Dhirubhai Ambani Institute of Information and Communication Technology

Gandhinagar, Gujarat

PROJECT – LDPC

Group 4 (subgroup 2)

TEACHING ASSISTANT (TA)- KHUSHI SHAH

Professor – Yash Vasavda

Course - CT216 (INTRODUCTION TO COMMUNICATION TECHNOLOGY)

- Patel Dishant (202201260)
- Yadav Alpheshkumar (202201264)
- Varnika Chhawcharia (202201269)
- Siddhant Gupta (202201272)
- Kunj Bhuva (202201275)
- Zeel Ghori (202201287)
- Patel Naisargi (202201291)
- Vivek Chaudhari (202201294)
- Pandya Ansh (202201303)
- Kushang Vivek (202201304)

We declare that:

→ The work that we are presenting is our own work.

→ We have not copied the work (the code, the results, etc.) that someone else has done.

→ Concepts, understanding and insights we will be describing are our own.

→ We make this pledge truthfully. We know that violation of this solemn pledge can carry grave consequences.

Hard Decision Decoder

Probability of success in decoding vs EbNo in dB

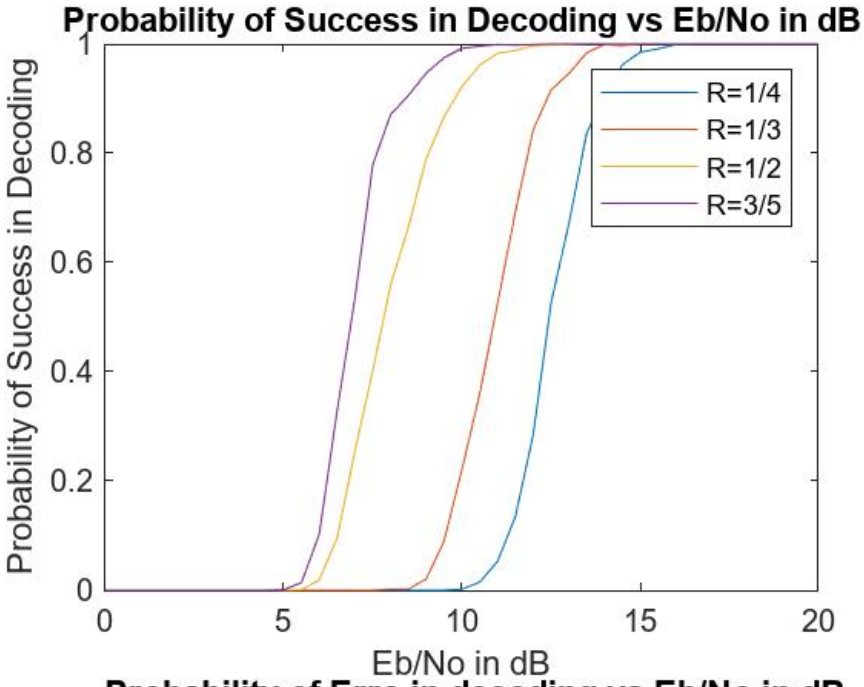
```
% load 5G NR LDPC base H matrix, use both NR_2_6_52 and NR_1_5_352
baseGraph5GNR = 'NR_2_6_52';
Number_of_rate = 1;
% varying Eb/No in Db from 0 to 20 in step size of 0.5
EbNodB = 1:0.5:20;
% Convert the base H matrix to binary H matrix
[B,Hfull,z] = nrldpc_Hmatrix(baseGraph5GNR);
% 5G NR specific details
[mb,nb] = size(B); kb = nb - mb;
% Number of information bits
kNumInfoBits = kb * z;
% varying this in the set {1/4, 1/3, 1/2, 3/5} for 2_6_52 and in the range
{1/3, 1/2, 3/5 and 4/5} for 1_5_352
code_rate = [1/4, 1/3, 1/2, 3/5];
error_detection_probability = zeros(length(code_rate),length(EbNodB));
for code_rate = [1/4, 1/3, 1/2, 3/5]

    % Some 5G NR specific details
    k_pc = kb-2; nbRM = ceil(k_pc/code_rate)+2;
    % Number of encoded bits
    nBlockLength = nbRM * z;

    % Next three lines are some 5G NR specific details
    H = Hfull(:,1:nBlockLength);
    nChecksNotPunctured = mb*z - nb*z + nBlockLength;
    % this is the binary H matrix
    H = H(1:nChecksNotPunctured,:);

    % Number of CNs (we have denoted this as U = N - K in the class)
    Nchecks = size(H,1);
    % Generate information (or message) bit vector
    b = randi([0 1],[kNumInfoBits 1]);
    % Encode using 5G NR LDPC base matrix
    c = nrldpc_encode(B,z,b');
    c = c(1:nBlockLength)';

    No_of_cols = length(H);
    No_of_rows = height(H);

    degree_CN = 0;
    % calculate maximum degree of CN from all check nodes
    for i =1:1:No_of_rows
        temp = 0;
        for j=1:1:No_of_cols
            if H(i,j) == 1
```

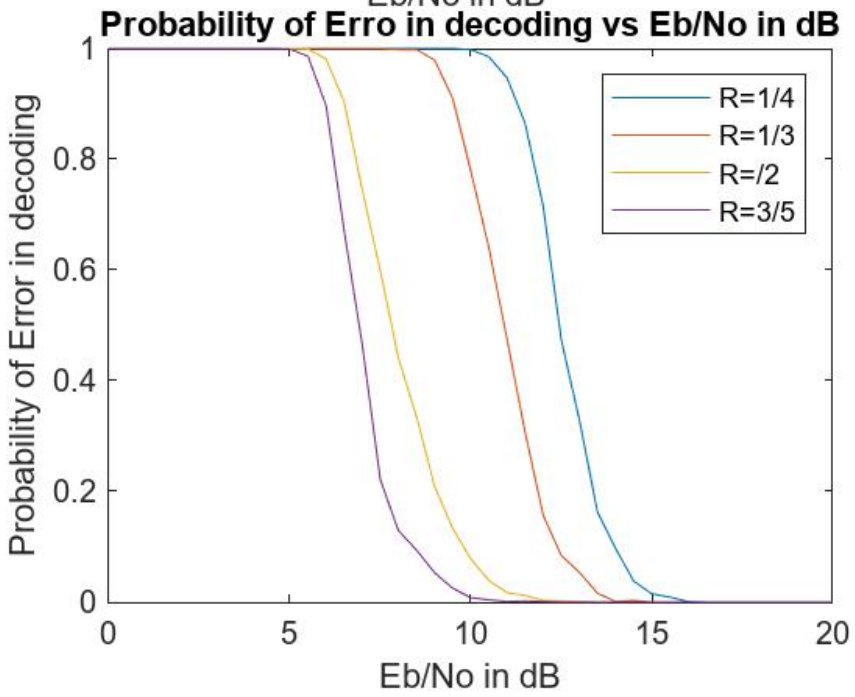
```
                temp = temp + 1;
            end
        end
        if temp > degree_CN
            degree_CN = temp;
        end
    end
    % calculate maximum degree of VN from all variable nodes
    degree_VN = 0;
    for i =1:1:No_of_cols
        temp =0;
        for j =1:1:No_of_rows
            if H(j,i) == 1
                temp = temp + 1;
            end
        end
        if temp > degree_VN
            degree_VN = temp;
        end
    end

    degree_each_VN = sum(H,1);
    connection_CN = zeros(No_of_rows, degree_CN);
    connection_VN = zeros(No_of_cols, degree_VN);

    % connection_CN(i,j) will give the position of VN where jth connection of
ith CN is connected
    connection_CN = make_connection_CN(degree_CN,H);
    % connection_VN(i,j) will give the position of CN where jth connection of
ith VN is connected
    connection_VN = make_connection_VN(degree_VN,H);
    % encoded codeword which will be transmitted through AWGN channel
    current_message=c';
    transmit_msg = current_message;
    % modulated codeword which will be transmitted through AWGN channel
    modulated_msg = 1 - 2*transmit_msg;

    Number_of_sim = 10000;

    iteration_max = 50;

    EbNo = 10.^(EbNodB./10);

    sigma = sqrt(1./(2.*code_rate.*EbNo));
    success_decoding_probability = zeros(1,length(EbNodB));


    for V = 1:1:length(EbNodB)
        success =0;
        iteration_success = zeros(Number_of_sim, iteration_max);
        iteration_probability = zeros(1,iteration_max);
```

```
        for U = 1:1:Number_of_sim
            flag =0;
            % output from AWGN channel which contains noise also with
variation sigma^2 and mean 0
            received_msg = modulated_msg + sigma(1,V)*randn(1,No_of_cols);
            % demodulation of modulated message
            demodulated_msg = received_msg<0;
            % sent_by_CN(i,j) information sent by ith CN to jth connection
            sent_by_CN = zeros(size(connection_CN));
            % received_by_CN(i,j) information received by ith CN from jth
connection
            received_by_CN = zeros(size(connection_CN));
            % sent_by_VN(i,j) information sent by ith VN to jth connection
            sent_by_VN = zeros(size(connection_VN));
            % received_by_VN(i,j) information received by ith VN from jth
connection
            received_by_VN = zeros(size(connection_VN));
            current_msg = demodulated_msg;
            % current message will be demodualted message
            for iteration = 1:1:iteration_max
              if iteration == 1
                  for o = 1:1:No_of_cols
                      sent_by_VN(o , : )= repmat(demodulated_msg(o),
1,degree_VN);
                  end
              else
                      sent_by_VN = repetition_code_1(received_by_VN,
current_msg, connection_VN, degree_each_VN);
              end
              received_by_CN = received_at_CN(sent_by_VN, connection_VN,
connection_CN);
              sent_by_CN = SPC(received_by_CN, connection_CN);
              received_by_VN = received_at_VN(sent_by_CN, connection_CN,
connection_VN );
              current_msg = repetition_code_2(received_by_VN, current_msg,
degree_each_VN);
              % comapring the transmitted message and the decoded message
after every iteration
              comparing = mod(current_msg + transmit_msg, 2);
              % breaking the decoding process if transmitted message and the
decoded message matches
              if sum(comparing,2)==0
                  flag = 1;
                  iteration_success(U,iteration:iteration_max) =
iteration_success(U,iteration:iteration_max)  +1;
                  break;
              end
            end
              % counting the times the code has successfully decoded the
message
                success = success + flag;
```

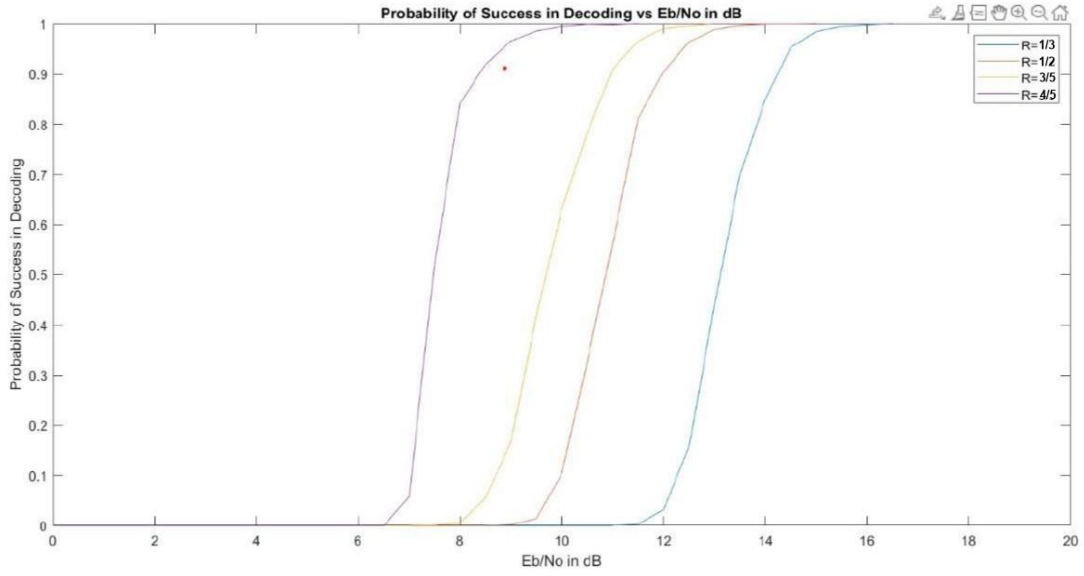
```
        end
         success_decoding_probability(1,V)= success/Number_of_sim;
         error_detection_probability(Number_of_rate, V) = 1 -
success_decoding_probability(1,V);



    end
    Number_of_rate = Number_of_rate + 1;
    % PLotting the graph of successful decoding probability with varying Eb/No
in dB for various code rates
    plot(EbNodB, success_decoding_probability);
    hold on;


end
xlabel('Eb/No in dB');
ylabel('Probability of Success in Decoding');
title('Probability of Success in Decoding vs Eb/No in dB');
legend('R=1/4' ,'R=1/3' ,'R=1/2', 'R=3/5' );
hold off;


figure ;
for i=1:1:4
    error_in_decoding = error_detection_probability(i,:) ;
     % PLotting the graph of  probability of error occured in decoding with
varying Eb/No in dB for various code rates
    plot(EbNodB, error_in_decoding);
hold on;
end
xlabel('Eb/No in dB');
ylabel('Probability of Error in decoding');
title('Probability of Error in decoding vs Eb/No in dB');
legend('R=1/4' ,'R=1/3' ,'R=1/2', 'R=3/5');
```

