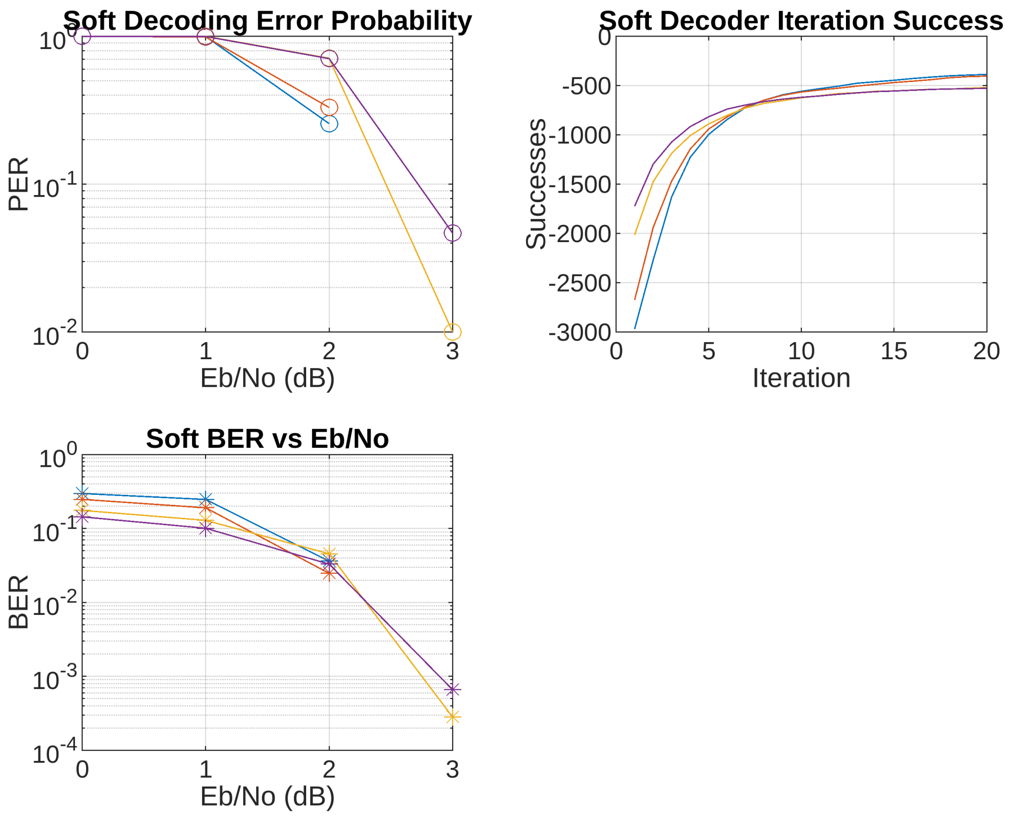
```
% Finalized LDPC 5G NR Soft Decoder with Functionality Test, Spot Checks,
and Final Graphs
clc; clear; close all;
tic;
% Base graphs
baseGraphs = {'NR_2_6_52', 'NR_1_5_352'};
codeRates_BG = {[1/4, 1/3, 1/2, 3/5], [1/3, 1/2, 3/5, 4/5]};
EbNodB_vec = 0:1:10;
Nsim = 300; maxIter = 20;

figure; tiledlayout(2,2);

for bgIdx = 1:2
    bg = baseGraphs{bgIdx};
    rates = codeRates_BG{bgIdx};
    load([bg, '.txt'], bg); B = eval(bg);
    [mb, nb] = size(B); kb = nb - mb; z = 52;

    for r = 1:length(rates)
        R = rates(r);
        k_pc = kb-2; nbRM = ceil(k_pc/R)+2;
        nBlock = nbRM*z; kBlock = kb*z;

        Hfull = gen_H(B,z); H = Hfull(:,1:nBlock);
        H = H(1:(mb*z - nb*z + nBlock), :);

        % Tanner graph representation
        v2c = get_v2c_map(H); c2v = get_c2v_map(H);

        % Metrics
        BER = zeros(1,length(EbNodB_vec)); PER = zeros(1,length(EbNodB_vec));
        iterSucc = Nsim * ones(1, maxIter);

        for snrIdx = 1:length(EbNodB_vec)
            EbNo = 10^(EbNodB_vec(snrIdx)/10);
            sigma2 = 1 / (2 * R * EbNo);

            bitErr = 0; frameErr = 0;
            for s = 1:Nsim
                msg = randi([0 1], kBlock, 1);
                code = ldpc_encode(B, z, msg');
                code = code(1:nBlock);
                tx = 1 - 2 * code;
                rx = tx + sqrt(sigma2) * randn(1, nBlock);

                llr_init = 2 * rx / sigma2;
                [decoded, iterSucc] = soft_decode(llr_init, H, v2c, c2v,
iterSucc, maxIter);

                e = mod(decoded(1:kBlock) + msg', 2);
                bitErr = bitErr + sum(e);
                frameErr = frameErr + any(e);
            end

            BER(snrIdx) = bitErr / (Nsim * kBlock);
            PER(snrIdx) = frameErr / Nsim;
        end

        nexttile(1); semilogy(EbNodB_vec, PER, '-o'); hold on; grid on;
        title('Soft Decoding Error Probability'); xlabel('Eb/No (dB)');
ylabel('PER');

        nexttile(2); plot(1:maxIter, iterSucc, '-'); hold on; grid on;
        title('Soft Decoder Iteration Success'); xlabel('Iteration');
ylabel('Successes');

        nexttile(3); semilogy(EbNodB_vec, BER, '-*'); hold on; grid on;
        title('Soft BER vs Eb/No'); xlabel('Eb/No (dB)'); ylabel('BER');
    end

end
```



```
Index exceeds the number of array elements. Index must not exceed 52.

