Table of Contents

Exercise 11.3

```
alphabet = [2, 3, 5];
variance = [1, 5, 35/3];
for iter = 1:3
    figure()                          % used to plot figure eyediag3
    N=1000; m=pam(N,alphabet(iter),variance(iter));          % random
 +/-1 signal of length N
    M=20; mup=zeros(1,N*M); mup(1:M:N*M)=m;  % oversampling by factor
 of M
    ps=ones(1,M);                                 % square pulse width M
    x=filter(ps,1,mup);             % convolve pulse shape with mup
    neye=5;
    c=floor(length(x)/(neye*M))
    xp=x(N*M-neye*M*c+1:N*M);       % dont plot transients at start
    q=reshape(xp,neye*M,c);         % plot in clusters of size
 5*Mt=(1:198)/50+1;
    subplot(3,1,1), plot(q)
    title(['Eye diagram for rectangular pulse shape using
 ',num2str(alphabet(iter))])

    N=1000; m=pam(N,alphabet(iter),variance(iter));          % random
 +/-1 signal of length N
    M=20; mup=zeros(1,N*M); mup(1:M:N*M)=m;  % oversampling by factor
 of M
    ps=hamming(M);                                % square pulse width M
    x=filter(ps,1,mup);             % convolve pulse shape with mup
    %x=x+0.15*randn(size(x));
    neye=5;
    c=floor(length(x)/(neye*M))
    xp=x(N*M-neye*M*c+1:N*M);       % dont plot transients at start
    q=reshape(xp,neye*M,c);         % plot in clusters of size
 5*Mt=(1:198)/50+1;
    subplot(3,1,2), plot(q)
    title(['Eye diagram for hamming pulse shape using
 ',num2str(alphabet(iter))])
```

```
    N=1000; m=pam(N,alphabet(iter),variance(iter));          % random
 +/-1 signal of length N
    M=20; mup=zeros(1,N*M); mup(1:M:N*M)=m;  % oversampling by factor
 of M
    L=10; ps=srrc(L,0,M,50);
    ps=ps/max(ps);         % sinc pulse shape L symbols wide
    x=filter(ps,1,mup);    % convolve pulse shape with mup
    %x=x+0.15*randn(size(x));
    neye=5;
    c=floor(length(x)/(neye*M))
    xp=x(N*M-neye*M*(c-3)+1:N*M);  % dont plot transients at start
    q=reshape(xp,neye*M,c-3);      % plot in clusters of size
 5*Mt=(1:198)/50+1;
    subplot(3,1,3), plot(q)
    axis([0,100,alphabet(iter) *-1,alphabet(iter)])
    title(['Eye diagram for sinc pulse shape using
 ',num2str(alphabet(iter))])
end
```

c =

200

c =

200

c =

200

c =

200

c =

200

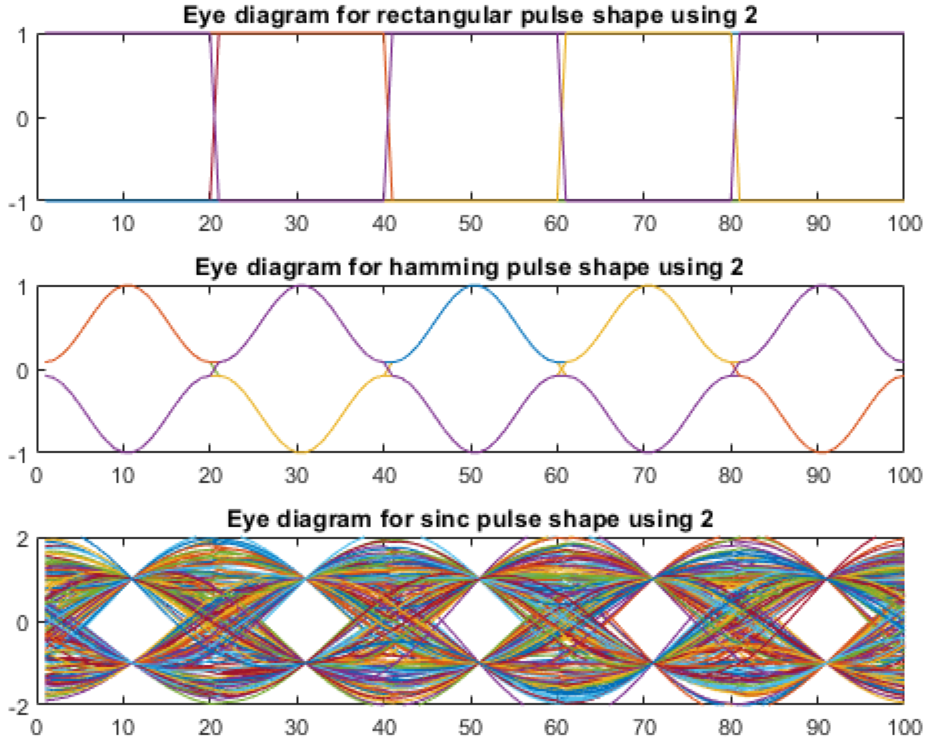
c =

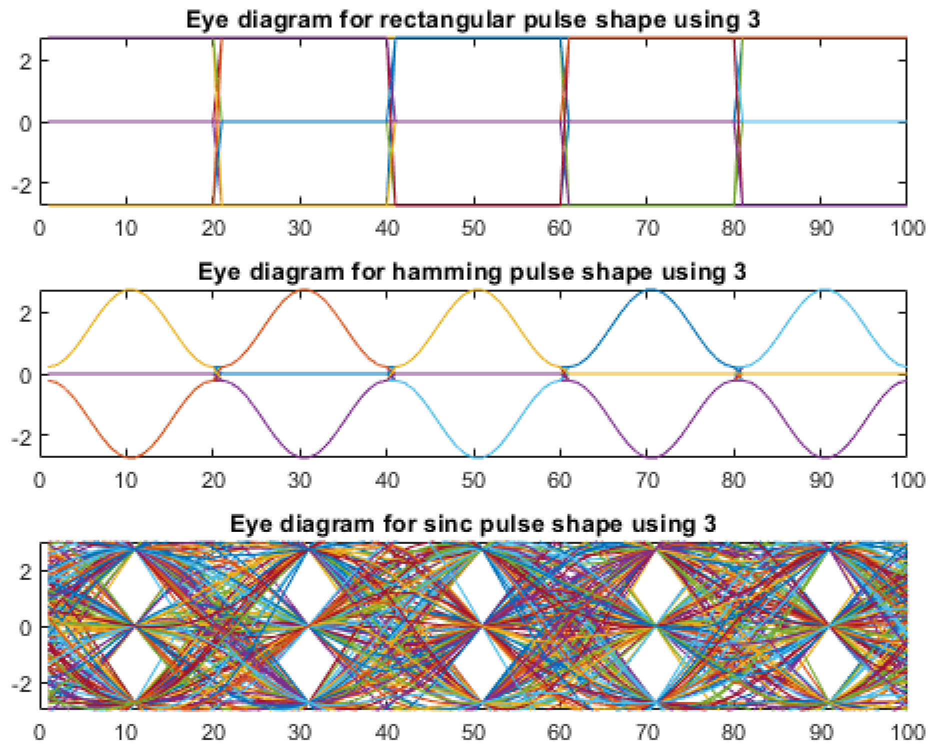
200

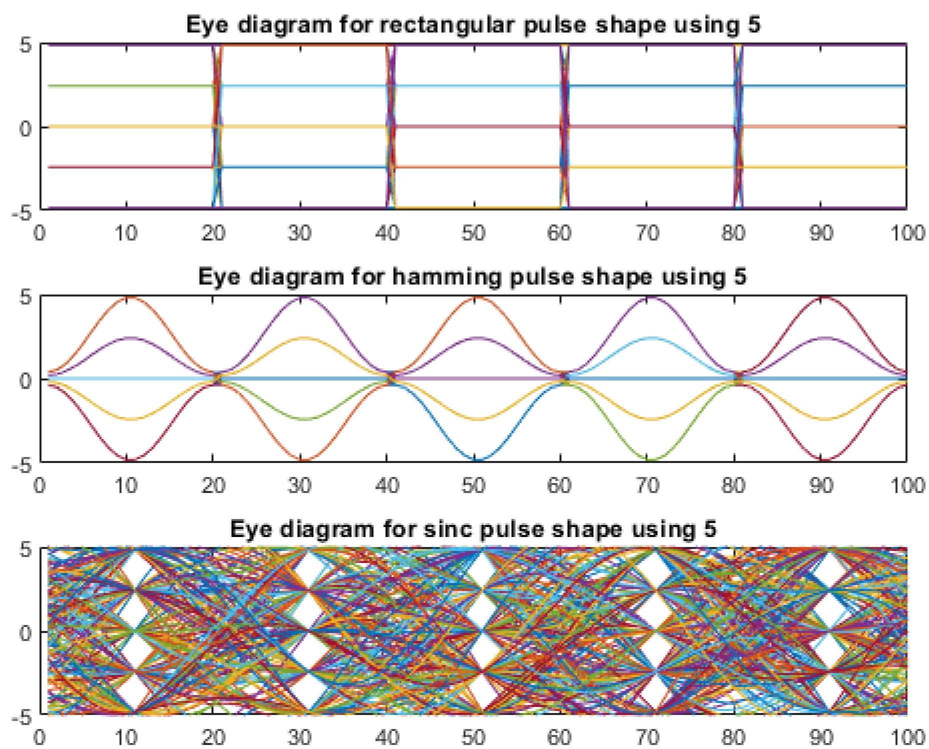
c =

200

```
c =

   200


c =

   200
```


Eye diagram for rectangular pulse shape using 2
Eye diagram for hamming pulse shape using 2
Eye diagram for sinc pulse shape using 2


Eye diagram for rectangular pulse shape using 3
Eye diagram for hamming pulse shape using 3
Eye diagram for sinc pulse shape using 3