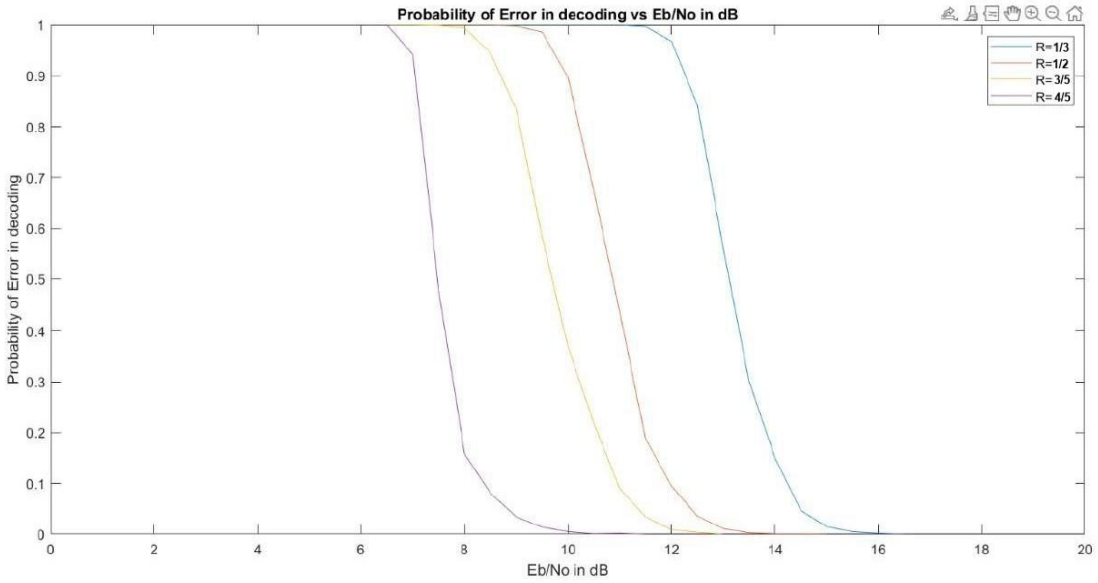
GRAPH OF HARD_DECISION DECODER FOR NR-2-6-52 BASE MATRIX



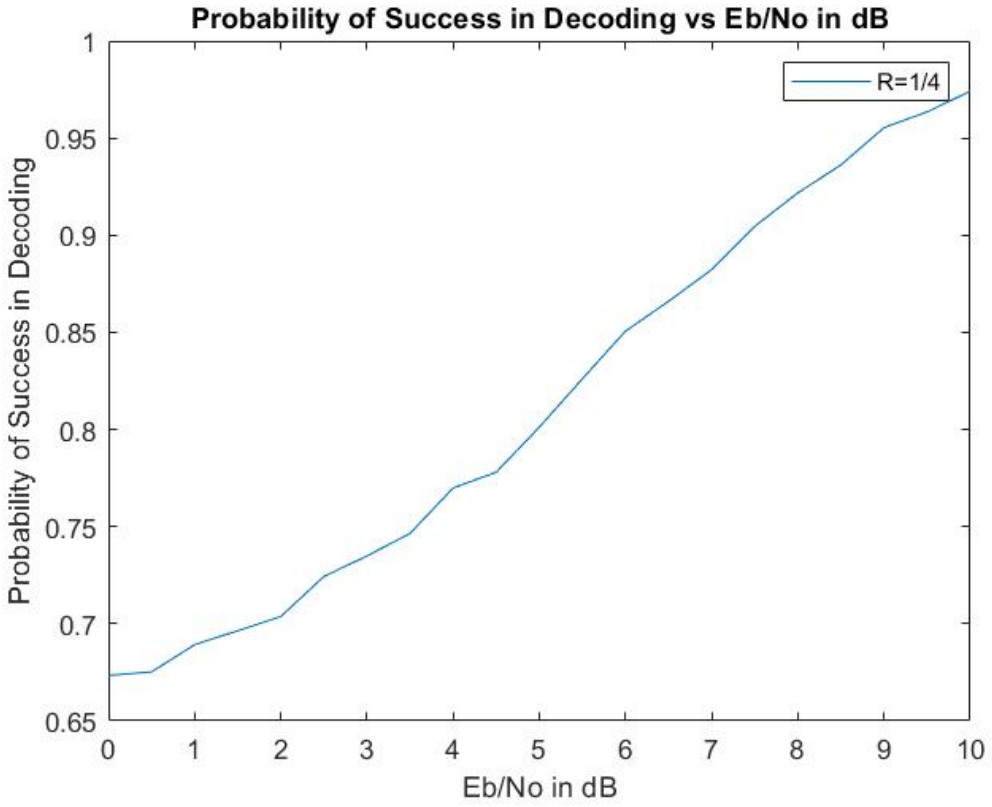


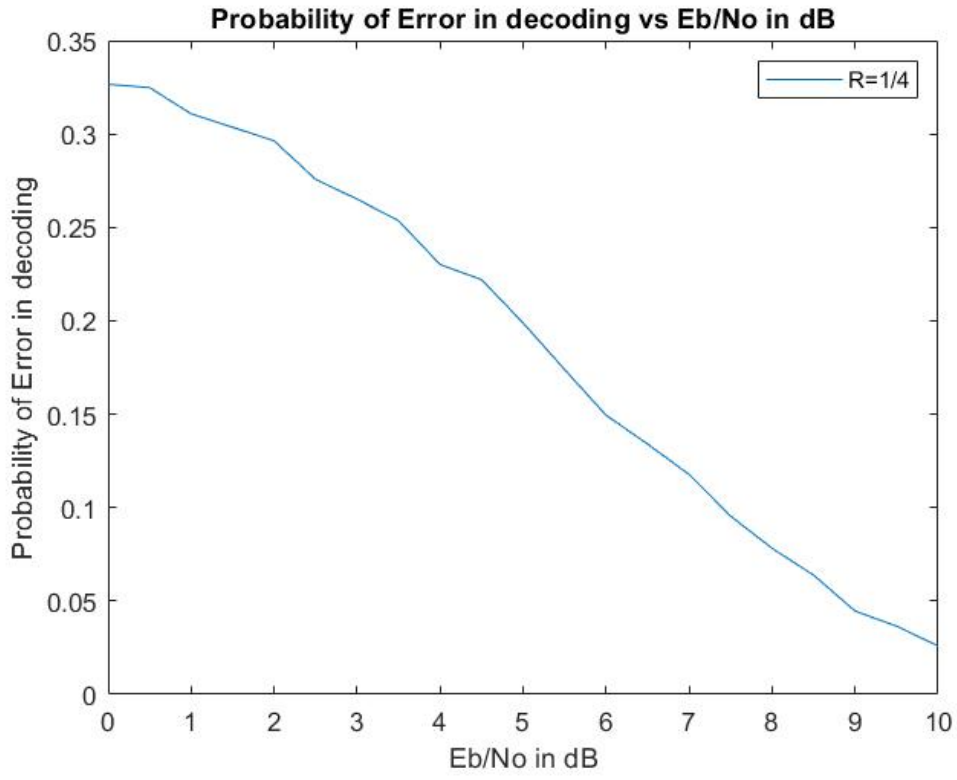
GRAPH OF HARD_DECISION FOR NR_1_5_352 BASE MATRIX





GRAPH OF HARD_DECISION FOR 9x12 MATRIX- Rate = 3/12





Hard Decision Decoder Bit Error Probability

BEP in decoding VS EbNodB

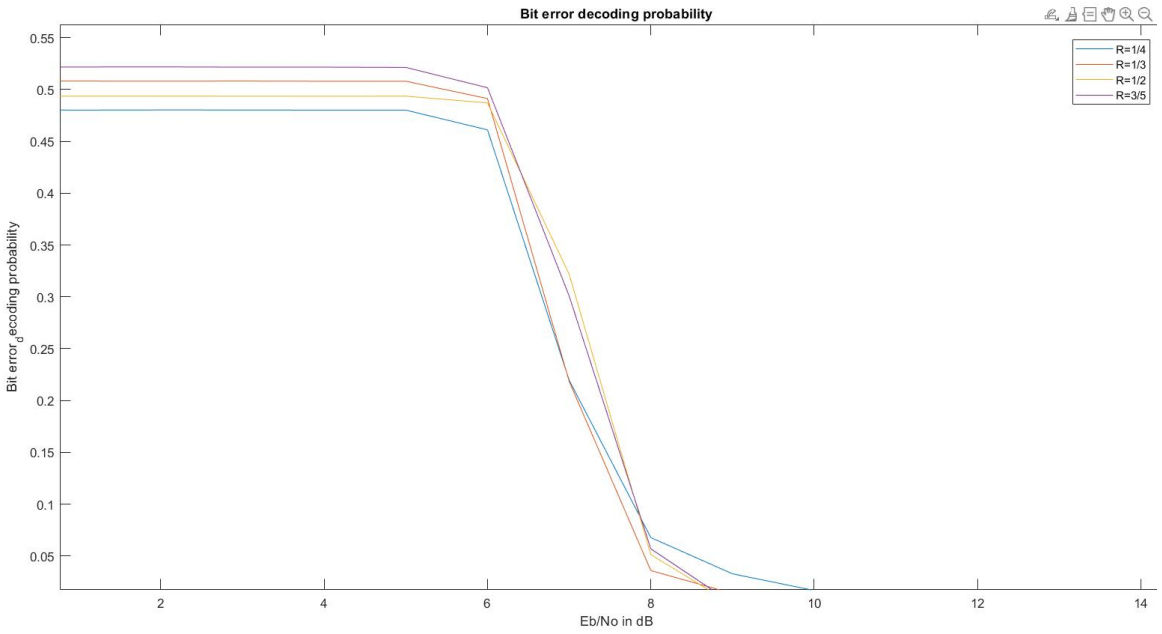
```
% load 5G NR LDPC base H matrix, use both NR_2_6_52 and NR_1_5_352
baseGraph5GNR = 'NR_2_6_52';
Number_of_rate = 1;
% varying Eb/No in Db from 0 to 20 in step size of 0.5
EbNodB = 1:1:20;
% Convert the base H matrix to binary H matrix
[B,Hfull,z] = nrldpc_Hmatrix(baseGraph5GNR);
% 5G NR specific details
[mb,nb] = size(B); kb = nb - mb;
% Number of information bits
kNumInfoBits = kb * z;
% varying this in the set {1/4, 1/3, 1/2, 3/5} for 2_6_52 and in the range
{1/3, 1/2, 3/5 and 4/5} for 1_5_352
code_rate = [1/4, 1/3, 1/2, 3/5];
error_detection_probability = zeros(length(code_rate),length(EbNodB));
for code_rate = [1/4, 1/3, 1/2, 3/5]

    % Some 5G NR specific details
    k_pc = kb-2; nbRM = ceil(k_pc/code_rate)+2;
    % Number of encoded bits
    nBlockLength = nbRM * z;

    % Next three lines are some 5G NR specific details
    H = Hfull(:,1:nBlockLength);
    nChecksNotPunctured = mb*z - nb*z + nBlockLength;
    % this is the binary H matrix
    H = H(1:nChecksNotPunctured,:);

    % Number of CNs (we have denoted this as U = N - K in the class)
    Nchecks = size(H,1);
    % Generate information (or message) bit vector
    b = randi([0 1],[kNumInfoBits 1]);
    % Encode using 5G NR LDPC base matrix
    c = nrldpc_encode(B,z,b');
    c = c(1:nBlockLength)';

    No_of_cols = length(H);
    No_of_rows = height(H);

    degree_CN = 0;
    % calculate maximum degree of CN from all check nodes
    for i =1:1:No_of_rows
        temp = 0;
        for j=1:1:No_of_cols
```

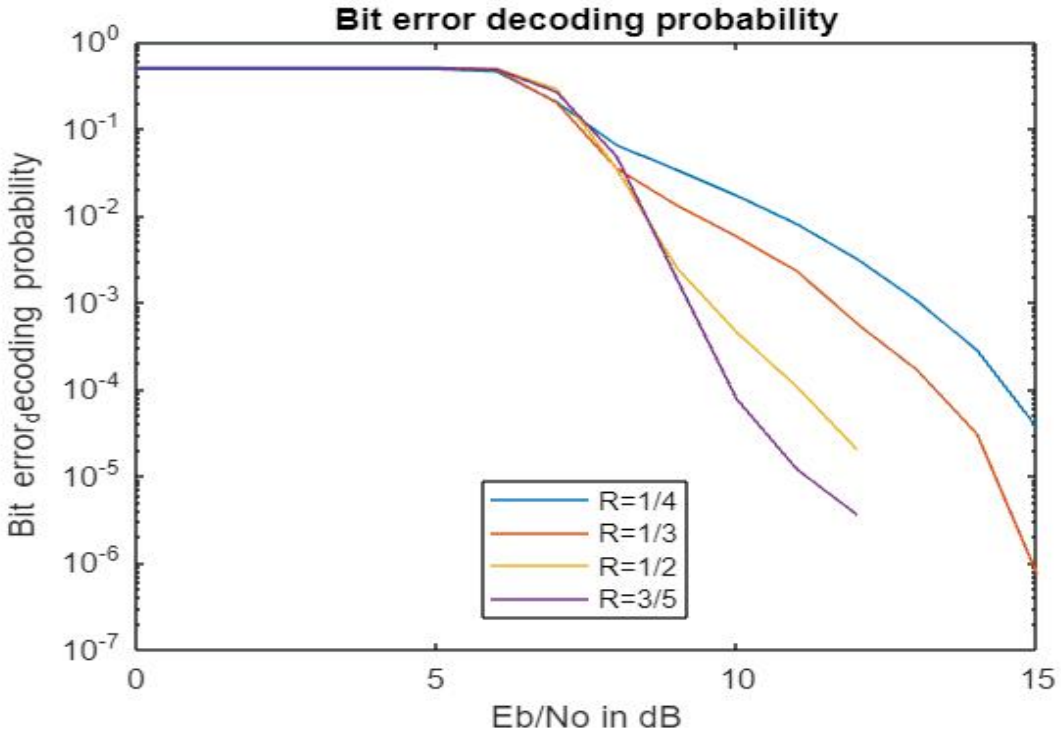
```
            if H(i,j) == 1
                temp = temp + 1;
            end
        end
        if temp > degree_CN
            degree_CN = temp;
        end
    end
    % calculate maximum degree of VN from all variable nodes
    degree_VN = 0;
    for i =1:1:No_of_cols
        temp =0;
        for j =1:1:No_of_rows
            if H(j,i) == 1
                temp = temp + 1;
            end
        end
        if temp > degree_VN
            degree_VN = temp;
        end
    end

    degree_each_VN = sum(H,1);
    connection_CN = zeros(No_of_rows, degree_CN);
    connection_VN = zeros(No_of_cols, degree_VN);

    % connection_CN(i,j) will give the position of VN where jth connection of
ith CN is connected
    connection_CN = make_connection_CN(degree_CN,H);
    % connection_VN(i,j) will give the position of CN where jth connection of
ith VN is connected
    connection_VN = make_connection_VN(degree_VN,H);
    % encoded codeword which will be transmitted through AWGN channel
    current_message=c';
    transmit_msg = current_message;
    % modulated codeword which will be transmitted through AWGN channel
    modulated_msg = 1 - 2*transmit_msg;

    Number_of_sim = 10000;

    iteration_max = 50;

    EbNo = 10.^(EbNodB./10);

    sigma = sqrt(1./(2.*code_rate.*EbNo));
    %success_decoding_probability = zeros(1,length(EbNodB));

    bit_error_decoding_probability = zeros(1,length(EbNodB));

    for V = 1:1:length(EbNodB)
        success = 0;
```

```
        error = 0;
        for U = 1:1:Number_of_sim
            flag =0;
            % output from AWGN channel which contains noise also with
variation sigma^2 and mean 0
            received_msg = modulated_msg + sigma(1,V)*randn(1,No_of_cols);
            % demodulation of modulated message
            demodulated_msg = received_msg<0;
            % sent_by_CN(i,j) information sent by ith CN to jth connection
            sent_by_CN = zeros(size(connection_CN));
            % received_by_CN(i,j) information received by ith CN from jth
connection
            received_by_CN = zeros(size(connection_CN));
            % sent_by_VN(i,j) information sent by ith VN to jth connection
            sent_by_VN = zeros(size(connection_VN));
            % received_by_VN(i,j) information received by ith VN from jth
connection
            received_by_VN = zeros(size(connection_VN));
            current_msg = demodulated_msg;
            % current message will be demodualted message
            for iteration = 1:1:iteration_max
              if iteration == 1
                  for o = 1:1:No_of_cols
                      sent_by_VN(o , : )= repmat(demodulated_msg(o),
1,degree_VN);
                  end
              else
                      sent_by_VN = repetition_code_1(received_by_VN,
current_msg, connection_VN, degree_each_VN);
              end
              received_by_CN = received_at_CN(sent_by_VN, connection_VN,
connection_CN);
              sent_by_CN = SPC(received_by_CN, connection_CN);
              received_by_VN = received_at_VN(sent_by_CN, connection_CN,
connection_VN );
              current_msg = repetition_code_2(received_by_VN, current_msg,
degree_each_VN);
              % comapring the transmitted message and the decoded message
after every iteration
              comparing = mod(current_msg + transmit_msg, 2);
              % breaking the decoding process if transmitted message and the
decoded message matches
              if sum(comparing,2)==0
                  flag = 1;
                  break;
              end
            end
              % counting the times the code has successfully decoded the
message
                success = success + flag;
```

