```
clc; clear; close all;

baseGraph5GNR = 'NR_2_6_52';
codeRates = [1/4, 1/3, 1/2, 3/5];

[B, Hfull, z] = nrldpc_Hmatrix(baseGraph5GNR);
[mb, nb] = size(B);
kb = nb - mb;

EbNodB = 0:0.5:10;  % Wider range
Nsim = 10;          % Number of Monte Carlo runs
maxItr = 15;        % Max decoding iterations

figure;
hold on;

for rIdx = 1:length(codeRates)
    codeRate = codeRates(rIdx);
    kNumInfoBits = kb * z;
    k_pc = kb - 2;
    nbRM = ceil(k_pc / codeRate) + 2;
    nBlockLength = nbRM * z;

    H = Hfull(:, 1:nBlockLength);
    nChecksNotPunctured = mb*z - nb*z + nBlockLength;
    H = H(1:nChecksNotPunctured, :);

    [u, n] = size(H);         % u = CNs, n = VNs
    k = n - u;

    % Create adjacency lists
    adj_VN = cell(1, n);
    for l = 1:n
        adj_VN{l} = find(H(:, l))';
    end

    adj_CN = cell(1, u);
    for l = 1:u
        adj_CN{l} = find(H(l, :));
    end

    plotvec = zeros(1, length(EbNodB));  % BER for each SNR

    for idx = 1:length(EbNodB)
        jEb = EbNodB(idx);
        EbNo = 10^(jEb/10);
        sigma = sqrt(1 / (2 * codeRate * EbNo));
        errors = 0;

        for NsimIdx = 1:Nsim
```

```
            b = randi([0 1], [kNumInfoBits 1]);  % Random message bits
            c = nrldpc_encode(B, z, b');         % Encode
            c = c(1:nBlockLength)';

            s = 1 - 2 * c;                       % BPSK modulation
            r = s + sigma * randn(nBlockLength, 1);  % AWGN channel
            r_hard = (r < 0);                    % Hard decision
            VN = r_hard;                         % Initialize VNs

            msg_CN_2_VN = cell(1, n);            % Messages from CNs to VNs

            for it = 1:maxItr
                % First iteration
                if it == 1
                    for i = 1:u
                        sumxor = mod(sum(VN(adj_CN{i})), 2);
                        for j = 1:length(adj_CN{i})
                            idx_vn = adj_CN{i}(j);
                            tempxor = mod(sumxor + VN(idx_vn), 2);
                            msg_CN_2_VN{idx_vn}(end + 1) = tempxor;
                        end
                    end
                else
                    % VN to CN messages
                    msg_VN_2_CN = cell(1, u);
                    for i = 1:n
                        cnt = r_hard(i);
                        for j = 1:length(adj_VN{i})
                            cnt = cnt + msg_CN_2_VN{i}(j);
                        end
                        for j = 1:length(adj_VN{i})
                            tempcnt = cnt - msg_CN_2_VN{i}(j);
                            msg_VN_2_CN{adj_VN{i}(j)}(end+1) =
double(tempcnt > (length(adj_VN{i}) / 2));
                        end
                    end

                    % VN update
                    tempVN = zeros(1, n);
                    for i = 1:n
                        cnt1 = r_hard(i);
