Table of Contents

Exercise 11.3

```
figure(1)
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 for +/- 1')

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 for +/- 1')

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 for +/- 1')

figure(2)
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,0.333333333,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 for +/- 3')

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,0.333333333,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 for +/- 3')

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,0.333333,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 for +/- 3')

figure(3)
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,0.2,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 for +/- 5')

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,0.2,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 for +/- 5')

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,0.2,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,-6,6])
title('Eye diagram for sinc pulse shape for +/- 5')


c =

   200


c =

   200


c =

   200


c =
```

200

c =

200

c =

200

c =

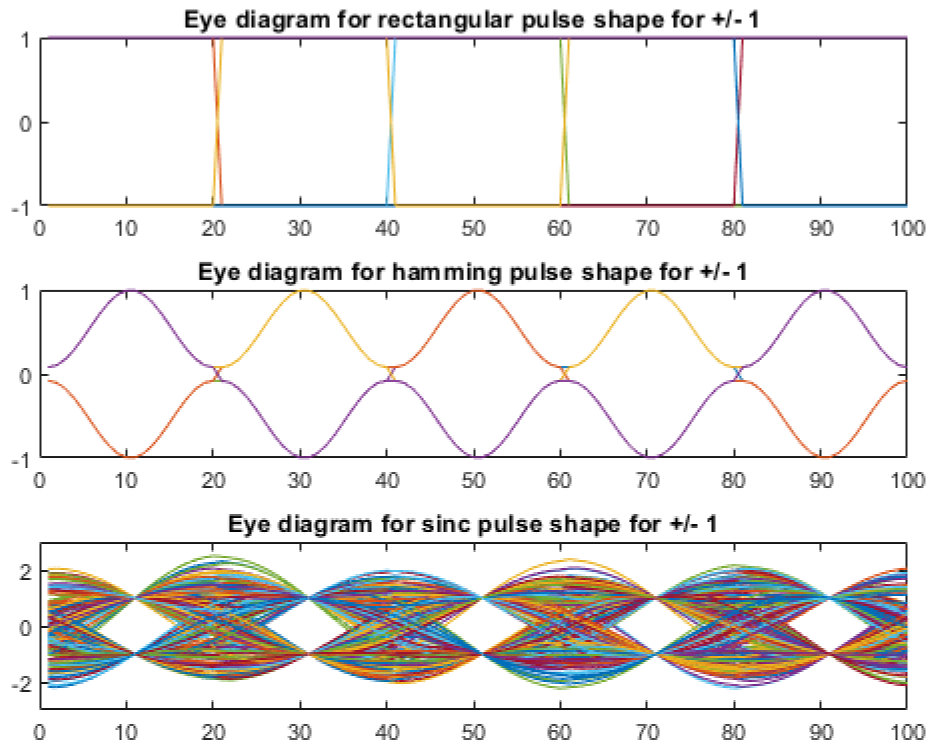
