% Objective-1: Plot signal constellation diagram of received 8-PSK signal in AWGN

clc;

clear all;

close all;

printf("Name: Jeevesh Wagh\n");

printf("Roll No: 32375\n");

printf("Batch: L8\n");

pkg load communications

% Number of bits to be transmitted using 8-PSK

N = 3000;

% Random input bits generation

x = randi([0,1],1,N);

M = 8; % Number of Symbols in 8-PSK

yy = []; % Initialize symbol list

% Symbol Generation

for j = 1:3:length(x)-2 % Prevent out-of-bounds access

if x(j)==0 && x(j+1)==0 && x(j+2)==0

y = cosd(0) + 1j\*sind(0);

elseif x(j)==0 && x(j+1)==0 && x(j+2)==1

y = cosd(45) + 1j\*sind(45);

elseif x(j)==0 && x(j+1)==1 && x(j+2)==1

y = cosd(90) + 1j\*sind(90);

elseif x(j)==0 && x(j+1)==1 && x(j+2)==0

y = cosd(135) + 1j\*sind(135);

elseif x(j)==1 && x(j+1)==1 && x(j+2)==0

y = cosd(180) + 1j\*sind(180);

elseif x(j)==1 && x(j+1)==1 && x(j+2)==1

y = cosd(225) + 1j\*sind(225);

elseif x(j)==1 && x(j+1)==0 && x(j+2)==1

y = cosd(270) + 1j\*sind(270);

elseif x(j)==1 && x(j+1)==0 && x(j+2)==0

y = cosd(315) + 1j\*sind(315);

end

yy = [yy y]; % Append symbol

end

% Plot Constellation Diagram without Noise

figure;

scatterplot(yy);

title('8-PSK Constellation (No Noise)');

% Set SNR value

EbN0db = 20; % Change this value to see noisy constellation

EbN0 = 10^(EbN0db/10);

% AWGN Channel

n = (1/sqrt(2)) \* (randn(1,length(yy)) + 1j\*randn(1,length(yy)));

sigma = sqrt(1/((log2(M)) \* EbN0));

% Received Symbols

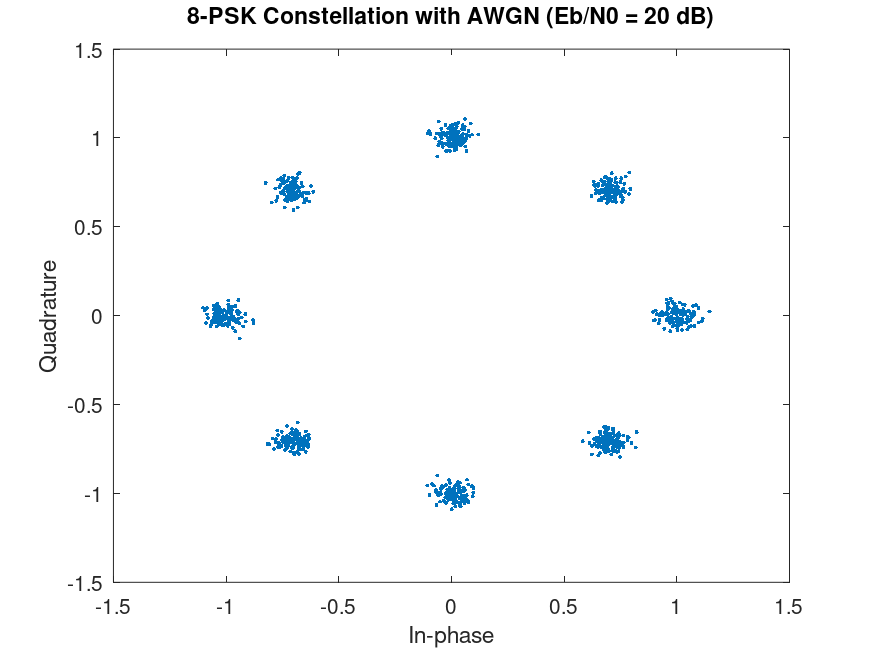
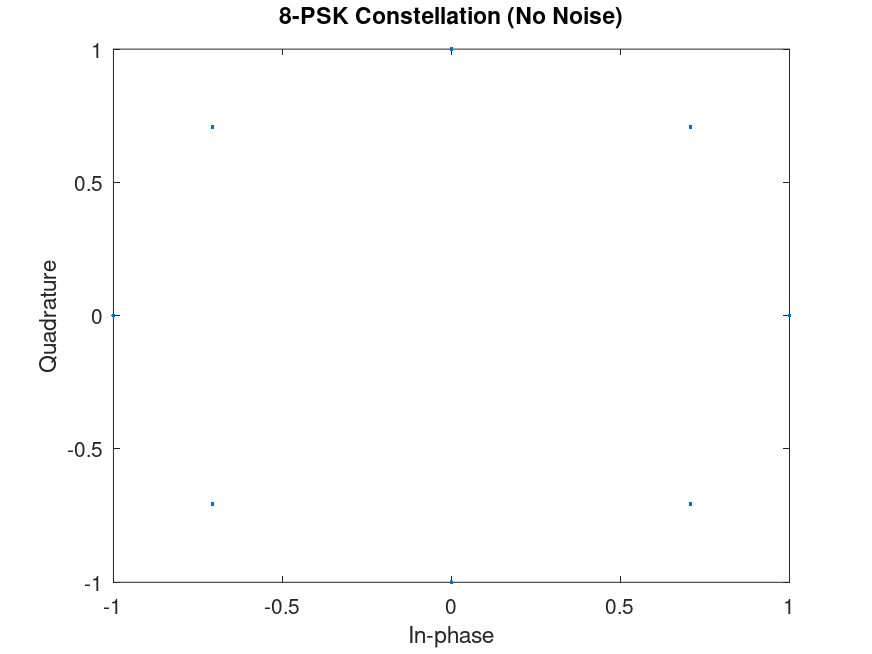
r = yy + sigma \* n;

% Plot Constellation Diagram with Noise

figure;

scatterplot(r);

title(['8-PSK Constellation with AWGN (Eb/N0 = ', num2str(EbN0db), ' dB)']);



% <<Objective-2>>

% Aim: Simulation study of Performance of 8-PSK.

