% <<Experiment-4 PART-B (8-PSK)>>

% << Objective-2 >>

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

% Objective-1: Write a program to plot signal constellation diagram of received

% 8-PSK signal in the presence of AWGN.

% Objective-2: Write a program to plot Practical and Theoretical BER vs SNR graph

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

% Note: For objective-1, see separate octave files named <my\_8PSK\_constellation.m>

clc;

clear all;

close all;

pkg load communications

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

% Too large value may slow down the program

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

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

% Symbol Generation

yy = [];

for i=1:3:length(x)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

endif

% Transmitted Symbols

yy = [yy y];

endfor

% Detection based on euclidean distance

ber\_simulated = [];

ber\_theoretical = [];

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);

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

sigma = sqrt(1/((log2(M))\*EbN0)); % Symbol energy is normalized to Unity

r = yy + sigma\*n;

% Calculation of Euclidian Distances of received symbols from reference symobols

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];

endfor

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

endfor

% Estimation of Bits

x\_estimated = [];

for i=1:length(r)

if ref\_symbols(min\_dist\_index(i))== cosd(0)+1j\*sind(0);

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

elseif ref\_symbols(min\_dist\_index(i))== cosd(45)+1j\*sind(45);

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

elseif ref\_symbols(min\_dist\_index(i))== cosd(90)+1j\*sind(90);

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

elseif ref\_symbols(min\_dist\_index(i))== cosd(135)+1j\*sind(135);

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

elseif ref\_symbols(min\_dist\_index(i))== cosd(180)+1j\*sind(180);

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

elseif ref\_symbols(min\_dist\_index(i))== cosd(225)+1j\*sind(225);

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

elseif ref\_symbols(min\_dist\_index(i))== cosd(270)+1j\*sind(270);

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

elseif ref\_symbols(min\_dist\_index(i))== cosd(315)+1j\*sind(315);

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

endif

endfor

% BER Computation

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

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

endfor

EbN0db = 0:15;

% BER Plotting

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

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

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

ylabel('BER');

grid on;

legend('Simulated', 'Theoretical');

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