Table of Contents

Exercise 9.1

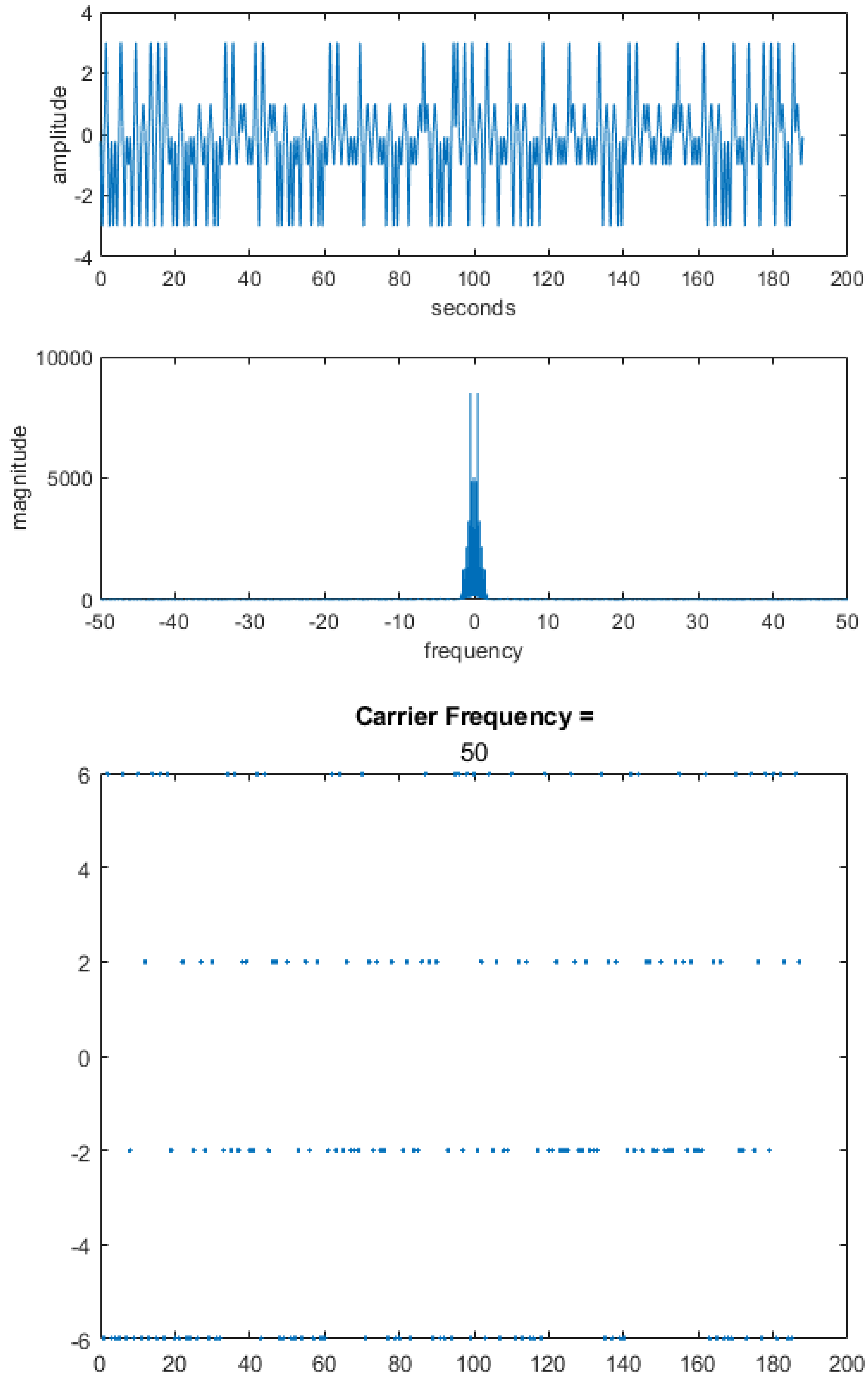
encode text string as T-spaced 4-PAM sequence

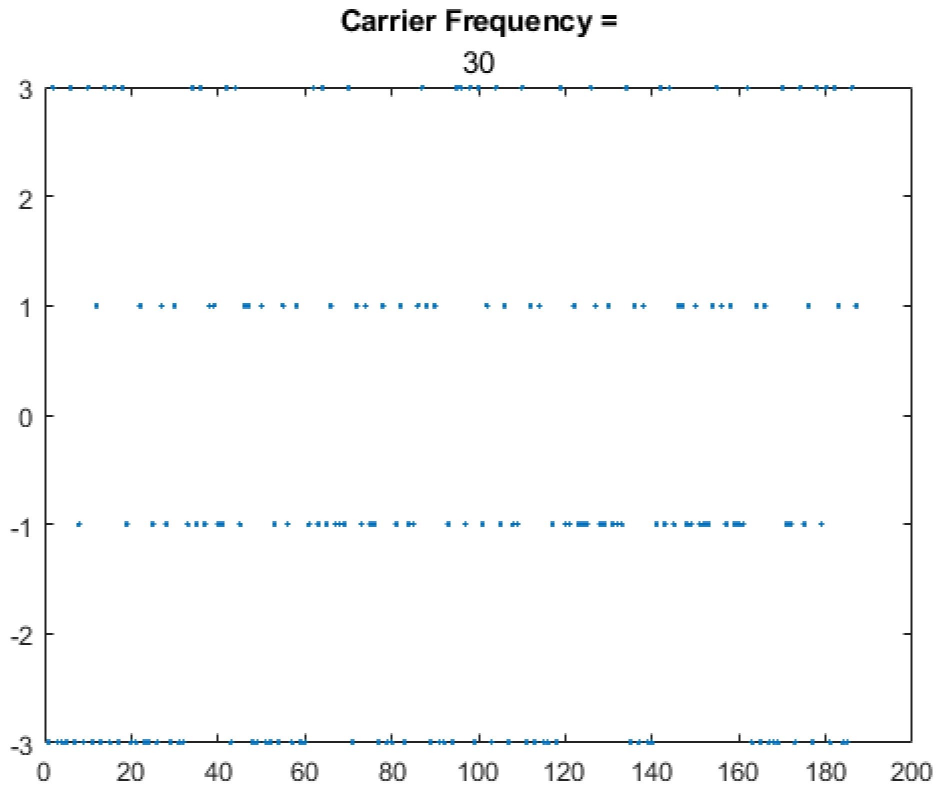
```
str='01234 I wish I were an Oscar Meyer wiener 56789';
m=letters2pam(str); N=length(m); % 4-level signal of length N
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
M=100;                        % oversampling factor
mup=zeros(1,N*M);             % Hamming pulse filter with
mup(1:M:N*M)=m;               % T/M-spaced impulse response
p=hamming(M);                 % blip pulse of width M
x=filter(p,1,mup);            % convolve pulse shape with data
figure, plotspec(x,1/M)    % baseband AM modulation
t=1/M:1/M:length(x)/M;        % T/M-spaced time vector
fc=[50, 30, 3, 1, 0.5];                        % carrier frequency
for index=1:5
c=cos(2*pi*fc(index)*t);              % carrier
r=c.*x;                       % modulate message with carrier
% am demodulation of received signal sequence r
c2=cos(2*pi.*fc(index)*t);              % synchronized cosine for
 mixing
x2=r.*c2;                      % demod received signal
fl=50; fbe=[0 0.1 0.2 1];      % LPF parameters
damps=[1 1 0 0 ];
b=firpm(fl,fbe,damps);         % create LPF impulse response
x3=2*filter(b,1,x2);           % LPF and scale signal
% extract upsampled pulses using correlation implemented
% as a convolving filter; filter with pulse and normalize
y=filter(fliplr(p)/(pow(p)*M),1,x3);
% set delay to first symbol-sample and increment by M
z=y(0.5*fl+M:M:N*M);           % downsample to symbol rate
figure, plot([1:length(z)],z,'.') % plot soft decisions
title('Carrier Frequency =', num2str(fc(index)))
% decision device and symbol matching performance assessment
mprime=quantalph(z,[-3,-1,1,3])'; % quantize alphabet
cvar=(mprime-z)*(mprime-z)'/length(mprime), % cluster variance
lmp=length(mprime);
pererr=100*sum(abs(sign(mprime-m(1:lmp))))/lmp, % symbol error
% decode decision device output to text string
reconstructed_message=pam2letters(mprime)
%Discussion: According to the textbook, Nyquist sampling of the
 received signal occurs when
```