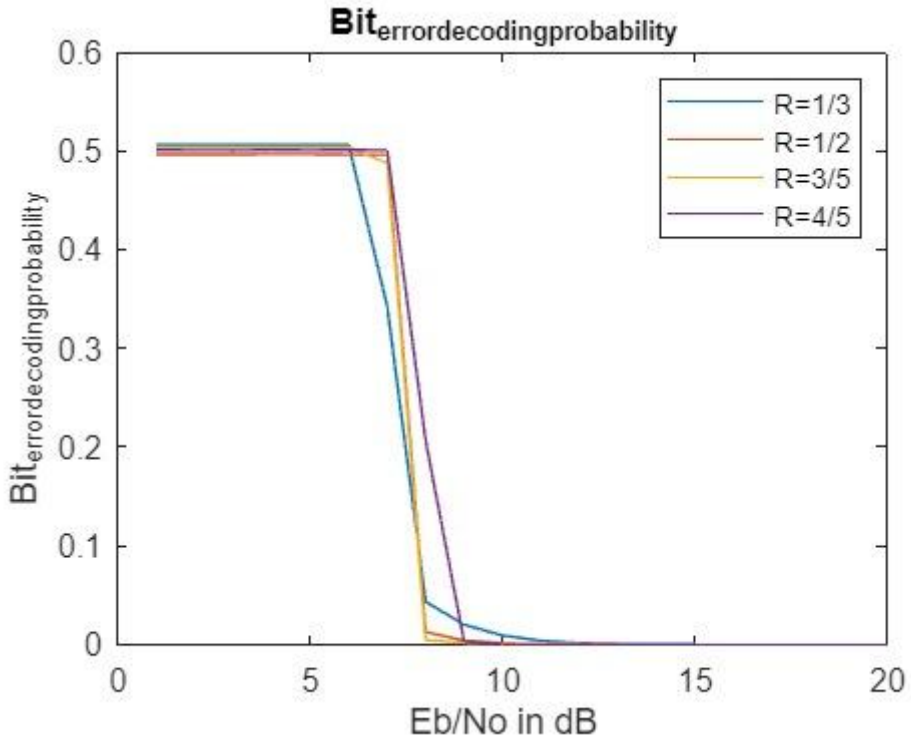
```
                error = error + sum(comparing, 2);
        end
         %success_decoding_probability(1,V)= success/Number_of_sim;
         %error_detection_probability(Number_of_rate, V) = 1 -
success_decoding_probability(1,V);
         bit_error_decoding_probability(1,V) =
error/(length(transmit_msg)*Number_of_sim);
      end
     Number_of_rate = Number_of_rate + 1;
    % PLotting the graph of bit_error_decoding_probability with varying Eb/No
in dB for various code rates
    plot(EbNodB, bit_error_decoding_probability);
    hold on;
end
xlabel('Eb/No in dB');
ylabel('Bit_error_decoding_probability');
title('Bit_error_decoding_probability');
legend('R=1/4' ,'R=1/3' ,'R=1/2', 'R=3/5');
hold off;
```

Graph For Bit Error Decoding Probability NR-2-6-52

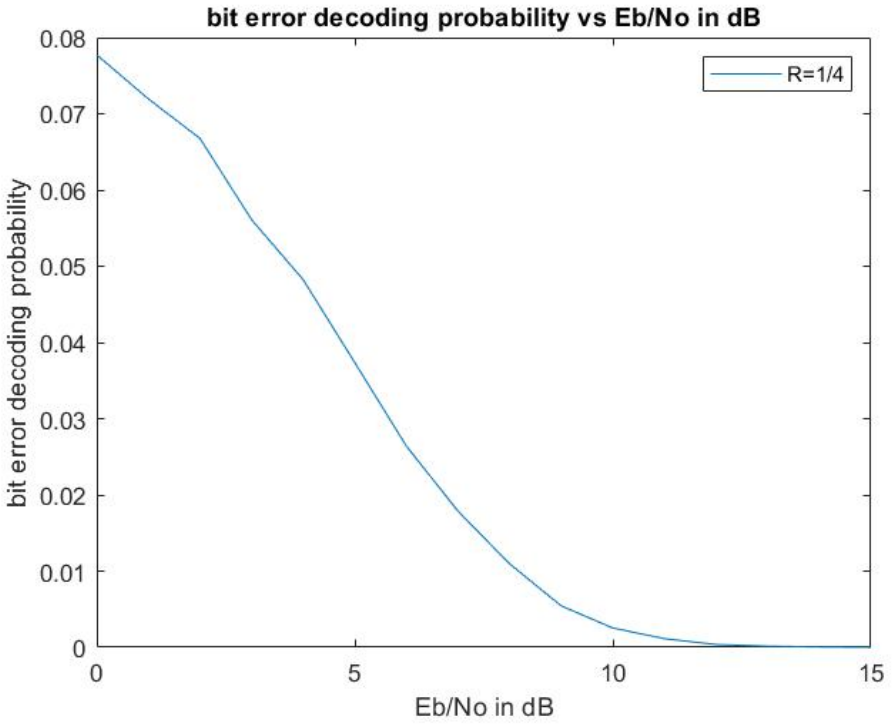


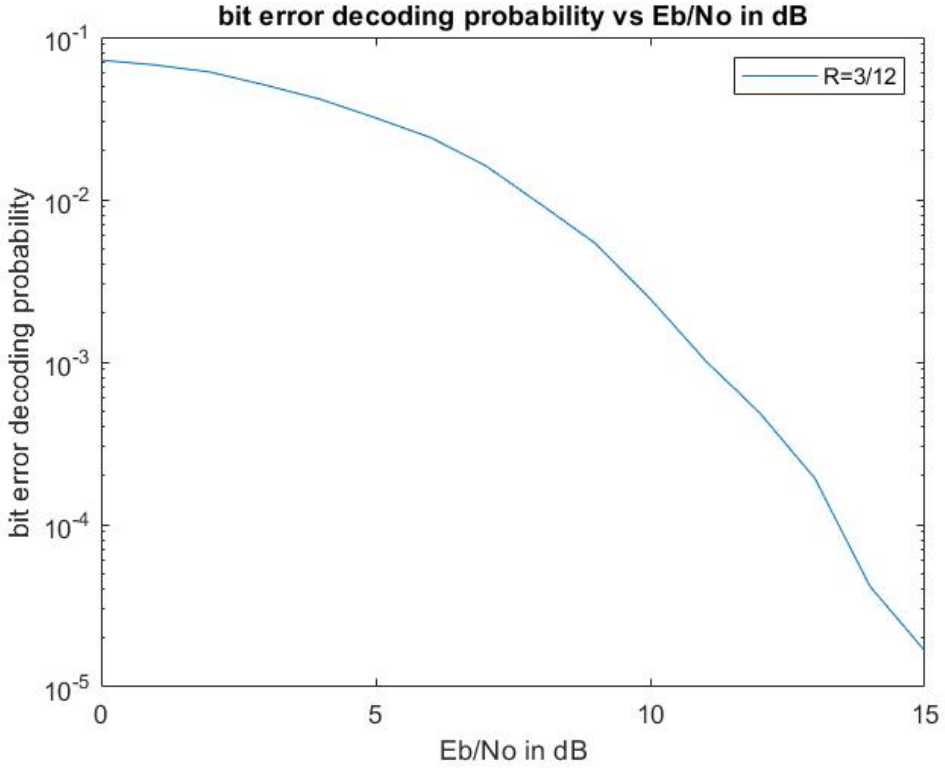


Graph For Bit Error Decoding Probability NR-1-5-352



Graph For Bit Error Decoding Probability for 9X12 H matrix





Hard Decision Decoder Iteration

Probability of successful decoding at the ith iteration for various values of EbNo in dB

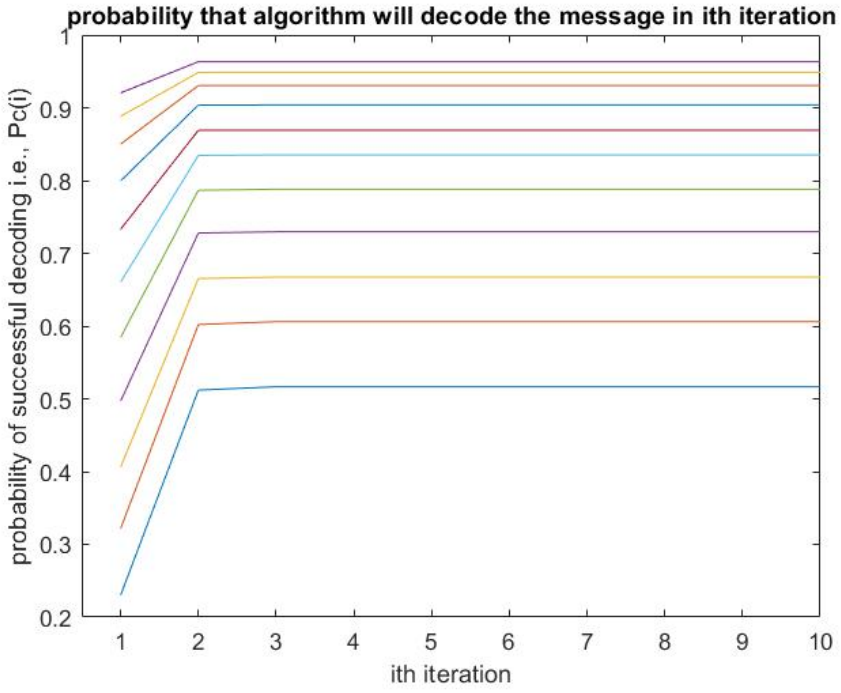
```
    % load 5G NR LDPC base H matrix, use both NR_2_6_52 and NR_1_5_352
    baseGraph5GNR = 'NR_2_6_52';
    Number_of_rate = 1;
    % varying Eb/No in Db from 13 to 15 in step size of 0.2
    EbNodB = 11:0.5:15;
    % varying this in the set {1/4, 1/3, 1/2, 3/5} for 2_6_52 and in the range
{1/3, 1/2, 3/5 and 4/5} for 1_5_352
    code_rate = [1/4];
    error_detection_probability = zeros(length(code_rate),length(EbNodB));
    % Convert the base H matrix to binary H matrix
    [B,Hfull,z] = nrldpc_Hmatrix(baseGraph5GNR);
    % 5G NR specific details
    [mb,nb] = size(B); kb = nb - mb;
    % Number of information bits
    kNumInfoBits = kb * z;
    % Some 5G NR specific details
    k_pc = kb-2; nbRM = ceil(k_pc/code_rate)+2;
    % Number of encoded bits
    nBlockLength = nbRM * z;

    % Next three lines are some 5G NR specific details
    H = Hfull(:,1:nBlockLength);
    nChecksNotPunctured = mb*z - nb*z + nBlockLength;
    % this is the binary H matrix
    H = H(1:nChecksNotPunctured,:);

    % Number of CNs (we have denoted this as U = N - K in the class)
    Nchecks = size(H,1);
    % Generate information (or message) bit vector
    b = randi([0 1],[kNumInfoBits 1]);
    % Encode using 5G NR LDPC base matrix
    c = nrldpc_encode(B,z,b');
    c = c(1:nBlockLength)';

    No_of_cols = length(H);
    No_of_rows = height(H);

    degree_CN = 0;
    % calculate maximum degree of CN from all check nodes
    for i =1:1:No_of_rows
        temp = 0;
        for j=1:1:No_of_cols
            if H(i,j) == 1
```

```
                temp = temp + 1;
            end
        end
        if temp > degree_CN
            degree_CN = temp;
        end
    end
    % calculate maximum degree of VN from all variable nodes
    degree_VN = 0;
    for i =1:1:No_of_cols
        temp =0;
        for j =1:1:No_of_rows
            if H(j,i) == 1
                temp = temp + 1;
            end
        end
        if temp > degree_VN
            degree_VN = temp;
        end
    end

    degree_each_VN = sum(H,1);
    connection_CN = zeros(No_of_rows, degree_CN);
    connection_VN = zeros(No_of_cols, degree_VN);

    % connection_CN(i,j) will give the position of VN where jth connection of
ith CN is connected
    connection_CN = make_connection_CN(degree_CN,H);
    % connection_VN(i,j) will give the position of CN where jth connection of
ith VN is connected
    connection_VN = make_connection_VN(degree_VN,H);
    % encoded codeword which will be transmitted through AWGN channel
    current_message=c';
    transmit_msg = current_message;
    % modulated codeword which will be transmitted through AWGN channel
    modulated_msg = 1 - 2*transmit_msg;

    Number_of_sim = 10000;

    iteration_max = 50;
    total_iteration = 1:1:iteration_max;
    EbNo = 10.^(EbNodB./10);

    sigma = sqrt(1./(2.*code_rate.*EbNo));
    % success_decoding_probability = zeros(1,length(EbNodB));


    for V = 1:1:length(EbNodB)
        success =0;
        iteration_success = zeros(Number_of_sim, iteration_max);
        iteration_probability = zeros(1,iteration_max);
```

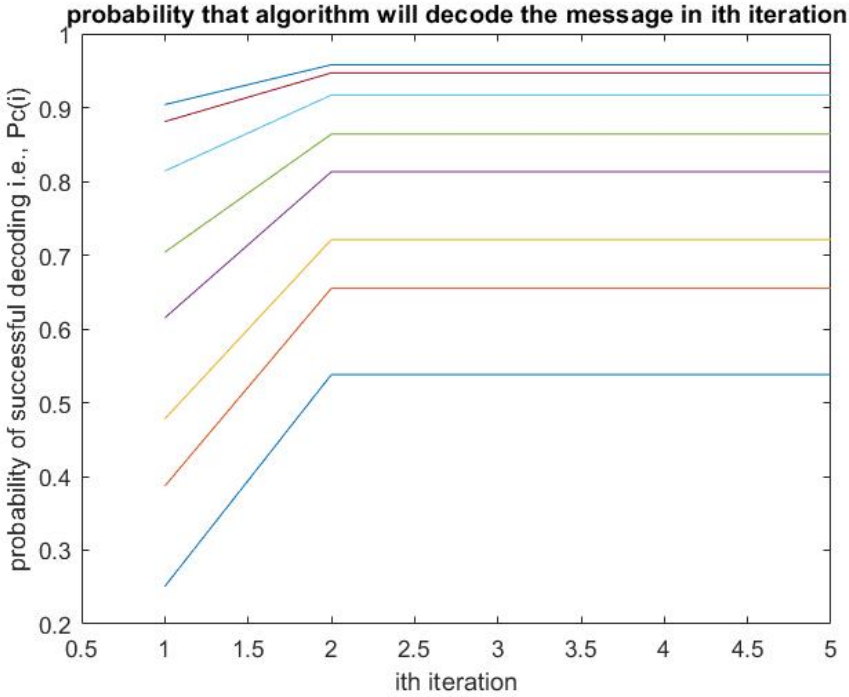
```
        for U = 1:1:Number_of_sim
            flag =0;
            % output from AWGN channel which contains noise also with
variation sigma^2 and mean 0
            received_msg = modulated_msg + sigma(1,V)*randn(1,No_of_cols);
            % demodulation of modulated message
            demodulated_msg = received_msg<0;
            % sent_by_CN(i,j) information sent by ith CN to jth connection
            sent_by_CN = zeros(size(connection_CN));
            % received_by_CN(i,j) information received by ith CN from jth
connection
            received_by_CN = zeros(size(connection_CN));
            % sent_by_VN(i,j) information sent by ith VN to jth connection
            sent_by_VN = zeros(size(connection_VN));
            % received_by_VN(i,j) information received by ith VN from jth
connection
            received_by_VN = zeros(size(connection_VN));
            current_msg = demodulated_msg;
            % current message will be demodualted message
            for iteration = 1:1:iteration_max
              if iteration == 1
                  for o = 1:1:No_of_cols
                      sent_by_VN(o , : )= repmat(demodulated_msg(o),
1,degree_VN);
                  end
              else
                      sent_by_VN = repetition_code_1(received_by_VN,
demodulated_msg, connection_VN, degree_each_VN);
              end
              received_by_CN = received_at_CN(sent_by_VN, connection_VN,
connection_CN);
              sent_by_CN = SPC(received_by_CN, connection_CN);
              received_by_VN = received_at_VN(sent_by_CN, connection_CN,
connection_VN );
              current_msg = repetition_code_2(received_by_VN, demodulated_msg,
degree_each_VN);
              % comapring the transmitted message and the decoded message
after every iteration
              comparing = mod(current_msg + transmit_msg, 2);
              % breaking the decoding process if transmitted message and the
decoded message matches
              if sum(comparing,2)==0
                  flag = 1;
                  iteration_success(U,iteration:iteration_max) =
iteration_success(U,iteration:iteration_max)  +1;
                  break;
              end
            end
              % counting the times the code has successfully decoded the
message
                success = success + flag;
```

```
        end
        % success_decoding_probability(1, V) = success/Number_of_sim;
        % error_detection_probability(Number_of_rate, V) = 1 -
success_decoding_probability(1, V);
        summation =sum(iteration_success,1);
        iteration_probability= summation./Number_of_sim;
        plot(total_iteration, iteration_probability);
        hold on;

    end
    Number_of_rate = Number_of_rate + 1;
    xlabel('ith iteration');
    ylabel('probability of successful decoding i.e., Pc(i)');
    title("probability that algorithm will decode the message in ith
iteration");
    xlim([0.5, 5]);
```

