Homework #4 Solutions

Pseudo-random binary sequences are also known as pseudo-noise (PN) sequences because the sequences resemble noise. PN sequences appear to be random but instead have structure. Moreover, PN sequences are periodic whereas noise is aperiodic (i.e. the period is infinitely long). A PN sequence can look random when observed for fewer samples than the period.

PN sequences are widely used to

- generate test, measurement and calibration signals
- generate training signals in communication systems (homework assignments 4-7)
- scramble and descramble data in communication systems (lab 4)
- generate additive dither in data converters (lecture 10)

4.1.Spectral Analysis of a Random Signal

(a) Johnson, Sethares & Klein, *Software Receiver Design*, exercise B.2 on page 414. Matlab's `randn` function is designed so that the mean is always (approximately) zero and the variance is (approximately) unity. Consider a signal defined by *w* = *a* `randn` + *b*; that is, the output of `randn` is scaled and offset. What are the mean and variance of *w*? Hint: use (B.1) and (B.2). What values must *a* and *b* have to create a signal that has mean 1.0 and variance 5.0?

Solution: Let x=`randn`. Matlab generates a value for x from a zero-mean normal random variable with unit variance. (Normal distribution is the Gaussian distribution.) Using statistical properties of random variables and assuming that *a* and *b* are constants, we take the expectation with respect to random variable *x* of the random variable *w* as follows:

1. $E[w] = E[ax+b] = aE[x]+b$. Using formula (B.1), we obtain the mean

$$m_W = \frac{1}{N}\sum_{k=1}^{N} w[k] = \frac{1}{N}\sum_{k=1}^{N}(ax[k]+b) = \frac{a}{N}\sum_{k=1}^{N} x[k] + \frac{N}{N}b = am_X + b.$$

2. $\text{var}[w] = \text{var}[ax+b] = a^2\,\text{var}[x]$. Using formula (B.2), we obtain the variance

$$v_W = \frac{1}{N}\sum_{k=1}^{N}(w[k]-m_W)^2 = \frac{1}{N}\sum_{k=1}^{N}(ax[k]+b-am_X-b)^2 = \frac{a^2}{N}\sum_{k=1}^{N}(x[k]-m_X)^2 = a^2 v_X$$

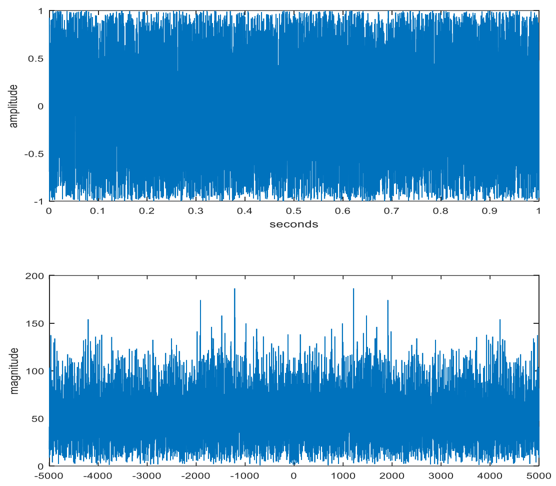
To create a signal of mean 1 and variance 5, we use the above relations. Since the mean of *x* is zero, the mean of *w* is *b*. Thus *b*=1. Since the variance of *x* is 1, the variance of *w* is

$$|a| = \sqrt{\frac{v_W}{v_X}} = \sqrt{5}.$$

(b) Johnson, Sethares & Klein, exercise 3.4, on page 43. Please submit the plots with your homework solution.

(b) subpart (a) Use `rand` to create a signal that is uniformly distributed on [−1, 1]. Find the spectrum of the signal by mimicking the code in `specnoise.m`.

Solution: To obtain a random variable uniformly distributed on [-1,1] from a random variable *x* uniformly distributed on [0,1], we would need to multiply *x* by 2 to expand its range from [0,1] to [0,2] and subtract 1 to make the range [-1,1]. The above procedure can be applied because adding a constant to, or multiplying a constant with, a uniformly-distributed random variable produces another uniformly-distributed random variable. The same results can be obtained by using the relations described in the previous part.



