Table of Contents

Exercise 9.1

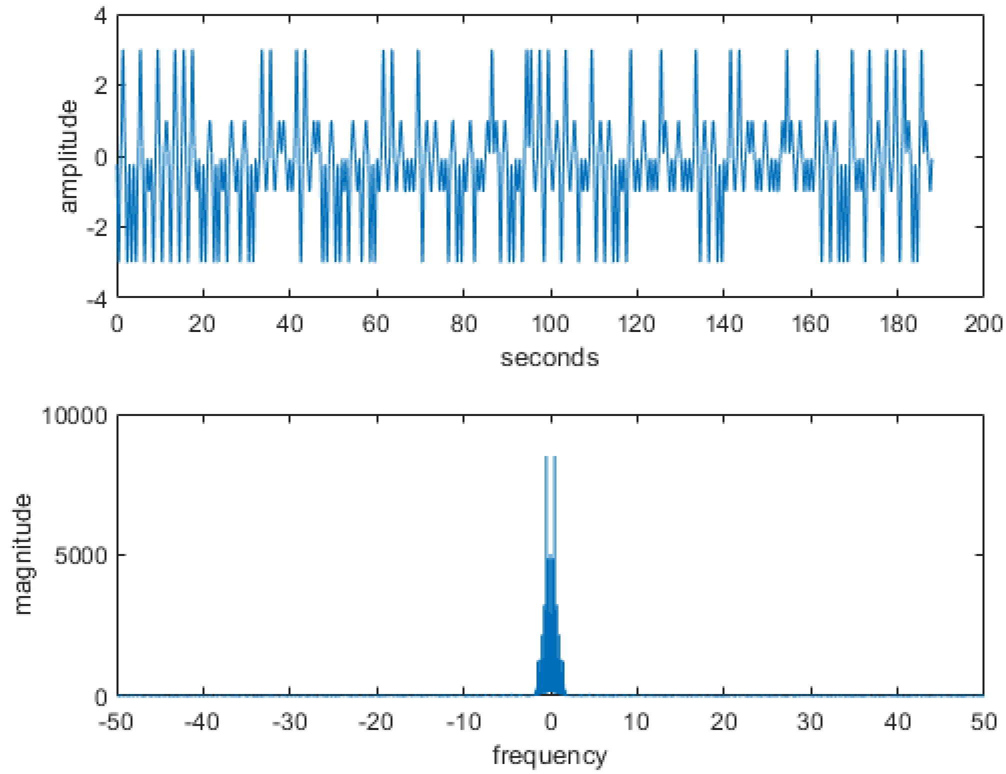
Testing different carrier frequencies fc = 50, 30, 3, 1, 0.5

```
%TRANSMITTER
% 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, 20,3 , 1 , 0.5];    % carrier frequency
for iterator = 1:6
    c=cos(2*pi*fc(iterator)*t);             % carrier
    r=c.*x;                          % modulate message with carrier

    %RECEIVER
    % am demodulation of received signal sequence r
    c2=cos(2*pi*fc(iterator)*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(['Frequency =  ',num2str(fc(iterator)),'Hz'])
    % 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);
```

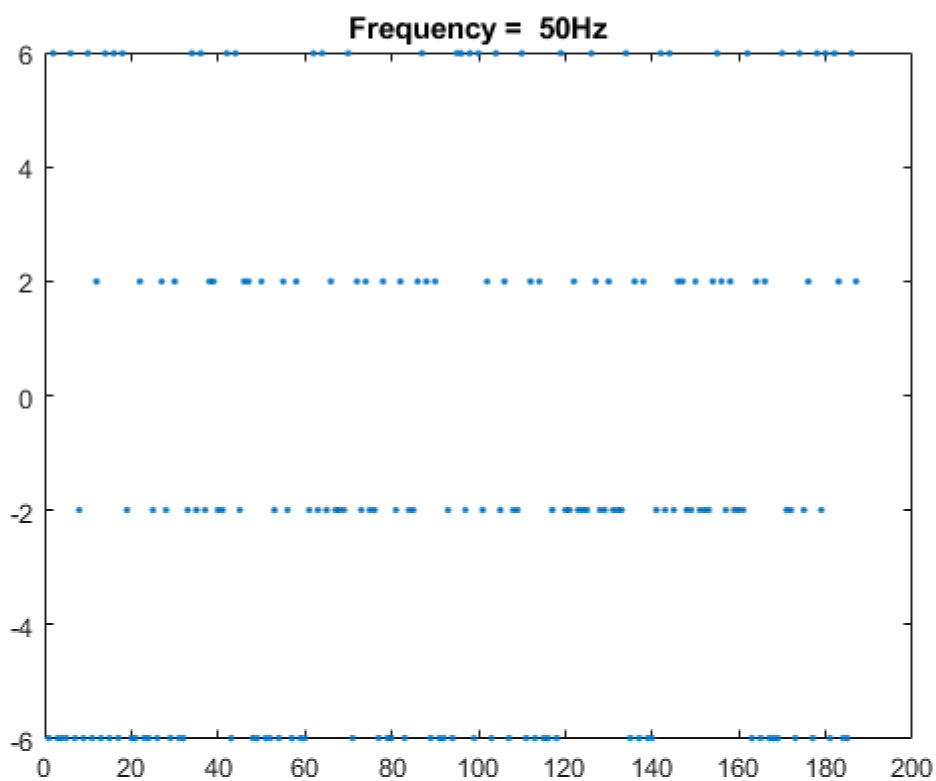
```
    pererr=100*sum(abs(sign(mprime-m(1:lmp))))/lmp, % symbol error
    % decode decision device output to text string
    reconstructed_message=pam2letters(mprime)
    fprintf('This is the reconstructed message ^ using this frequency
 %d\n',fc(iterator));
end

% DISCUSSION
% As long as the sample frequency M is twice the hightest frequency in
 the
% recieved signal, which is the carrier frequency plus the baseband
 signal.
% So as long as the carrier frequency is above 1 it will correctly
% reconstruct the message


cvar =

    4.8454


pererr =

     0


ans =

    'dropping last 3 PAM symbols'


reconstructed_message =

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

This is the reconstructed message ^ using this frequency 50

cvar =

   2.9259e-05


pererr =

     0


ans =

    'dropping last 3 PAM symbols'


reconstructed_message =
```

```
    '01234 I wish I were an Oscar Meyer wiener 5678'
This is the reconstructed message ^ using this frequency 30
cvar =
   2.9259e-05

pererr =
     0

ans =
    'dropping last 3 PAM symbols'

reconstructed_message =
    '01234 I wish I were an Oscar Meyer wiener 5678'
This is the reconstructed message ^ using this frequency 20
cvar =
   4.1304e-05

pererr =
     0

ans =
    'dropping last 3 PAM symbols'

reconstructed_message =
    '01234 I wish I were an Oscar Meyer wiener 5678'
This is the reconstructed message ^ using this frequency 3
cvar =
    0.0911

pererr =
     0
```

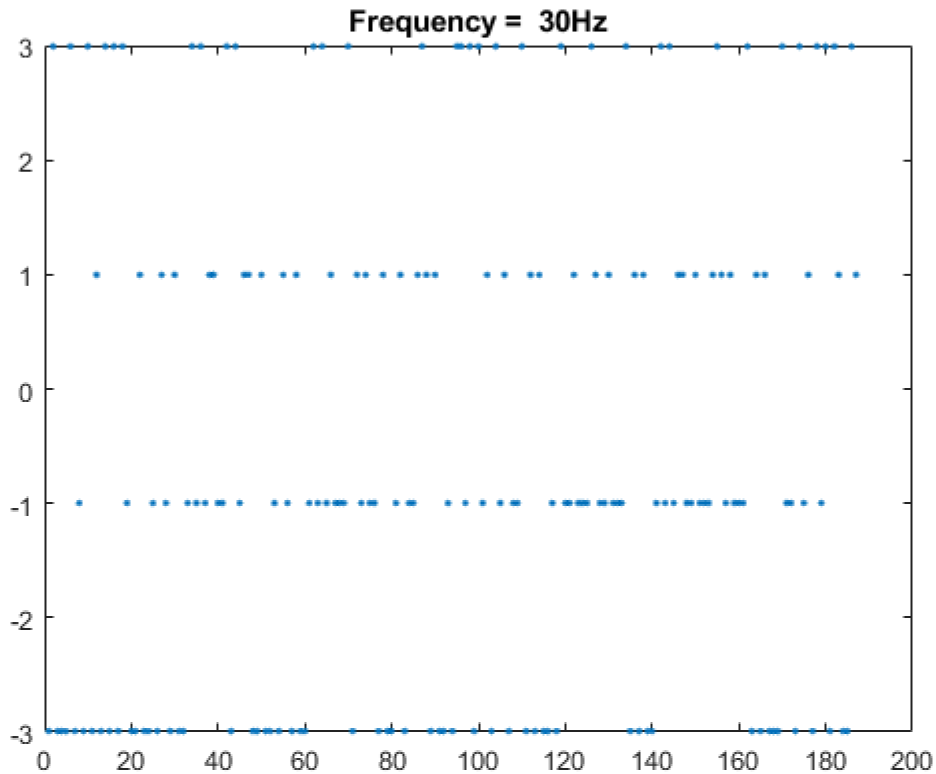
```
ans =

    'dropping last 3 PAM symbols'

reconstructed_message =

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

This is the reconstructed message ^ using this frequency 1

cvar =

    0.2104

pererr =

   48.6631

ans =

    'dropping last 3 PAM symbols'

reconstructed_message =

    'eeffeeYefifieYefefeeejeZffefeYeiefefiejefeeffi'

This is the reconstructed message ^ using this frequency 5.000000e-01
```