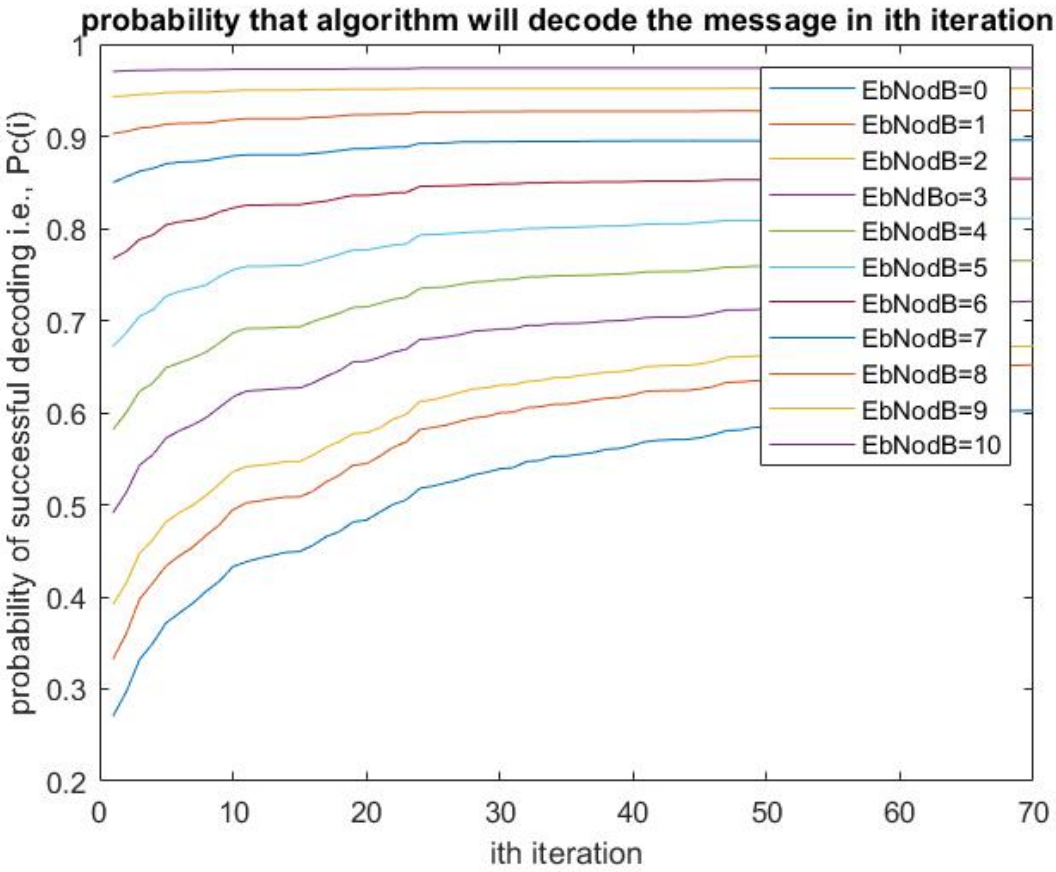
GRAPH OF HARD_DECISION FOR NR_2_6_52 BASE MATRIX



GRAPH OF HARD_DECISION FOR NR_1_5_352 BASE MATRIX



GRAPH OF HARD_DECISION FOR 9x12 BASE MATRIX



Soft Decision Decoder

Probability of success in decoding vs EbNo in dB

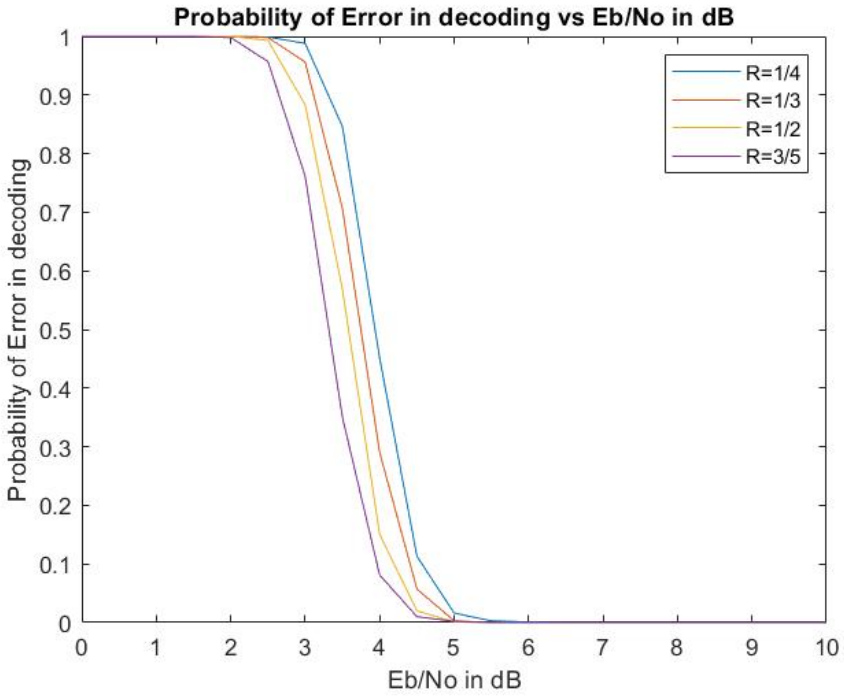
```
% load 5G NR LDPC base H matrix, use both NR_2_6_52 and NR_1_5_352
baseGraph5GNR = 'NR_2_6_52';
Number_of_rate = 1;
% varying this in the set {1/4, 1/3, 1/2, 3/5} for 2_6_52 and in the range
{1/3, 1/2, 3/5 and 4/5} for 1_5_352
code_rate = [1/4, 1/3, 1/2, 3/5];
% varying Eb/No in Db from 0 to 20 in step size of 0.5
EbNodB = 0:0.5:10;
% Convert the base H matrix to binary H matrix
[B,Hfull,z] = nrldpc_Hmatrix(baseGraph5GNR);
% 5G NR specific details
[mb,nb] = size(B); kb = nb - mb;
% Number of information bits
kNumInfoBits = kb * z;
error_detection_probability = zeros(length(code_rate),length(EbNodB));
for code_rate = [1/4 1/3, 1/2, 3/5]

    % Some 5G NR specific details
    k_pc = kb-2; nbRM = ceil(k_pc/code_rate)+2;
    nBlockLength = nbRM * z; % Number of encoded bits

    % Next three lines are some 5G NR specific details
    H = Hfull(:,1:nBlockLength);
    nChecksNotPunctured = mb*z - nb*z + nBlockLength;
    % this is the binary H matrix
    H = H(1:nChecksNotPunctured,:);
    % Number of CNs (we have denoted this as U = N - K in the class)
    Nchecks = size(H,1);
    % Generate information (or message) bit vector
    b = randi([0 1],[kNumInfoBits 1]);
    % Encode using 5G NR LDPC base matrix
    c = nrldpc_encode(B,z,b');
    c = c(1:nBlockLength)';

    No_of_cols = length(H);
    No_of_rows = height(H);

    degree_CN = 0;
    % calculate maximum degree of CN from all check nodes
    for i =1:1:No_of_rows
        temp = 0;
        for j=1:1:No_of_cols
            if H(i,j) == 1
                temp = temp + 1;
            end
```

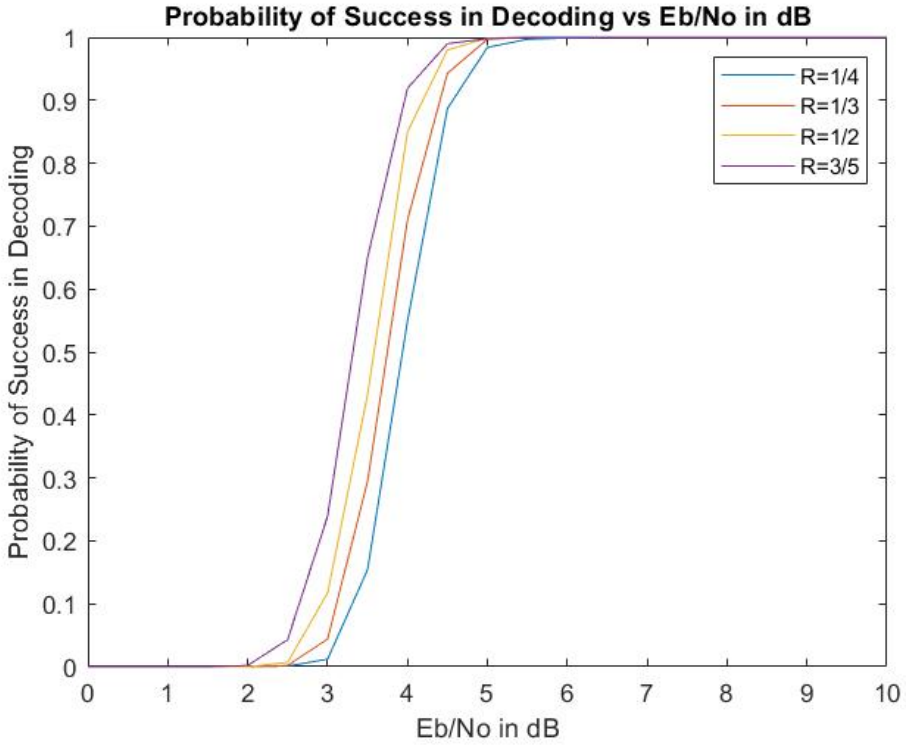
```
        end
        if temp > degree_CN
            degree_CN = temp;
        end
    end
    % calculate maximum degree of VN from all variable nodes
    degree_VN = 0;
    for i =1:1:No_of_cols
        temp =0;
        for j =1:1:No_of_rows
            if H(j,i) == 1
                temp = temp + 1;
            end
        end
        if temp > degree_VN
            degree_VN = temp;
        end
    end

    connection_CN = zeros(No_of_rows, degree_CN);
    connection_VN = zeros(No_of_cols, degree_VN);
    % connection_CN(i,j) will give the position of VN where jth connection of
ith CN is connected
    connection_CN = make_connection_CN(degree_CN,H);
    % connection_VN(i,j) will give the position of CN where jth connection of
ith VN is connected
    connection_VN = make_connection_VN(degree_VN,H);
    % encoded codeword which will be transmitted through AWGN channel
    current_message=c';
    transmit_msg = current_message;
    % modulated codeword which will be transmitted through AWGN channel
    modulated_msg = 1 - 2*transmit_msg;

    Number_of_sim = 10000;

    iteration_max = 50;


    EbNo = 10.^(EbNodB./10);

    sigma = sqrt(1./(2.*code_rate.*EbNo));
    success_decoding_probability = zeros(1, length(EbNodB));

    for V = 1:1:length(EbNodB)
        success = 0;

        for U = 1:1:Number_of_sim
            flag = 0;
            % output from AWGN channel which contains noise also with
variation sigma^2 and mean 0
            received_msg = modulated_msg + sigma(1,V)*randn(1,No_of_cols);
```

```
            likelihood_ratio = (2*received_msg)./(sigma(1,V)*sigma(1,V));
            % sent_by_CN(i,j) information sent by ith CN to jth connection
            sent_by_CN = zeros(size(connection_CN));
            % received_by_CN(i,j) information received by ith CN from jth
connection
            received_by_CN = zeros(size(connection_CN));
            % sent_by_VN(i,j) information sent by ith VN to jth connection
            sent_by_VN = zeros(size(connection_VN));
            % received_by_VN(i,j) information received by ith VN from jth
connection
            received_by_VN = zeros(size(connection_VN));
            % current log likelihood ratio of all the recieved bits
            current_lh = likelihood_ratio;
            for iteration = 1:1:iteration_max
              if iteration == 1
                  for o = 1:1:No_of_cols
                      sent_by_VN(o , : )= repmat(likelihood_ratio(o),
1,degree_VN);
                  end
              else
                      sent_by_VN = repetition_code_1(received_by_VN,
current_lh, connection_VN);
              end
              received_by_CN = received_at_CN(sent_by_VN, connection_VN,
connection_CN);
              sent_by_CN = SPC(received_by_CN, connection_CN);
              received_by_VN = received_at_VN(sent_by_CN, connection_CN,
connection_VN );
              current_lh = repetition_code_2(received_by_VN, current_lh);
              % converting into message bits from the log likelihood ratio
              current_msg = current_lh < 0;
              % comapring the transmitted message and the decoded message
after every iteration
              comparing = mod(current_msg+transmit_msg, 2);
              % breaking the decoding process if transmitted message and the
decoded message matches
              if sum(comparing, 2) == 0
                  flag =1;
                  break;
              end
            end
            % counting the times the code has successfully decoded the message
            success = success + flag;

        end
        success_decoding_probability(1, V) = success/Number_of_sim;
        error_detection_probability(Number_of_rate, V) = 1 -
success_decoding_probability(1, V);
    end

    Number_of_rate = Number_of_rate + 1;
```

