```
tic;

colors = [ 0.0, 0.7, 0.8;
           0.12, 0.34, 0.57;
           0.91, 0.15, 0.76;
           0.31, 0.12, 0.77;
           0.93, 0.13, 0.65;
           0.55, 0.51, 0.87;
           0.61, 0.78, 0.79;
           0.01, 0.31, 0.39;
           0.71, 0.25, 0.81;
           0.83, 0.69, 0.44;
           0.06, 0.40, 0.74;
           0.18, 0.18, 0.53;
           0.34, 0.72, 0.53;
           0.94, 0.38, 0.64;
           0.70, 0.15, 0.88;
           0.60, 0.67, 0.09;
           0.91, 0.29, 0.31;
           0.80, 0.86, 0.31;
           0.19, 0.93, 0.42;
           0.95, 0.79, 0.21;
           0.14, 0.41, 0.05
         ];

% for matrix NR_2_6_52
baseGraph5GNR = 'NR_2_6_52'; % load 5G NR LDPC base H matrix
coderate = [1/4 1/3 1/2 3/5];
eb_no_dbvec = 0:0.5:10;

[B,Hfull,z] = nrldpc_Hmatrix(baseGraph5GNR,52); % Convert base H matrix
nsim = 1000;
max_it = 20;
iterations = 1:1:max_it;

% We want a separate figure for each code rate
for idxCR = 1:length(coderate)
    cr = coderate(idxCR);
    disp(['Simulating code rate = ', num2str(cr)]);

    %-----------------------------------------
    % Perform rate matching (5G NR-specific details)
    %-----------------------------------------
    [mb,nb] = size(B);
    kb = nb - mb;
    kNumInfoBits = kb * z; % Number of information bits

    k_pc = kb - 2;
    nbRM = ceil(k_pc/cr) + 2;
    nBlockLength = nbRM * z;
```

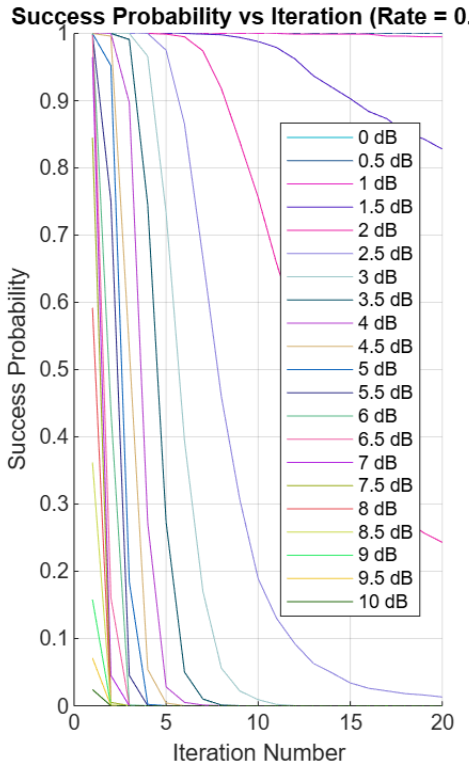
```
    H = Hfull(:,1:nBlockLength);
    nChecksNotPunctured = mb*z - nb*z + nBlockLength;
    H = H(1:nChecksNotPunctured,:);
    [row,col] = size(H);

    L = zeros(size(H));
    k = col - row;

    cn_to_vn_map = cn_vn(H);
    vn_to_cn_map = vn_cn(H);

    %-----------------------------------------
    % Set up storage for plotting
    %-----------------------------------------
    decoding_error = zeros(1,length(eb_no_dbvec));   % Decoding block-error
probability
    successProb_it = zeros(max_it, length(eb_no_dbvec));
    % ^ successProb_it(it, idxEbNo) will store the success probability
    %   at iteration "it" for each Eb/No.

    %-----------------------------------------
    % Start a new figure for this code rate
    %-----------------------------------------
    figure('Name',['Code rate = ',num2str(cr)], 'NumberTitle','off');

    % We can make two subplots:
    % 1) Success Probability vs. iteration
    % 2) Decoding error probability vs. Eb/No
    %
    % Subplot 1: success probability vs iteration
    subplot(1,2,1);
    hold on;
    title(['Success Probability vs Iteration (Rate = ',num2str(cr),')']);
    xlabel('Iteration Number');
    ylabel('Success Probability');
    grid on;

    %-----------------------------------------
    % Main Eb/No loop
    %-----------------------------------------
    for d_iter = 1:length(eb_no_dbvec)
        eb_no_db = eb_no_dbvec(d_iter);
        eb_no = 10^(eb_no_db/10);
        sigma = sqrt(1/(2*cr*eb_no));

        % Counters
        successCount = 0;
        % We'll track how many blocks succeeded at each iteration
        iterationSuccess = nsim .* ones(1, max_it);
```

