# Lab 10

# CT304 Digital Communication

**Raj Shah: 202201403**

**Bhoomish Patel: 202201414**

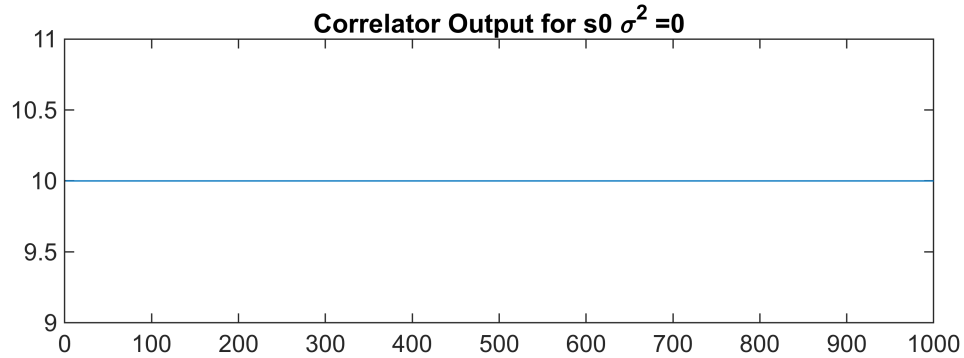
**Manthan Parmar: 202201416**

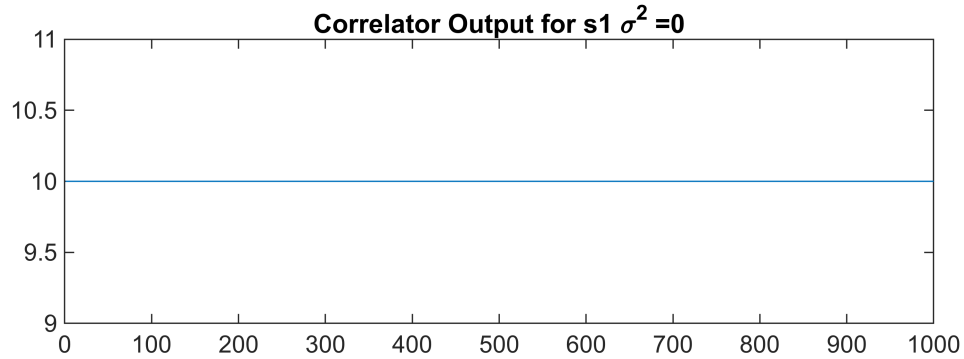
**Rakshit Pandhi: 202201426**

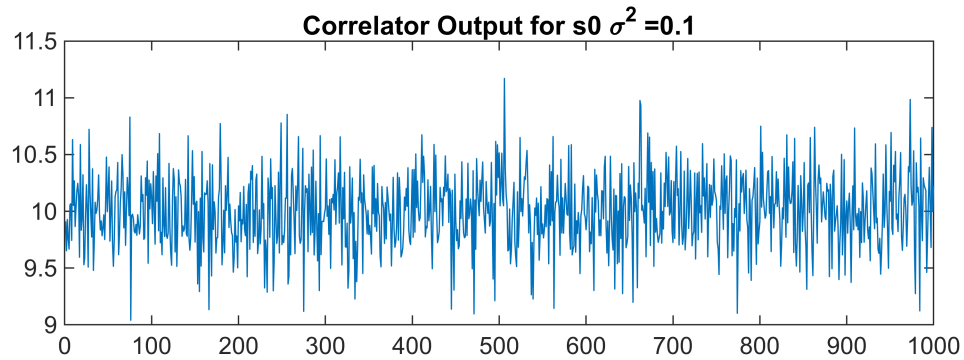
**Q1)**

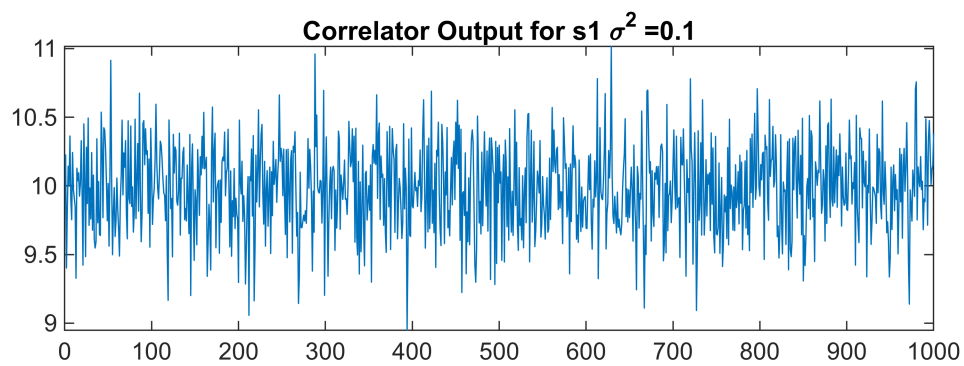
**5.1**

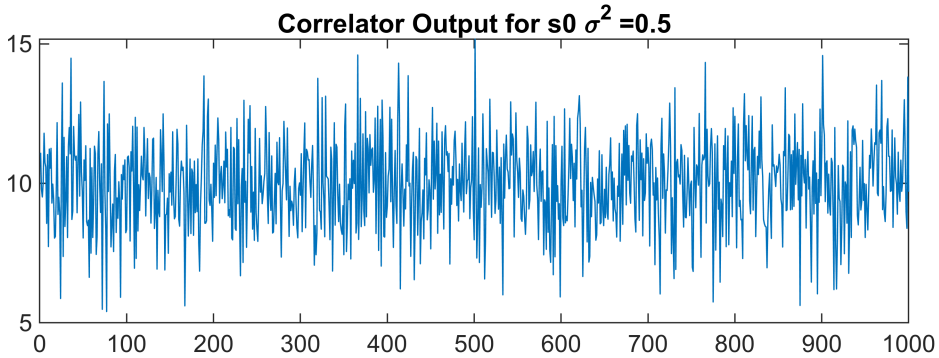
```
A = 1;
sigma_val = [0,0.1,0.5,1.0];
n_samples = 10;
n_trials = 1000;

s0 = A * ones(1,n_samples);
s1 =[A * ones(1,5) -A * ones(1,5)];

for sigma = sigma_val
    corr_out = zeros(2,n_trials);
    for trial = 1:n_trials
        noise = sigma * randn(1,n_samples);

        r0 = s0 + noise;
        r1 = s1 + noise;

        corr_out(1,trial) = sum(r0.*s0);
        corr_out(2,trial) = sum(r1.*s1);
    end
    figure;
    subplot(2,1,1);
    plot(corr_out(1,:));
    title(['Correlator Output for s0 \sigma^2 =',num2str(sigma)]);

    subplot(2,1,2);
    plot(corr_out(2,:));
    title(['Correlator Output for s1 \sigma^2 =',num2str(sigma)]);
end
```

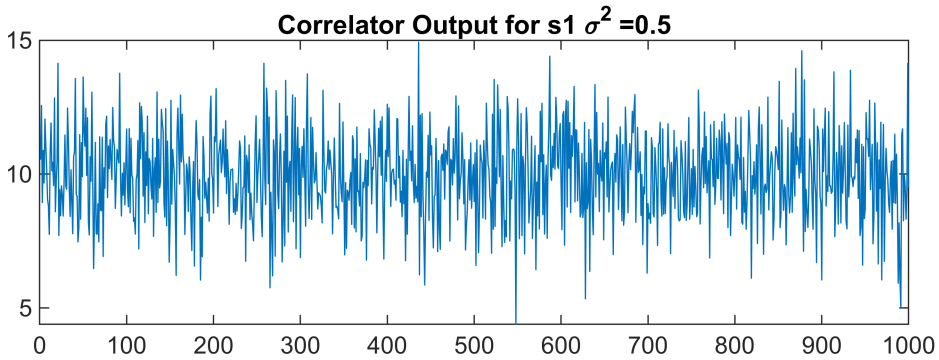

Correlator Output for s0 $\sigma^2$ =0
11
10.5
10
9.5
9
0
100
200
300
400
500
600
700
800
900
1000