The following adaptation of the `specnoise.m` file produced the plot to the right.

```
% specnoise.m plot the spectrum of a noise signal
time=1;                         % length of time
Ts=1/10000;                     % time interval between samples
x=2*rand(1,time/Ts)-1;          % Ts points of noise for time seconds
plotspec(x,Ts)                  % call plotspec to draw spectrum
```

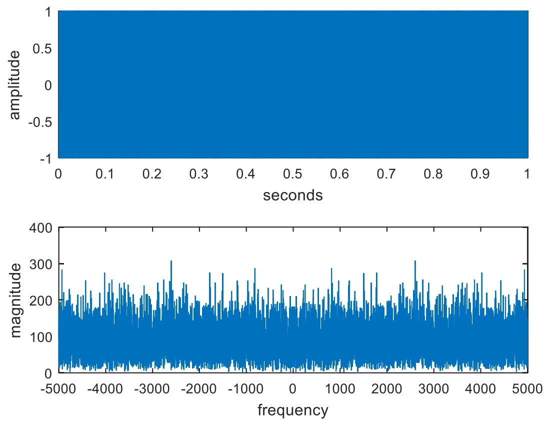
(b) subpart (b). Use `rand` and the `sign` function to create a signal that is +1 with probability ½ and −1 with probability ½. Find the spectrum of the signal.

Solution: To generate a random variable that is +1 with probability ½ and -1 with probability ½, we need two events each with probability ½, and assign to each event either 1 or -1. We can use the random variable in part (a) which is positive with probability of ½ and negative with probability of ½. If it is positive we choose +1; otherwise, we choose -1. See code and plot next.

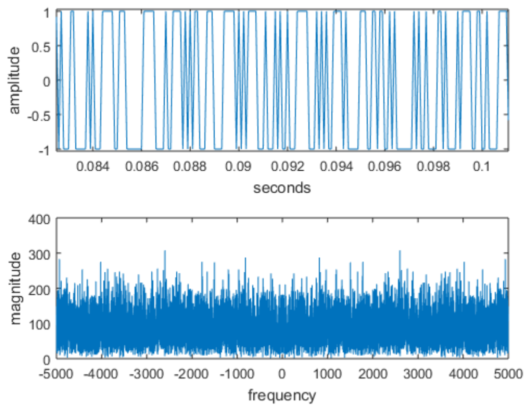
```
% specnoise.m plot the spectrum of a noise signal
time=1;                      % length of time
Ts=1/10000;                  % time interval between samples
w=2*rand(1,time/Ts)-1;       % Ts points of noise for time seconds
x=sign(w);
plotspec(x,Ts)               % call plotspec to draw spectrum
```

The `rand` function generates 0.5 with a very small probability (1 in 2^{64} since the result is a 64-bit floating-point number). In that case, *x* will be a zero since `sign`(0) = 0. This could be overcome by checking whether a 0 occurred in the vector w and replacing it arbitrarily with 1.

Plot



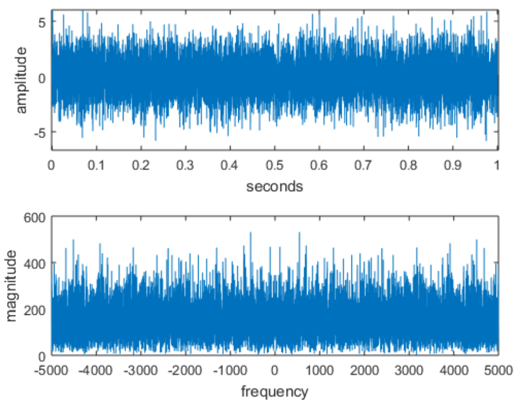
Zoomed-in on time domain plot



(b) subpart (c) Use `randn` to create a signal that is normally distributed with mean 0 and variance 3. Find the spectrum of the signal.