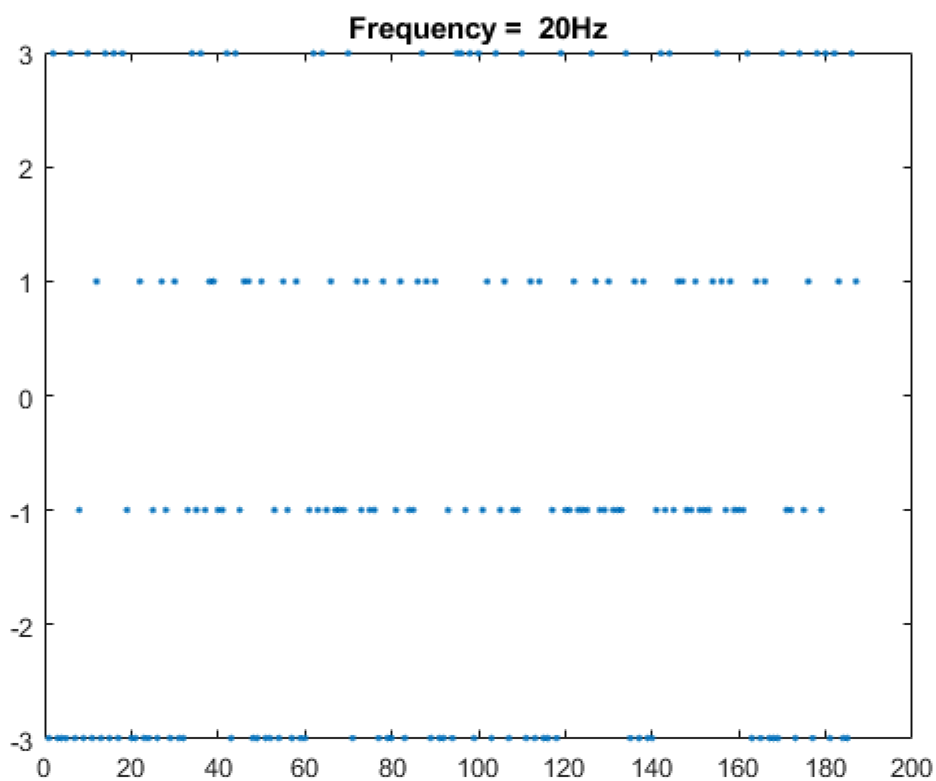


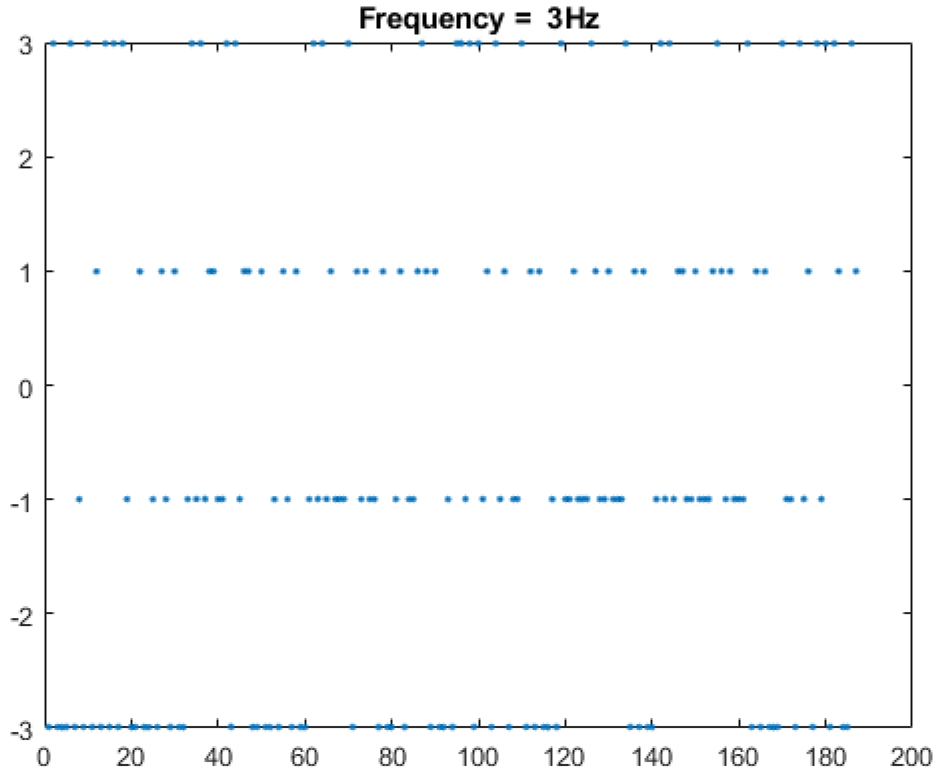
amplitude
seconds
magnitude
frequency

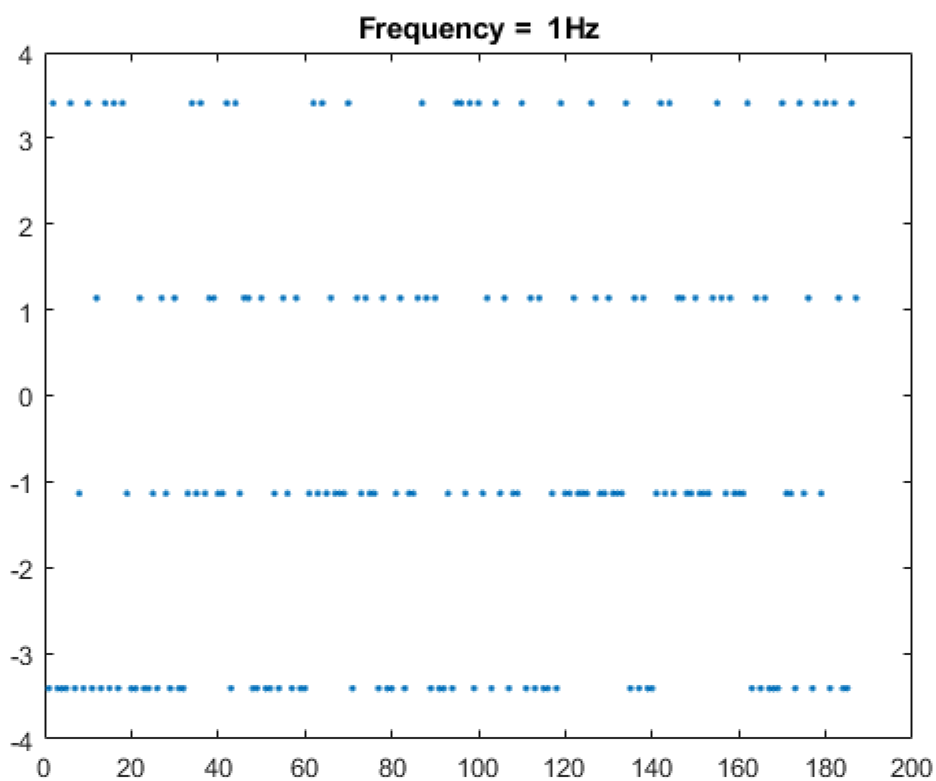


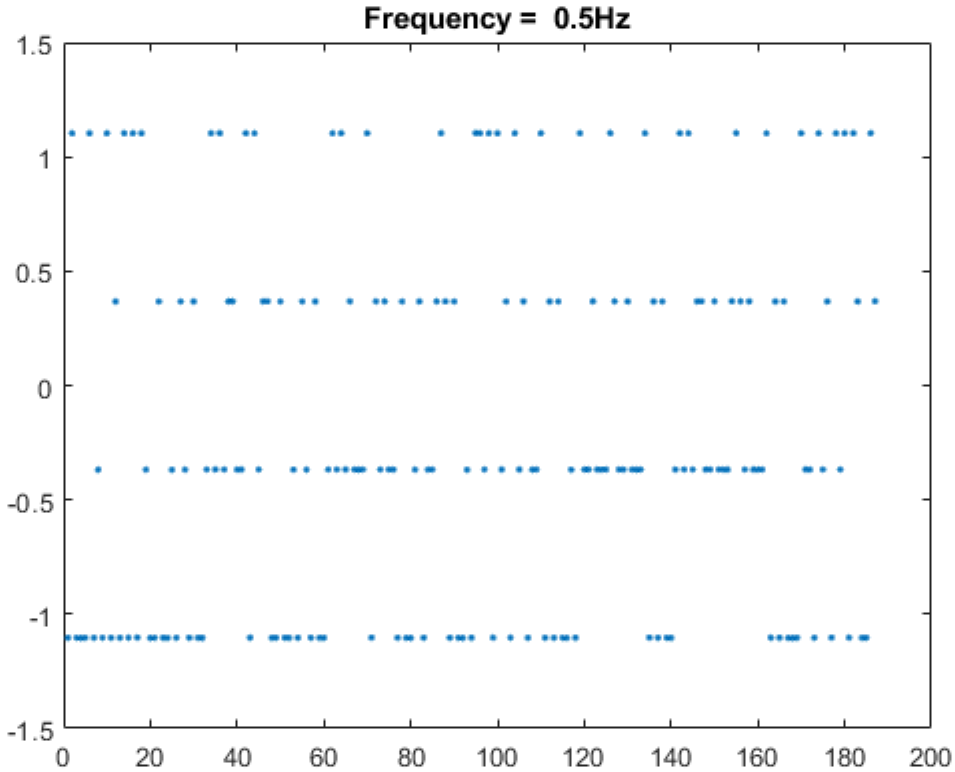
Frequency = 50Hz


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


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


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


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

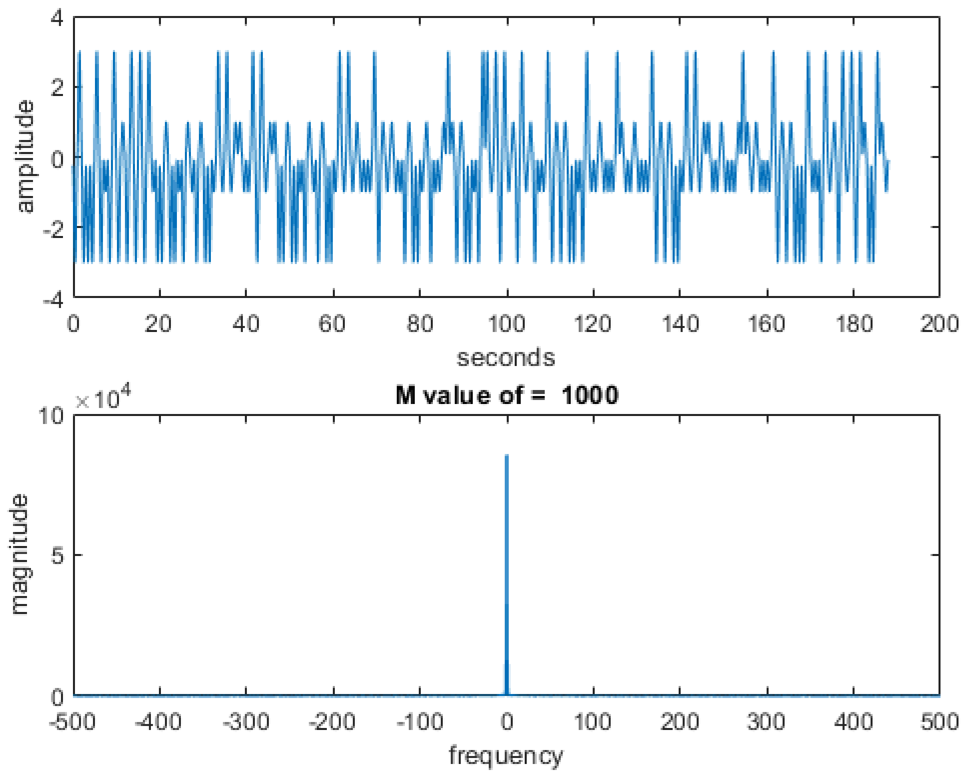


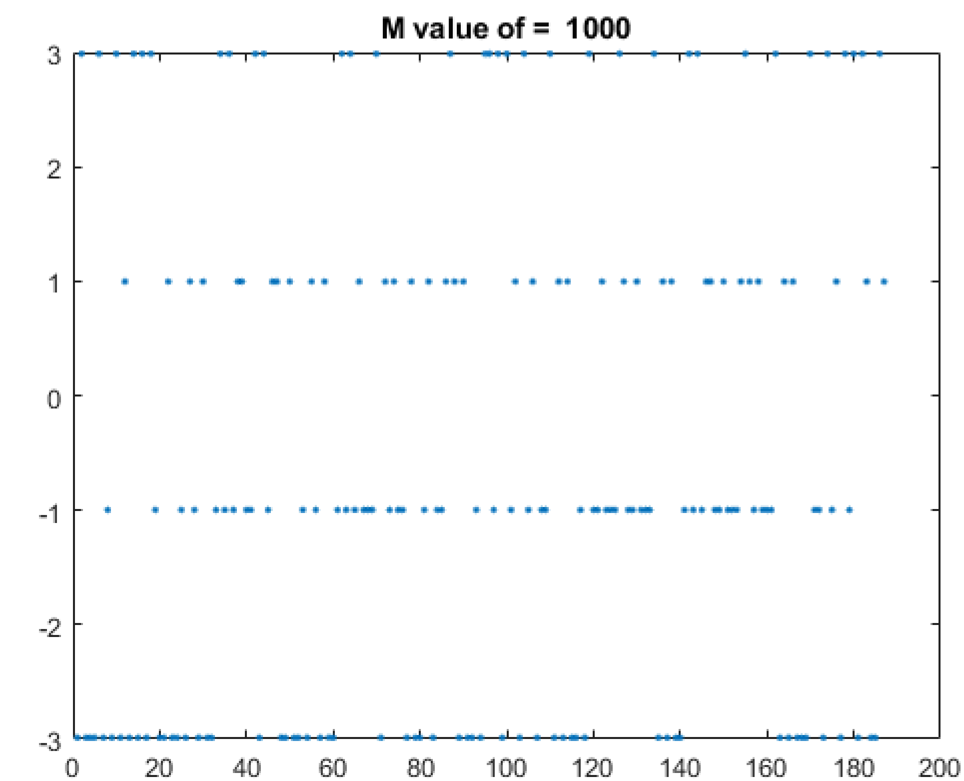
Exercise 9.2

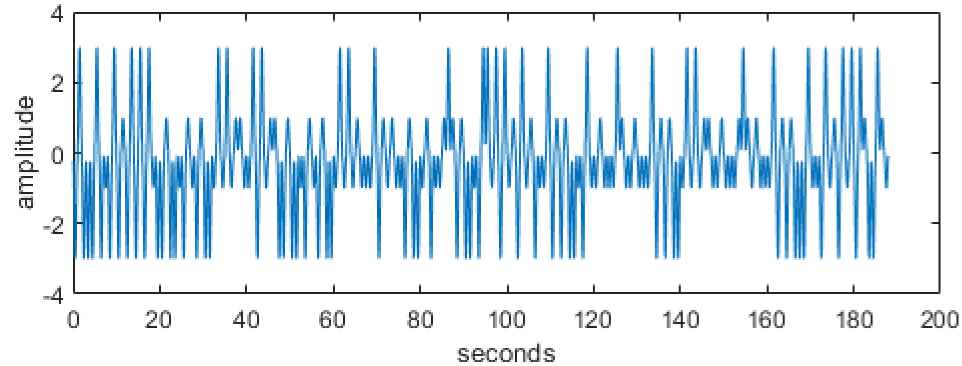
```
clc
%TRANSMITTER
% encode text string as T-spaced 4-PAM sequence
str='01234 I wish I were an Oscar Meyer wiener 56789';
m=letters2pam(str); N=length(m); % 4-level signal of length N
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
M=[1000, 25, 10];                        % oversampling factor
for iterator = 1:3
    mup=zeros(1,N*M(iterator));         % Hamming pulse filter with
    mup(1:M(iterator):N*M(iterator))=m; % T/M-spaced impulse response
    p=hamming(M(iterator));             % blip pulse of width M
    x=filter(p,1,mup);            % convolve pulse shape with data
    figure, plotspec(x,1/M(iterator))    % baseband AM modulation
    title(['M value of =  ',num2str(M(iterator))])
    t=1/M(iterator):1/M(iterator):length(x)/M(iterator);% T/M-spaced
 time vector
    fc=20;                        % carrier frequency
    c=cos(2*pi*fc*t);             % carrier
    r=c.*x;                       % modulate message with carrier

    %RECEIVER
    % 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(iterator)),1,x3);
    % set delay to first symbol-sample and increment by M
    z=y(0.5*fl+M(iterator):M(iterator):N*M(iterator));   % downsample
 to symbol rate
    figure, plot([1:length(z)],z,'.') % plot soft decisions
    title(['M value of =  ',num2str(M(iterator))])
    % 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)
end


% DISCUSSION

% Oversampling works on the idea of taking more samples then are
 needed,
% thus increasing the bandwidth. The overall goal for any system is
 still
% operating at the Nyquist rate. The oversampled signal is then passed
% through a low pass filter to eliminate components from the "mirror"
 side
% of it. A large value of M = 1000 would obviously be enough, but it
 can go
% as low 25. However, an oversample rate of 10 is not enough to
 increase
% the bandwidth to that appropriate level.


cvar =

   6.5588e-05


pererr =

     0


ans =

    'dropping last 3 PAM symbols'
```

