Table of Contents

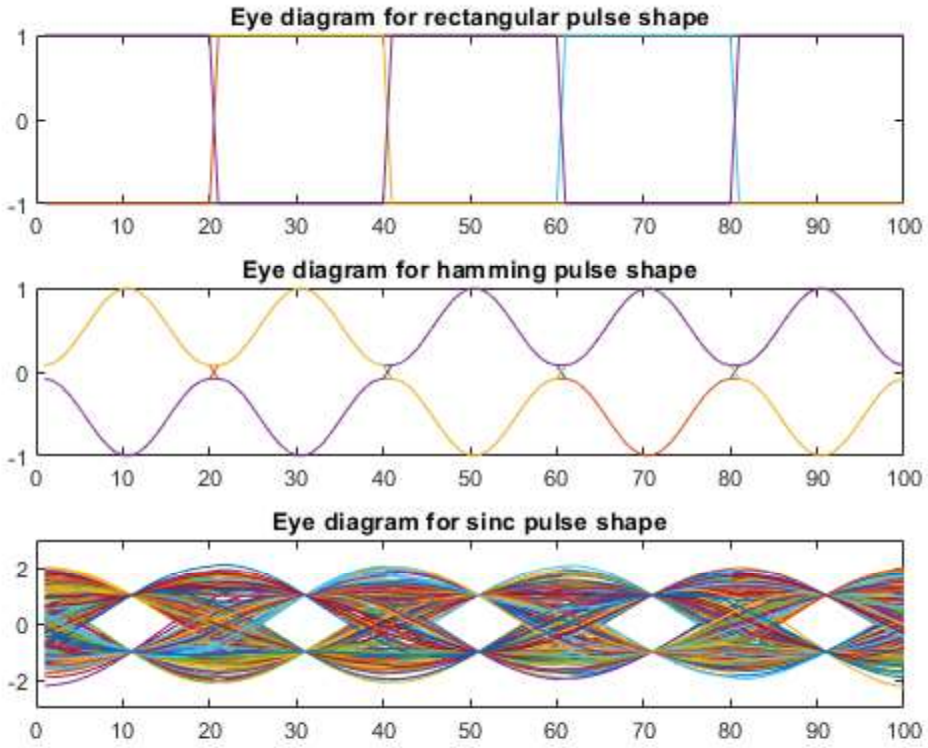
Problem 11.3 1
Problem 11.4 5
Problem 12.1 7
Problem 12.2 8
Problem 12.3 10
Problem 13.1 13
Problem 13.2a 19
Problem 13.2b 20
Problem 13.2c 24
Problem 13.2d 25

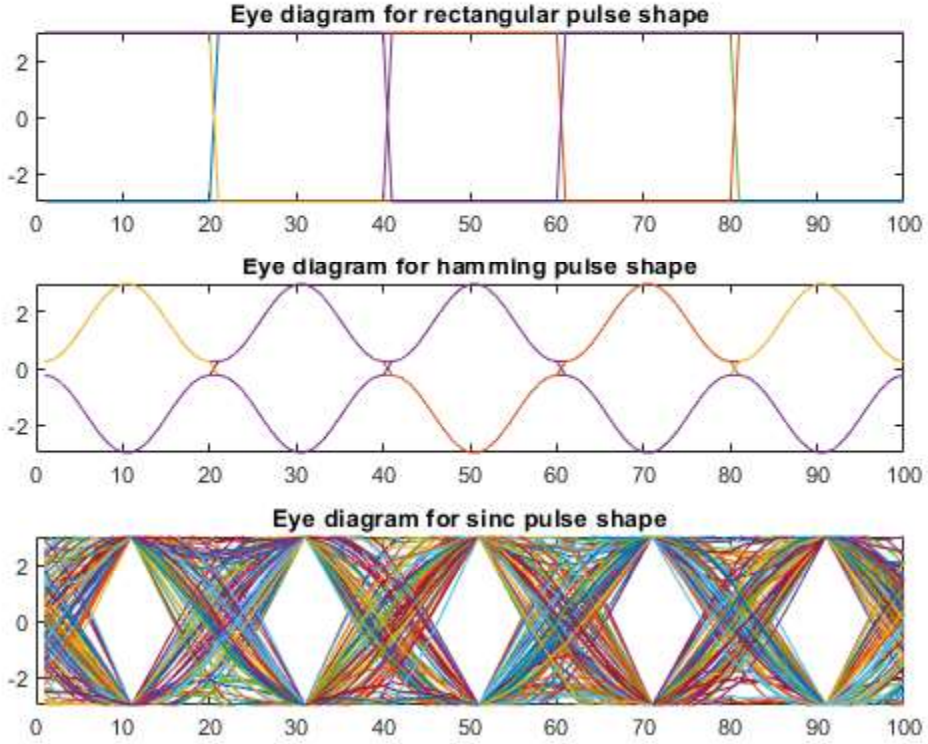
Problem 11.3

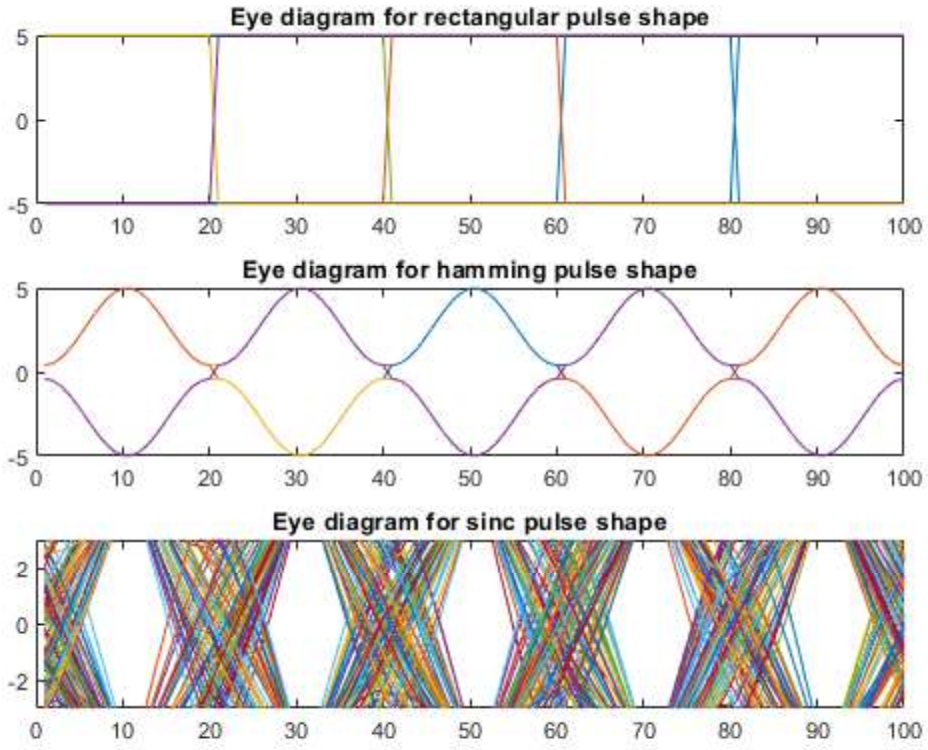
```
figure                          % used to plot figure eyediag3
N=1000; m=pam(N,2,1);           % random +/-1 signal of length N
M=20; mup=zeros(1,N*M); mup(1:M:N*M)=m;  % oversampling by factor of M
ps=ones(1,M);                   % square pulse width M
x=filter(ps,1,mup);             % convolve pulse shape with mup
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')

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+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')

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+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,-3,3])
title('Eye diagram for sinc pulse shape')


figure(2)                      % 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,1/3,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')

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,1/3,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')

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,1/3,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,-3,3])
title('Eye diagram for sinc pulse shape')

figure(3)                      % 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,1/5,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')

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,1/5,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')

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,1/5,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,-3,3])
title('Eye diagram for sinc pulse shape')
```


