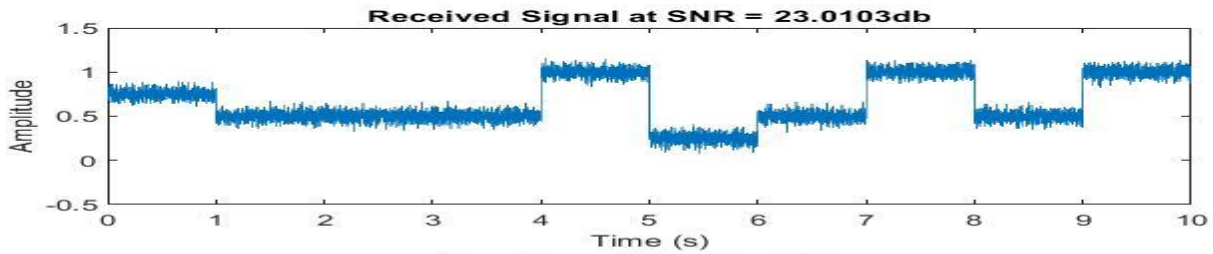

Received Signal at SNR = 8.0103db
Amplitude
Time (s)
Impulses of Received bits
Amplitude (volts)
Time (seconds)-->

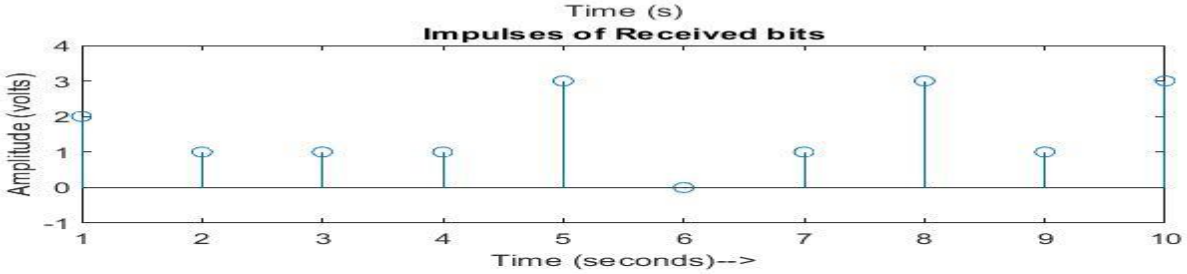

Received Signal at SNR = 13.0103db
Amplitude
Time (s)


Impulses of Received bits
Amplitude (volts)
Time (seconds)-->


Received Signal at SNR = 18.0103db
Amplitude
Time (s)


Impulses of Received bits
Amplitude (volts)
Time (seconds)-->


Received Signal at SNR = 23.0103db
Amplitude
Time (s)


Impulses of Received bits
Amplitude (volts)
Time (seconds)-->

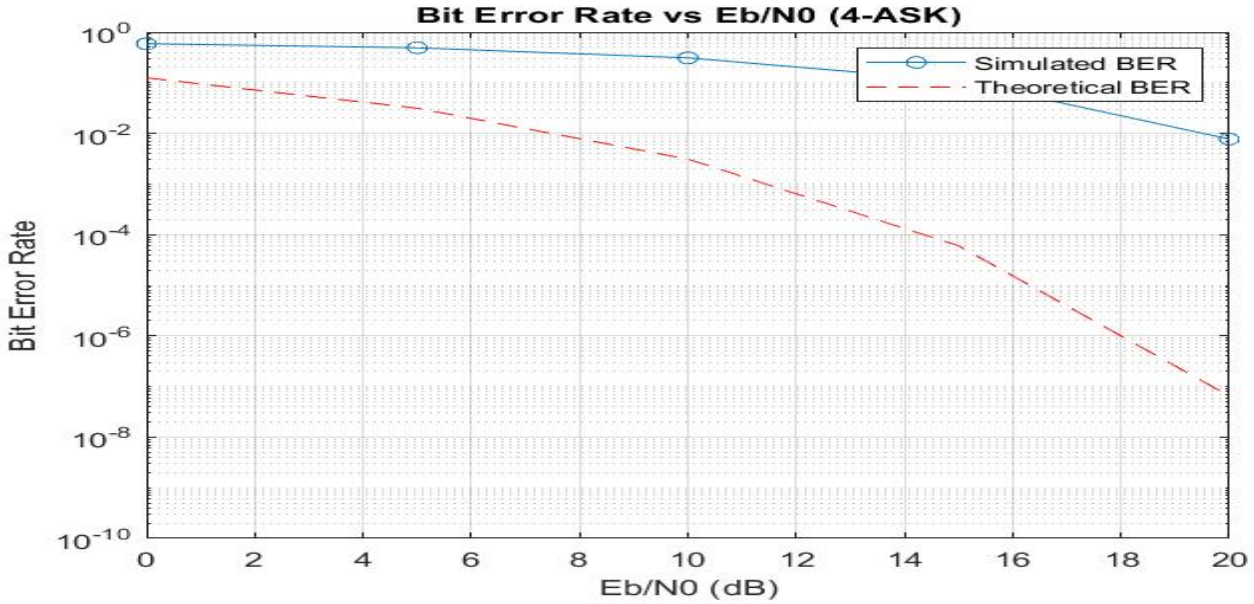


Figure 19: A semi-log plot comparing theoretical and simulated BER values.

QASK Results:

- The baseband signal correctly displayed mapping of binary pairs into in-phase and quadrature amplitudes.
- The passband signal showed effective modulation of both in-phase and quadrature components.
- The spectrum analysis confirmed the expected bandwidth for QASK.
- Constellation diagrams verified correct mapping of symbols under noisy conditions.
- BER simulations indicated a gradual improvement in performance with increasing SNR.

Conclusion

The simulation of BPSK, BFSK, and QASK modulation schemes provided valuable insights into their behavior and performance under various noise conditions.

BPSK: Demonstrated robustness and simplicity, offering strong performance with low BER in noisy environments.

BFSK: Showed reliability with distinct frequency modulation, making it suitable for applications requiring frequency-based separation.

QASK: Highlighted its efficiency in encoding more data per symbol, though it requires a higher SNR to achieve similar BER performance as BPSK and BFSK.

Overall, each modulation scheme has unique strengths, and the choice depends on specific application requirements such as bandwidth efficiency, robustness, and complexity. The simulations confirmed the theoretical expectations, validating the implementation and analysis.