Table of Contents

Hw8 - part II

Teacher : Dr.Emadi TA : Mr.Chalaki Author: [SeyedAli] - [SeyedHosseini] E-mail: [alishosseini79@aut.ac.ir]

```
%Student-Number : [9723042]
% University: Amirkabir University of Technology
```

clear recent data

```
clc;
close all;
clear ;
```

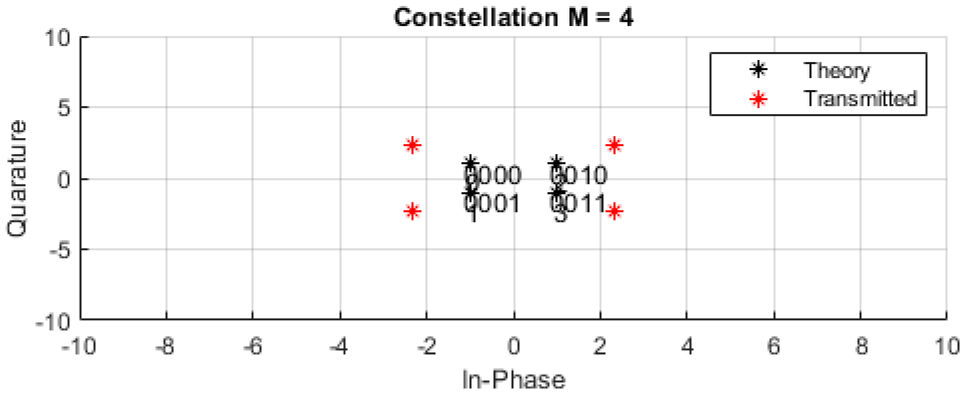
Initialization

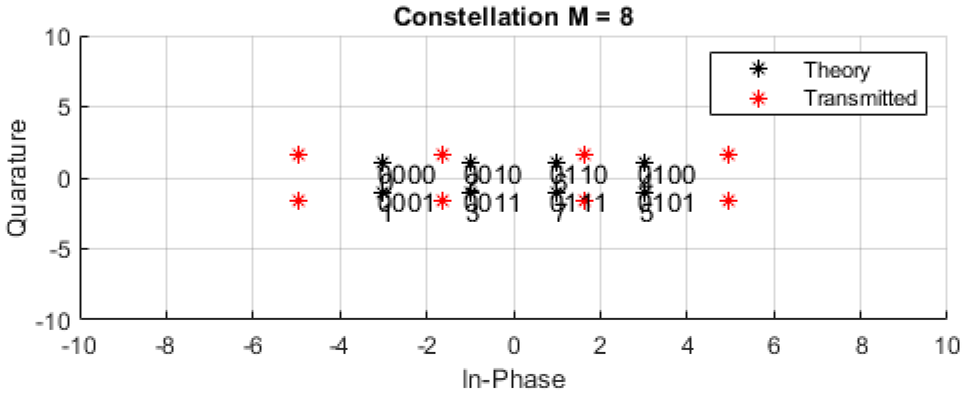
```
clc;
N = 1e3; %Numbers of bits %cant publish with N = 10^5 !
M =[4 8 16 32 64]; %M or # of symbols
E_b = 0 : 0.1 : 13; % in dB
N_0 = 2; %sigma^2 / 2 = 1 => N0 = 2 = sigma^2
SPS = [ 1, 10]; %Symbol per Sample
```

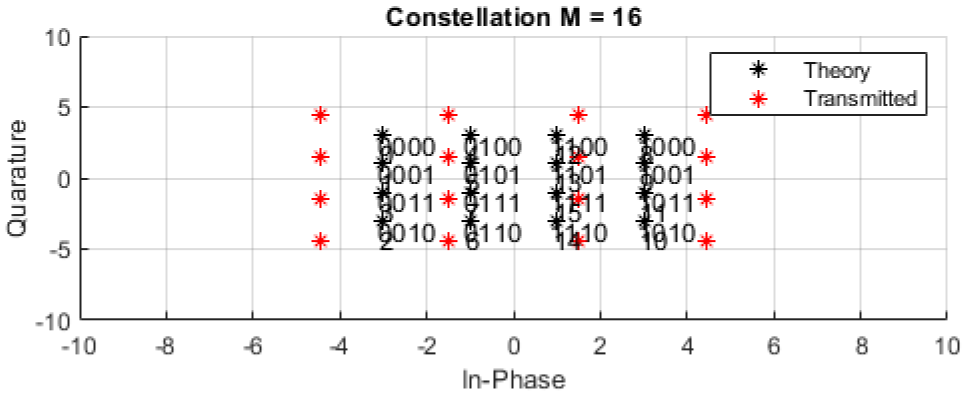
Random bit Generation and scatter plotting

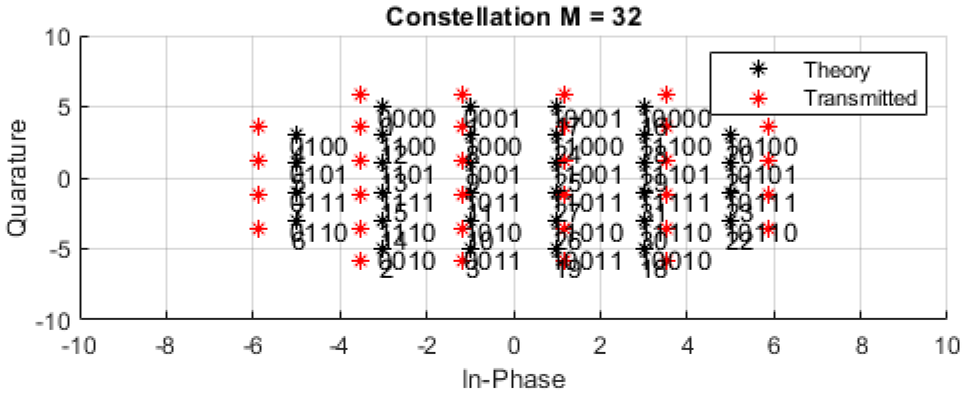
```
clc;
for counter = 1 : length(M)

    data = [0 : M(counter) - 1]; %data generation
    symgray = qammod(data, M(counter), 'gray'); %Modulation by Order
 Gray
    mapgray = qamdemod(symgray, M(counter),'gray'); %DeModulation by
 Order Gray
    numbers = symgray(randi(numel(symgray), [1, N]));%Generation of
 Numbers in Order of #symbols * N
    t = numbers / std(numbers);% Transmitted bits : To Normalize: (x -
 u) / sigma
```