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

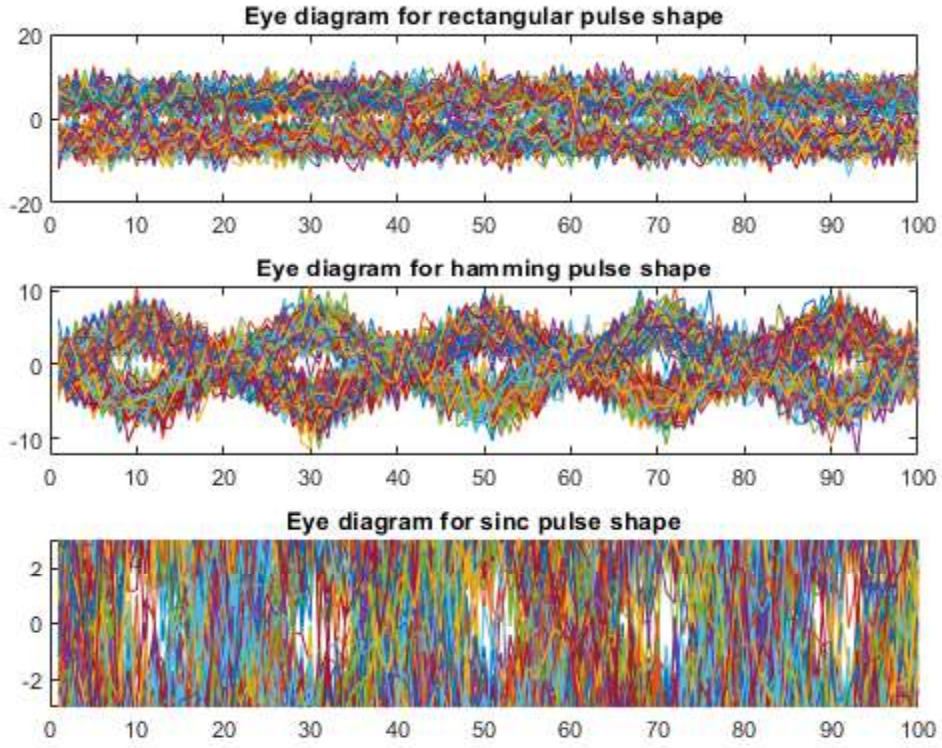

Eye diagram for rectangular pulse shape
Eye diagram for hamming pulse shape
Eye diagram for sinc pulse shape



Problem 11.4

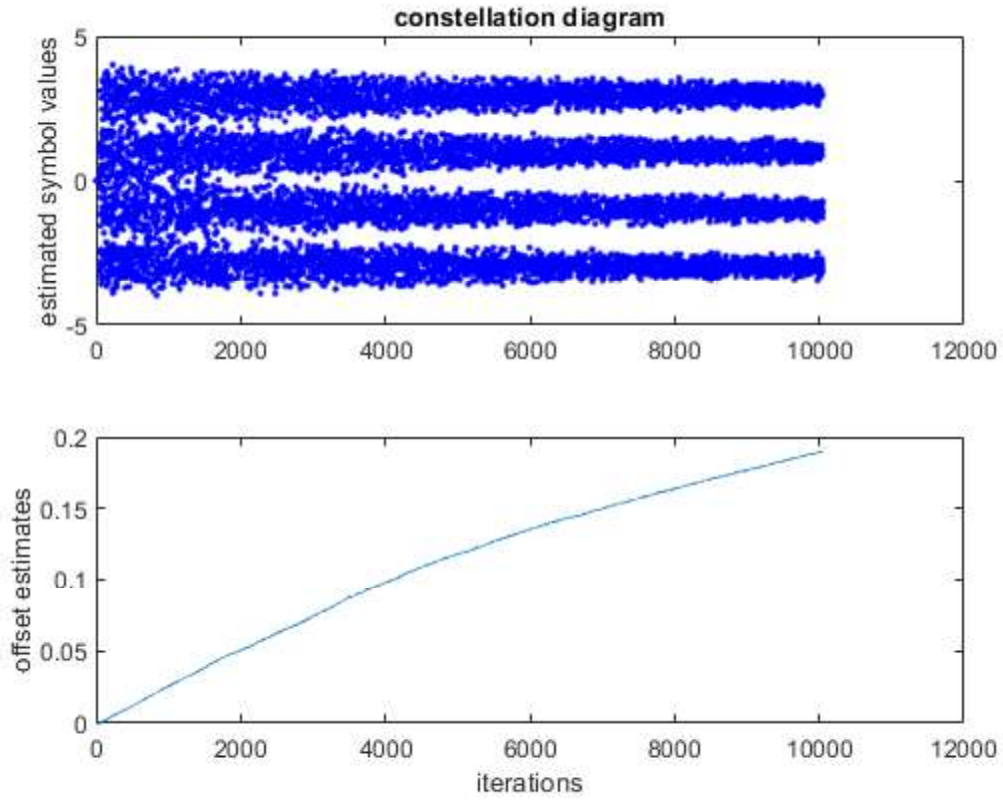
```
%v = 0.4
%For rectangular pulse, V=2.5 is the highest noise
figure(4)
N=1000; m=pam(N,2,1);           % random +/-1 signal of length N
M=20; mup=zeros(1,N*M); mup(1:M:N*M)=m;  % oversampling by factor of M
ps=ones(1,M);                            % square pulse width M
x=filter(ps,1/5,mup);             % convolve pulse shape with mup
x=x+2.5*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')

%For the hamming pulse, V=1.9 is the highest noise
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,1/5,mup);             % convolve pulse shape with mup
x=x+1.9*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')

%For the sinc pulse,V = 1.9 is the highest noise
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,1/5,mup);     % convolve pulse shape with mup
x=x+1.9*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,-3,3])
title('Eye diagram for sinc pulse shape')
%At V=0.4, the eye is still visible but is nearly closed
```



