Code for BER of BPSK modulation-

```
clc;
clear all;
close all;

% Generating random binary data of length 100000
len = 100000;
data = randi(2, 1, len) - 1;

% Mapping the data to BPSK signal constellation
% So here 0 is mapped to sqrt(E_b) and 1 is mapped to -sqrt(E_b)
E_b = 1;
symbols = sqrt(E_b) * (1 - 2 * data);

% Multiplying the random fading coefficient and adding noise to it
SNR_in_dB = -5 : 1 : 10;
SNR = 10 .^ (SNR_in_dB / 10);

BER_rayleigh = zeros(1, length(SNR_in_dB));
BER_rician = zeros(1, length(SNR_in_dB));
BER_nakagami = zeros(1, length(SNR_in_dB));
BER_AWGN = zeros(1, length(SNR_in_dB));

for i = 1 : length(SNR_in_dB)
    gauss1 = randn([1, len]);
    gauss2 = randn([1, len]);

    % Multiplying the Random Fading Coefficient
    % For rayleigh fading-
    h1 = abs(complex(gauss1, gauss2)) / sqrt(2);
    rayleigh_faded_sym = h1 .* symbols;

    % For rician fading-
    mean = 1;
    h2 = abs(complex(mean + gauss1, gauss2)) / sqrt(2);
    rician_faded_sym = h2 .* symbols;
```

```
    % For Nakagami-m fading
    m = 2;
    h3 = abs(sqrt(m*abs(complex(gauss1,
gauss2)))./sqrt(sum(abs(complex(gauss1,
gauss2)).^2)/len).*complex(gauss1, gauss2));
    nakagami_faded_sym = h3 .* symbols;

    % No fading
    awgn_sym = symbols;



    % Adding AWGN noise to signal
    AWGN_mean = complex(0, 0);
    AWGN_sigma = 1 / sqrt(2 * SNR(i));
    noise = AWGN_mean + AWGN_sigma * complex(randn([1, len]),
randn([1, len]));

    rayleigh_noisy = rayleigh_faded_sym + noise;
    rician_noisy = rician_faded_sym + noise;
    nakagami_noisy = nakagami_faded_sym + noise;
    awgn_noisy = awgn_sym + noise;

    % Detecting symbols using ML rule
    rayleigh_detected_sym = ML_detector_BPSK(rayleigh_noisy ./ h1,
E_b);
    rician_detected_sym = ML_detector_BPSK(rician_noisy ./ h2, E_b);
    nakagami_detected_sym = ML_detector_BPSK(nakagami_noisy ./ h3,
E_b);
    awgn_detected_sym = ML_detector_BPSK(awgn_noisy, E_b);

    % Detecting bits
    rayleigh_bits = (1 - (rayleigh_detected_sym / sqrt(E_b))) / 2;
    rician_bits = (1 - (rician_detected_sym / sqrt(E_b))) / 2;
    nakagami_bits = (1 - (nakagami_detected_sym / sqrt(E_b))) / 2;
    awgn_bits = (1 - (awgn_detected_sym / sqrt(E_b))) / 2;

    % BER calculation
```

```
    BER_rayleigh(i) = calculate_BER(rayleigh_bits, data);
    BER_rician(i) = calculate_BER(rician_bits, data);
    BER_nakagami(i) = calculate_BER(nakagami_bits, data);
    BER_AWGN(i) = calculate_BER(awgn_bits, data);

end

semilogy(SNR_in_dB, BER_rayleigh, 'b.-', 'linewidth', 1);
hold on;
semilogy(SNR_in_dB, BER_rician,'g.-','linewidth',1);
hold on;
semilogy(SNR_in_dB, BER_nakagami,'r.-','linewidth',1);
hold on;
semilogy(SNR_in_dB, BER_AWGN,'k.-','linewidth',1);
hold on;

title('BER in BPSK modulation');
xlabel('SNR (in dB)');
ylabel('BER');
legend("BER of Rayleigh", "BER of Rician", "BER of Nakagami-m", "BER
with AWGN");

function [detected_symbols] = ML_detector_BPSK(received_symbols, E_b)
    len = length(received_symbols);
    detected_symbols = zeros([1, len]);
    for i = 1 : len
        if real(received_symbols(i)) >= 0
            detected_symbols(i) = sqrt(E_b);
        else
            detected_symbols(i) = -1 * sqrt(E_b);
        end
    end
end

function [BER] = calculate_BER(received_bits, transmitted_bits)

    len = length(transmitted_bits);
```

```
    BER = sum(abs((received_bits - transmitted_bits))) / len;

end
```

Obtained Plot-