ECE 300 Communication Theory Matlab Project 2 Using Reed-Solomon

Table of Contents

Zheng Liu, Jing Jiang, Tianshu Ren

Project Description

Our Goal in part 2 is to Achieve BER of 10^-6 at 12 dB SNR over moderate ISI channel using whatever means possible.

Algorithm Initialization

```
clear all;close all;clc
numIter = 5;  % The number of iterations of the simulation
nSym = 1000;    % Real symbol number needs to be scaled because of
 encoding. See bit generation code.
SNR_Vec = 0:2:16;
lenSNR = length(SNR_Vec);

M = 4;        % The M-ary number, 2 corresponds to binary modulation

%chan = 1;          % No channel
chan = [1 .2 .4]; % Somewhat invertible channel impulse response,
 Moderate ISI
%chan = [0.227 0.460 0.688 0.460 0.227]';   % Not so invertible,
 severe ISI


% Create a vector to store the BER computed during each iteration
berVec = zeros(numIter, lenSNR);
eqlBerVec = zeros(numIter, lenSNR);
% Run the simulation numIter amount of times
```

Block code

```
N = 15;  % Codeword length
K = 2;  % Message length
S = K;  % Shortened message length
```

```
   % N and K have to be paired up. All valid pairs are shown by
 bchnumerr.
   % Small K can deal with more bits of error but low efficiency. BCH
 works
   % only if N and K are picked so that many parity bits are used.
 Therefore,
   % it is not ideal for maximizing bit rate. Demo of it working uses
 4-QAM.
```

Reed Solomon

```
enc = comm.RSEncoder(N,K,'BitInput',true);
dec = comm.RSDecoder(N,K,'BitInput',true);


%numPuncs = N - K;
%m = log2(N + 1);
%enc.PuncturePatternSource = 'Property';
%enc.PuncturePattern = [ones(N-K-numPuncs,1); zeros(numPuncs,1)];
% Set the shortened message length values
%enc.ShortMessageLength = S;
% Specify the field of GF(2^6) in the RS encoder/decoder System
 objects, by setting the PrimitivePolynomialSource property ...
% to 'Property' and the PrimitivePolynomial property to a 6th degree
 primitive polynomial.
%enc.PrimitivePolynomialSource = 'Property';
%enc.PrimitivePolynomial = de2bi(primpoly(m, 'nodisplay'), 'left-
msb');

%dec.PuncturePatternSource = 'Property';
%dec.PuncturePattern = [ones(N-K-numPuncs,1); zeros(numPuncs,1)];
% Set the shortened message length values
%dec.ShortMessageLength = S;
% Specify the field of GF(2^6) in the RS encoder/decoder System
 objects, by setting the PrimitivePolynomialSource property ...
% to 'Property' and the PrimitivePolynomial property to a 6th degree
 primitive polynomial.
%dec.PrimitivePolynomialSource = 'Property';
%dec.PrimitivePolynomial = de2bi(primpoly(m, 'nodisplay'), 'left-
msb');


%Tried user-defined parameters above but did not get improvement.
%Researched puncture but did not know how to use it.
```

