EE 6340/3861 Wireless Communications
InClass Assignment 2
Anshul Gupta | EE20BTECH11004

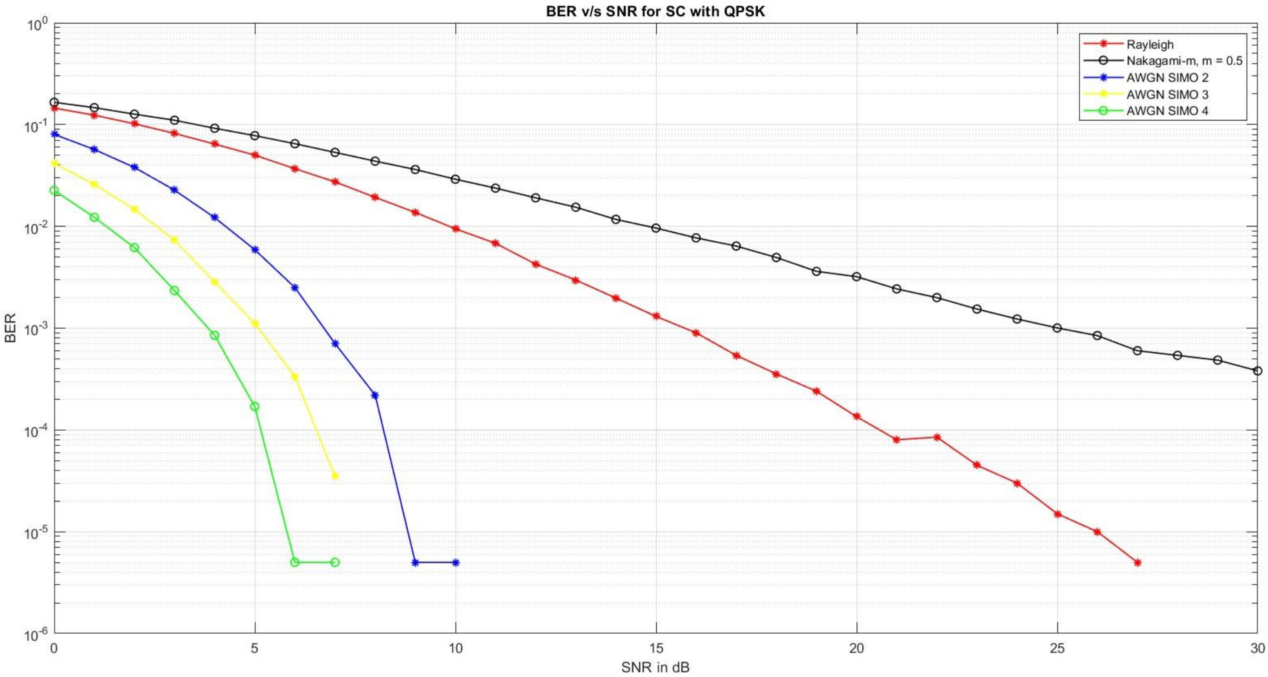
Part 1: Selection Combining for Rayleigh and Nakagami – m fading with QPSK along with SIMO AWGN for 2, 3 and 4 receivers.

```
clc;
close all;
N = 2*1e5;
SNR_dB = 0:30;
SNR = 10.^(SNR_dB/10);

ber_rayleigh = qpsk(N, SNR, 1, 'rayleigh');
ber_nakagami = qpsk(N, SNR, 1, 'nakagami');
ber_awgn = qpsk(N, SNR, 1, 'awgn');
ber_awgn_3 = qpsk(N, SNR, 1, 'awgn3');
ber_awgn_4 = qpsk(N, SNR, 1, 'awgn4');

% Plotting
semilogy(SNR_dB, ber_rayleigh,'r*-', 'linewidth', 1);
hold on;
semilogy(SNR_dB, ber_nakagami,'ko-', 'linewidth', 1);
hold on;
semilogy(SNR_dB, ber_awgn,'b*-', 'linewidth', 1);
hold on;
semilogy(SNR_dB, ber_awgn_3,'y*-', 'linewidth', 1);
hold on;
semilogy(SNR_dB, ber_awgn_4,'go-', 'linewidth', 1);
hold off;

legend('Rayleigh', 'Nakagami-m, m = 0.5', 'AWGN SIMO 2', 'AWGN SIMO 3', 'AWGN SIMO
4', 'Location', 'northeast');
xlabel('SNR in dB')
ylabel('BER')
title('BER v/s SNR for SC with QPSK')
grid on;

% QPSK function with selection combing for two receivers
function [BER_sc] = qpsk(N, SNR, Eb, fading_type)
    BER_sc = zeros(1, length(SNR));
    data = randi(2, 1, N) - 1;
    sybmols = sqrt(Eb/2)*[1+1i, 1-1i, -1+1i, -1-1i];
    input = zeros(1, N/2);
    j=1;
    for i=1:2:N
        input(j) = sybmols(2*data(i) + data(i+1) + 1);
        j = j + 1;
    end

    % Loop through SNR values
    for k = 1:length(SNR)
        %% Generate channel gains according to the selected fading type
        if strcmp(fading_type, 'awgn') || strcmp(fading_type, 'awgn3') ||
strcmp(fading_type, 'awgn4')
            h1 = ones(1, N/2);
            h2 = ones(1, N/2);
```

```matlab
        elseif strcmp(fading_type, 'rayleigh')
            h1 = (randn(1, N/2) + 1i*randn(1, N/2))/sqrt(2);
            h2 = (randn(1, N/2) + 1i*randn(1, N/2))/sqrt(2);
        elseif strcmp(fading_type, 'nakagami')
            m_nakagami = 0.5;
            nakagami_pdf = makedist('Nakagami','mu',m_nakagami,'omega',1);
            h1 = random(nakagami_pdf,1, N/2) .* exp(1i*2*pi*rand(1, N/2));
            h2 = random(nakagami_pdf,1, N/2) .* exp(1i*2*pi*rand(1, N/2));
        else
            error('Invalid fading type');
        end

        %% Selection combining
