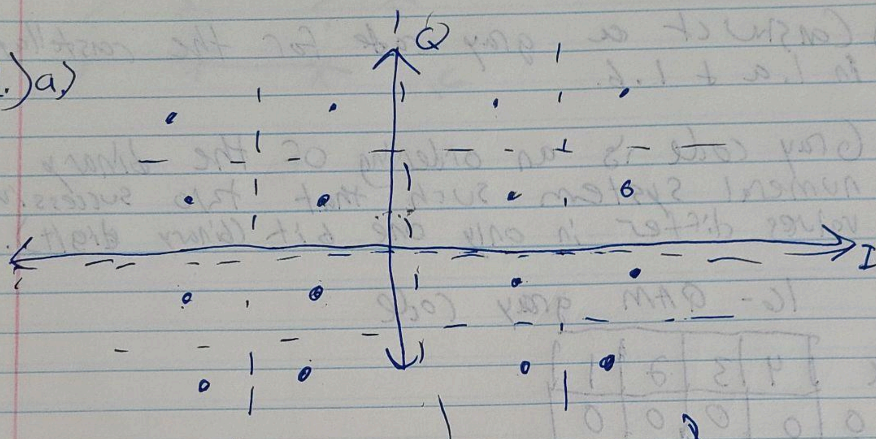
1.)a)



1.b) $\bar{n} = E\{\#\text{ of nearest neighbors}\}$

$\bar{n} = \sum P_i N_i$, where P_i = probability of constellation point i

N_i = Number of nearest neighbors for constellation point.

$$\bar{n} = E\left\{\frac{4 \text{ outer symbols} \times 2 + 4 \text{ inner symbols} \times 4 + 8 \text{ middle symbols} \times 3}{\text{Symbol total} = 16}\right\} = E\left\{\frac{8 + 16 + 24}{16}\right\} = 3$$

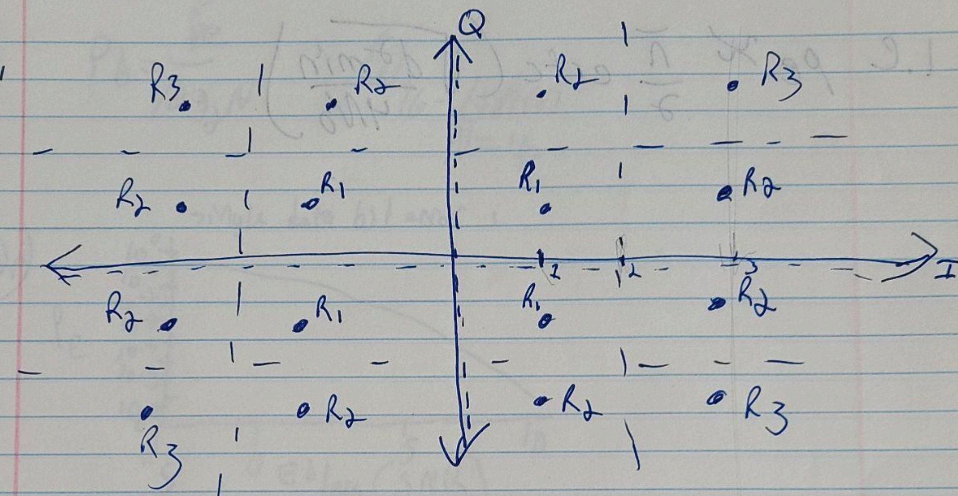
1.c.) Construct a gray code for the constellation in 1.a + 1.b.

Gray code is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit).

16-QAM gray code

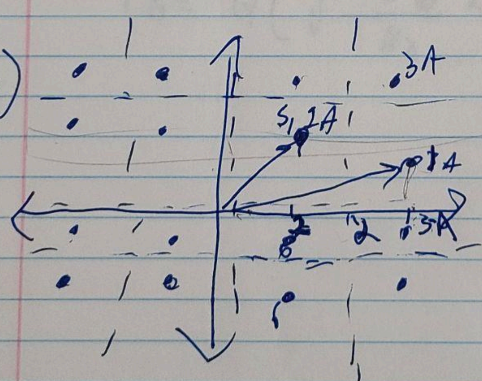
index	4	3	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

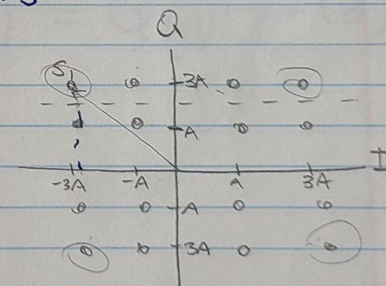
d.



R1 (Region 1) = 4 nearest neighbors
R2 (Region 2) = 3 nearest neighbors
R3 (Region 3) = 2 nearest neighbors

e.)





