

Southern Polytechnic College of

Engineering and Engineering Technology

Module 6 Report

EE 4490 - Wireless Communications

Alamouti Code and STBC

Rafael Rosales

Jared Rawlins

Matthew Stotter

**Objective**

1. OFDM implementation with a cyclic prefix (amount of cyclic prefix being a multiple of your group number). Generate the signal, transmit it and recover it t the receiver.

**Method/Results**

1. The goal with the following code was to encode a string message (“potato” in this case), modulate it with a QPSK modulator and an OFDM modulator, add noise, then demodulate it, and use the decoder we created in module 5 to output the original string.
   1. Matlab code:

% OFDM implementation with a cyclic prefix ( amount of cyclic prefix being a multiple of your group number).

% Generate the signal, transmit it, and recover it at the receiver.

clc;

close all;

clear all;

M = 4; % Modulation alphabet

bitsPerSymbol = log2(M); % Bits/symbol

numSubCarriers = 128; % Number of OFDM subcarriers

cyclicPrefixLength = 32; % OFDM cyclic prefix length

maxBitErrors = 100; % Maximum number of bit errors

maxNumBits = 1e7; % Maximum number of bits transmitted

in\_message\_string = 'potato';

in\_message\_binary = reshape((dec2bin(in\_message\_string) - 48).',[],1);

zeroarray = zeros(192, 1);

dataIn = [in\_message\_binary.' zeroarray.'].';

% Set the QPSK modulator and demodulator so that they accept binary inputs.

qpskModulator = comm.QPSKModulator('BitInput',true);

qpskDemodulator = comm.QPSKDemodulator('BitOutput',true);

% Set the OFDM modulator and demodulator pair according to the simulation parameters.

ofdmModulator = comm.OFDMModulator('FFTLength',numSubCarriers,'CyclicPrefixLength',cyclicPrefixLength);

ofdmDemodulator = comm.OFDMDemodulator('FFTLength',numSubCarriers,'CyclicPrefixLength',cyclicPrefixLength);

% Set the NoiseMethod property of the AWGN channel object

% to Variance and define the VarianceSource property

% so that the noise power can be set from an input port.

channel = comm.AWGNChannel('NoiseMethod','Variance', ...

'VarianceSource','Input port');

% Set the ResetInputPort property to true

% to enable the error rate calculator to be reset during the simulation.

errorRate = comm.ErrorRate('ResetInputPort',true);

% Use the info function of the ofdmModulator object

% to determine the input and output dimensions of the OFDM modulator.

ofdmDimensions = info(ofdmModulator)

% Determine the number of data subcarriers from the ofdmDimensions structure variable.

numDC = ofdmDimensions.DataInputSize(1)

% Determine the OFDM frame size (in bits)

% from the number of data subcarriers and the number of bits per symbol.

frameSize = [bitsPerSymbol\*numDC 1]

% set the size of the SNR vector based on the parameters and

% the size of the EbNoVector

EbNoVector = (0:10)';

snrVector = EbNoVector + 10\*log10(bitsPerSymbol) + 10\*log10(numDC/numSubCarriers);

% Initialize the BER

berVector = zeros(length(EbNoVector),3);

errorStats = zeros(1,3);

% Simulate the communication link over the range of Eb/No values.

% For each Eb/No value, the simulation runs until either maxBitErrors are recorded

% OR the total number of transmitted bits exceeds maxNumBits.

for m = 1:length(EbNoVector)

SNR = snrVector(m);

while errorStats(2) <= maxBitErrors && errorStats(3) <= maxNumBits

%dataIn = randi([0,1],frameSize); % Generate binary data

txQPSK = qpskModulator(dataIn); % Apply QPSK modulation

txSignal = ofdmModulator(txQPSK); % Apply OFDM modulation

powerdB = 10\*log10(var(txSignal)); % Calculate Tx signal power

noiseVariance = 10.^(0.1\*(powerdB-SNR));

% Calculate the noise variance

rxSignal = channel(txSignal,noiseVariance);

% Pass the signal through a noisy channel

rxQPSK = ofdmDemodulator(rxSignal); % Apply OFDM demodulation

dataOut = qpskDemodulator(rxQPSK); % Apply QPSK demodulation

errorStats = errorRate(dataIn,dataOut,0);% Collect error statistics

end

berVector(m,:) = errorStats; % Save BER data

errorStats = errorRate(dataIn,dataOut,1);

% Reset the error rate calculator

end

% theoreticalBER = berfading(EbNoVector,'psk',M,1);

theoreticalBER = berawgn(EbNoVector,'psk',M,'nondiff');