        h_max = max(h1, h2);

        % y = h_max*x + n
        if strcmp(fading_type, 'awgn')
            r_sc1 = input + cgn(0, 0, SNR(k), N/2);
            r_sc2 = input + cgn(0, 0, SNR(k), N/2);
            recieved = (r_sc1 + r_sc2)/2;
        elseif strcmp(fading_type, 'awgn3')
            r_sc1 = input + cgn(0, 0, SNR(k), N/2);
            r_sc2 = input + cgn(0, 0, SNR(k), N/2);
            r_sc3 = input + cgn(0, 0, SNR(k), N/2);
            recieved = (r_sc1 + r_sc2 + r_sc3)/3;
        elseif strcmp(fading_type, 'awgn4')
            r_sc1 = input + cgn(0, 0, SNR(k), N/2);
            r_sc2 = input + cgn(0, 0, SNR(k), N/2);
            r_sc3 = input + cgn(0, 0, SNR(k), N/2);
            r_sc4 = input + cgn(0, 0, SNR(k), N/2);
            recieved = (r_sc1 + r_sc2 + r_sc3 + r_sc4)/4;
        else
            recieved = abs(h_max).*(input) + cgn(0, 0, SNR(k), N/2);
            % equalization in case of channel fading
            recieved = recieved ./ abs(h_max);
        end

        %% decoding
        [~, detected_bits_sc] = demap(recieved, sybmols);

        %% computing BER
        BER_sc(k) = sum(detected_bits_sc ~= data)/N;
    end
end

function [detected_symbols, detected_bits] = demap(recieved, symbols)
    len = length(recieved);
    detected_symbols = zeros(1, len);
    detected_index = zeros(1, len);
    for i = 1 : len
        [~, index] = min(abs(symbols - recieved(i)));
        detected_symbols(i) = symbols(index);
        detected_index(i) = index;
    end

    detected_index = detected_index - 1;
    detected_bits = zeros (1, 2*len);
    for i = 1:len
        if detected_index(i) == 3
```

```
            detected_bits(2*i) = 1;
            detected_bits(2*i - 1) = 1;

        elseif detected_index(i) == 2
            detected_bits(2*i) = 0;
            detected_bits(2*i - 1) = 1;

        elseif detected_index(i) == 1
            detected_bits(2*i) = 1;
            detected_bits(2*i - 1) = 0;

        else
            detected_bits(2*i) = 0;
            detected_bits(2*i - 1) = 0;
        end
    end
end

function cn = cgn(real_mean, imag_mean, snr, N)
    AWGN_mean = complex(real_mean, imag_mean);
    AWGN_sigma = 1/sqrt(2*snr);
    cn = AWGN_mean + AWGN_sigma*complex(randn([1, N]), randn([1, N]));
end
```