200

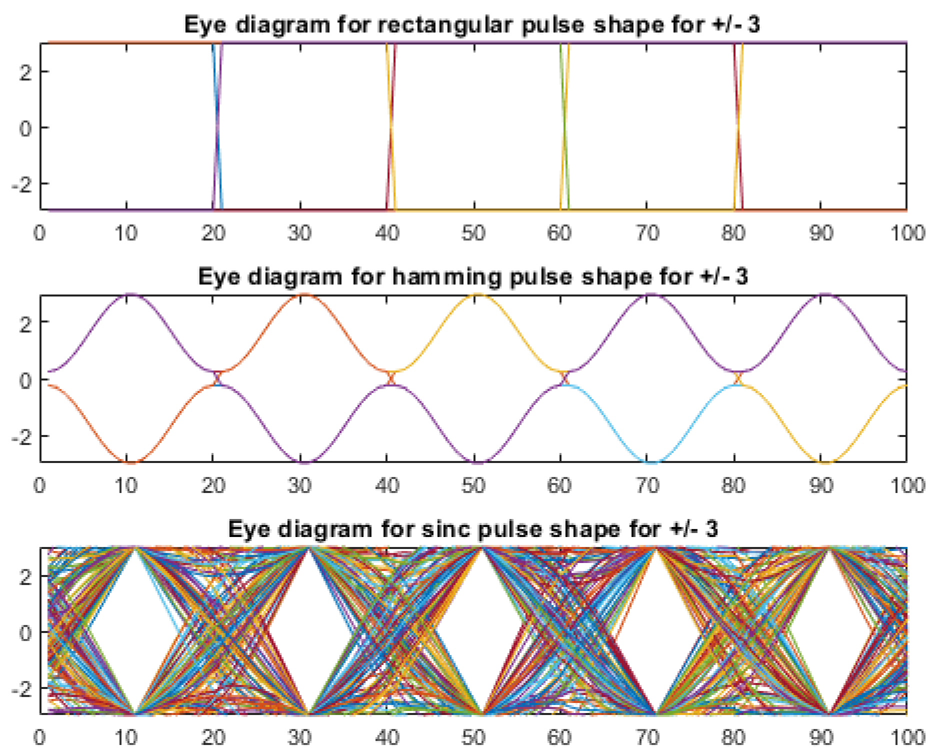
c =

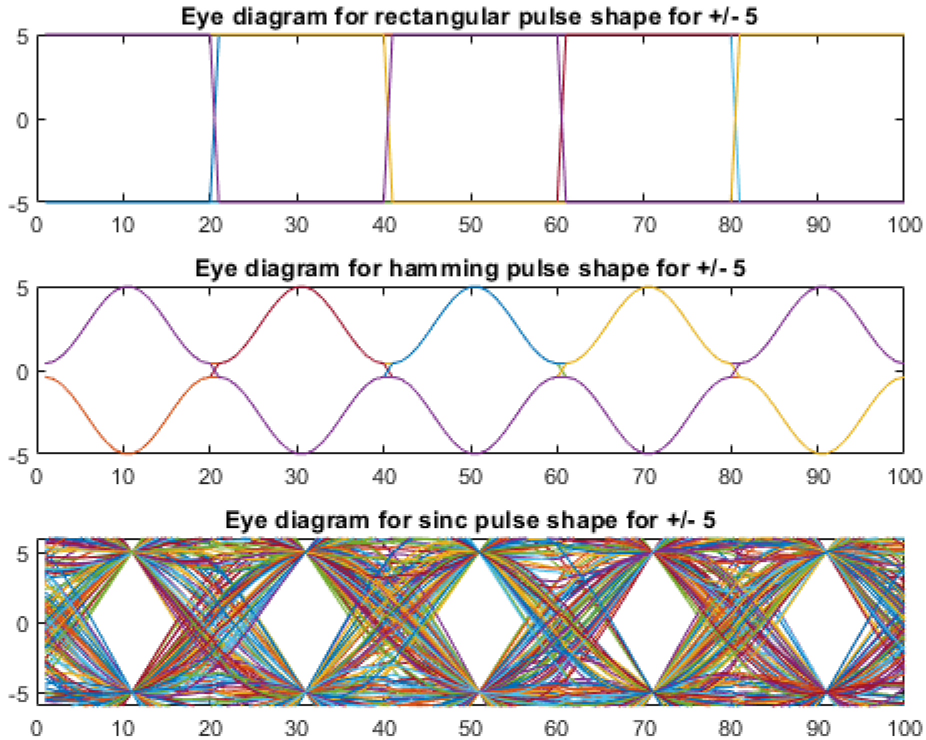
200

c =

200


Eye diagram for rectangular pulse shape for +/- 1
Eye diagram for hamming pulse shape for +/- 1
Eye diagram for sinc pulse shape for +/- 1
0 10 20 30 40 50 60 70 80 90 100


Eye diagram for rectangular pulse shape for +/- 3
Eye diagram for hamming pulse shape for +/- 3
Eye diagram for sinc pulse shape for +/- 3
0 10 20 30 40 50 60 70 80 90 100

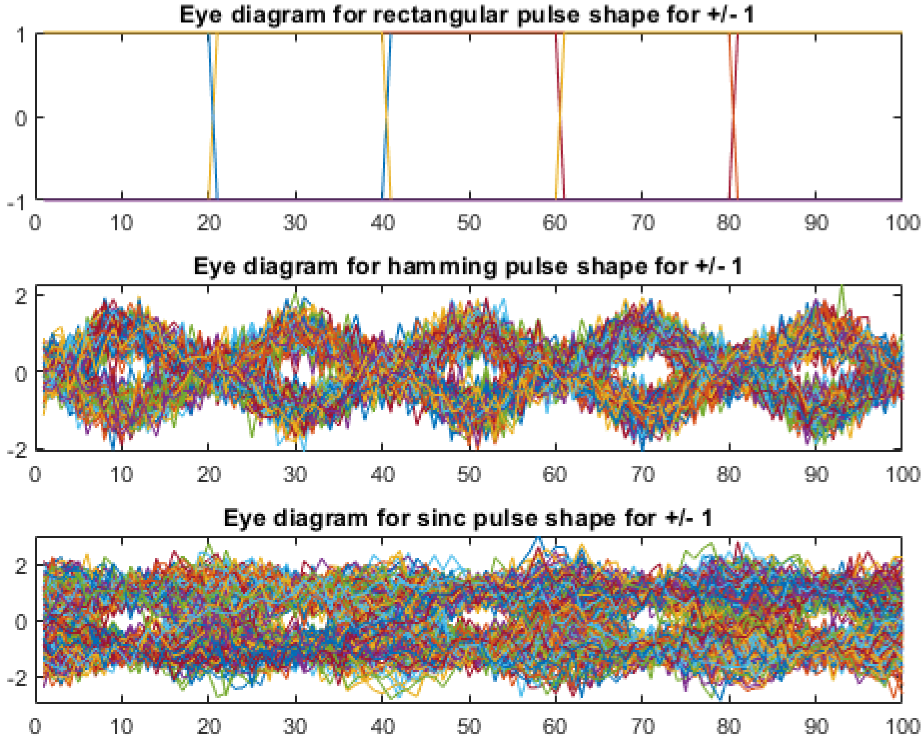


Exercise 11.4

```
v=0.35;
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 for +/- 1')

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+v*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 +/- 1')

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+v*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 for +/- 1')
%For the value of v to fulfill the expression v*rand, the highest
 value
%that still allows for the eyes to still be considered open would be
 v=0.4 for the hamming pulse,
%and even then the eyes aren't open enough that allow for the clearest
%opening. For the sinc pulse shape, the highest v value that wwould
 allow
%the eyes to be open would be v=0.35.
```