Solution: To generate a random variable that is normally distributed with mean 0 and variance 3, we use the previous equations derived in part (a). Since mean remains zero, we set *b* to zero and *a* to sqrt(3) to obtain a variance of 3. We then use the `randn` command to generate our signal.

```
% specnoise.m plot the spectrum of a noise signal
time=1;                       % length of time
Ts=1/10000;                   % time interval between samples
x=sqrt(3)*randn(1,time/Ts);  % Ts points of noise for time seconds
plotspec(x,Ts)                % call plotspec to draw spectrum
```



(c) Plot the spectrum of a pseudo-noise (PN) sequence of length 31 of +1 and -1 amplitudes; i.e., a bit of value ‘1’ maps to +1 and a bit of value ‘0’ maps to -1. For the spectrum plots, please plot both the discrete Fourier transform (e.g. using the `fft` command) and the discrete-time Fourier transform (e.g. using the `freqz` command). You can find a length 31 PN sequence in Appendix L of the course reader, or you could generate one in Matlab, or you could use the sequence that you generated in lab 4. Are there any frequencies at which the discrete Fourier transform or discrete-time Fourier transform is exactly zero?

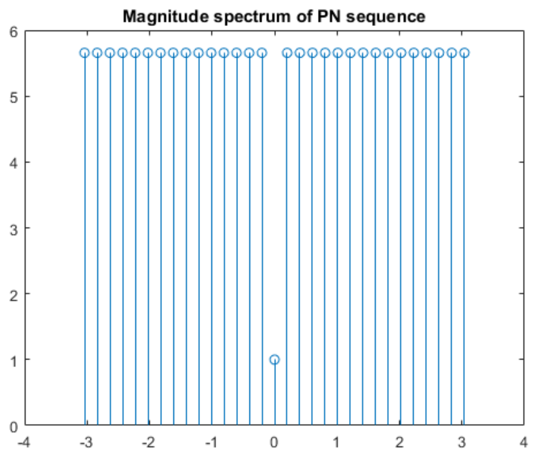
Solution:

```
%% Here's the length 31 pseudo-noise (PN) sequence from table on page L-6 in
%% the course reader.  The bit sequence is the "Output Bit" column, which
%% repeats on the last (32nd) entry.  We map a bit of value '1' to amplitude
%% 1 and a bit of value '0' to amplitude -1.
pnseq = [1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 -1 1 1 -1 -1 -1 1 1
1];

%% Given a sequence of length N, the fast Fourier transform (FFT) uniformly
%% samples the fundamental period of the of the discrete-time Fourier domain.
%% The fundamental period is 2 pi, and hence, each FFT coefficient corresponds
%% to frequency bin size of 2 pi / N.
binsize = 2*pi/length(pnseq);

%% One FFT coefficient corresponds to discrete-time frequency of zero rad/sample.
w = -pi + binsize/2 : binsize : pi - binsize/2;

%% Plot the magnitude spectrum of the PN sequence
stem(w, abs(fftshift(fft(pnseq))));
title(‘Magnitude spectrum of PN sequence’)
```

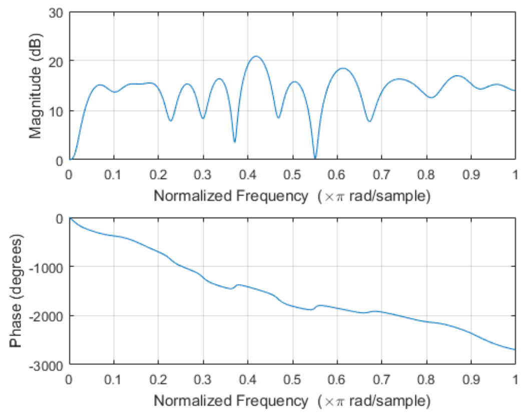


The pseudo-noise sequence is periodic with a period of 31 samples. When applying the fast Fourier transform of length 31, the fast Fourier transform assumes that the time-domain signal is periodic with period of 31 samples. The fast Fourier transform of the pseudo-noise sequence does not go to zero at any point. The DC value is 1.0 and other magnitude values are 5.6569. The FFT magnitude spectrum is flat except at DC.