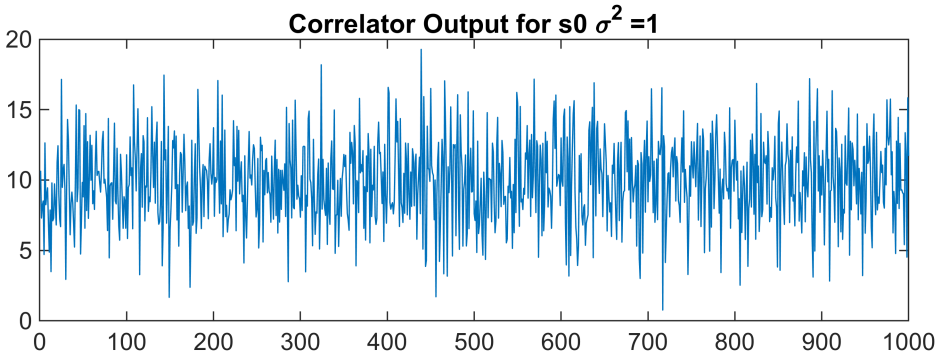

Correlator Output for s1 $\sigma^2$ =0
11
10.5
10
9.5
9
0
100
200
300
400
500
600
700
800
900
1000

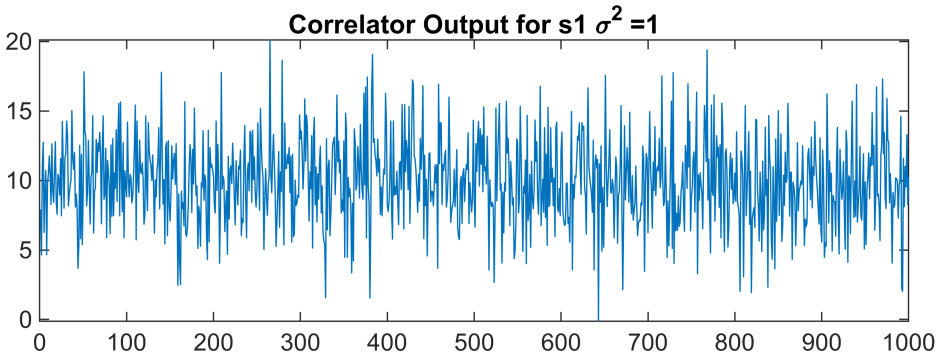

Correlator Output for s0 $\sigma^2$ =0.1
11.5
11
10.5
10
9.5
9
0
100
200
300
400
500
600
700
800
900
1000


Correlator Output for s1 $\sigma^2$ =0.1
11
10.5
10
9.5
9
0
100
200
300
400
500
600
700
800
900
1000