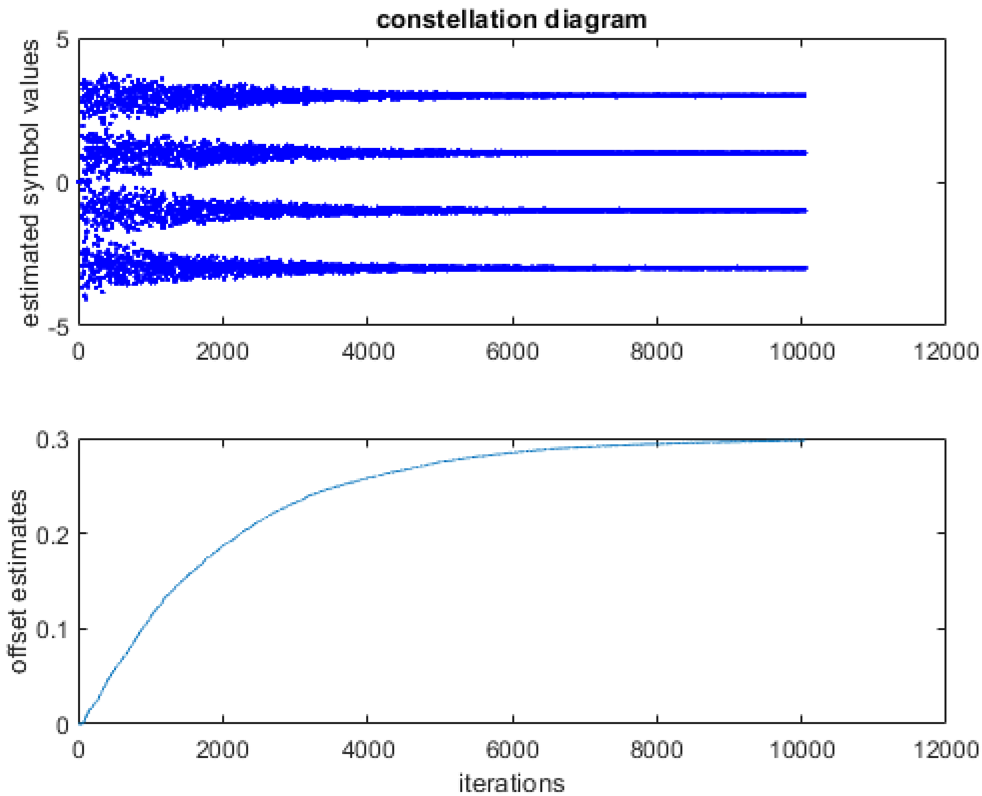
```
c =

   200


c =

   200


c =

   200
```



Exercise 12.1a

```
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
tnow=l*m+1; tau=0; xs=zeros(1,n);   % initialize variables
tausave=zeros(1,n); tausave(1)=tau; i=0;
mu=0.001;                            % 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
```

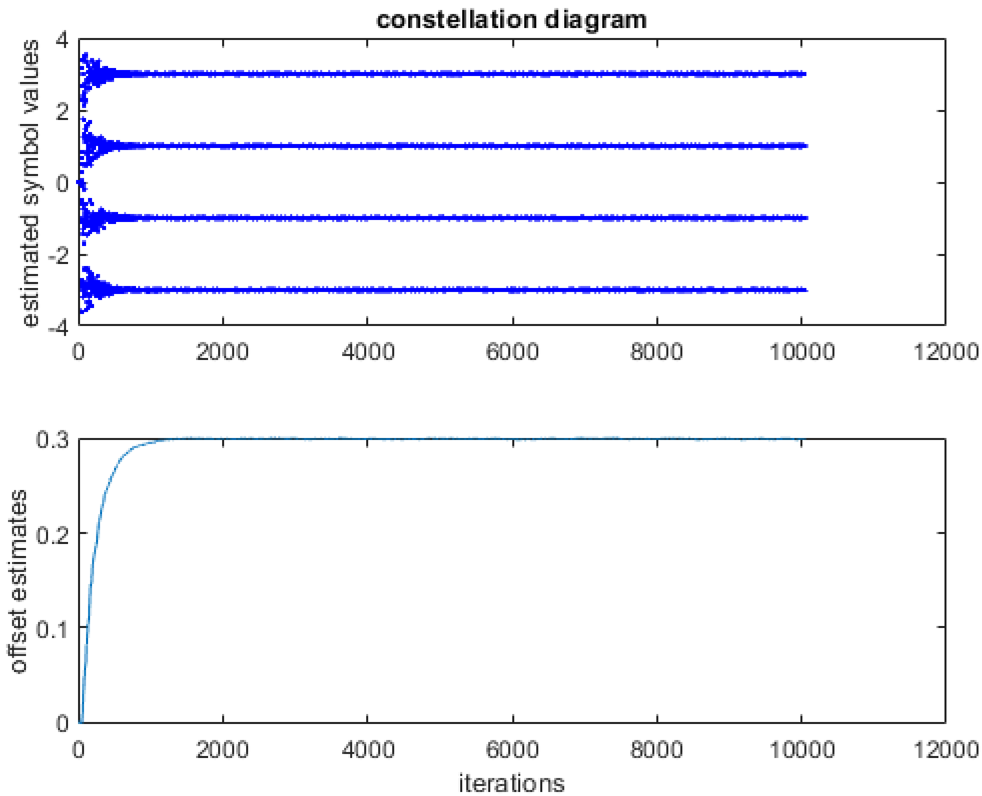
```
  x_deltap=interpsinc(x,tnow+tau+delta,l); % value to right
  x_deltam=interpsinc(x,tnow+tau-delta,l); % value to left
  dx=x_deltap-x_deltam;                  % numerical derivative
  qx=quantalph(xs(i),[-3,-1,1,3]);       % quantize to alphabet
  tau=tau+mu*dx*(qx-xs(i));              % alg update: DD
  tnow=tnow+m; tausave(i)=tau;           % save for plotting
end
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')
%Increasing the value of mu decreases how soon the algorithm
 converges, with a
%range of 0.001 to around 0.51 allowing the algorithm to function
 properly, with proper
%convergence plots.
```