We can also plot the discrete-time Fourier transform using MATLAB, which is next.

```
pnseq = [1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 -1 1 1 -1 -1 -1 1 1
1];
freqz(pnseq);
```

As shown below, all frequencies are present in the pseudo-noise signal. Magnitude spectrum varies in value from 0 dB to 21 dB, which is between 1 and 11.2 in linear units.



The fact that a PN sequence contains all frequencies makes PN sequences *fairly* robust to linear time-invariant (LTI) distortion and additive noise. As a consequence, PN sequences are useful in characterizing unknown channels, such as communication channels:

- Estimate propagation delay (Δ) through the unknown system. The receiver correlates the received signal with the pseudo-noise sequence. The first peak would correspond to the estimate of delay. In practice, there can be some error in detecting the first peak due to noise.
- Estimate the overall gain of the unknown system given the estimate of the propagation delay

(Δ) through the unknown system. One can discard the first Δ samples of the received signal and then estimating the gain by dividing the remaining received samples by the PN sequence samples and averaging the results.

- Estimate impulse response $h[n]$ of an LTI model of an unknown system given the PN input sequence $x[n]$ and the received sequence $y[n]$. The receiver can estimate $h[n]$ by using either
    - Frequency domain methods $H(\omega) = Y(\omega) / X(\omega)$ and inverse transforming, or
    - Time domain methods $y[n] = h[n] * x[n]$ and backsolve for $h[n]$. Note that each value of $n = 0, 1, \ldots$ gives one equation and one unknown.

The frequency domain method is possible because the PN sequence $x[n]$ contains all discrete-time frequencies, and its Fourier transform $X(\omega)$ never goes to zero. The DC component is $1 / L$ where L is the length of the PN sequence, which decreases as L increases. For large values of L, a DC offset could be added to the PN sequence to prevent the DC component from getting close to zero.

Problem 4.2. Johnson, Sethares & Klein, *Software Receiver Design,* exercise 8.8, page 160. In the code for correx.m, define `header` to be

(a) A sequence of length 31 of all +1 entries. The Matlab command `ones` might be helpful.
(b) A maximal length pseudo-noise sequence of length 31 with entries +1 and -1.

Explain why there is a difference in detection performance between the two different headers.

Solution: For each header, the average signal power is 1. This is computed by adding the squares of the signal values and dividing by the number of samples. Average noise power is σ^2. By changing the value of σ, we change the noise power and hence the signal-to-noise ratio.

First, we run the specified headers (all ones and pseudo-noise) 10000 times.

a) For the header consisting of all ones:

```
totalCorrect = 0;
totalRuns = 10000;
for run = 1:totalRuns
  % Begin correx.m from Software Receiver Design book
  header = round(ones(1,31));   % header is a predefined string
  loc=30; r=25;           % place header in position loc
  data=[ sign(randn(1,loc-1)) header sign(randn(1,r)) ]; % generate signal
  sd=0.25; data=data+sd*randn(size(data)) ; % add noise
  y=xcorr(header, data) ; % do cross correlation
  [m, ind]=max(y);        % location of largest correlation
  headstart=length(data)-ind+1; % place where header is detected
  % End of code from Software Receiver Design book
  if ( loc == headstart )
    totalCorrect = totalCorrect + 1;
  end
end
totalCorrect / totalRuns
```

After 10,000 runs, probability of correctly detecting a header of all ones through correlation is:

- 47% of the time when average signal power is four times average noise power (`sd=0.5`)
- 33% of the time when the average signal power equals the average noise power (`sd=1`)
- 16% of the time when average signal power is one-fourth of average noise power (`sd=2`)