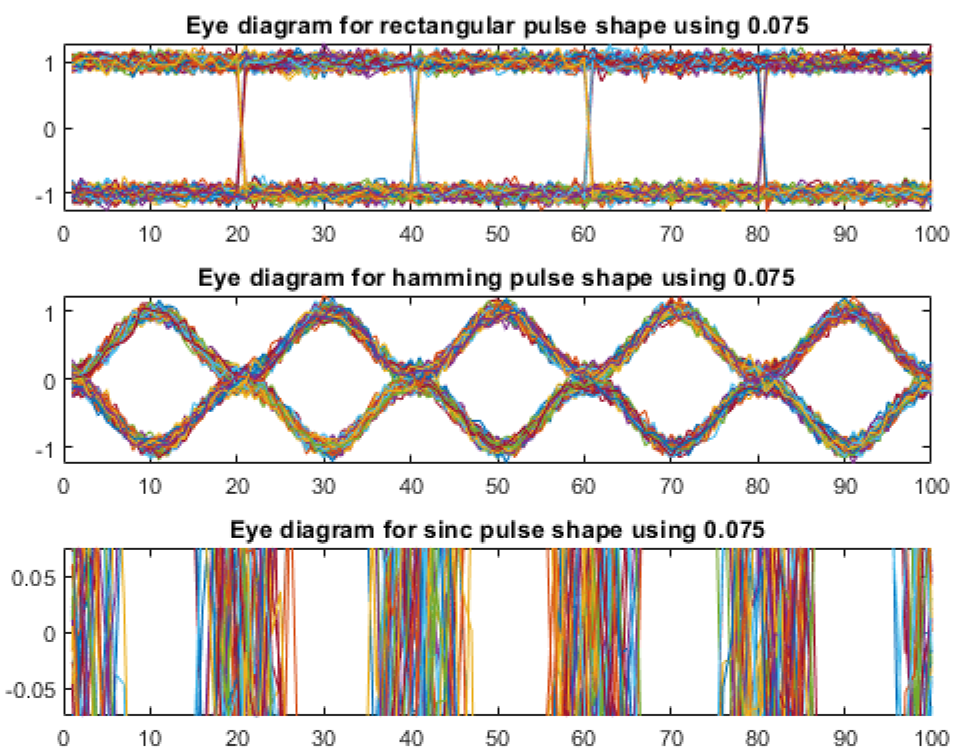

Eye diagram for rectangular pulse shape using 5
Eye diagram for hamming pulse shape using 5
Eye diagram for sinc pulse shape using 5

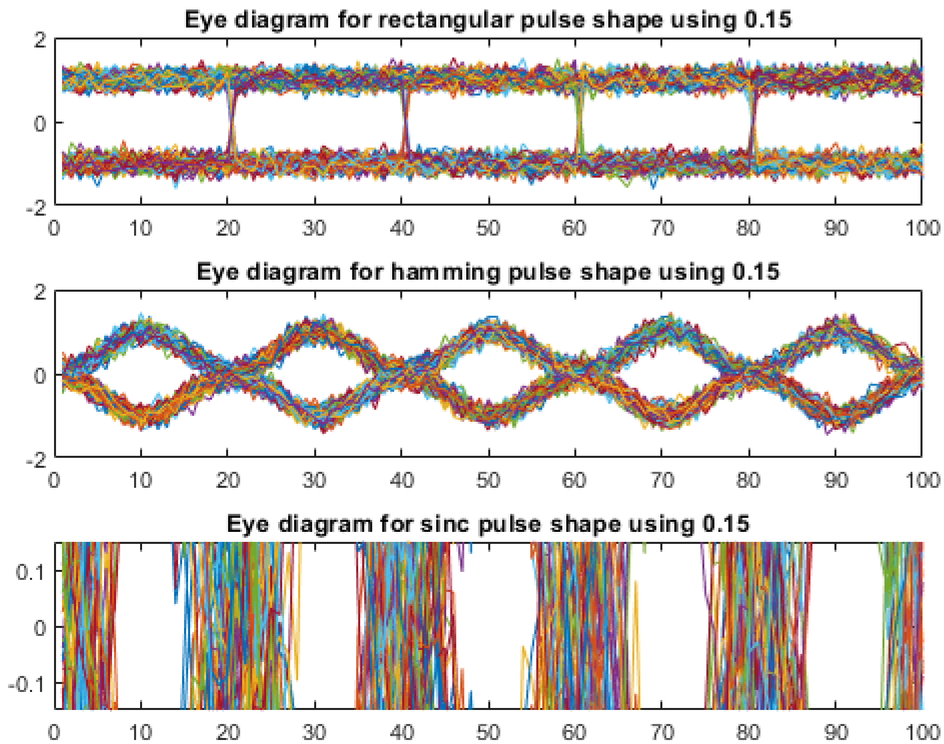
Exercise 11.4

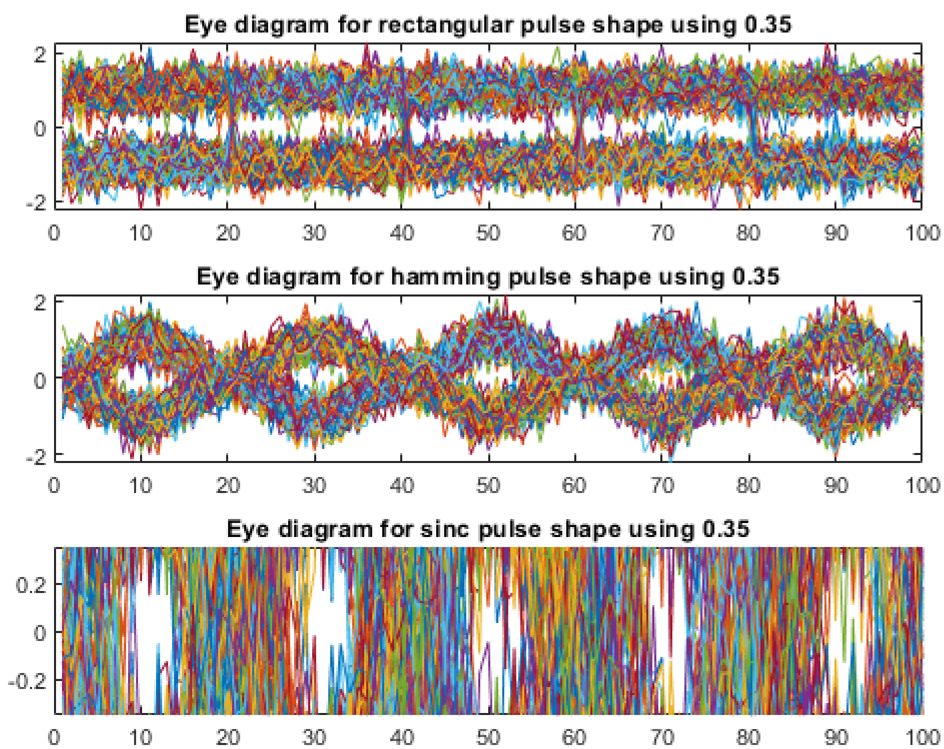
```
random_values = [0.075,0.15, 0.35 , 0.45, 0.55];
for iter = 1:5
    figure                              % used to plot figure eyediag3
    N=1000; m=pam(N,2,1);                  % random +/-1 signal of length N
    M=20; mup=zeros(1,N*M); mup(1:M:N*M)=m;  % oversampling by factor
 of M
    ps=ones(1,M);                                    % square pulse width M
    x=filter(ps,1,mup);                % convolve pulse shape with mup
    x=x+random_values(iter)*randn(size(x));
    neye=5;
    c=floor(length(x)/(neye*M))
    xp=x(N*M-neye*M*c+1:N*M);          % dont plot transients at start
    q=reshape(xp,neye*M,c);            % plot in clusters of size
 5*Mt=(1:198)/50+1;
    subplot(3,1,1), plot(q)
    title(['Eye diagram for rectangular pulse shape using
 ',num2str(random_values(iter))])

    N=1000; m=pam(N,2,1);                  % random +/-1 signal of length N
    M=20; mup=zeros(1,N*M); mup(1:M:N*M)=m;  % oversampling by factor
 of M
    ps=hamming(M);                                   % square pulse width M
    x=filter(ps,1,mup);                % convolve pulse shape with mup
    x=x+random_values(iter)*randn(size(x));
    neye=5;
    c=floor(length(x)/(neye*M))
    xp=x(N*M-neye*M*c+1:N*M);          % dont plot transients at start
    q=reshape(xp,neye*M,c);            % plot in clusters of size
 5*Mt=(1:198)/50+1;
    subplot(3,1,2), plot(q)
    title(['Eye diagram for hamming pulse shape using
 ',num2str(random_values(iter))])

    N=1000; m=pam(N,2,1);                  % random +/-1 signal of length N
    M=20; mup=zeros(1,N*M); mup(1:M:N*M)=m;  % oversampling by factor
 of M
    L=10; ps=srrc(L,0,M,50);
    ps=ps/max(ps);          % sinc pulse shape L symbols wide
    x=filter(ps,1,mup);     % convolve pulse shape with mup
    x=x+random_values(iter)*randn(size(x));
    neye=5;
    c=floor(length(x)/(neye*M))
    xp=x(N*M-neye*M*(c-3)+1:N*M);  % dont plot transients at start
    q=reshape(xp,neye*M,c-3);      % plot in clusters of size
 5*Mt=(1:198)/50+1;
    subplot(3,1,3), plot(q)
    axis([0,100,random_values(iter) *-1,random_values(iter)])
    title(['Eye diagram for sinc pulse shape using
 ',num2str(random_values(iter))])
end
```

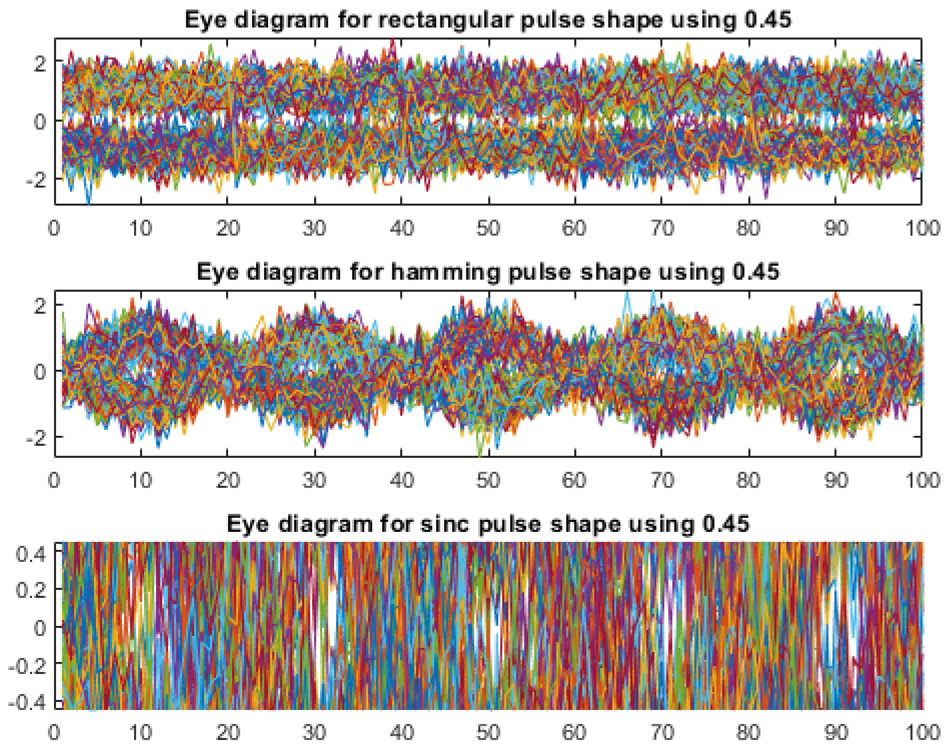
```
% ANSWER
% It appears that the largest value of v is around 0.5, anything after
 that
% and it appears that the "eye" is no longer visible.


c =

   200


c =

   200


c =

   200


c =

   200


c =

   200


c =

   200


c =

   200


c =

   200


c =

   200


c =

   200
```

