Table of Contents

ECE 300

```
clear all;
close all;
clc;
```

Setup for Transmission

Determines the number of iterations, number of symbols, the SNR range, what modulation is used and the amount of training symbols. The setup variables are declared here

```
numIter = 20;  % The number of iterations of the simulation
nSym = 1000;    % The number of symbols per packet
SNR_Vec = 0:2:16;
lenSNR = length(SNR_Vec);
symbolTrain = 60;

M = 2;        % The M-ary number, 2 corresponds to binary modulation
k = log2(M);

% Different Channel that were used in testing the BER
%chan = 1;
chan = [1 .2 .4];
%chan = [0.227 0.460 0.688 0.460 0.227]';

% Create a vector to store the BER computed during each iteration
berVec = zeros(numIter, lenSNR);
berVecQAM = zeros(numIter, lenSNR);
```

Simulation

Generates random bits, reshapes the bits for modulation and then add noise and channel. The receiver has an equalizer. Different equalizers were shown and the BER was simulated

```
for i = 1:numIter

    % Random bit generation
    bits = randi(2,[nSym*k, 1])-1;
    bits4 = randi(2,[nSym*2, 1])-1;
    bits16 = randi(2,[nSym*4, 1])-1;

    % Reshapes the message for different modulation
    msg = bi2de(reshape(bits, [nSym k]));
```

```
   msg4 = bi2de(reshape(bits4, [nSym 2]));
   msg16 = bi2de(reshape(bits16, [nSym 4]));

   for j = 1:lenSNR


       % Modulates the signal from the msg vector
       tx = qammod(msg,M);  % BPSK modulate the signal
       tx4 = qammod(msg4, 4);
       tx16 = qammod(msg16, 16);

       % Chooses which channel is used
       if isequal(chan,1)
           txChan = tx;
       elseif isa(chan,'channel.rayleigh')
           reset(chan) % Draw a different channel each iteration
           txChan = filter(chan,tx);
       else
           txChan = filter(chan,1,tx);  % Apply the channel.
       end

       % Scale the noise to match for each symbol
       if (M == 2)
           txNoisy = awgn(txChan,3+SNR_Vec(j),'measured'); % Add AWGN
       else
           txNoisy = awgn(txChan,10*log10(k)+SNR_Vec(j),'measured');
       end

       % Special tx noise to do 4 QAM and 16 QAM runs concurrently to
show
       % that the noise wasa properly scaled
       txNoisy4 = awgn(tx4, 10*log10(log2(4))+SNR_Vec(j),'measured');
       txNoisy16 = awgn(tx16,
10*log10(log2(16))+SNR_Vec(j),'measured');

       % Demod the special case for 4 QAM and 16 QAM
       rx4 = qamdemod(txNoisy4, 4);
       rx4 = de2bi(rx4);
       rxMSG4 = reshape(rx4, [nSym*2, 1]);
       [~, berVecQAM(i,j,1)] = biterr(bits4, rxMSG4);
       rx16 = qamdemod(txNoisy16, 16);
       rx16 = de2bi(rx16);
       rxMSG16 = reshape(rx16, [nSym*4, 1]);
       [~, berVecQAM(i,j,2)] = biterr(bits16, rxMSG16);
       berQAM(:,1) = mean(berVecQAM(:,:,1));
       berQAM(:,2) = mean(berVecQAM(:,:,2));

       % First Equalizer: DFE step size: 0.01
       eq1 = dfe(5, 3, lms(0.01));
       eq1.SigConst = qammod((0:M-1)',M)';
       eq1.ResetBeforeFiltering = 1;
       [symbolest1, y(:,1)] = equalize(eq1, txNoisy,
tx(1:symbolTrain));
       rx(:,1) = qamdemod(y(:,1),M);
```