The reason for this low success rate is because a header of all ones is falsely identified when the bit before header in a sequence is ‘1’. If the header [1 1 … 1] appears in the following sequence

1 -1 1 1 1 -1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 [1 1 …1] 1 -1 1 -1 -1 -1 -1 1 1 1-1

receiver will falsely identify the header if the value before the marker is 1. This happens 50% of the time (assuming -1 and 1 are equally likely). This matches the detection rate in simulation.

b) For the PN header:

```
sd = 0.65;  % 0.1, 0.3, 0.5, 1, 2
totalCorrect = 0;
totalRuns = 10000;
for run = 1:totalRuns
  %% Begin correx.m from Software Receiver Design book
   header = round([1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 -1 1 1 -1 -1
-1 1 1 1]);
  loc=30; r=25;          % place header in position loc
  data=[ sign(randn(1,loc-1)) header sign(randn(1,r)) ]; % generate signal
  data=data+sd*randn(size(data)) ; % add noise
  y=xcorr(header, data) ; % do cross correlation
  [m, ind]=max(y);        % location of largest correlation
  headstart=length(data)-ind+1; % place where header is detected
  %% End of code from Software Receiver Design book
  if ( loc == headstart )
    totalCorrect = totalCorrect + 1;
  end
end
totalCorrect / totalRuns
```

After 10,000 runs, probability of correctly detecting a PN header through correlation is

- 100% of the time when average signal power is four times average noise power (`sd=0.5`)
- 98% of the time when the average signal power equals the average noise power (`sd=1`)
- 58% of the time when average signal power is one-fourth of average noise power (`sd=2`)

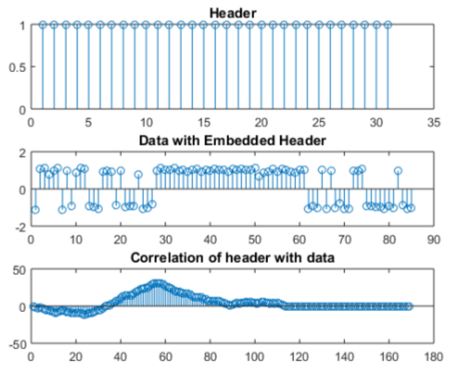
When there is no noise, the bit just before the header cannot fool the receiver, as a shift in the PN header will result in nearly zero correlation. Correlation will falsely identify the header when 31 consecutive random bits are equal to the PN header. Probability is 1 in 2^{31}, or $2^{-31} = 4.65 \times 10^{-10}$.

When we add noise, we obtain the detection rates below. Increasing the average noise power causes very little change in detection rate until the average noise power reaches 0.4 of the average signal power.

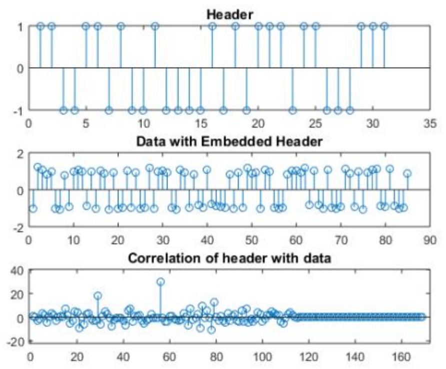
Standard deviation σ	**Header of all ones**	**Pseudo-noise (PN) header**
0.1	0.4976	1
0.3	0.5040	1
0.5	0.4687	1
1	0.3320	0.9848
2	0.1562	0.5868

Above, the detection rate for the PN header is 2-4x better than that of the header of all ones for a standard derivation less than 1, and about 3x or 4x better for a standard derivation greater than 1.