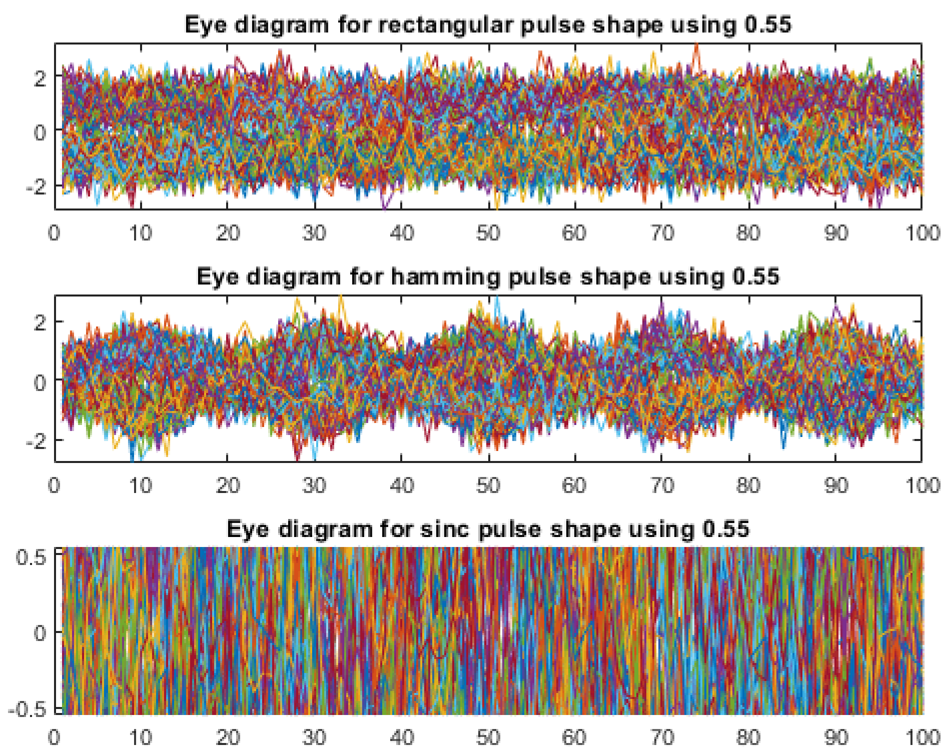
```
c =

   200


c =

   200


c =

   200


c =

   200


c =

   200
```




Eye diagram for rectangular pulse shape using 0.15
Eye diagram for hamming pulse shape using 0.15
Eye diagram for sinc pulse shape using 0.15


Eye diagram for rectangular pulse shape using 0.35
Eye diagram for hamming pulse shape using 0.35
Eye diagram for sinc pulse shape using 0.35


Eye diagram for rectangular pulse shape using 0.45
Eye diagram for hamming pulse shape using 0.45
Eye diagram for sinc pulse shape using 0.45


Eye diagram for rectangular pulse shape using 0.55
Eye diagram for hamming pulse shape using 0.55
Eye diagram for sinc pulse shape using 0.55

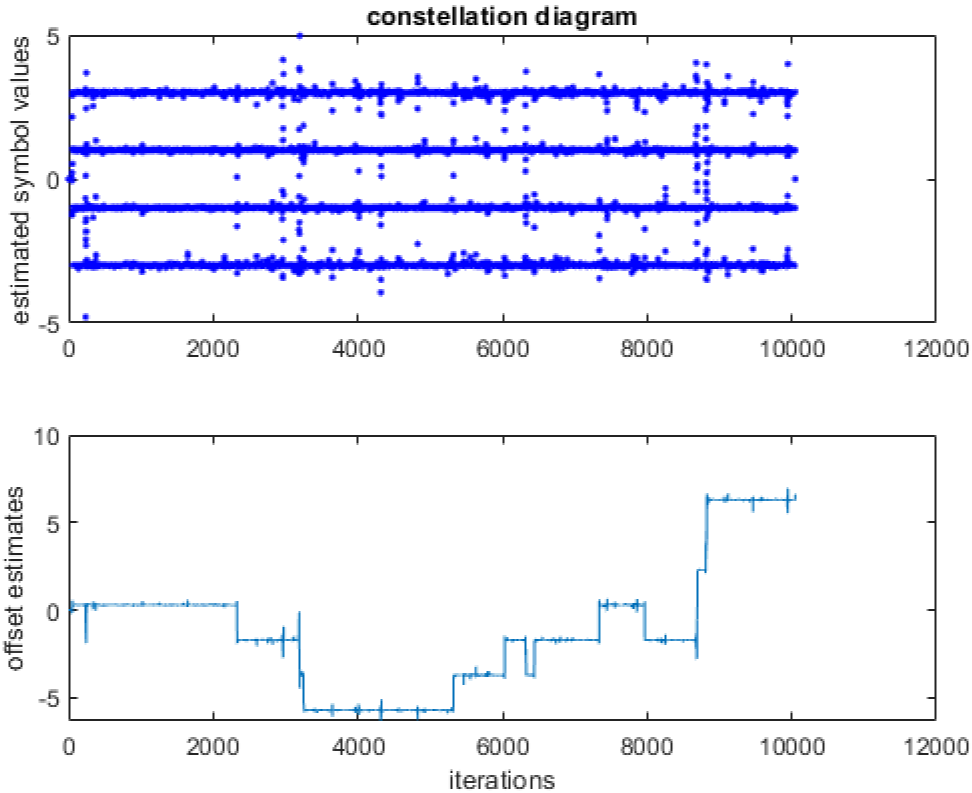
Exercise 12.1 a

clockrecDD.m: clock recovery minimizing 4-PAM cluster variance to minimize J(tau) = (Q(x(kT/M+tau))-x(kT/M+tau))^2

```
% prepare transmitted signal
n=10000;                          % number of data points
m=2;                              % oversampling factor
beta=0.3;                         % rolloff parameter for srrc
l=50;                             % 1/2 length of pulse shape (in
 symbols)
chan=[1];                         % T/m "channel"
toffset=-0.3;                     % initial timing offset
pulshap=srrc(l,beta,m,toffset);   % srrc pulse shape with timing offset
s=pam(n,4,5);                     % random data sequence with var=5
sup=zeros(1,n*m);                 % upsample the data by placing...
sup(1:m:n*m)=s;                   % ... m-1 zeros between each data
 point
hh=conv(pulshap,chan);            % ... and pulse shape
r=conv(hh,sup);                   % ... to get received signal
matchfilt=srrc(l,beta,m,0);       % matched filter = srrc pulse shape
x=conv(r,matchfilt);              % convolve signal with matched filter

% clock recovery algorithm
tnow=l*m+1; tau=0; xs=zeros(1,n);   % initialize variables
tausave=zeros(1,n); tausave(1)=tau; i=0;
mu_values = [0.005, 0.01, 0.02, 0.05]
mu=2;                             % algorithm stepsize
delta=0.1;                          % time for derivative
while tnow<length(x)-2*l*m          % run iteration
  i=i+1;
  xs(i)=interpsinc(x,tnow+tau,l);   % interp value at tnow+tau
  x_deltap=interpsinc(x,tnow+tau+delta,l); % value to right
  x_deltam=interpsinc(x,tnow+tau-delta,l); % value to left
  dx=x_deltap-x_deltam;             % numerical derivative
  qx=quantalph(xs(i),[-3,-1,1,3]);  % quantize to alphabet
  tau=tau+mu*dx*(qx-xs(i));         % alg update: DD
  tnow=tnow+m; tausave(i)=tau;      % save for plotting
end

% plot results
figure();
subplot(2,1,1), plot(xs(1:i-2),'b.')        % plot constellation
 diagram
title('constellation diagram');
ylabel('estimated symbol values')
subplot(2,1,2), plot(tausave(1:i-2))        % plot trajectory of tau
ylabel('offset estimates'), xlabel('iterations')

%ANSWER
% It appears that mu seems the affect the amount of noise present in
 the
```

```
% signal, around 0.2 noise can start to appear visible. However if mu
 goes
% as large as 2 it no longer converges to the correct value of the
 offset.
```

```
mu_values =

    0.0050    0.0100    0.0200    0.0500
```

