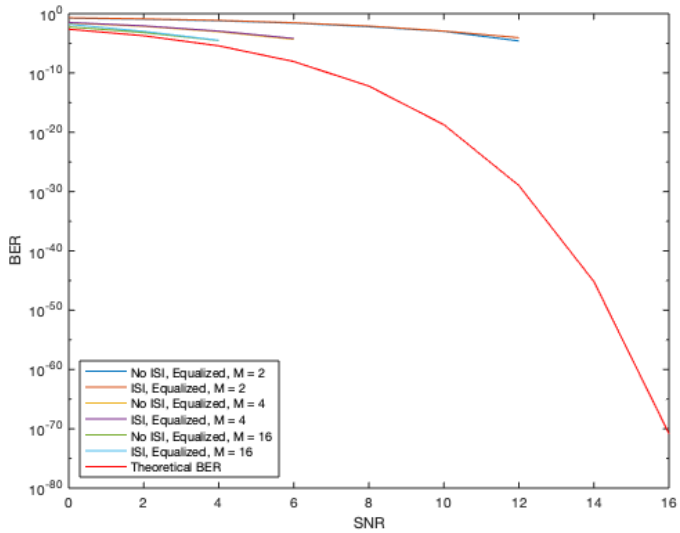
```
% Comm Theory Final Project Part 1, Kevin Jiang and Yuecen Wang
clear;close all;clc

numIter = 200;  % The number of iterations of the simulation
nSym = 1000;    % The number of symbols per packet
SNR_Vec = 0:2:16;
lenSNR = length(SNR_Vec); % Vector containing values of SNR to iterate through
Marray = [2, 2, 4, 4, 16, 16]; % Vector containing values of M, each value is used to time, once for no ISI, once for with ISI
lenM = length(Marray);
eqlms = dfe(5,3, lms(0.005)); % Decision Feedback Equalizer using LMS, which worked had better BER than RLS for same nubmer of bits.
ber = zeros(lenM, lenSNR); % Matrix to hold bit error rates
for mindex = 1:lenM
    M = Marray(mindex); % The M-ary number, 2 corresponds to binary modulation
    if (mod(mindex,2) == 0)
        chan = [1 .2 .4];      % Apply channel for even numbered indices of Marray
    else
        chan = 1; % Don't apply channel for odd number indices of M array
    end
    %chan = [0.227 0.460 0.688 0.460 0.227]';   % Not so invertible, severe ISI

    %Create equalizer object with signal constellation based on M
    sigConst = qammod(0:M-1,M);
    eqlms.SigConst = sigConst;
    eqlms.ResetBeforeFiltering = 0;
    trainLen = 200; % Length of training signal, here first 200 bits of recieved signal used as training
    % Create a vector to store the BER computed during each iteration
    berVec = zeros(numIter, lenSNR); % Vector for holding mean of ber at each SNR
    % Run the simulation numIter amount of times
for i = 1:numIter

    bits = randi([0 1],  1, nSym*log2(M));     % Generate random bits
    % New bits must be generated at every
    % iteration
    bitLen = length(bits);

    % Convert binary to decimal
    msg = zeros(1, bitLen/log2(M));
    msgindex = 1;
    for bitindex = 1:log2(M):(bitLen)
        msg(msgindex) = bi2de(bits(bitindex:bitindex+log2(M)-1), 'left-msb');
        msgindex = msgindex+1;
    end


    for j = 1:lenSNR % one iteration of the simulation at each SNR Value


        tx = qammod(msg,M);  % BPSK modulate the signal

        if isequal(chan,1)
            txChan = tx;
        else
            txChan = filter(chan,1,tx);  % Apply the channel.
        end

        txNoisy = awgn(txChan,SNR_Vec(j)+ 10*log10(log2(M))); % Add AWGN

        eqSig = equalize(eqlms,txNoisy,tx(1:trainLen)); % Equalize, training with first 200 bits of signal

        rxEq = qamdemod(eqSig,M); % Demodulate

        % Converting the symbols back to bits
        rxMSG = zeros(1, bitLen);
        rxMSGindex = 1;
        for rxEqindex = 1:1:length(rxEq)
            rxMSG(rxMSGindex:rxMSGindex+log2(M)-1) = de2bi(rxEq(rxEqindex), log2(M), 2, 'left-msb');
            rxMSGindex = rxMSGindex + log2(M);
        end


        % Compute and store the BER for this iteration
        [~, berVec(i,j)] = biterr(bits(trainLen+1:end), rxMSG(trainLen+1:end));  % We're interested in the BER, which is the 2nd output of BITERR

    end  % End SNR iteration
end      % End numIter iteration
    ber(mindex,:) = mean(berVec,1);
end

% Compute and plot the mean BER

figure;
semilogy(SNR_Vec, ber(1,:));
for k = 2:lenM
    hold on
    semilogy(SNR_Vec, ber(k,:));
end
```

```
% Compute the theoretical BER for this scenario
berTheory = berawgn(SNR_Vec + 10*log10(log2(M)),'psk',log2(M),'nondiff');
hold on
semilogy(SNR_Vec,berTheory,'r')
xlabel('SNR');
ylabel('BER');
legend('No ISI, Equalized, M = 2', 'ISI, Equalized, M = 2', 'No ISI, Equalized, M = 4', ...
    'ISI, Equalized, M = 4', 'No ISI, Equalized, M = 16', 'ISI, Equalized, M = 16', 'Theoretical BER', 'Location', 'southwest')
```



Published with MATLAB® R2019b