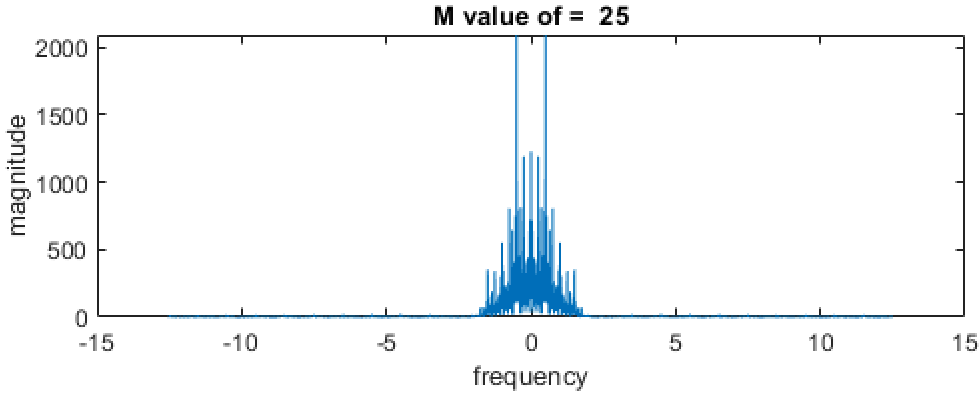
```
reconstructed_message =

    '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
M value of = 1000
magnitude
frequency

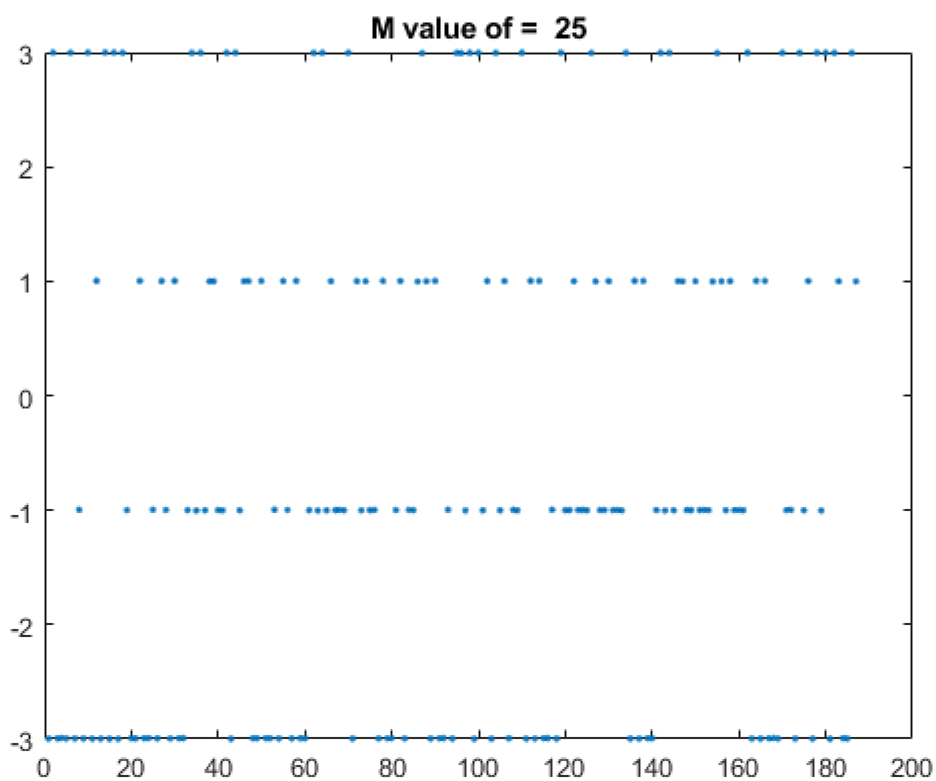

M value of = 1000



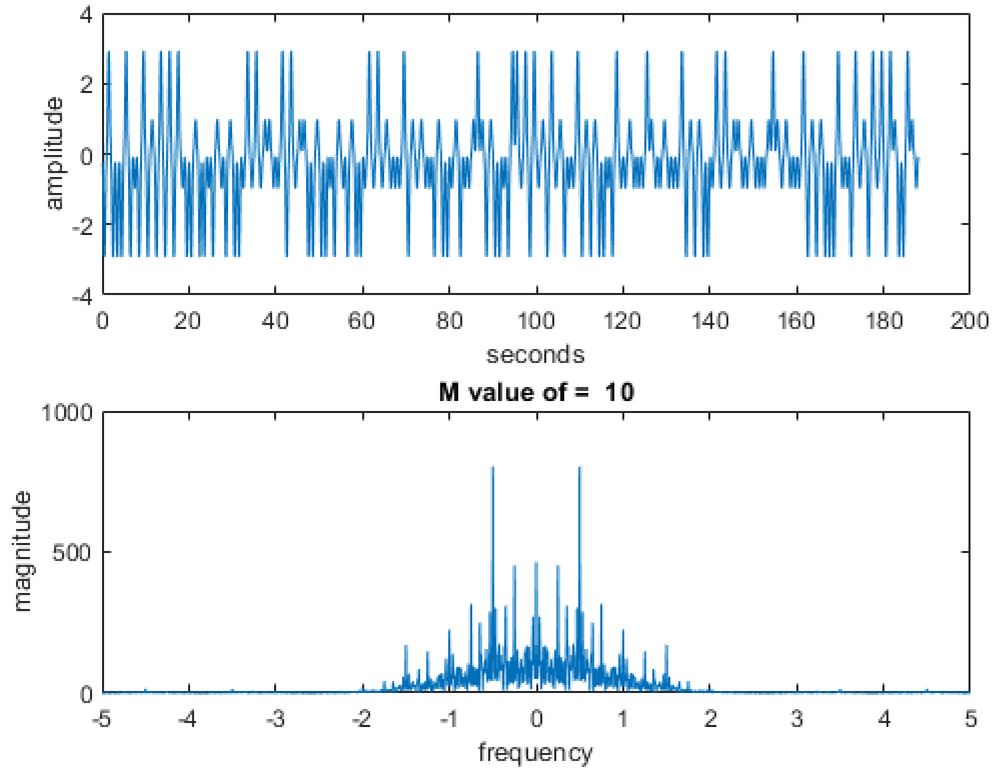
amplitude
seconds

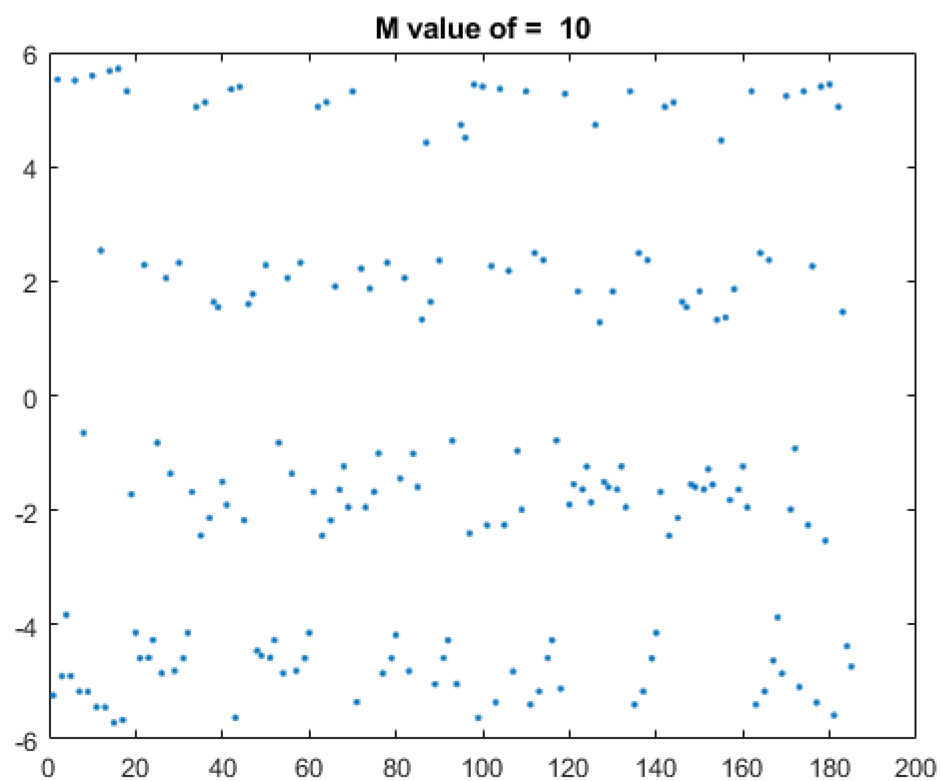


M value of = 25
magnitude
frequency



M value of = 25


amplitude
seconds
M value of = 10
magnitude
frequency

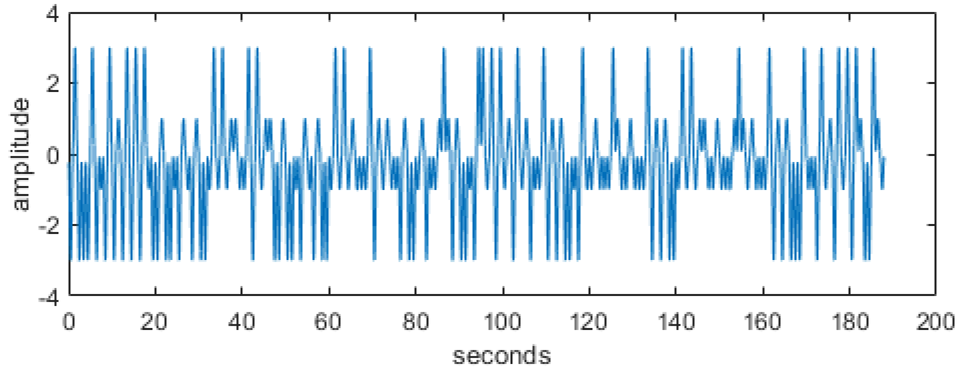

M value of = 10

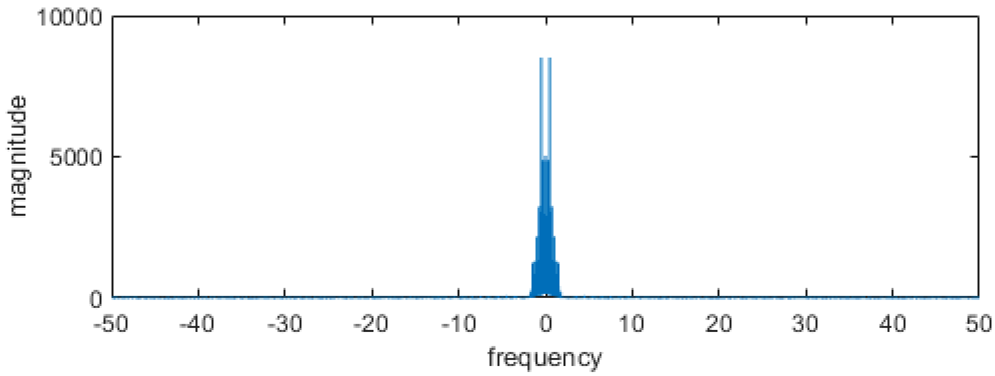
Exercise 9.3 LPF REMOVED

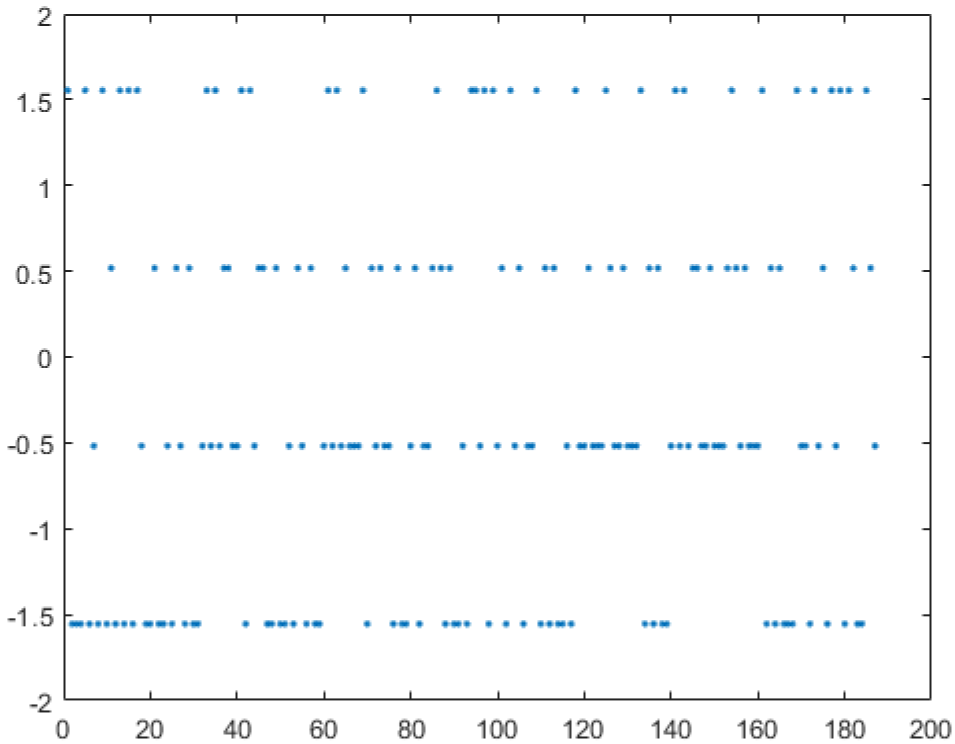
```
%TRANSMITTER
% encode text string as T-spaced 4-PAM sequence
str='01234 I wish I were an Oscar Meyer wiener 56789';
m=letters2pam(str); N=length(m); % 4-level signal of length N
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
M=100;                        % 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=20;                        % carrier frequency
c=cos(2*pi*fc*t);             % carrier
r=c.*x;                       % modulate message with carrier