Problem 12.1

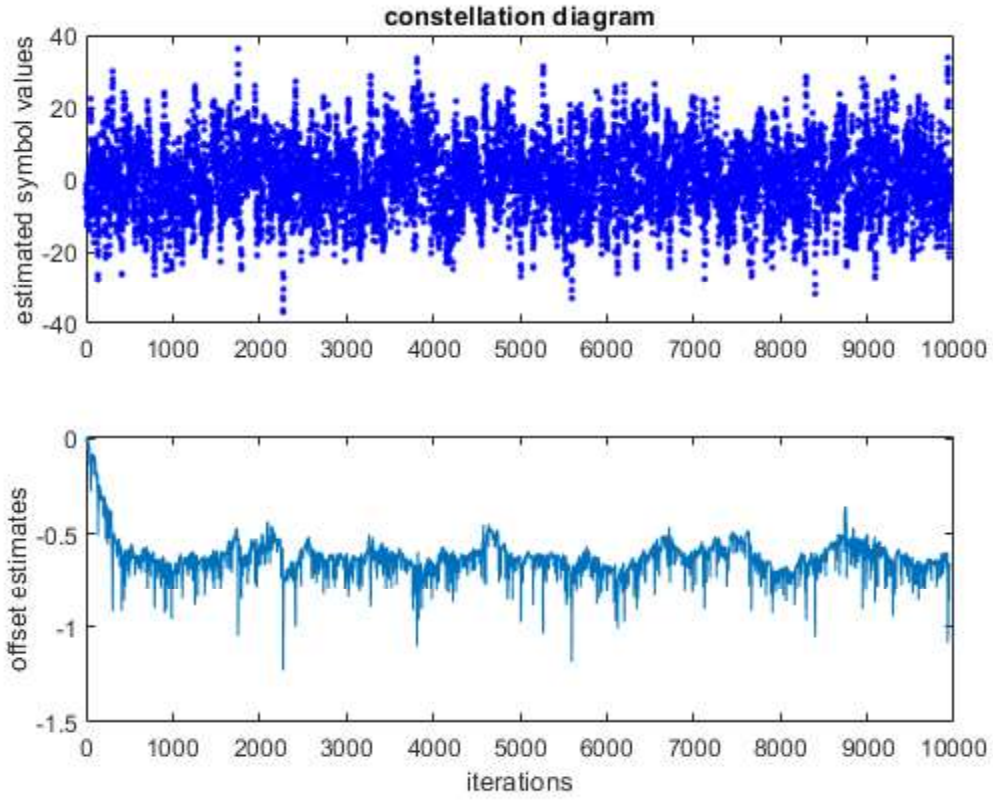
```
%At mu>=0.002, the constellation diagram loses its shape
%Tau converges to 0.3 around when the constellation diagram lines
 converge
%to a central line

figure(5)
% 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=0.0002;                            % 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')
```



Problem 12.2

```
%The constellation Diagram is significantly worse on the rectangular
 shaped
%pulse, so the SRRC would be better

figure(6)
% 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=ones(1,M);                   % rectangular 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
```

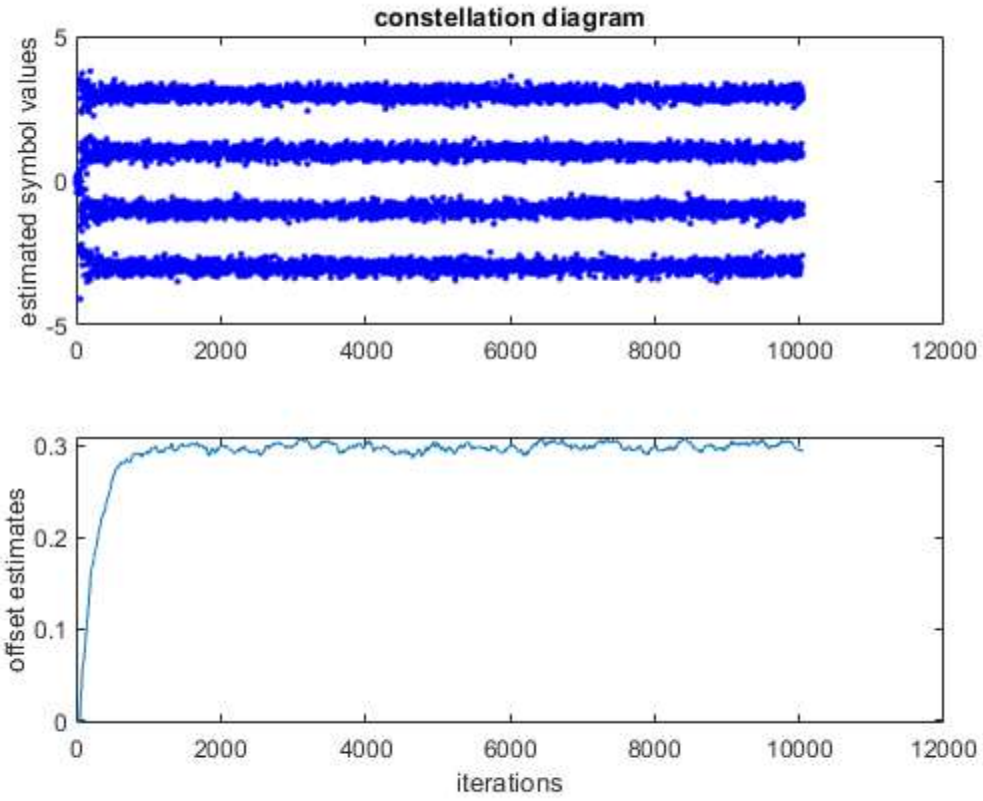
```
% clock recovery algorithm
tnow=l*m+1; tau=0; xs=zeros(1,n);    % initialize variables
tausave=zeros(1,n); tausave(1)=tau; i=0;
mu=0.01;                             % algorithm stepsize
delta=0.1;                           % time for derivative
while tnow<length(x)-2*l*m           % run iteration
  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')
```