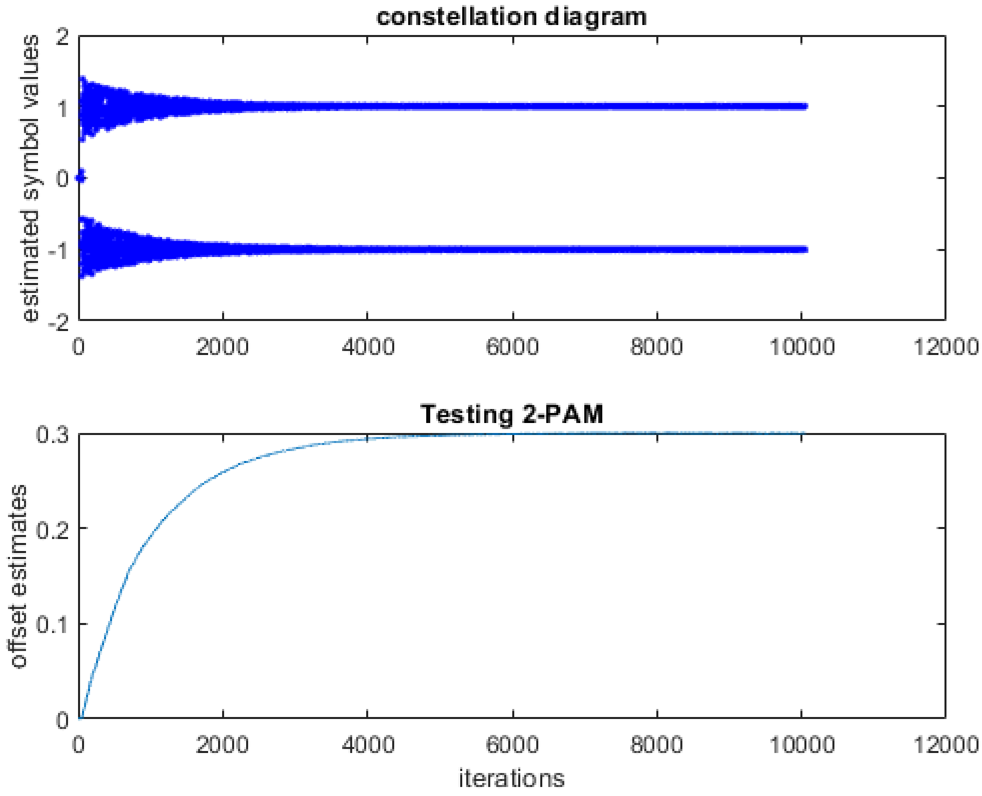


Exercise 12.1 b Testing 2-PAM

prepare transmitted signal

```
n=10000;                          % number of data points
m=2;                              % oversampling factor
beta=0.3;                         % rolloff parameter for srrc
l=50;                             % 1/2 length of pulse shape (in
 symbols)
chan=[1];                         % T/m "channel"
toffset=-0.3;                     % initial timing offset
pulshap=srrc(l,beta,m,toffset);   % srrc pulse shape with timing offset
s=pam(n,2,1);                     % random data sequence with var=5
sup=zeros(1,n*m);                 % upsample the data by placing...
```

```
sup(1:m:n*m)=s;                 % ... m-1 zeros between each data
 point
hh=conv(pulshap,chan);          % ... and pulse shape
r=conv(hh,sup);                 % ... to get received signal
matchfilt=srrc(l,beta,m,0);     % matched filter = srrc pulse shape
x=conv(r,matchfilt);            % convolve signal with matched filter

% clock recovery algorithm
tnow=l*m+1; tau=0; xs=zeros(1,n);   % initialize variables
tausave=zeros(1,n); tausave(1)=tau; i=0;
mu=0.01;                            % algorithm stepsize
delta=0.1;                          % time for derivative
while tnow<length(x)-2*l*m          % run iteration
  i=i+1;
  xs(i)=interpsinc(x,tnow+tau,l);   % interp value at tnow+tau
  x_deltap=interpsinc(x,tnow+tau+delta,l); % value to right
  x_deltam=interpsinc(x,tnow+tau-delta,l); % value to left
  dx=x_deltap-x_deltam;             % numerical derivative
  qx=quantalph(xs(i),[-3,-1,1,3]);  % quantize to alphabet
  tau=tau+mu*dx*(qx-xs(i));         % alg update: DD
  tnow=tnow+m; tausave(i)=tau;      % save for plotting
end

% plot results
figure();
subplot(2,1,1), plot(xs(1:i-2),'b.')        % plot constellation
 diagram
title('constellation diagram');
ylabel('estimated symbol values')
subplot(2,1,2), plot(tausave(1:i-2))        % plot trajectory of tau
ylabel('offset estimates'), xlabel('iterations')
title('Testing 2-PAM');
```

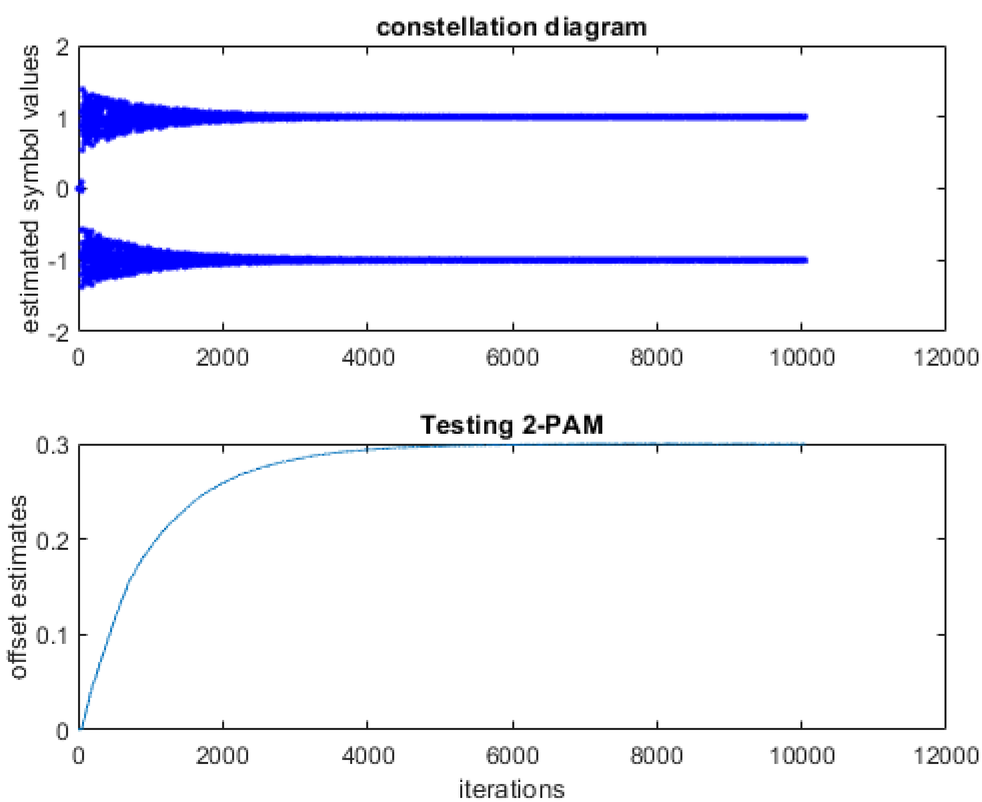


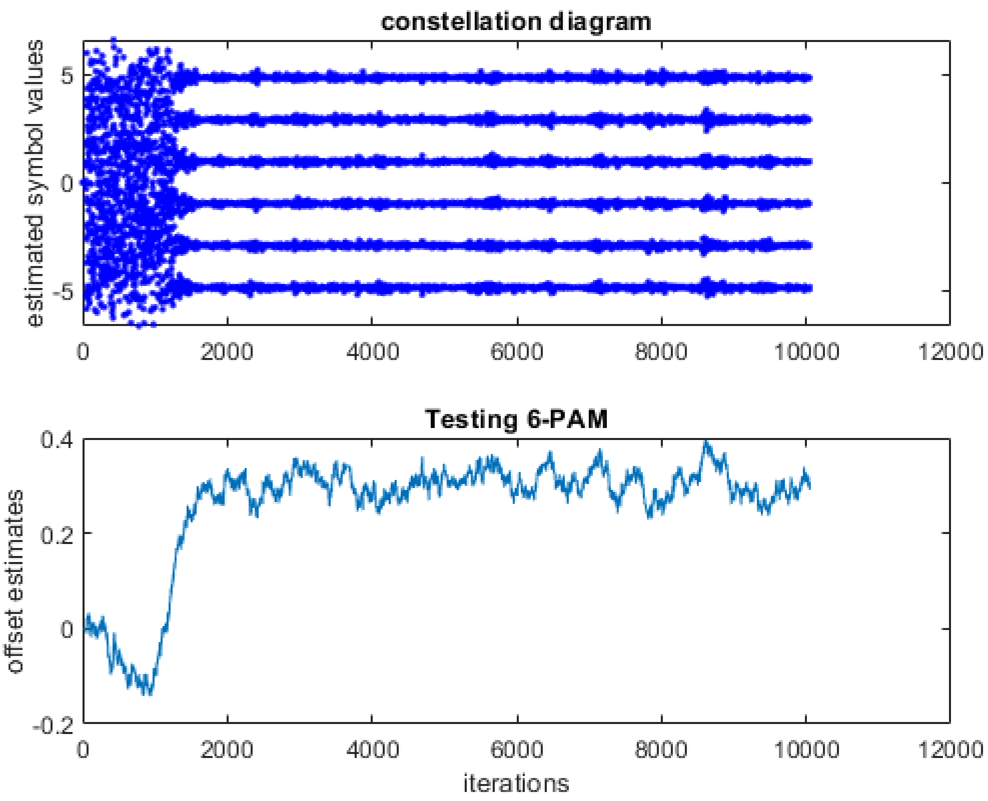
Exercise 12.1 b Testing 6-PAM

prepare transmitted signal

```
n=10000;                          % number of data points
m=2;                              % oversampling factor
beta=0.3;                         % rolloff parameter for srrc
l=50;                             % 1/2 length of pulse shape (in
 symbols)
chan=[1];                         % T/m "channel"
toffset=-0.3;                     % initial timing offset
pulshap=srrc(l,beta,m,toffset);   % srrc pulse shape with timing offset
s=pam(n,6,11);                     % random data sequence with var=5
sup=zeros(1,n*m);                 % upsample the data by placing...
sup(1:m:n*m)=s;                   % ... m-1 zeros between each data
 point
hh=conv(pulshap,chan);            % ... and pulse shape
r=conv(hh,sup);                   % ... to get received signal
matchfilt=srrc(l,beta,m,0);       % matched filter = srrc pulse shape
x=conv(r,matchfilt);              % convolve signal with matched filter

% clock recovery algorithm
tnow=l*m+1; tau=0; xs=zeros(1,n);   % initialize variables
tausave=zeros(1,n); tausave(1)=tau; i=0;
mu=0.01;                            % algorithm stepsize
delta=0.1;                          % time for derivative
```

```
while tnow<length(x)-2*l*m          % run iteration
  i=i+1;
  xs(i)=interpsinc(x,tnow+tau,l);   % interp value at tnow+tau
  x_deltap=interpsinc(x,tnow+tau+delta,l); % value to right
  x_deltam=interpsinc(x,tnow+tau-delta,l); % value to left
  dx=x_deltap-x_deltam;             % numerical derivative
  qx=quantalph(xs(i),[-3,-1,1,3]);  % quantize to alphabet
  tau=tau+mu*dx*(qx-xs(i));         % alg update: DD
  tnow=tnow+m; tausave(i)=tau;      % save for plotting
end

% plot results
figure();
subplot(2,1,1), plot(xs(1:i-2),'b.')        % plot constellation
 diagram
title('constellation diagram');
ylabel('estimated symbol values')
subplot(2,1,2), plot(tausave(1:i-2))        % plot trajectory of tau
ylabel('offset estimates'), xlabel('iterations')
title('Testing 6-PAM');
 % Answer
 % It appears that the more levels are used to represent a signal more
 % noise appears in the signal as well.
```