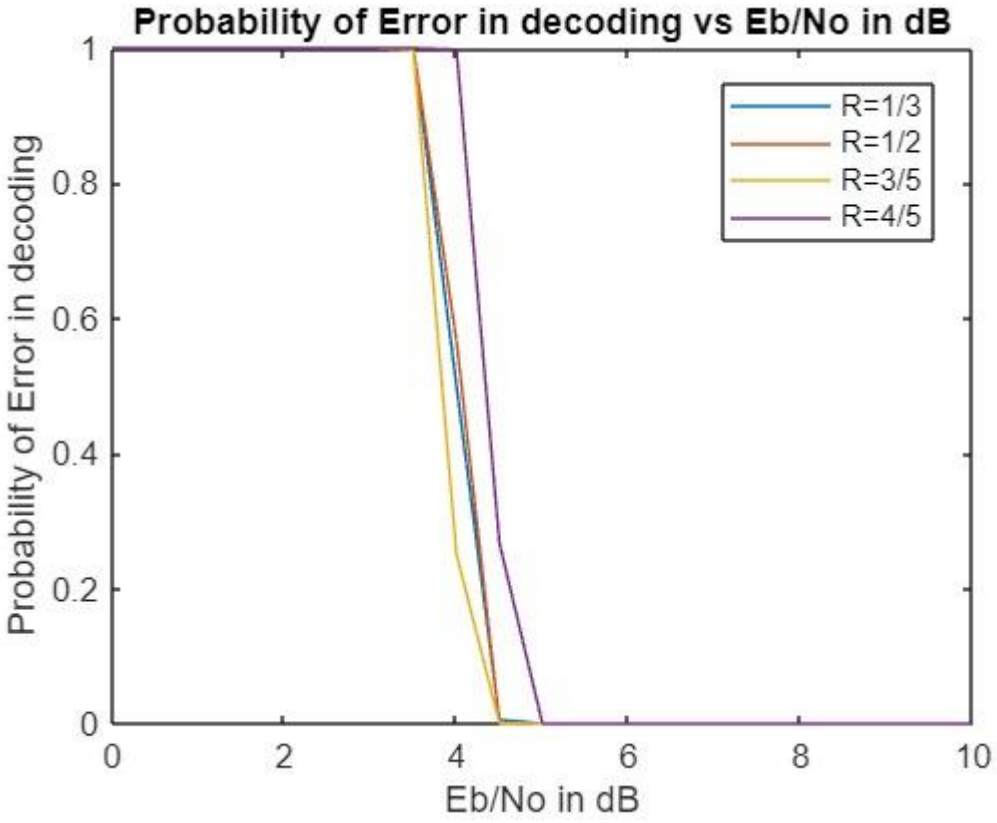
```
    % PLotting the graph of successful decoding probability with varying Eb/No
in dB for various code rates
    plot(EbNodB, success_decoding_probability);
    hold on;
end
xlabel('Eb/No in dB');
ylabel('Probability of Success in Decoding');
title('Probability of Success in Decoding vs Eb/No in dB');
legend('R=1/4' ,'R=1/3' ,'R=1/2', 'R=3/5');
hold off;


figure ;
for i=1:1:4
    error_in_decoding = error_detection_probability(i,:) ;
    % PLotting the graph of  probability of error occured in decoding with
varying Eb/No in dB for various code rates
    plot(EbNodB, error_in_decoding);
hold on;
end
xlabel('Eb/No in dB');
ylabel('Probability of Error in decoding');
title('Probability of Error in decoding vs Eb/No in dB');
legend('R=1/4' ,'R=1/3' ,'R=1/2', 'R=3/5');
```

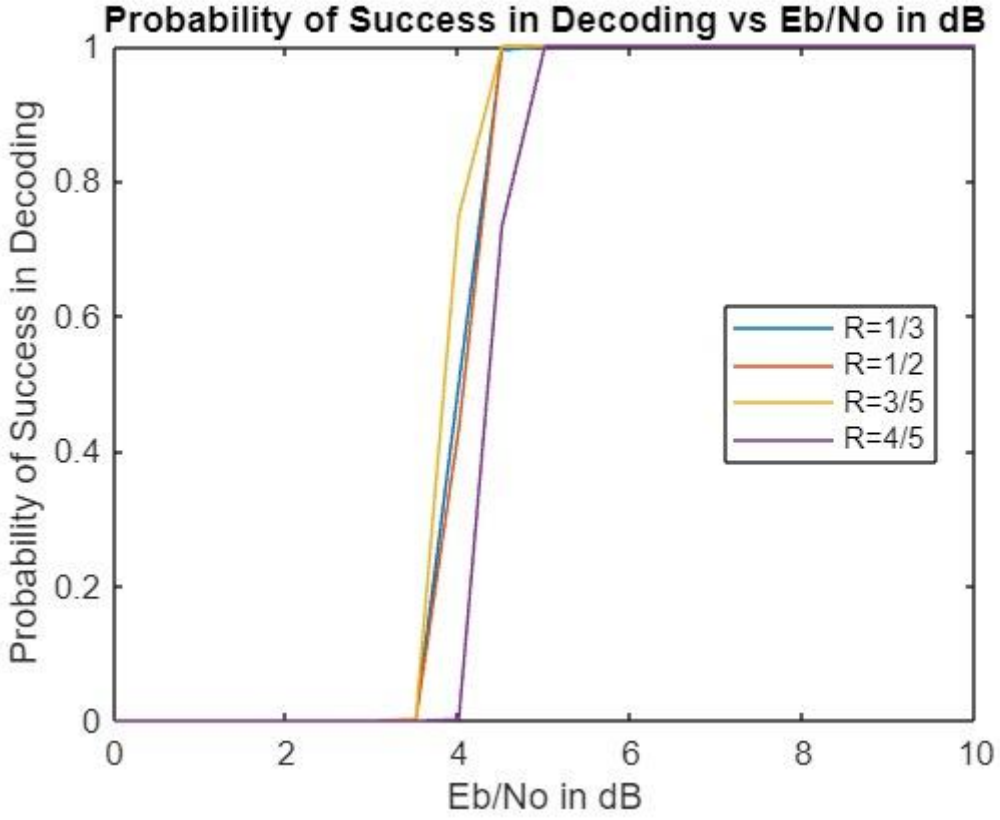
GRAPH OF SOFT DECISION FOR NR_2_6_52



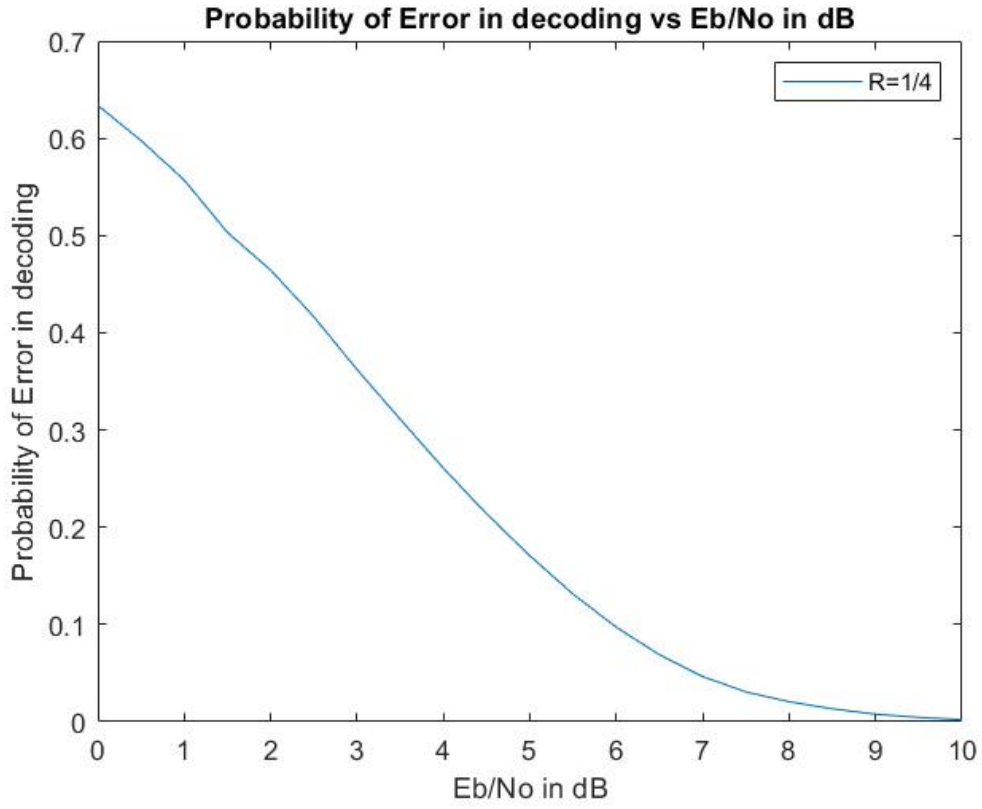


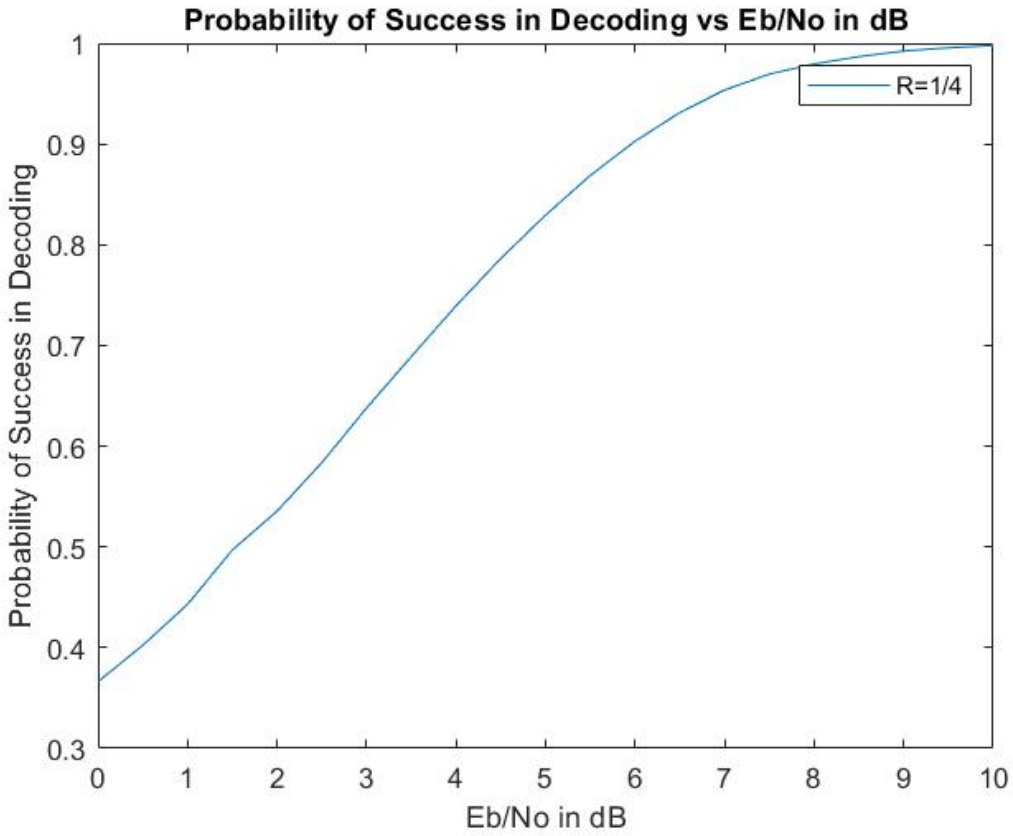
GRAPH_OF_SOFT_DECISION FOR NR_1_5_352





GRAPH_OF_SOFT_DECISION FOR 9x12





Soft Decision Decoder Bit Error Probability

BEP in decoding VS EbNodB

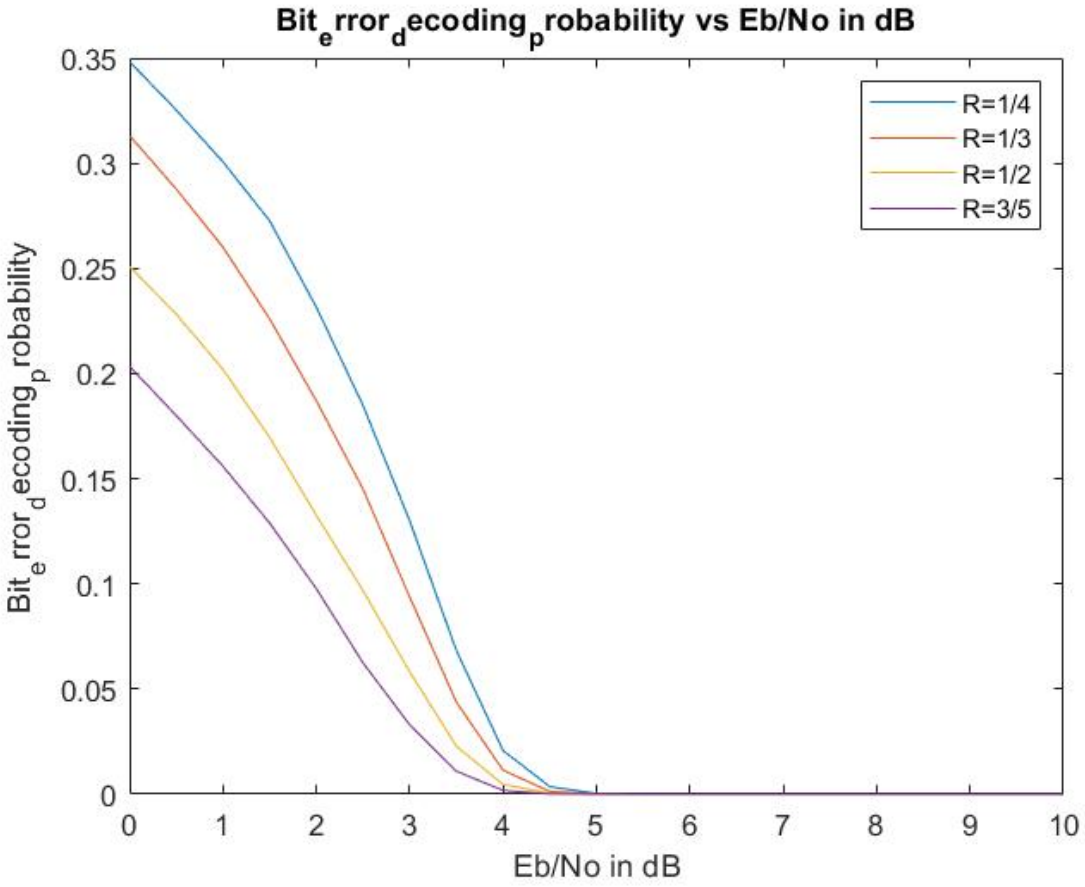
```
% load 5G NR LDPC base H matrix, use both NR_2_6_52 and NR_1_5_352
baseGraph5GNR = 'NR_2_6_52';
Number_of_rate = 1;
% varying this in the set {1/4, 1/3, 1/2, 3/5} for 2_6_52 and in the range
{1/3, 1/2, 3/5 and 4/5} for 1_5_352
code_rate = [1/4, 1/3, 1/2, 3/5];
% varying Eb/No in Db from 0 to 20 in step size of 0.5
EbNodB = 0:0.5:10;
% Convert the base H matrix to binary H matrix
[B,Hfull,z] = nrldpc_Hmatrix(baseGraph5GNR);
% 5G NR specific details
[mb,nb] = size(B); kb = nb - mb;
% Number of information bits
    kNumInfoBits = kb * z;
%error_detection_probability = zeros(length(code_rate),length(EbNodB));
for code_rate = [1/4 1/3, 1/2, 3/5]

    % Some 5G NR specific details
    k_pc = kb-2; nbRM = ceil(k_pc/code_rate)+2;
    nBlockLength = nbRM * z; % Number of encoded bits

    % Next three lines are some 5G NR specific details
    H = Hfull(:,1:nBlockLength);
    nChecksNotPunctured = mb*z - nb*z + nBlockLength;
    % this is the binary H matrix
    H = H(1:nChecksNotPunctured,:);
    % Number of CNs (we have denoted this as U = N - K in the class)
    Nchecks = size(H,1);
    % Generate information (or message) bit vector
    b = randi([0 1],[kNumInfoBits 1]);
    % Encode using 5G NR LDPC base matrix
    c = nrldpc_encode(B,z,b');
    c = c(1:nBlockLength)';

    No_of_cols = length(H);
    No_of_rows = height(H);

    degree_CN = 0;
    % calculate maximum degree of CN from all check nodes
    for i =1:1:No_of_rows
        temp = 0;
        for j=1:1:No_of_cols
            if H(i,j) == 1
                temp = temp + 1;
            end
        end
        if temp > degree_CN
```