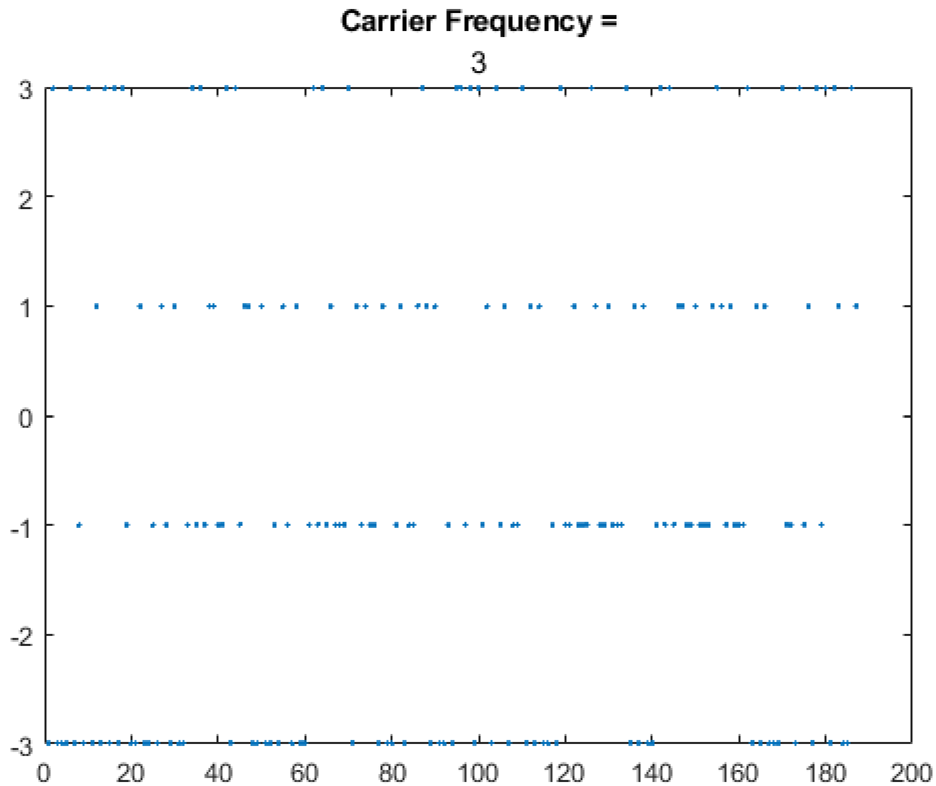
```
%the sample frequency is twice that of the highest frequency in the
%received signal, which is why 0.5 did not work for a carrier
 frequency.
end
```

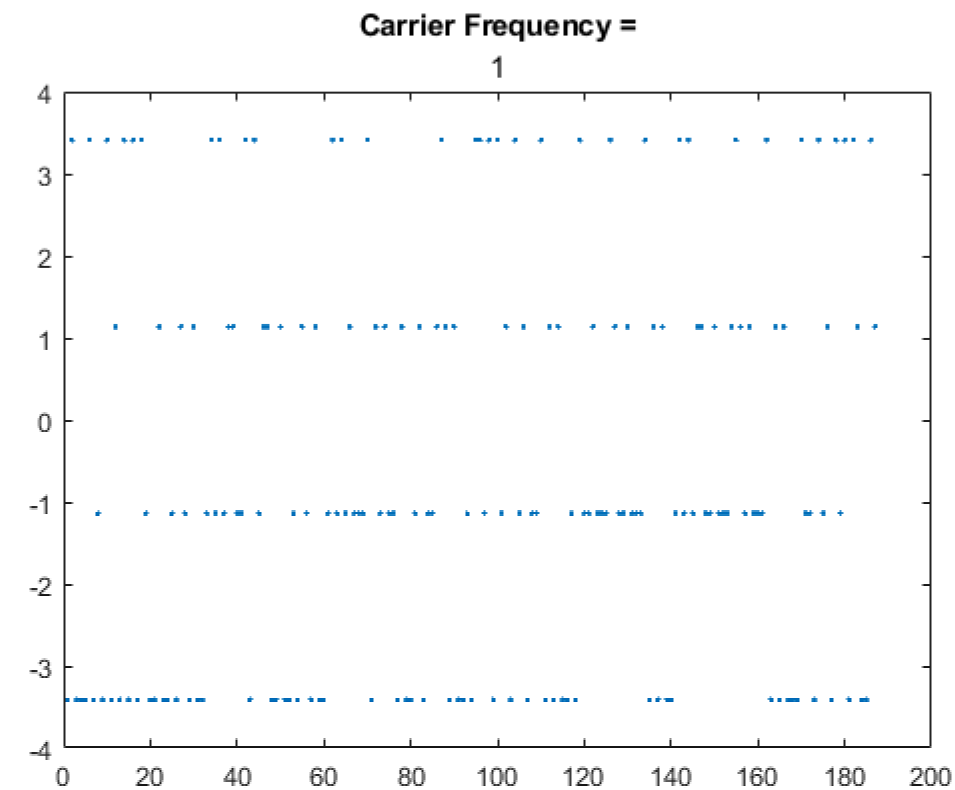
```
cvar =

    4.8454


pererr =

     0


ans =

    'dropping last 3 PAM symbols'


reconstructed_message =

    '01234 I wish I were an Oscar Meyer wiener 5678'


cvar =

   2.9259e-05


pererr =

     0


ans =

    'dropping last 3 PAM symbols'


reconstructed_message =

    '01234 I wish I were an Oscar Meyer wiener 5678'


cvar =

   4.1304e-05


pererr =

     0
```

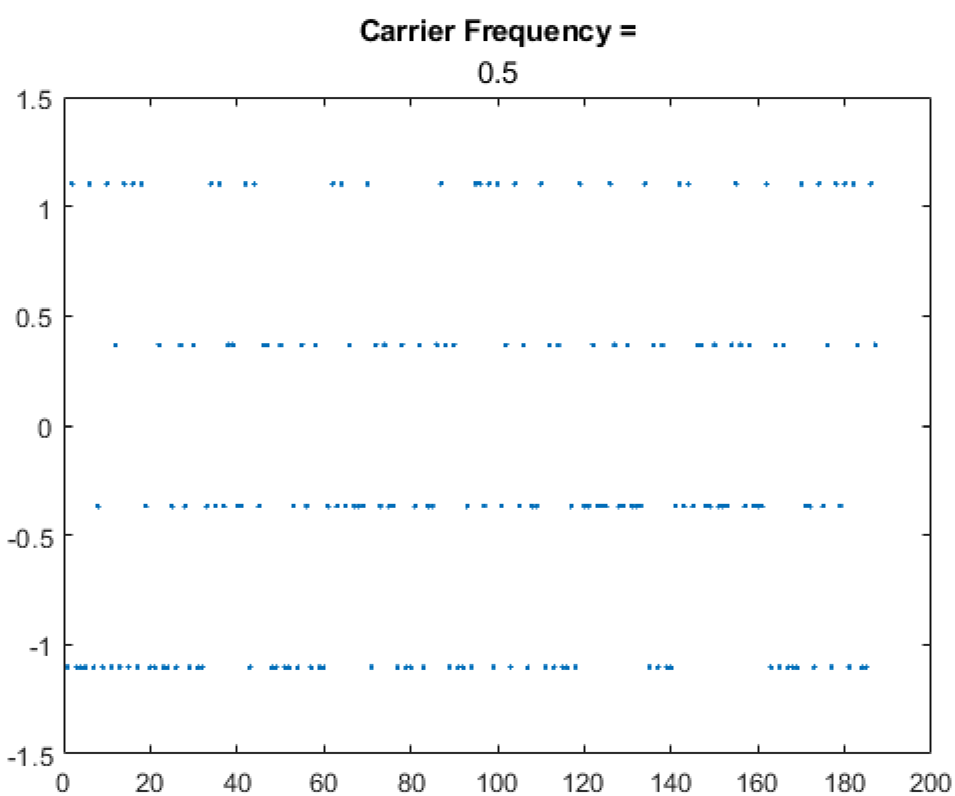
```
ans =

    'dropping last 3 PAM symbols'

reconstructed_message =

    '01234 I wish I were an Oscar Meyer wiener 5678'

cvar =

    0.0911

pererr =

     0

ans =

    'dropping last 3 PAM symbols'

reconstructed_message =

    '01234 I wish I were an Oscar Meyer wiener 5678'

cvar =

    0.2104

pererr =

   48.6631

ans =

    'dropping last 3 PAM symbols'

reconstructed_message =

    'eeffeeYefifieYefefeeejeZffefeYeiefefiejefeeffi'
```


4
2
0
-2
-4
amplitude
0
20
40
60
80
100
120
140
160
180
200
seconds
10000
5000
0
magnitude
-50
-40
-30
-20
-10
0
10
20
30
40
50
frequency
Carrier Frequency =
50
6
4
2
0
-2
-4
-6
0
20
40
60
80
100
120
140
160
180
200


Carrier Frequency =
30
3
2
1
0
-1
-2
-3
0
20
40
60
80
100
120
140
160
180
200


Carrier Frequency =
3
3
2
1
0
-1
-2
-3
0
20
40
60
80
100
120
140
160
180
200


Carrier Frequency =
1
4
3
2
1
0
-1
-2
-3
-4
0
20
40
60
80
100
120
140
160
180
200