                        for j = 1:length(adj_VN{i})
                            cnt1 = cnt1 + msg_CN_2_VN{i}(j);
                        end
                        tempVN(i) = double(cnt1 > ((length(adj_VN{i}) + 1)/
2));
                    end

                    if isequal(tempVN, VN)
                        break;
```

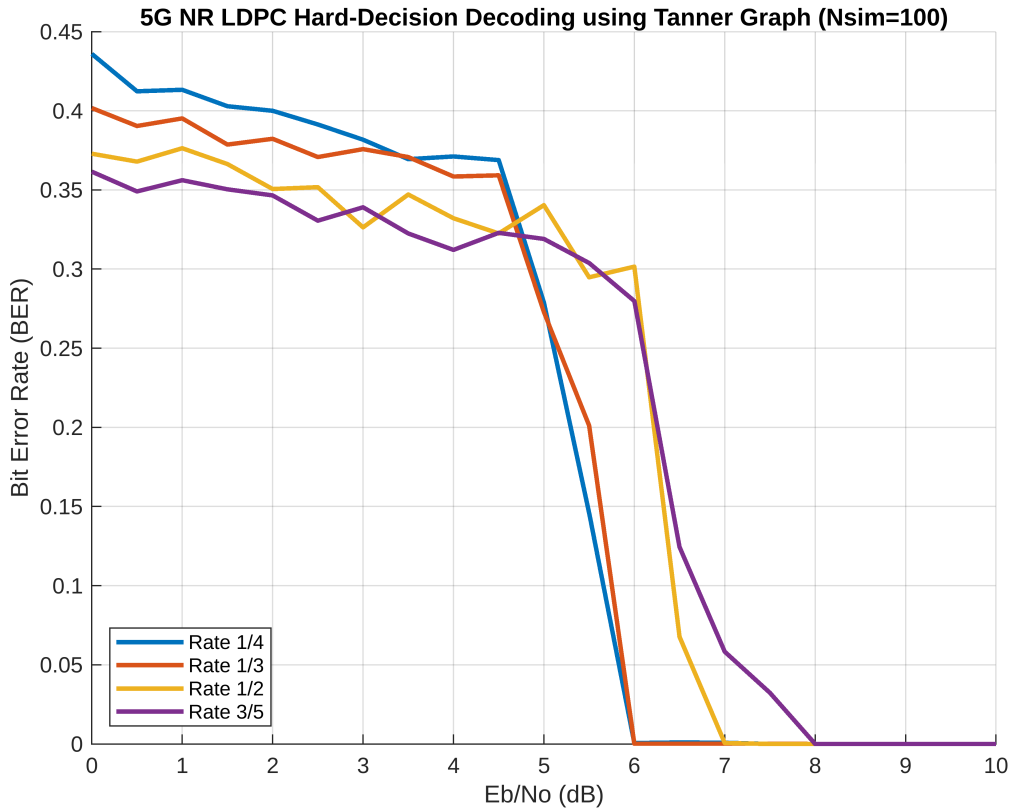
```
                    end

                    VN = tempVN;

                    % CN to VN messages update
                    msg_CN_2_VN = cell(1, n);
                    for i = 1:u
                        sumxor = mod(sum(msg_VN_2_CN{i}), 2);
                        for j = 1:length(adj_CN{i})
                            idx_vn = adj_CN{i}(j);
                            tempxor = mod(sumxor + msg_VN_2_CN{i}(j), 2);
                            msg_CN_2_VN{idx_vn}(end + 1) = tempxor;
                        end
                    end
                end
            end

            % Compare first kNumInfoBits
            decoded = VN(1:kNumInfoBits);
            errors = errors + sum(decoded ~= b');
        end

        plotvec(idx) = errors / (Nsim * kNumInfoBits);
        fprintf('Rate %.2f | Eb/No = %.1f dB | BER = %.5f\n', codeRate, jEb,
plotvec(idx));
    end

    semilogy(EbNodB, plotvec, 'LineWidth', 2);
end
```

```
Rate 0.25 | Eb/No = 0.0 dB | BER = 0.43596
Rate 0.25 | Eb/No = 0.5 dB | BER = 0.41231
Rate 0.25 | Eb/No = 1.0 dB | BER = 0.41327
Rate 0.25 | Eb/No = 1.5 dB | BER = 0.40288
Rate 0.25 | Eb/No = 2.0 dB | BER = 0.40000
Rate 0.25 | Eb/No = 2.5 dB | BER = 0.39135
Rate 0.25 | Eb/No = 3.0 dB | BER = 0.38173
Rate 0.25 | Eb/No = 3.5 dB | BER = 0.36942
Rate 0.25 | Eb/No = 4.0 dB | BER = 0.37115
Rate 0.25 | Eb/No = 4.5 dB | BER = 0.36885
Rate 0.25 | Eb/No = 5.0 dB | BER = 0.27885
Rate 0.25 | Eb/No = 5.5 dB | BER = 0.14596
Rate 0.25 | Eb/No = 6.0 dB | BER = 0.00058
Rate 0.25 | Eb/No = 6.5 dB | BER = 0.00096
Rate 0.25 | Eb/No = 7.0 dB | BER = 0.00077
Rate 0.25 | Eb/No = 7.5 dB | BER = 0.00000
Rate 0.25 | Eb/No = 8.0 dB | BER = 0.00000
Rate 0.25 | Eb/No = 8.5 dB | BER = 0.00000
Rate 0.25 | Eb/No = 9.0 dB | BER = 0.00000
Rate 0.25 | Eb/No = 9.5 dB | BER = 0.00000
Rate 0.25 | Eb/No = 10.0 dB | BER = 0.00000
Rate 0.33 | Eb/No = 0.0 dB | BER = 0.40173
Rate 0.33 | Eb/No = 0.5 dB | BER = 0.39038
Rate 0.33 | Eb/No = 1.0 dB | BER = 0.39519
Rate 0.33 | Eb/No = 1.5 dB | BER = 0.37865
Rate 0.33 | Eb/No = 2.0 dB | BER = 0.38231
```

```
Rate 0.33 | Eb/No = 2.5 dB | BER = 0.37077
Rate 0.33 | Eb/No = 3.0 dB | BER = 0.37577
Rate 0.33 | Eb/No = 3.5 dB | BER = 0.37077
Rate 0.33 | Eb/No = 4.0 dB | BER = 0.35846
Rate 0.33 | Eb/No = 4.5 dB | BER = 0.35923
Rate 0.33 | Eb/No = 5.0 dB | BER = 0.27288
Rate 0.33 | Eb/No = 5.5 dB | BER = 0.20115
Rate 0.33 | Eb/No = 6.0 dB | BER = 0.00019
Rate 0.33 | Eb/No = 6.5 dB | BER = 0.00019
Rate 0.33 | Eb/No = 7.0 dB | BER = 0.00019
Rate 0.33 | Eb/No = 7.5 dB | BER = 0.00019