Iteration

```
tic
for i = 1:numIter

    %  bits = randint(1, nSym*M, [0 1]);     % Generate random bits
    bits = randi(2,[K*nSym*log2(M), 1])-1;
```

```
   % New bits must be generated at every
   % iteration

   %msg = bits;
   msg = bi2de(reshape(bits,log2(M),[]).');
   encodedMSG = enc(bits);


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

       tx =
qammod(encodedMSG,M,'UnitAveragePower',true,'InputType','bit');  %
BPSK modulate the signal
       tx2 =
qammod(bits,M,'UnitAveragePower',true,'InputType','bit');

       if isequal(chan,1)
           txChan = tx;
           txChan2 = tx2;
       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.
           txChan2 = filter(chan,1,tx2);
       end

       % Convert from EbNo to SNR.
       % Note: Because No = 2*noiseVariance^2, we must add ~3 dB
       % to get SNR (because 10*log10(2) ~= 3).
       txNoisy =
awgn(txChan,10*log10(log2(M))+SNR_Vec(j),'measured'); % Add AWGN
       txNoisy2 =
awgn(txChan2,10*log10(log2(M))+SNR_Vec(j),'measured');
       %txNoisy = awgn(txChan,3+SNR_Vec(j),'measured'); % Add AWGN
       rx = txNoisy;
       rx2 = txNoisy2;
       %rx_demod = qamdemod(txNoisy,M); % Demodulate

       % Again, if M was a larger number, I'd need to convert my
symbols
       % back to bits here.

       rxMSG =
qamdemod(rx2,M,'UnitAveragePower',true,'OutputType','bit');
       noEqlMSG =
qamdemod(rx,M,'UnitAveragePower',true,'OutputType','bit');

       %rxMSG = reshape(de2bi(rx,log2(M)),[],1);



       %Equilizer
```

```
        mu = 0.001; %step size
        trainlen = 200;
        n = 8; %number of weights
        const = qammod((0:1:M-1),M);
        trainSig=tx(1:trainlen);

        %LMS decision-feedback equalizer
        nfwd = 16;
        nfbk = 12;
        dfeLMS = dfe(nfwd,nfbk,lms(mu));
        dfeLMS.SigConst = const; % Set signal constellation.
        dfeLMS.ResetBeforeFiltering = 0;

        %RLS decision-feedback equalizer
        dfeRLS = dfe(nfwd,nfbk,rls(0.99,0.9));
        dfeRLS.SigConst = const; % Set signal constellation.
        dfeRLS.ResetBeforeFiltering = 0;

        %lms,linear
        %trainMSG = reshape(de2bi(tx(1:trainlen),log2(M)),[],1);
        linLMS = lineareq(n, lms(mu)); % Create an equalizer object.
        linLMS.SigConst = const; % Set signal constellation.
        linLMS.ResetBeforeFiltering = 0;

        %rls,linear
        linRLS = lineareq(n, rls(1,0.1)); % Create an equalizer
 object.
        linRLS.SigConst = const; % Set signal constellation.
        linRLS.ResetBeforeFiltering = 0;

        %lms,decision feedback

        %delay = (numRefTap-1)/eqobj.nSampPerSym;
        [y,eqlSig] = equalize(linRLS,rx,trainSig); % Equalize.
        eqlMSG =
 qamdemod(eqlSig,M,'UnitAveragePower',true,'OutputType','bit');
        decodedMSG = dec(eqlMSG);


        % Compute and store the BER for this iteration

        rxMSG = bi2de(reshape(rxMSG,log2(M),[]).');
        %decodedMSG = bi2de(reshape(decodedMSG,log2(M),[]).');
        [zzz, berVec(i,j)] = biterr(msg, rxMSG);  % We're interested
 in the BER, which is the 2nd output of BITERR

        [zzz, eqlBerVec(i,j)] = biterr(bits(trainlen+1:end),
 decodedMSG(trainlen+1:end));        %For Block Code
        [zzz, eqlBerVec(i,j)] = biterr(bits(trainlen+1:end),
 decodedMSG(trainlen+1:end));
   end  % End SNR iteration
end      % End numIter iteration
time = toc;
time
```