Carrier Frequency =
0.5
1.5
1
0.5
0
-0.5
-1
-1.5
0
20
40
60
80
100
120
140
160
180
200

Exercise 9.2

encode text string as T-spaced 4-PAM sequence

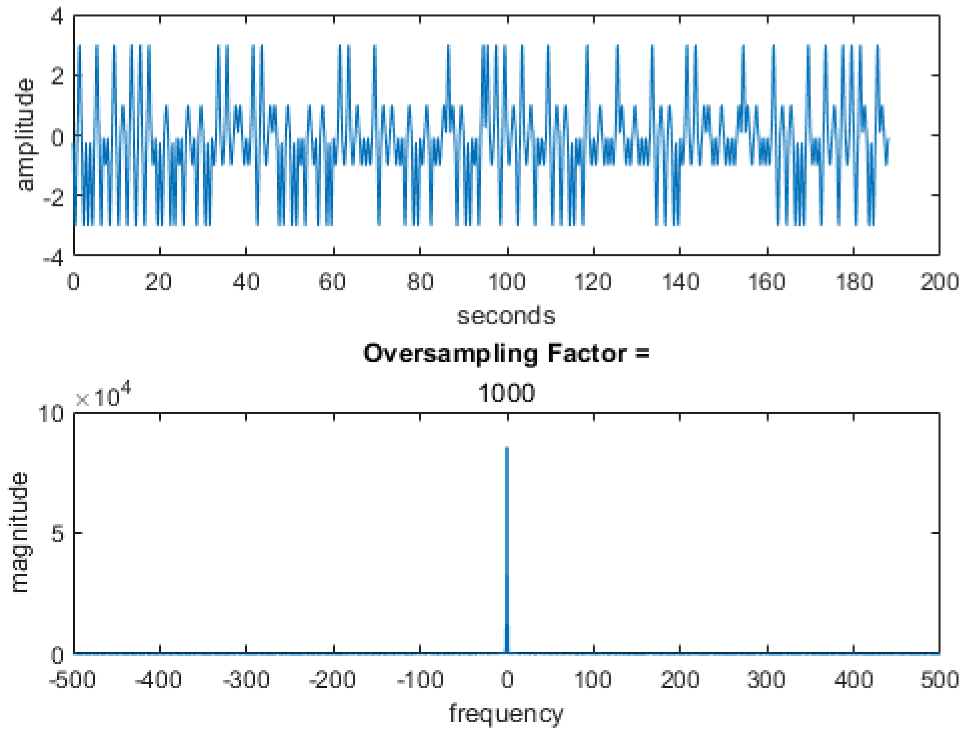
```
str='01234 I wish I were an Oscar Meyer wiener 56789';
m=letters2pam(str); N=length(m); % 4-level signal of length N
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
M=[1000, 25, 10];                          % oversampling factor
for index = 1:3
mup=zeros(1,N*M(index));                % Hamming pulse filter with
mup(1:M(index):N*M(index))=m;                  % T/M-spaced impulse
 response
p=hamming(M(index));                    % blip pulse of width M
x=filter(p,1,mup);               % convolve pulse shape with data
figure, plotspec(x,1/M(index))    % baseband AM modulation
title('Oversampling Factor =', num2str(M(index)))
t=1/M(index):1/M(index):length(x)/M(index);        % T/M-spaced time
 vector
fc=20;                           % carrier frequency
c=cos(2*pi*fc*t);                % carrier
r=c.*x;                          % modulate message with carrier
% am demodulation of received signal sequence r
c2=cos(2*pi*fc*t);                % synchronized cosine for mixing
x2=r.*c2;                         % demod received signal
fl=50; fbe=[0 0.1 0.2 1];         % LPF parameters
damps=[1 1 0 0 ];
b=firpm(fl,fbe,damps);            % create LPF impulse response
x3=2*filter(b,1,x2);              % LPF and scale signal
% extract upsampled pulses using correlation implemented
% as a convolving filter; filter with pulse and normalize
y=filter(fliplr(p)/(pow(p)*M(index)),1,x3);
% set delay to first symbol-sample and increment by M
z=y(0.5*fl+M(index):M(index):N*M(index));           % downsample to
 symbol rate
figure(2), plot([1:length(z)],z,'.') % plot soft decisions
% decision device and symbol matching performance assessment
mprime=quantalph(z,[-3,-1,1,3])'; % quantize alphabet
cvar=(mprime-z)*(mprime-z)'/length(mprime), % cluster variance
lmp=length(mprime);
pererr=100*sum(abs(sign(mprime-m(1:lmp))))/lmp, % symbol error
% decode decision device output to text string
reconstructed_message=pam2letters(mprime)
%Because M provides padding and expands the bandwidth of the received
%signal before the padding is removed by a LPF, the signal can still
 be
%properly transmitted because there is enough bandwidth to get the
 whole
%signal across. This is why M=25 works but M=10 does not.
end
```

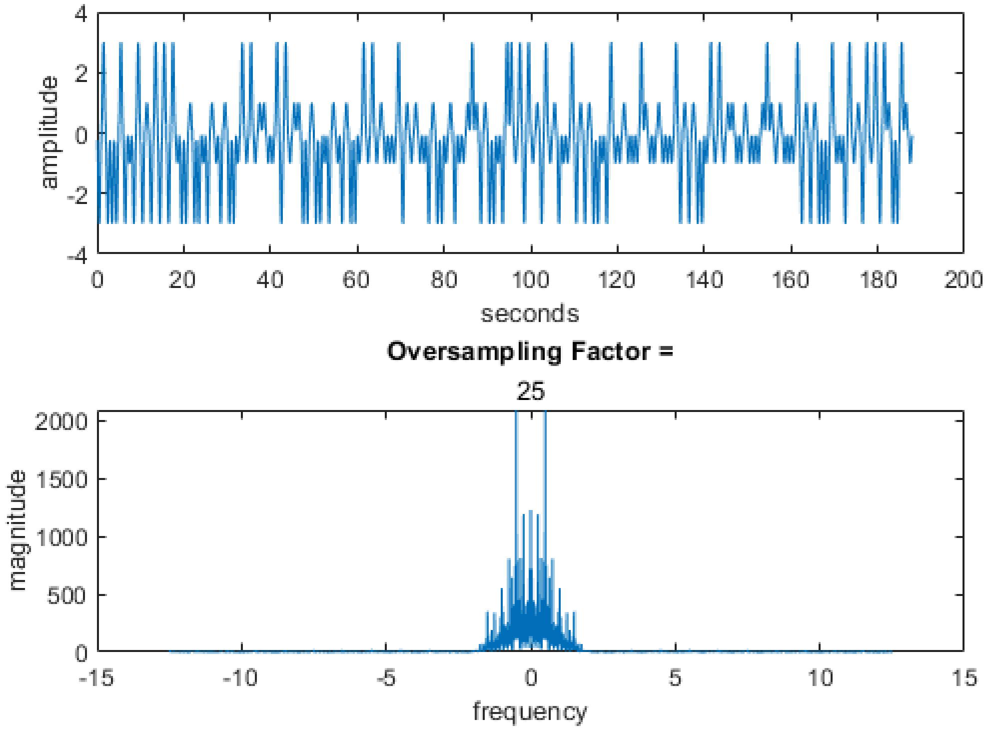
```
cvar =
```

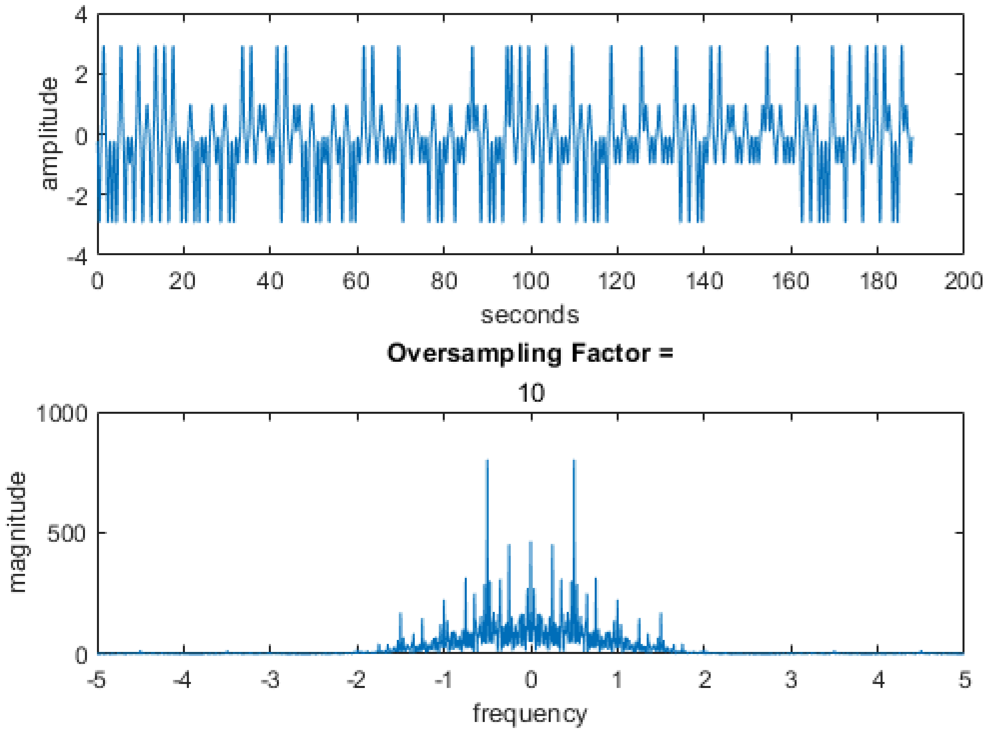
```
   6.5588e-05

pererr =

     0

ans =

    'dropping last 3 PAM symbols'

reconstructed_message =

    '01234 I wish I were an Oscar Meyer wiener 5678'

cvar =

   1.1634e-05

pererr =

     0

ans =

    'dropping last 3 PAM symbols'

reconstructed_message =

    '01234 I wish I were an Oscar Meyer wiener 5678'

cvar =

    2.2154

pererr =

   17.2973

ans =

    'dropping last 1 PAM symbols'
```