Exercise 12.1a

```
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
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
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')
```

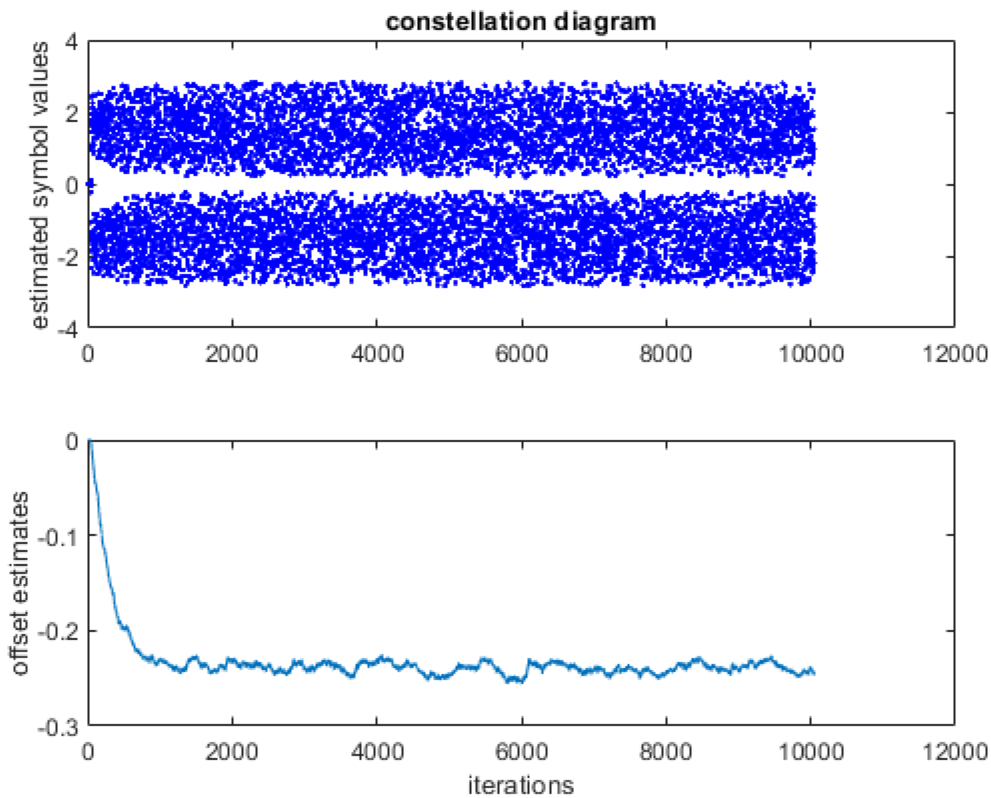


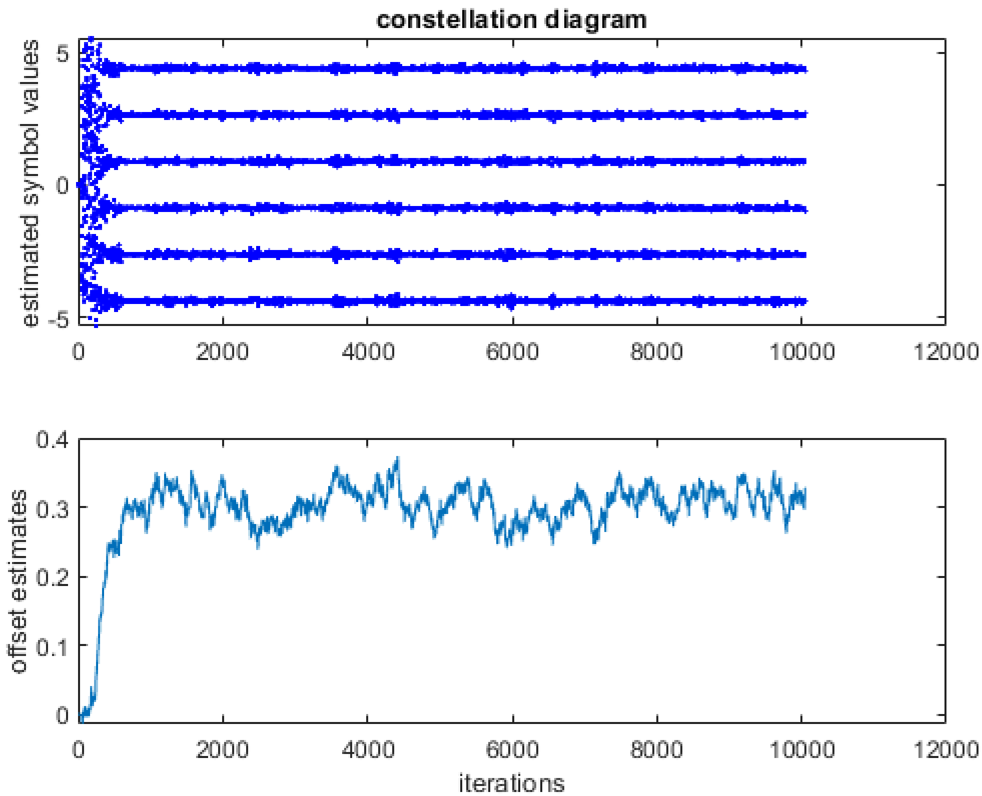
Exercise 12.1b

```
figure(1)
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,2,3);                     % 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
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
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(2)
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,6,9);                     % 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
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
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')
```

```
%For a 2-PAM system with a variance of 3, the constellation diagram is
 more disorganized and
%does not properly converge into a thin horizontal line, but stays at
 a
%constant width across all iterations. The offset estimate also