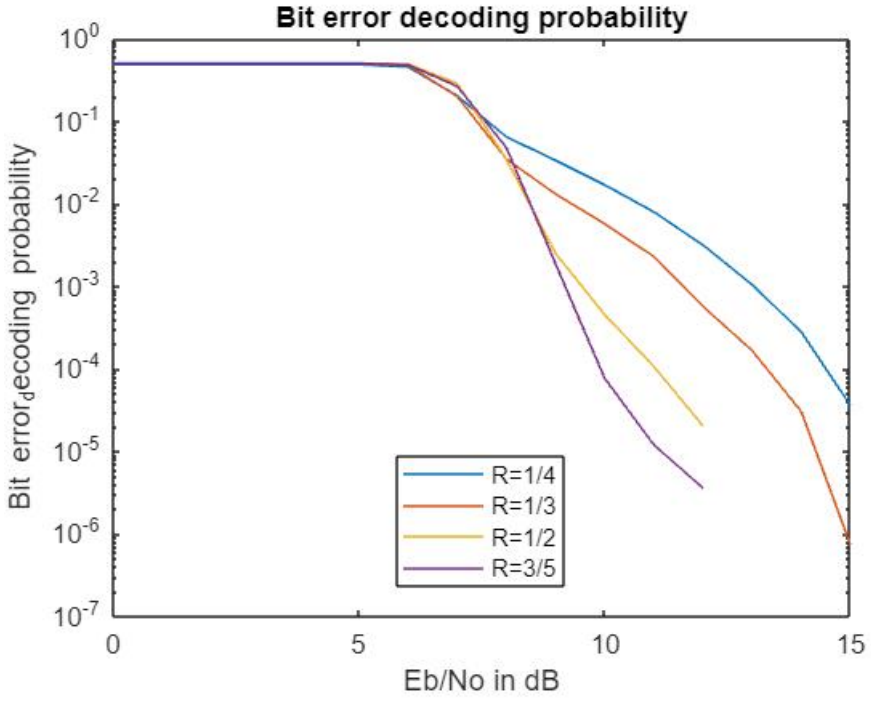
```
            degree_CN = temp;
        end
    end
    % calculate maximum degree of VN from all variable nodes
    degree_VN = 0;
    for i =1:1:No_of_cols
        temp =0;
        for j =1:1:No_of_rows
            if H(j,i) == 1
                temp = temp + 1;
            end
        end
        if temp > degree_VN
            degree_VN = temp;
        end
    end

    connection_CN = zeros(No_of_rows, degree_CN);
    connection_VN = zeros(No_of_cols, degree_VN);
    % connection_CN(i,j) will give the position of VN where jth connection of
ith CN is connected
    connection_CN = make_connection_CN(degree_CN,H);
    % connection_VN(i,j) will give the position of CN where jth connection of
ith VN is connected
    connection_VN = make_connection_VN(degree_VN,H);
    % encoded codeword which will be transmitted through AWGN channel
    current_message=c';
    transmit_msg = current_message;
    % modulated codeword which will be transmitted through AWGN channel
    modulated_msg = 1 - 2*transmit_msg;

    Number_of_sim = 10000;

    iteration_max = 50;


    EbNo = 10.^(EbNodB./10);

    sigma = sqrt(1./(2.*code_rate.*EbNo));
    %success_decoding_probability = zeros(1, length(EbNodB));
    bit_error_decoding_probability = zeros(1,length(EbNodB));
    for V = 1:1:length(EbNodB)
        success = 0;

        error = 0;
        for U = 1:1:Number_of_sim
            flag = 0;
            % output from AWGN channel which contains noise also with
variation sigma^2 and mean 0
            received_msg = modulated_msg + sigma(1,V)*randn(1,No_of_cols);
            likelihood_ratio = (2*received_msg)./(sigma(1,V)*sigma(1,V));
```

```
            % sent_by_CN(i,j) information sent by ith CN to jth connection
            sent_by_CN = zeros(size(connection_CN));
            % received_by_CN(i,j) information received by ith CN from jth
connection
            received_by_CN = zeros(size(connection_CN));
            % sent_by_VN(i,j) information sent by ith VN to jth connection
            sent_by_VN = zeros(size(connection_VN));
            % received_by_VN(i,j) information received by ith VN from jth
connection
            received_by_VN = zeros(size(connection_VN));
            % current log likelihood ratio of all the recieved bits
            current_lh = likelihood_ratio;
            for iteration = 1:1:iteration_max
              if iteration == 1
                  for o = 1:1:No_of_cols
                      sent_by_VN(o , : )= repmat(likelihood_ratio(o),
1,degree_VN);
                  end
              else
                      sent_by_VN = repetition_code_1(received_by_VN,
current_lh, connection_VN);
              end
              received_by_CN = received_at_CN(sent_by_VN, connection_VN,
connection_CN);
              sent_by_CN = SPC(received_by_CN, connection_CN);
              received_by_VN = received_at_VN(sent_by_CN, connection_CN,
connection_VN );
              current_lh = repetition_code_2(received_by_VN, current_lh);
              % converting into message bits from the log likelihood ratio
              current_msg = current_lh < 0;
              % comapring the transmitted message and the decoded message
after every iteration
              comparing = mod(current_msg+transmit_msg, 2);
              % breaking the decoding process if transmitted message and the
decoded message matches
              if sum(comparing, 2) == 0
                  flag =1;
                  break;
              end
            end
            % counting the times the code has successfully decoded the message
            success = success + flag;
            error = error + sum(comparing, 2);
        end
        %success_decoding_probability(1, V) = success/Number_of_sim;
        %error_detection_probability(Number_of_rate, V) = 1 -
success_decoding_probability(1, V);
        bit_error_decoding_probability(1,V) =
error/(length(transmit_msg)*Number_of_sim);
    end
```

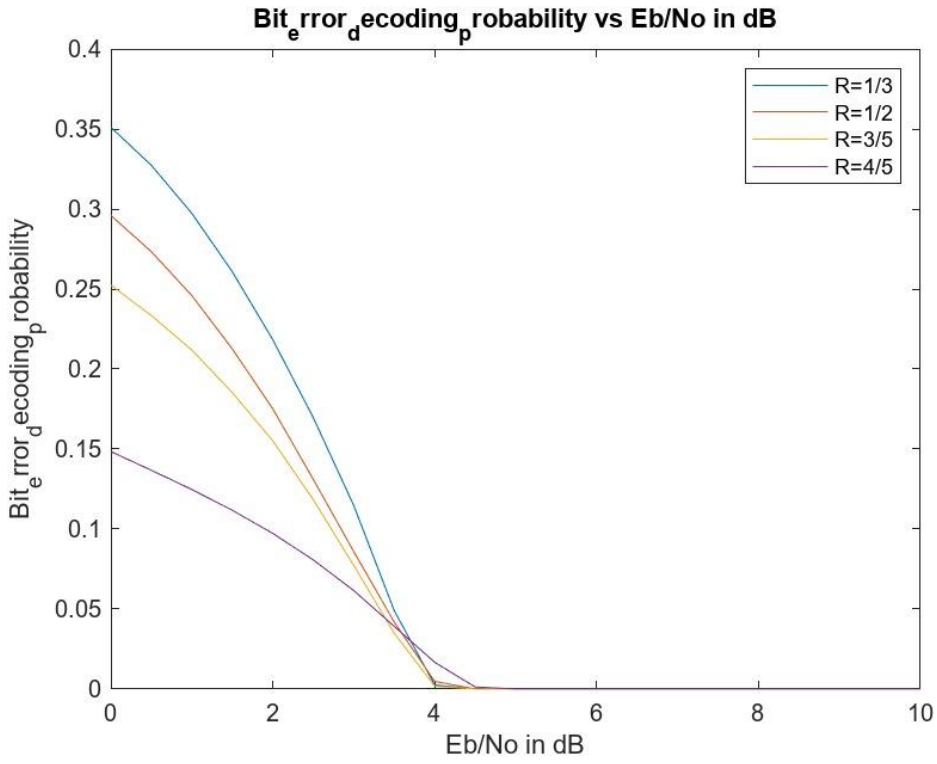
```
    Number_of_rate = Number_of_rate + 1;
    % PLotting the graph of bit_error_decoding_probability with varying Eb/No
in dB for various code rates
    plot(EbNodB, bit_error_decoding_probability);
    hold on;
end
xlabel('Eb/No in dB');
ylabel('Bit_error_decoding_probability');
title('Bit_error_decoding_probability vs Eb/No in dB');
legend('R=1/4' ,'R=1/3' ,'R=1/2', 'R=3/5');
hold off;
```

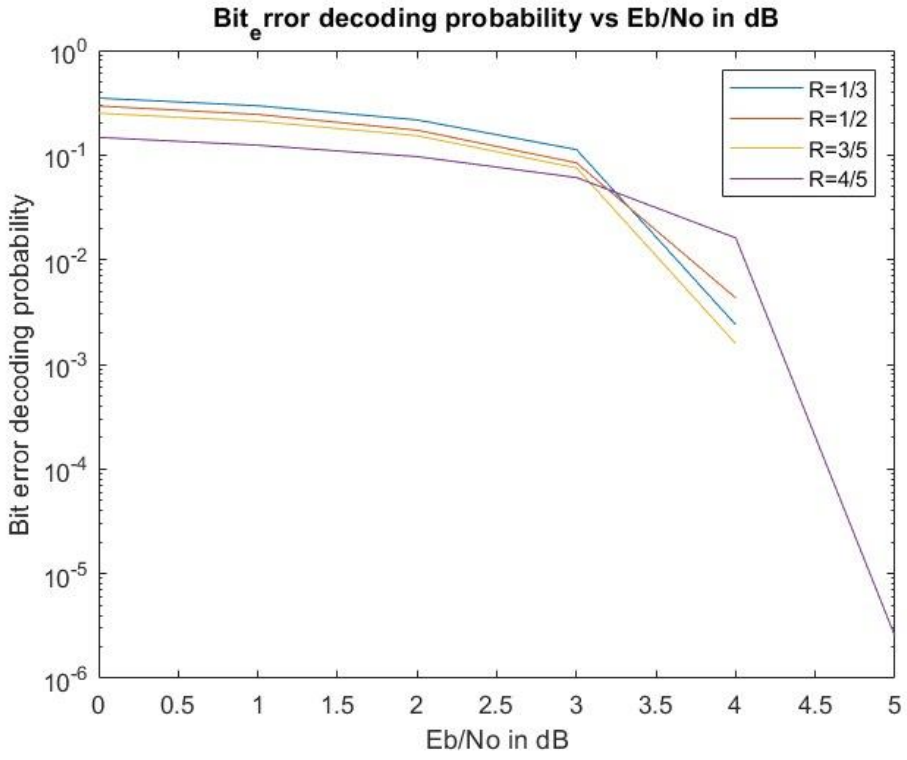
Graph For Bit Error Decoding Probability NR-2-6-52



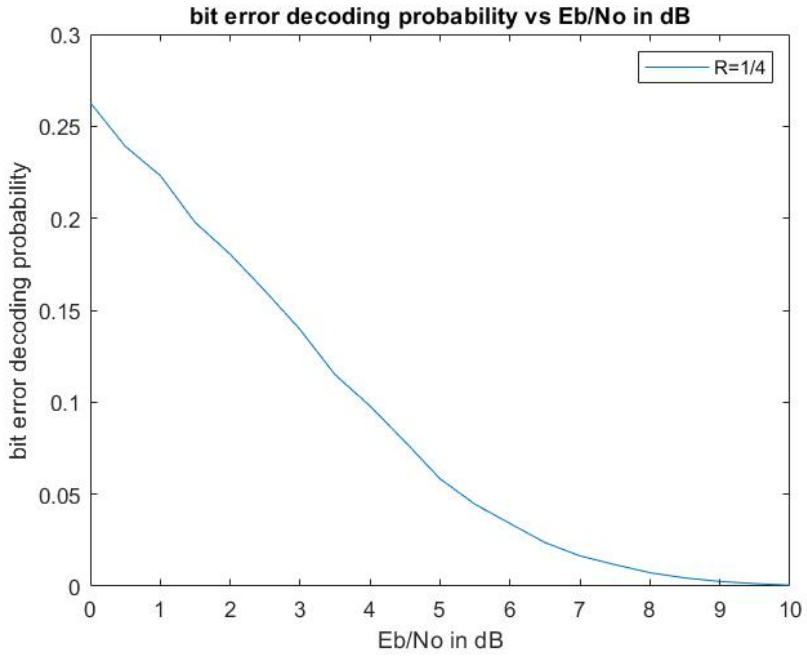


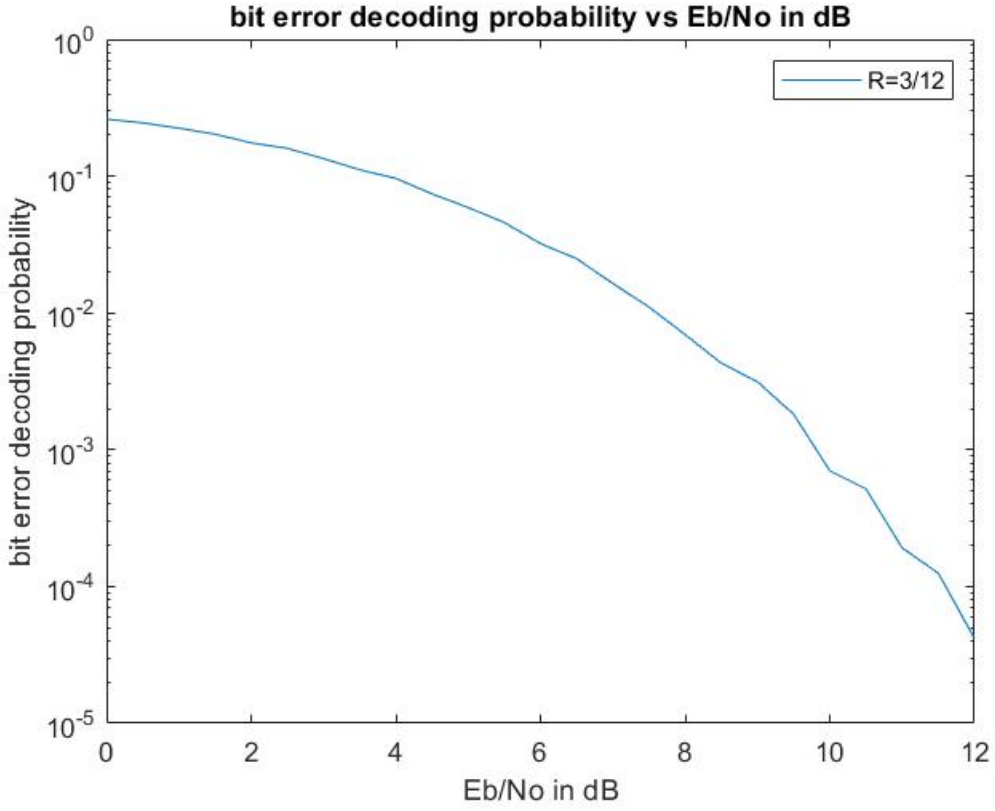
Graph For Bit Error Decoding Probability NR-1-5-352





Graph For Bit Error Decoding Probability 9X12 H matrix





Soft Decision Decoder Iteration

Probability of successful decoding at the ith iteration for various values of EbNo in dB