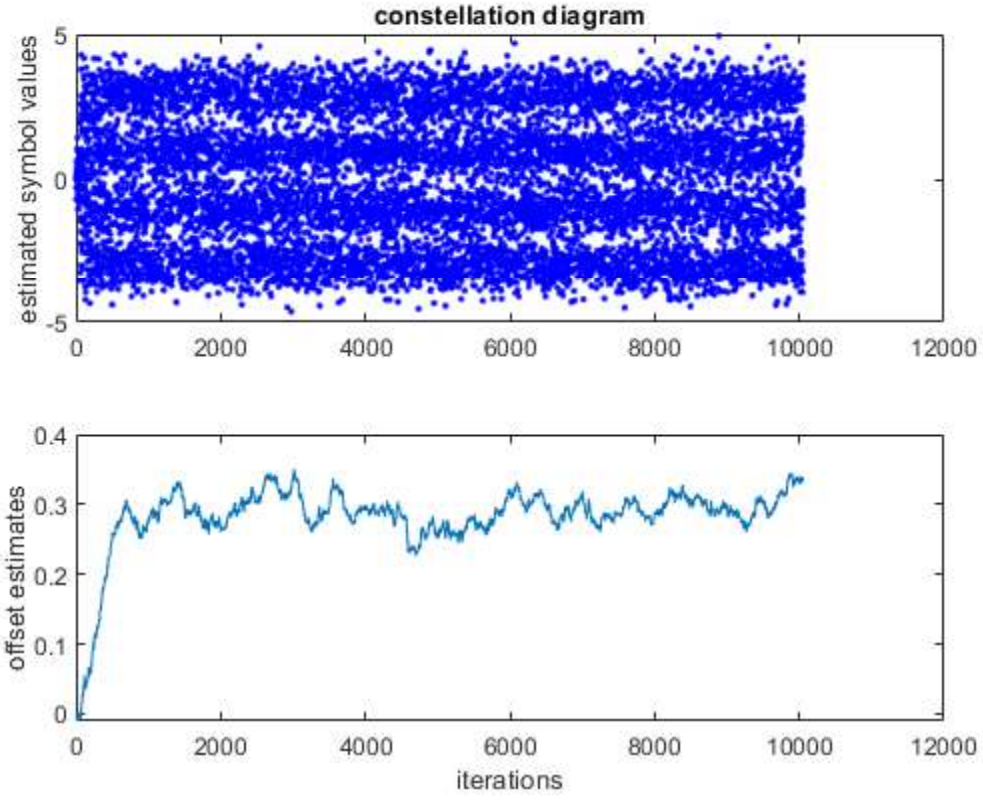


Problem 12.3

```
%How much the noise affects the convergence of tau depends of course
 on how
%much noise is added, but with enough noise it does affect the final
%converged value

figure(7)
% 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')

figure(8)
% 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+0.5*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')
```


constellation diagram
estimated symbol values
5
0
-5
0
2000
4000
6000
8000
10000
12000
offset estimates
0.3
0.2
0.1
0
iterations

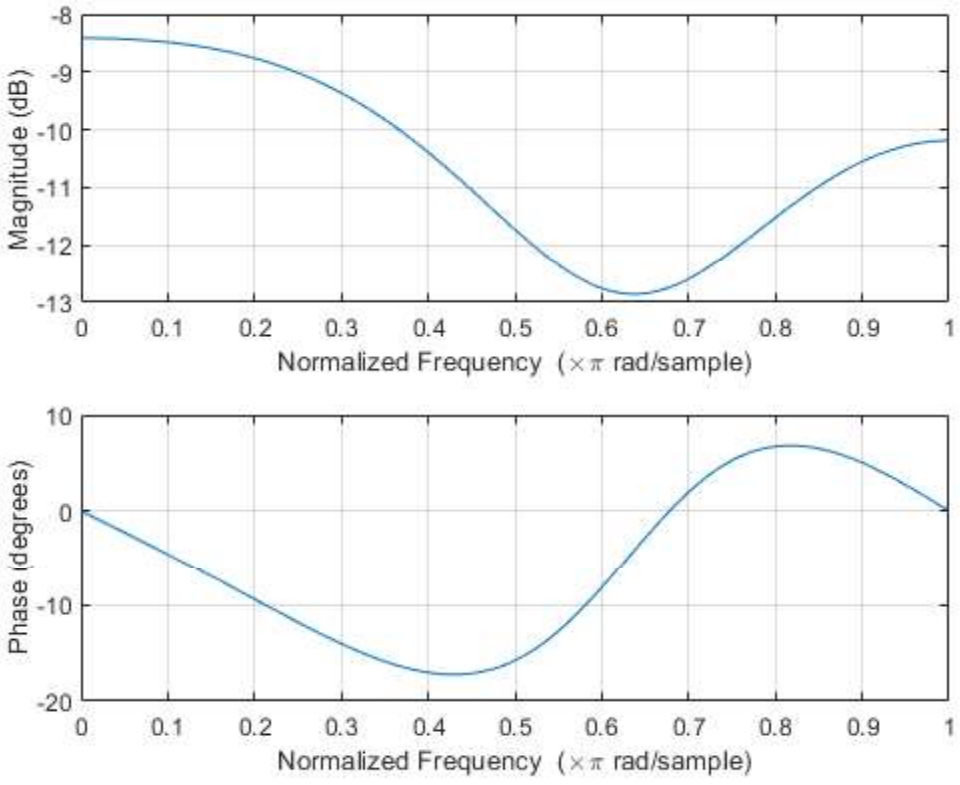

constellation diagram
estimated symbol values
5
0
-5
0
2000
4000
6000
8000
10000
12000
offset estimates
0.4
0.3
0.2
0.1
0
iterations