%RECEIVER
% 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
% set delay to first symbol-sample and increment by M
z=x2(0.5*fl+M:M:N*M);           % downsample to symbol rate
figure, 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)

fprintf('This is the reconstructed when the LPF has been removed');

% DISCUSSION REGARDING THE LPF
% Without the LPF because of oversampling there are a lot more
 components
% in our signal that we do not need. For the new bandwidth to work
 properly
% the oversampled signal has to be filterd first to remove the
 "mirrored"
% components of the origanl signal.


cvar =

    0.2683
```

```
pererr =

   82.3529


ans =

    'dropping last 3 PAM symbols'


reconstructed_message =

    '######e##¥#¥#e#######©#i#####e#¥####¥#©######¥'
```

This is the reconstructed when the LPF has been removed





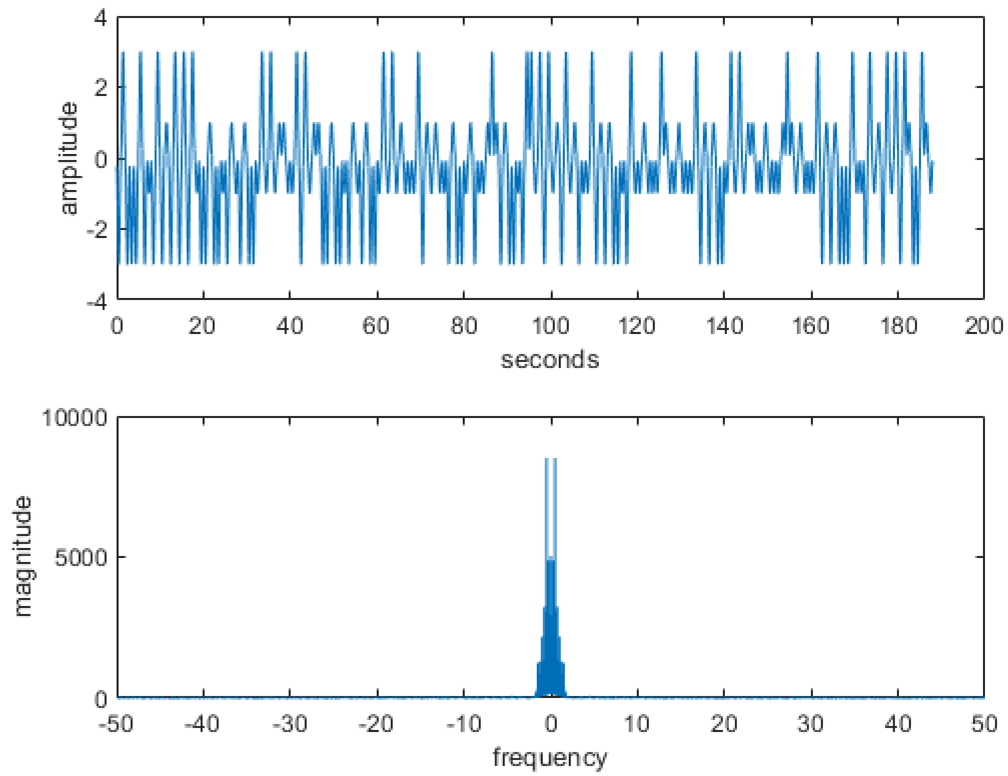


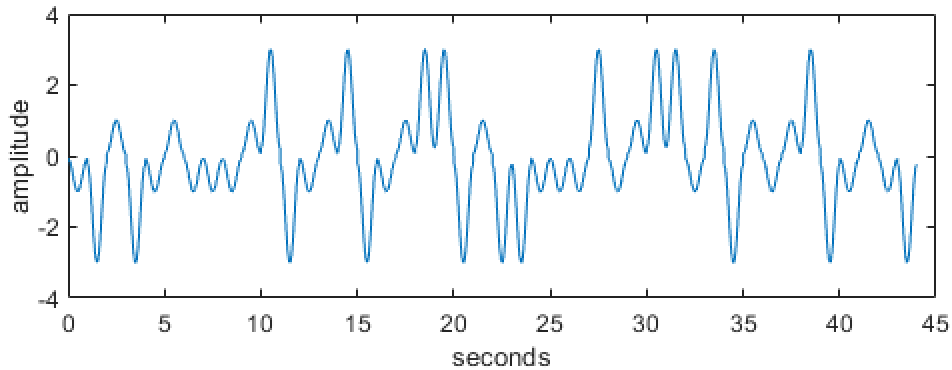
Exercise 9.3 Adding another user

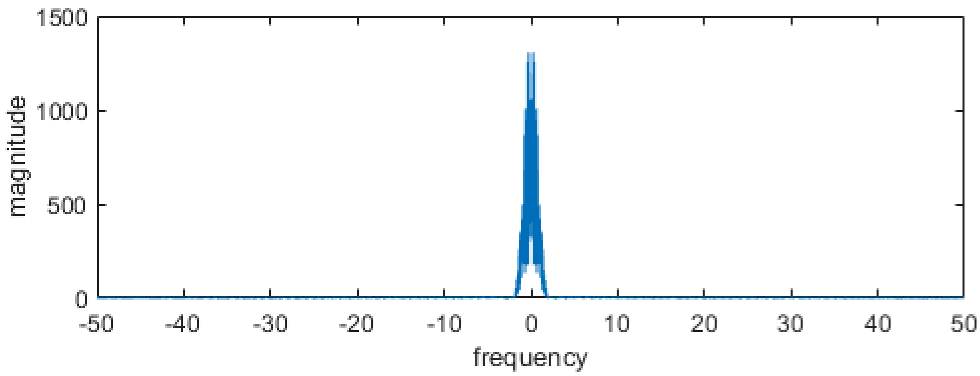
```
%TRANSMITTER
% encode text string as T-spaced 4-PAM sequence
str='01234 I wish I were an Oscar Meyer wiener 56789';
m=letters2pam(str); N=length(m); % 4-level signal of length N
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
M=100;                         % 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


%TRANSMITTER
% encode text string as T-spaced 4-PAM sequence
new_str='Hello World';
new_m=letters2pam(new_str); new_N=length(new_m); % 4-level signal of
 length N
```

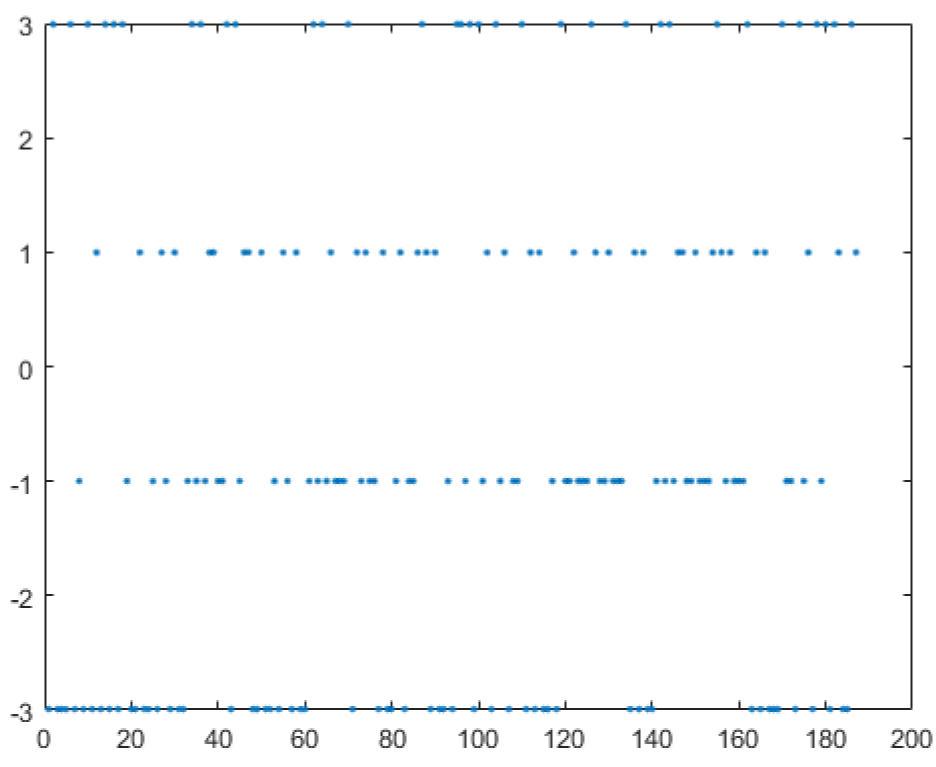
```
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
new_M=100;                        % oversampling factor
new_mup=zeros(1,new_N*new_M);           % Hamming pulse filter with
new_mup(1:new_M:new_N*new_M)=new_m;            % T/M-spaced impulse
 response
new_p=hamming(new_M);                 % blip pulse of width M
new_x=filter(new_p,1,new_mup);          % convolve pulse shape with
 data
figure, plotspec(new_x,1/new_M)    % baseband AM modulation
new_t=1/new_M:1/new_M:length(new_x)/new_M;       % T/M-spaced time
 vector
new_fc=30;                        % carrier frequency
new_c=cos(2*pi*new_fc*new_t);           % carrier
new_r=new_c.*new_x;                     % modulate message with
 carrier