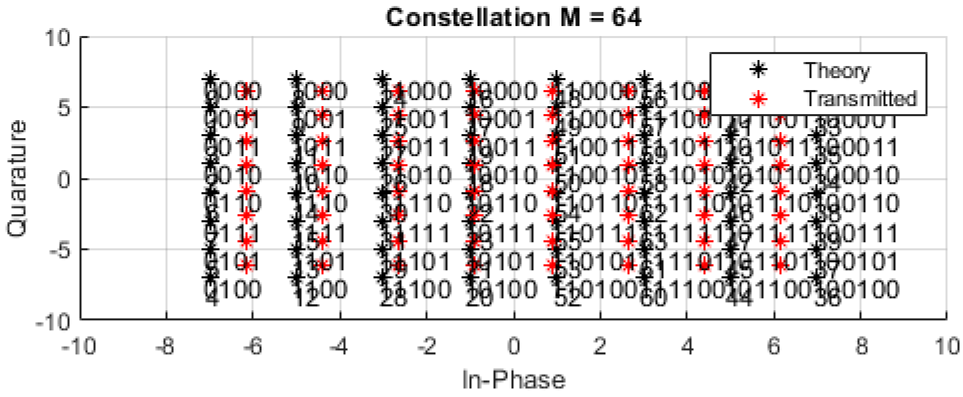
```
   E_s = log2(M(counter)) * 10 .^ (E_b / 10); % Energy of each
symbols
   t = sqrt(E_s)' * t;      % To have Energy E_s all symbols scaled
with sqrt(E_s)
   t_rect = rectpulse(t,SPS(1)); %Repeat symbols
   % Scattering
   figure(counter);
   subplot(2, 1, 1);
   scatter(real(symgray), imag(symgray), '* black');
hold on; %scatter qammod symbols
   scatter(real(t_rect(75, :)), imag(t_rect(75, :)), '*
red');%scatter transmitted symbols
   grid on;
   for k = 1 : M(counter)      % Show the gray code and symbols
sequence #
       text(real(symgray(k)) - 0.15, imag(symgray(k)) - 0.6, ...
           dec2base(mapgray(k), 2, 4));
       text(real(symgray(k)) - 0.1, imag(symgray(k)) - 1.2, ...
           num2str(mapgray(k)));
   end
   axis([-10 10 -10 10])
   legend('Theory', 'Transmitted')
   title(['Constellation M = ', num2str(M(counter))]);
   xlabel('In-Phase');
   ylabel('Quarature');
```




Constellation M = 8
Theory
Transmitted
Quarature
In-Phase
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10
0000
0010
0110
0100
0001
0011
0111
0101
1
3
7
5

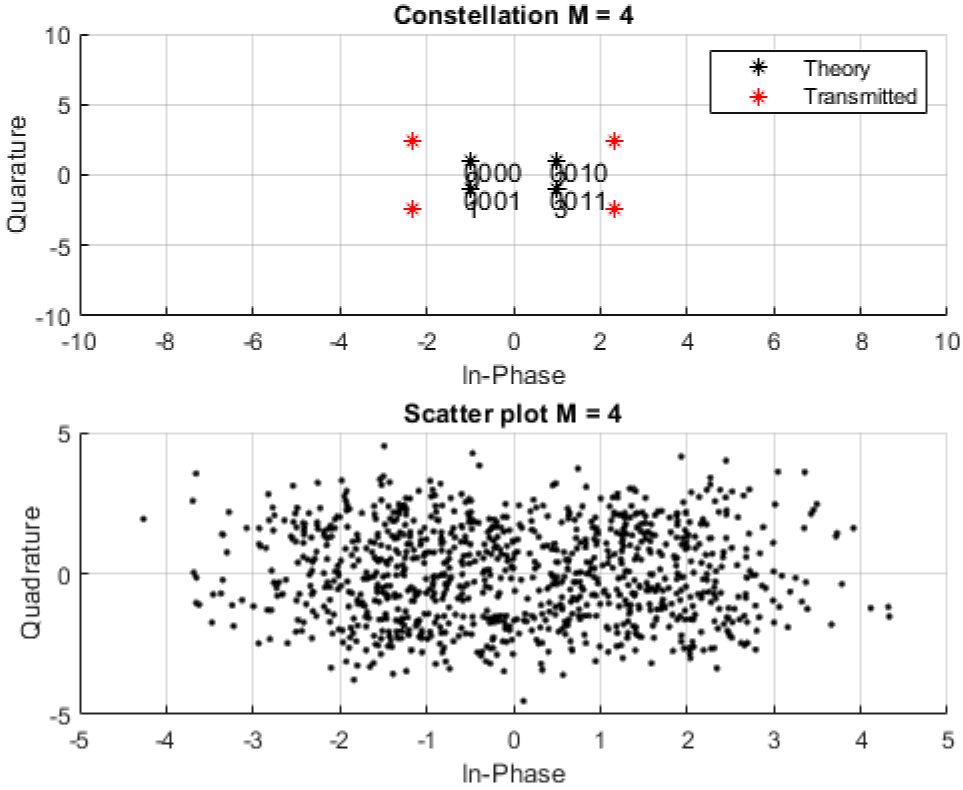

Constellation M = 16
Theory
Transmitted
Quarature
In-Phase
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10
0000
0100
1100
1000
0001
0101
1101
1001
0011
0111
1111
1011
0010
0110
1110
1010
2
6
15
14
10