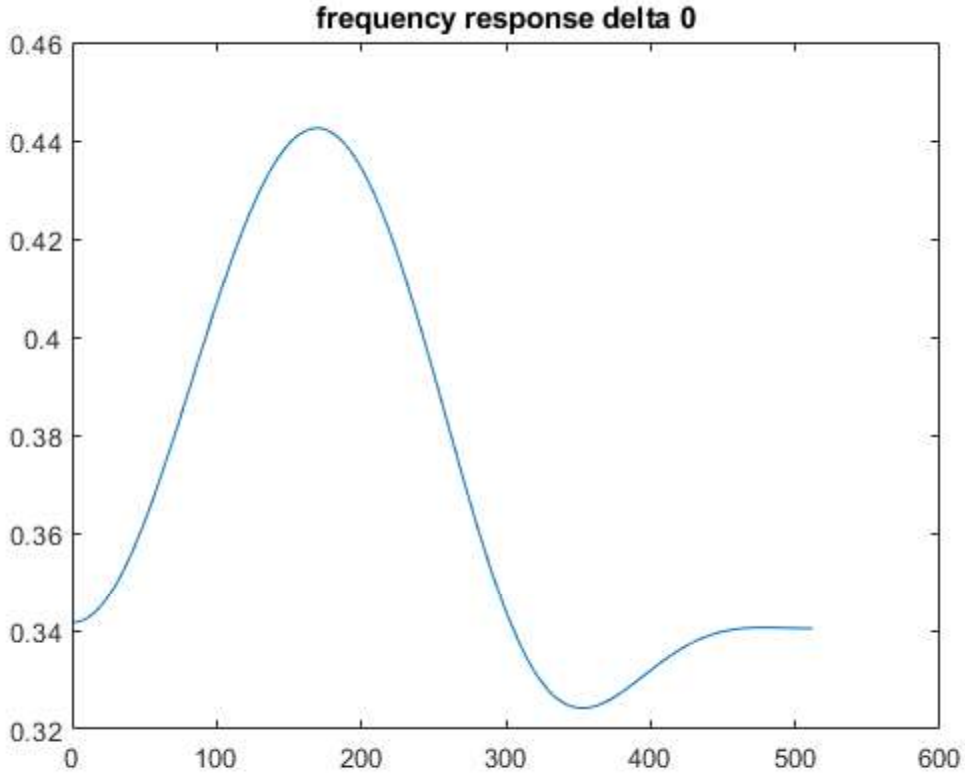
Problem 13.1

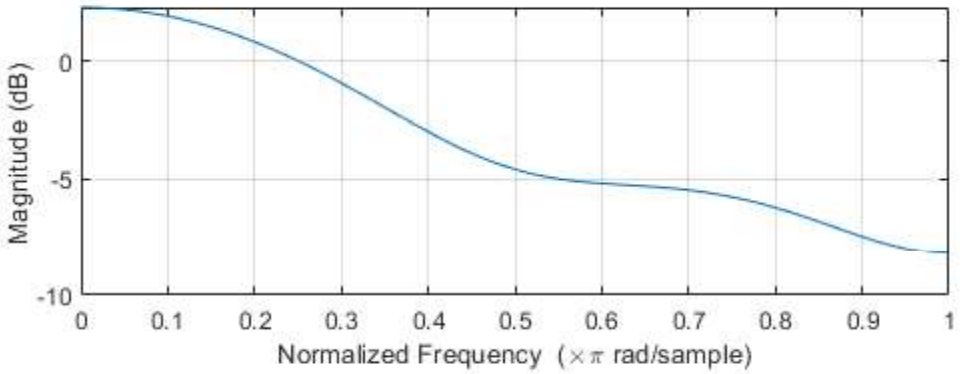
```
%Delta = 3 was the closest fproduct to unity

figure
% 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
r=filter(b,1,s);                    % output of channel
n=3;                                % length of equalizer - 1
delta=0;                            % 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
fproduct = f(1)*f(2)*f(3)*f(4)
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)))

[hf,af] = freqz(f);
freqz(f);
[hb,ab] = freqz(b);
product = hf.*hb;
figure
plot(1:size(product),abs(product));
title('frequency response delta 0');

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
fproduct = f(1)*f(2)*f(3)*f(4)
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)))

[hf,af] = freqz(f);
figure
freqz(f);
[hb,ab] = freqz(b);
product = hf.*hb;
figure
plot(1:size(product),abs(product));
title('frequency response delta 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
```

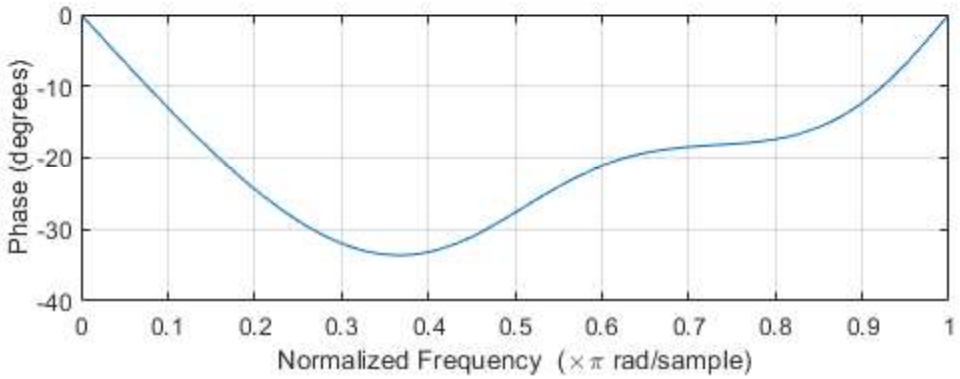
```
f=inv(R'*R)*R'*S;                    % calculate equalizer f
fproduct = f(1)*f(2)*f(3)*f(4)
Jmin=S'*S-S'*R*inv(R'*R)*R'*S;       % Jmin for this f and delta
y=filter(f,1,r);                     % equalizer is a filter
dec=sign(y);                         % quantize and find errors
err=0.5*sum(abs(dec(delta+1:m)-s(1:m-delta)))

[hf,af] = freqz(f);
figure
freqz(f);
[hb,ab] = freqz(b);
product = hf.*hb;
figure
plot(1:size(product),abs(product));
title('frequency response delta 2');

delta=3;                             % 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
fproduct = f(1)*f(2)*f(3)*f(4)
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)))

[hf,af] = freqz(f);
figure
freqz(f);
[hb,ab] = freqz(b);
product = hf.*hb;
figure
plot(1:size(product),abs(product));
title('frequency response delta 3');


fproduct =

  -1.3155e-05


err =

   511


fproduct =

    0.0036


err =
```