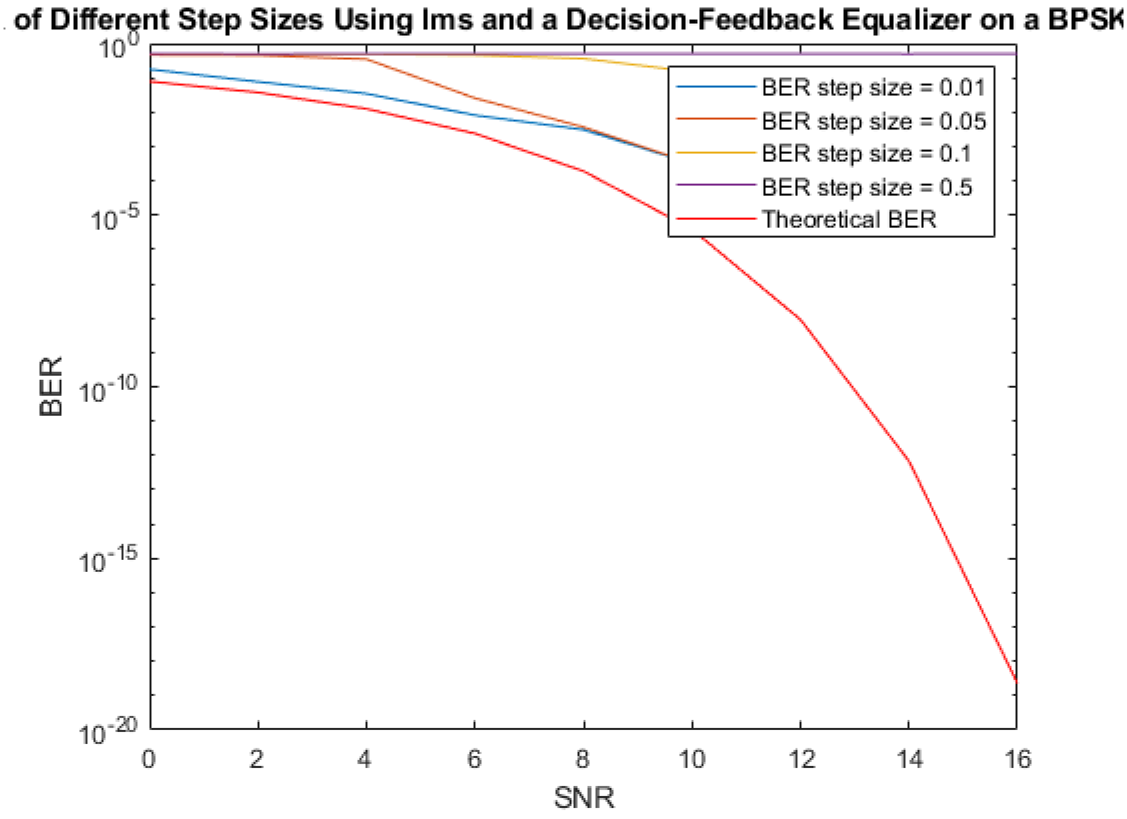
```
        rxMSG = rx;
        [~, berVec(i,j,1)] = biterr(msg(symbolTrain+1:end), 
rxMSG(symbolTrain+1:end, 1));
        ber(:,1) = mean(berVec(:,:,1));

        % Second Equalizer: DFE step size: 0.05
        eq2 = dfe(5, 3, lms(0.05));
        eq2.SigConst = qammod((0:M-1)',M)';
        eq2.ResetBeforeFiltering = 1;
        [symbolest2, y(:,2)] = equalize(eq2, txNoisy, 
tx(1:symbolTrain));
        rx(:,2) = qamdemod(y(:,2),M);
        rxMSG = rx;
        [~, berVec(i,j,2)] = biterr(msg(symbolTrain+1:end), 
rxMSG(symbolTrain+1:end, 2));
        ber(:,2) = mean(berVec(:,:,2));

        % Third Equalizer: DFE step size: 0.1
        eq3 = dfe(5, 3, lms(0.1));
        eq3.SigConst = qammod((0:M-1)',M)';
        eq3.ResetBeforeFiltering = 1;
        [symbolest3, y(:,3)] = equalize(eq3, txNoisy, 
tx(1:symbolTrain));
        rx(:,3) = qamdemod(y(:,3),M);
        rxMSG = rx;
        [~, berVec(i,j,3)] = biterr(msg(symbolTrain+1:end), 
rxMSG(symbolTrain+1:end, 3));
        ber(:,3) = mean(berVec(:,:,3));

        % Fourth Equalizer: DFE step size: 0.5
        eq4 = dfe(5, 3, lms(0.5));
        eq4.SigConst = qammod((0:M-1)',M)';
        eq4.ResetBeforeFiltering = 1;
        [symbolest4, y(:,4)] = equalize(eq4, txNoisy, 
tx(1:symbolTrain));
        rx(:,4) = qamdemod(y(:,4),M);
        rxMSG = rx;
        [~, berVec(i,j,4)] = biterr(msg(symbolTrain+1:end), 
rxMSG(symbolTrain+1:end, 4));
        ber(:,4) = mean(berVec(:,:,4));

        % Fifth Equalizer: Linear Equalizer LMS step size: 0.01
        eq5 = lineareq(8, lms(0.01));
        eq5.SigConst = qammod((0:M-1)',M)';
        eq5.ResetBeforeFiltering = 1;
        [symbolest5, y(:,5)] = equalize(eq5, txNoisy, 
