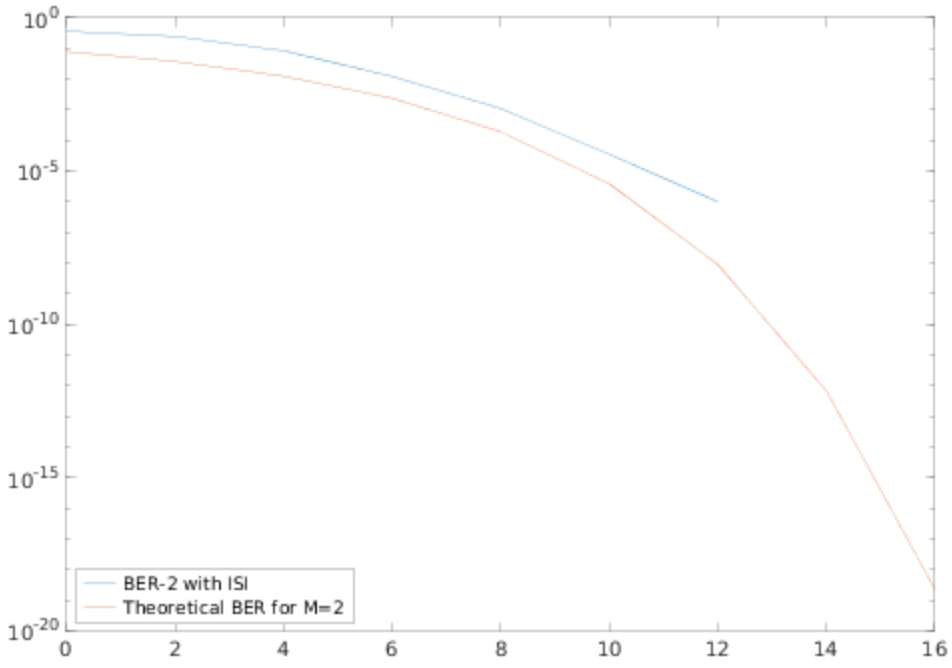
```
clear all; clc; close all;

numIterations = 10000;  % The number of iterations of the simulation
numSymbols = 1000;
numTraining = 150;
m_ary = [2, 4, 16];        % The M-ary number, 2 corresponds to binary
 modulation

SNR_Vec = 0:2:16;
SNRlen = length(SNR_Vec);

chan = [1, 0.2, 0.4];

tic;
M = 2;
codeWordLen = 15;
msgLen = 7;
numWords = ceil(numSymbols/codeWordLen);
trainingBits = (numTraining/codeWordLen) * msgLen; % should always be
 int. Error check?

BERvec2 = zeros(numIterations, SNRlen);

enc = comm.BCHEncoder(codeWordLen, msgLen);
dec = comm.BCHDecoder(codeWordLen, msgLen);
refconst = qammod(0:15,16);


parfor ii=1:numIterations
    %generate numSymbols number of symbols: this is our message, not
    %necessarily in binary
    msg = randi([0, M-1], msgLen * numWords, 1);

    %encode some stuffs

    msg_enc = step(enc, msg);
    %msg_enc = encodeMsg(msg, codeWordLen, msgLen);

    for jj=1:SNRlen
        tx = qammod(msg_enc, M);


        if isequal(chan, 1);
            txChan = tx; % Apply the channel
            txNoisy = txChan; % "add noise"
        else
            txChan = filter(chan,1,tx);  % Apply the channel.
            txNoisy = awgn(txChan,SNR_Vec(jj)); % add noise

            %make the eq
            %Some previous attempts
                %lineq = comm.LinearEqualizer('Algorithm','LMS',
 'NumTaps',6,'StepSize',0.01); % doesn't work on matlab 2018a
```

```
                %eq1 = lineareq(6, lms(0.001));
                %txNoisy = filter(eq1.weights, 1, txNoisy);
            eq1 = dfe(12,6, lms(0.01));
            eq1.SigConst = qammod(0:M-1, M, 'UnitAveragePower', true);
            eq1.ResetBeforeFiltering = 0;

            txNoisy = equalize(eq1,txNoisy,tx(1:numTraining)); %
 Equalize.

            reset(eq1); % clean up - to be removed?
        end

        rx = qamdemod(txNoisy, M);
        %rxTmp = de2bi(rx, 'left-msb').'; %transpose here
        %rxMsg = rxTmp(:);

        % #tb/15 = #sets
        % (#tb % 15) - 8 = # of extra. If negative, set 0
        %dec_msg = decodeMsg(rx, codeWordLen, msgLen);
        dec_msg = step(dec, rx);

        [~, BERvec2(ii,jj)] = biterr(msg(trainingBits+1:end),
 dec_msg(trainingBits+1:end));
    end
end

ber2 = mean(BERvec2,1);

figure(2);
semilogy(SNR_Vec, ber2, 'DisplayName', "BER-2 with ISI")
hold on;
berTheory2 = berawgn(SNR_Vec,'psk', 2,'nondiff');
semilogy(SNR_Vec,berTheory2,'DisplayName', 'Theoretical BER for M=2')
legend('Location', 'southwest')
toc
```

```
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 8).
Elapsed time is 548.271951 seconds.
```



4 and 16

```
function ret_vect = encodeMsg(bits, N, K)
%This function encodes using BCH

    % N is the codeword length
    % K is message length
    numBits = size(bits,1);

    %obtain gf form of blocks of 7 bit messages
    gf_bits = gf(reshape(bits,[K numBits/K]));

    %each column of code_bch correlates to a codeword
    code_bch = bchenc(gf_bits.',N,K).';

    code_bch_vect = code_bch.x;
    ret_vect = code_bch_vect(:);
end


% BCH 15-7 decoder

% demodulated matrix comes in column-wise (codewords along the
 columns)

function decoded = decodeMsg(demodulated, N, K)
    % N is the codeword length
    % K is message length
```

```
    numBits = size(demodulated,1);

    %obtain gf form of blocks of 7 bit messages
    gf_bits = gf(reshape(demodulated,[N numBits/N]));

    gfed = transpose(gf_bits);  % "G-F-ed" = convert to GF array
           % transpose to make it row-wise b/c bchdec acts on rows
    decoded = bchdec(gfed,N,K);
    decoded = decoded.x;
    decoded = decoded.';
    decoded = decoded(:);
end

% decoded matrix is returned row-wise
```

Published with MATLAB® R2019b