%RECEIVER
% am demodulation of received signal sequence r
c2=cos(2*pi*fc*t);             % synchronized cosine for mixing
x2=r.*c2;                      % demod received signal
new_c2 = cos(2*pi*new_fc*new_t);
new_x2 = new_r.*new_c2;
% Combining signals
sx = size(x2);
sy = size(new_x2);
max_a = max(sx(1), sy(1));
x2 = [[x2;zeros(abs([max_a 0]-sx))],...
    [new_x2;zeros(abs([max_a,0]-sy))]];
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
% 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)


cvar =

   2.9259e-05


pererr =
```

```
     0


ans =

    'dropping last 3 PAM symbols'


reconstructed_message =

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




amplitude
seconds

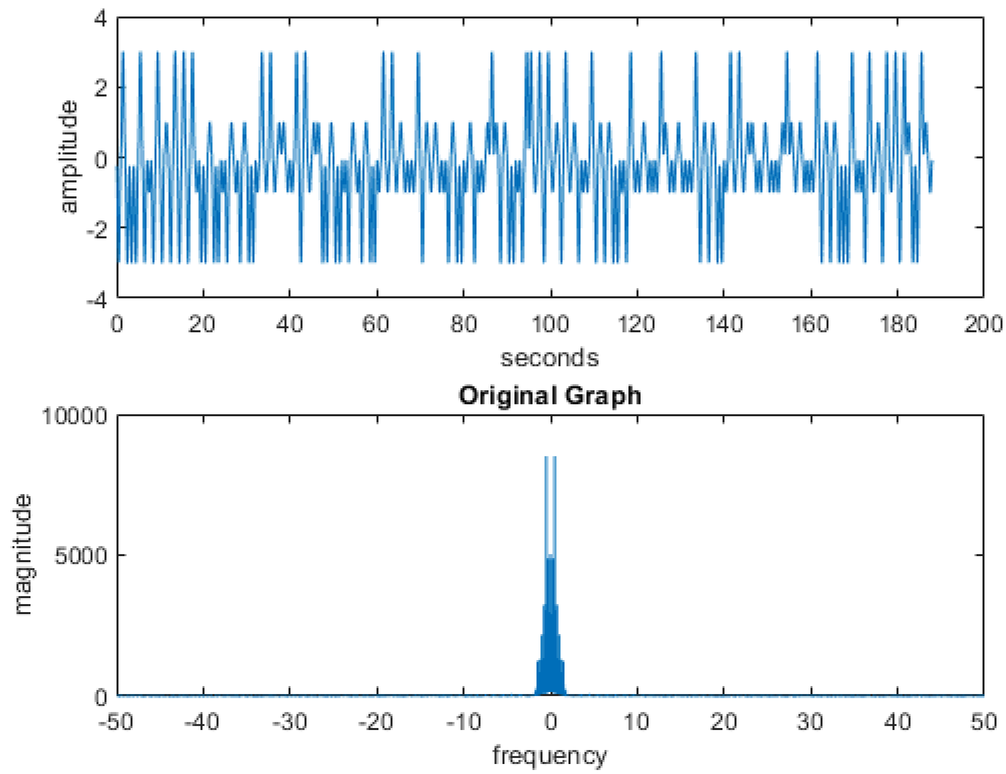

magnitude
frequency

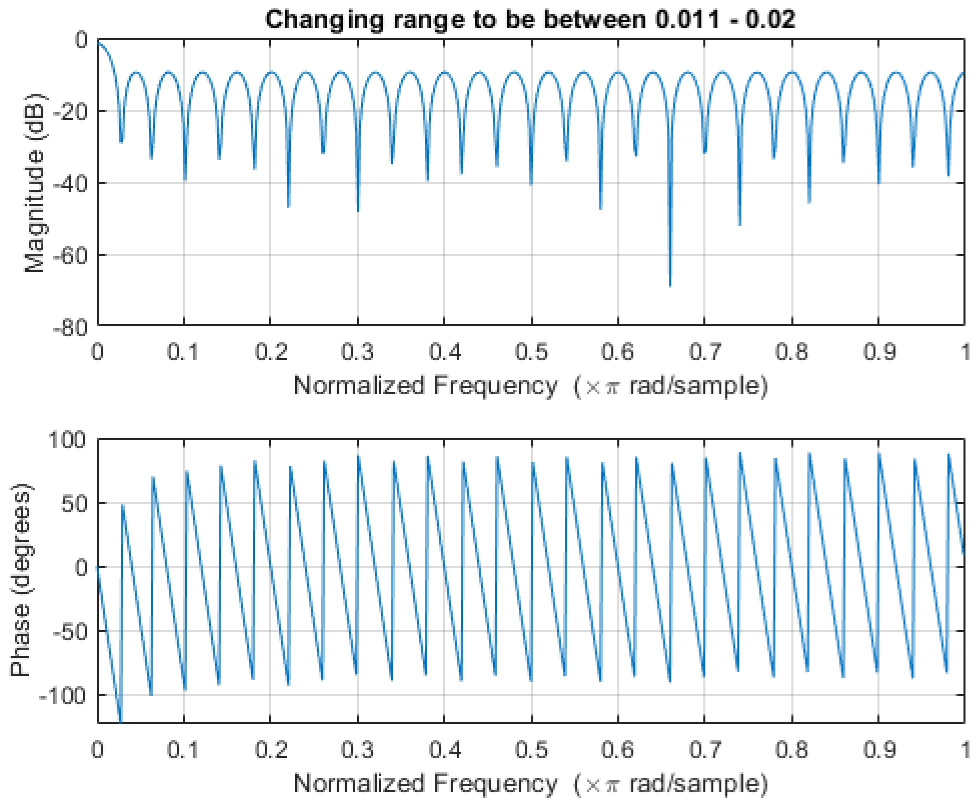


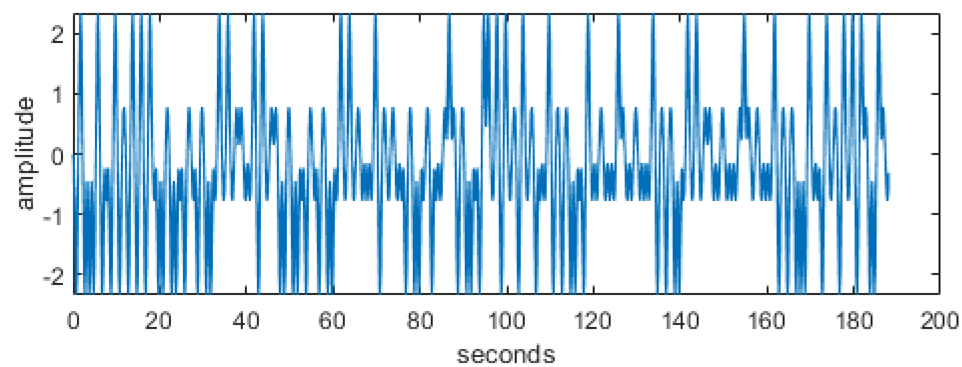
Exercise 9.4

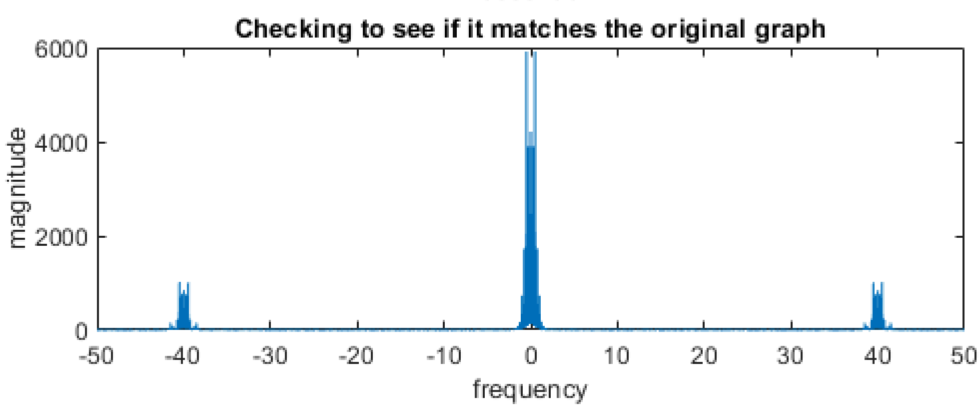
```
%TRANSMITTER
% encode text string as T-spaced 4-PAM sequence
str='01234 I wish I were an Oscar Meyer wiener 56789';
m=letters2pam(str); N=length(m); % 4-level signal of length N
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
M=100;                        % 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
title('Original Graph')
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

%RECEIVER
% 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 0.011 0.02 1];      % LPF parameters
damps=[1 1 0 0 ];
b=firpm(fl,fbe,damps);         % create LPF impulse response
figure
freqz(b)
title('Changing range to be between 0.011 - 0.02')
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
figure, plotspec(x3, 1/M)
title('Checking to see if it matches the original graph')
% 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
% It appears to be the the lowest the cutoff frequency can be is
 around the
% range of 0.011 in the normalized scale which is around 0.55 Hz. The
% highest it could be would be the Nyquist rate of 50.
```

```
cvar =

    0.4789


pererr =

     0


ans =

    'dropping last 3 PAM symbols'


reconstructed_message =

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




Changing range to be between 0.011 - 0.02
Magnitude (dB)
0
-20
-40
-60
-80
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency ($\times\pi$ rad/sample)
Phase (degrees)
100
50
0
-50
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency ($\times\pi$ rad/sample)


amplitude
2
1
0
-1
-2
0
20
40
60
80
100
120
140
160
180
200
seconds