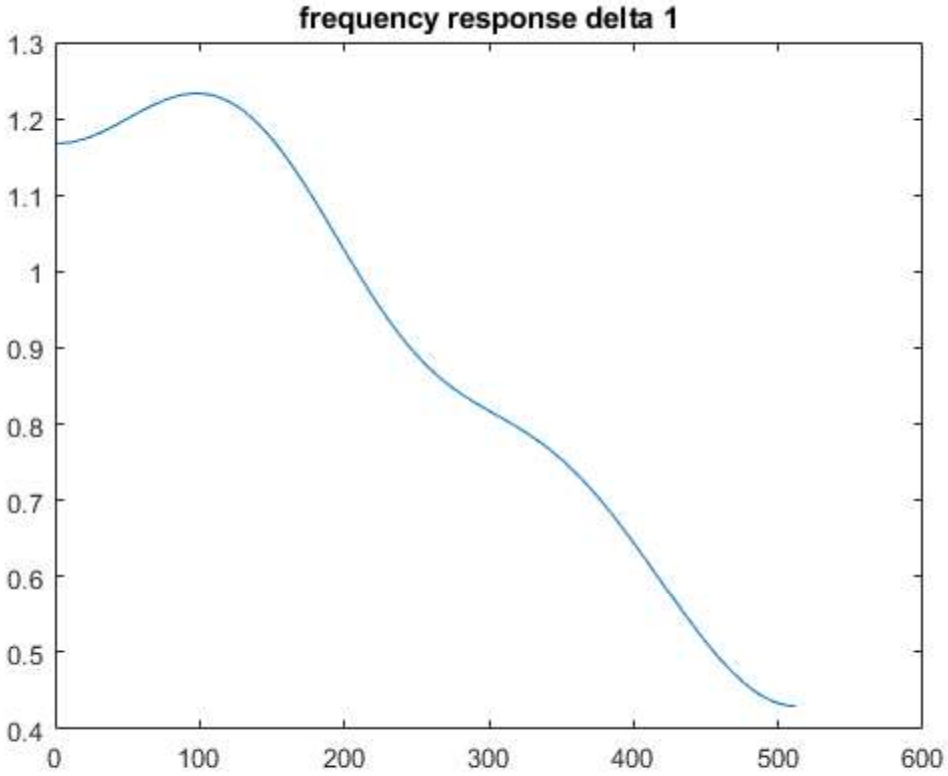
```
     0

fproduct =

   -0.0076

err =

     0

fproduct =

   -0.0055

err =

     0
```

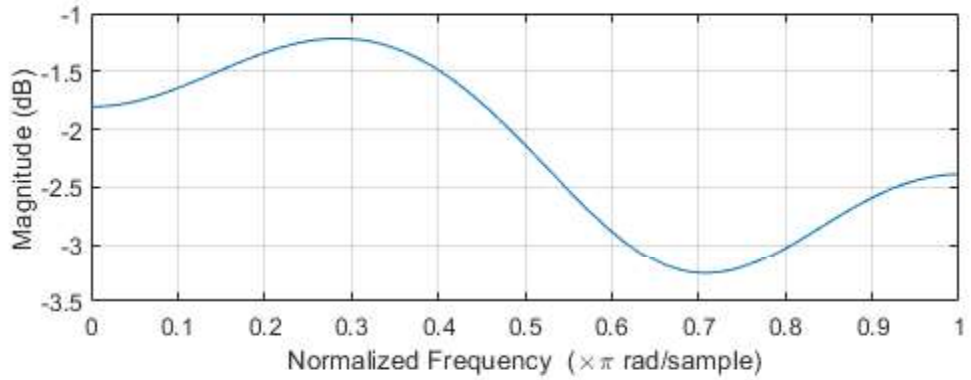


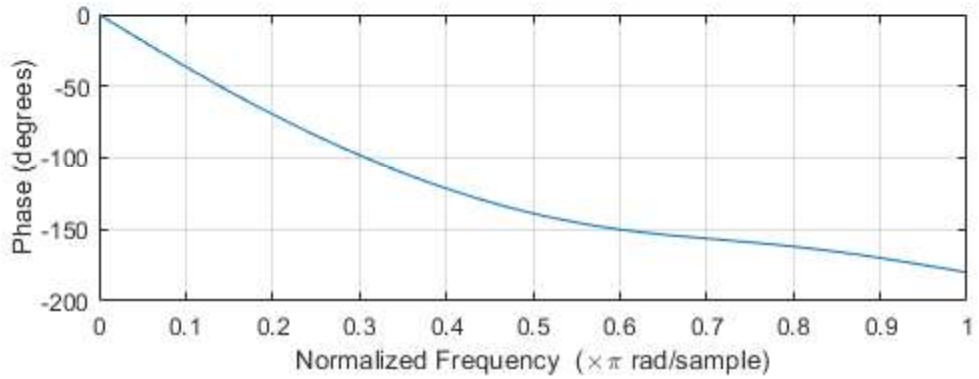

frequency response delta 0
0.46
0.44
0.42
0.4
0.38
0.36
0.34
0.32
0
100
200
300
400
500
600

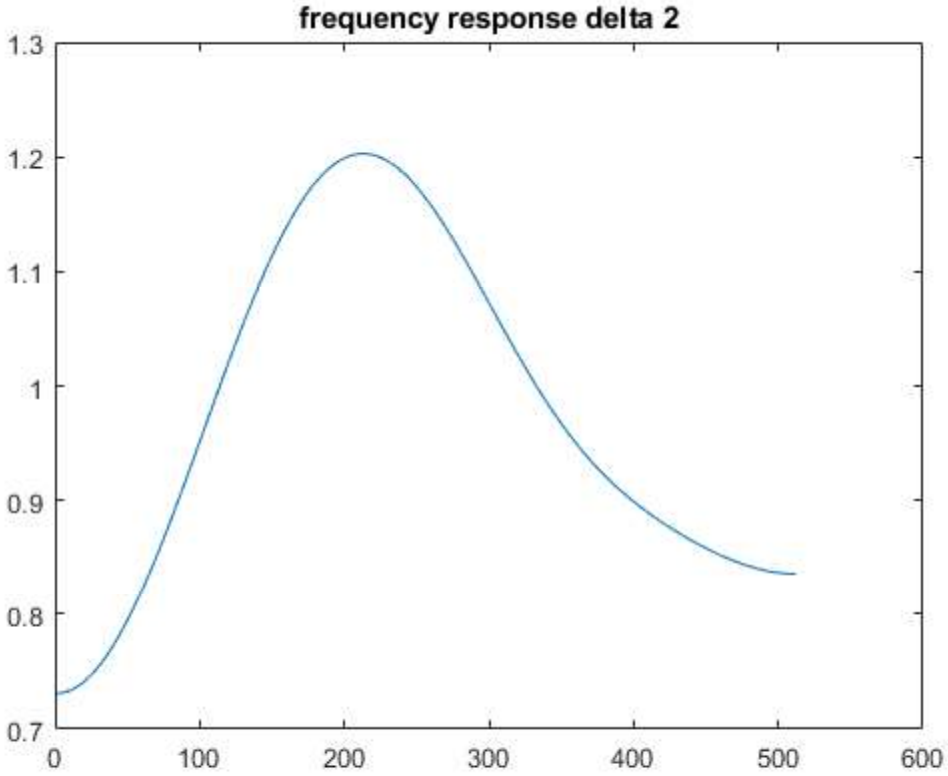

Magnitude (dB)
0
-5
-10
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)