tx(1:symbolTrain));
        rx(:,5) = qamdemod(y(:,5),M);
        rxMSG = rx;
        [~, berVec(i,j,5)] = biterr(msg(symbolTrain+1:end), 
rxMSG(symbolTrain+1:end, 5));
        ber(:,5) = mean(berVec(:,:,5));

        % Sixth Equalizer: Linear Equalizer LMS step size: 0.05
        eq6 = lineareq(8, lms(0.05));
        eq6.SigConst = qammod((0:M-1)',M)';
        eq6.ResetBeforeFiltering = 1;
        [symbolest6, y(:,6)] = equalize(eq6, txNoisy,
tx(1:symbolTrain));
        rx(:,6) = qamdemod(y(:,6),M);
        rxMSG = rx;
        [~, berVec(i,j,6)] = biterr(msg(symbolTrain+1:end),
rxMSG(symbolTrain+1:end, 6));
        ber(:,6) = mean(berVec(:,:,6));

        % Seventh Equalizer: Linear Equalizer RLS Forgetting Factor: 1
        eq7 = lineareq(8, rls(1));
        eq7.SigConst = qammod((0:M-1)',M)';
        eq7.ResetBeforeFiltering = 1;
        [symbolest7, y(:,7)] = equalize(eq7, txNoisy,
tx(1:symbolTrain));
        rx(:,7) = qamdemod(y(:,7),M);
        rxMSG = rx;
        [~, berVec(i,j,7)] = biterr(msg(symbolTrain+1:end),
rxMSG(symbolTrain+1:end, 7));
        ber(:,7) = mean(berVec(:,:,7));

        % Eighth Equalizer: Linear Equalizer RLS Forgetting Factor:
0.7
        eq8 = lineareq(8, rls(0.7));
        eq8.SigConst = qammod((0:M-1)',M)';
        eq8.ResetBeforeFiltering = 1;
        [symbolest8, y(:,8)] = equalize(eq8, txNoisy,
tx(1:symbolTrain));
        rx(:,8) = qamdemod(y(:,8),M);
        rxMSG = rx;
        [~, berVec(i,j,8)] = biterr(msg(symbolTrain+1:end),
rxMSG(symbolTrain+1:end, 8));
        ber(:,8) = mean(berVec(:,:,8));

        % Nineth Equalizer: DFE RLS step size: 0.01 with 25 training
symbols
        eq9 = dfe(5, 3, lms(0.01));