 decreases
%until a rough constant of around -2.5. For the 6-PAM system with a
%variance of 9, the constellation diagram become more organized into 6
%horizontal lines, tapering off near the end, and the offset estimate
 does
%not stay a constant number, but the average can be assumed to be
 around
%0.3.
```





Exercise 12.2

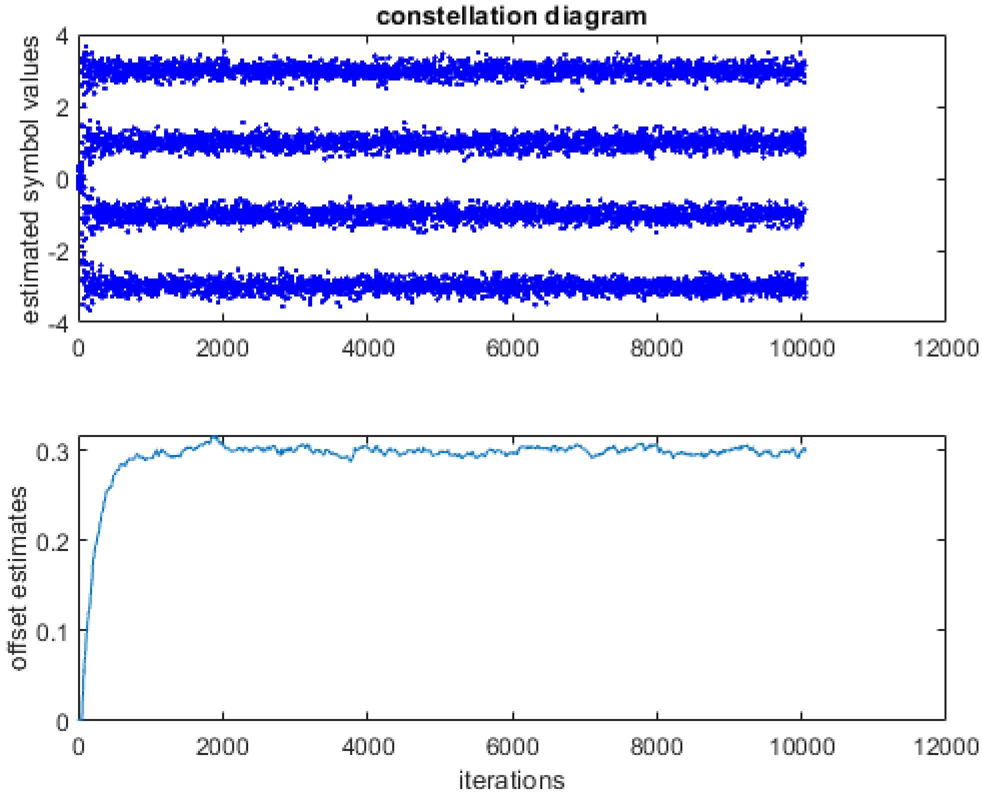
```
n=10000;                          % number of data points
m=2;                              % oversampling factor
beta=0.3;                         % rolloff parameter for srrc
l=50;                             % 1/2 length of pulse shape (in
 symbols)
chan=[1];                         % T/m "channel"
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
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
%Implementing a rectangular pulse shape does not seem to provide a
%reasonable constellation diagram, and the offset estimate does not
%properly diverge into a value of 0.3, so the better choice for this
 type
%of pulse shaping would be srrc.
```