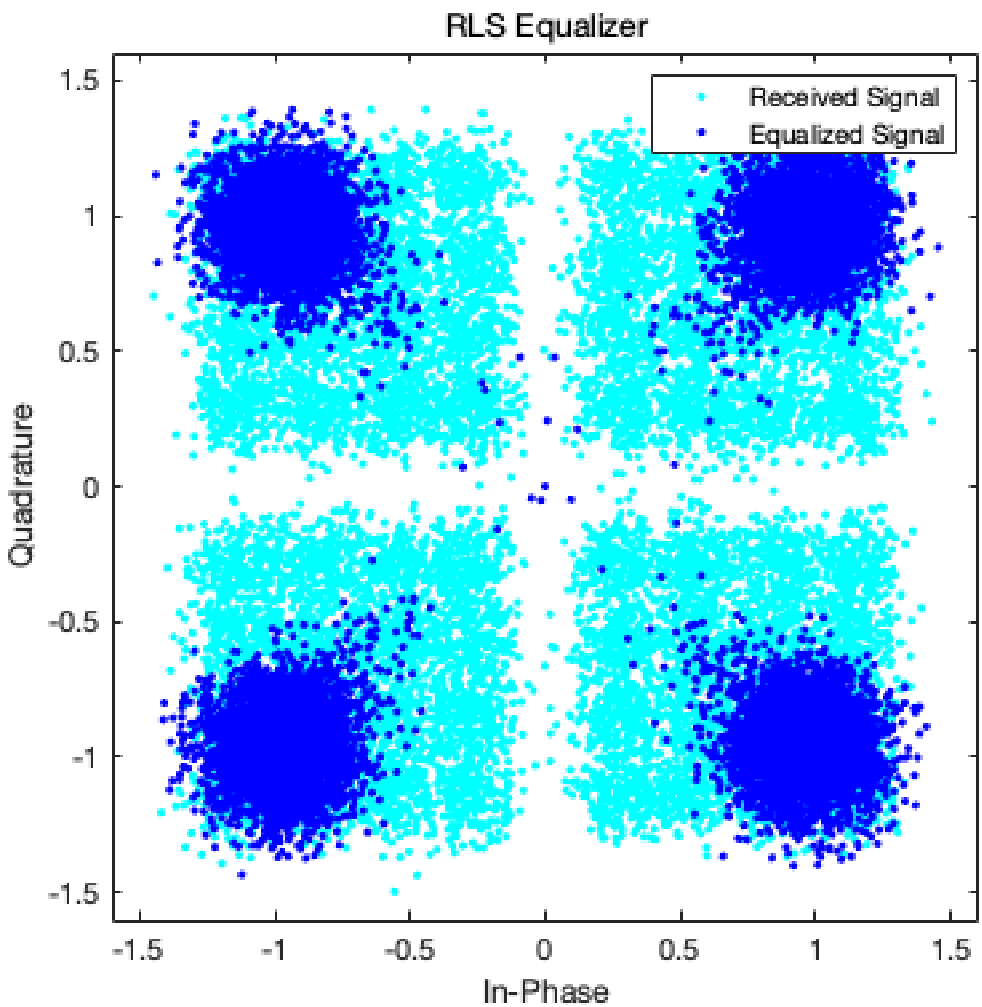
```
time =

   42.3645
```

Plots

Played with M-ary QAM Scatter plot to see how equilizers work on

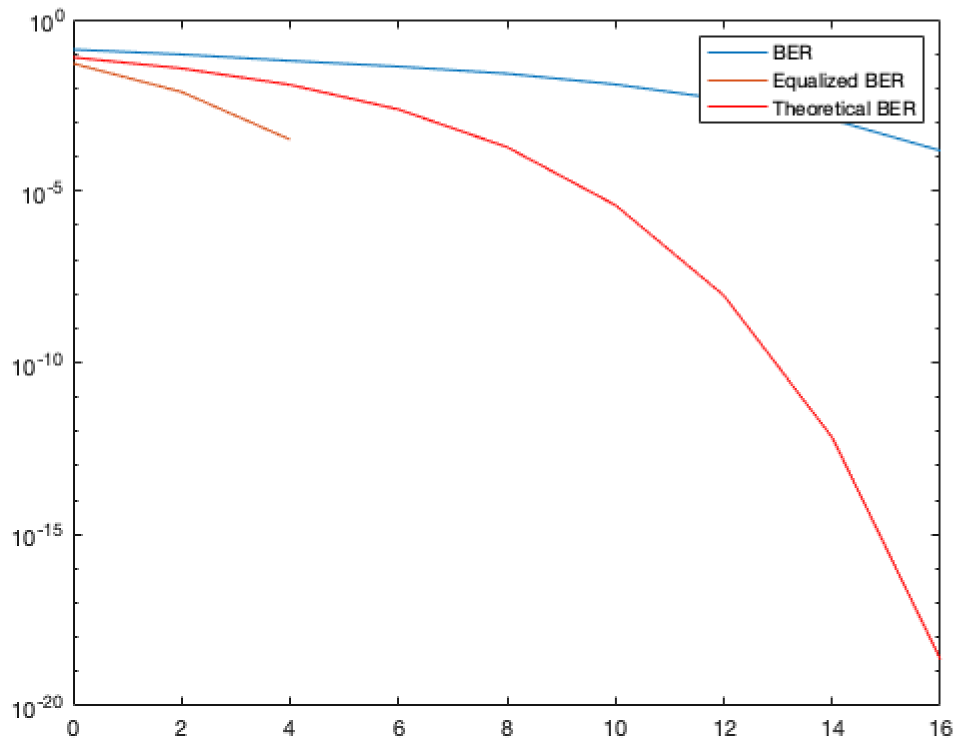
```
h = scatterplot(rx,1,0,'c.');
hold on
scatterplot(y,1,0,'b.',h)
legend('Received Signal','Equalized Signal')
title('RLS Equalizer')
hold off
%
```



Compute and plot the mean BER. The result at 12dB is 0. Which fits the requirement.

```
figure;
ber = mean(berVec,1);
eqlBer = mean(eqlBerVec,1);
semilogy(SNR_Vec, ber);
hold on
semilogy(SNR_Vec, eqlBer);

berTheory = berawgn(SNR_Vec,'qam',M);
hold on
semilogy(SNR_Vec,berTheory,'r')
legend('BER', 'Equalized BER','Theoretical BER')
```



Published with MATLAB® R2018b