Rate 0.33 | Eb/No = 8.0 dB | BER = 0.00000
Rate 0.33 | Eb/No = 8.5 dB | BER = 0.00000
Rate 0.33 | Eb/No = 9.0 dB | BER = 0.00000
Rate 0.33 | Eb/No = 9.5 dB | BER = 0.00000
Rate 0.33 | Eb/No = 10.0 dB | BER = 0.00000
Rate 0.50 | Eb/No = 0.0 dB | BER = 0.37288
Rate 0.50 | Eb/No = 0.5 dB | BER = 0.36788
Rate 0.50 | Eb/No = 1.0 dB | BER = 0.37635
Rate 0.50 | Eb/No = 1.5 dB | BER = 0.36635
Rate 0.50 | Eb/No = 2.0 dB | BER = 0.35058
Rate 0.50 | Eb/No = 2.5 dB | BER = 0.35173
Rate 0.50 | Eb/No = 3.0 dB | BER = 0.32635
Rate 0.50 | Eb/No = 3.5 dB | BER = 0.34712
Rate 0.50 | Eb/No = 4.0 dB | BER = 0.33212
Rate 0.50 | Eb/No = 4.5 dB | BER = 0.32250
Rate 0.50 | Eb/No = 5.0 dB | BER = 0.34038
Rate 0.50 | Eb/No = 5.5 dB | BER = 0.29481
Rate 0.50 | Eb/No = 6.0 dB | BER = 0.30154
Rate 0.50 | Eb/No = 6.5 dB | BER = 0.06769
Rate 0.50 | Eb/No = 7.0 dB | BER = 0.00058
Rate 0.50 | Eb/No = 7.5 dB | BER = 0.00000
Rate 0.50 | Eb/No = 8.0 dB | BER = 0.00000
Rate 0.50 | Eb/No = 8.5 dB | BER = 0.00000
Rate 0.50 | Eb/No = 9.0 dB | BER = 0.00000
Rate 0.50 | Eb/No = 9.5 dB | BER = 0.00000
Rate 0.50 | Eb/No = 10.0 dB | BER = 0.00000
Rate 0.60 | Eb/No = 0.0 dB | BER = 0.36154
Rate 0.60 | Eb/No = 0.5 dB | BER = 0.34904
Rate 0.60 | Eb/No = 1.0 dB | BER = 0.35615
Rate 0.60 | Eb/No = 1.5 dB | BER = 0.35038
Rate 0.60 | Eb/No = 2.0 dB | BER = 0.34654
Rate 0.60 | Eb/No = 2.5 dB | BER = 0.33058
Rate 0.60 | Eb/No = 3.0 dB | BER = 0.33904
Rate 0.60 | Eb/No = 3.5 dB | BER = 0.32250
Rate 0.60 | Eb/No = 4.0 dB | BER = 0.31212
Rate 0.60 | Eb/No = 4.5 dB | BER = 0.32288
Rate 0.60 | Eb/No = 5.0 dB | BER = 0.31904
Rate 0.60 | Eb/No = 5.5 dB | BER = 0.30385
Rate 0.60 | Eb/No = 6.0 dB | BER = 0.27981
Rate 0.60 | Eb/No = 6.5 dB | BER = 0.12442
Rate 0.60 | Eb/No = 7.0 dB | BER = 0.05827
Rate 0.60 | Eb/No = 7.5 dB | BER = 0.03231
Rate 0.60 | Eb/No = 8.0 dB | BER = 0.00000
Rate 0.60 | Eb/No = 8.5 dB | BER = 0.00000
Rate 0.60 | Eb/No = 9.0 dB | BER = 0.00000
Rate 0.60 | Eb/No = 9.5 dB | BER = 0.00000
Rate 0.60 | Eb/No = 10.0 dB | BER = 0.00000
```

```
legend('Rate 1/4', 'Rate 1/3', 'Rate 1/2', 'Rate 3/5', 'Location',
'SouthWest');
xlabel('Eb/No (dB)');
```

```
ylabel('Bit Error Rate (BER)');
title('5G NR LDPC Hard-Decision Decoding using Tanner Graph (Nsim=100)');
grid on;
```




```
function [B,H,z] = nrldpc_Hmatrix(BG)
    load(sprintf('%s.txt', BG), BG);
    B = eval(BG);
    [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
end

function cword = nrldpc_encode(B,z,msg)
```

```
    [m,n] = size(B);
    cword = zeros(1,n*z);
    cword(1:(n-m)*z) = msg;
    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

    p1_sh = B(2,n-m+1);
    if p1_sh == -1
        p1_sh = B(3,n-m+1);
    end
    cword((n-m)*z+1:(n-m+1)*z) = mul_sh(temp, z - p1_sh);

    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

    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
```