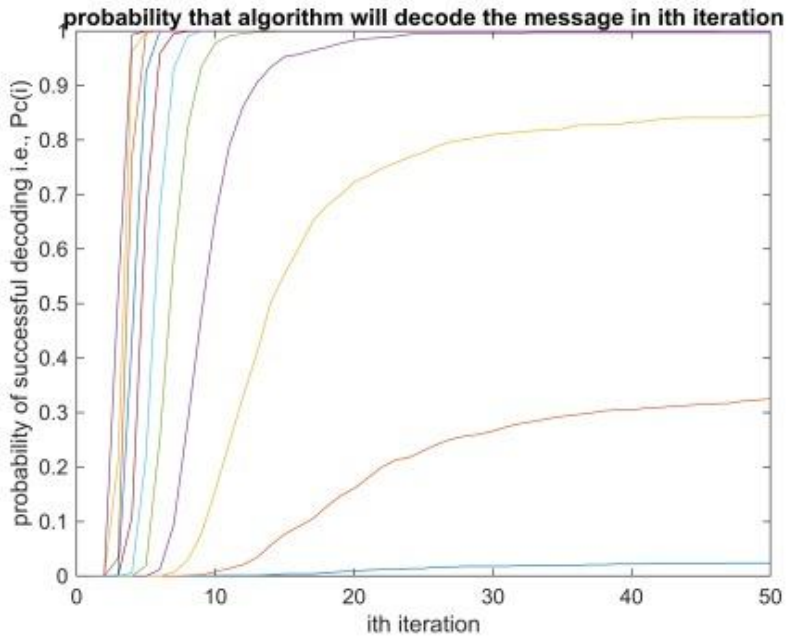
```
    % load 5G NR LDPC base H matrix, use both NR_2_6_52 and NR_1_5_352
    baseGraph5GNR = 'NR_2_6_52';
    Number_of_rate = 1;
    % varying this in the set {1/4, 1/3, 1/2, 3/5} for 2_6_52 and in the range
{1/3, 1/2, 3/5 and 4/5} for 1_5_352
    code_rate = [1/4] ;
    % varying Eb/No in Db from 6 to 10 in step size of 0.5
    EbNodB = 1:0.5:6;
    error_detection_probability = zeros(length(code_rate),length(EbNodB));
    % Convert the base H matrix to binary H matrix
    [B,Hfull,z] = nrldpc_Hmatrix(baseGraph5GNR);
    % 5G NR specific details
    [mb,nb] = size(B); kb = nb - mb;
    % Number of information bits
    kNumInfoBits = kb * z;
    % Some 5G NR specific details
    k_pc = kb-2; nbRM = ceil(k_pc/code_rate)+2;
    nBlockLength = nbRM * z; % Number of encoded bits

    % Next three lines are some 5G NR specific details
    H = Hfull(:,1:nBlockLength);
    nChecksNotPunctured = mb*z - nb*z + nBlockLength;
    % this is the binary H matrix
    H = H(1:nChecksNotPunctured,:);
    % Number of CNs (we have denoted this as U = N - K in the class)
    Nchecks = size(H,1);
    % Generate information (or message) bit vector
    b = randi([0 1],[kNumInfoBits 1]);
    % Encode using 5G NR LDPC base matrix
    c = nrldpc_encode(B,z,b');
    c = c(1:nBlockLength)';

    No_of_cols = length(H);
    No_of_rows = height(H);

    degree_CN = 0;
    % calculate maximum degree of CN from all check nodes
    for i =1:1:No_of_rows
        temp = 0;
        for j=1:1:No_of_cols
            if H(i,j) == 1
                temp = temp + 1;
            end
        end
        if temp > degree_CN
            degree_CN = temp;
```

```
        end
    end
    % calculate maximum degree of VN from all variable nodes
    degree_VN = 0;
    for i =1:1:No_of_cols
        temp =0;
        for j =1:1:No_of_rows
            if H(j,i) == 1
                temp = temp + 1;
            end
        end
        if temp > degree_VN
            degree_VN = temp;
        end
    end

    connection_CN = zeros(No_of_rows, degree_CN);
    connection_VN = zeros(No_of_cols, degree_VN);
    % connection_CN(i,j) will give the position of VN where jth connection of
ith CN is connected
    connection_CN = make_connection_CN(degree_CN,H);
    % connection_VN(i,j) will give the position of CN where jth connection of
ith VN is connected
    connection_VN = make_connection_VN(degree_VN,H);
    % encoded codeword which will be transmitted through AWGN channel
    current_message=c';
    transmit_msg = current_message;
    % modulated codeword which will be transmitted through AWGN channel
    modulated_msg = 1 - 2*transmit_msg;

    Number_of_sim = 10000;

    iteration_max = 50;
    total_iteration = 1:1:iteration_max;

    EbNo = 10.^(EbNodB./10);

    sigma = sqrt(1./(2.*code_rate.*EbNo));
    % success_decoding_probability = zeros(1, length(EbNodB));

    for V = 1:1:length(EbNodB)
        success = 0;
        iteration_success = zeros(Number_of_sim, iteration_max);
        iteration_probability = zeros(1,iteration_max);
        for U = 1:1:Number_of_sim
            flag = 0;
            % output from AWGN channel which contains noise also with
variation sigma^2 and mean 0
            received_msg = modulated_msg + sigma(1,V)*randn(1,No_of_cols);
            likelihood_ratio = (2*received_msg)./(sigma(1,V)*sigma(1,V));
            % sent_by_CN(i,j) information sent by ith CN to jth connection
```

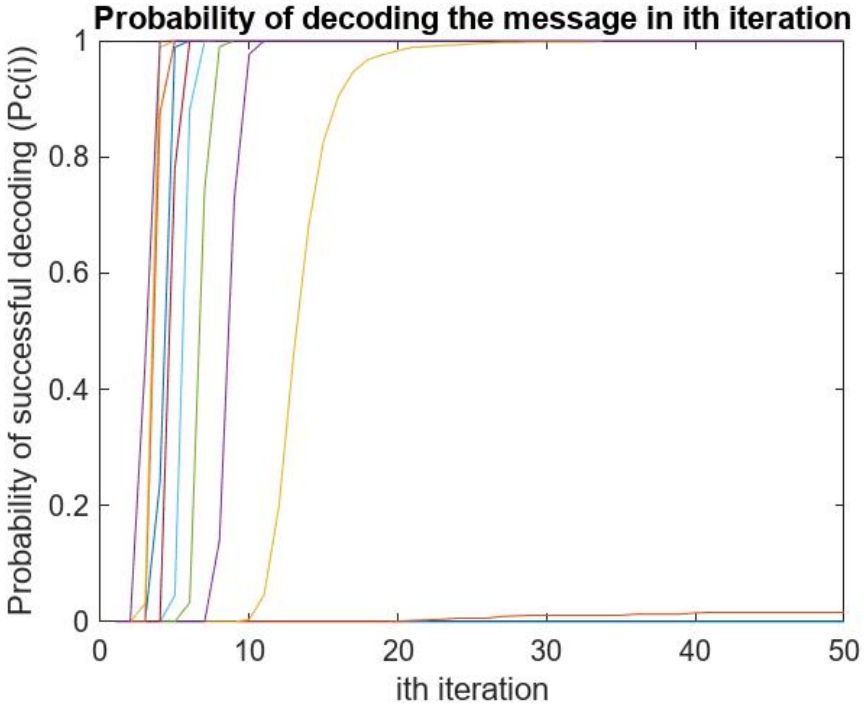
```
            sent_by_CN = zeros(size(connection_CN));
            % received_by_CN(i,j) information received by ith CN from jth
connection
            received_by_CN = zeros(size(connection_CN));
            % sent_by_VN(i,j) information sent by ith VN to jth connection
            sent_by_VN = zeros(size(connection_VN));
            % received_by_VN(i,j) information received by ith VN from jth
connection
            received_by_VN = zeros(size(connection_VN));
            % current log likelihood ratio of all the recieved bits
            current_lh = likelihood_ratio;
            for iteration = 1:1:iteration_max
              if iteration == 1
                  for o = 1:1:No_of_cols
                      sent_by_VN(o , : )= repmat(likelihood_ratio(o),
1,degree_VN);
                  end
              else
                      sent_by_VN = repetition_code_1(received_by_VN,
likelihood_ratio, connection_VN);
              end
              received_by_CN = received_at_CN(sent_by_VN, connection_VN,
connection_CN);
              sent_by_CN = SPC(received_by_CN, connection_CN);
              received_by_VN = received_at_VN(sent_by_CN, connection_CN,
connection_VN );
              current_lh = repetition_code_2(received_by_VN, likelihood_ratio);
              % converting into message bits from the log likelihood ratio
              current_msg = current_lh < 0;
              % comapring the transmitted message and the decoded message
after every iteration
              comparing = mod(current_msg+transmit_msg, 2);
              % breaking the decoding process if transmitted message and the
decoded message matches
              if sum(comparing, 2) == 0
                  flag =1;
                   iteration_success(U,iteration:iteration_max) =
iteration_success(U,iteration:iteration_max)  +1;
                  break;
              end
            end
            % counting the times the code has successfully decoded the message
            success = success + flag;

        end
        % success_decoding_probability(1, V) = success/Number_of_sim;
        % error_detection_probability(Number_of_rate, V) = 1 -
success_decoding_probability(1, V);
        summation =sum(iteration_success,1);

        iteration_probability= summation/Number_of_sim;
```

```
        plot(total_iteration, iteration_probability);
        hold on
    end

    Number_of_rate = Number_of_rate + 1;
    xlabel('ith iteration');
    ylabel('probability of successful decoding i.e., Pc(i)');
    title("probability that algorithm will decode the message in ith
iteration");
```

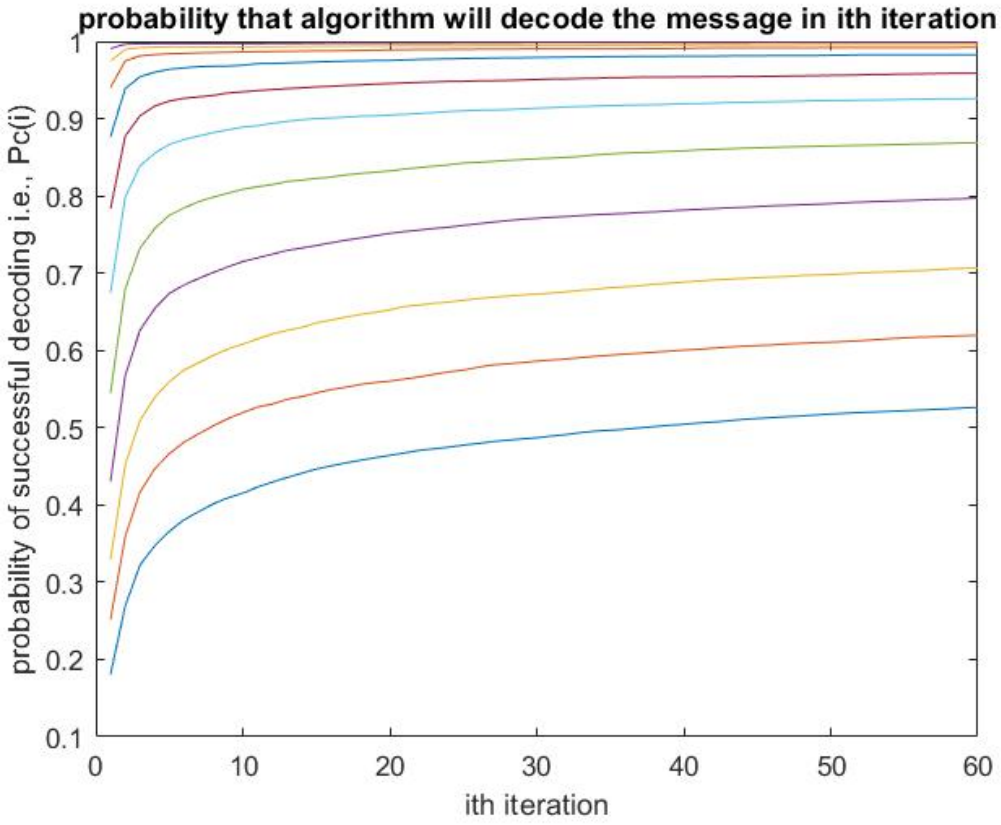
GRAPH OF SOFT DECISION DECODER FOR NR_2_6_52



GRAPH OF SOFT DECISION DECODER FOR NR_1_5_352



GRAPH OF SOFT DECISION DECODER FOR 9x12 MATRIX



Functions

```
function [B,H,z] = nrldpc_Hmatrix(BG) % constructs the parity check H matrix
(given in the starter code)


load(sprintf('%s.txt',BG),BG);
B = NR_2_6_52;
[mb,nb] = size(B);
z = 52;
H = zeros(mb*z,nb*z);
Iz = eye(z); I0 = zeros(z);
for kk = 1:mb
    tmpvecR = (kk-1)*z+(1:z);
    for kk1 = 1:nb
        tmpvecC = (kk1-1)*z+(1:z);
        if B(kk,kk1) == -1
            H(tmpvecR,tmpvecC) = I0;
        else
            H(tmpvecR,tmpvecC) = circshift(Iz,-B(kk,kk1));
        end
    end
end


[U,N]=size(H); K = N-U;
P = H(:,1:K);
G = [eye(K); P];
Z = H*G;


end
% shifting the identity matrix circularly by k positions to the right (given
in the starter code)
function Y = mul_sh(x, k)
    if (k==-1)
        Y = zeros(1, length(x));
    else
        Y = [x(k+1:end) x(1:k)];
    end
end


% generating valid codeword with the help of the parity check H matrix (given
in the starter code)
function cword = nrldpc_encode(B,z,msg)
```

```
%B: base matrix
%z: expansion factor
%msg: message vector, length = (#cols(B)-#rows(B))*z
%cword: codeword vector, length = #cols(B)*z


[m,n] = size(B);


cword = zeros(1,n*z);
cword(1:(n-m)*z) = msg;


%double-diagonal encoding
temp = zeros(1,z);
for i = 1:4 %row 1 to 4
    for j = 1:n-m %message columns
        temp = mod(temp + mul_sh(msg((j-1)*z+1:j*z),B(i,j)),2);
    end
end
if B(2,n-m+1) == -1
    p1_sh = B(3,n-m+1);
else
    p1_sh = B(2,n-m+1);
end
cword((n-m)*z+1:(n-m+1)*z) = mul_sh(temp,z-p1_sh); %p1
%Find p2, p3, p4
for i = 1:3
    temp = zeros(1,z);
    for j = 1:n-m+i
        temp = mod(temp + mul_sh(cword((j-1)*z+1:j*z),B(i,j)),2);
    end
    cword((n-m+i)*z+1:(n-m+i+1)*z) = temp;
end
%Remaining parities
for i = 5:m
    temp = zeros(1,z);
    for j = 1:n-m+4
        temp = mod(temp + mul_sh(cword((j-1)*z+1:j*z),B(i,j)),2);
    end
    cword((n-m+i-1)*z+1:(n-m+i)*z) = temp;
end


end


% making connections of CNs with their respective VNs
function connection_CN = make_connection_CN(degree_CN,H)
    col = length(H);
    row = height(H);
    connection_CN = zeros(row, degree_CN);
```

```
    for i = 1:row
        temp = 1;
        for j = 1:col
            if H(i,j) == 1
                connection_CN(i,temp) = j;
                temp = temp + 1;
            end
        end
    end
end


% making connections of VNs with their respective CNs
function connection_VN = make_connection_VN(degree_VN,H)
    col = length(H);
    row = height(H);
    connection_VN = zeros(col, degree_VN);
    for i = 1:col
        temp = 1;
        for j = 1:row
            if H(j,i) == 1
                connection_VN(i,temp) = j;
                temp = temp + 1;
            end
        end
    end
end


% calculating information at VNs
function information = repetition_code_1(received_by_VN, demodulated_msg,
connection_of_VN, degree_each_VN)
    [r, c] = size(connection_of_VN);
    information = zeros(size(connection_of_VN));
    row_sum = sum(received_by_VN,2);
    row_sum= row_sum';
    row_sum = row_sum + demodulated_msg;
    for i= 1:1:r
        for j=1:1:c
            if connection_of_VN(i,j) ~= 0
                row_new_sum = row_sum(1,i);
                row_new_sum = row_new_sum - received_by_VN(i,j);
                if row_new_sum > degree_each_VN(1,i)/2
                    information(i, j) = 1;
                else
                    information(i, j) = 0;
                end
            end
        end


    end
```

```
end


% received information at CNs from their respective VNs
function information = received_at_CN(sent_by_VN, connection_VN, connection_CN)
 [r, c]= size(connection_VN);
    information = zeros(size(connection_CN));
    temp = ones(1, height(connection_CN));
    for i = 1:1:r
        for j = 1:1:c
            if connection_VN(i,j) ~= 0
                information(connection_VN(i,j), temp(1,connection_VN(i,j))) =
sent_by_VN(i,j);
                temp(1,connection_VN(i,j)) = temp(1,connection_VN(i,j)) +1 ;
            end
        end
    end
end


% calculating information ratio at CNs
function information = SPC(received_by_CN, connection_of_CN)
    [r, c] = size(connection_of_CN);
    information = zeros(size(connection_of_CN));
    row_sum = sum(received_by_CN,2);
    row_sum= row_sum';
    for i = 1:1:r
        for j = 1:1:c
            if connection_of_CN(i,j) ~= 0
                row_new_sum = row_sum(1,i);
                row_new_sum = row_new_sum - received_by_CN(i,j);
                if mod(row_new_sum,2) == 1
                    information(i,j) = 1;
                else
                    information(i,j) = 0;
                end
            end
        end
    end
end


% received information at VNs from their respective CNs
function information = received_at_VN(sent_by_CN, connection_CN, connection_VN)
    [r, c]= size(connection_CN);
    information = zeros(size(connection_VN));
    temp = ones(1, height(connection_VN));
    for i = 1:1:r
        for j = 1:1:c
            if connection_CN(i,j) ~= 0
```

```
                information(connection_CN(i,j), temp(1,connection_CN(i,j))) =
sent_by_CN(i,j);
                temp(1,connection_CN(i,j)) = temp(1,connection_CN(i,j)) +1 ;
            end
        end
    end
end


% calculating information at VNs after completion of each iteration
function information = repetition_code_2(received_by_VN, demodulated_msg,
degree_each_VN)
    [r, c] = size(received_by_VN);
    information = zeros(1,length(demodulated_msg));
    row_sum = sum(received_by_VN,2);
    row_sum= row_sum';
    row_sum = row_sum + demodulated_msg;
    for i = 1:1:r
        if row_sum(1,i) > (degree_each_VN(1,i)+1)/2
            information(1,i)= 1;
        else
            information(1,i)= 0;
        end
    end


end
```