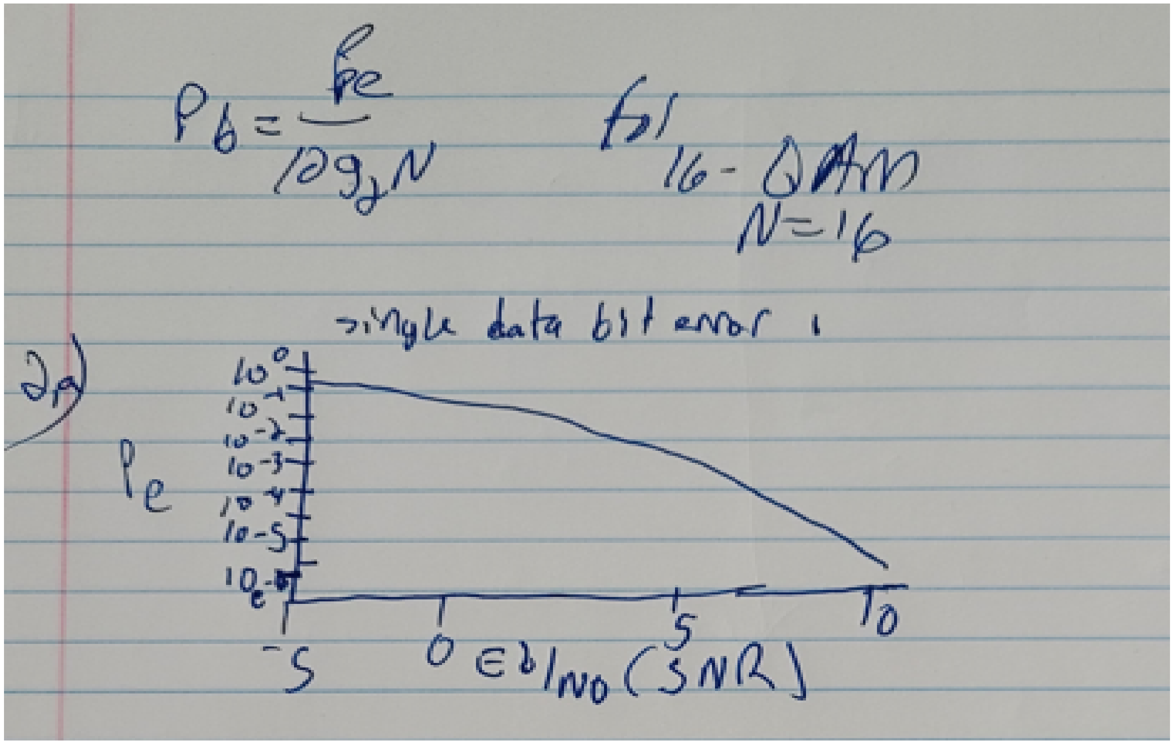
$S_1 = 3A \times (3A) = 9A^2$

$Mag^2(v_E) = \sqrt{(y_2-y_1)^2 + (x_2-x_1)^2}$

$\sqrt{(\ldots)^2 + (-3A-\ldots)^2} = \sqrt{(1A-0)^2 + (3A-0)^2}$

2.

a.


$$P_b = \frac{P_e}{\log_2 N}$$

for 16-QAM
$N=16$

single data bit error ,

2a)

b.