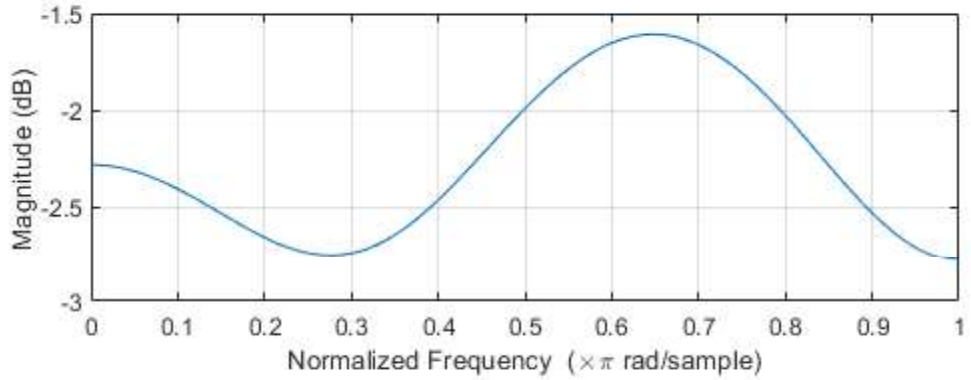

Phase (degrees)
0
-10
-20
-30
-40
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)

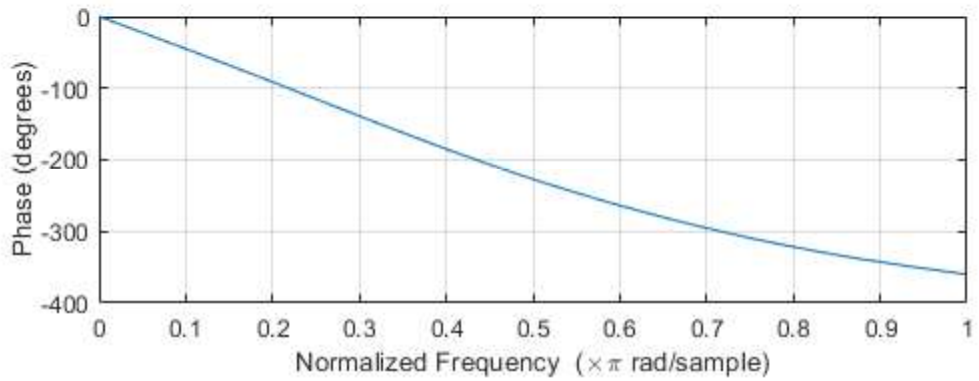

frequency response delta 1
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0
100
200
300
400
500
600

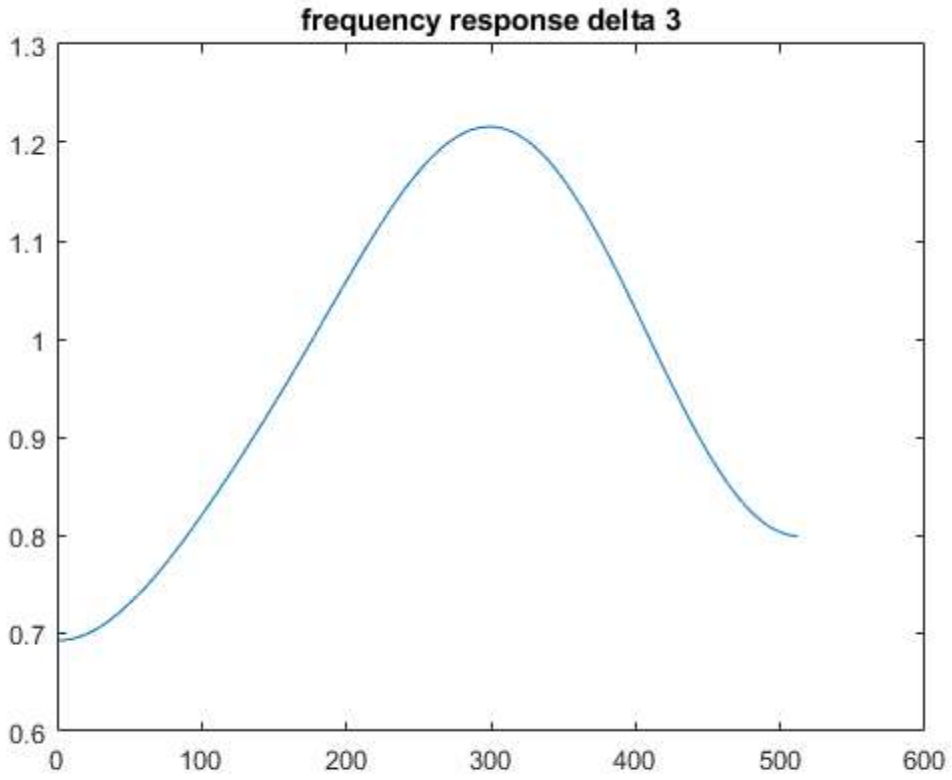

Magnitude (dB)
-1
-1.5
-2
-2.5
-3
-3.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)


Phase (degrees)
0
-50
-100
-150
-200
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)


frequency response delta 2
1.3
1.2
1.1
1
0.9
0.8
0.7
0
100
200
300
400
500
600


Magnitude (dB)
-1.5
-2
-2.5
-3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)