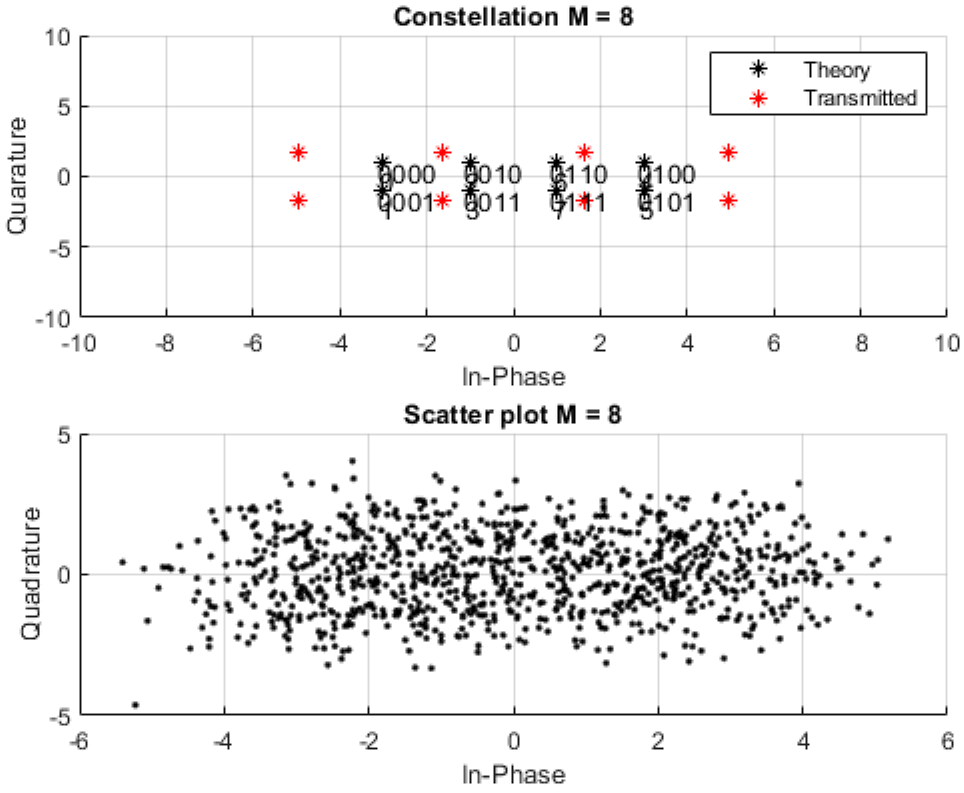

Constellation M = 32
Theory
Transmitted
Quarature
In-Phase
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10

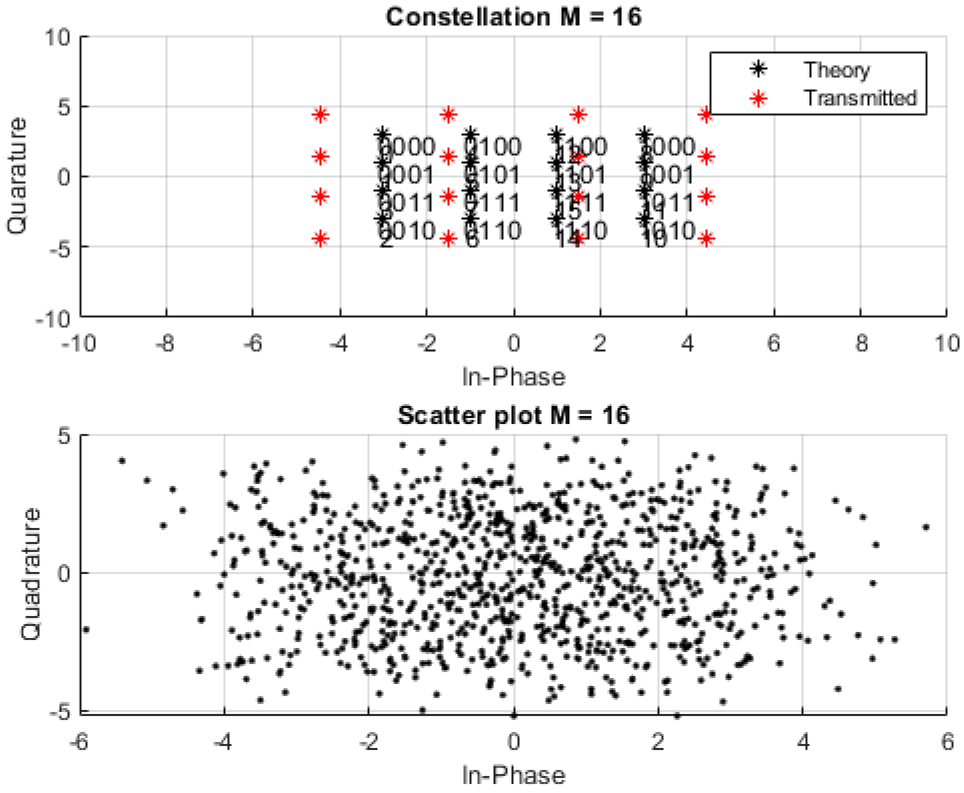

Constellation M = 64
Theory
Transmitted
Quarature
In-Phase
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10