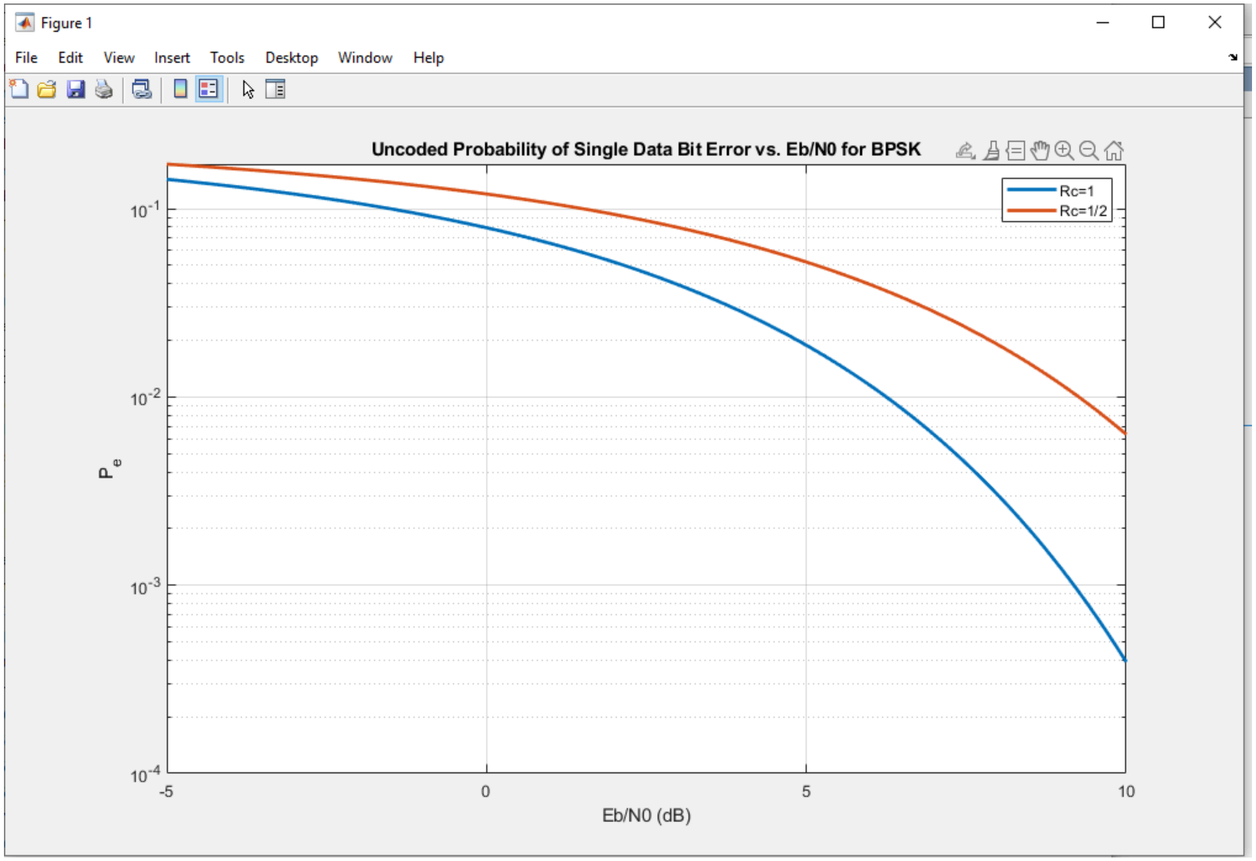


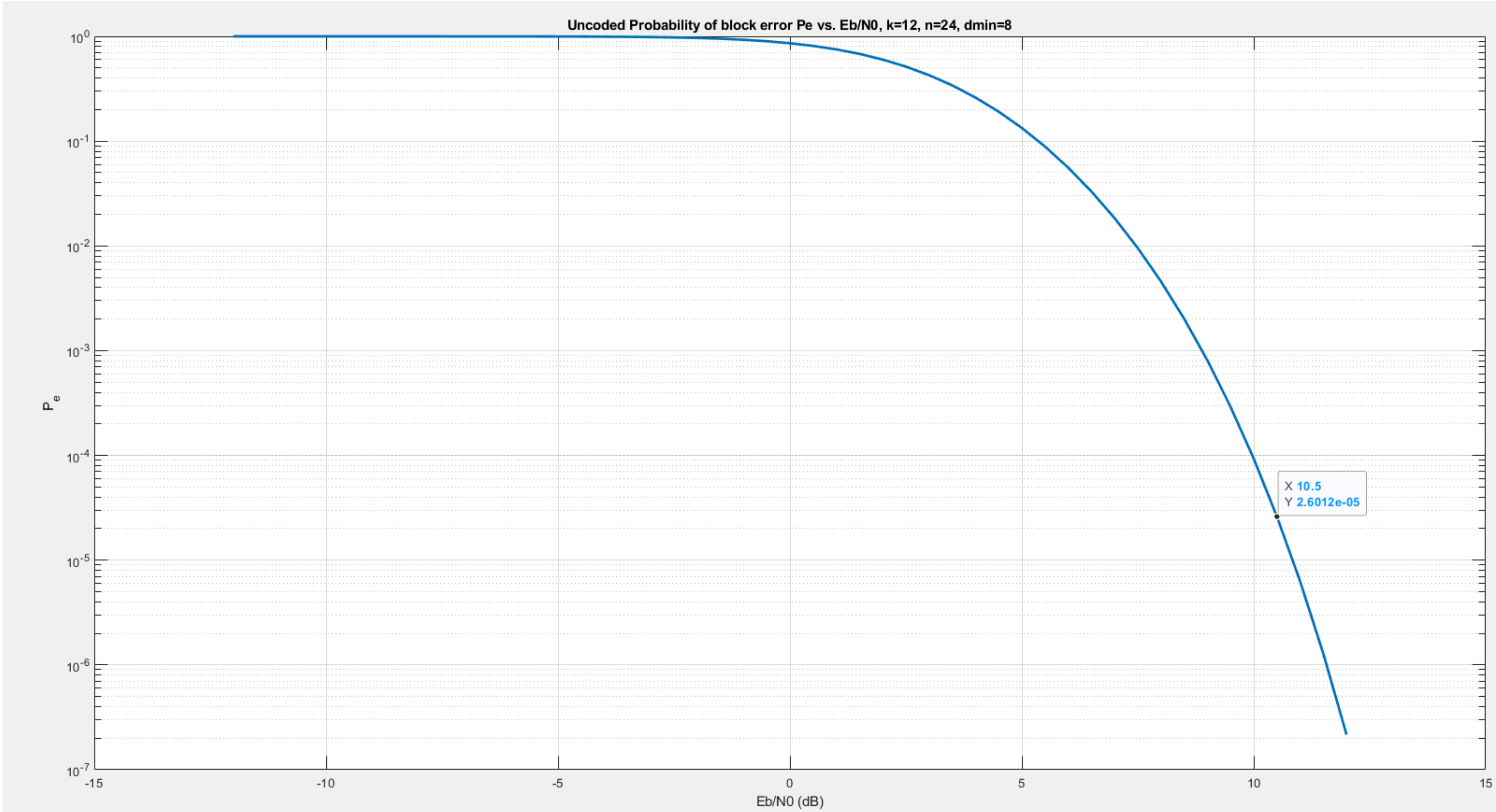
Figure 1
File Edit View Insert Tools Desktop Window Help
Uncoded Probability of Single Data Bit Error vs. Eb/N0 for BPSK
Rc=1
Rc=1/2
10^{-1}
10^{-2}
10^{-3}
10^{-4}
P_e
-5
0
5
10
Eb/N0 (dB)

3.

prelab5_3_a.m

```matlab
%% Problem 3.a
% define block code
k = 12;
n = 24;
dmin = 8;
% Define Eb/N0 values in dB
%EbN0dB = linspace(-5, 10, 100);
EbN0dB = -12:0.5:12; % Range of Eb/N0 values in dB

% Convert Eb/N0 to linear scale
EbN0 = 10.^((EbN0dB / 10));

% Calculate the theoretical Pe for BPSK
Pb_theoretical = qfunc(sqrt(2*EbN0));

% calculate block error probability using binomial distribution formula

Pe_theoretical = 1 - (1-Pb_theoretical).^n;

% Plot the results
figure()
semilogy(EbN0dB, Pe_theoretical, 'LineWidth', 2);
title('Uncoded Probability of block error Pe vs. Eb/N0, k=12, n=24, dmin=8');
xlabel('Eb/N0 (dB)');
ylabel('P_e');
grid on;
```

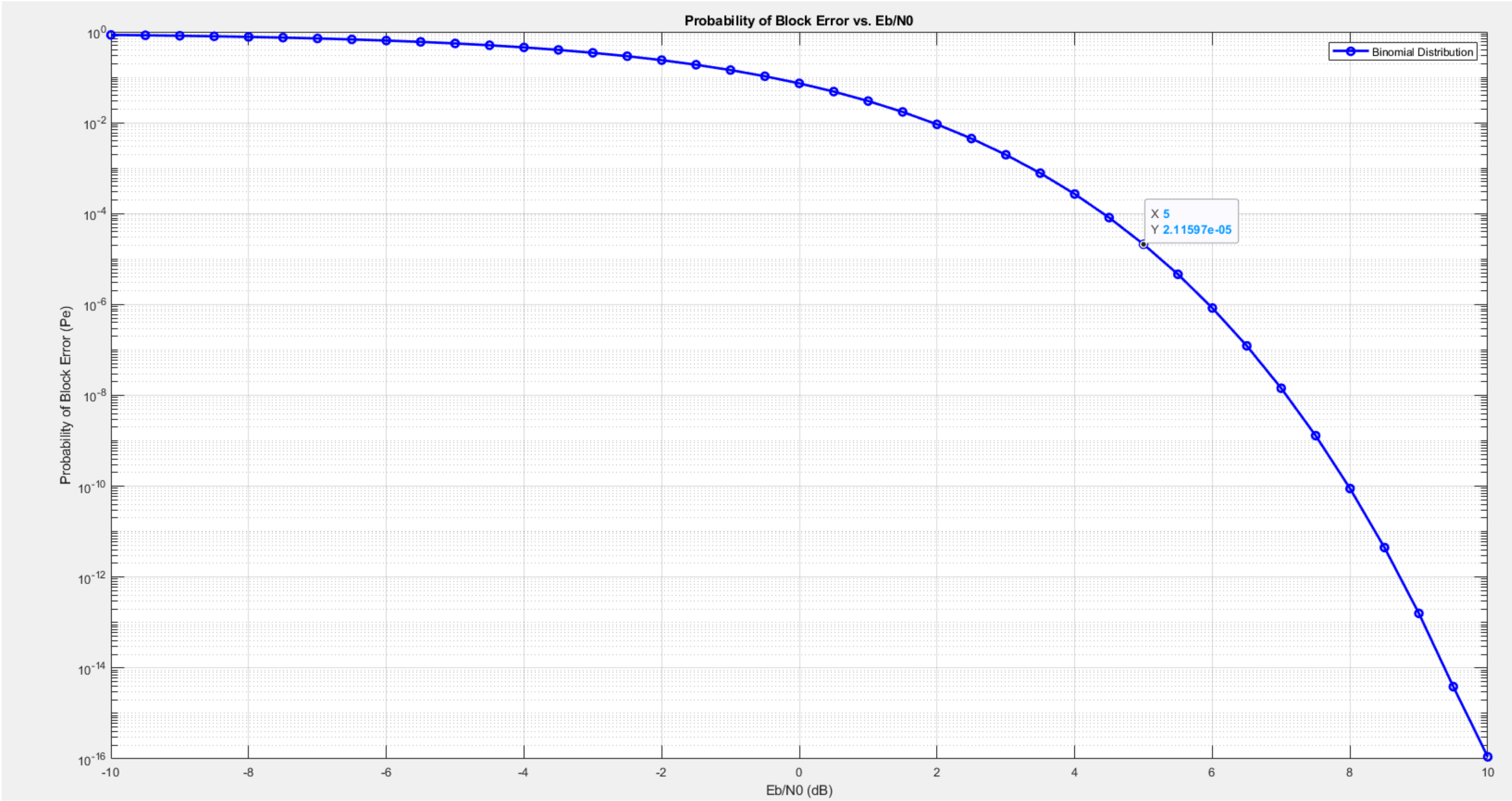
a.