SOFT DECISION DECODING METHOD 2

```
H = [1 0 0 0 0 1 0 1 0 1 0 0;
    1 0 0 1 1 0 0 0 0 0 1 0;
    0 1 0 0 1 0 1 0 1 0 0 0;
    0 0 1 0 0 1 0 0 0 0 1 1;
    0 0 1 0 0 0 1 1 0 0 0 1;
    0 1 0 0 1 0 0 0 1 0 1 0;
    1 0 0 1 0 0 1 0 0 1 0 0;
    0 1 0 0 0 1 0 1 0 1 0 0;
    0 0 1 1 0 0 0 0 1 0 0 1];
No_of_cols = length(H);
No_of_rows = height(H);


EbNodB = 0:0.5:15;
EbNo = 10.^(EbNodB./10);
code_rate=1/4;
transmit_msg= zeros(1,No_of_cols);
modulated_msg = 1 - 2*transmit_msg;
Number_of_sim = 10000;
iteration_max = 50;
sigma = sqrt(1./(2.*code_rate.*EbNo));
success_decoding_probability= zeros(1,length(EbNodB));
error_decoding_probability= zeros(1,length(EbNodB));
bit_error_decoding_probability = zeros(1,length(EbNodB));
for  V = 1:1:length(EbNodB)
    success = 0;
    error =0;
    for U= 1:1:Number_of_sim
        flag = 0;
        received_msg = modulated_msg + sigma(1,V)*randn(1,No_of_cols);
        r_likelihood_ratio = (2*received_msg)./(sigma(1,V)*sigma(1,V));
        L=zeros(size(H));
        [rows, cols] =size(H);


        total_likelihood_ratio = zeros(1,cols);
        for i=1:rows
            for j=1:cols
                if H(i, j) == 1
                    L(i, j) = r_likelihood_ratio(j);
                end
            end
        end


        for it= 1:1:iteration_max
            S = prod((2*(L>=0)-1), 2);
            S = S';
```

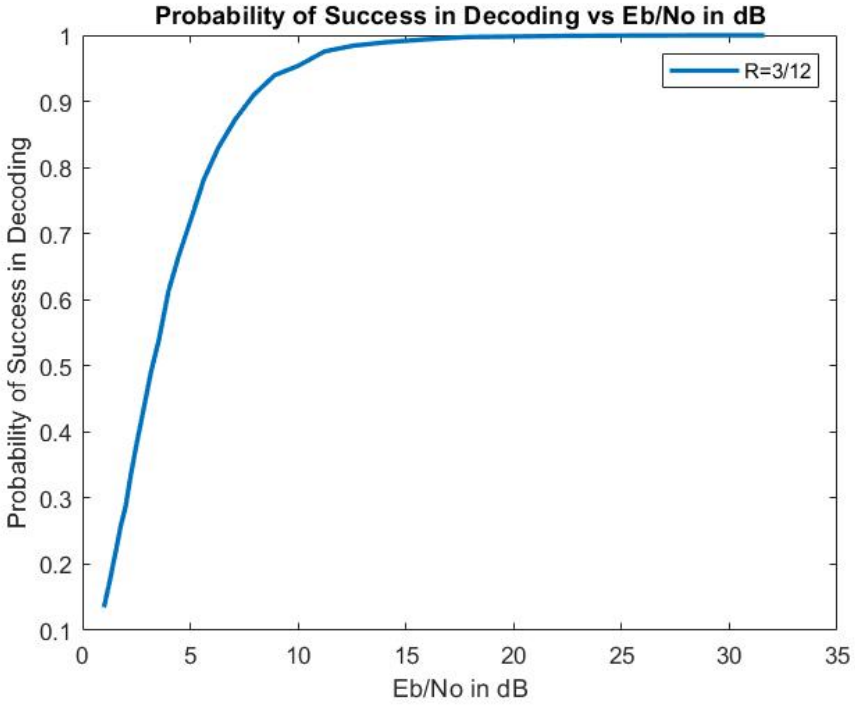
```
            L_abs = abs(L);
            for i=1:1:rows
                row_value = L_abs(i, :);
                [min_val_1, min_index, min_val_2] = calc_row_min(row_value);
                for j = 1:1:cols
                    if L(i,j) ~= 0
                        if j == min_index
                            L(i,j)=min_val_2*(2*(L(i,j)>=0)-1)*S(1,i);
                        else
                            L(i,j)=min_val_1*(2*(L(i,j)>=0)-1)*S(1,i);
                        end
                    end
                end




            end
            total_likelihood_ratio = sum(L);
            for i=1:rows
                for j=1:cols
                    if H(i, j) == 1
                        L(i, j) = total_likelihood_ratio(j)-L(i,j);
                    end
                end
            end
            decoded_msg = total_likelihood_ratio < 0;
            comparing = mod(decoded_msg+transmit_msg, 2);
            if sum(comparing,2)==0;
                flag =1;
                break;
            end
        end


        success = success + flag;
        error = error + sum(comparing, 2);


    end
    success_decoding_probability(1,V) = success/Number_of_sim;
    error_decoding_probability(1,V) = 1 - success_decoding_probability(1,V);
    bit_error_decoding_probability(1,V) =
error/(length(transmit_msg)*Number_of_sim);


end




figure
```

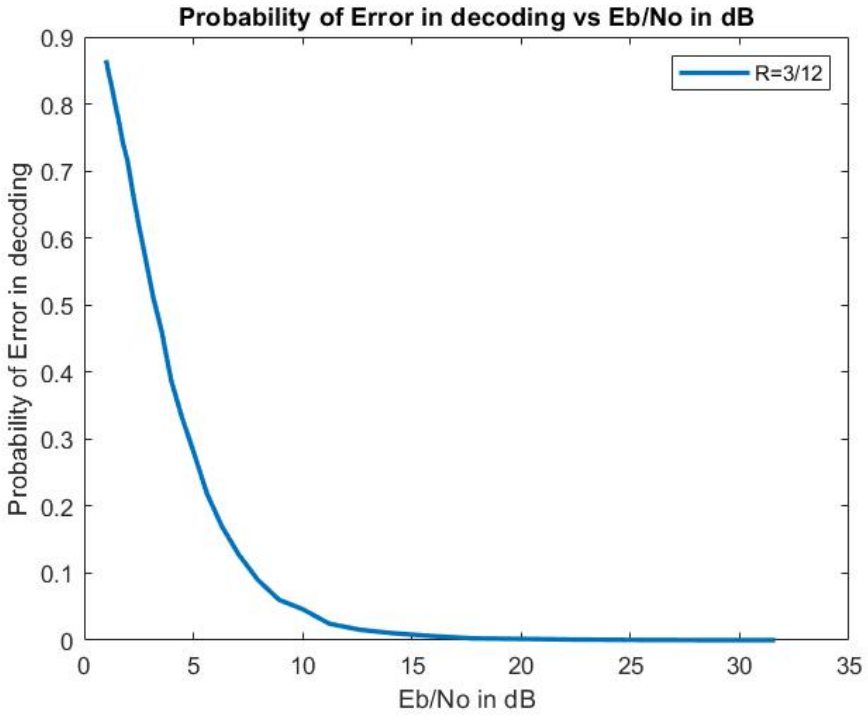
```
plot(EbNo,success_decoding_probability, 'LineWidth', 2);
xlabel('Eb/No in dB');
ylabel('Probability of Success in Decoding');
title('Probability of Success in Decoding vs Eb/No in dB');
legend('R=3/12');




figure;
plot(EbNo,error_decoding_probability, 'LineWidth', 2);
xlabel('Eb/No in dB');
ylabel('Probability of Error in decoding');
title('Probability of Error in decoding vs Eb/No in dB');
legend('R=3/12');


figure
plot(EbNodB, bit_error_decoding_probability, 'LineWidth', 2);
xlabel('Eb/No in dB');
ylabel('bit error decoding probability');
title('bit error decoding probability vs Eb/No in dB');
legend('R=3/12');


figure
semilogy(EbNodB, bit_error_decoding_probability, 'LineWidth', 2);
xlabel('Eb/No in dB');
ylabel('bit error decoding probability in logarithmic scale');
title('bit error decoding probability vs Eb/No in dB');
legend('R=3/12');
```

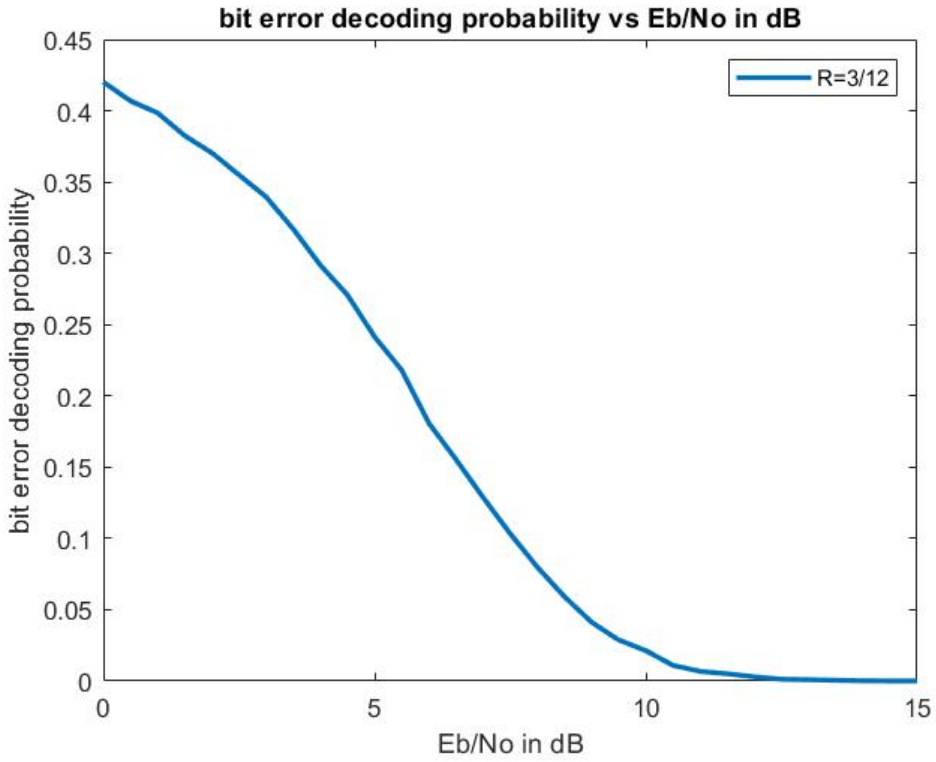
```
function [min_val_1, min_index, min_val_2] = calc_row_min(row_value)


min_val_1 = intmax('int64');
min_val_2 = intmax('int64');
min_index = 1;


for i = 1:1:length(row_value)
    if row_value(1,i) ~= 0
        if row_value(1,i) <= min_val_1
            min_val_1 = row_value(1,i);
            min_index = i;
        end
    end
end
for i = 1:1:length(row_value)
    if row_value(1,i) <= min_val_2
        if row_value(1,i) ~= 0
```

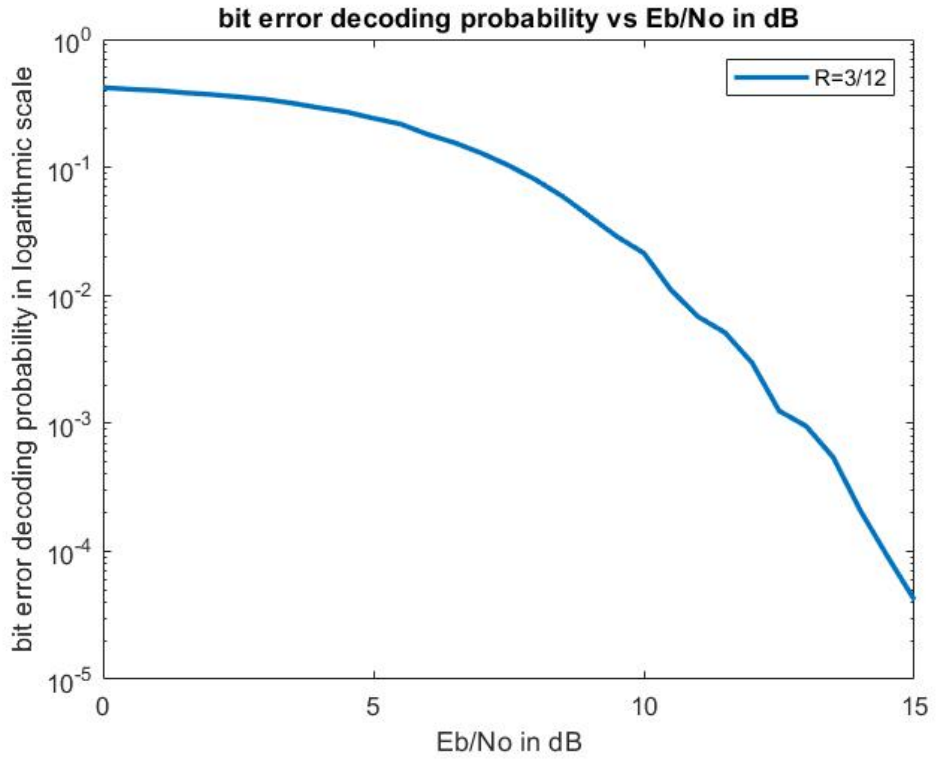
```
            if i ~= min_index
                min_val_2 = row_value(1,i);
            end
        end
    end
end
end
```

GRAPHS OF 9x12 H MATRIX USING METHOD 2 OF SOFT DECISION DECODING






bit error decoding probability vs Eb/No in dB
R=3/12
bit error decoding probability
Eb/No in dB
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15


bit error decoding probability vs Eb/No in dB
R=3/12
bit error decoding probability in logarithmic scale
Eb/No in dB
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
0
5
10
15