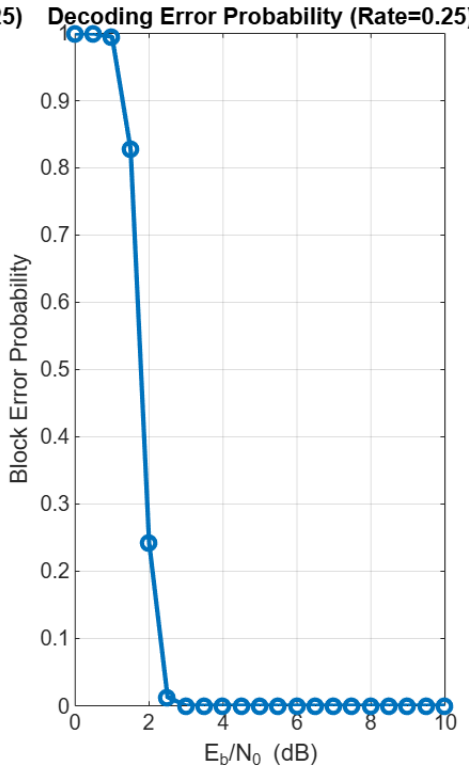
```
for sim = 1:nsim
    %--------------------------------
    % Generate random message bits
    %--------------------------------
    org_msg = randi([0 1],[k 1]);

    % Encode
    encoded_msg = nrldpc_encode(B,z,org_msg');
    encoded_msg = encoded_msg(1:nBlockLength);

    % BPSK modulation
    bpsk_msg = 1 - 2.*encoded_msg;

    % AWGN
    noise = sigma * randn(1,nBlockLength);
    received_bpsk = bpsk_msg + noise;

    % Soft decoding initialization
    received_bits = (received_bpsk<0);
    prev_msg = received_bits;
    vn_sum_vec = zeros(1,col);

    %--------------------------------
    % Iterative decoding
    %--------------------------------
    for it = 1:max_it

        % VN -> CN
        if it == 1
            % First iteration: pass received LLR to CN
            for i = 1:col
                for j = vn_to_cn_map{i,1}
                    L(j,i) = received_bpsk(1,i);
                end
            end
        else
            % Subsequent iterations:
            % Subtract the old extrinsic from the sum
            for i = 1:col
                for j = vn_to_cn_map{i,1}
                    L(j,i) = vn_sum_vec(i) - L(j,i);
                end
            end
        end

        % CN -> VN using min-sum
        for i = 1:row
            min1 = 1e9;
            min2 = 1e9;
```

```
                    pos = -1;
                    total_sign = 1;

                    for j = cn_to_vn_map{i,1}
                        ele = abs(L(i,j));
                        % find min1 and min2
                        if ele <= min1
                            min2 = min1;
                            min1 = ele;
                            pos = j;
                        elseif ele <= min2 && ele > min1
                            min2 = ele;
                        end
                        % product of signs
                        if L(i,j)~=0
                            total_sign = total_sign * sign(L(i,j));
                        end
                    end

                    % send the message
                    for j = cn_to_vn_map{i,1}
                        if j ~= pos
                            L(i,j) = total_sign * sign(L(i,j)) * min1;
                        else
                            L(i,j) = total_sign * sign(L(i,j)) * min2;
                        end
                    end
                end

                % VN sum
                for i = 1:col
                    sum1 = received_bpsk(1,i);
                    sum1 = sum1 + sum(L(:,i));
                    vn_sum_vec(i) = sum1;
                end

                c_hat = (vn_sum_vec < 0);

                % Check if the first k bits match
                if sum(xor(c_hat(1:k), org_msg')) == 0
                    successCount = successCount + 1;
                    break;
                else
                    iterationSuccess(it) = iterationSuccess(it) - 1;
                end

                % Early stopping if no change
                if sum(xor(prev_msg, c_hat)) == 0
                    % If nothing changed, the decoder won't change in future
iterations
```