Exercise 12.2

prepare transmitted signal

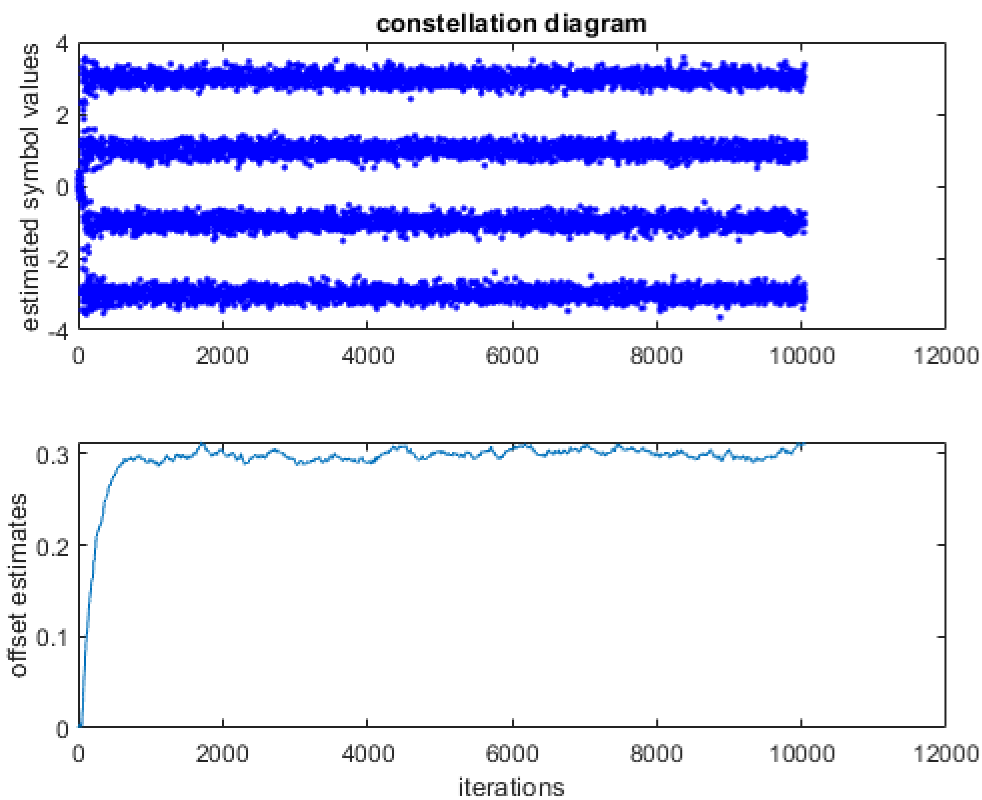
```
n=10000;                          % number of data points
m=2;                              % oversampling factor
beta=0.3;                         % rolloff parameter for srrc
l=50;                             % 1/2 length of pulse shape (in
 symbols)
chan=[1];                         % T/m "channel"
toffset=-0.3;                     % initial timing offset
x = (m+toffset)/2:beta:(m+toffset)/2;
%pulshap = 1*rectpuls(x, 1);
pulshap = rect_pulse_maker(1, beta, m, toffset);
s=pam(n,4,5);                     % random data sequence with var=5
sup=zeros(1,n*m);                 % upsample the data by placing...
sup(1:m:n*m)=s;                   % ... m-1 zeros between each data
 point
hh=conv(pulshap,chan);            % ... and pulse shape
r=conv(hh,sup);                   % ... to get received signal
%matchfilt=srrc(l,beta,m,0);       % matched filter = srrc pulse shape
matchfilt = rect_pulse_maker(1, beta, m, 0);
x=conv(r,matchfilt);              % convolve signal with matched filter

% clock recovery algorithm
tnow=l*m+1; tau=0; xs=zeros(1,n);   % initialize variables
```

```
tausave=zeros(1,n); tausave(1)=tau; i=0;
mu=0.01;                            % algorithm stepsize
delta=0.1;                          % time for derivative
while tnow<length(x)-2*l*m          % run iteration
  i=i+1;
  xs(i)=interpsinc(x,tnow+tau,l);   % interp value at tnow+tau
  x_deltap=interpsinc(x,tnow+tau+delta,l); % value to right
  x_deltam=interpsinc(x,tnow+tau-delta,l); % value to left
  dx=x_deltap-x_deltam;             % numerical derivative
  qx=quantalph(xs(i),[-3,-1,1,3]);  % quantize to alphabet
  tau=tau+mu*dx*(qx-xs(i));         % alg update: DD
  tnow=tnow+m; tausave(i)=tau;      % save for plotting
end

% ANSWER
% It appears that a rectuangular pulse has very poor spectral
 properites
% and because it varies so high from the maxima you end with a lot of
% interference. A raised consine can control the excess bandwidth but
 that
% is hard to do in a rectangular pulse leading to a noisy signal that
 does
% not converge properly.
```

Exercise 12.3

```
% clockrecDD.m: clock recovery minimizing 4-PAM cluster variance
% to minimize J(tau) = (Q(x(kT/M+tau))-x(kT/M+tau))^2

% prepare transmitted signal
n=10000;                         % number of data points
m=2;                             % oversampling factor
beta=0.3;                        % rolloff parameter for srrc
l=50;                            % 1/2 length of pulse shape (in
 symbols)
chan=[1];                        % T/m "channel"
toffset=-0.3;                    % initial timing offset
pulshap=srrc(l,beta,m,toffset);  % srrc pulse shape with timing offset
s=pam(n,4,5);                    % random data sequence with var=5
sup=zeros(1,n*m);                % upsample the data by placing...
sup(1:m:n*m)=s;                  % ... m-1 zeros between each data
 point
hh=conv(pulshap,chan);           % ... and pulse shape
r=conv(hh,sup);                  % ... to get received signal
matchfilt=srrc(l,beta,m,0);      % matched filter = srrc pulse shape
x=conv(r,matchfilt);        % convolve signal with matched filter
x=x+0.15*randn(size(x));
% clock recovery algorithm
tnow=l*m+1; tau=0; xs=zeros(1,n);   % initialize variables
tausave=zeros(1,n); tausave(1)=tau; i=0;
mu=0.01;                            % algorithm stepsize
delta=0.1;                          % time for derivative
while tnow<length(x)-2*l*m          % run iteration
```

```
  i=i+1;
  xs(i)=interpsinc(x,tnow+tau,l);   % interp value at tnow+tau
  x_deltap=interpsinc(x,tnow+tau+delta,l); % value to right
  x_deltam=interpsinc(x,tnow+tau-delta,l); % value to left
  dx=x_deltap-x_deltam;             % numerical derivative
  qx=quantalph(xs(i),[-3,-1,1,3]);  % quantize to alphabet
  tau=tau+mu*dx*(qx-xs(i));         % alg update: DD
  tnow=tnow+m; tausave(i)=tau;      % save for plotting
end

% plot results
subplot(2,1,1), plot(xs(1:i-2),'b.')        % plot constellation
 diagram
title('constellation diagram');
ylabel('estimated symbol values')
subplot(2,1,2), plot(tausave(1:i-2))        % plot trajectory of tau
ylabel('offset estimates'), xlabel('iterations')
% ANSWER
% It appears that it stills converges to the correct value however it
 is a
% lot more noisy then before.
```