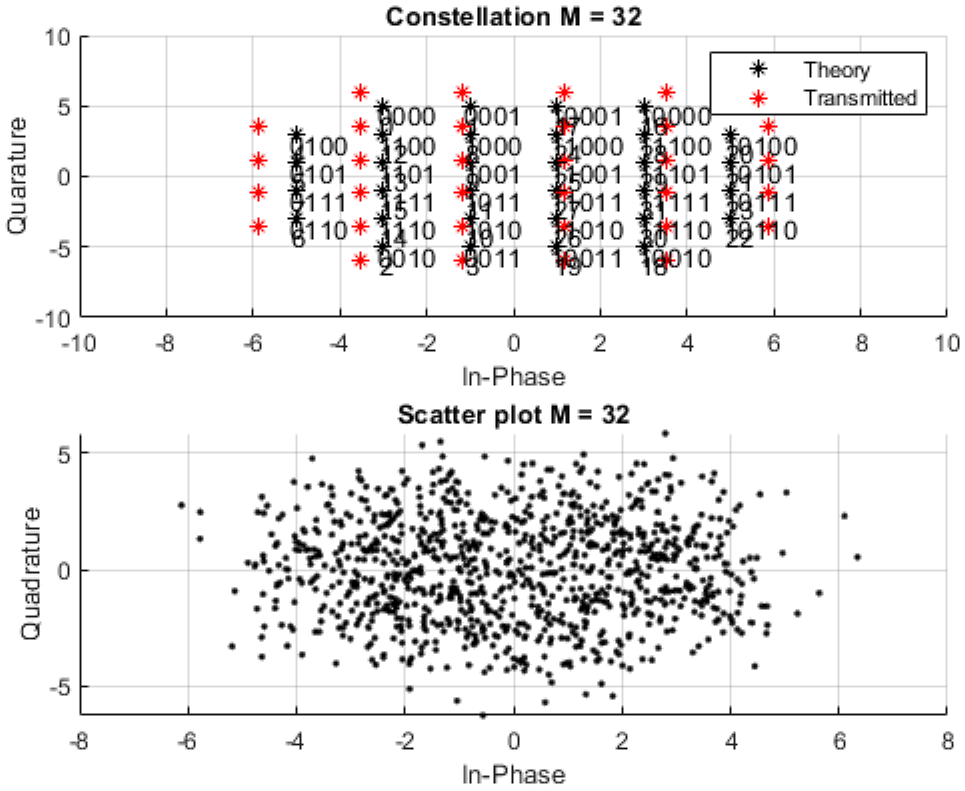
Channel and Noise Generation

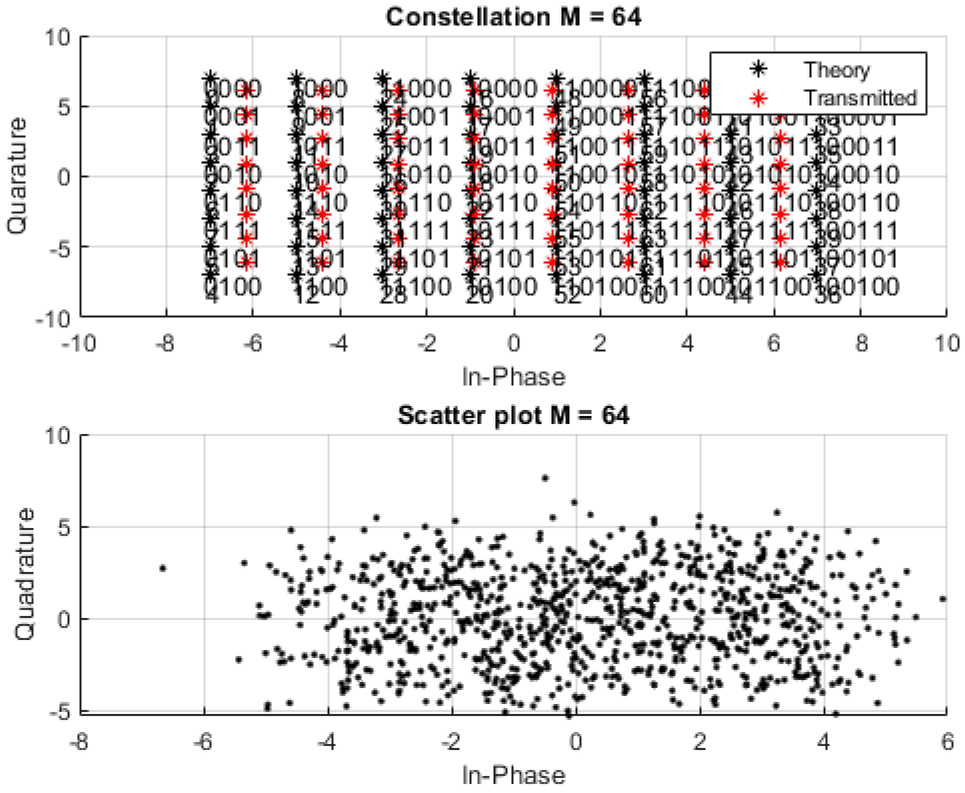
```
    n_i = randn(1, length(t)); %In-Phase noise
    n_q = randn(1, length(t)); %Quadrature noise
    n = sqrt(N_0 / 2) * (n_i / std(n_i) + ...
        1i * n_q / std(n_q));  %Noise generating with variance 1
    r = t_rect + n;      % Recieved Signal
    subplot(2, 1, 2);
    scatter(real(r(25, :)), imag(r(25, :)), '.k'); % Scatter Recieved
signal
    grid on;
    title(['Scatter plot M = ', num2str(M(counter))]);
    xlabel('In-Phase');
    ylabel('Quadrature');
```




Constellation M = 8
Theory
Transmitted
0000
0010
0110
0100
0001
0011
0111
0101
Quarature
In-Phase
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10
Scatter plot M = 8
Quadrature
In-Phase
5
0
-5
-6
-4
-2
0
2
4
6

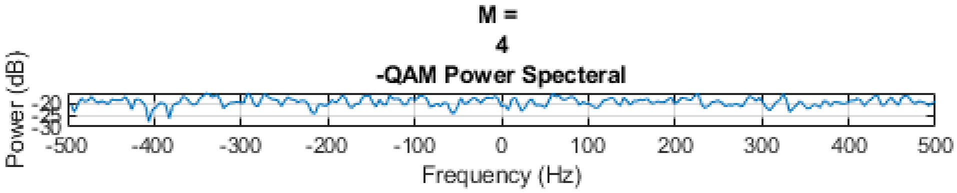

Constellation M = 16
Theory
Transmitted
0000
0100
1100
1000
0001
0101
1101
1001
0011
0111
1111
1011
0010
0110
1110
1010
Quarature
In-Phase
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10
Scatter plot M = 16
Quadrature
In-Phase
5
0
-5
-6
-4
-2
0
2
4
6

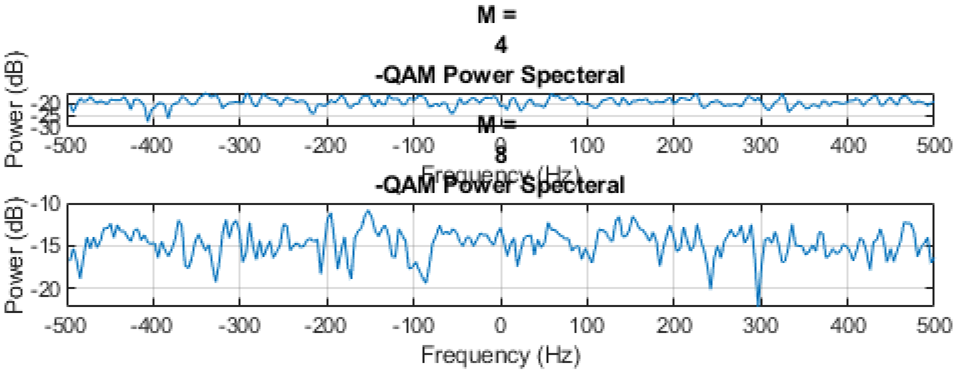

Constellation M = 32
Quarature
In-Phase
Theory
Transmitted
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10
Scatter plot M = 32
Quadrature
In-Phase
5
0
-5
-8
-6
-4
-2
0
2
4
6
8