% Objective-2: Plot Practical and Theoretical BER vs SNR graph

% of received 8-PSK signal in the presence of AWGN for ML receiver.

clc;

clear all;

close all;

pkg load communications

N = 3000; % Number of bits to be transmitted using 8-PSK

x = randi([0,1],1,N); % Random input bits generation

M = 8; % Number of Symbols in 8-PSK

yy = [];

% Symbol Generation

for j = 1:3:length(x)-2 % Ensure no out-of-bound access

if x(j)==0 && x(j+1)==0 && x(j+2)==0

y = cosd(0)+1j\*sind(0);

elseif x(j)==0 && x(j+1)==0 && x(j+2)==1

y = cosd(45)+1j\*sind(45);

elseif x(j)==0 && x(j+1)==1 && x(j+2)==1

y = cosd(90)+1j\*sind(90);

elseif x(j)==0 && x(j+1)==1 && x(j+2)==0

y = cosd(135)+1j\*sind(135);

elseif x(j)==1 && x(j+1)==1 && x(j+2)==0

y = cosd(180)+1j\*sind(180);

elseif x(j)==1 && x(j+1)==1 && x(j+2)==1

y = cosd(225)+1j\*sind(225);

elseif x(j)==1 && x(j+1)==0 && x(j+2)==1

y = cosd(270)+1j\*sind(270);

elseif x(j)==1 && x(j+1)==0 && x(j+2)==0

y = cosd(315)+1j\*sind(315);

end

yy = [yy y];

end

% Detection based on Euclidean distance

ber\_simulated = [];

ber\_theoretical = [];

% Define 8-PSK reference symbols

ref\_symbols = [ ...

cosd(0)+1j\*sind(0), ...

cosd(45)+1j\*sind(45), ...

cosd(90)+1j\*sind(90), ...

cosd(135)+1j\*sind(135), ...

cosd(180)+1j\*sind(180), ...

cosd(225)+1j\*sind(225), ...

cosd(270)+1j\*sind(270), ...

cosd(315)+1j\*sind(315)];

for EbN0db = 0:15

EbN0 = 10^(EbN0db/10);

% Add AWGN

n = (1/sqrt(2))\*(randn(1,length(yy)) + 1j\*randn(1,length(yy)));

sigma = sqrt(1 / (log2(M) \* EbN0));

r = yy + sigma \* n;

% ML Detection (Euclidean Distance)

% Calculation of Euclidean Distances of received symbols from reference symbols

min\_dist\_index = [];

for i=1:length(r)

Dist = [];

for k=1:length(ref\_symbols)

dist = sqrt((real(r(i))-real(ref\_symbols(k)))^2 + (imag(r(i))-imag(ref\_symbols(k)))^2);

Dist = [Dist dist];

end

min\_dist\_index = [min\_dist\_index find(Dist==min(Dist))];

end

% Estimate bits from detected symbols

x\_estimated = [];

for i = 1:length(min\_dist\_index)

idx = min\_dist\_index(i);

if ref\_symbols(idx)==cosd(0)+1j\*sind(0)

x\_estimated = [x\_estimated 0 0 0];

elseif ref\_symbols(idx)==cosd(45)+1j\*sind(45)

x\_estimated = [x\_estimated 0 0 1];

elseif ref\_symbols(idx)==cosd(90)+1j\*sind(90)

x\_estimated = [x\_estimated 0 1 1];

elseif ref\_symbols(idx)==cosd(135)+1j\*sind(135)

x\_estimated = [x\_estimated 0 1 0];

elseif ref\_symbols(idx)==cosd(180)+1j\*sind(180)

x\_estimated = [x\_estimated 1 1 0];

elseif ref\_symbols(idx)==cosd(225)+1j\*sind(225)

x\_estimated = [x\_estimated 1 1 1];

elseif ref\_symbols(idx)==cosd(270)+1j\*sind(270)

x\_estimated = [x\_estimated 1 0 1];

elseif ref\_symbols(idx)==cosd(315)+1j\*sind(315)

x\_estimated = [x\_estimated 1 0 0];

end

end

% BER Computation

ber\_simulated = [ber\_simulated sum(x ~= x\_estimated)/N];

% Theoretical BER for 8-PSK (approximate formula)

ber\_theoretical = [ber\_theoretical ((1 / log2(M)) \* erfc(sqrt(3 \* EbN0) \* sind(180 / M)))];

end

% Plotting

EbN0db = 0:15;

figure;

semilogy(EbN0db, ber\_simulated, 'ro-', EbN0db, ber\_theoretical, 'k\*-');

title('BER vs Eb/N0 Plot for 8-PSK');

xlabel('Eb/N0 (dB)');

ylabel('Bit Error Rate (BER)');

grid on;

legend('Simulated', 'Theoretical');

axis([0 15 10^-3 1]);