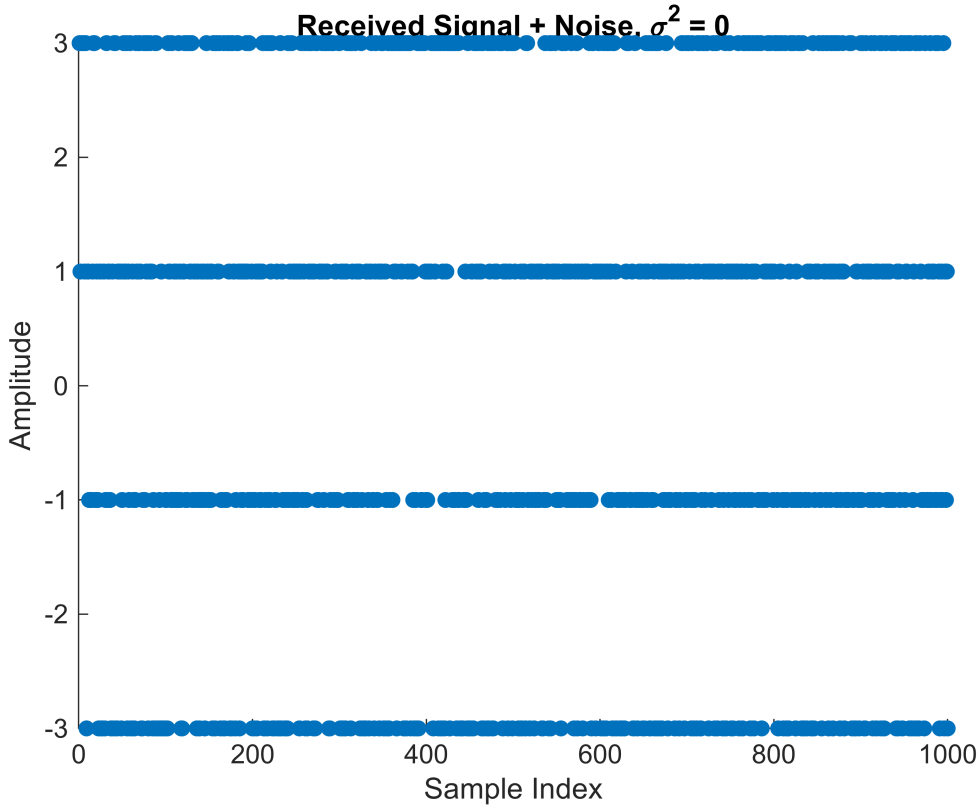



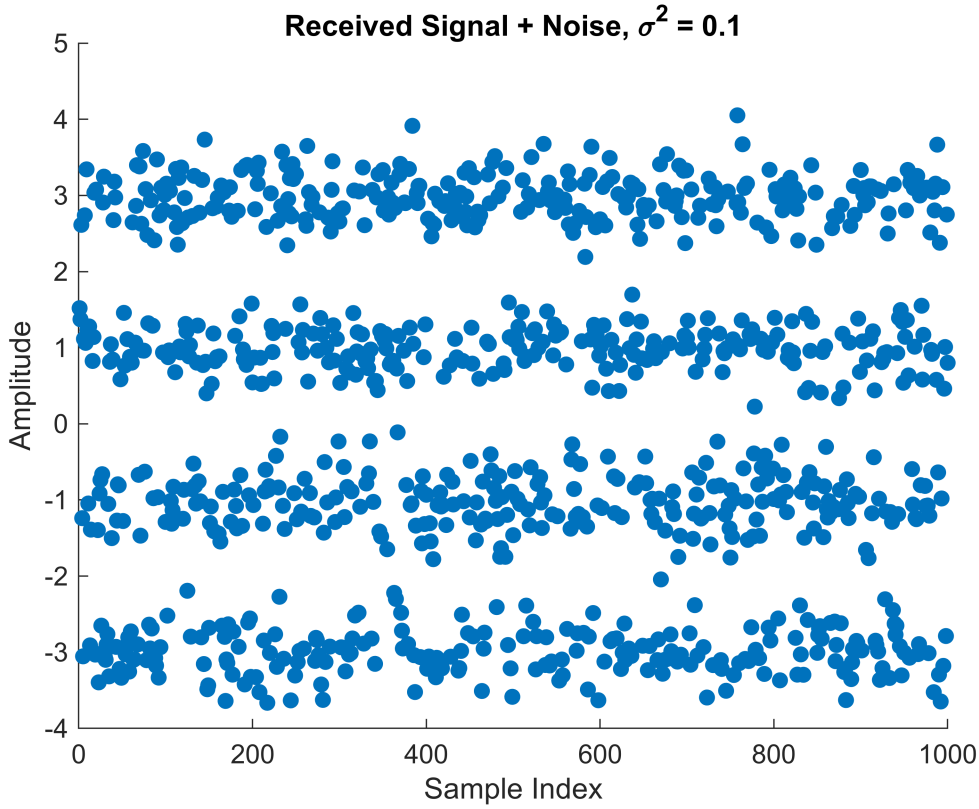



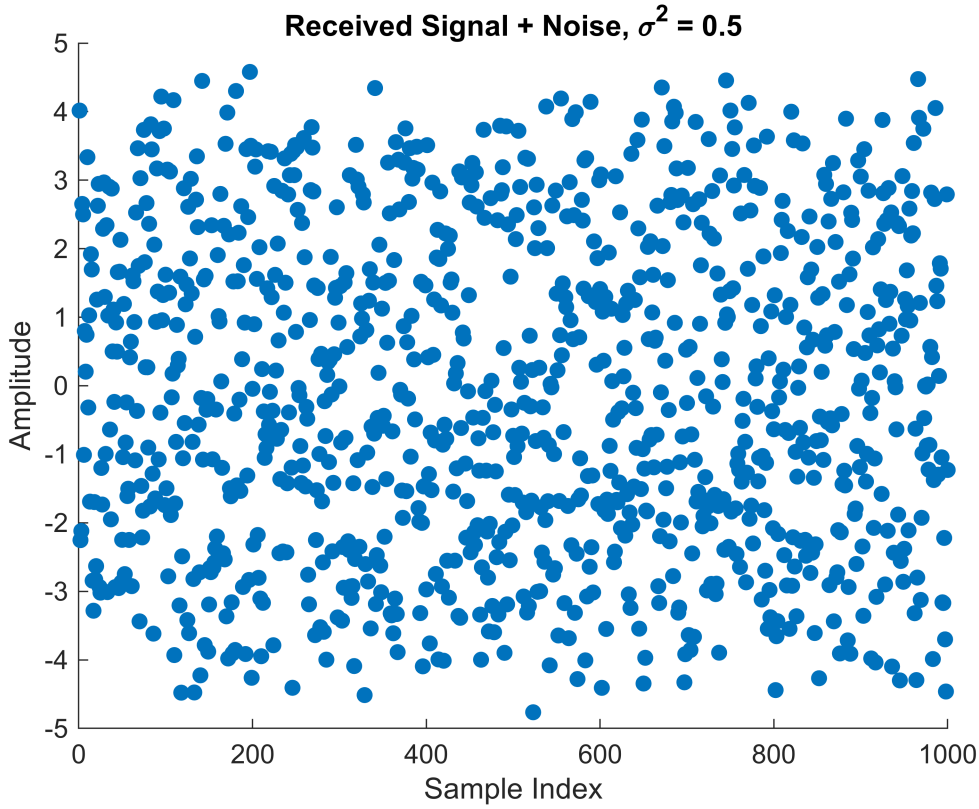

**5.11**

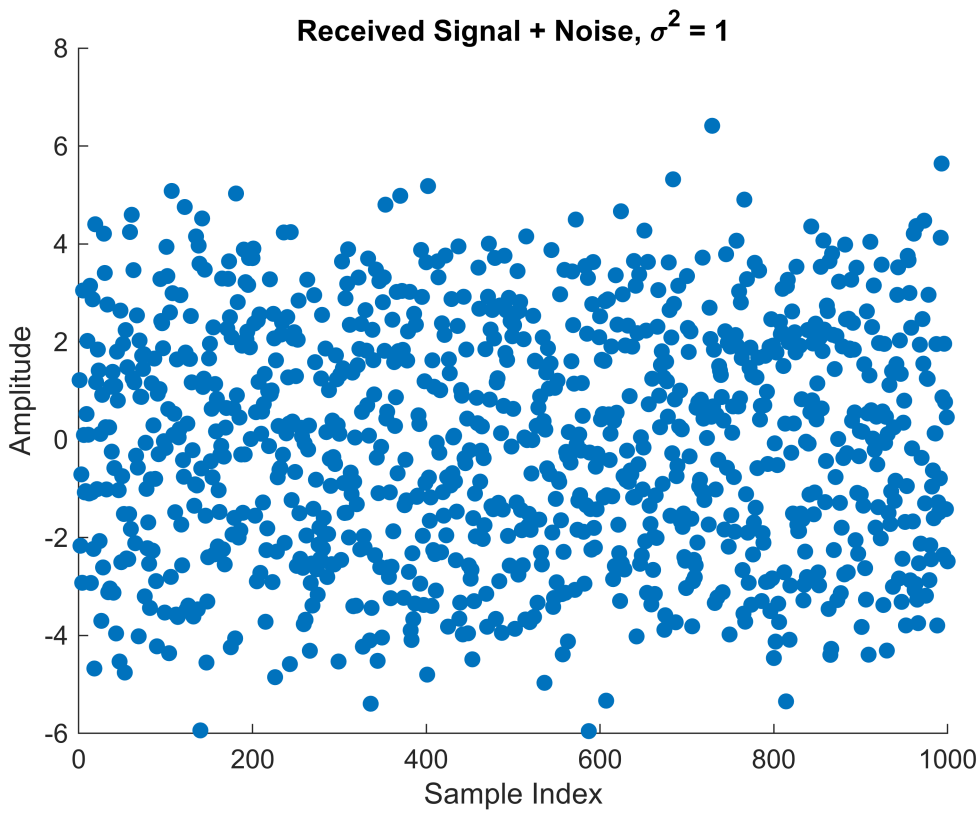
```
symbols = [-3,-1,1,3];
```

```
n_symbol = 10000;
sigma_val = [0,0.1,0.5,1.0];
th_err_rate = zeros(1,length(sigma_val));
sim_err_rate = zeros(1,length(sigma_val));
qfunc = @(x) 0.5 * erfc(x / sqrt(2));


for i = 1:length(sigma_val)
    sigma = sqrt(sigma_val(i));

    tx_sym = symbols(randi(length(symbols),1,n_symbol));

    noise = sigma * randn(1,n_symbol);

    rx_sig = tx_sym + noise;

    det_sym = zeros(1,n_symbol);
    det_sym(rx_sig<-2) = -3;
    det_sym(rx_sig>=-2 & rx_sig< 0) = -1;
    det_sym(rx_sig>=0 & rx_sig<2) = 1;
    det_sym(rx_sig>=2) = 3;

    n_errors = sum(det_sym ~=tx_sym);
    sim_err_rate(i) = n_errors/n_symbol;
    th_err_rate(i) = 2*(1-(1 - qfunc(2/sigma))/2);


    figure;
    scatter(1:1000,rx_sig(1:1000),'filled');
    title(['Received Signal + Noise, \sigma^2 = ',num2str(sigma_val(i))]);
    xlabel('Sample Index');
    ylabel('Amplitude');
end
```


Received Signal + Noise, $\sigma^2$ = 0
Amplitude
Sample Index
3
2
1
0
-1
-2
-3
0
200
400
600
800
1000