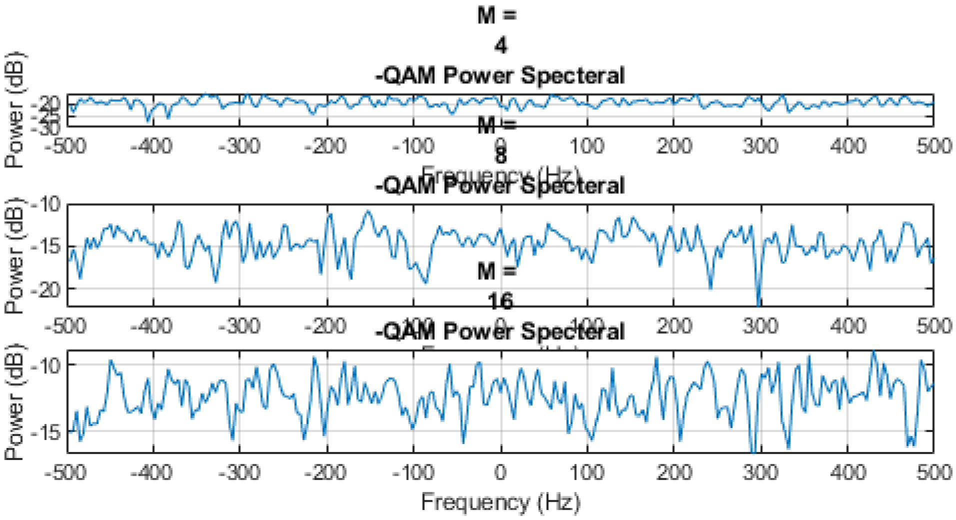

Constellation M = 64
Quarature
In-Phase
Theory
Transmitted
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10
Scatter plot M = 64
Quadrature
In-Phase
10
5
0
-5
-8
-6
-4
-2
0
2
4
6

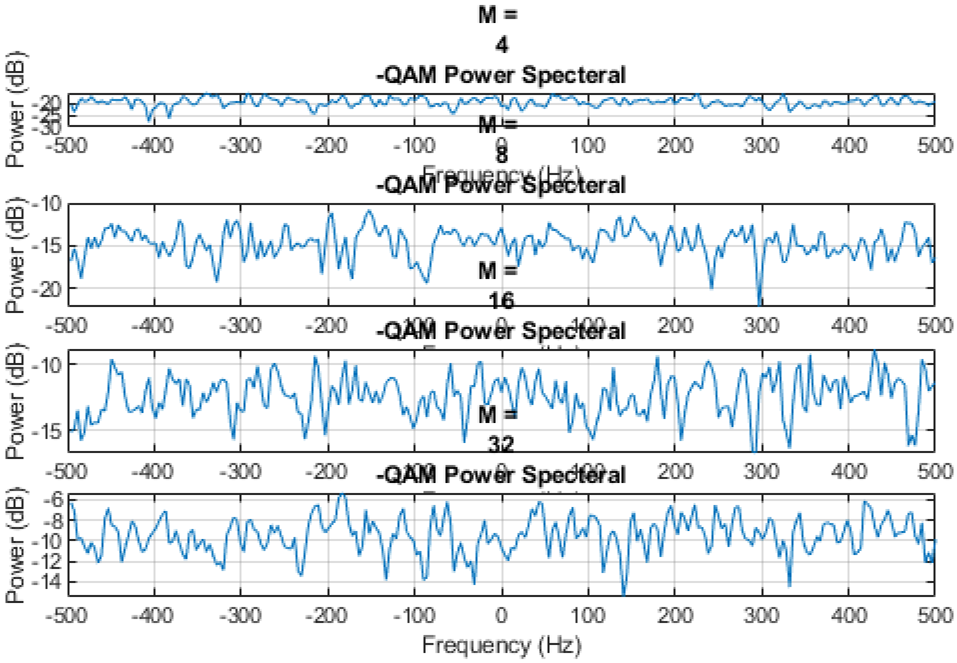
Pwelch

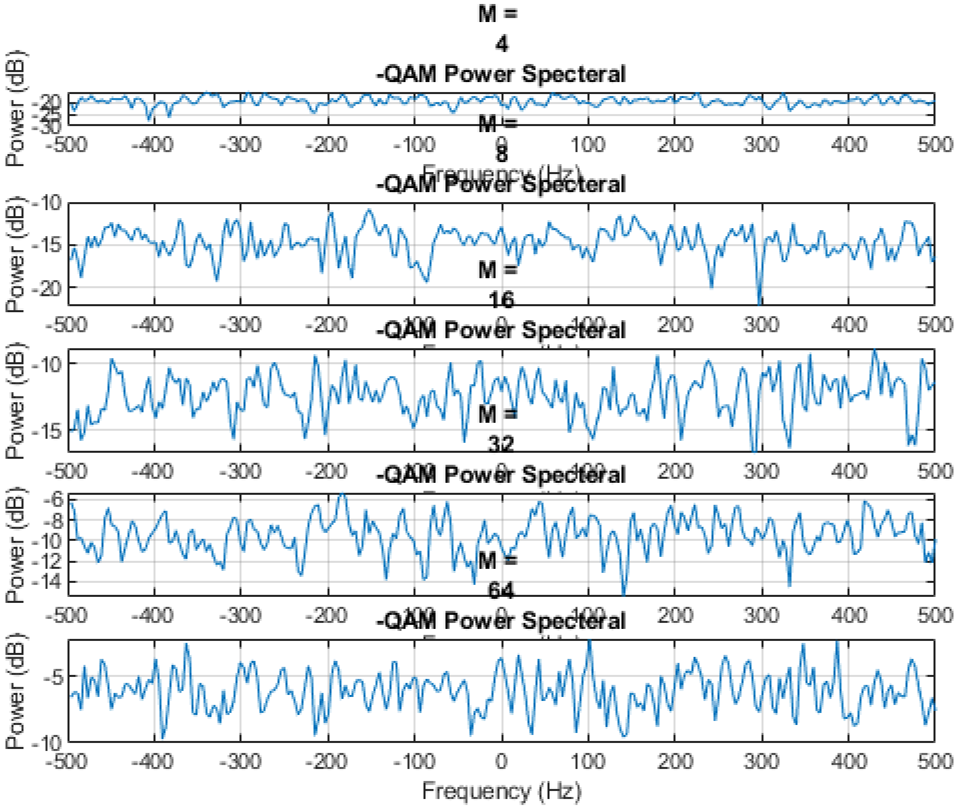
```
clc;
[pxx,f] = pwelch(numbers,[],[],[],1000,'centered','power');
figure(6)
subplot(5,1,counter)
plot(f,pow2db(pxx))
title(["M = ",num2str(M(counter)),"-QAM Power Specteral"])
grid on;
xlabel('Frequency (Hz)')
ylabel('Power (dB)')
```