        eq9.SigConst = qammod((0:M-1)',M)';
        eq9.ResetBeforeFiltering = 1;
        [symbolest9, y(:,9)] = equalize(eq9, txNoisy, tx(1:25));
        rx(:,9) = qamdemod(y(:,9),M);
        rxMSG = rx;
        [~, berVec(i,j,9)] = biterr(msg(26:end), rxMSG(26:end, 9));
        ber(:,9) = mean(berVec(:,:,9));

        % Tenth Equalizer: DFE RLS step size: 0.01 with 50 training
symbols
        eq10 = dfe(5, 3, lms(0.01));
        eq10.SigConst = qammod((0:M-1)',M)';
        eq10.ResetBeforeFiltering = 1;
        [symbolest10, y(:,10)] = equalize(eq10, txNoisy, tx(1:50));
        rx(:,10) = qamdemod(y(:,10),M);
        rxMSG = rx;
        [~, berVec(i,j,10)] = biterr(msg(51:end), rxMSG(51:end, 10));
        ber(:,10) = mean(berVec(:,:,10));

        % Eleventh Equalizer: DFE RLS step size: 0.01 with 75 training
 symbols
        eq11 = dfe(5, 3, lms(0.01));
        eq11.SigConst = qammod((0:M-1)',M)';
        eq11.ResetBeforeFiltering = 1;
        [symbolest11, y(:,11)] = equalize(eq11, txNoisy, tx(1:75));
        rx(:,11) = qamdemod(y(:,11),M);
        rxMSG = rx;
        [~, berVec(i,j,11)] = biterr(msg(76:end), rxMSG(76:end, 11));
        ber(:,11) = mean(berVec(:,:,11));

        % Twelfth Equalizer: DFE RLS step size: 0.01 with 100 training
 symbols
        eq12 = dfe(5, 3, lms(0.01));
        eq12.SigConst = qammod((0:M-1)',M)';
        eq12.ResetBeforeFiltering = 1;
        [symbolest12, y(:,12)] = equalize(eq12, txNoisy, tx(1:100));
        rx(:,12) = qamdemod(y(:,12),M);
        rxMSG = rx;
        [~, berVec(i,j,12)] = biterr(msg(101:end), rxMSG(101:end,
 12));
        ber(:,12) = mean(berVec(:,:,12));



    end  % End SNR iteration
end      % End numIter iteration
```

