**Tittle: Simulate BER Performance over Rayleigh Fading wireless channel with BPSK Transmission**

**Program:**

clear

N = 10^6 % number of bits or symbols

rand('state',100); % initializing the rand() function

randn('state',200); % initializing the randn() function

% Transmitter

ip = rand(1,N)>0.5; % generating 0,1 with equal probability

s = 2\*ip-1; % BPSK modulation 0 -> -1; 1 -> 1

n = 1/sqrt(2)\*[randn(1,N) + j\*randn(1,N)]; % white gaussian noise, 0dB variance

Eb\_N0\_dB = [-3:10]; % multiple Eb/N0 values

for ii = 1:length(Eb\_N0\_dB)

% Noise addition

y = s + 10^(-Eb\_N0\_dB(ii)/20)\*n; % additive white gaussian noise

% receiver - hard decision decoding

ipHat = real(y)>0;

% counting the errors

nErr(ii) = size(find([ip- ipHat]),2);

end

simBer = nErr/N; % simulated ber

theoryBer = 0.5\*erfc(sqrt(10.^(Eb\_N0\_dB/10))); % theoretical ber

% plot

close all

figure

semilogy(Eb\_N0\_dB,theoryBer,'b.-');

hold on

semilogy(Eb\_N0\_dB,simBer,'mx-');

axis([-3 10 10^-5 0.5])

grid on

legend('theory', 'simulation');

xlabel('Eb/No, dB');

ylabel('Bit Error Rate');

title('Bit error probability curve for BPSK modulation');

1. Set the number of bits or symbols (**N**) to 10^6.
2. Initialize the random number generator (**rand()** and **randn()**) states to specific values using **rand('state', 100)** and **randn('state', 200)**.
3. Transmitter section:
   * Generate a random binary sequence of length **N** with equal probability of 0 and 1 using **ip = rand(1, N) > 0.5**.
   * Perform BPSK modulation, mapping 0 to -1 and 1 to 1, by multiplying the binary sequence by 2 and subtracting 1: **s = 2 \* ip - 1**.
4. Add AWGN to the transmitted signal:
   * Generate complex Gaussian noise with zero mean and unit variance using **randn(1, N) + j \* randn(1, N)**. The factor **1/sqrt(2)** scales the noise to have a variance of 0.5.
   * Multiply the noise by **10^(-Eb\_N0\_dB(ii)/20)** to adjust its power based on the desired signal-to-noise ratio (**Eb/N0**) values. This represents the AWGN channel.
   * Add the noise to the transmitted signal: **y = s + 10^(-Eb\_N0\_dB(ii)/20) \* n**.
5. Receiver section:
   * Perform hard decision decoding by thresholding the received signal. The real part of **y** is compared to zero to estimate the transmitted symbols: **ipHat = real(y) > 0**.
6. Calculate the number of bit errors (**nErr**) by comparing the estimated symbols (**ipHat**) to the original transmitted symbols (**ip**).
7. Repeat the simulation for multiple **Eb/N0** values defined in the range **[-3:10]**.
8. Calculate the simulated bit error rate (**simBer**) by dividing the number of bit errors (**nErr**) by the total number of symbols (**N**).
9. Calculate the theoretical bit error rate (**theoryBer**) using the formula **0.5 \* erfc(sqrt(10.^(Eb\_N0\_dB/10)))**, where **erfc()** is the complementary error function.
10. Plot the theoretical and simulated bit error rate curves using **semilogy()** to create a logarithmic scale plot.
11. Set the plot limits, add grid lines, legends, and axis labels.
12. Display the bit error probability curve for the BPSK modulation scheme.

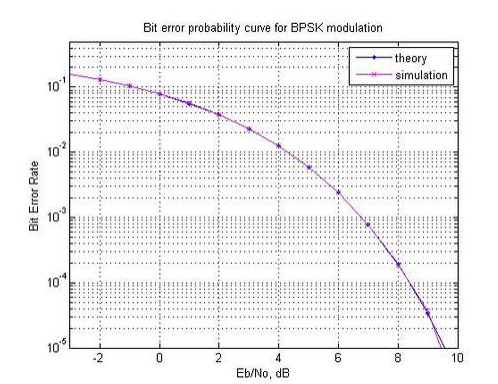
The code allows you to observe how the bit error rate changes with different **Eb/N0** values, representing different levels of signal-to-noise ratio. The simulated results (**simBer**) are compared to the theoretical results (**theoryBer**) to validate the simulation.

Note: The code is written in MATLAB or Octave syntax.

Top of Form

Regenerate response

Output:

Bottom of Form