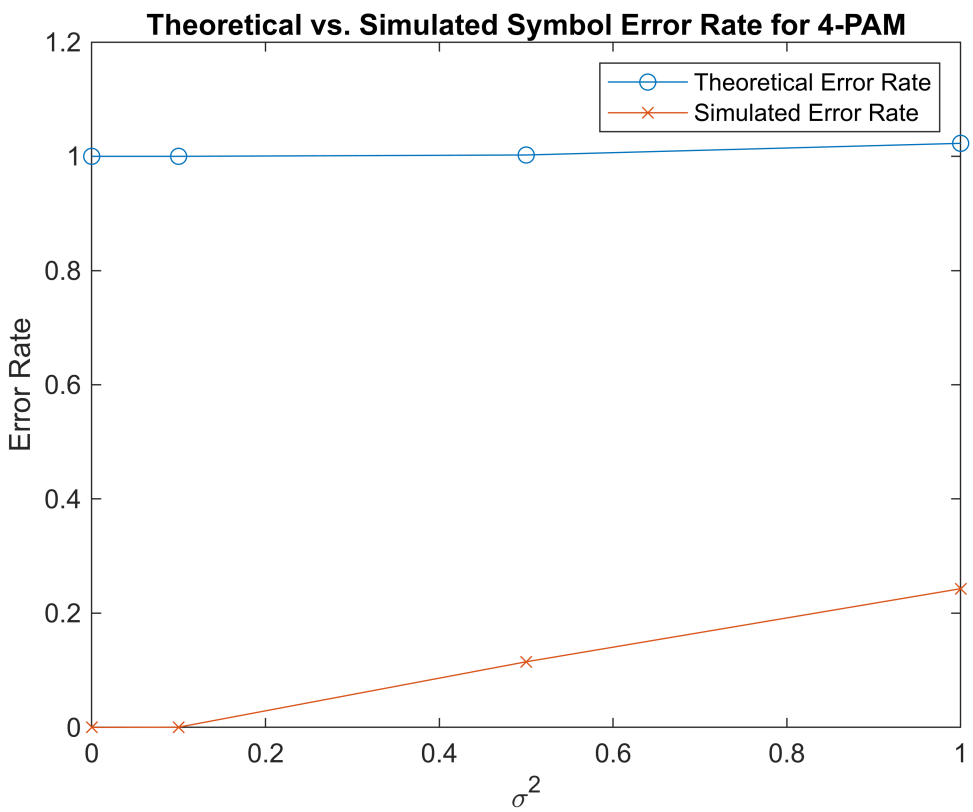

Received Signal + Noise, $\sigma^2$ = 0.1
Amplitude
Sample Index
5
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000




```
figure;
plot(sigma_val, th_err_rate, '-o', 'DisplayName', 'Theoretical Error Rate');
```

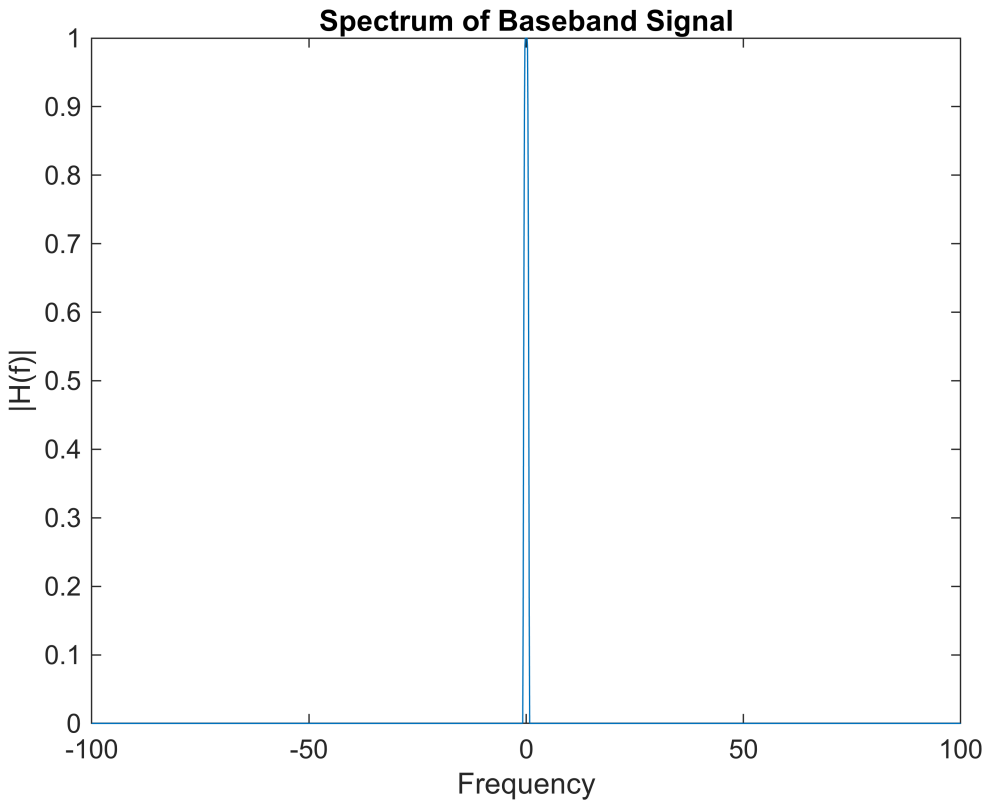
```
hold on;
plot(sigma_val, sim_err_rate, '-x', 'DisplayName', 'Simulated Error Rate');
legend;
xlabel('\sigma^2');
ylabel('Error Rate');
title('Theoretical vs. Simulated Symbol Error Rate for 4-PAM');
```




**Q2)**

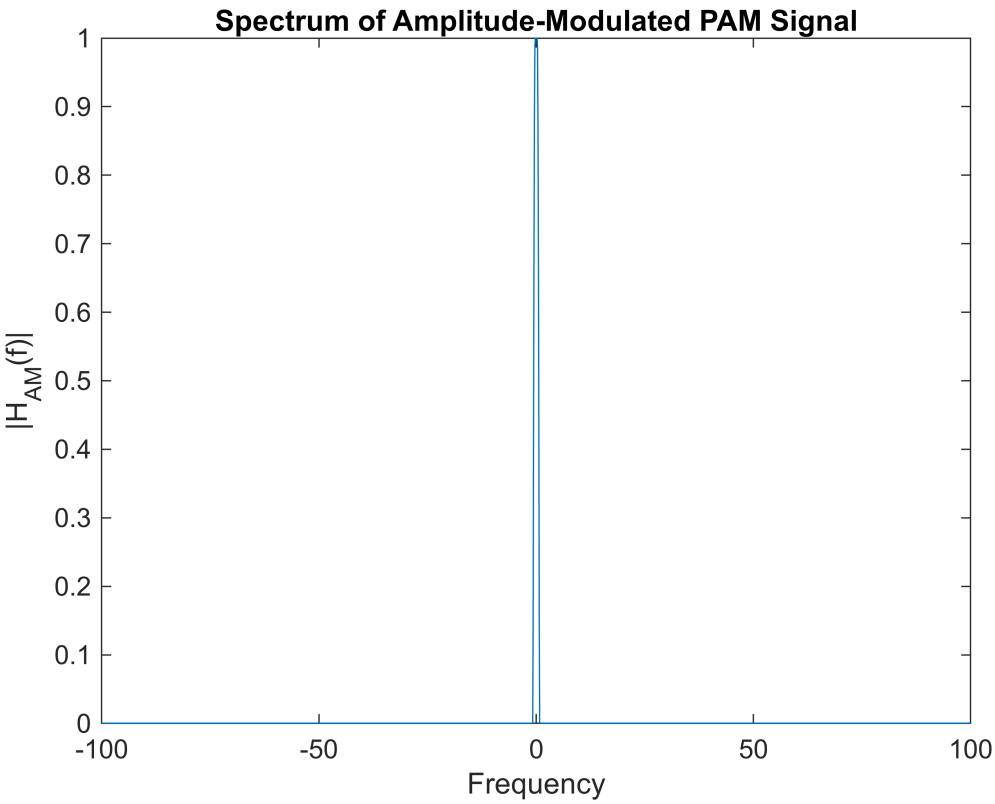
**7.1**

```
T = 1; % Symbol duration
fc = 40 / T; % Carrier frequency
alpha = 0.5; % Rolloff factor for square-root raised-cosine filter

f = -100:0.1:100;

H_f = (abs(f) <= (1 - alpha) / (2 * T)) + ((abs(f) > (1 - alpha) / (2 * T)) &
(abs(f) <= (1 + alpha) / (2 *T))) .*cos(pi * T / (2 * alpha) * (abs(f) - (1 -
alpha) / (2 * T)));

figure;
plot(f, H_f);
title('Spectrum of Baseband Signal');
xlabel('Frequency');
ylabel('|H(f)|');
```



