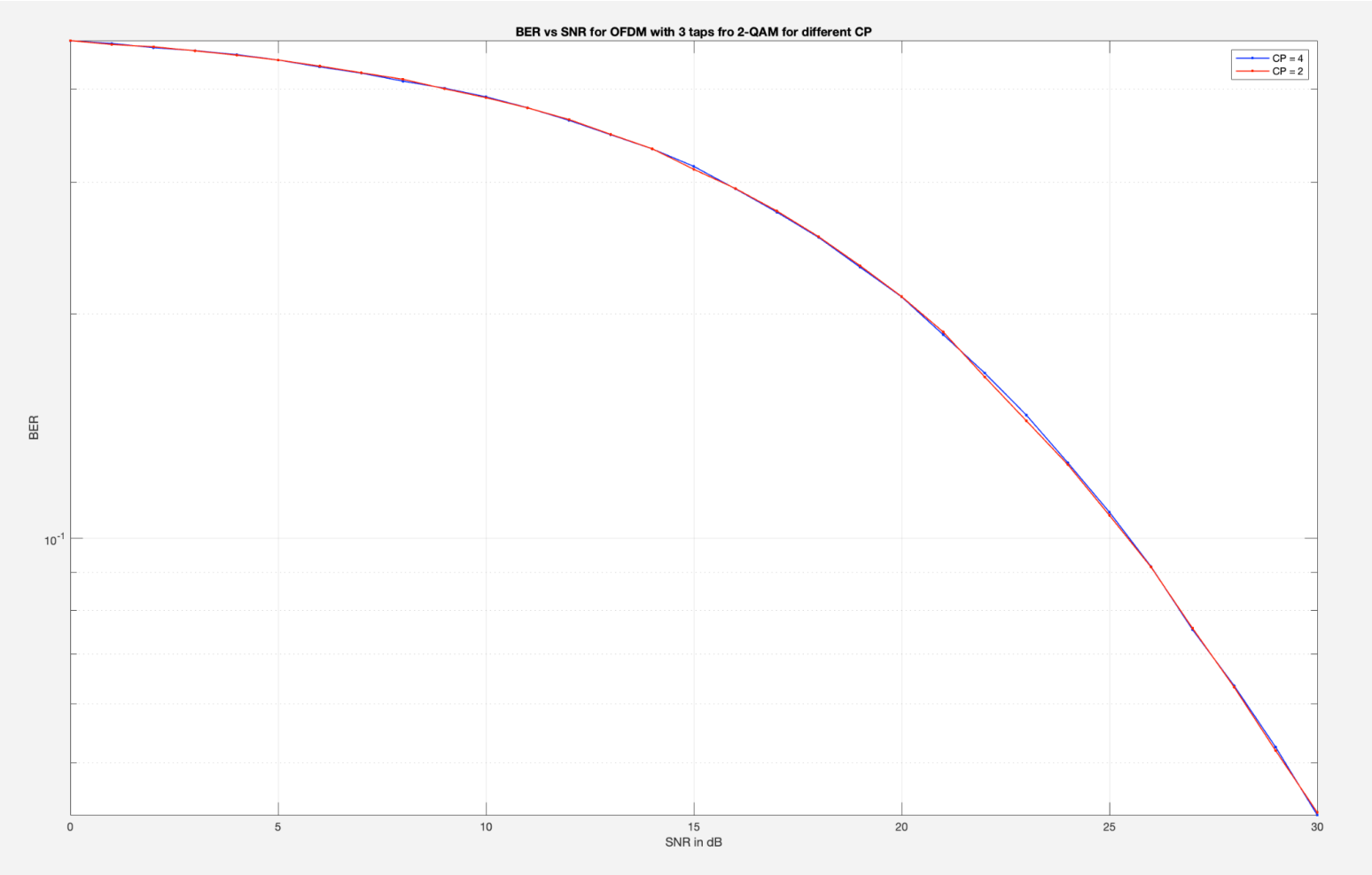
In-Class Assignment 3

Code-

```
clc;
clear all;
close all;

% Step - 1
M = 2;
N = 64;

SNR_in_dB = 0 : 1 : 30;
SNR = 10 .^ (SNR_in_dB / 10);

BER_1 = zeros(1, length(SNR));
BER_2 = zeros(1, length(SNR));

for i_snr = 1 : length(SNR)

    bit_errs = 0;

    for rep = 1 : 10000
        % Step - 2
        len = N * log2(M);
        data = randi(2, 1, len) - 1;

        % Step - 3
        E_b = 2;
        symbols = (1 - 2 * data) / sqrt(2);

        % Step - 4
        ifft_symbols = ifft(symbols);

        % Step - 5
        cyclic_prefix = [ifft_symbols(len - 3 : len), ifft_symbols];
        len_cyclic_prefix = length(cyclic_prefix);

        % Step - 6
        L = 3;
        tap_power = [0.3, 0.8, 0.2];
        h = sqrt(tap_power / 2) .* (randn(1, L) + 1j * randn(1, L));

        % Step - 7
        circular_conv = cconv(cyclic_prefix, h, len_cyclic_prefix);
        received = circular_conv + (1 /  sqrt(SNR(i_snr))) * (randn(1, len_cyclic_prefix) + 1j * randn(1, len_cyclic_prefix));

        % Step - 8
        removed_cp = received( : , 5 : len_cyclic_prefix);

        % Step - 9
        fft_removed_cp = fft(removed_cp);

        % Step - 10
        fft_h = fft(h, len);
        received_eq = fft_removed_cp ./ fft_h;
        predicted_bits = zeros(1, len);
        for i = 1 : len

            if received_eq(i) >= 0

                predicted_bits(i) = 0;

            else

                predicted_bits(i) = 1;

            end

        end
        bit_errors = sum(abs(data - predicted_bits));
        bit_errs = bit_errs + bit_errors;
    end
```

```
    BER_1(i_snr) = bit_errs / 640000;
end

for i_snr = 1 : length(SNR)

    bit_errs = 0;

    for rep = 1 : 10000