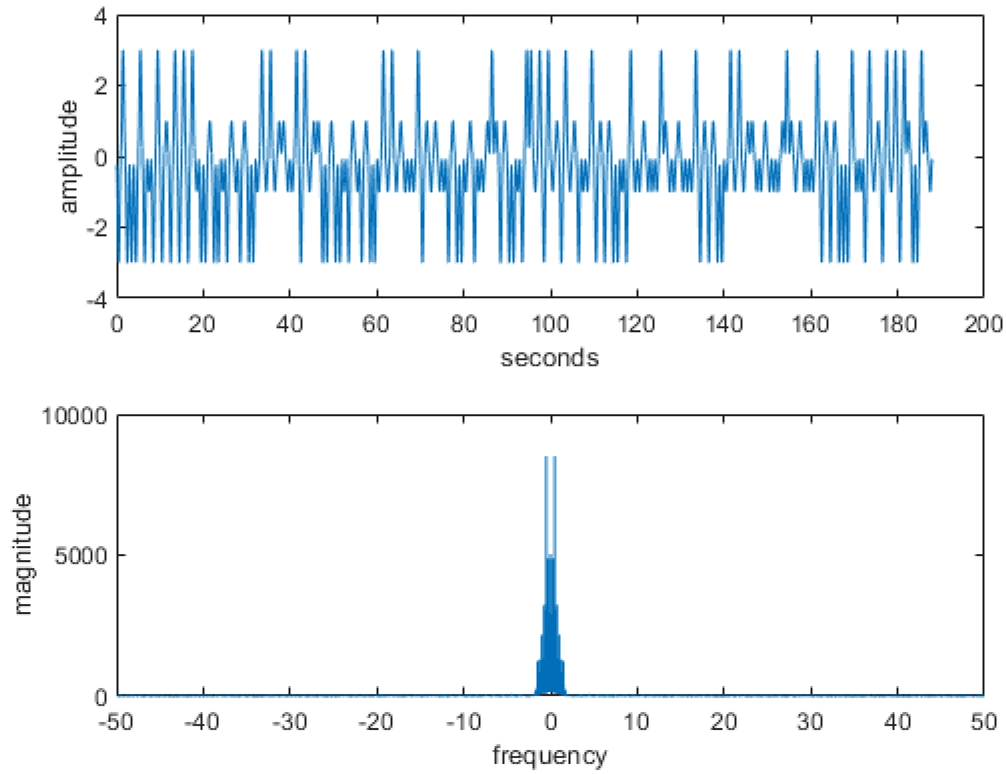

Checking to see if it matches the original graph
magnitude
6000
4000
2000
0
-50
-40
-30
-20
-10
0
10
20
30
40
50
frequency

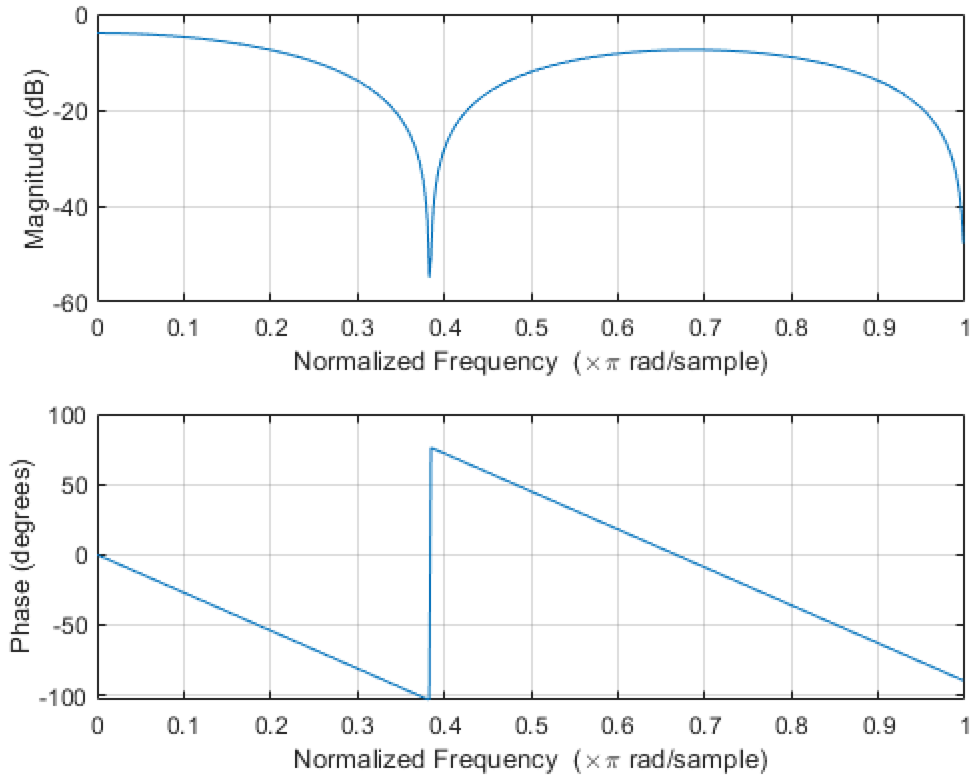
Exercise 9.5

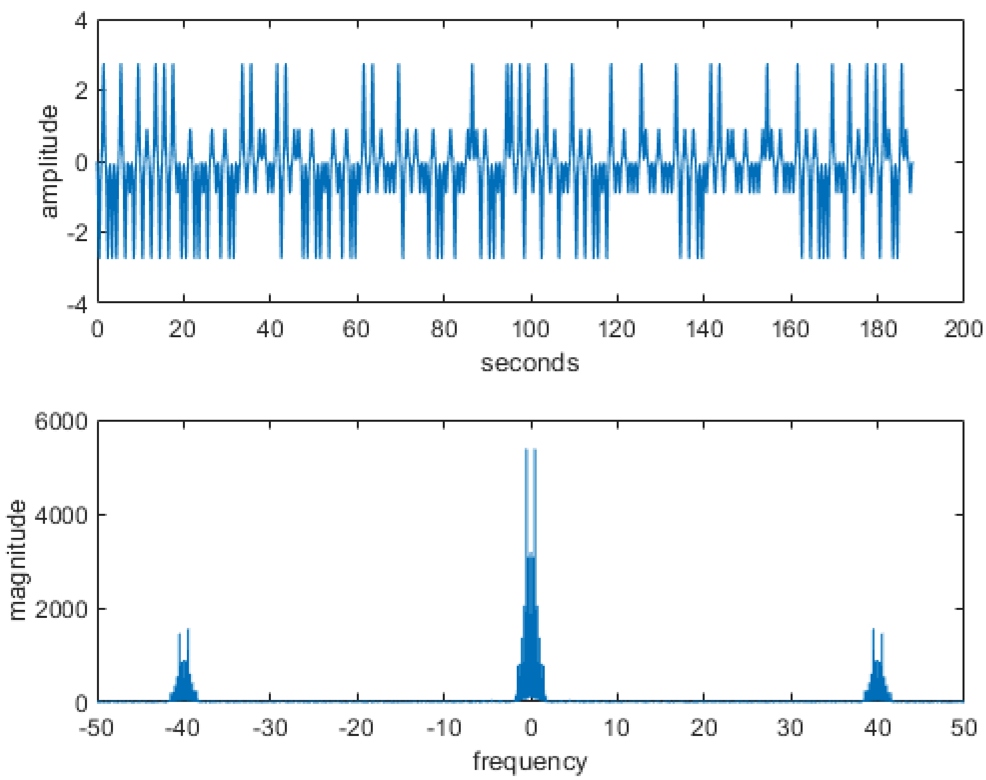
```
close all
%TRANSMITTER
% encode text string as T-spaced 4-PAM sequence
str='01234 I wish I were an Oscar Meyer wiener 56789';
m=letters2pam(str); N=length(m); % 4-level signal of length N
% zero pad T-spaced symbol sequence to create upsampled
% T/M-spaced sequence of scaled T-spaced pulses (T=1)
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=20;                        % carrier frequency
c=cos(2*pi*fc*t);             % carrier
r=c.*x;                       % modulate message with carrier

%RECEIVER
% am demodulation of received signal sequence r
c2=cos(2*pi*fc*t);             % synchronized cosine for mixing
x2=r.*c2;                      % demod received signal
fl=3; fbe=[0 0.1 0.2 1];      % LPF parameters
damps=[1 1 0 0 ];
b=firpm(fl,fbe,damps);         % create LPF impulse response
figure(3)
freqz(b)
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
figure, plotspec(x3, 1/M)
% 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
% The smallest you can make the filter is 4, once it reaches 3 the
 output
% can no longer be properly decoded
```

Warning: Integer operands are required for colon operator when used as index.

```
cvar =

    0.4597


pererr =

   48.6631


ans =

    'dropping last 3 PAM symbols'


reconstructed_message =

    'eeffeeYefifieYefefeeejeZffefeYeiefefiejefeeffi'
```




Magnitude (dB)
Normalized Frequency (×π rad/sample)
Phase (degrees)
Normalized Frequency (×π rad/sample)


amplitude
seconds
magnitude
frequency

QPSK

```
clear all
clc
%TRANSMITTER
% encode text string as T-spaced 4-PAM sequence
str='01234 I wish I were an Oscar Meyer wiener 56789';
% NEW FUNCTION USED TO CONVERT TO QPSK
m=QPSK_converter(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=20;                          % carrier frequency
c=cos(2*pi*fc*t);               % carrier
r=c.*x;                         % modulate message with carrier

%RECEIVER
% 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
figure(3)
freqz(b)
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
figure, plotspec(x3, 1/M)
% decision device and symbol matching performance assessment
mprime=quantalph(z,[-1-1i,-1+1i,1-1i,1+1i])'; % 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