Exercise 12.3

```
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));
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
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')
%Adding noise to the system causes the program to still converge to an
```

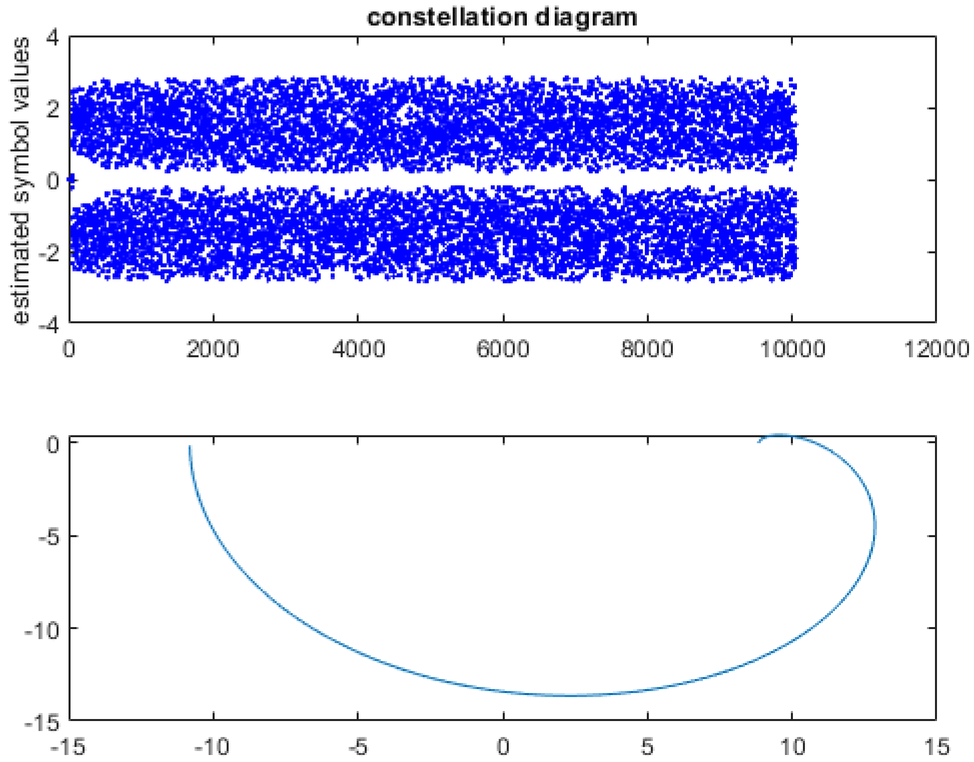
```
%offset estimate of 0.3, but adds variance so that it is not a
 straight line
%as in the earlier examples. With an increase of noise comes an
 increase in
%the amount of shakiness that comes with the offset estimate plot.
```