EbNo = (0:length(dataIn))';

snrVect = EbNo + 10\*log10(bitsPerSymbol) + 10\*log10(numDC/numSubCarriers);

for index = length(dataIn)

release(ofdmModulator)

SNR = snrVect(index);

qpskTx = qpskModulator(dataIn); % Apply QPSK modulation

signalTx = ofdmModulator(qpskTx); % Apply OFDM modulation

powerdB = 10\*log10(var(txSignal)); % Calculate Tx signal power

noiseVariance = 10.^(0.1\*(powerdB-SNR)); % Calculate the noise variance

signalRx = channel(txSignal,noiseVariance); % Pass the signal through a noisy channel

qpskRx = ofdmDemodulator(rxSignal); % Apply OFDM demodulation

out = qpskDemodulator(rxQPSK); % Apply QPSK demodulation

End

out\_message\_binary = out(1:42,1);

out\_message\_string = char(bin2dec(char(reshape(out\_message\_binary,7,[]).' + 48))).';

display(out\_message\_string);

% Plot the data

figure(2)

semilogy(EbNoVector,berVector(:,1),'\*')

hold on

semilogy(EbNoVector,theoreticalBER)

legend('Simulation','Theoretical','Location','Best')

xlabel('Eb/No (dB)')

ylabel('Bit Error Rate')

grid on

hold off

* 1. Utilizing the OFDM modulator from the Matlab communications toolbox proved to be a bit difficult. The transmission of 234 bits seemed to be required, so we added 192 zeros to the end of ‘potato’, which is 42 bits long. Accompanying the decoder was a test which calculated the bit error rate of the same transmission over various signal to noise ratios. We did this for a cyclic prefix of 6, 12, 24, and 60 bits. This yielded the following results:

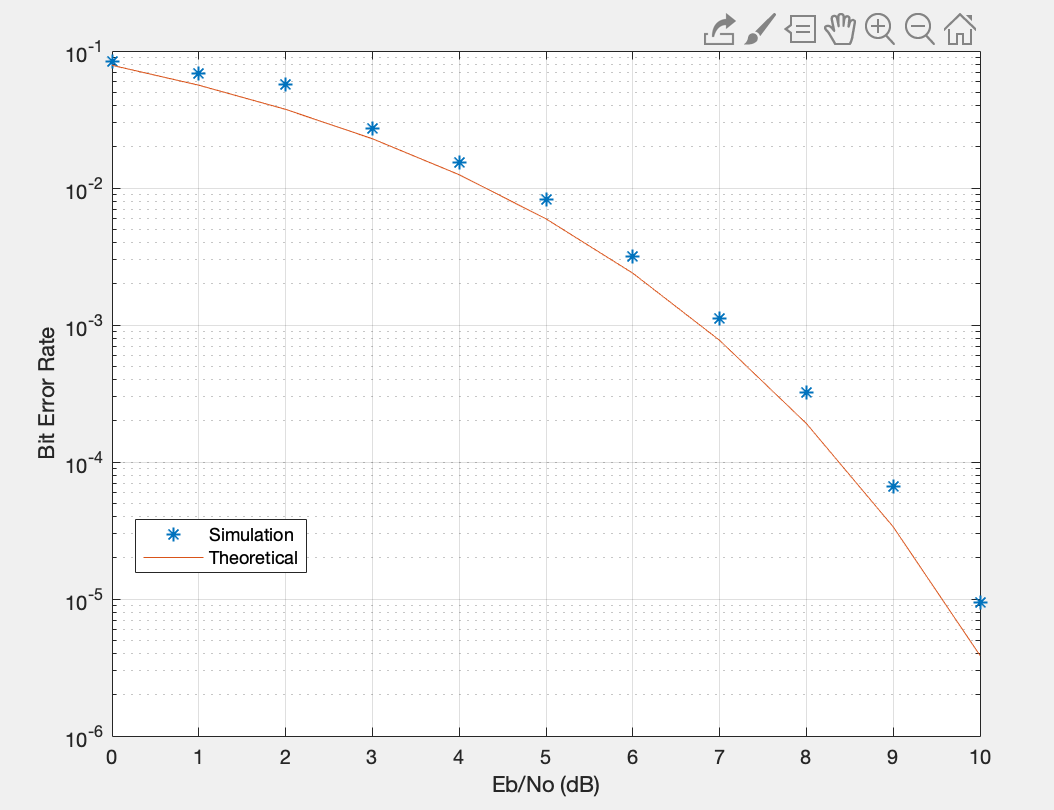


Figure 1. BER for cyclic prefix of 6

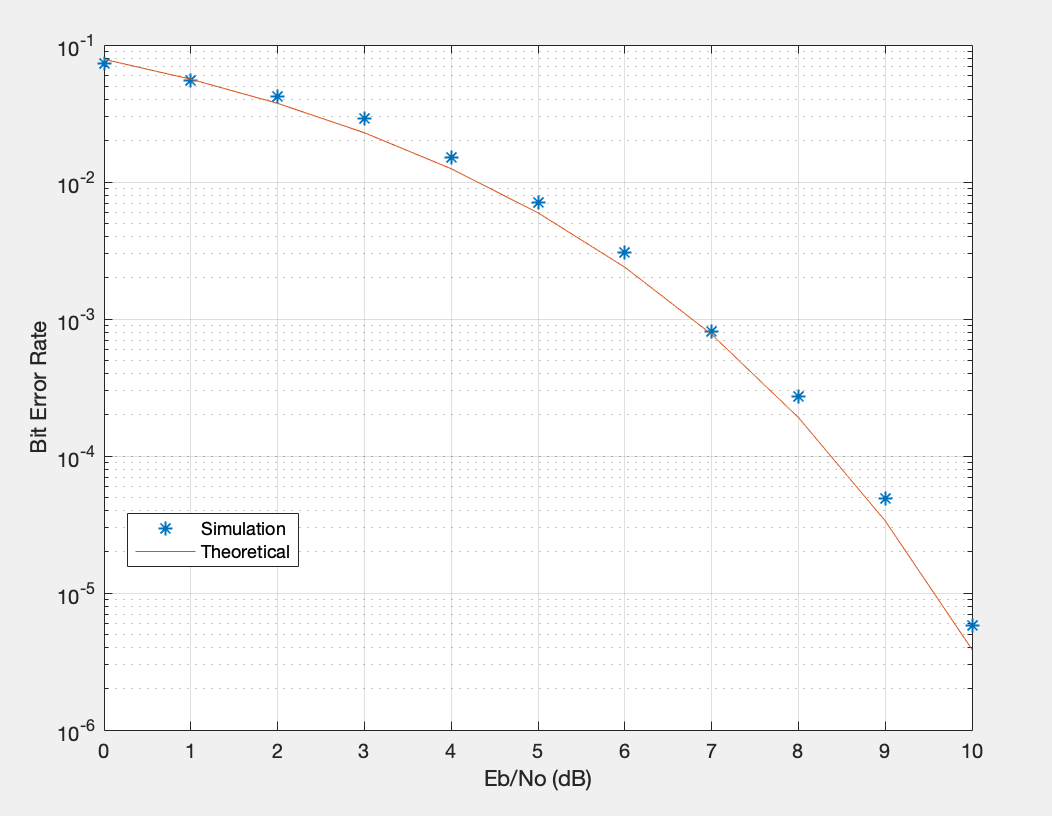


Figure 1. BER for cyclic prefix of 12

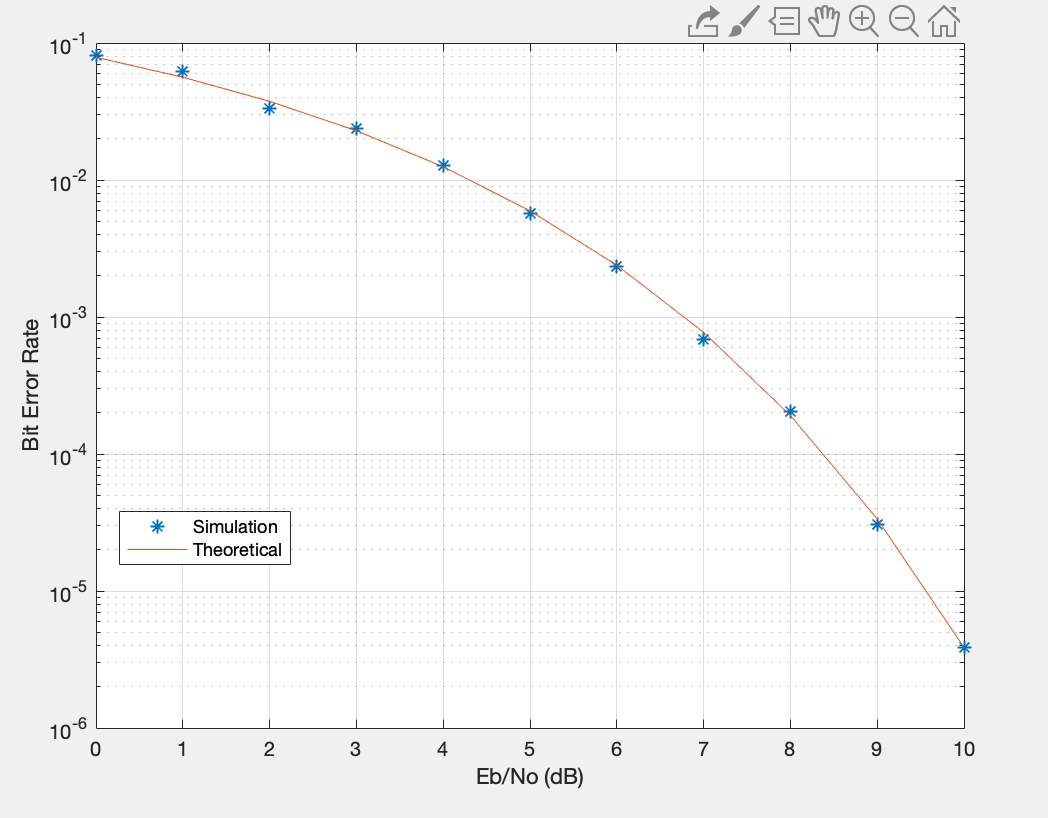


Figure 1. BER for cyclic prefix of 24

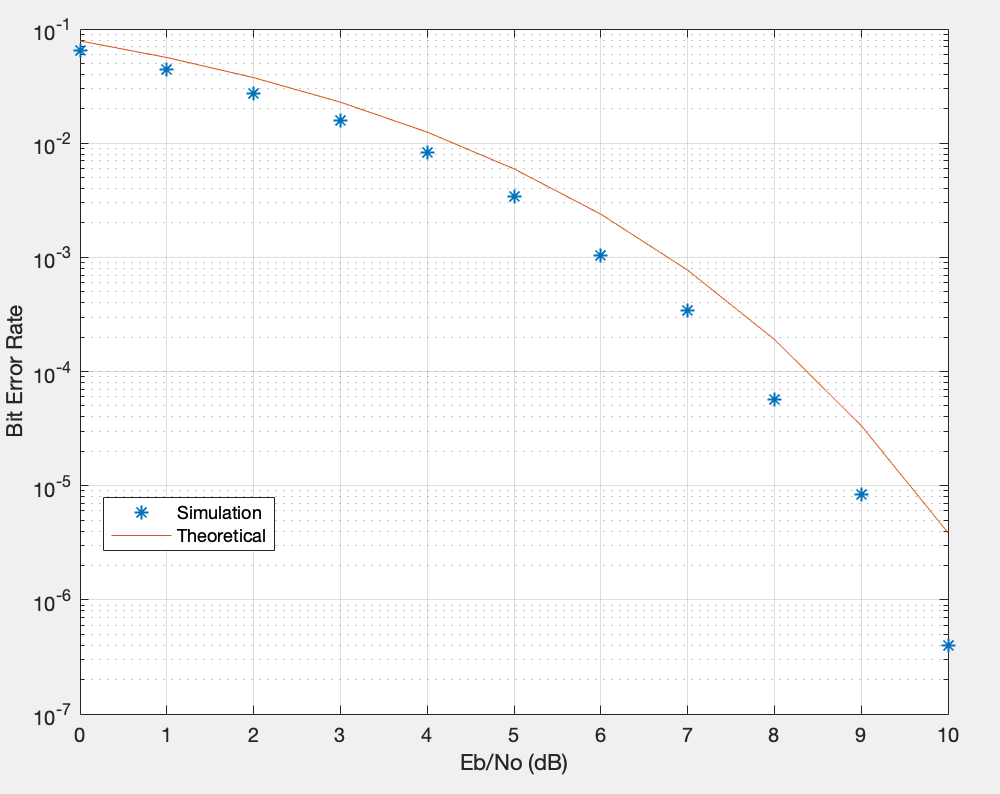


Figure 1. BER for cyclic prefix of 60

* 1. As can be seen above, a larger cyclic prefix results in a lesser bit error rate.

**References**

[1] “comm.OFDMModulator,” *MATLAB*, 2016. [Online]. Available: https://www.mathworks.com/help/comm/ref/comm.ofdmmodulator-system-object.html. [Accessed: 20-Apr-2019].