Uncoded Probability of block error Pe vs. Eb/N0, k=12, n=24, dmin=8
P_e
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
-15
-10
-5
0
5
10
15
Eb/N0 (dB)
X 10.5
Y 2.6012e-05

b.

prelab5_3_b.m

```matlab
% MATLAB script for plotting Pe vs. Eb/N0 using the binomial distribution formula
% Parameters
Eb_N0_dB = -10:0.5:10; % Range of Eb/N0 values in dB
Eb_N0 = 10.^(Eb_N0_dB / 10); % Convert dB to linear scale

% Transmission parameters
n = 24; % Number of bits in a block
k = 12; % Number of information bits
R_c = k/n;

Pe = zeros(size(Eb_N0));

% Calculate the theoretical probability of a single bit erorr for BPSK
Pb_theoretical = qfunc(sqrt(2*Eb_N0*R_c));

% The probability of block error can be found using a cumulative binomial
% distribution formula. This loop calculates the probability of of block
% error at a specific SNR
for idx = 1:length(Eb_N0)
    % Calculate the pobility of a data block having 12 bits correct given the
    % probability of a single data bit eror for BPSK. This will tell us the
    % probability of a block error given that we have n number of bits in a
    % block(aka trials) and k information bits(i.e. expected possible successes).
    Pe(idx) = 1 - binocdf(floor((n-k)/2), n, Pb_theoretical(idx));
end

% Plot Pe vs. Eb/N0
figure(1);
semilogy(Eb_N0_dB, Pe, 'b-o', 'LineWidth', 2);
grid on;
title('Probability of Block Error vs. Eb/N0');
xlabel('Eb/N0 (dB)');
ylabel('Probability of Block Error (Pe)');
legend('Binomial Distribution');

```



Probability of Block Error vs. Eb/N0
Binomial Distribution
X 5
Y 2.11597e-05
Probability of Block Error (Pe)
Eb/N0 (dB)

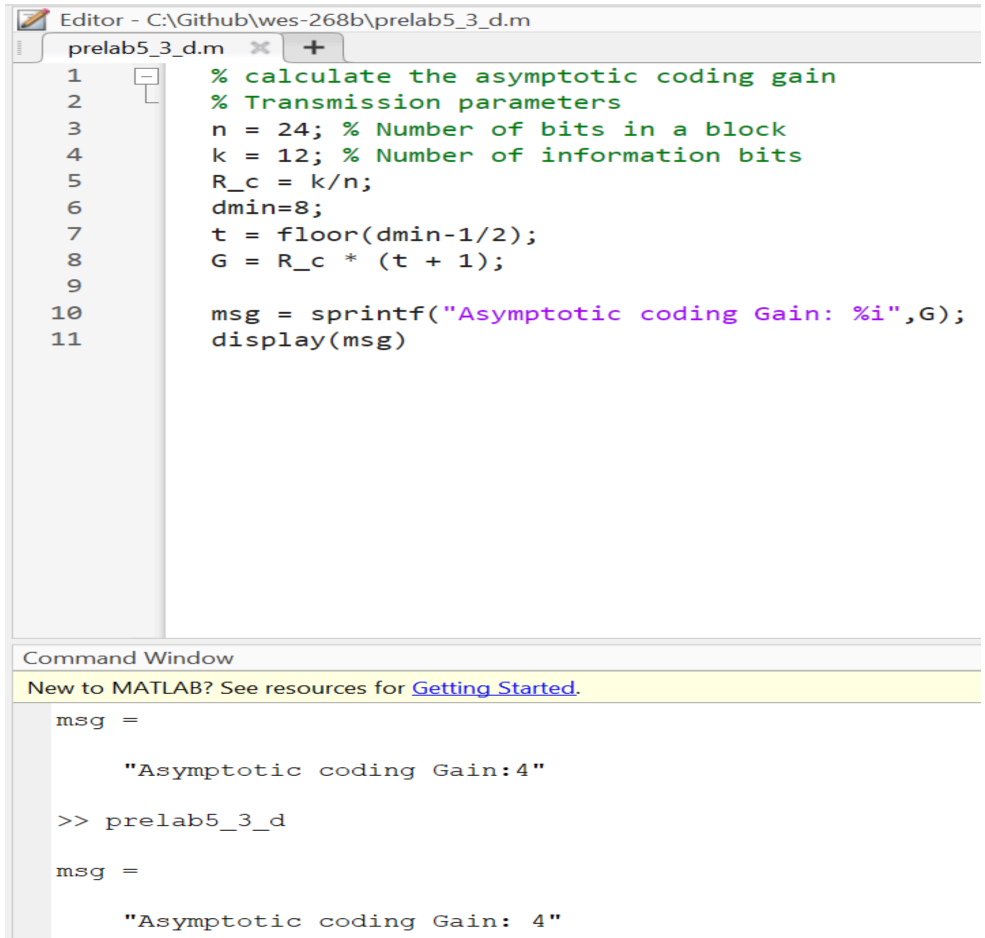
c. Coding gain

i. SDBER = Single Data Bit Error rate Eb/N0(10e-5) = 7

ii. BLKBER = Blocker Error rate Eb/N0(10e-5) = 5

iii. Coding Gain = BLKBER - SDBER = 10-5 = 5

d. Yes, it's close



Editor - C:\Github\wes-268b\prelab5_3_d.m

prelab5_3_d.m

```
% calculate the asymptotic coding gain
% Transmission parameters
n = 24; % Number of bits in a block
k = 12; % Number of information bits
R_c = k/n;
dmin=8;
t = floor(dmin-1/2);
G = R_c * (t + 1);

msg = sprintf("Asymptotic coding Gain: %i",G);
display(msg)
```

Command Window

New to MATLAB? See resources for Getting Started.

```
msg =

    "Asymptotic coding Gain:4"

>> prelab5_3_d

msg =

    "Asymptotic coding Gain: 4"
```

i.