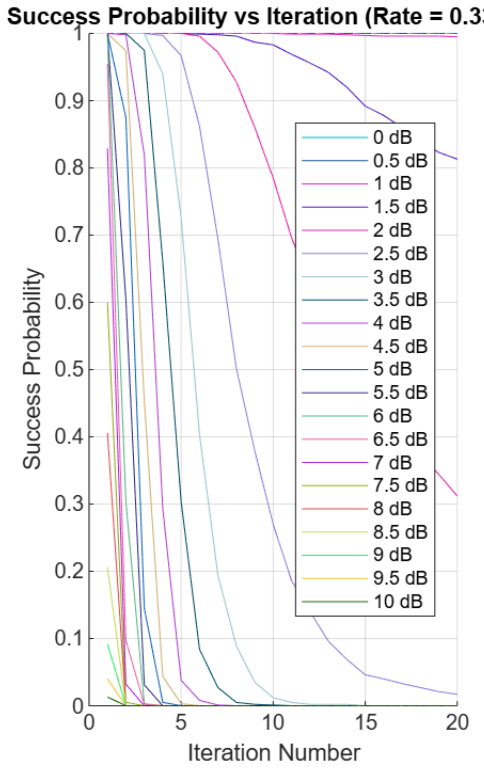
```
                    for tmp_itr = it+1:max_it
                        iterationSuccess(tmp_itr) =
iterationSuccess(tmp_itr) - 1;
                    end
                    break;
                end
                prev_msg = c_hat;
            end
        end

        %---------------------------------
        % Store final decoding error probability
        %---------------------------------
        decoding_error(d_iter) = (nsim - successCount) / nsim;

        %---------------------------------
        % Convert iterationSuccess => success probability at each iteration
        %---------------------------------
        successProb_it(:, d_iter) = 1 - (iterationSuccess ./ nsim);

        % Now plot success probability vs iteration for this Eb/No
        % on the left subplot:
        plot(iterations, 1 - (iterationSuccess ./ nsim), ...
             'Color', colors(d_iter,:), ...
             'DisplayName', [num2str(eb_no_db), ' dB']);
    end

    % Add legend for all Eb/No curves
    legend('show', 'Location', 'best');
    hold off;

    %---------------------------------
    % Subplot 2: Decoding error probability vs Eb/No
    %---------------------------------
    subplot(1,2,2);
    plot(eb_no_dbvec, decoding_error, '-o', 'LineWidth', 2);
    grid on;
    xlabel('E_b/N_0 (dB)');
    ylabel('Block Error Probability');
    title(['Decoding Error Probability (Rate=', num2str(cr), ')']);

end % end of for loop over code rates
```