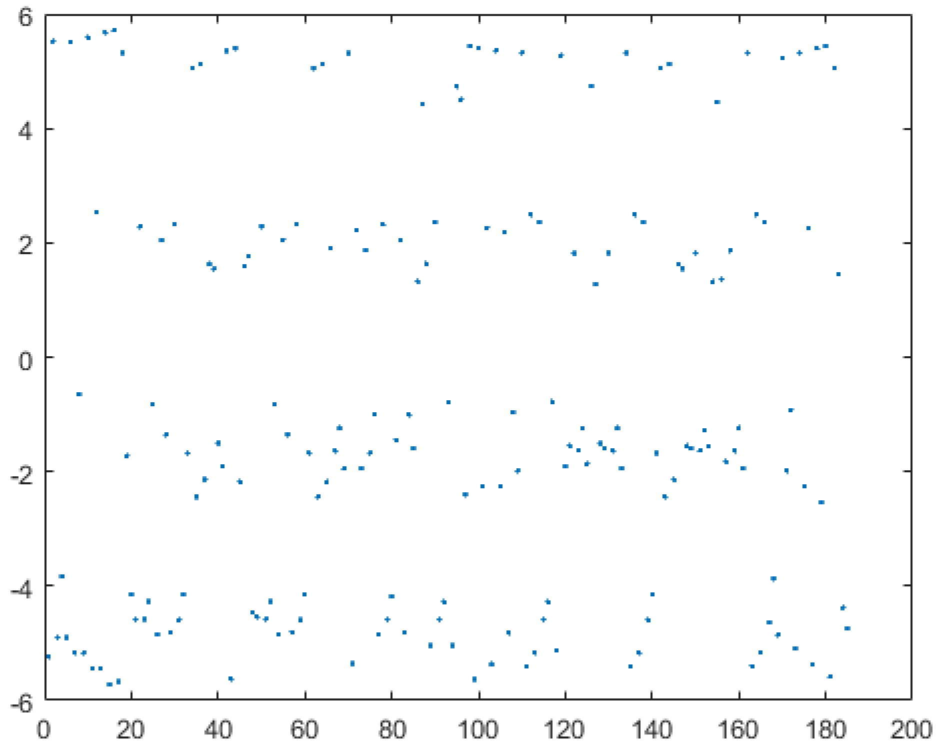
reconstructed_message =

'013340M0s)s(0M0s%se0qn0O331s0Meyes0s)enes05338'




amplitude
seconds
Oversampling Factor =
25
magnitude
frequency


amplitude
seconds
Oversampling Factor =
10
magnitude
frequency



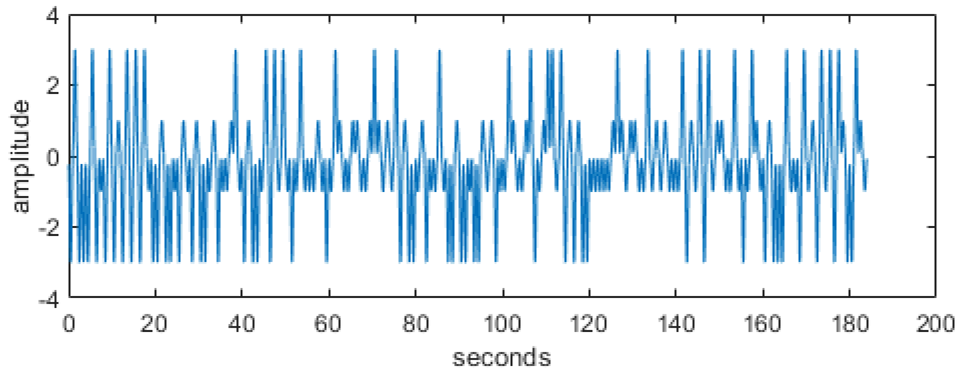
Exercise 9.3

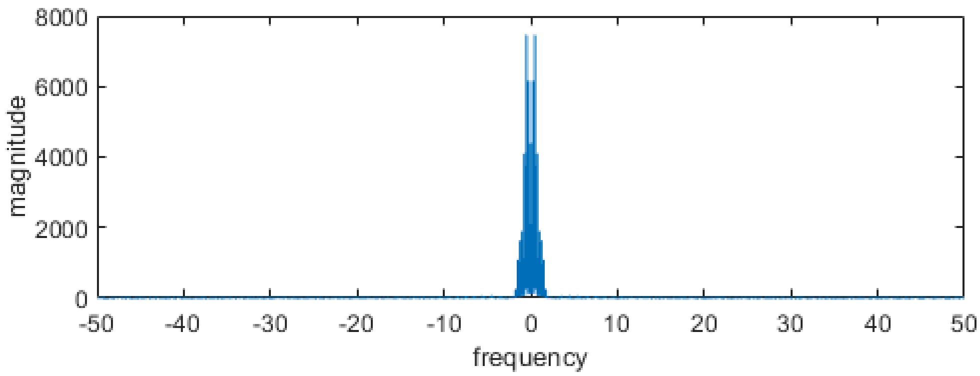
encode text string as T-spaced 4-PAM sequence

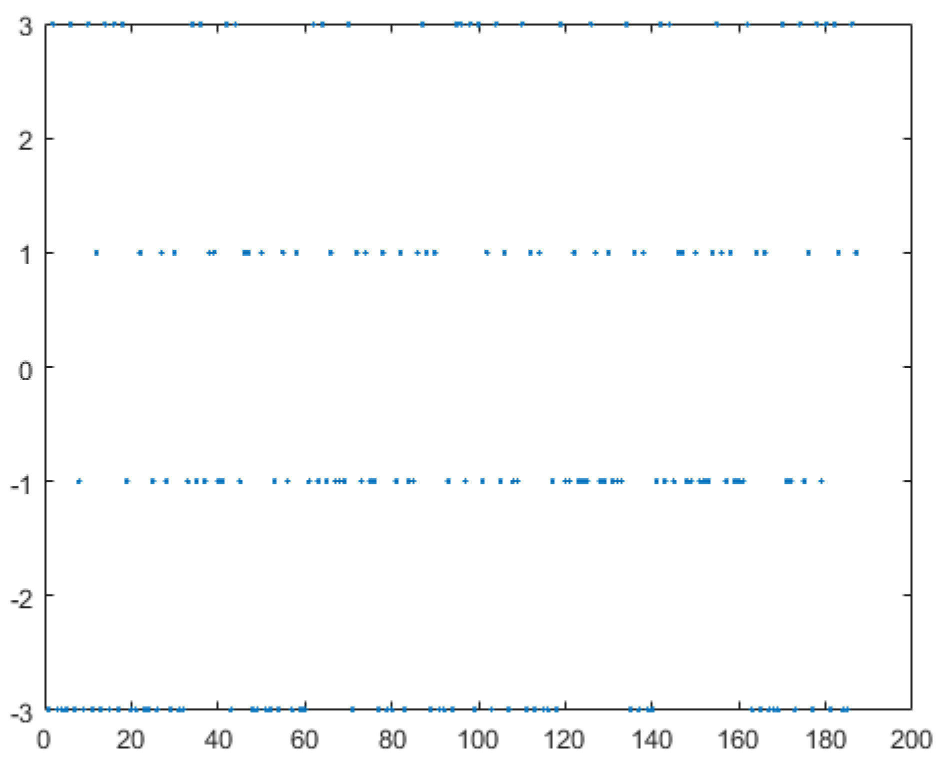
```
str='01234 I wish I were an Oscar Meyer wiener 56789';
m=letters2pam(str); N=length(m); % 4-level signal of length N
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
M=100;                           % oversampling factor
mup=zeros(1,N*M);                % Hamming pulse filter with
mup(1:M:N*M)=m;                  % T/M-spaced impulse response
p=hamming(M);                    % blip pulse of width M
x=filter(p,1,mup);               % convolve pulse shape with data
figure(1), plotspec(x,1/M)       % baseband AM modulation
t=1/M:1/M:length(x)/M;           % T/M-spaced time vector
fc=20;                           % carrier frequency
c=cos(2*pi*fc*t);                % carrier
r=c.*x;                          % modulate message with carrier

% encode text string as T-spaced 4-PAM sequence
str2='01234 I am studying at Baylor University 56789';
m2=letters2pam(str2); N2=length(m2); % 4-level signal of length N
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
M2=100;                           % oversampling factor
mup2=zeros(1,N2*M2);                % Hamming pulse filter with
```