M =
4
-QAM Power Specteral
Power (dB)
-20
-25
-30
-500
-400
-300
-200
-100
0
100
200
300
400
500
Frequency (Hz)
M =
8
-QAM Power Specteral
Power (dB)
-10
-15
-20
-500
-400
-300
-200
-100
0
100
200
300
400
500
Frequency (Hz)


M =
4
-QAM Power Specteral
Power (dB)
-20
-25
-30
-500
-400
-300
-200
-100
0
100
200
300
400
500
Frequency (Hz)
M =
8
-QAM Power Specteral
Power (dB)
-10
-15
-20
-500
-400
-300
-200
-100
0
100
200
300
400
500
M =
16
-QAM Power Specteral
Power (dB)
-10
-15
-500
-400
-300
-200
-100
0
100
200
300
400
500
Frequency (Hz)


M =
4
-QAM Power Specteral
Power (dB)
-20
-25
-30
M =
8
-QAM Power Specteral
-10
-15
-20
M =
16
-QAM Power Specteral
-10
-15
M =
32
-QAM Power Specteral
-6
-8
-10
-12
-14
-500
-400
-300
-200
-100
0
100
200
300
400
500
Frequency (Hz)


M =
4
-QAM Power Specteral
Power (dB)
-20
-25
-30
M =
8
-QAM Power Specteral
-10
-15
-20
M =
16
-QAM Power Specteral
-10
-15
M =
32
-QAM Power Specteral
-6
-8
-10
-12
-14
M =
64
-QAM Power Specteral
-5
-10
-500
-400
-300
-200
-100
0
100
200
300
400
500
Frequency (Hz)

Decision Point

```
  h = ones(1,2) /2 ; %TO calculate Mean of 2 Near Points
  E_Dec = sqrt(E_s)' * sort(symgray / std(numbers)); %Enveloping
syms by Es
  E_dec = zeros(size(E_Dec, 1), M(counter) + 1);
  for j = 1 : size(E_Dec, 1)
   E_dec(j, :) = conv(E_Dec(j, :), h);
  end
  E_dec(:, M(counter) + 1) = [];      % Removing two unsued data,
  E_dec(:, 1) = [];
```