4. Section 4

prelab5_1_4_a.m

```matlab
%% Problem 1.4.a
% MATLAB script for constructing a systematic Generator matrix G.

% What is a Generator matrix? ...

% What is a parity check matrix? ...

% Lets say we are given a parity check matrix (H).
H = [1,0,1,1,1,0,0;
     1,1,0,1,0,1,0;
     0,1,1,1,0,0,1];
% How can we obtain the generator matrix (G)?

% The generator matrix can be obtained from the parity check matrix (H) by ...

% Check if H is a valid parity matrix, number of rows can not be greater
% than or equal to number of columns
[row, col] = size(H);
if row >= col
    error('Invalid parity matrix. Number of rows should be less than the number of columns.');
end

% Calculate the systematic generator matrix G
k = col - row;  % Number of information bits
I_k = eye(k);   % Form the identity matrix for k bits

% Create a systematic generator matrix G
K_t = H(:, 1:k)'; % grab information bits from parity check matrix
G = [I_k K_t];    % combine identity matrix with information bits transposed
disp('Systematic Generator Matrix G:');
```

ommand Window

```
Systematic Generator Matrix G:
     1     0     0     0     1     1     0
     0     1     0     0     0     1     1
     0     0     1     0     1     0     1
     0     0     0     1     1     1     1
```

a.

prelab5_1_4_b.m

```
%% Problem 1.4.b
% MATLAB script for showing that the all-ones-vector c =[ 1 1 1 1 1 1 1] is a valid codeword

% Given a parity check matrix (H) how can one show that an arbitrary row vector
% "vec" is a valid codeword?
%   Multiplying the transpose of any valid codeword by the parity check
%   matrix produces a zero-value.

% Lets say we are given a parity check matrix (H).
H = [1,0,1,1,1,0,0;
     1,1,0,1,0,1,0;
     0,1,1,1,0,0,1];

% Create arbitrary row vector to determine if it's a valid codeword, i.e.
% determine if it was generated by the generator matrix (G).

c =[ 1 1 1 1 1 1 1];

% Check if the codeword is a valid codeword using the parity check matrix H
syndrome = mod(c * H', 2);  % Calculate the syndrome

% If the syndrome is all zeros, the codeword is valid
isValid = all(syndrome == 0);

if isValid
    disp('The codeword is valid.');
else
    disp('The codeword is invalid.');
end
```

Command Window

```
>> prelab5_1_4_b
The codeword is valid.
```

b.

c. Hamming(7,4) code dmin calculation

    i. dmin = n - k = (7-4) = 3.

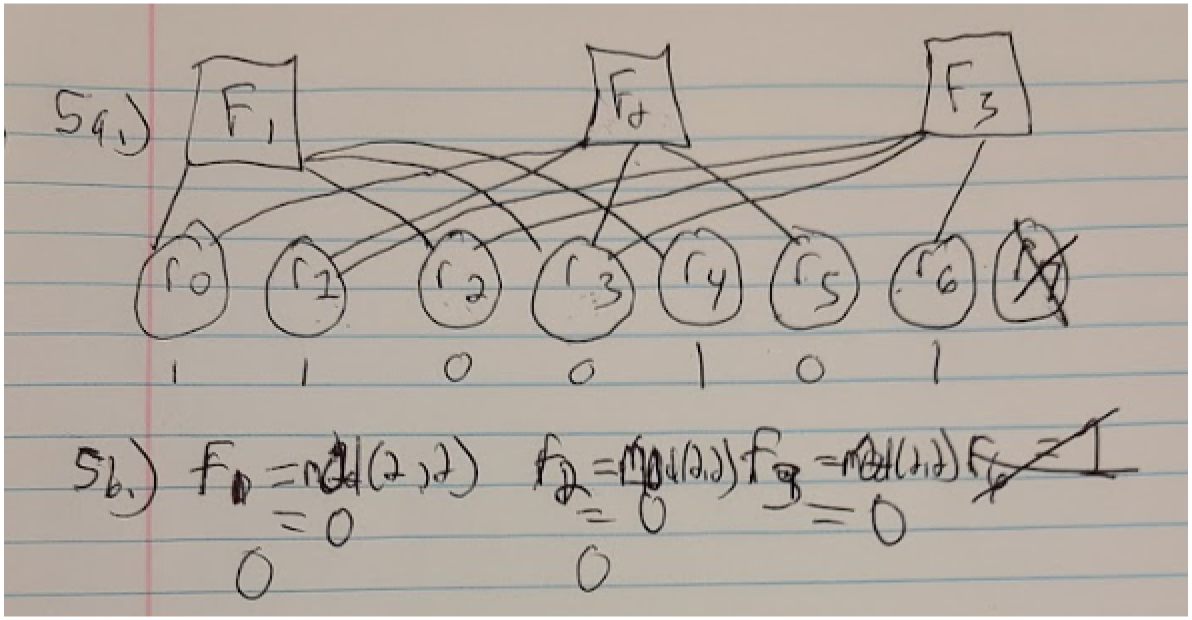
prelab5_1_4_d.m

```
% Lets say we are given a parity check matrix (H).
H = [1,0,1,1,1,0,0;
     1,1,0,1,0,1,0;
     0,1,1,1,0,0,1];

% Create arbitrary row vector to determine if it's a valid codeword, i.e.
% determine if it was generated by the generator matrix (G).
b =[ 1 0 0 0 0 0 1];

%% Problem 1.4.d.i Check if the codeword is a valid codeword using the parity check matrix (H)
% Calculate the syndrome
syndrome = mod(b * H', 2);
fprintf('syndrome vector: [')
fprintf('%d, ', syndrome(1:end-1))
fprintf('%d]\n', syndrome(end))

% If the syndrome is all zeros, the codeword is valid
isValid = all(syndrome == 0);

if isValid
    disp('The codeword is valid.');
else
    disp('The codeword is invalid.');
end
%% %% Problem 1.4.d.ii Now find the most likely transmitted codeword
% Find the position of the codeword error in the syndrome.
errorPosition = bin2dec(num2str(flip(syndrome)));
% Correct the error
b(errorPosition) = mod(b(errorPosition) + 1, 2);
decodedCodeword = b;
```

d.