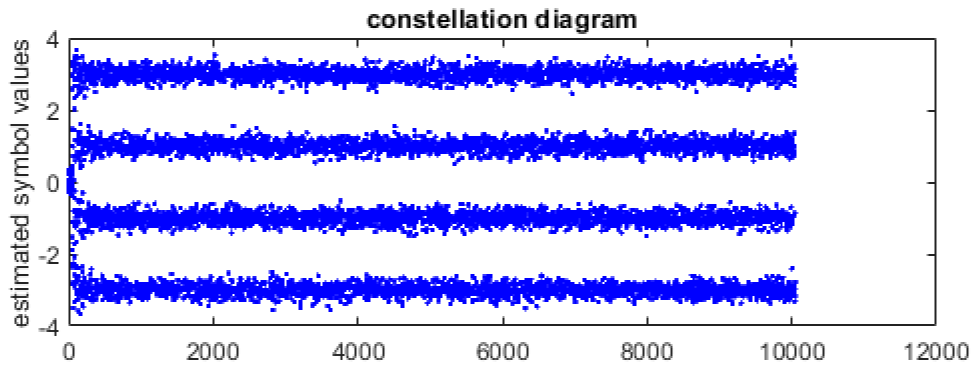


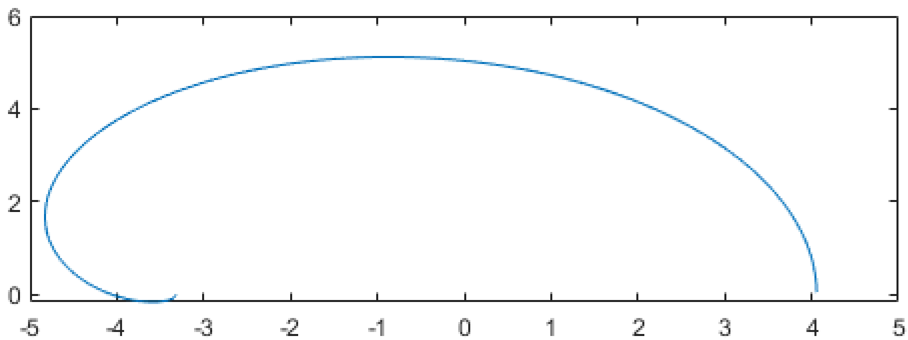
Exercise 13.1

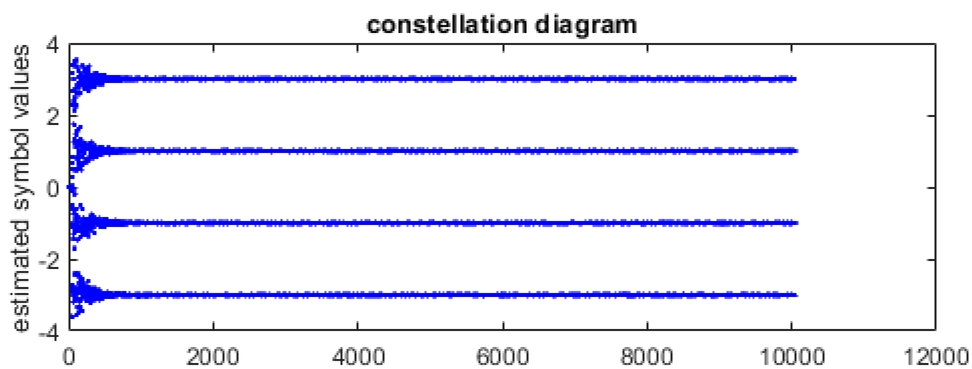
```
b=[0.5 1 -0.6];                    % define channel
m=1000; s=sign(randn(1,m));        % binary source of length m
r=filter(b,1,s);                   % output of channel
n=3;                               % length of equalizer - 1
delta=3;                           % use delay <=n*length(b)
p=length(r)-delta;
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
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(1)
plot(freqz(b,f(1)))
figure(2)
plot(freqz(b,f(2)))
figure(3)
```

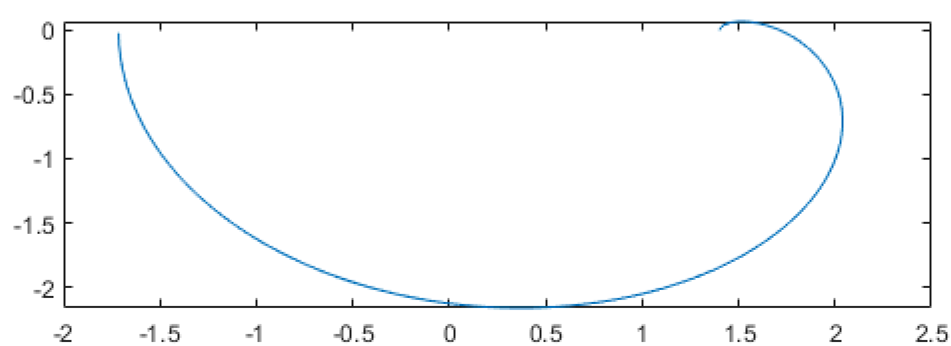
```
plot(freqz(b,f(3)))
figure(4)
plot(freqz(b,f(4)))


f =

    0.1018
   -0.2711
    0.6417
    0.2949


Jmin =

   44.9446


err =

     0
```