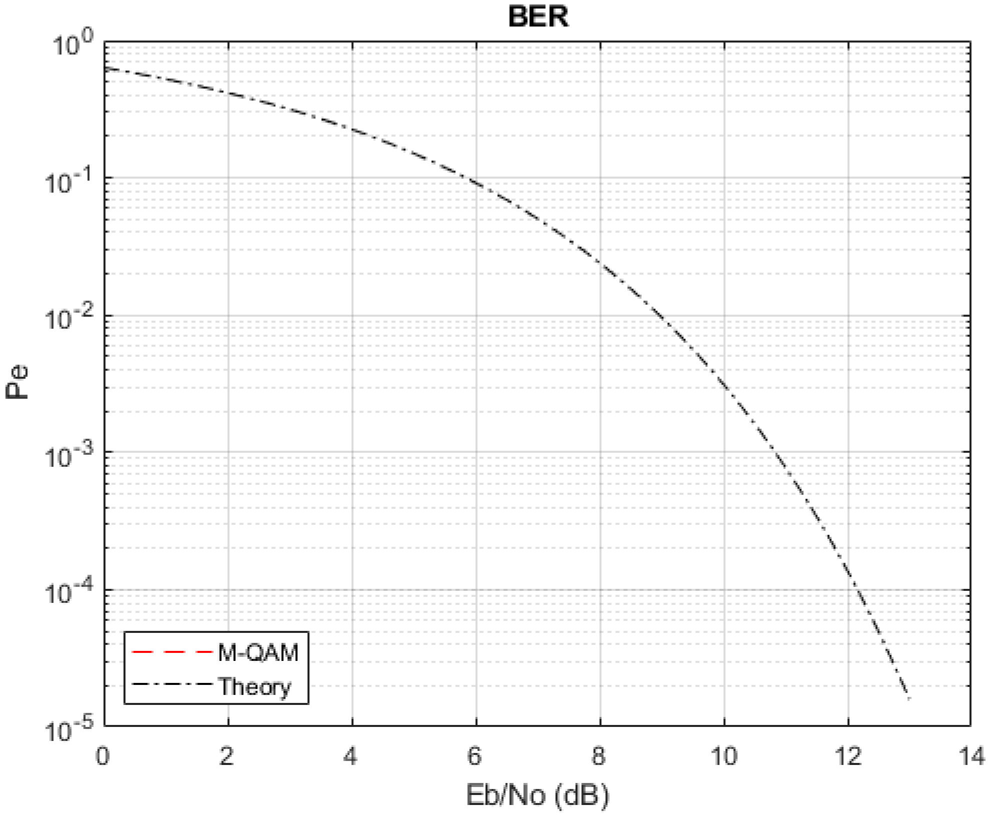
Decision Making

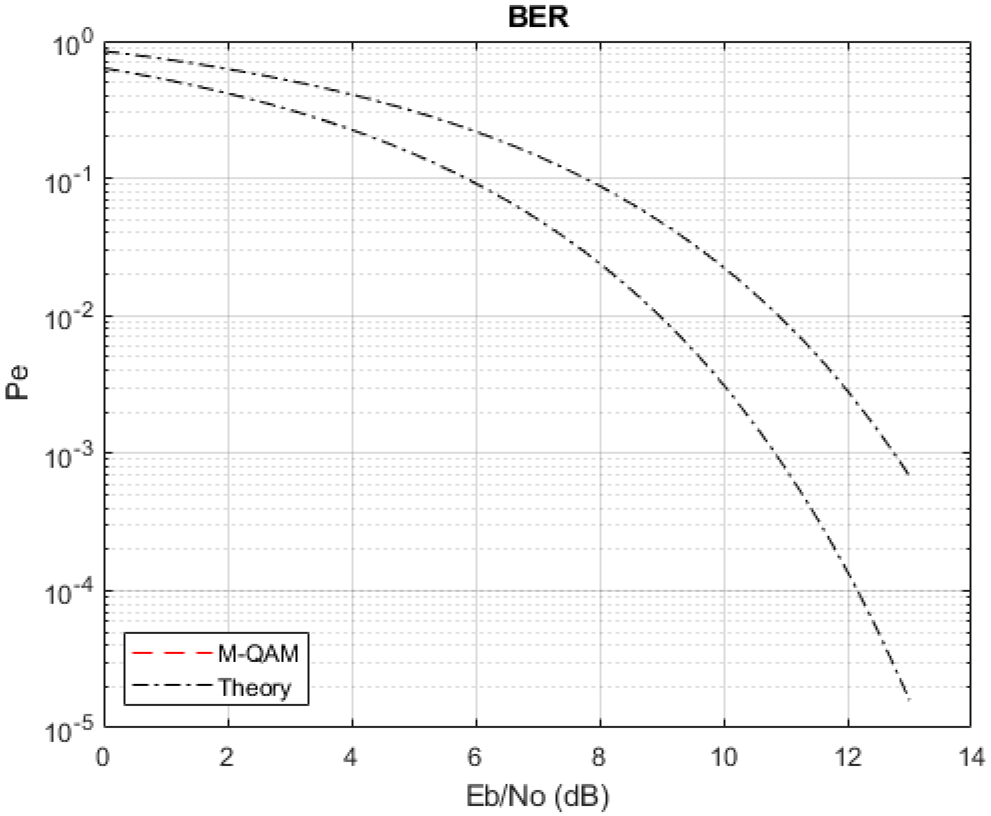
```
clc; % We know that M-QAM is like sqrt(M) PAM in In-Phase and
Quadrature
  decision = zeros(size(r)); %Preallocating
  pe = zeros(length(E_b), 1); %Preallocating
  symgrayR_sort = sort(real(symgray));
  symgrayI_sort = sort(imag(symgray));
  for row = 1 : size(r, 1)    %In This part, We see Edec1`s In phase
and Quadrature part
      for column = 1 : size(r, 2) % and find minimum distance as
threshold
          for i = 2 : size(E_dec,2) %Our goal is to assign bits to
syms in that reigon
              for j = 2 : size(E_dec, 2)  %and for 4 marginal points
we have different if`s
                  if (real(r(row, column)) >= E_dec(row, j - 1)) ...
                          && (real(r(row, column)) < E_dec(row, j))
                      if (imag(r(row, column)) >= E_dec(i - 1,j -
1)) ...
                          && (real(r(row, column)) < E_dec(i, j))
                          decision(row, column) =
symgrayR_sort(i) ...
                              + 1i*symgrayI_sort(j) ; %
                      end
                  end
                  if (real(r(row, column)) < E_dec(row, 1))
                      if(imag(r(row,column)) < E_dec(1,1))
                          decision(row, column) =
symgrayR_sort(1) ...
                              + 1i*symgrayI_sort(1);
                      elseif(imag(r(row,column)) >=
E_dec(size(E_dec, 1),1))
                          decision(row, column) = symgrayR_sort...
                              (length(symgrayR_sort))+
1i*symgrayI_sort(1);
                      end
                  elseif (real(r(row, column)) >= E_dec(row,
size(E_dec, 2)))
```

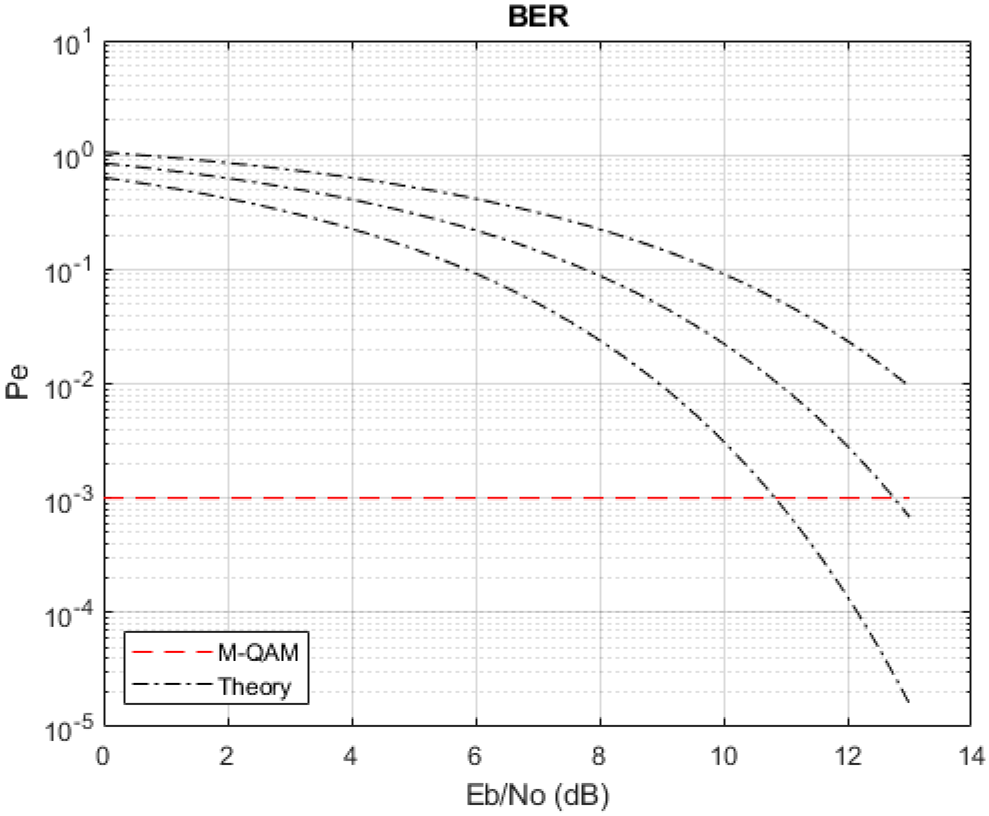
```
                        if(imag(r(row,column)) < E_dec(1,size(E_dec,
 2)))
                            decision(row, column) =
 symgrayR_sort(1) ...
                                +
 1i*symgrayI_sort(length(symgrayI_sort));
                        elseif(imag(r(row,column)) >=
 E_dec(size(E_dec, 1)...
                                ,size(E_dec, 2)))
                            decision(row, column) = symgrayR_sort ...
                                (length(symgrayR_sort))+
 1i*symgrayI_sort....
                                (length(symgrayI_sort));
                        end
                    end
                end
            end
        end
    % Ber Calculation
        if decision(row, column) ~= numbers(1, column)
                    pe(row) = pe(row) + 1;      % Check if the
 decisioned array
                    % - index is equal to transmitted amount
        end
    end
%     %% SPS Demodulation
%     h = ones(1, SPS(2)) / SPS(2);
%     y = zeros(size(E, 1), size(r, 2) + M - 1);
%     for counter = 1 : size(E, 1)
%         y(counter, :) = conv(r(counter, :), h); % Conv for
 normalized sum calculation
%     end
%     temp_normalized = zeros(size(E_s, 1), N); %preallocation
%     for row = 1 : size(E_s, 1)
%         for column = 1 : num_bit
%             temp_normalized(row, column) = y(row, column * SPS(2));
  % Optimum point selection
%         end
%     end
```