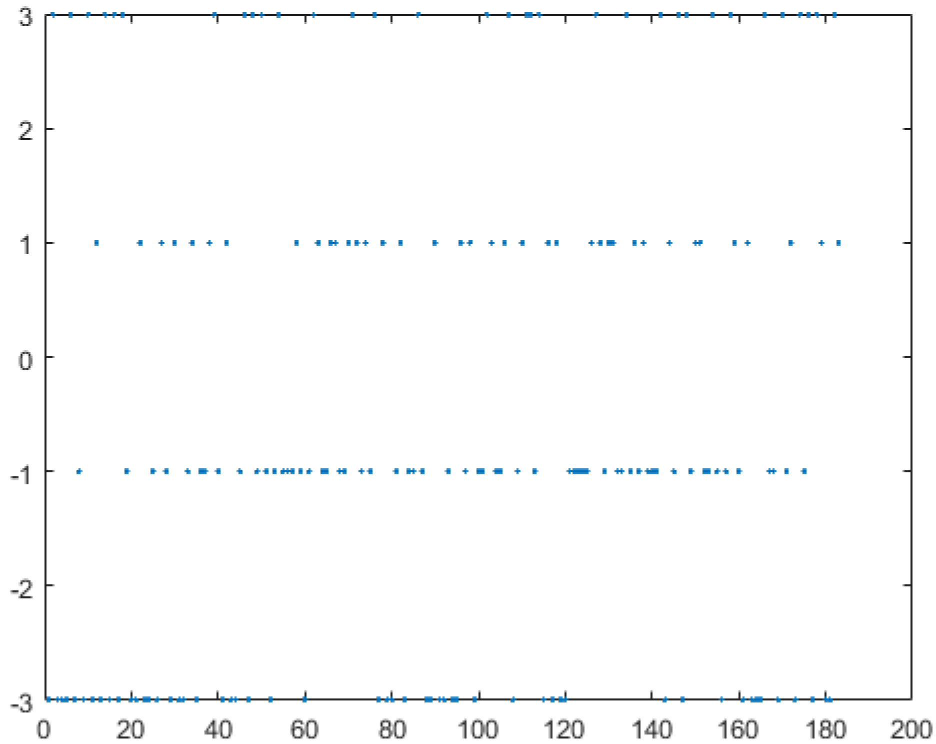
```
mup2(1:M2:N2*M2)=m2;                 % T/M-spaced impulse response
p2=hamming(M2);                   % blip pulse of width M
x5=filter(p2,1,mup2);              % convolve pulse shape with data
figure(1), plotspec(x5,1/M2)      % baseband AM modulation
t2=1/M2:1/M2:length(x5)/M2;          % T/M-spaced time vector
fc2=30;                          % carrier frequency
c5=cos(2*pi*fc2*t2);               % carrier
r2=c5.*x5;                         % modulate message with carrier

% am demodulation of received signal sequence r
c2=cos(2*pi*fc*t);               % synchronized cosine for mixing
x2=r.*c2;                        % demod received signal
c6=cos(2*pi*fc2*t2);
x6=r2.*c6;
fl=50; fbe=[0 0.1 0.2 1];        % LPF parameters
damps=[1 1 0 0 ];
b=firpm(fl,fbe,damps);           % create LPF impulse response
b2=firpm(fl,fbe,damps);
x3=2*filter(b,1,x2);             % LPF and scale signal
x7=2*filter(b,1,x6);
% extract upsample7 pulses using correlation implemented
% as a convolving filter; filter with pulse and normalize
y=filter(fliplr(p)/(pow(p)*M),1,x3);
y2=filter(fliplr(p2)/(pow(p2)*M2),1,x7);
% set delay to first symbol-sample and increment by M
z=y(0.5*fl+M:M:N*M);             % downsample to symbol rate
z2=y2(0.5*fl+M2:M2:N2*M2);
figure(2), plot([1:length(z)],z,'.') % plot soft decisions
figure(3), plot([1:length(z2)],z2,'.')
% decision device and symbol matching performance assessment
mprime=quantalph(z,[-3,-1,1,3])'; % quantize alphabet
cvar=(mprime-z)*(mprime-z)'/length(mprime), % cluster variance
lmp=length(mprime);
pererr=100*sum(abs(sign(mprime-m(1:lmp))))/lmp, % symbol error

mprime2=quantalph(z2,[-3,-1,1,3])'; % quantize alphabet
cvar2=(mprime2-z2)*(mprime2-z2)'/length(mprime2), % cluster variance
lmp2=length(mprime2);
pererr2=100*sum(abs(sign(mprime2-m2(1:lmp2))))/lmp2, % symbol error
% decode decision device output to text string
reconstructed_message=pam2letters(mprime)
reconstructed_message2=pam2letters(mprime2)
%I believe that, if the LPF was removed from the beginning, the signal
%would become more distorted. This is because the LPF removes the
 unwanted
%data points that were generated by the M value for the oversampling
%factor. Without that, the unwanted buffering data points would be in
 the
%final received signal, which would prevent a lot of the actual data
%points from being in the final received signal. Adding another user
 to
%send a signal through the same LPF would result in interference
 between
%the two signals.
```

```
cvar =

   2.9259e-05


pererr =

     0


cvar2 =

   2.8355e-05


pererr2 =

     0


ans =

    'dropping last 3 PAM symbols'


reconstructed_message =

    '01234 I wish I were an Oscar Meyer wiener 5678'


ans =

    'dropping last 3 PAM symbols'


reconstructed_message2 =

    '01234 I am studying at Baylor University 5678'
```