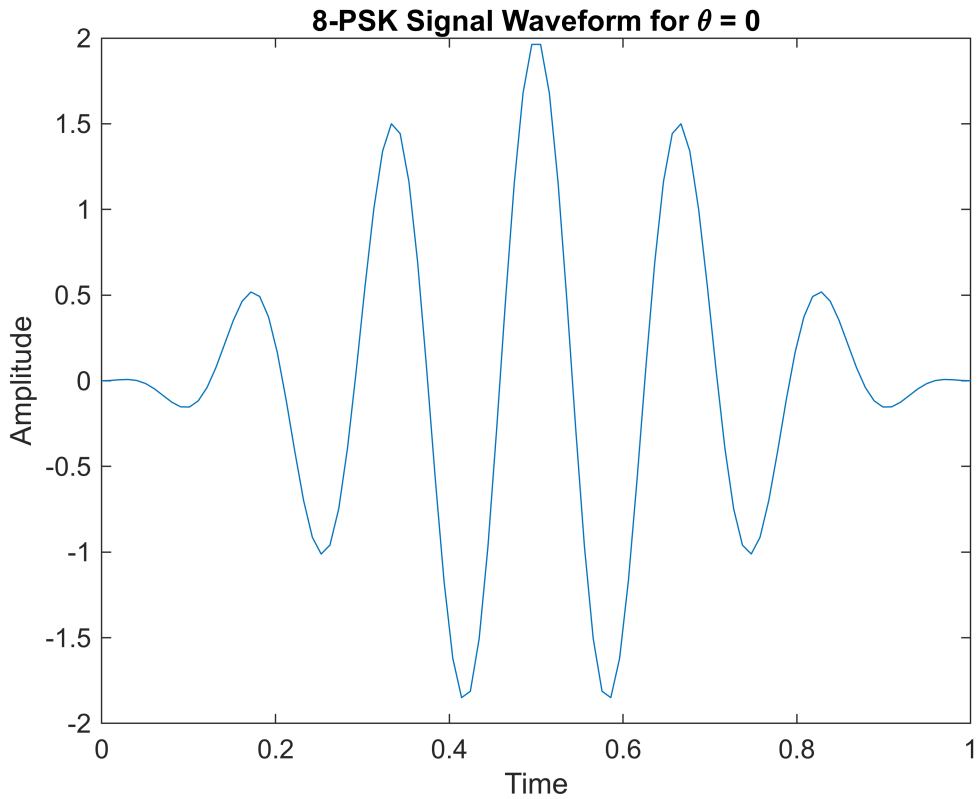

**7.7**

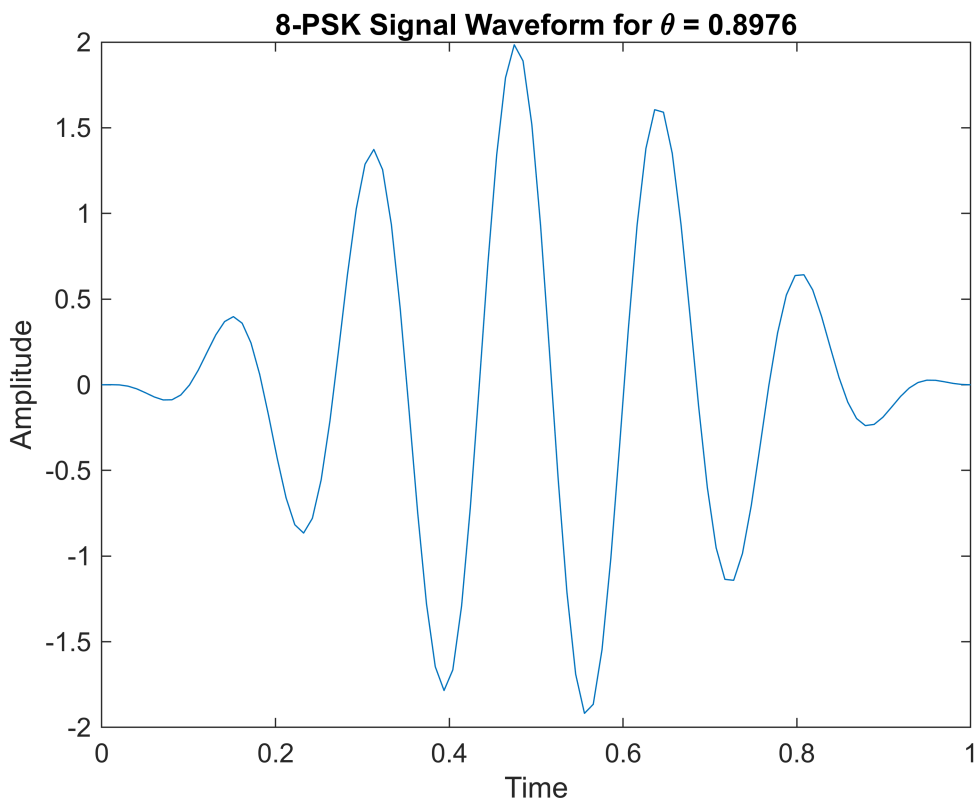
```
% Amplitude-modulated PAM Signal
AM_PAM_signal = cos(2 * pi * fc * T) .* H_f;

% Plot the AM PAM signal spectrum
figure;
plot(f, abs(AM_PAM_signal));
title('Spectrum of Amplitude-Modulated PAM Signal');
xlabel('Frequency');
ylabel('|H_{AM}(f)|');
```