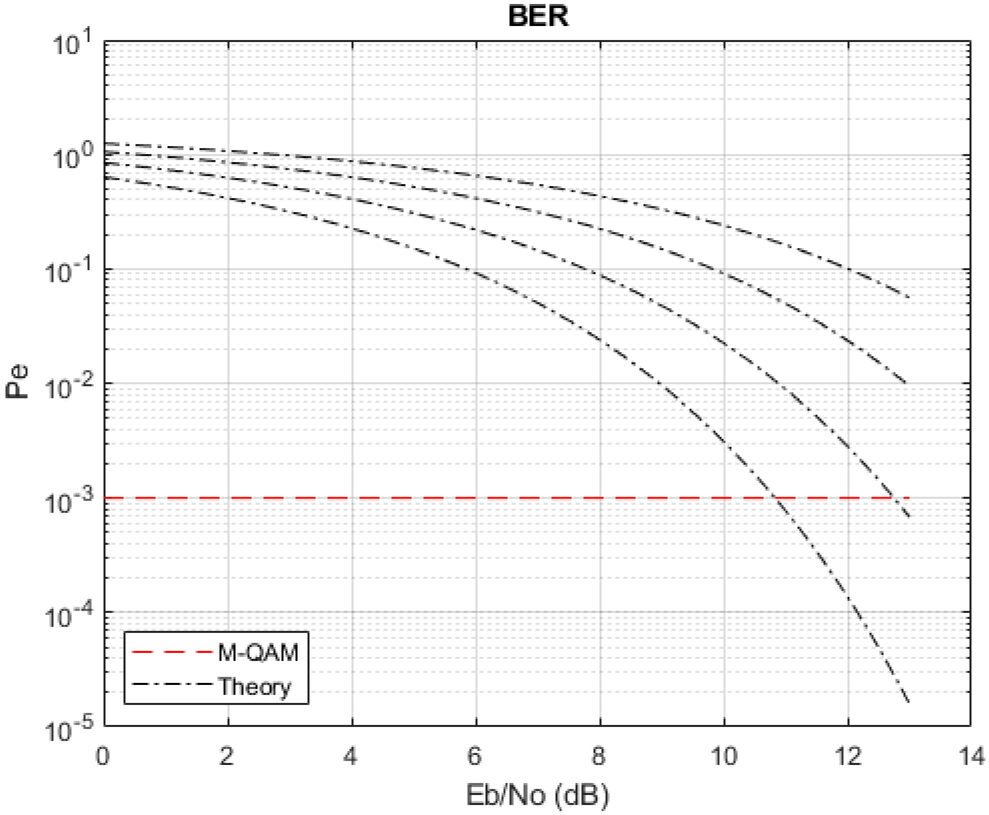
PLOT

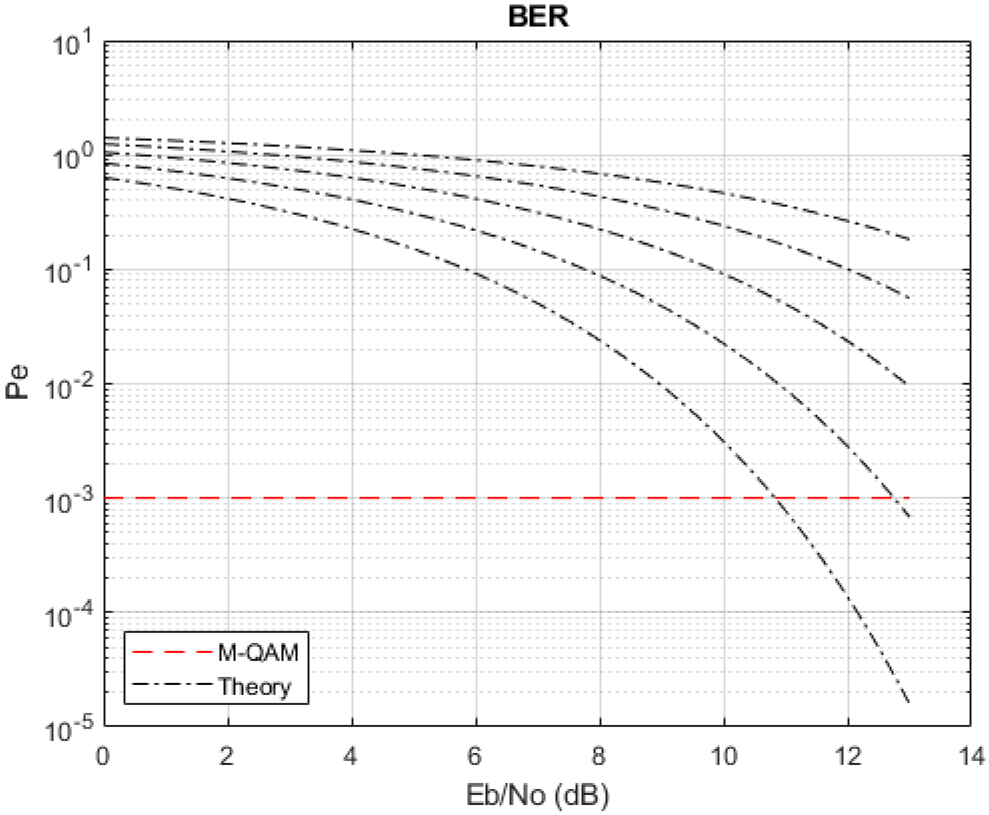
```
pe = pe' / N;     % Normalize the error
% Plotting BER figure
figure(7)
semilogy(E_b, pe, '-- red'); hold on;
Pe_theory=4 * qfunc(sqrt(3 * E_s / ((M(counter) - 1) * N_0)));
semilogy(E_b, Pe_theory, '-. black'); hold on;
xlabel('Eb/No (dB)');
ylabel('Pe');
title('BER');
grid on;
legend('M-QAM', 'Theory', 'Location', 'SouthWest')
```


BER
Pe
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
2
4
6
8
10
12
14
Eb/No (dB)
M-QAM
Theory


BER
Pe
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
2
4
6
8
10
12
14
Eb/No (dB)
M-QAM
Theory


BER
Pe
Eb/No (dB)
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
2
4
6
8
10
12
14
M-QAM
Theory


BER
Pe
Eb/No (dB)
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
2
4
6
8
10
12
14
M-QAM
Theory



```
end
```

Published with MATLAB® R2020b