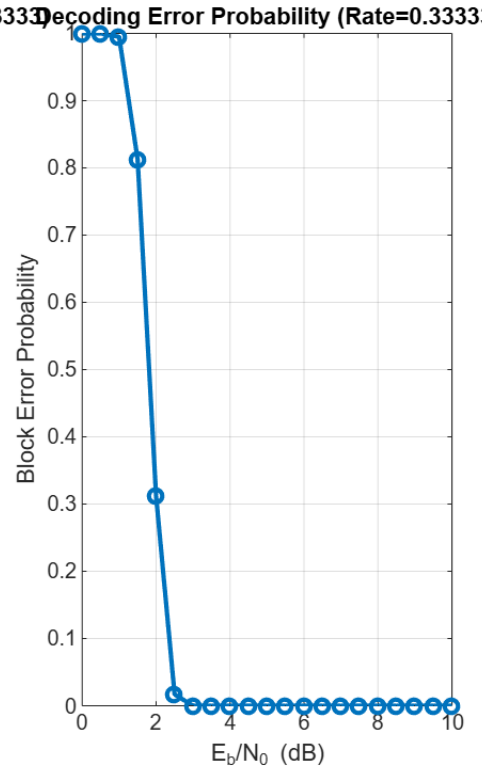
```
Simulating code rate = 0.25
```



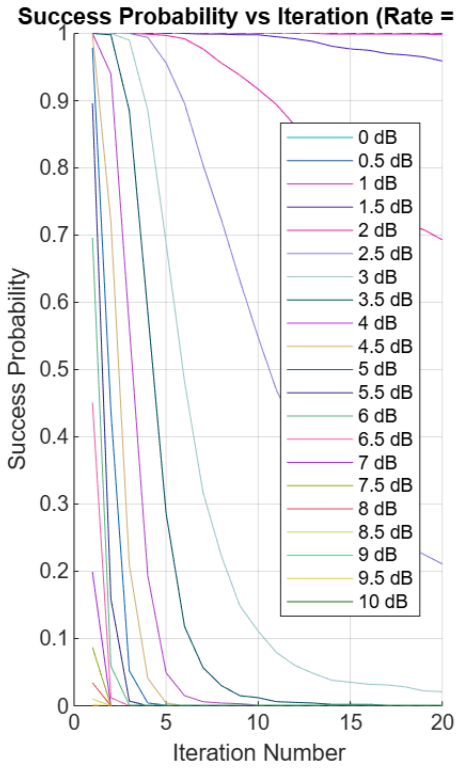


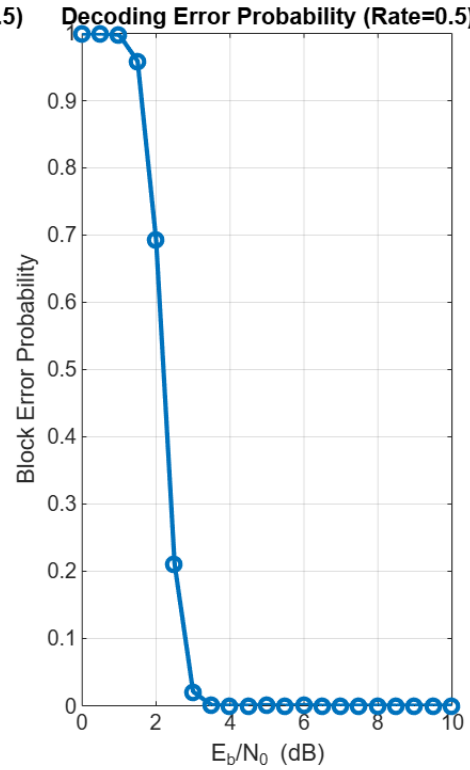

```
Simulating code rate = 0.33333
```



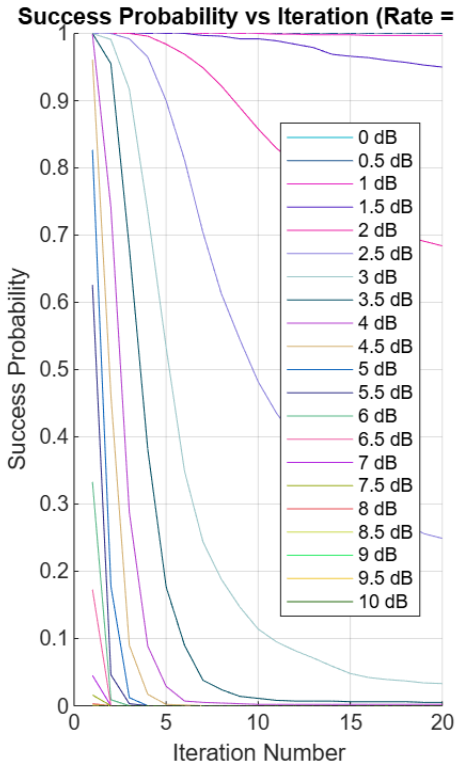


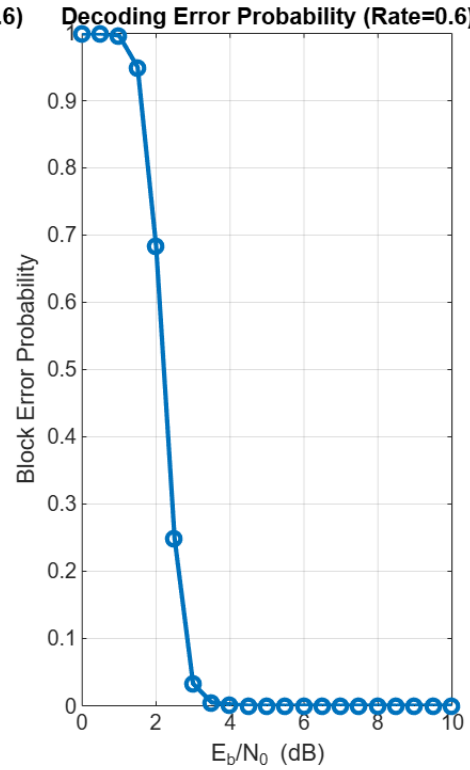

```
Simulating code rate = 0.5
```






```
Simulating code rate = 0.6
```






```
toc;
```

```
Elapsed time is 1454.587408 seconds.
```

```
%==============================
%   AUXILIARY FUNCTIONS BELOW
%==============================
function [B,H,z] = nrldpc_Hmatrix(BG,z)
    load(sprintf('%s.txt',BG),BG);
    B = NR_2_6_52;
    [mb,nb] = size(B);
    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
end

function out = cn_vn(H)
    [row, col] = size(H);
    out = cell(row,1);
    for i = 1:row
        out{i,1} = [];
        for j = 1:col
            if(H(i,j)==1)
                out{i,1} = [out{i,1}, j];
            end
        end
    end
end

function out = vn_cn(H)
    [row, col] = size(H);
    out = cell(col,1);
    for i = 1:col
        out{i,1} = [];
        for j = 1:row
            if(H(j,i)==1)
                out{i,1} = [out{i,1}, j];
            end
        end
```

```
    end
end

function cword = nrldpc_encode(B,z,msg)
    [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
        for j = 1:n-m
            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

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
```