amplitude
seconds


magnitude
frequency



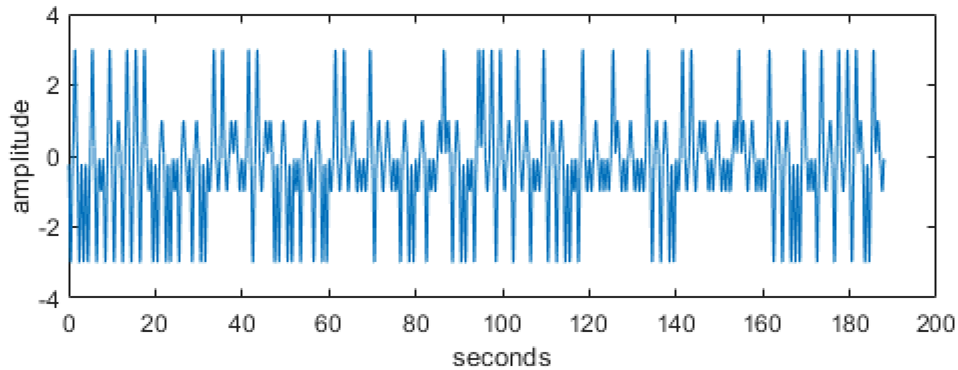


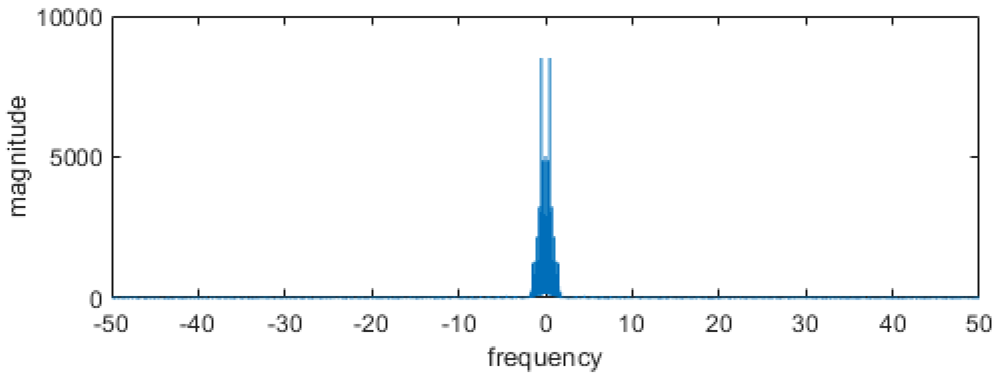
Exercise 9.4

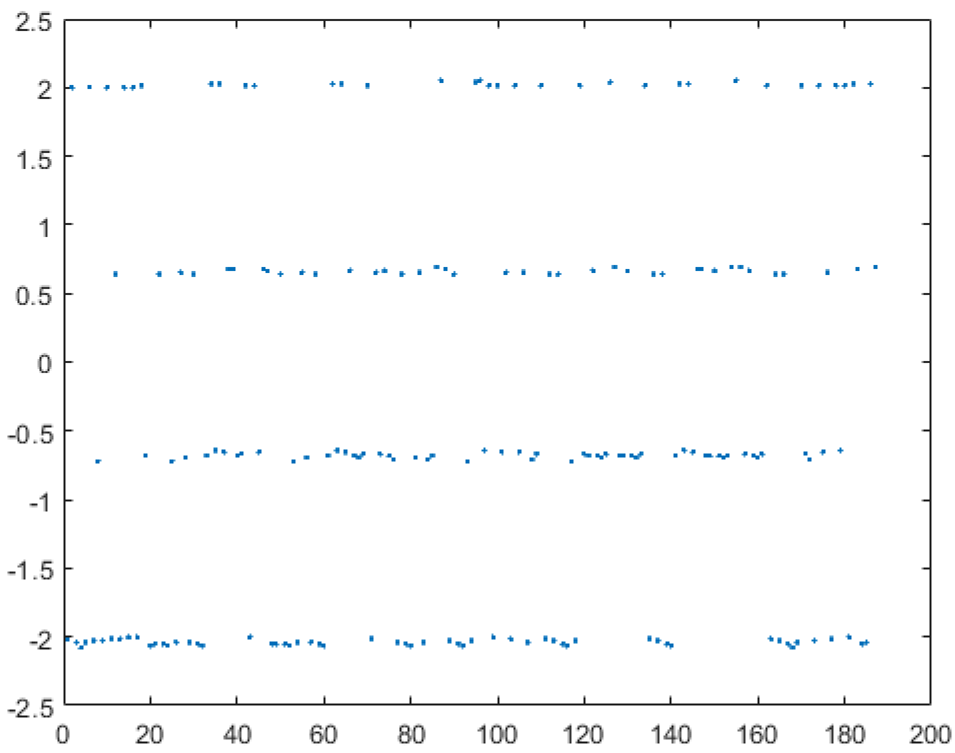
encode text string as T-spaced 4-PAM sequence

```
str='01234 I wish I were an Oscar Meyer wiener 56789';
m=letters2pam(str); N=length(m); % 4-level signal of length N
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
M=100;                          % oversampling factor
mup=zeros(1,N*M);               % Hamming pulse filter with
mup(1:M:N*M)=m;                 % T/M-spaced impulse response
p=hamming(M);                   % blip pulse of width M
x=filter(p,1,mup);              % convolve pulse shape with data
figure(1), plotspec(x,1/M)      % baseband AM modulation
t=1/M:1/M:length(x)/M;          % T/M-spaced time vector
fc=20;                          % carrier frequency
c=cos(2*pi*fc*t);               % carrier
r=c.*x;                         % modulate message with carrier
% am demodulation of received signal sequence r
c2=cos(2*pi*fc*t);               % synchronized cosine for mixing
x2=r.*c2;                        % demod received signal
fl=50; fbe=[0 0.0124 0.0177 1];        % LPF parameters
damps=[1 1 0 0];
b=firpm(fl,fbe,damps);           % create LPF impulse response
x3=2*filter(b,1,x2);             % LPF and scale signal
% extract upsampled pulses using correlation implemented
```

```
% as a convolving filter; filter with pulse and normalize
y=filter(fliplr(p)/(pow(p)*M),1,x3);
% set delay to first symbol-sample and increment by M
z=y(0.5*fl+M:M:N*M);           % downsample to symbol rate
figure(2), plot([1:length(z)],z,'.') % plot soft decisions
% decision device and symbol matching performance assessment
mprime=quantalph(z,[-3,-1,1,3])'; % quantize alphabet
cvar=(mprime-z)*(mprime-z)'/length(mprime), % cluster variance
lmp=length(mprime);
pererr=100*sum(abs(sign(mprime-m(1:lmp))))/lmp, % symbol error
% decode decision device output to text string
reconstructed_message=pam2letters(mprime)

%With a Nyquist sampling frequency of 50, the lowest normalized
 frequency
%that will correctly output the desired signal is 0.0176. There does
 not
%appear to be an upper limit to where the LPF should have the cutoff
%frequency for it to work properly.


cvar =

    0.5120


pererr =

     0


ans =

    'dropping last 3 PAM symbols'


reconstructed_message =

    '01234 I wish I were an Oscar Meyer wiener 5678'
```


amplitude
seconds


magnitude
frequency



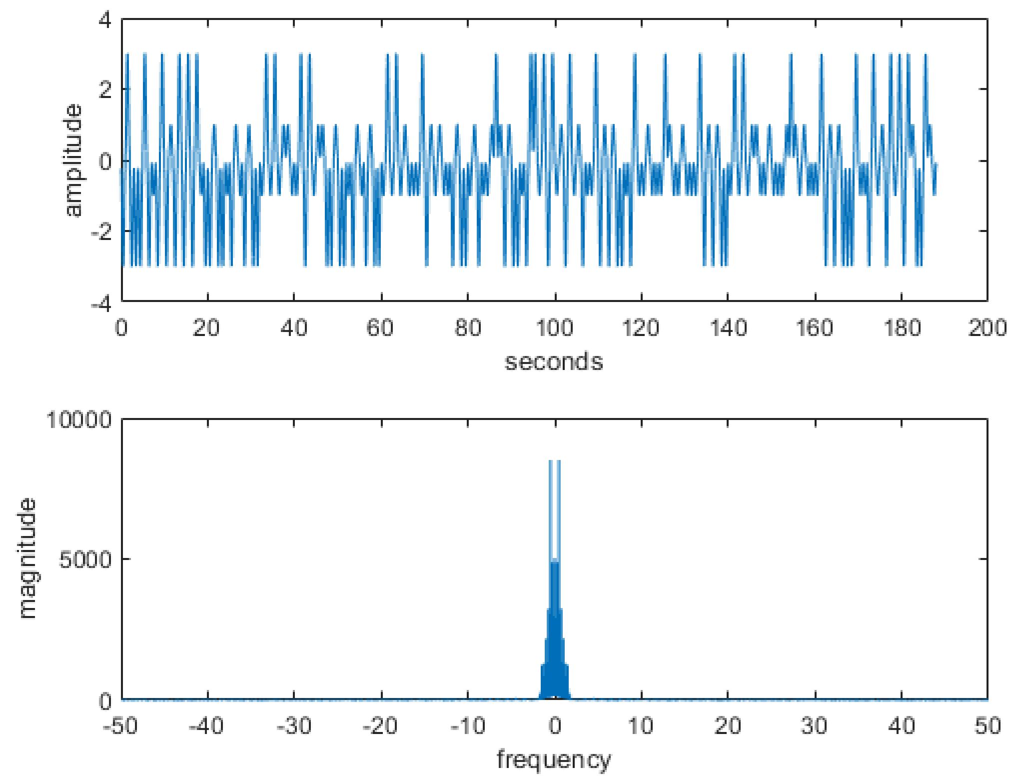
Exercise 9.5

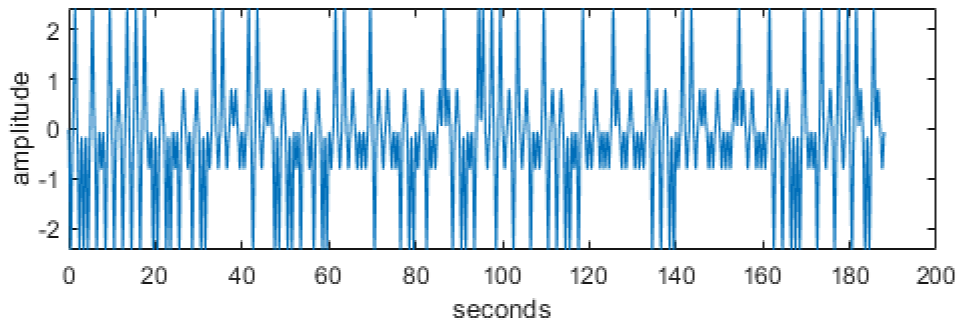
encode text string as T-spaced 4-PAM sequence