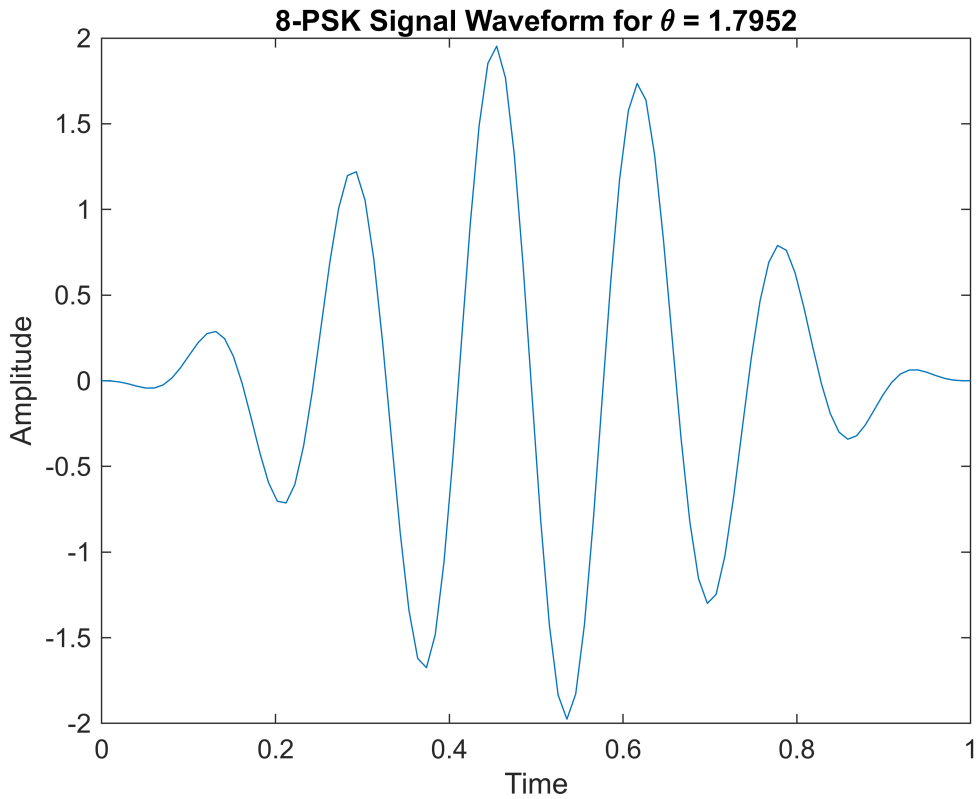


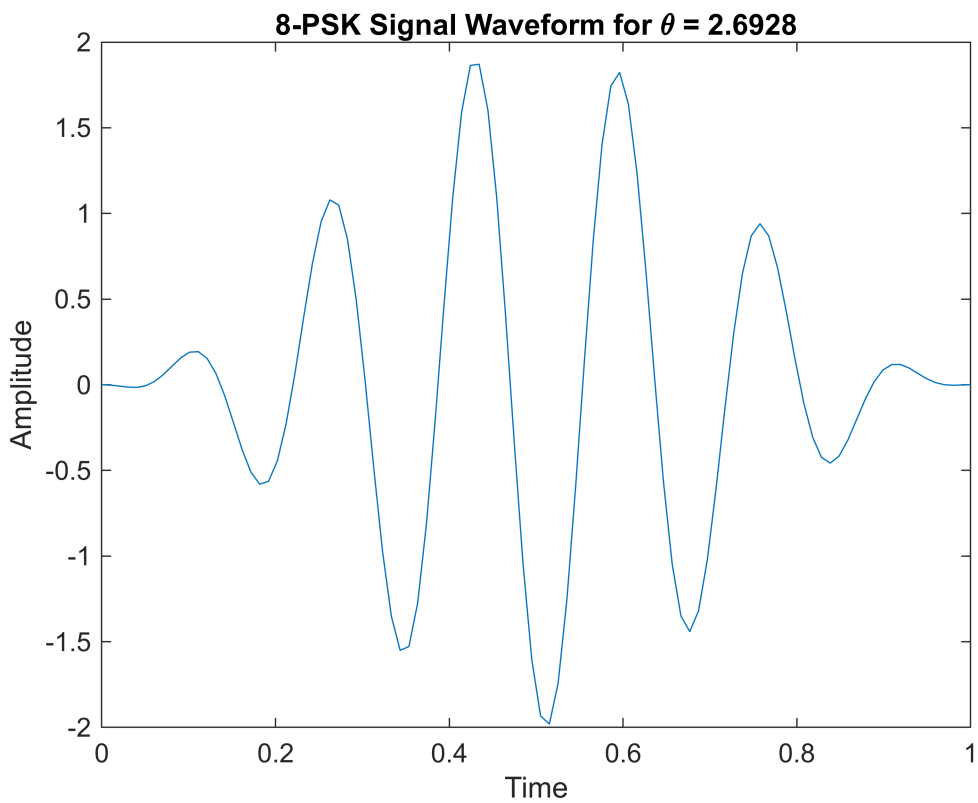

**7.9**

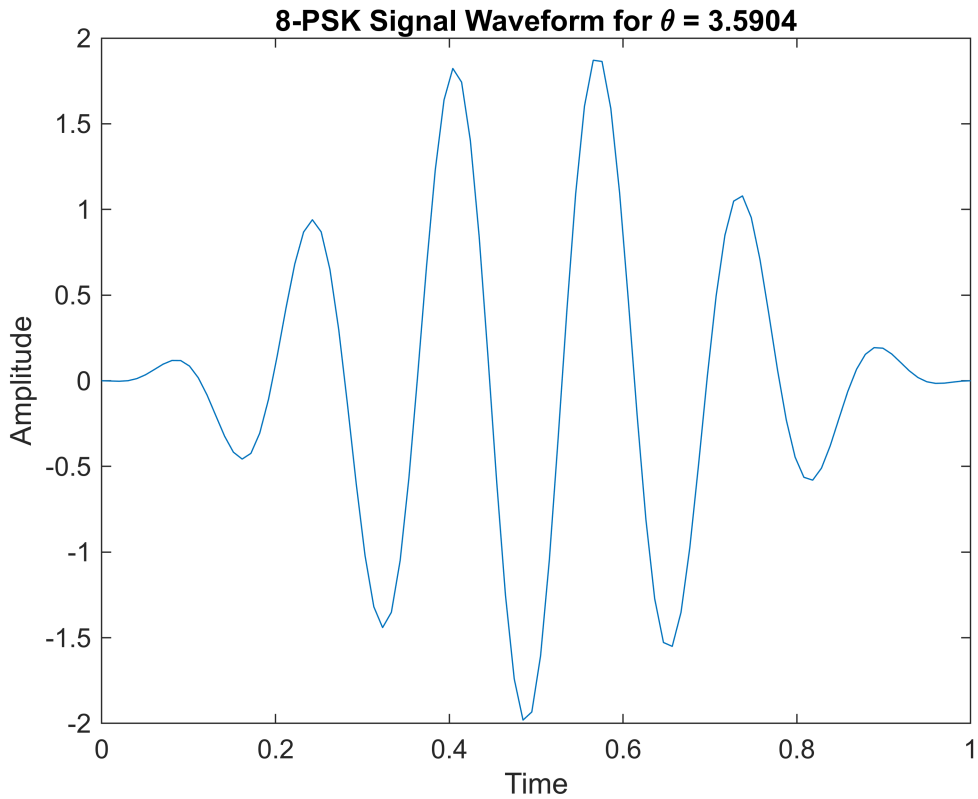
```
T = 1; % Symbol duration
t = linspace(0, T, 100);
gy_t = 1 - cos(2 * pi * t / T);

% Generate 8-PSK signal waveforms
theta = linspace(0, 2 * pi, 8);

for i = 1:8
    ps8_waveform = gy_t .* cos(2 * pi * 6 * T * t + theta(i));
    figure;
    plot(t, ps8_waveform);
    title(['8-PSK Signal Waveform for \theta = ', num2str(theta(i))]);
    xlabel('Time');
    ylabel('Amplitude');
end
```