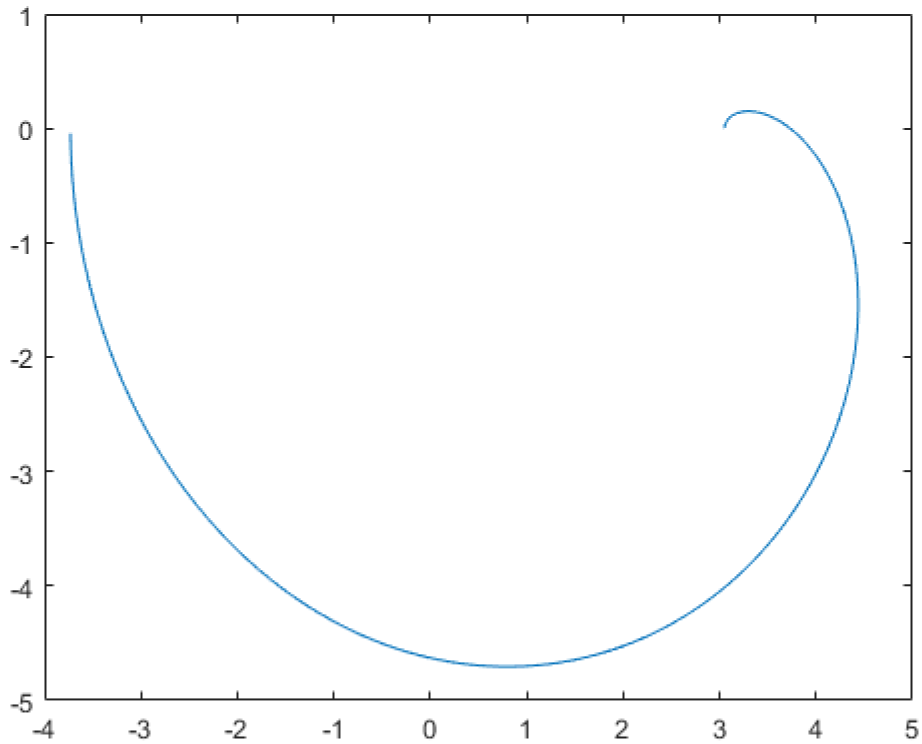



constellation diagram
estimated symbol values
4
2
0
-2
-4
0
2000
4000
6000
8000
10000
12000


6
4
2
0
-5
-4
-3
-2
-1
0
1
2
3
4
5


constellation diagram
estimated symbol values
4
2
0
-2
-4
0
2000
4000
6000
8000
10000
12000


0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5



Exercise 13.2a

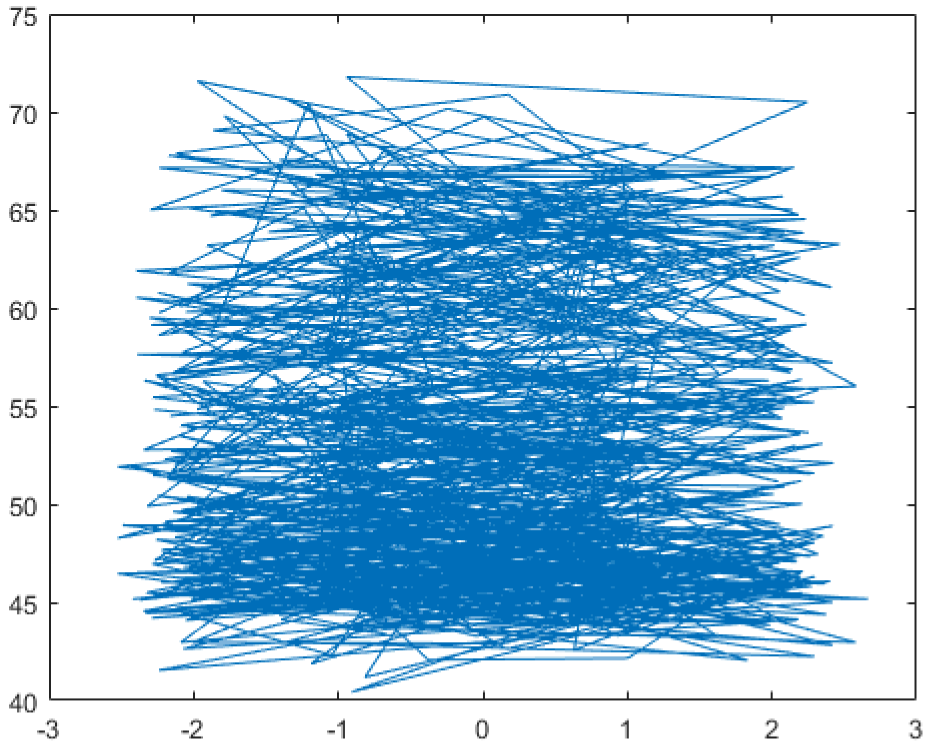
```
sd = 0.2;
b=[0.5 1 -0.6];                    % define channel
m=1000; s=sign(randn(1,m));        % binary source of length m
r=filter(b,1,s);                   % output of channel
r=filter(b,1,s)+sd*randn(size(s));
n=3;                               % length of equalizer - 1
delta=3;                           % 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
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)))
%The highest sd value that would produce an error value of zero across
%multiple tests (it was discovered that the Jmin and error values
 would
%vary if the same code was run multiple times with no changes to
 variables
%or parameters) was 0.2.
```

f =

```
    0.0957
   -0.2666
    0.6285
    0.2979


Jmin =

   70.0831


err =

     0
```

Exercise 13.2b

```
index =1;
for sd = 0.0002:0.0002:0.2;
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)+sd.*randn(size(s));
n=3;                               % length of equalizer - 1
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
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)));
Jmin_plot(index)=Jmin;
index=index+1;
end
plot(r, Jmin_plot)
```



Exercise 13.2c

```
sd=0.127;
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)+sd*randn(size(s));
n=1;                                 % 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)))
%For an sd value of 0.125, a majority of the tests run resulted in an
 err
%value of 0 (see previous exercise for explanation, meaning that 0.125
 is
%the highest sd value that can be used%for the noise variance without
 an
%error.
```

```
f =

    0.6283
    0.3444


Jmin =

  196.0498


err =

     0
```

Exercise 13.2d

```
%The better equalizer is the one shown in Exercise 13.2a, because that
 can
%result in a higher noise randomness value without sacrificing error,
%meaning that that equalizer can account for more noise while still
 being
%functional, so it is better when compared to the equalizer shown in
 13.2c.
```

Published with MATLAB® R2020b