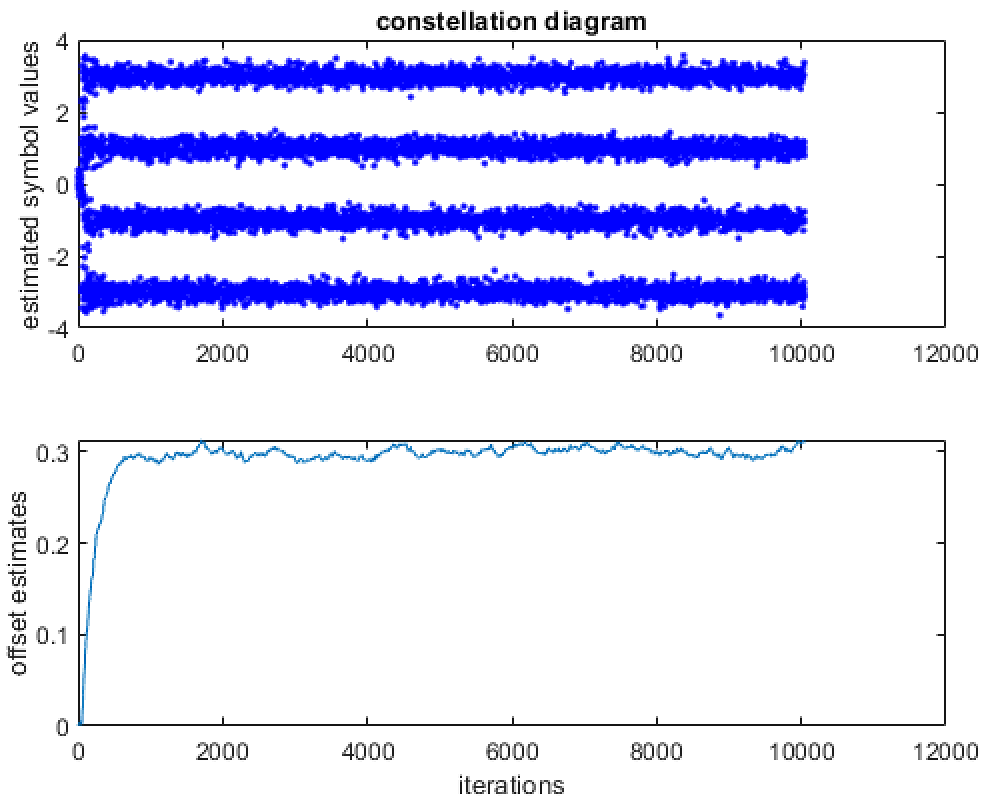


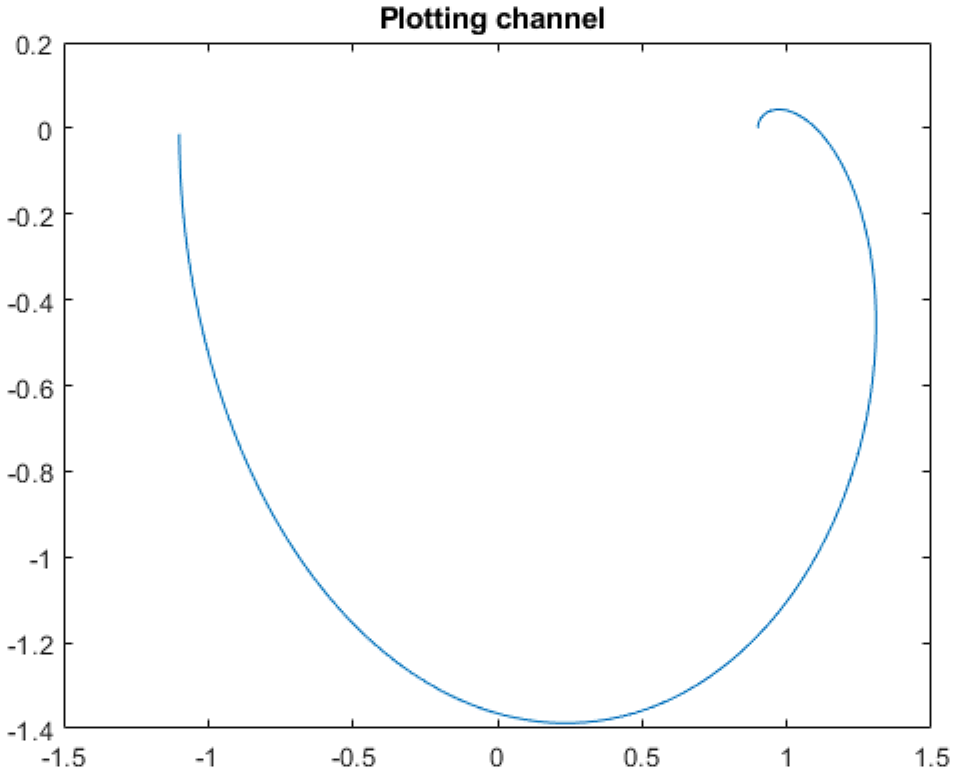
Exercise 13.1

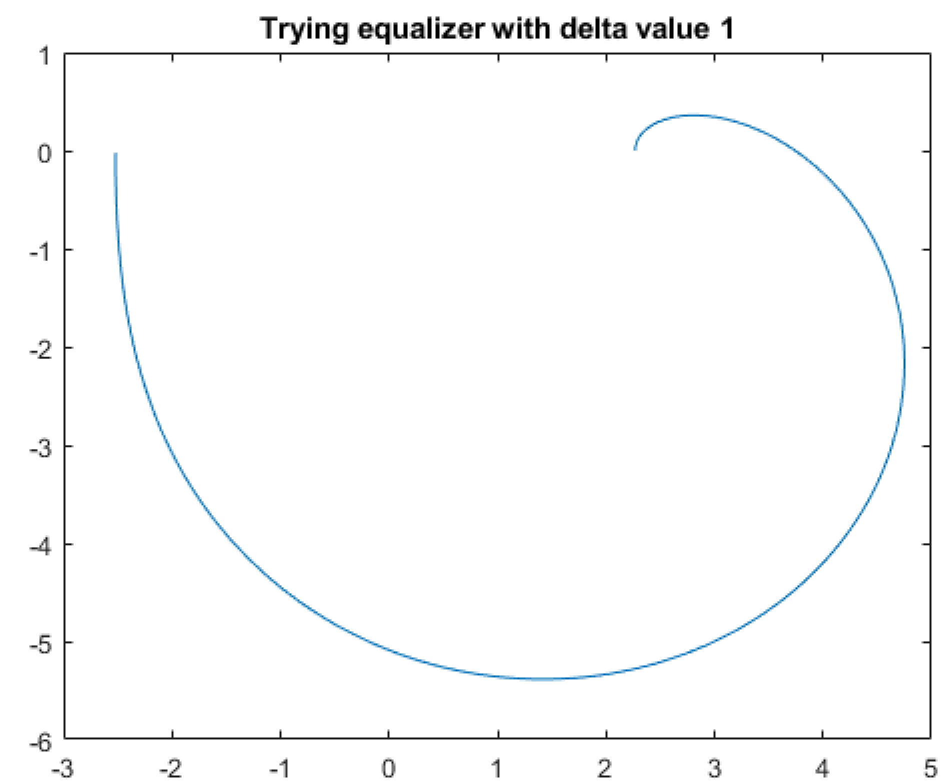
LSequalizer.m find a LS equalizer f for the channel b

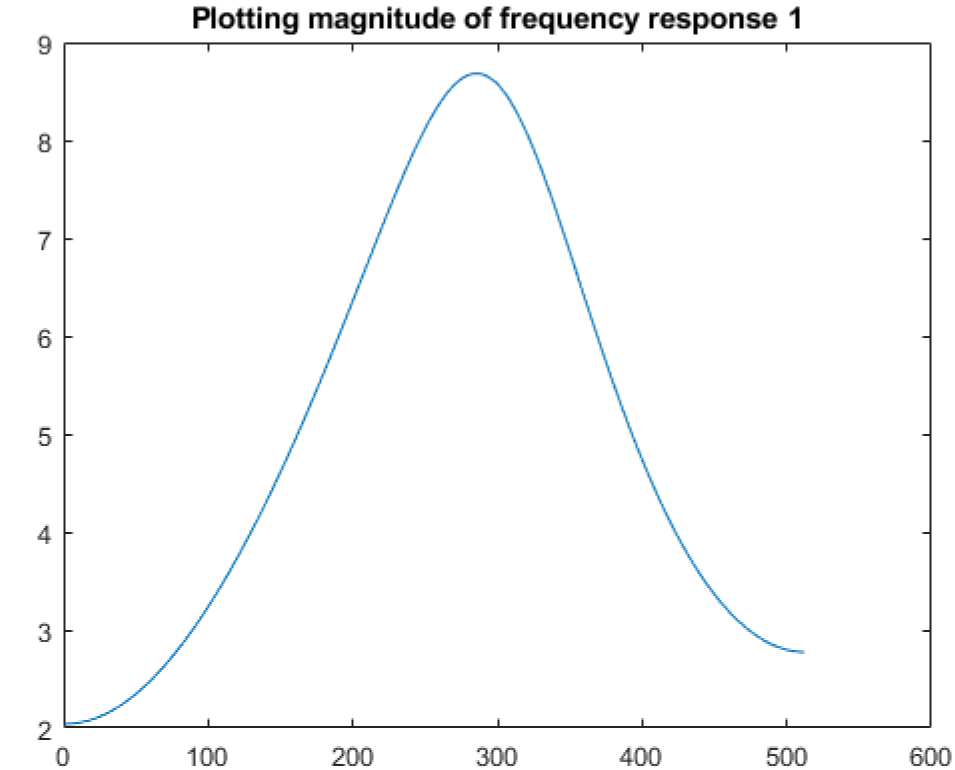
```
 b=[0.5 1 -0.6];                    % define channel
    figure
    plot(freqz(b));
    title('Plotting channel ');
    [hb, wb] = freqz(b);
    m=1000; s=sign(randn(1,m));        % binary source of length m
    r=filter(b,1,s);                   % output of channel
    n=3;                               % length of equalizer - 1
for iter = 1:4
    delta=iter-1;                           % use delay <=n*length(b)
    p=length(r)-delta;
    R=toeplitz(r(n+1:p),r(n+1:-1:1));  % build matrix R
    S=s(n+1-delta:p-delta)';           % and vector S
    f=inv(R'*R)*R'*S;                   % calculate equalizer f
    Jmin=S'*S-S'*R*inv(R'*R)*R'*S      % Jmin for this f and delta
    y=filter(f,1,r);                   % equalizer is a filter
    dec=sign(y);                       % quantize and find errors
    err=0.5*sum(abs(dec(delta+1:m)-s(1:m-delta)))
    figure
    plot(freqz(b,f))
    title(['Trying equalizer with delta value ', num2str(iter)])

    [h,w] = freqz(b,f);
    figure
    plot(abs(h.*hb))
    title(['Plotting magnitude of frequency response ',
 num2str(iter)]);
end
% ANSWER
% Even though the magnitude of teh channel and the equalizers seems to
 be
% opposite they never really cancel out and reach unity.


Jmin =

  816.1689


err =

   395


Jmin =

  134.4687


err =

     0
```

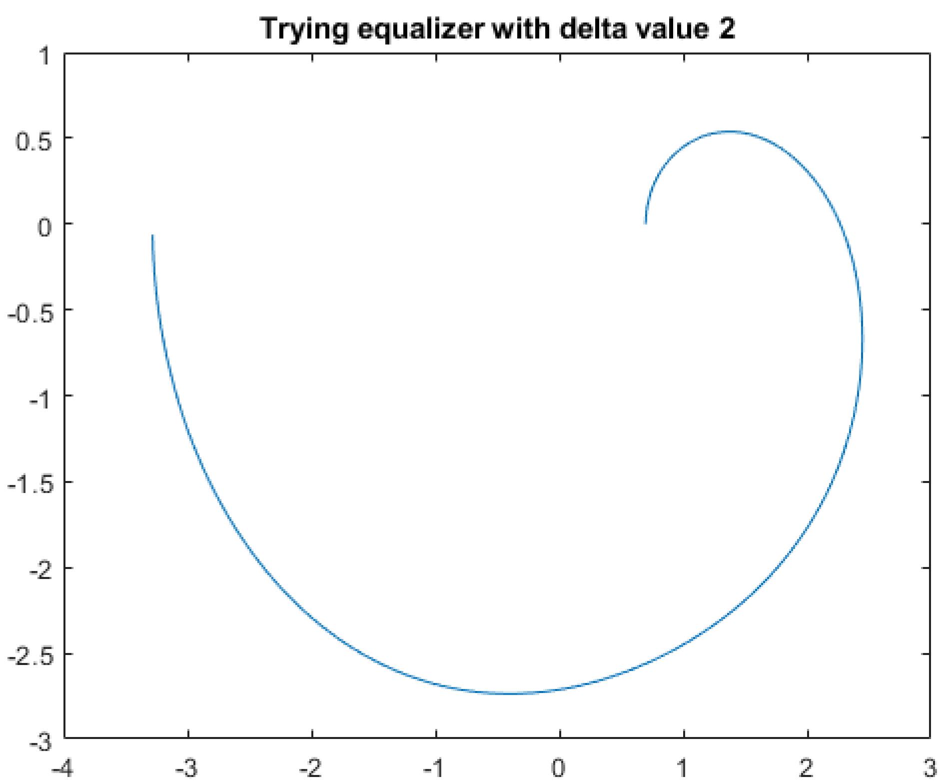
```
Jmin =

   31.0168


err =

     0


Jmin =

   44.8434


err =

     0
```