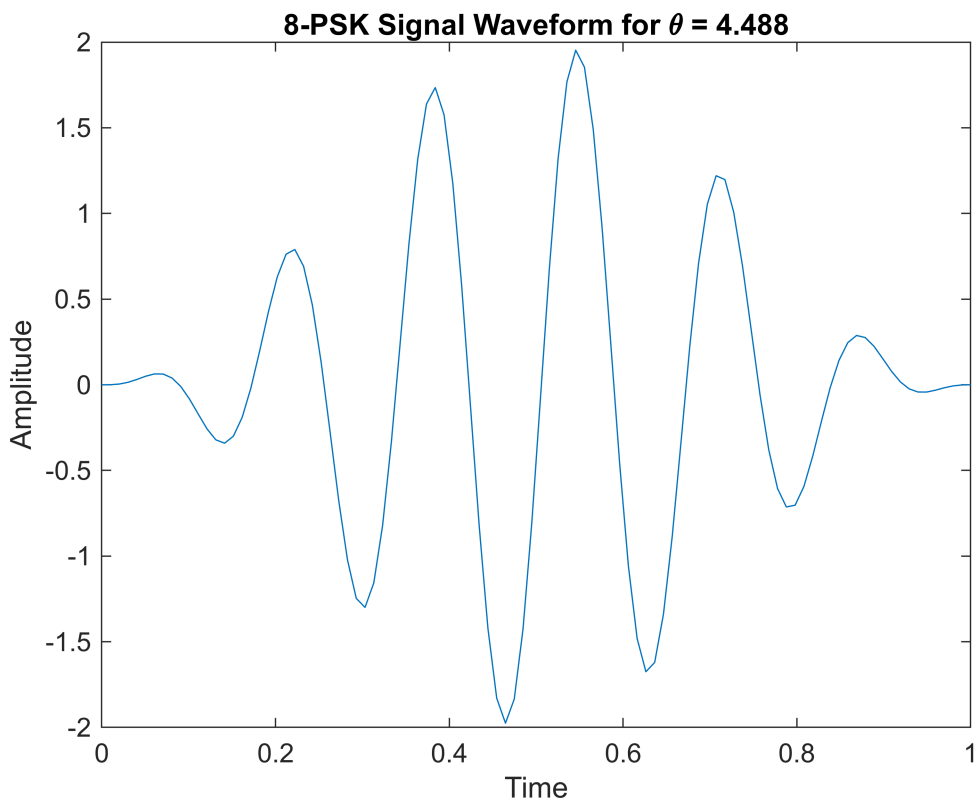

8-PSK Signal Waveform for θ = 0
Amplitude
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.2
0.4
0.6
0.8
1

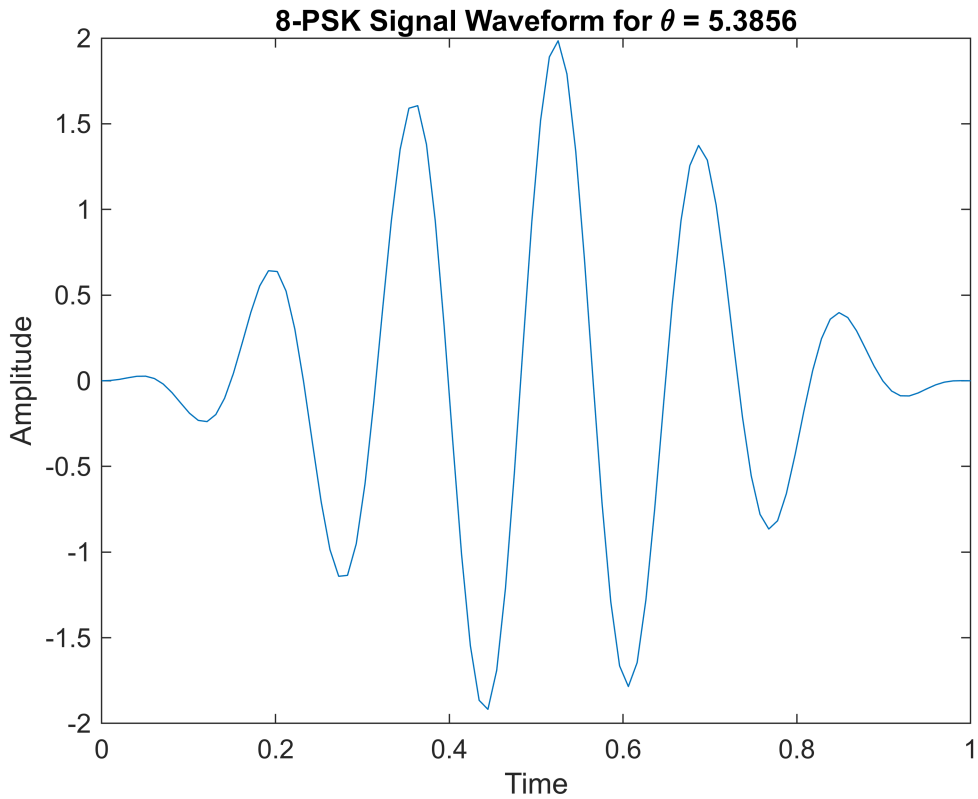

8-PSK Signal Waveform for θ = 0.8976
Amplitude
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.2
0.4
0.6
0.8
1

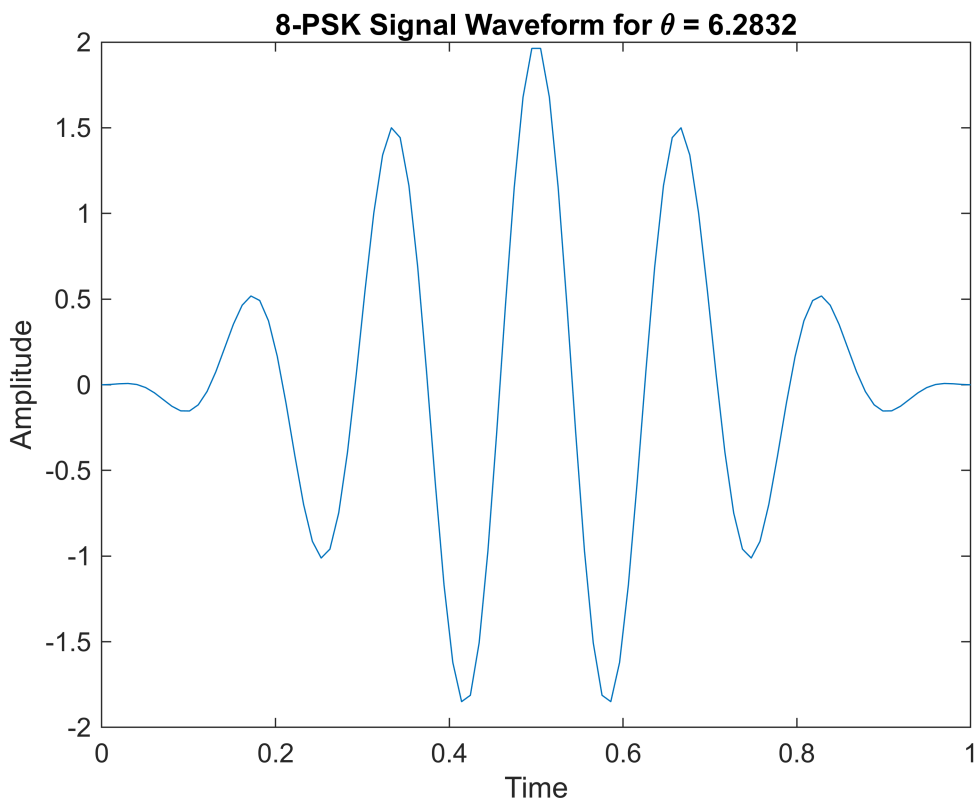

8-PSK Signal Waveform for θ = 1.7952
Amplitude
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.2
0.4
0.6
0.8
1


8-PSK Signal Waveform for θ = 2.6928
Amplitude
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.2
0.4
0.6
0.8
1


8-PSK Signal Waveform for θ = 3.5904
Amplitude
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.2
0.4
0.6
0.8
1


8-PSK Signal Waveform for θ = 4.488
Amplitude
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.2
0.4
0.6
0.8
1