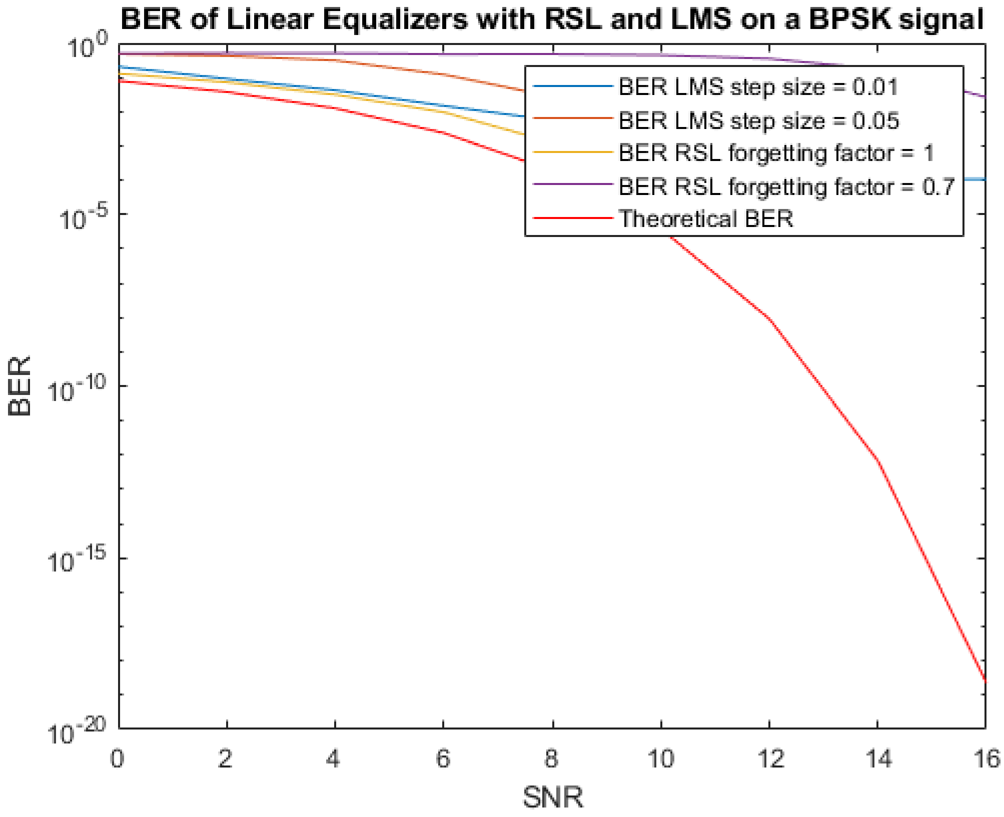
Plot for BERs

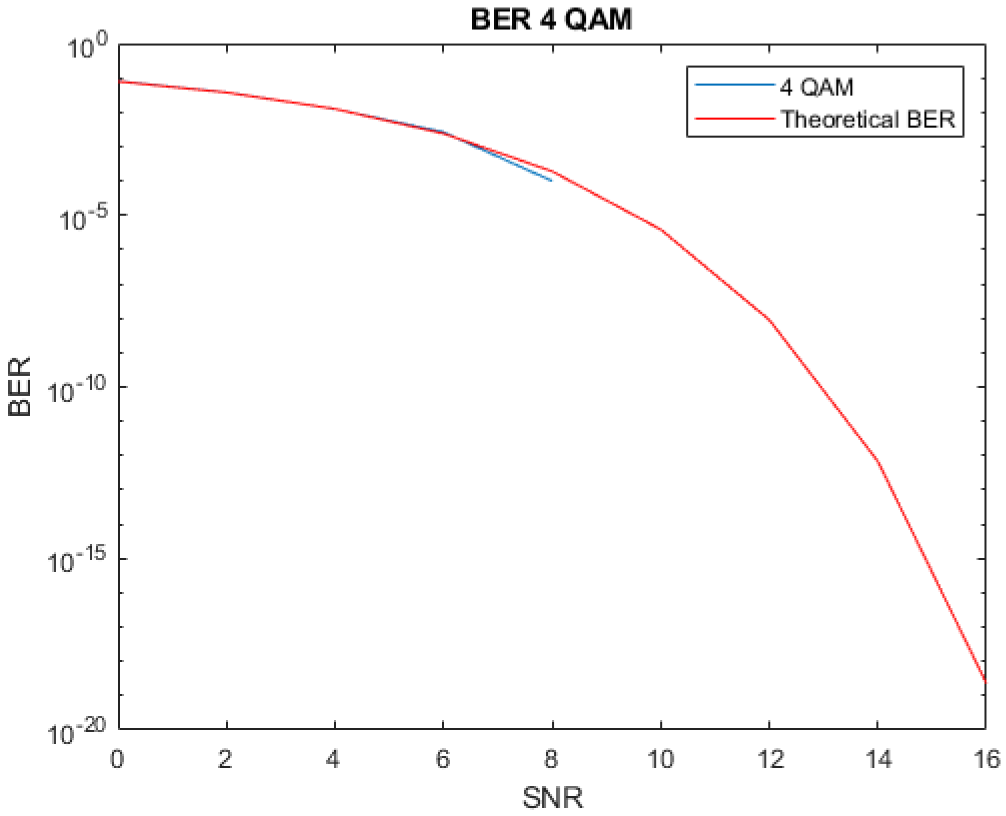
Takes the mean BER from the demod and constructs graph comparing different equalizer and symbol training