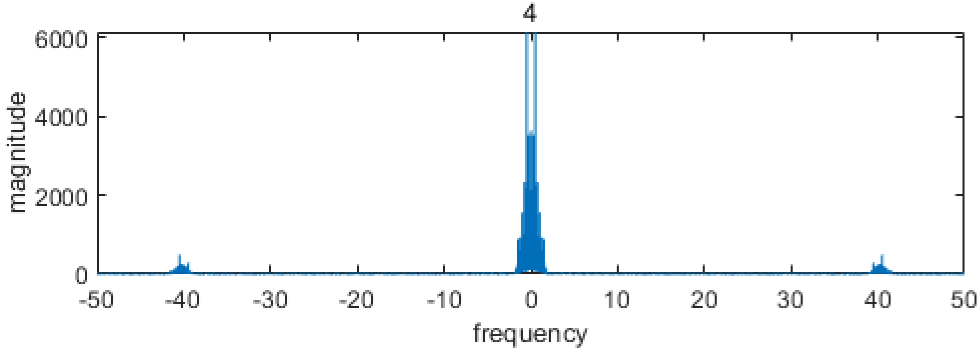
```
str='01234 I wish I were an Oscar Meyer wiener 56789';
m=letters2pam(str); N=length(m); % 4-level signal of length N
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
M=100;                        % oversampling factor
mup=zeros(1,N*M);             % Hamming pulse filter with
mup(1:M:N*M)=m;               % T/M-spaced impulse response
p=hamming(M);                 % blip pulse of width M
x=filter(p,1,mup);            % convolve pulse shape with data
figure(1), plotspec(x,1/M)    % baseband AM modulation
t=1/M:1/M:length(x)/M;        % T/M-spaced time vector
fc=20;                        % carrier frequency
c=cos(2*pi*fc*t);             % carrier
r=c.*x;                       % modulate message with carrier
% am demodulation of received signal sequence r
c2=cos(2*pi*fc*t);             % synchronized cosine for mixing
x2=r.*c2;                      % demod received signal
fl=4; fbe=[0 0.1 0.2 1];      % LPF parameters
damps=[1 1 0 0 ];
b=firpm(fl,fbe,damps);         % create LPF impulse response
x3=2*filter(b,1,x2);           % LPF and scale signal
% extract upsampled pulses using correlation implemented
% as a convolving filter; filter with pulse and normalize
y=filter(fliplr(p)/(pow(p)*M),1,x3);
% set delay to first symbol-sample and increment by M
z=y(0.5*fl+M:M:N*M);           % downsample to symbol rate
figure(2), plot([1:length(z)],z,'.') % plot soft decisions
% decision device and symbol matching performance assessment
mprime=quantalph(z,[-3,-1,1,3])'; % quantize alphabet
cvar=(mprime-z)*(mprime-z)'/length(mprime), % cluster variance
lmp=length(mprime);
pererr=100*sum(abs(sign(mprime-m(1:lmp))))/lmp, % symbol error
% decode decision device output to text string
reconstructed_message=pam2letters(mprime)
plotspec(x3, 1/M)
title('Number of Terms =', num2str(fl))
%At an fl value of 4, the side lobes at 40 begin to reappear, and is
 the
%last value of fl before the received message begins to become
 inaccurate
%to the transmitted message. At an fl value of 3, which is the minimum
%value that that value can hold, the message is completely gone, and
 the
%side lobes at +/- 40 are higher. These can be found using the
 plotspec
%function.
```

```
cvar =
```

```
    0.3810

pererr =

     0

ans =

    'dropping last 3 PAM symbols'

reconstructed_message =

    '01234 I wish I were an Oscar Meyer wiener 5678'
```







Question 2

encode text string as T-spaced 4-PAM sequence

```
str='01234 I wish I were an Oscar Meyer wiener 56789';
m=letters2pam(str); N=length(m); % 4-level signal of length N
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
M=100;                        % oversampling factor
mup=zeros(1,N*M);             % Hamming pulse filter with
mup(1:M:N*M)=m;               % T/M-spaced impulse response
p=hamming(M);                 % blip pulse of width M
x=filter(p,1,mup);            % convolve pulse shape with data
figure(1), plotspec(x,1/M)    % baseband AM modulation
t=1/M:1/M:length(x)/M;        % T/M-spaced time vector
fc=20;                        % carrier frequency
c=cos(2*pi*fc*t);             % carrier
r=c.*x;                       % modulate message with carrier
% am demodulation of received signal sequence r
c2=cos(2*pi*fc*t);             % synchronized cosine for mixing
x2=r.*c2;                      % demod received signal
fl=50; fbe=[0 0.1 0.2 1];      % LPF parameters
damps=[1 1 0 0 ];
b=firpm(fl,fbe,damps);         % create LPF impulse response
x3=2*filter(b,1,x2);           % LPF and scale signal
% extract upsampled pulses using correlation implemented
```

```
% as a convolving filter; filter with pulse and normalize
y=filter(fliplr(p)/(pow(p)*M),1,x3);
% set delay to first symbol-sample and increment by M
z=y(0.5*fl+M:M:N*M);           % downsample to symbol rate
figure(2), plot([1:length(z)],z,'.') % plot soft decisions
% decision device and symbol matching performance assessment
mprime=quantalph(z,[-1-1i,-1+1i,1-1i,1+1i])' % quantize alphabet

%mprime, which is the array of modulated values, shows the array with
 the
%values of -1-j, -1+jj, 1-j, and 1+j replacing the PAM values of -3,
 -1, 1,
%and 3, respectively. This was done by using the quantalph function,
 and
%passing in z.


mprime =

  Columns 1 through 4

  -1.0000 - 1.0000i   1.0000 + 1.0000i  -1.0000 - 1.0000i  -1.0000 -
 1.0000i

  Columns 5 through 8

  -1.0000 - 1.0000i   1.0000 + 1.0000i  -1.0000 - 1.0000i  -1.0000 +
 1.0000i

  Columns 9 through 12

  -1.0000 - 1.0000i   1.0000 + 1.0000i  -1.0000 - 1.0000i   1.0000 -
 1.0000i

  Columns 13 through 16

  -1.0000 - 1.0000i   1.0000 + 1.0000i  -1.0000 - 1.0000i   1.0000 +
 1.0000i

  Columns 17 through 20

  -1.0000 - 1.0000i   1.0000 + 1.0000i  -1.0000 + 1.0000i  -1.0000 -
 1.0000i

  Columns 21 through 24

  -1.0000 - 1.0000i   1.0000 - 1.0000i  -1.0000 - 1.0000i  -1.0000 -
 1.0000i

  Columns 25 through 28

  -1.0000 + 1.0000i  -1.0000 - 1.0000i   1.0000 - 1.0000i  -1.0000 +
 1.0000i
```