% NEW FUNCTION USED TO CONVERT BACK INTO A BINARY STRING
reconstructed_message=QPSK_2_ascii(mprime)
fileID = fopen('decoded_message.txt','w');
fprintf(fileID,reconstructed_message);
fclose(fileID);
disp('Use python script to convert back into ascii')
```

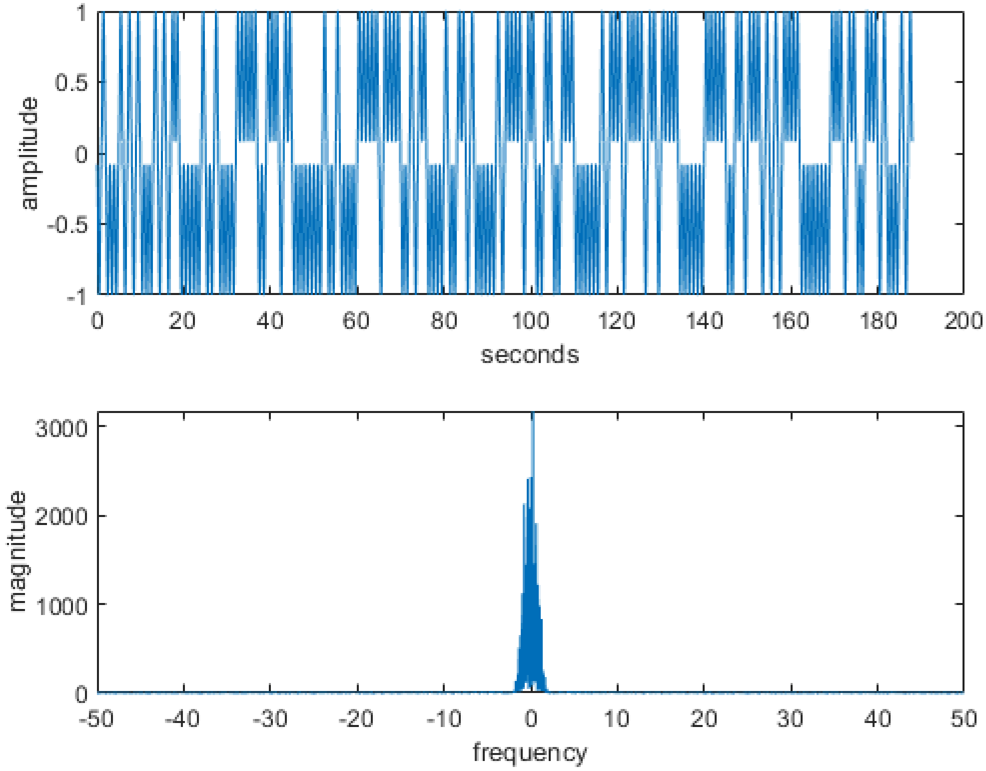
```
ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array
```

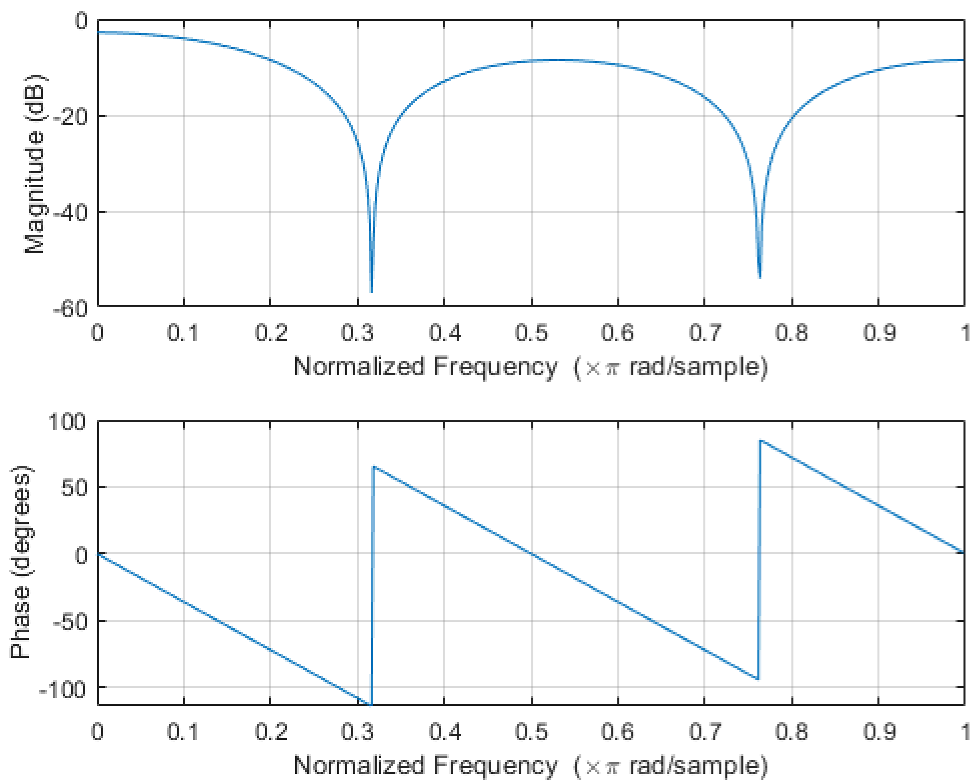
```
   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =
```

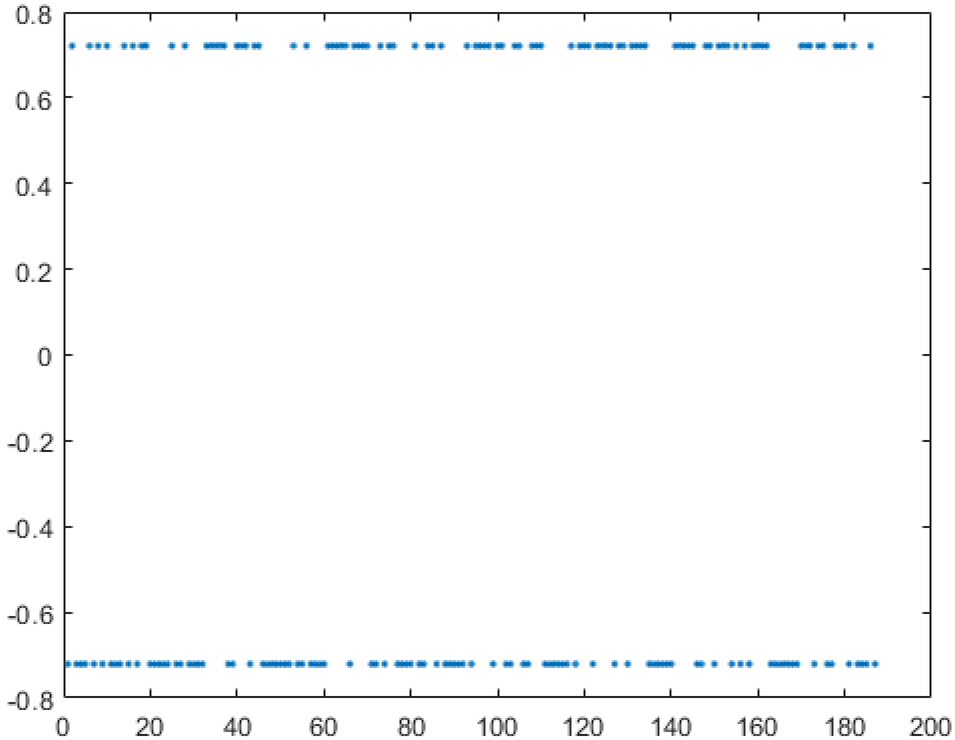
```
  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1
```

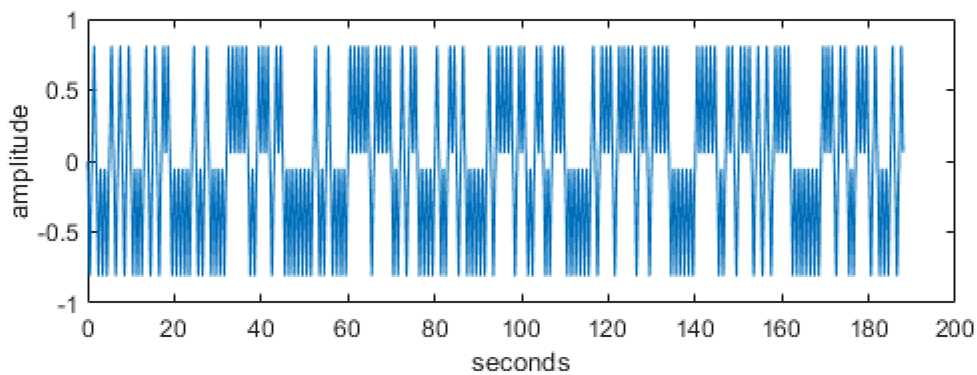
```
ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1

ans =

  1×2 logical array

   1   1
```

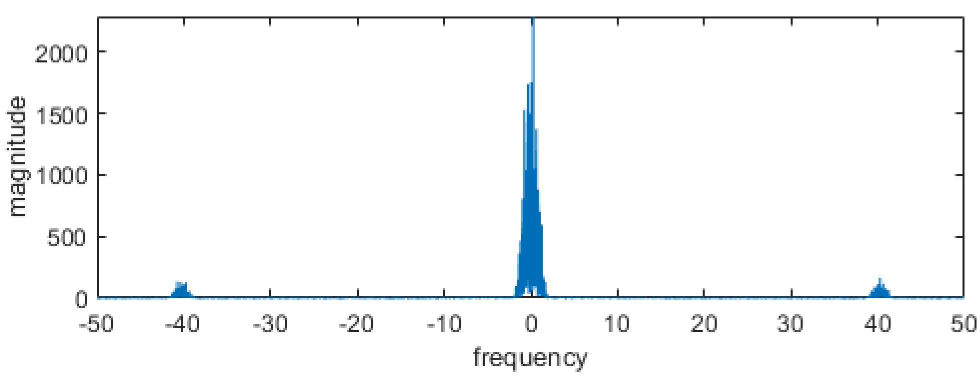
```
ans =

  1×2 logical array

   1   1


ans =

  1×2 logical array

   1   1

Warning: Imaginary parts of complex X and/or Y arguments ignored
Warning: Imaginary parts of complex X and/or Y arguments ignored
Warning: Imaginary parts of complex X and/or Y arguments ignored

cvar =

    0.1557


pererr =

     0


reconstructed_message =


 '00110000001100010011001000110011001101000010000001001001001000000111011101101001

Use python script to convert back into ascii
```


amplitude
seconds
magnitude
frequency


Magnitude (dB)
Normalized Frequency (×π rad/sample)
Phase (degrees)
Normalized Frequency (×π rad/sample)


0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
20
40
60
80
100
120
140
160
180
200


amplitude
seconds


magnitude
frequency

Published with MATLAB® R2019b

```python

import binascii
if __name__ == '__main__':
    f = open("decoded_message.txt", "r")
    str = f.read()
    f.close()
    counter = 1;
    binary_int = int(str, 2)
    decoded_message =  binascii.unhexlify('%x' % binary_int)
    print(decoded_message)

    # USE FOR TESTING ONLY

    # for x in str:
    #     print(x, end='')
    #     if(counter %8 == 0 and counter !=0):
    #         print(" ", end='')
    #     counter = counter +1

```

Python Results