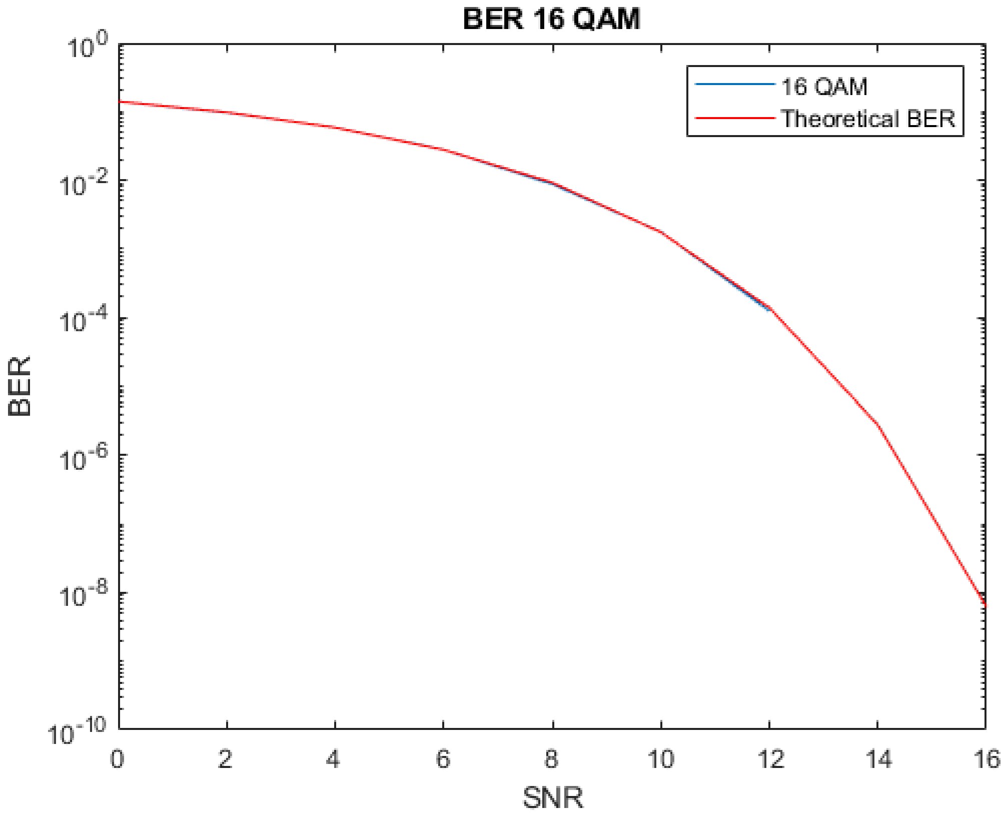
```
berTheory = berawgn(SNR_Vec,'pam',M);
figure;
semilogy(SNR_Vec, ber(:,1:4));
hold on
semilogy(SNR_Vec,berTheory,'r');
title('BER of Different Step Sizes Using lms and a Decision-Feedback
 Equalizer on a BPSK signal')%,'fontsize',18);
xlabel('SNR')%,'fontsize',18);
ylabel('BER')%,'fontsize',18);
legend({'BER step size = 0.01', 'BER step size = 0.05', 'BER step size
 = 0.1','BER step size = 0.5','Theoretical BER'})%, 'FontSize', 14)

figure;
semilogy(SNR_Vec, ber(:,5:8));
hold on
semilogy(SNR_Vec,berTheory,'r')
```