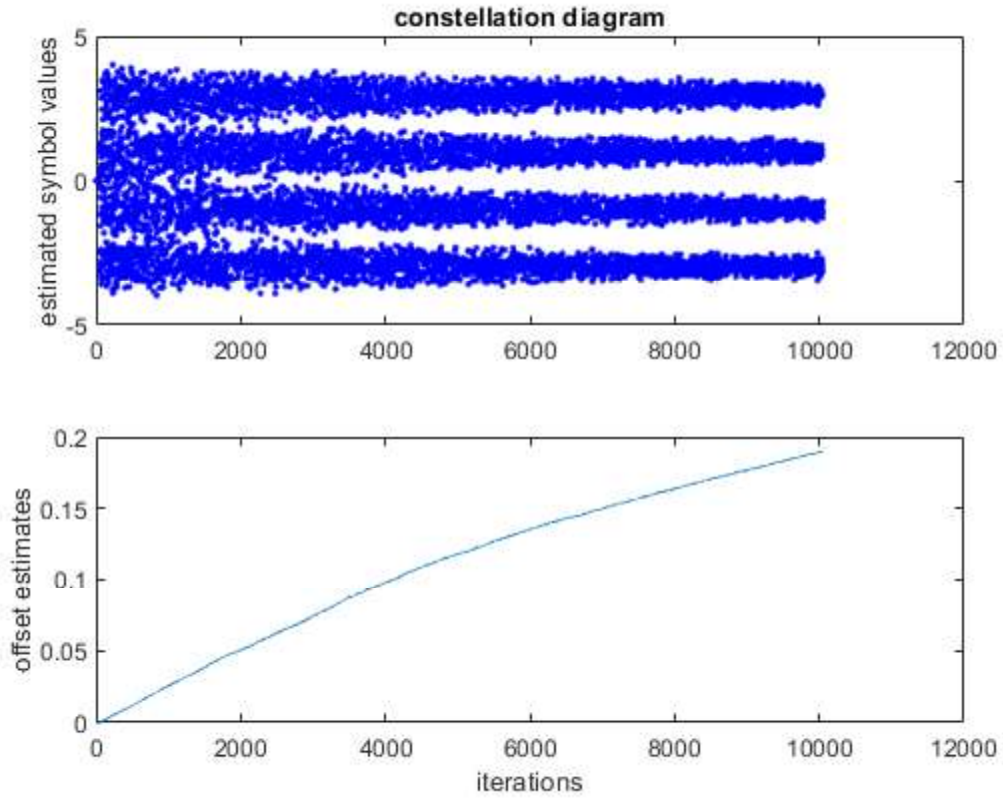

Phase (degrees)
0
-100
-200
-300
-400
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)



Problem 13.2a

```
%You start getting errors around when sd = 0.3
figure(10)
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);                      % output of channel
r=filter(b,1,s)+0.3*randn(size(s));
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)))


f =

   -0.2659
    0.6196
    0.2907
    0.1298
```

```
Jmin =

    77.9760


err =

     0
```



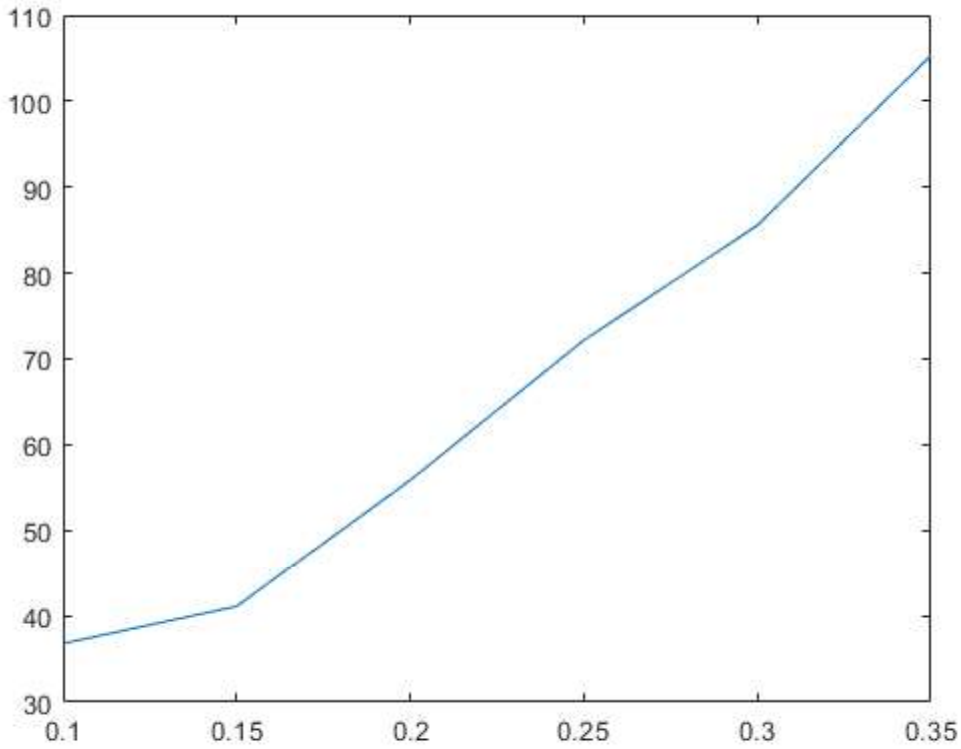
Problem 13.2b

```
Jmintrack = []
% 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
r=filter(b,1,s);                    % output of channel
sd = [0.1 0.15 0.2 0.25 0.3 0.35];
for i=1:6
r=filter(b,1,s)+sd(i)*randn(size(s));
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)))
Jmintrack = [Jmintrack,Jmin]
end

plot(0.1:0.05:0.35,Jmintrack)


Jmintrack =

     []


f =

   -0.2666
    0.6392
    0.2988
    0.1399


Jmin =

   36.7807


err =

     0


Jmintrack =

   36.7807


f =

   -0.2667
    0.6325
    0.2975
    0.1331


Jmin =

   41.0815


err =
```

```
     0


Jmintrack =

   36.7807   41.0815


f =

   -0.2656
    0.6244
    0.3020
    0.1331


Jmin =

   55.8673


err =

     0


Jmintrack =

   36.7807   41.0815   55.8673


f =

   -0.2618
    0.6199
    0.2911
    0.1327


Jmin =

   72.1830


err =

     0


Jmintrack =

   36.7807   41.0815   55.8673   72.1830
```

```
f =

   -0.2594
    0.6084
    0.2900
    0.1188


Jmin =

   85.5669


err =

     0


Jmintrack =

   36.7807   41.0815   55.8673   72.1830   85.5669


f =

   -0.2639
    0.5908
    0.2910
    0.1147


Jmin =

  105.3057


err =

     1


Jmintrack =

   36.7807   41.0815   55.8673   72.1830   85.5669  105.3057
```



Problem 13.2c

```
%This part starts showing errors after sd > 0.2
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);                     % output of channel
r=filter(b,1,s)+0.20*randn(size(s));
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)))


f =

    0.6632
    0.3613
    0.1402
    0.0670
```

```
Jmin =

  162.5432


err =

     0
```

Problem 13.2d

```
%The equalizer with the delay 2 is the better equalizer, because it
 allows
%larger amounts of noise before getting errors.
```

Published with MATLAB® R2019b