Columns 29 through 32

-1.0000 - 1.0000i   1.0000 - 1.0000i -1.0000 - 1.0000i -1.0000 - 1.0000i

Columns 33 through 36

-1.0000 + 1.0000i   1.0000 + 1.0000i -1.0000 + 1.0000i   1.0000 + 1.0000i

Columns 37 through 40

-1.0000 + 1.0000i   1.0000 - 1.0000i   1.0000 - 1.0000i -1.0000 + 1.0000i

Columns 41 through 44

-1.0000 + 1.0000i   1.0000 + 1.0000i -1.0000 - 1.0000i   1.0000 + 1.0000i

Columns 45 through 48

-1.0000 + 1.0000i   1.0000 - 1.0000i   1.0000 - 1.0000i -1.0000 - 1.0000i

Columns 49 through 52

-1.0000 - 1.0000i   1.0000 - 1.0000i -1.0000 - 1.0000i -1.0000 - 1.0000i

Columns 53 through 56

-1.0000 + 1.0000i -1.0000 - 1.0000i   1.0000 - 1.0000i -1.0000 + 1.0000i

Columns 57 through 60

-1.0000 - 1.0000i   1.0000 - 1.0000i -1.0000 - 1.0000i -1.0000 - 1.0000i

Columns 61 through 64

-1.0000 + 1.0000i   1.0000 + 1.0000i -1.0000 + 1.0000i   1.0000 + 1.0000i

Columns 65 through 68

-1.0000 + 1.0000i   1.0000 - 1.0000i -1.0000 + 1.0000i -1.0000 + 1.0000i

Columns 69 through 72

-1.0000 + 1.0000i   1.0000 + 1.0000i -1.0000 - 1.0000i   1.0000 - 1.0000i

Columns 73 through 76

-1.0000 + 1.0000i   1.0000 - 1.0000i -1.0000 + 1.0000i -1.0000 + 1.0000i

Columns 77 through 80

-1.0000 - 1.0000i   1.0000 - 1.0000i -1.0000 - 1.0000i -1.0000 - 1.0000i

Columns 81 through 84

-1.0000 + 1.0000i   1.0000 - 1.0000i -1.0000 - 1.0000i -1.0000 + 1.0000i

Columns 85 through 88

-1.0000 + 1.0000i   1.0000 - 1.0000i   1.0000 + 1.0000i   1.0000 - 1.0000i

Columns 89 through 92

-1.0000 - 1.0000i   1.0000 - 1.0000i -1.0000 - 1.0000i -1.0000 - 1.0000i

Columns 93 through 96

-1.0000 + 1.0000i -1.0000 - 1.0000i   1.0000 + 1.0000i   1.0000 + 1.0000i

Columns 97 through 100

-1.0000 + 1.0000i   1.0000 + 1.0000i -1.0000 - 1.0000i   1.0000 + 1.0000i

Columns 101 through 104

-1.0000 + 1.0000i   1.0000 - 1.0000i -1.0000 - 1.0000i   1.0000 + 1.0000i

Columns 105 through 108

-1.0000 + 1.0000i   1.0000 - 1.0000i -1.0000 - 1.0000i -1.0000 + 1.0000i

Columns 109 through 112

-1.0000 + 1.0000i   1.0000 + 1.0000i -1.0000 - 1.0000i   1.0000 - 1.0000i

Columns 113 through 116

```
 -1.0000 - 1.0000i   1.0000 - 1.0000i  -1.0000 - 1.0000i  -1.0000 -
1.0000i

 Columns 117 through 120

 -1.0000 + 1.0000i  -1.0000 - 1.0000i   1.0000 + 1.0000i  -1.0000 +
1.0000i

 Columns 121 through 124

 -1.0000 + 1.0000i   1.0000 - 1.0000i  -1.0000 + 1.0000i  -1.0000 +
1.0000i

 Columns 125 through 128

 -1.0000 + 1.0000i   1.0000 + 1.0000i   1.0000 - 1.0000i  -1.0000 +
1.0000i

 Columns 129 through 132

 -1.0000 + 1.0000i   1.0000 - 1.0000i  -1.0000 + 1.0000i  -1.0000 +
1.0000i

 Columns 133 through 136

 -1.0000 + 1.0000i   1.0000 + 1.0000i  -1.0000 - 1.0000i   1.0000 -
1.0000i

 Columns 137 through 140

 -1.0000 - 1.0000i   1.0000 - 1.0000i  -1.0000 - 1.0000i  -1.0000 -
1.0000i

 Columns 141 through 144

 -1.0000 + 1.0000i   1.0000 + 1.0000i  -1.0000 + 1.0000i   1.0000 +
1.0000i

 Columns 145 through 148

 -1.0000 + 1.0000i   1.0000 - 1.0000i   1.0000 - 1.0000i  -1.0000 +
1.0000i

 Columns 149 through 152

 -1.0000 + 1.0000i   1.0000 - 1.0000i  -1.0000 + 1.0000i  -1.0000 +
1.0000i

 Columns 153 through 156

 -1.0000 + 1.0000i   1.0000 - 1.0000i   1.0000 + 1.0000i   1.0000 -
1.0000i

 Columns 157 through 160
```

-1.0000 + 1.0000i   1.0000 - 1.0000i -1.0000 + 1.0000i -1.0000 + 1.0000i

Columns 161 through 164

-1.0000 + 1.0000i   1.0000 + 1.0000i -1.0000 - 1.0000i   1.0000 - 1.0000i

Columns 165 through 168

-1.0000 - 1.0000i   1.0000 - 1.0000i -1.0000 - 1.0000i -1.0000 - 1.0000i

Columns 169 through 172

-1.0000 - 1.0000i   1.0000 + 1.0000i -1.0000 + 1.0000i -1.0000 + 1.0000i

Columns 173 through 176

-1.0000 - 1.0000i   1.0000 + 1.0000i -1.0000 + 1.0000i   1.0000 - 1.0000i

Columns 177 through 180

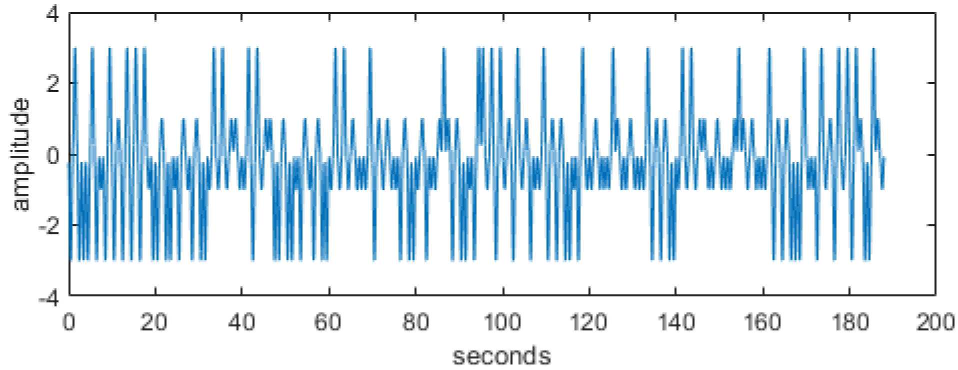
-1.0000 - 1.0000i   1.0000 + 1.0000i -1.0000 + 1.0000i   1.0000 + 1.0000i

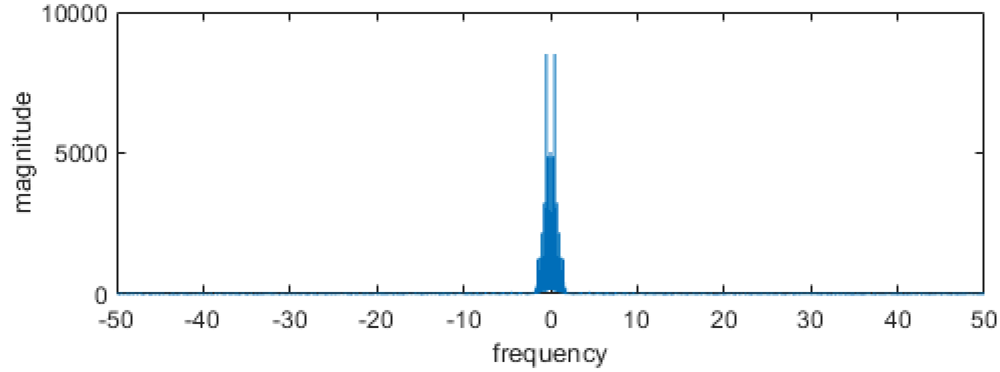
Columns 181 through 184

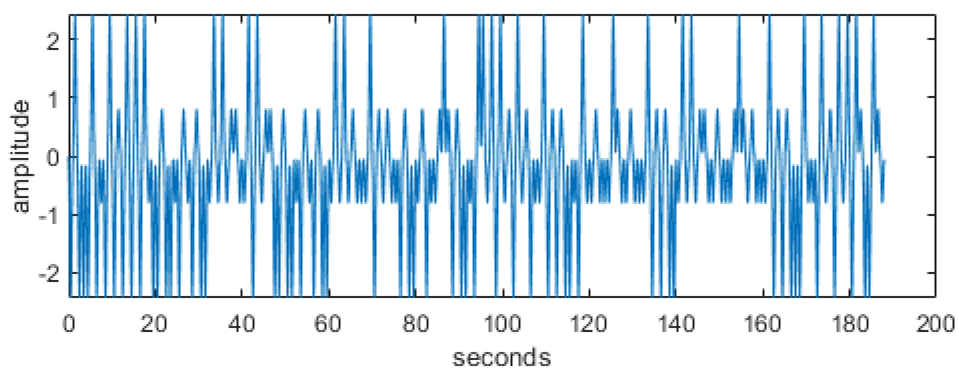
-1.0000 - 1.0000i   1.0000 + 1.0000i   1.0000 - 1.0000i -1.0000 - 1.0000i

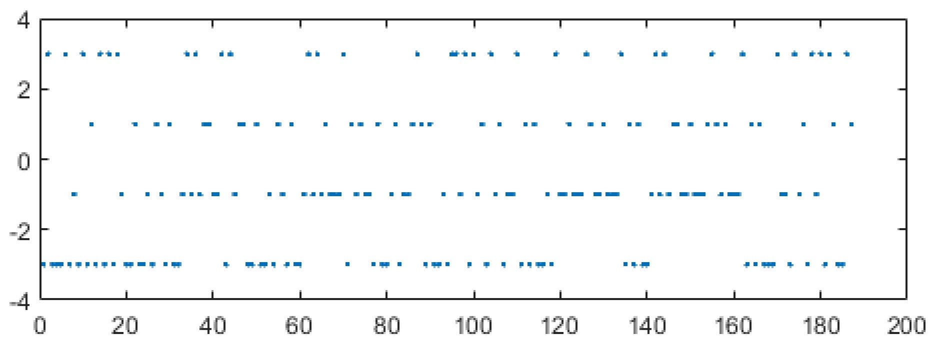
Columns 185 through 187

-1.0000 - 1.0000i   1.0000 + 1.0000i   1.0000 - 1.0000i


amplitude
seconds


magnitude
frequency


amplitude
seconds



Published with MATLAB® R2020b