        % Step - 2
        len = N * log2(M);
        data = randi(2, 1, len) - 1;

        % Step - 3
        E_b = 2;
        symbols = (1 - 2 * data) / sqrt(2);

        % Step - 4
        ifft_symbols = ifft(symbols);

        % Step - 5
        cyclic_prefix = [ifft_symbols(len - 1 : len), ifft_symbols];
        len_cyclic_prefix = length(cyclic_prefix);

        % Step - 6
        L = 3;
        tap_power = [0.3, 0.8, 0.2];
        h = sqrt(tap_power / 2) .* (randn(1, L) + 1j * randn(1, L));

        % Step - 7
        circular_conv = cconv(cyclic_prefix, h, len_cyclic_prefix);
        received = circular_conv + (1 /  sqrt(SNR(i_snr))) * (randn(1, len_cyclic_prefix) + 1j * randn(1, len_cyclic_prefix));

        % Step - 8
        removed_cp = received( : , 3 : len_cyclic_prefix);

        % Step - 9
        fft_removed_cp = fft(removed_cp);

        % Step - 10
        fft_h = fft(h, len);
        received_eq = fft_removed_cp ./ fft_h;
        predicted_bits = zeros(1, len);
        for i = 1 : len

            if received_eq(i) >= 0

                predicted_bits(i) = 0;

            else

                predicted_bits(i) = 1;

            end

        end
        bit_errors = sum(abs(data - predicted_bits));
        bit_errs = bit_errs + bit_errors;
    end
    BER_2(i_snr) = bit_errs / 640000;
end

semilogy(SNR_in_dB, BER_1, 'b.-', 'linewidth', 1);
hold on;
semilogy(SNR_in_dB, BER_2, 'r.-', 'linewidth', 1);
hold off;

legend('CP = 4', 'CP = 2');
xlabel('SNR in dB');
ylabel('BER');
title('BER vs SNR for OFDM with 3 taps fro 2-QAM for different CP');
grid on;
```

Plot-


BER vs SNR for OFDM with 3 taps fro 2-QAM for different CP
CP = 4
CP = 2
BER
10^{-1}
0
5
10
15
20
25
30
SNR in dB