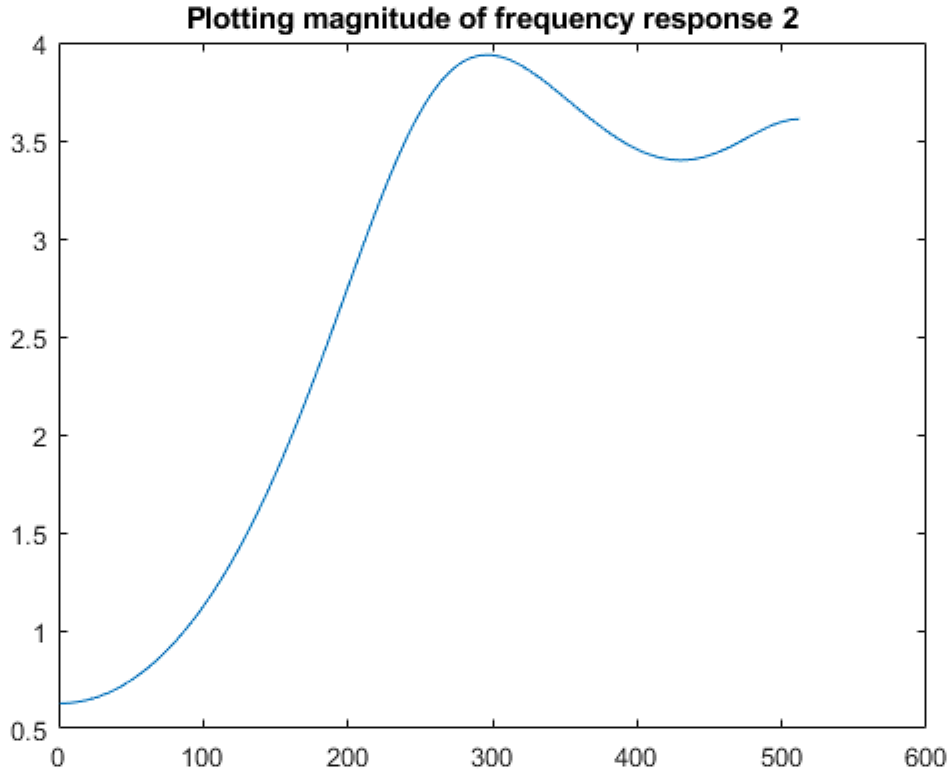


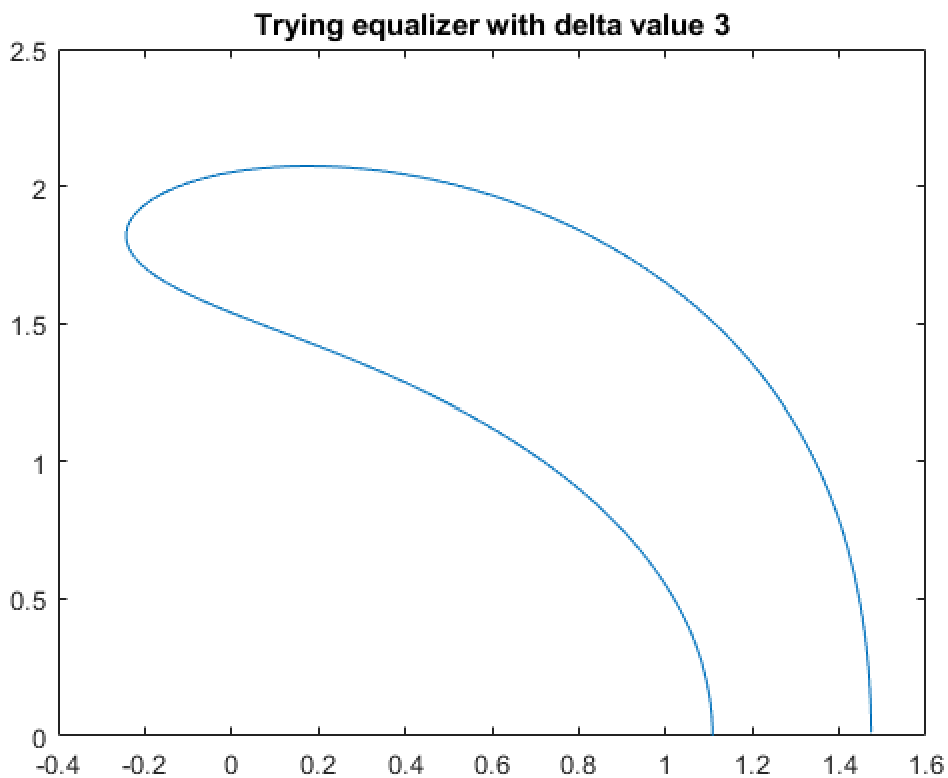

Plotting channel
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.5
-1
-0.5
0
0.5
1
1.5

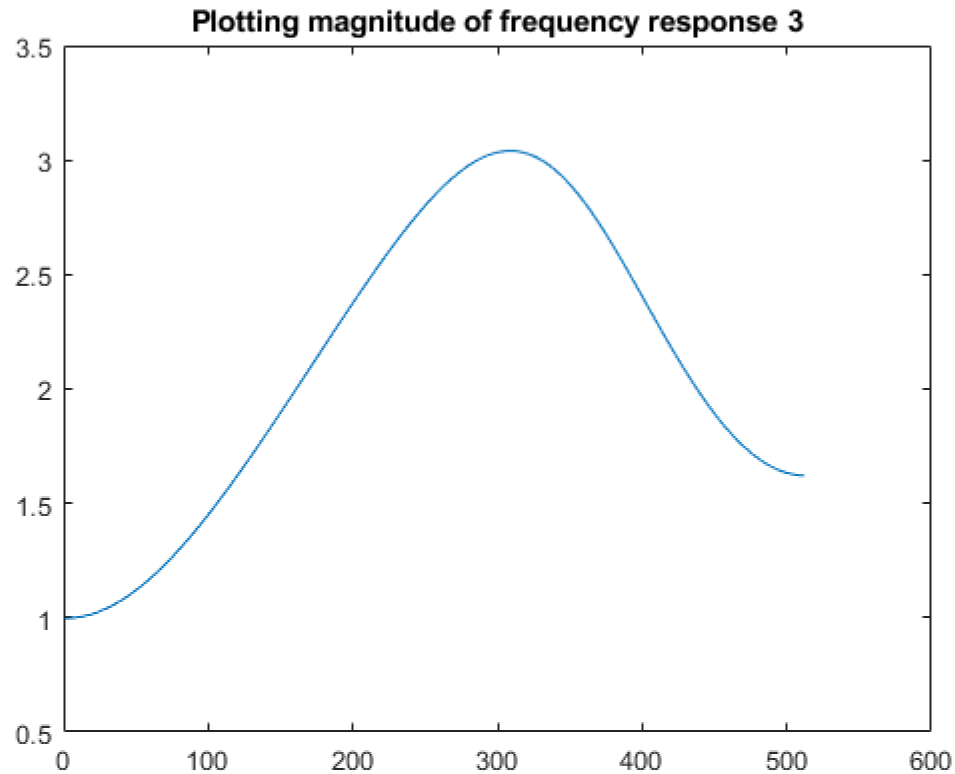

Trying equalizer with delta value 1
1
0
-1
-2
-3
-4
-5
-6
-3
-2
-1
0
1
2
3
4
5

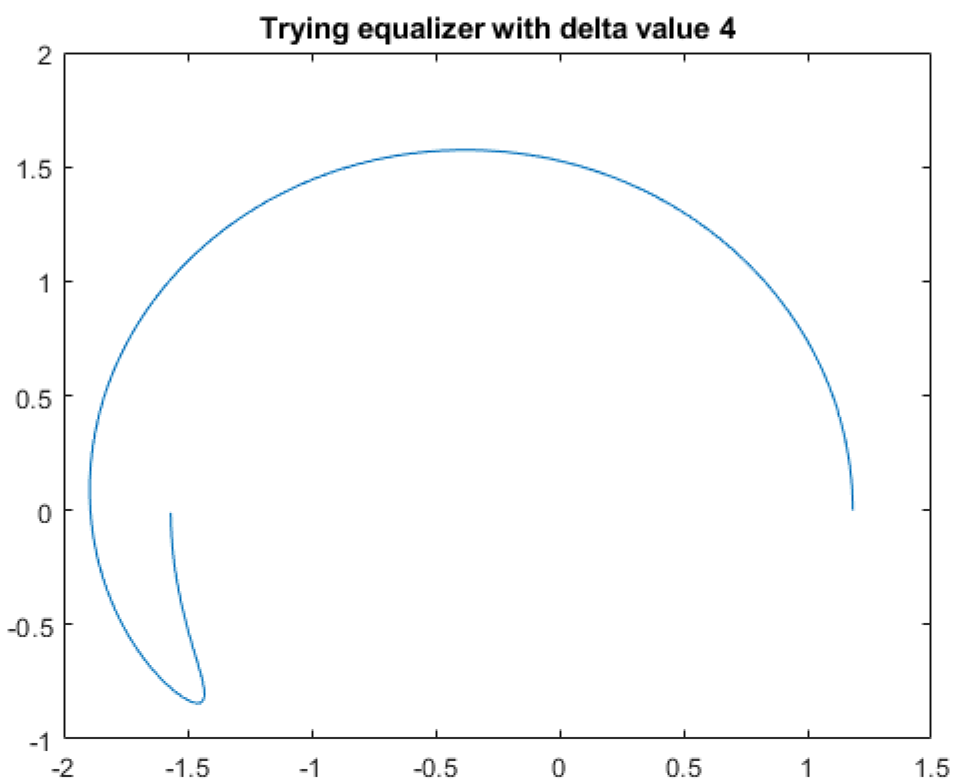

Plotting magnitude of frequency response 1
9
8
7
6
5
4
3
2
0
100
200
300
400
500
600

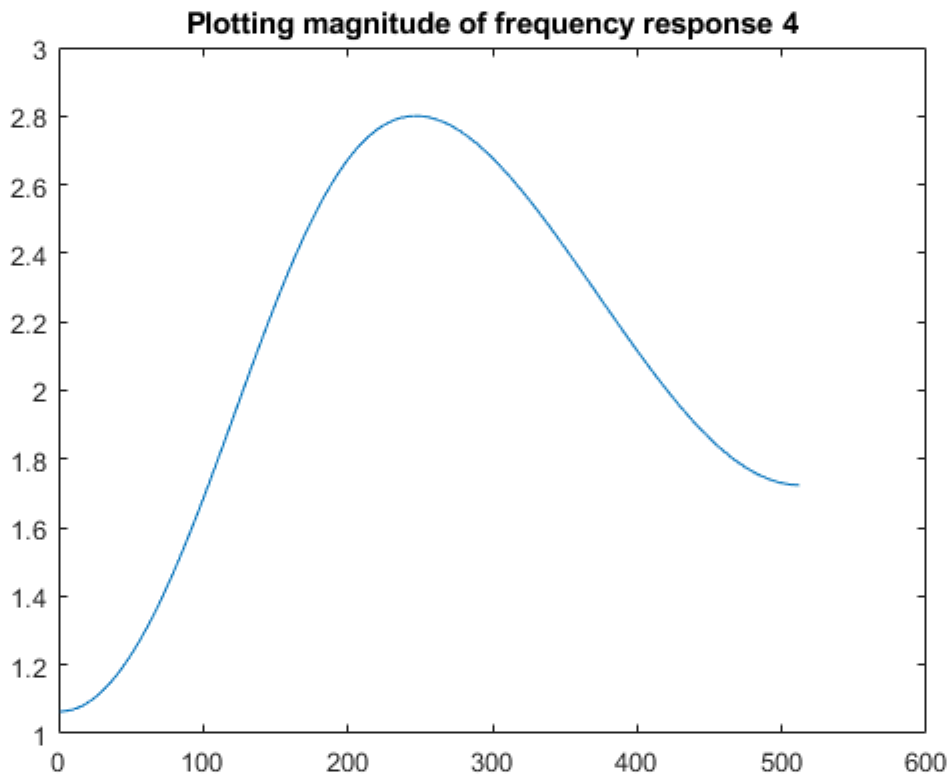

Trying equalizer with delta value 2
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-4
-3
-2
-1
0
1
2
3


Plotting magnitude of frequency response 2
4
3.5
3
2.5
2
1.5
1
0.5
0
100
200
300
400
500
600