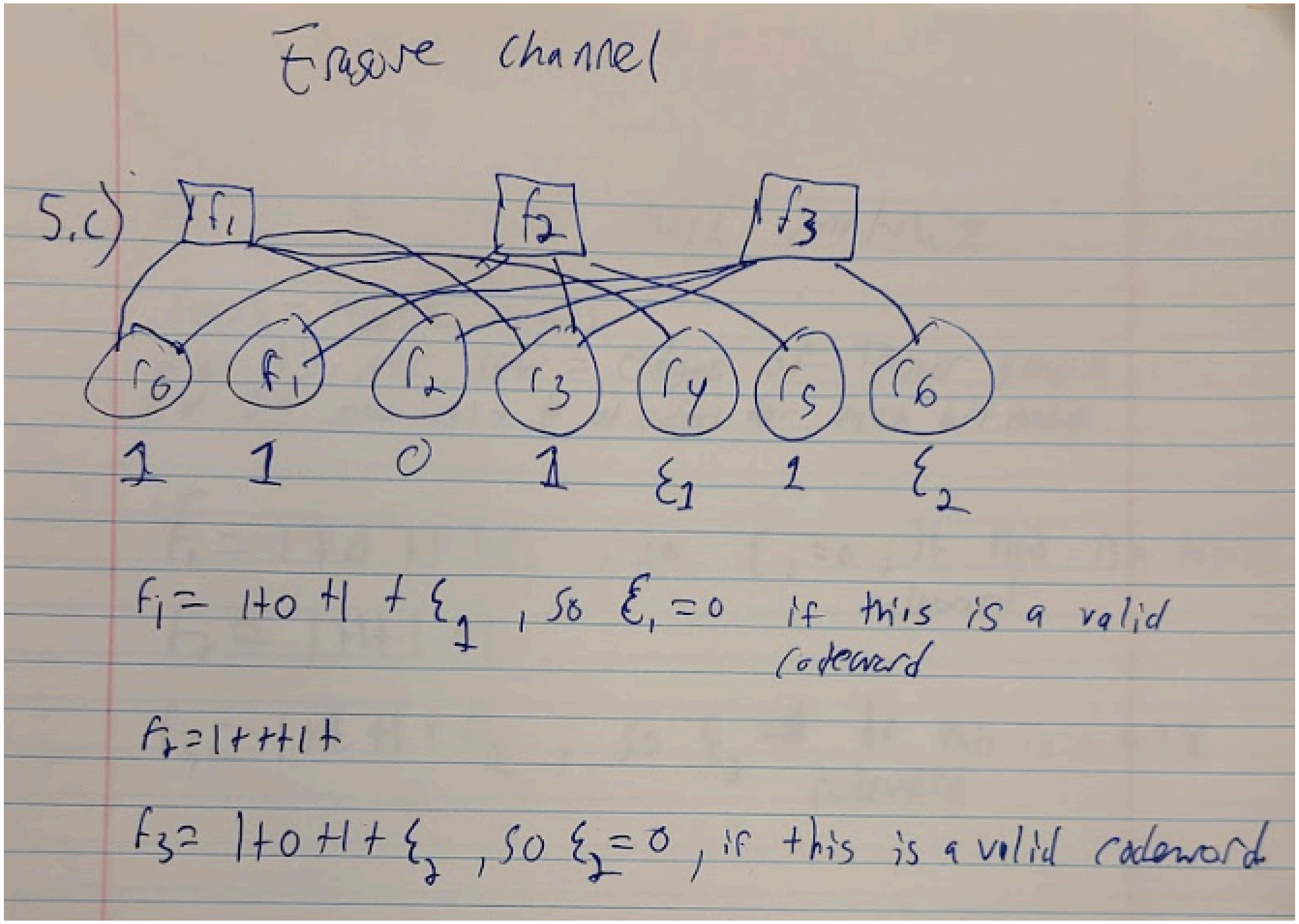
5. Parity Check Matrix and Tanner Graph

a.



b.

c.