Error in SoftDecoding>shift_block (line 115)
    if k==-1, y = zeros(1,length(x)); else, y = [x(k+1:end), x(1:k)]; end
```

```
Error in SoftDecoding>ldpc_encode (line 93)
            temp = mod(temp + shift_block(msg((j-1)*z+1:j*z), B(i,j)), 2);
```

```
function H = gen_H(B,z)
    [mb,nb] = size(B);
    H = zeros(mb*z, nb*z);
    Iz = eye(z); I0 = zeros(z);
    for i = 1:mb
        for j = 1:nb
            row = (i-1)*z + (1:z);
            col = (j-1)*z + (1:z);
            if B(i,j) == -1
                H(row,col) = I0;
            else
                H(row,col) = circshift(Iz, -B(i,j));
            end
        end
    end
end

function code = ldpc_encode(B,z,msg)
    [m,n] = size(B); code = zeros(1,n*z);
    code(1:(n-m)*z) = msg;
    temp = zeros(1,z);
    for i = 1:4
        for j = 1:n-m
            temp = mod(temp + shift_block(msg((j-1)*z+1:j*z), B(i,j)), 2);
        end
    end
    p1sh = B(2,n-m+1); if p1sh==-1, p1sh = B(3,n-m+1); end
    code((n-m)*z+1:(n-m+1)*z) = shift_block(temp, z-p1sh);
    for i = 1:3
        temp = zeros(1,z);
        for j = 1:n-m+i
            temp = mod(temp + shift_block(code((j-1)*z+1:j*z), B(i,j)), 2);
        end
        code((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 + shift_block(code((j-1)*z+1:j*z), B(i,j)), 2);
        end
        code((n-m+i-1)*z+1:(n-m+i)*z) = temp;
    end
end

function y = shift_block(x,k)
```

```
    if k==-1, y = zeros(1,length(x)); else, y = [x(k+1:end), x(1:k)]; end
end

function map = get_v2c_map(H)
    [~,c] = size(H); map = cell(c,1);
    for i = 1:c
        map{i} = find(H(:,i));
    end
end

function map = get_c2v_map(H)
    [r,~] = size(H); map = cell(r,1);
    for i = 1:r
        map{i} = find(H(i,:));
    end
end

function [dec_out, iterSucc] = hard_decode(rx, v2c, c2v, v2c_val, c2v_val,
iterSucc)
    dec_out = double(rx < 0); prev = dec_out;
    for it = 1:length(iterSucc)
        if it == 1
            for vn = 1:length(v2c)
                for cn = v2c{vn}'
                    v2c_val(cn,vn) = dec_out(vn);
                end
            end
        else
            for vn = 1:length(v2c)
                onesum = dec_out(vn) + sum(c2v_val(v2c{vn}, vn));
                for cn = v2c{vn}'
                    v2c_val(cn,vn) = (onesum - c2v_val(cn,vn)) >
(length(v2c{vn})/2);
                end
            end
        end
        for cn = 1:length(c2v)
            total = xor_all(v2c_val(cn, c2v{cn}));
            for vn = c2v{cn}
                c2v_val(cn,vn) = xor(total, v2c_val(cn,vn));
            end
        end
        for vn = 1:length(v2c)
            onesum = dec_out(vn) + sum(c2v_val(v2c{vn}, vn));
            dec_out(vn) = onesum > (length(v2c{vn})+1)/2;
        end
        if all(dec_out == prev), break; end
        prev = dec_out; iterSucc(it) = iterSucc(it) - 1;
    end
end
```

```
function [decoded, iterSucc] = soft_decode(llr_init, H, v2c, c2v, iterSucc,
maxIter)
    decoded = double(llr_init < 0);
    Lq = zeros(size(H));
    Lr = zeros(size(H));

    % Initialize Lq
    for vn = 1:length(v2c)
        for cn = v2c{vn}'
            Lq(cn,vn) = llr_init(vn);
        end
    end

    for it = 1:maxIter
        % CN update
        for cn = 1:length(c2v)
            for vn = c2v{cn}
                others = setdiff(c2v{cn}, vn);
                signs = sign(Lq(cn, others));
                mags = abs(Lq(cn, others));
                Lr(cn, vn) = prod(signs) * min(mags);
            end
        end

        % VN update + decision
        for vn = 1:length(v2c)
            total = llr_init(vn);
            for cn = v2c{vn}'
                total = total + Lr(cn,vn);
            end
            decoded(vn) = double(total < 0);
            for cn = v2c{vn}'
                Lq(cn,vn) = total - Lr(cn,vn);
            end
        end

        % Stop early
        if all(mod(H * decoded', 2) == 0), break; end
        iterSucc(it) = iterSucc(it) - 1;
    end
end
toe;
```