Figure 1:



Observations:

1. The BER for Rayleigh and Nakagami-m for $m = 1$ overlap. For $m < 1$, Nakagami-m lies above raleigh and for $m > 1$ lies below rayleigh.
2. If the variance is σ^2, for N recievers the variance is: $\frac{\sigma^2}{2N}$. So, as the number of reiceviers increase the varinace decreases and hence the BER goes down. Hence for AWGN with 2, 3 and 4 recievers the BER goes down as the number of recievers increase.
3. The BER with channel fading is more than AWGN. This shows the adverse effect of channel fading on the BER.
4. We can increase the number of recivers to counter the effect of fading to some extent.

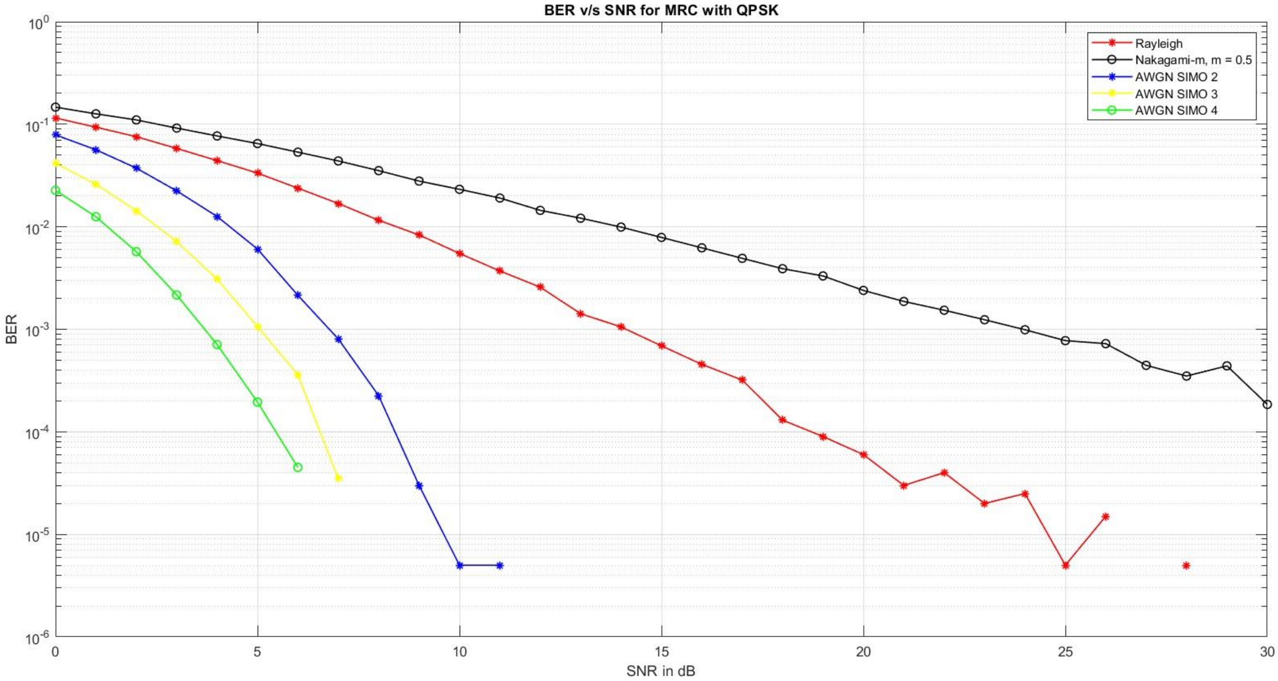
Part 2: Maximal Ratio Combining for Rayleigh and Nakagami – m fading with QPSK along with SIMO AWGN for 2, 3 and 4 receivers.

```
clc;
N = 2*1e5;
SNR_dB = 0:30;
SNR = 10.^(SNR_dB/10);

ber_rayleigh = qpsk(N, SNR, 1, 'rayleigh');
ber_nakagami = qpsk(N, SNR, 1, 'nakagami');
ber_awgn = qpsk(N, SNR, 1, 'awgn');
ber_awgn_3 = qpsk(N, SNR, 1, 'awgn3');
ber_awgn_4 = qpsk(N, SNR, 1, 'awgn4');

% plotting
semilogy(SNR_dB, ber_rayleigh,'r*-', 'linewidth', 1);
hold on;
semilogy(SNR_dB, ber_nakagami,'ko-', 'linewidth', 1);
hold on;
semilogy(SNR_dB, ber_awgn,'b*-', 'linewidth', 1);
hold on;
semilogy(SNR_dB, ber_awgn_3,'y*-', 'linewidth', 1);
hold on;
semilogy(SNR_dB, ber_awgn_4,'go-', 'linewidth', 1);
hold off;

legend('Rayleigh', 'Nakagami-m, m = 0.5', 'AWGN SIMO 2', 'AWGN SIMO 3', 'AWGN SIMO
4', 'Location', 'northeast');
xlabel('SNR in dB')
ylabel('BER')
title('BER v/s SNR for MRC with QPSK')
grid on;

% QPSK function with selection combing for two receivers
function [BER_sc] = qpsk(N, SNR, Eb, fading_type)
    BER_sc = zeros(1, length(SNR));

    %% Bit generation
    data = randi(2, 1, N) - 1;

    %% mapping to symbols
    symbols = sqrt(Eb/2)*[1+1i, 1-1i, -1+1i, -1-1i];
    input = zeros(1, N/2);
    j=1;
    for i=1:2:N
        input(j) = symbols(2*data(i) + data(i+1) + 1);
        j = j + 1;
    end

    % Loop through SNR values
    for k = 1:length(SNR)
        %% Generate channel gains according to the input fading type
        % Since we are working with two recievers we generate two channel
coefficients.
        if strcmp(fading_type, 'awgn') || strcmp(fading_type, 'awgn3') ||
strcmp(fading_type, 'awgn4')
            h1 = ones(1, N/2);
            h2 = ones(1, N/2);
```

```
        elseif strcmp(fading_type, 'rayleigh')
            h1 = (randn(1, N/2) + 1i*randn(1, N/2))/sqrt(2);
            h2 = (randn(1, N/2) + 1i*randn(1, N/2))/sqrt(2);
        elseif strcmp(fading_type, 'nakagami')
            m_nakagami = 0.5;
            nakagami_pdf = makedist('Nakagami','mu',m_nakagami,'omega',1);
            h1 = random(nakagami_pdf,1, N/2) .* exp(1i*2*pi*rand(1, N/2));