$f_1 = 1+0+1+\varepsilon_1$, so $\varepsilon_1 = 0$ if this is a valid codeword

$f_2 = 1+ + +1+$

$f_3 = 1+0+1+\varepsilon_2$, so $\varepsilon_2 = 0$, if this is a valid codeword

Matlab Simulations

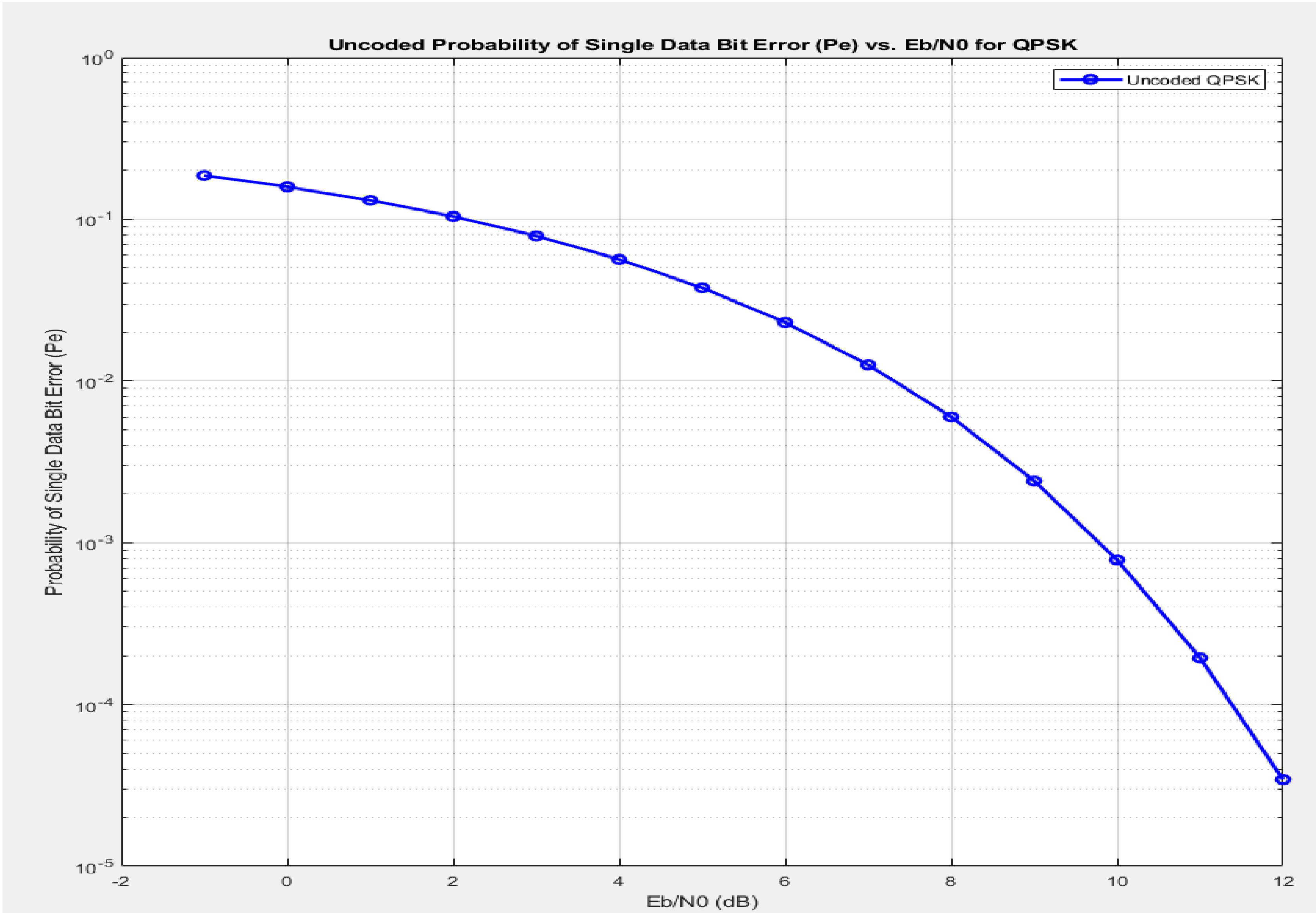
2 .1 Simulation of a scrambler

1. Simulation
   a. Plots

2.2 Simulation of QPSK with a data scrambler and a repetition code

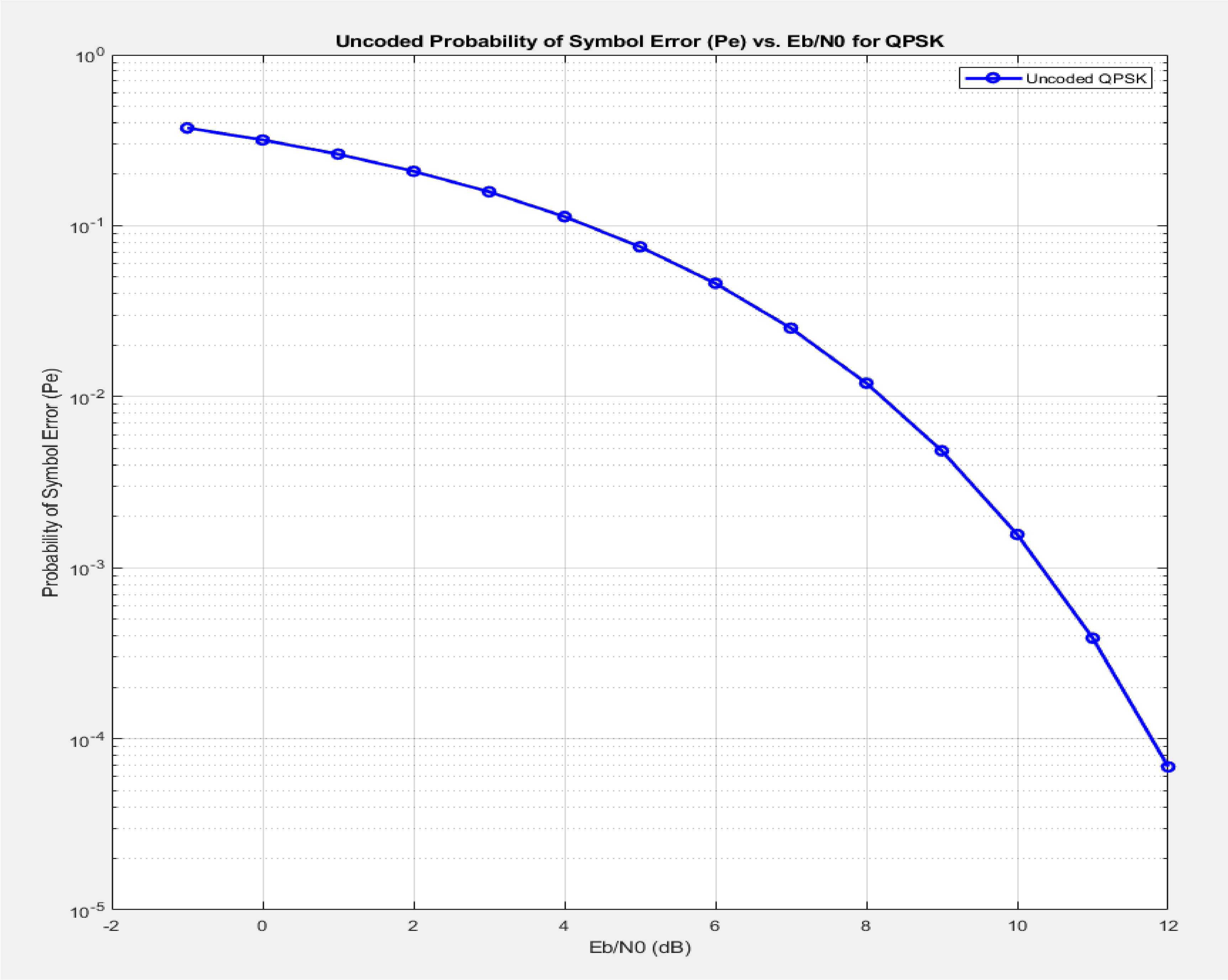
1. Uncoded probability of a bit error and the uncoded probability of a symbol error for QPSK down to an error rate of about 10^{-4}

a.



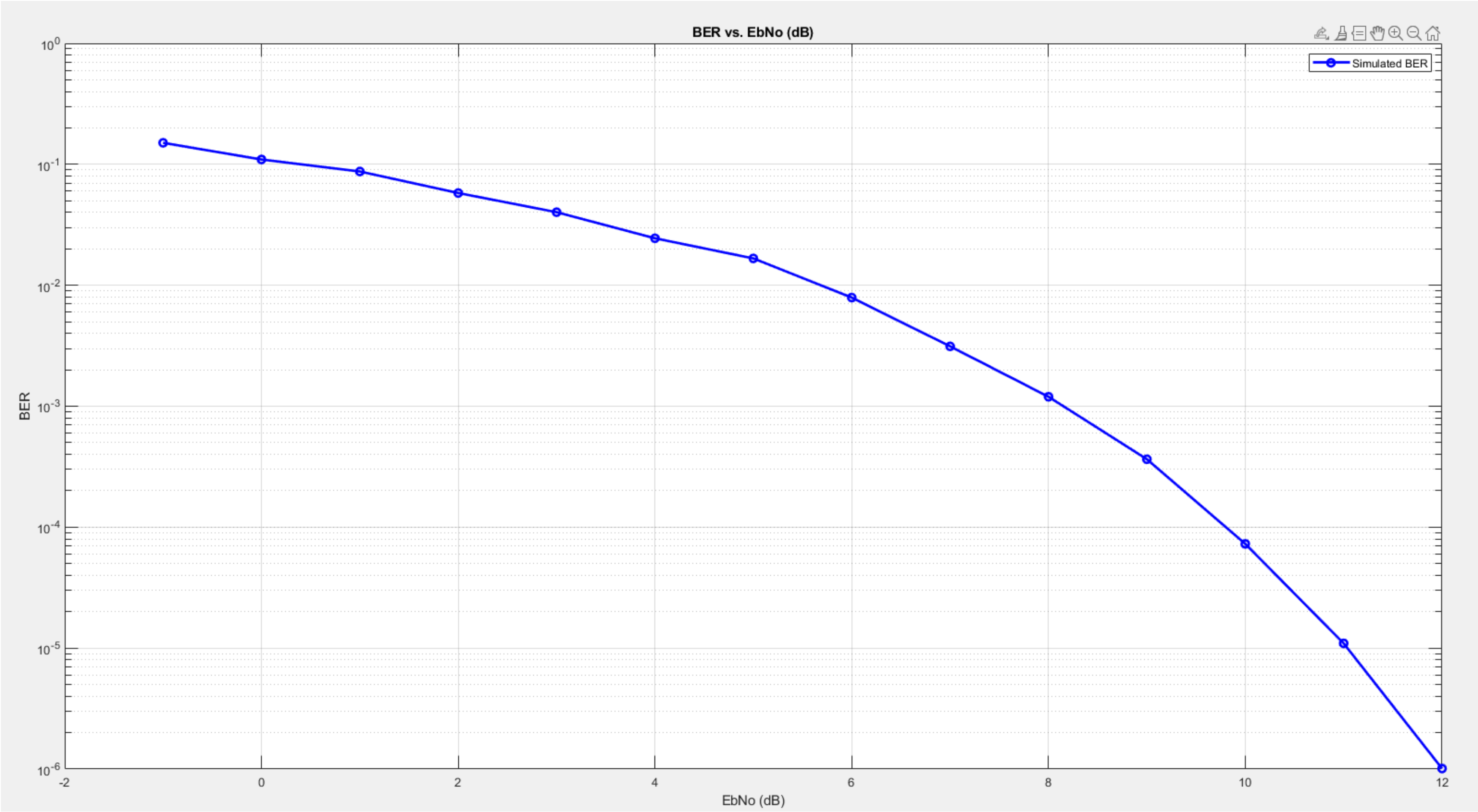
Uncoded Probability of Single Data Bit Error (Pe) vs. Eb/N0 for QPSK
Uncoded QPSK
Probability of Single Data Bit Error (Pe)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
-2
0
2
4
6
8
10
12
Eb/N0 (dB)