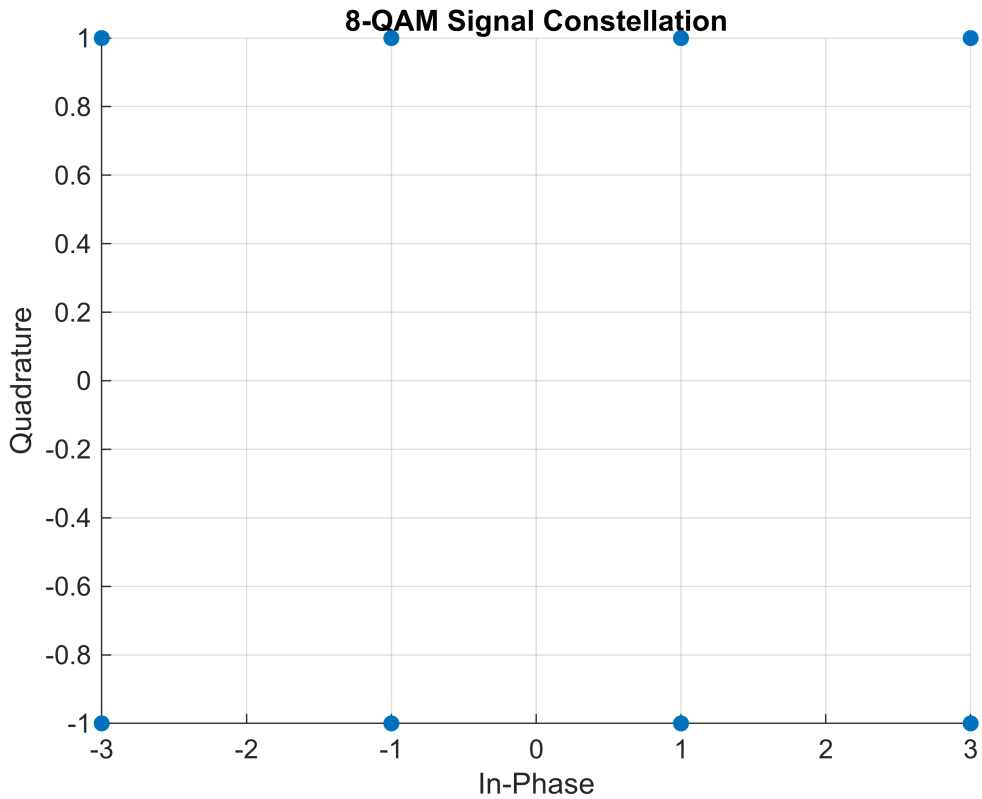

**7.12**

```
N = 10000; % Number of symbols
```

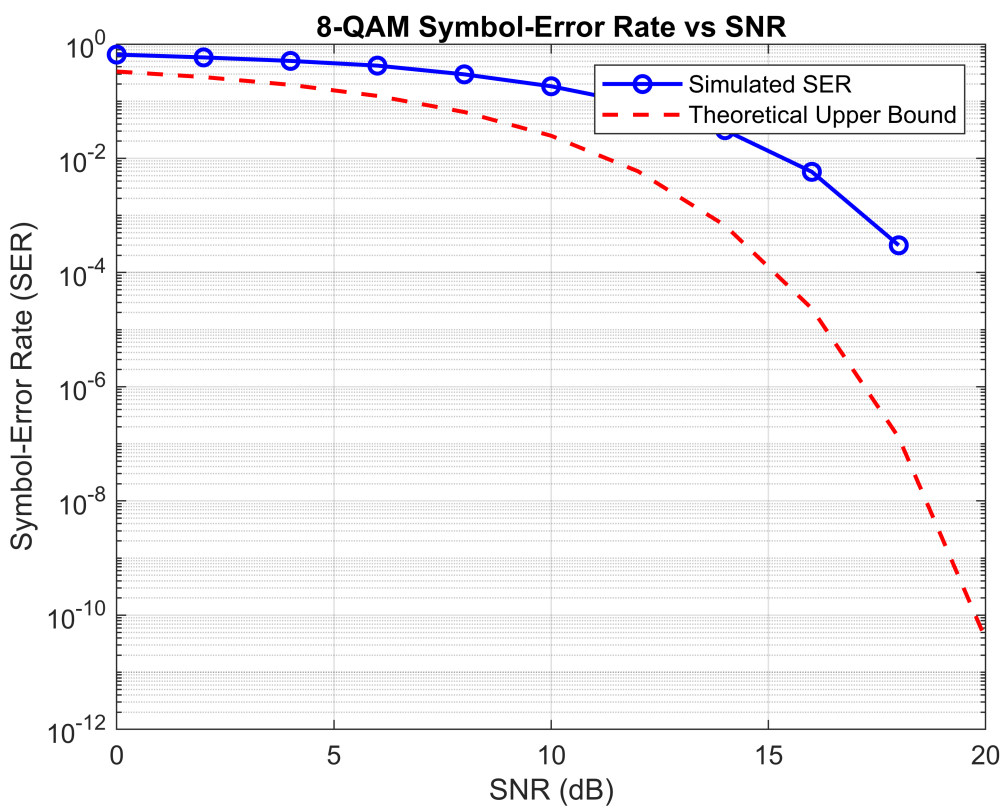
```
M = 4; % 4-PSK
SNR_dB = 10; % SNR in dB
SNR = 10^(SNR_dB / 10); % Linear SNR
sigma = sqrt(1 / (2 * SNR)); % Noise standard deviation

% Symbol generation and mapping
angles = (0:M-1) * 2 * pi / M;
symbols = exp(1j * angles);

% Generate random symbol indices and map them to symbols
symbol_indices = randi([1, M], 1, N); % Random indices from 1 to M
transmitted_symbols = symbols(symbol_indices); % Map indices to symbols

% Add AWGN
noise = sigma * (randn(1, N) + 1j * randn(1, N));
received_symbols = transmitted_symbols + noise;

% Phase detection
received_phases = angle(received_symbols);
detected_symbols = exp(1j * (round(received_phases / (2 * pi / M)) * (2 * pi / M)));

% Calculate symbol error rate
symbol_errors = sum(detected_symbols ~= transmitted_symbols) / N;
disp(['4-PSK Symbol Error Rate: ', num2str(symbol_errors)]);
```

```
4-PSK Symbol Error Rate: 0.3729
```

```
constellation_points = [-3+1j, -3-1j, -1+1j, -1-1j, 1+1j, 1-1j, 3+1j, 3-1j];
figure;
scatter(real(constellation_points), imag(constellation_points), 'filled');
title('8-QAM Signal Constellation');
xlabel('In-Phase');
ylabel('Quadrature');
grid on;
```




**7.17**

```
M = 8; % 8-QAM (3-bit symbols)
N = 10000; % Number of symbols
Eav = 1; % Average symbol energy (normalized)
SNR_dB = 0:2:20; % SNR range in dB
SNR_linear = 10.^(SNR_dB / 10); % Convert SNR to linear scale
sigma2 = Eav ./ (2 * SNR_linear); % Noise variance

% 8-QAM Constellation (Figure P7.12 assumption)
constellation = sqrt(Eav) * [1+1j, 1-1j, -1+1j, -1-1j,3+1j, 3-1j, -3+1j, -3-1j] /
sqrt(10);

SER = zeros(1, length(SNR_dB)); % Initialize SER array
% Monte Carlo Simulation
for k = 1:length(SNR_dB)
    errors = 0;
    for i = 1:N
        % Transmitted symbol
        tx_index = randi([1 M]);
        tx_symbol = constellation(tx_index);

        % Noise generation
        noise = sqrt(sigma2(k)) * (randn + 1j*randn);
        % Received symbol
        rx_symbol = tx_symbol + noise;
```

```
        % Decision rule (minimum Euclidean distance)
        [~, rx_index] = min(abs(rx_symbol - constellation));
        % Error count
        if rx_index ~= tx_index
            errors = errors + 1;
        end
    end

    % Symbol-Error Rate
    SER(k) = errors / N;
end

theoretical_SER = 2 * (1 - 1/sqrt(M)) * qfunc(sqrt(3 * SNR_linear / (M-1)));

% Plot results
figure;
semilogy(SNR_dB, SER, 'bo-', 'LineWidth', 1.5);
hold on;
semilogy(SNR_dB, theoretical_SER, 'r--', 'LineWidth', 1.5);
grid on;
xlabel('SNR (dB)');
ylabel('Symbol-Error Rate (SER)');
legend('Simulated SER', 'Theoretical Upper Bound');
title('8-QAM Symbol-Error Rate vs SNR');
```