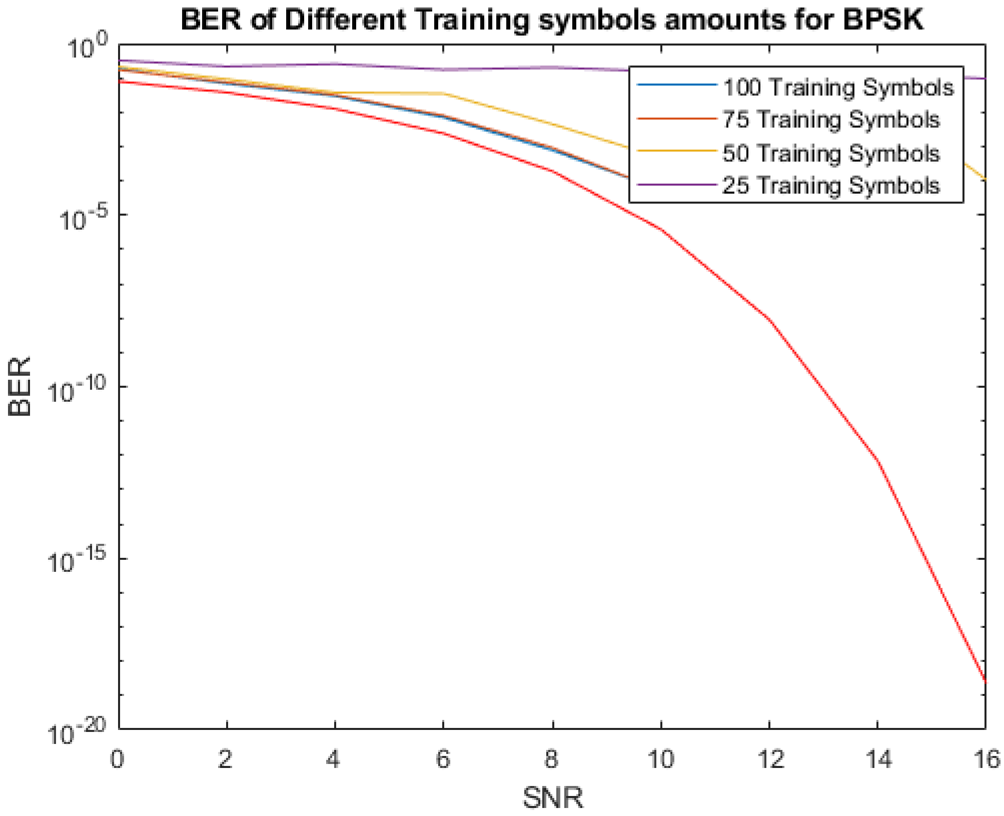
```
title('BER of Linear Equalizers with RSL and LMS on a BPSK
 signal')%,'fontsize',18);
xlabel('SNR')%,'fontsize',18);
ylabel('BER')%,'fontsize',18);
legend({'BER LMS step size = 0.01', 'BER LMS step size =
 0.05', 'BER RSL forgetting factor = 1','BER RSL forgetting factor =
 0.7','Theoretical BER'})%, 'FontSize', 14)

berTheory4QAM = berawgn(SNR_Vec,'qam',4);
figure;
semilogy(SNR_Vec, berQAM(:,1));
hold on
semilogy(SNR_Vec,berTheory4QAM,'r');
title('BER 4 QAM')%,'fontsize'),18);
xlabel('SNR')%,'fontsize',18);
ylabel('BER')%,'fontsize',18);
legend({'4 QAM','Theoretical BER'})%, 'FontSize', 14)

berTheory4QAM = berawgn(SNR_Vec,'qam',16);
figure;
semilogy(SNR_Vec, berQAM(:,2));
hold on
semilogy(SNR_Vec,berTheory4QAM,'r');
title('BER 16 QAM')%,'fontsize',18);
xlabel('SNR')%,'fontsize',18);
ylabel('BER')%,'fontsize',18);
legend({'16 QAM','Theoretical BER'})%, 'FontSize', 14)

figure;
semilogy(SNR_Vec, ber(:,[12 11 10 9]));
hold on
semilogy(SNR_Vec,berTheory,'r');
title('BER of Different Training symbols amounts for
 BPSK')%,'fontsize',18);
xlabel('SNR')%,'fontsize',18);
ylabel('BER')%,'fontsize',18);
legend({'100 Training Symbols','75 Training Symbols','50 Training
 Symbols','25 Training Symbols'})%, 'FontSize', 14)
```


. of Different Step Sizes Using lms and a Decision-Feedback Equalizer on a BPSK
BER
SNR
10^{0}
10^{-5}
10^{-10}
10^{-15}
10^{-20}
0
2
4
6
8
10
12
14
16
BER step size = 0.01
BER step size = 0.05
BER step size = 0.1
BER step size = 0.5
Theoretical BER


BER of Linear Equalizers with RSL and LMS on a BPSK signal
BER
SNR
10^{0}
10^{-5}
10^{-10}
10^{-15}
10^{-20}
0
2
4
6
8
10
12
14
16
BER LMS step size = 0.01
BER LMS step size = 0.05
BER RSL forgetting factor = 1
BER RSL forgetting factor = 0.7
Theoretical BER


BER 4 QAM
BER
SNR
10^0
10^-5
10^-10
10^-15
10^-20
0
2
4
6
8
10
12
14
16
4 QAM
Theoretical BER


BER 16 QAM
BER
SNR
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
0
2
4
6
8
10
12
14
16
16 QAM
Theoretical BER



Published with MATLAB® R2018a