b.

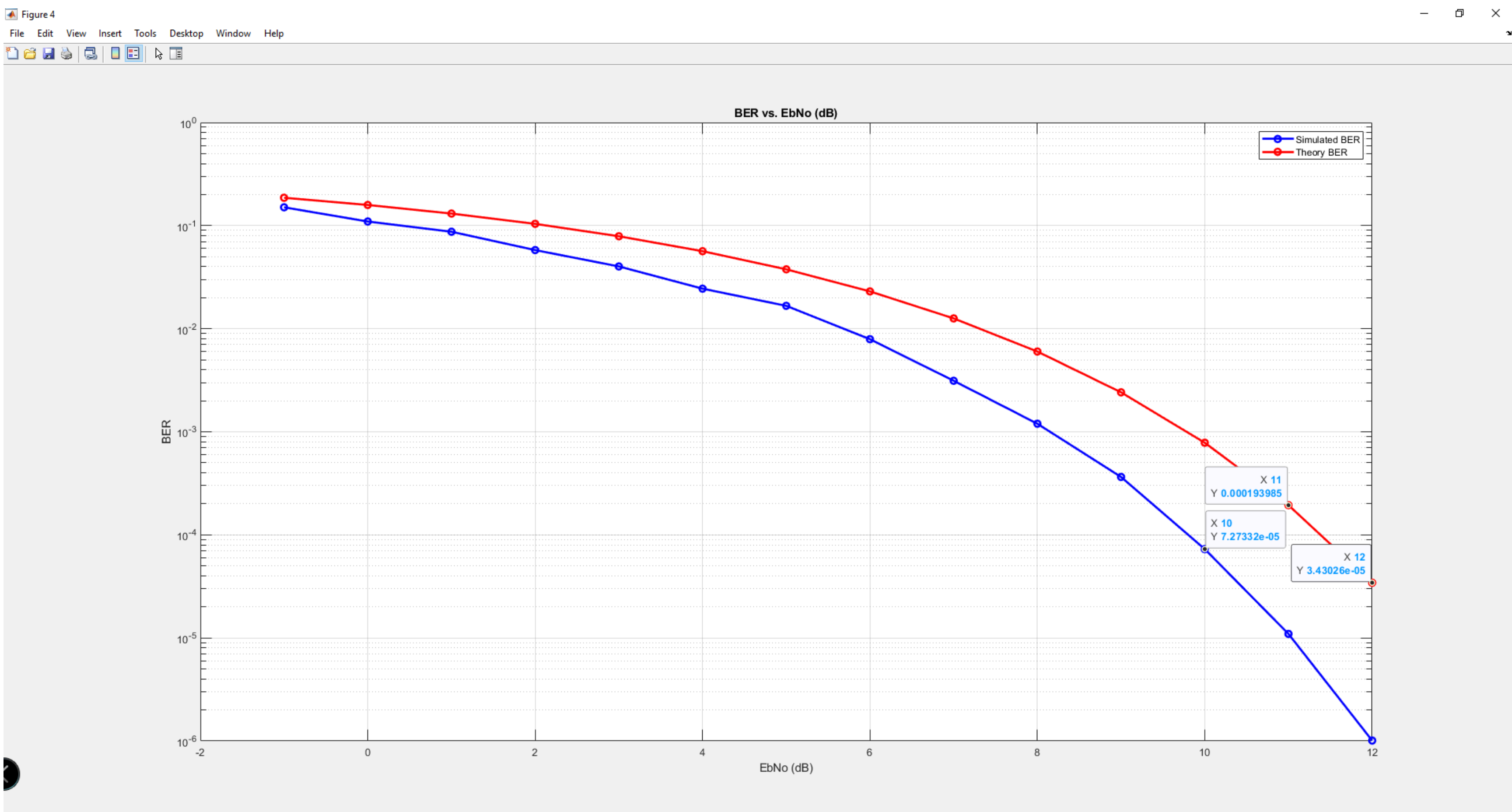


Uncoded Probability of Symbol Error (Pe) vs. Eb/N0 for QPSK
Uncoded QPSK
Probability of Symbol Error (Pe)
Eb/N0 (dB)

2. Coded probability of a bit error for one quadrature components using the (3, 1) repeat code

    a. 

3. Determine the coding gain at pe = 10−4 and compare this simulated value with theory for BPSK for a single quadrature component.
    a. Approx 2.5dB



b.

2.3 Hard and soft decision decoding

1. See prelab5_2_3_hard_dec.m
2. See prelab5_2_3_soft_dec.m
3. The term "hard decision" refers to the discrete and deterministic nature of the decoding process, where each received symbol is decisively classified as one of the possible transmitted symbols. This is in contrast to "soft decision" decoding, where the decoder considers the reliability or likelihood of each received symbol, often represented as probabilities. Soft decision decoding provides more precision at the cost of complexity.