Trying equalizer with delta value 3
2.5
2
1.5
1
0.5
0
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6


Plotting magnitude of frequency response 3
3.5
3
2.5
2
1.5
1
0.5
0
100
200
300
400
500
600


Trying equalizer with delta value 4
2
1.5
1
0.5
0
-0.5
-1
-2
-1.5
-1
-0.5
0
0.5
1
1.5



Exercise 13.2 a

LSequalizer.m find a LS equalizer f for the channel b

```
b=[0.5 1 -0.6];                     % define channel
m=1000; s=sign(randn(1,m));         % binary source of length m
sd = 0.3;
r=filter(b,1,s)+sd*randn(size(s));                      % output of
 channel
n=3;                                % length of equalizer - 1
delta=2;                            % use delay <=n*length(b)
p=length(r)-delta;
R=toeplitz(r(n+1:p),r(n+1:-1:1));   % build matrix R
S=s(n+1-delta:p-delta)';            % and vector S
f=inv(R'*R)*R'*S                    % calculate equalizer f
Jmin=S'*S-S'*R*inv(R'*R)*R'*S       % Jmin for this f and delta
y=filter(f,1,r);                    % equalizer is a filter
dec=sign(y);                        % quantize and find errors
err=0.5*sum(abs(dec(delta+1:m)-s(1:m-delta)))
% ANSWER
% It appears that the value where error first start appearing is
 around 0.3


f =
```

```
   -0.2706
    0.6171
    0.2828
    0.1253


Jmin =

   79.2882


err =

     0
```

Exercise 13.2b

```
x = []
for sd = 0.1:0.05:1
    b=[0.5 1 -0.6];                     % define channel
    m=1000; s=sign(randn(1,m));         % binary source of length m
    r=filter(b,1,s)+sd*randn(size(s));                  % output of
 channel
    n=3;                                % length of equalizer - 1
    delta=2;                            % use delay <=n*length(b)
    p=length(r)-delta;
    R=toeplitz(r(n+1:p),r(n+1:-1:1));  % build matrix R
    S=s(n+1-delta:p-delta)';            % and vector S
    f=inv(R'*R)*R'*S                    % calculate equalizer f
    Jmin=S'*S-S'*R*inv(R'*R)*R'*S       % Jmin for this f and delta
    y=filter(f,1,r);                    % equalizer is a filter
    dec=sign(y);                        % quantize and find errors
    err=0.5*sum(abs(dec(delta+1:m)-s(1:m-delta)))
    x = [x, Jmin]

end

figure
plot(0.1:0.05:1, x)
title('Jmin vs sd plot')
```

```
x =

     []


f =

   -0.2655
    0.6455
    0.3088
```

```
    0.1387


Jmin =

   39.1061


err =

     0


x =

   39.1061


f =

   -0.2741
    0.6383
    0.3082
    0.1452


Jmin =

   44.6709


err =

     0


x =

   39.1061   44.6709


f =

   -0.2778
    0.6249
    0.2920
    0.1281


Jmin =

   55.0269
```

```
err =

     0


x =

   39.1061   44.6709   55.0269


f =

   -0.2672
    0.6152
    0.2911
    0.1331


Jmin =

   64.2515


err =

     0


x =

   39.1061   44.6709   55.0269   64.2515


f =

   -0.2551
    0.6033
    0.2987
    0.1274


Jmin =

   84.2713


err =

     0


x =

   39.1061   44.6709   55.0269   64.2515   84.2713
```

```
f =

   -0.2744
    0.5907
    0.2580
    0.1113


Jmin =

   98.5940


err =

     0


x =

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940


f =

   -0.2656
    0.5859
    0.2831
    0.1118


Jmin =

  117.8857


err =

     8


x =

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857


f =

   -0.2612
    0.5644
    0.2828
    0.0944
```

```
Jmin =

  142.1911


err =

     6


x =

  Columns 1 through 7

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857

  Column 8

  142.1911


f =

   -0.2485
    0.5480
    0.2589
    0.1110


Jmin =

  169.2462


err =

    16


x =

  Columns 1 through 7

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857

  Columns 8 through 9

  142.1911  169.2462


f =
```

```
   -0.2448
    0.5221
    0.2601
    0.0855


Jmin =

  166.8200


err =

    13


x =

  Columns 1 through 7

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857

  Columns 8 through 10

  142.1911  169.2462  166.8200


f =

   -0.2282
    0.5234
    0.2492
    0.0974


Jmin =

  202.2782


err =

    21


x =

  Columns 1 through 7

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857

  Columns 8 through 11

  142.1911  169.2462  166.8200  202.2782
```

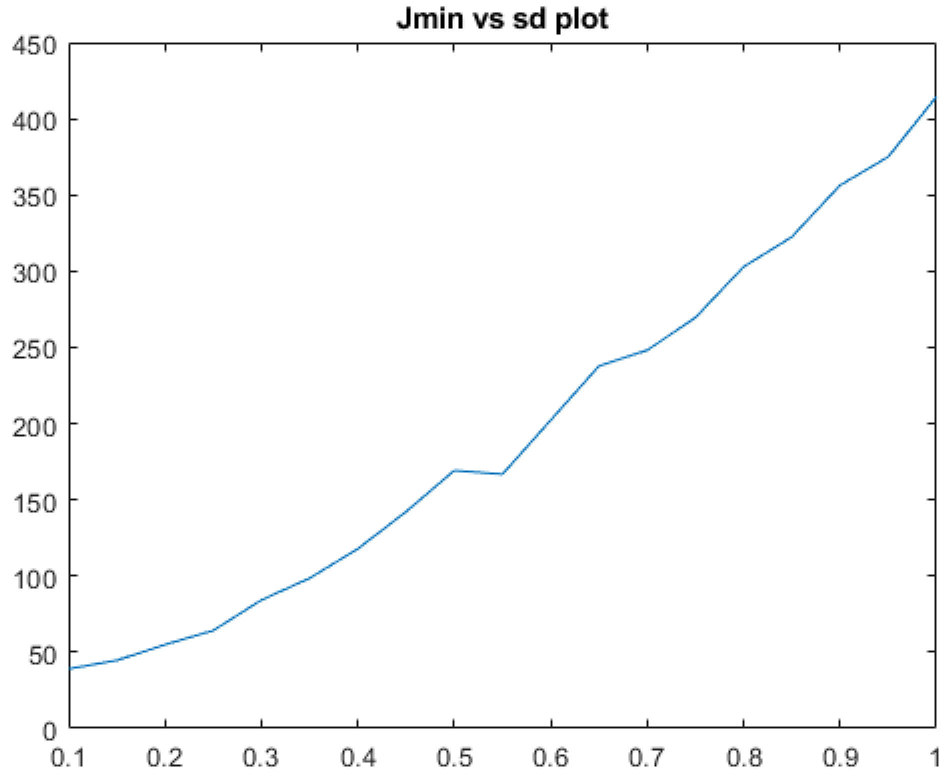
```
f =

   -0.2251
    0.4995
    0.2491
    0.0808


Jmin =

  237.7480


err =

    34


x =

  Columns 1 through 7

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857

  Columns 8 through 12

  142.1911  169.2462  166.8200  202.2782  237.7480


f =

   -0.2537
    0.4999
    0.2117
    0.0819


Jmin =

  248.1171


err =

    40


x =

  Columns 1 through 7

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857
```

```
  Columns 8 through 13

  142.1911  169.2462  166.8200  202.2782  237.7480  248.1171


f =

   -0.2199
    0.4590
    0.2379
    0.0784


Jmin =

  269.5336


err =

    55


x =

  Columns 1 through 7

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857

  Columns 8 through 14

  142.1911  169.2462  166.8200  202.2782  237.7480  248.1171  269.5336


f =

   -0.2254
    0.4366
    0.2161
    0.0649


Jmin =

  302.8387


err =

    60


x =
```

```
  Columns 1 through 7

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857

  Columns 8 through 14

  142.1911  169.2462  166.8200  202.2782  237.7480  248.1171  269.5336

  Column 15

  302.8387


f =

   -0.2080
    0.4357
    0.2247
    0.0714


Jmin =

  322.4758


err =

    77


x =

  Columns 1 through 7

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857

  Columns 8 through 14

  142.1911  169.2462  166.8200  202.2782  237.7480  248.1171  269.5336

  Columns 15 through 16

  302.8387  322.4758


f =

   -0.2076
    0.4092
    0.2084
    0.0780
```

```
Jmin =

  356.3454


err =

    89


x =

  Columns 1 through 7

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857

  Columns 8 through 14

  142.1911  169.2462  166.8200  202.2782  237.7480  248.1171  269.5336

  Columns 15 through 17

  302.8387  322.4758  356.3454


f =

   -0.1994
    0.4034
    0.1798
    0.0521


Jmin =

  375.0249


err =

    97


x =

  Columns 1 through 7

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857

  Columns 8 through 14

  142.1911  169.2462  166.8200  202.2782  237.7480  248.1171  269.5336
```

```
  Columns 15 through 18

  302.8387  322.4758  356.3454  375.0249


f =

   -0.1847
    0.3874
    0.1822
    0.0361


Jmin =

  414.4879


err =

   111


x =

  Columns 1 through 7

   39.1061   44.6709   55.0269   64.2515   84.2713   98.5940  117.8857

  Columns 8 through 14

  142.1911  169.2462  166.8200  202.2782  237.7480  248.1171  269.5336

  Columns 15 through 19

  302.8387  322.4758  356.3454  375.0249  414.4879
```



Exercise 13.2 c

LSequalizer.m find a LS equalizer f for the channel b

```
b=[0.5 1 -0.6];                    % define channel
m=1000; s=sign(randn(1,m));        % binary source of length m
sd = 0.19;
r=filter(b,1,s)+sd*randn(size(s));                  % output of
 channel
n=3;                               % length of equalizer - 1
delta=1;                           % use delay <=n*length(b)
p=length(r)-delta;
R=toeplitz(r(n+1:p),r(n+1:-1:1));  % build matrix R
S=s(n+1-delta:p-delta)';           % and vector S
f=inv(R'*R)*R'*S                   % calculate equalizer f
Jmin=S'*S-S'*R*inv(R'*R)*R'*S      % Jmin for this f and delta
y=filter(f,1,r);                   % equalizer is a filter
dec=sign(y);                       % quantize and find errors
err=0.5*sum(abs(dec(delta+1:m)-s(1:m-delta)))


%ANSWER
% It appears that when sd is around 0.2 error appears


f =
```

```
    0.6471
    0.3528
    0.1593
    0.0631


Jmin =

  163.2112


err =

     0
```

Exercise 13.2 d

ANSWER The first one with the larger delay seems to perform better, because the value of sd can be larger than the one with the smaller delay. This means it can take larger noise values and is therefore a more robust system.

Published with MATLAB® R2019b