            h2 = random(nakagami_pdf,1, N/2) .* exp(1i*2*pi*rand(1, N/2));
        else
            error('Invalid fading type');
        end

        %% MRC
        h_channel = abs(h1).^2 + abs(h2).^2;
        noise = conj(h1).*cgn(0,0,SNR(k), N/2) + conj(h2).*cgn(0,0,SNR(k), N/2);

        if strcmp(fading_type, 'awgn')
            r_1 = input + cgn(0, 0, SNR(k), N/2);
            r_2 = input + cgn(0, 0, SNR(k), N/2);
            recieved = (r_1 + r_2)/2;
        elseif strcmp(fading_type, 'awgn3')
            r_1 = input + cgn(0, 0, SNR(k), N/2);
            r_2 = input + cgn(0, 0, SNR(k), N/2);
            r_3 = input + cgn(0, 0, SNR(k), N/2);
            recieved = (r_1 + r_2 + r_3)/3;
        elseif strcmp(fading_type, 'awgn4')
            r_1 = input + cgn(0, 0, SNR(k), N/2);
            r_2 = input + cgn(0, 0, SNR(k), N/2);
            r_3 = input + cgn(0, 0, SNR(k), N/2);
            r_4 = input + cgn(0, 0, SNR(k), N/2);
            recieved = (r_1 + r_2 + r_3 + r_4)/4;
        else
            recieved = h_channel.*(input) + noise;
            % equalization in case of fading channel
            recieved = recieved ./abs(h_channel);
        end

        %% decision
        [~, detected_bits_sc] = demap(recieved, symbols);

        %% computing BER
        BER_sc(k) = sum(detected_bits_sc ~= data)/N;
    end
end

function [detected_symbols, detected_bits] = demap(recieved, symbols)
    len = length(recieved);
    detected_symbols = zeros(1, len);
    detected_index = zeros(1, len);
    for i = 1 : len
        [~, index] = min(abs(symbols - recieved(i)));
        detected_symbols(i) = symbols(index);
        detected_index(i) = index;
    end

    detected_index = detected_index - 1;
    detected_bits = zeros (1, 2*len);
    for i = 1:len
        if detected_index(i) == 3
```

```
            detected_bits(2*i) = 1;
            detected_bits(2*i - 1) = 1;

        elseif detected_index(i) == 2
            detected_bits(2*i) = 0;
            detected_bits(2*i - 1) = 1;

        elseif detected_index(i) == 1
            detected_bits(2*i) = 1;
            detected_bits(2*i - 1) = 0;

        else
            detected_bits(2*i) = 0;
            detected_bits(2*i - 1) = 0;
        end
    end
end

% function to generate complex gaussian noise
function cn = cgn(real_mean, imag_mean, snr, N)
    AWGN_mean = complex(real_mean, imag_mean);
    AWGN_sigma = 1/sqrt(2*snr);
    cn = AWGN_mean + AWGN_sigma*complex(randn([1, N]), randn([1, N]));
end
```

Figure 2:



Observations:

1. The graphs follow similar trends as observed for Selection combining.
2. The curves for MRC are shifted below as compared to SC. Therefore we can say that MRC is *better* than SC. The reason MRC is better than SC because it takes a weighted sum of all the channels whereas SC selects the best of all the channels.