Header of all ones

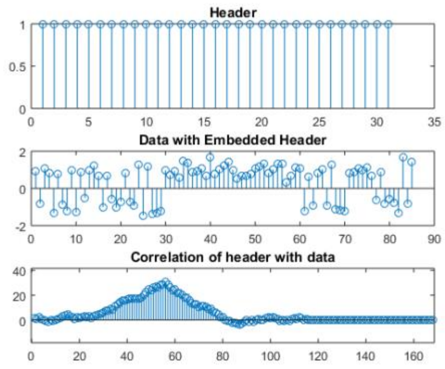


Header of PN sequence

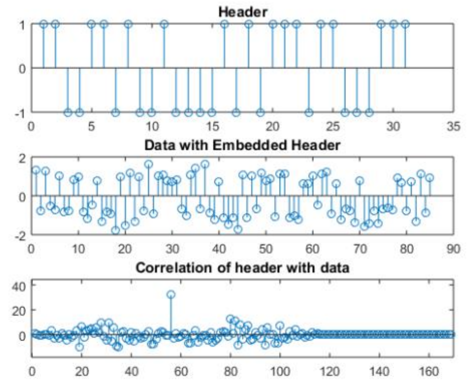


sd = 0.1

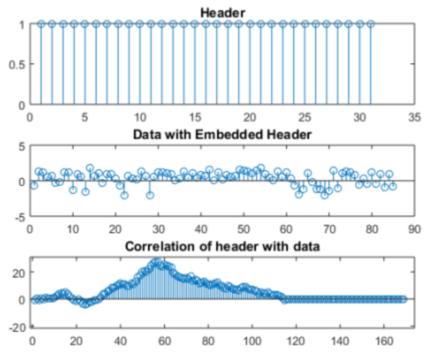
Header of all ones

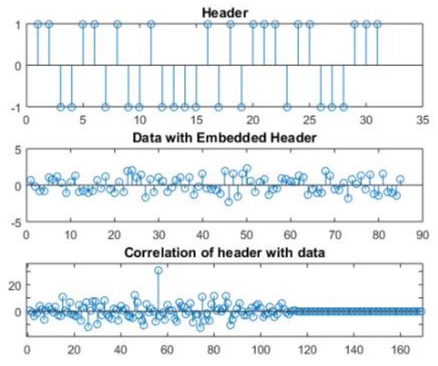


Header of PN sequence

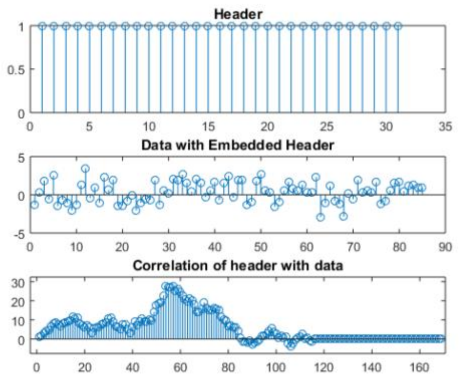


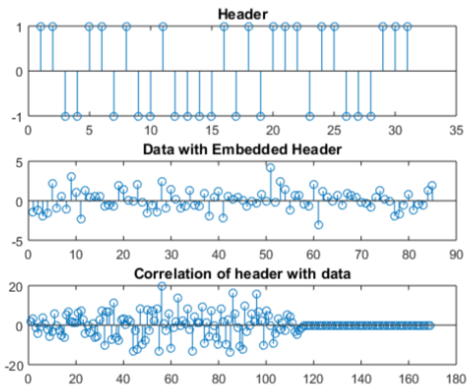
sd = 0.3



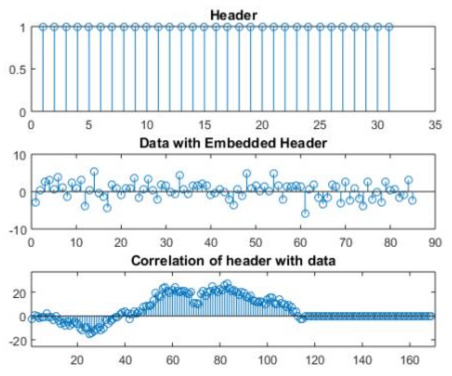


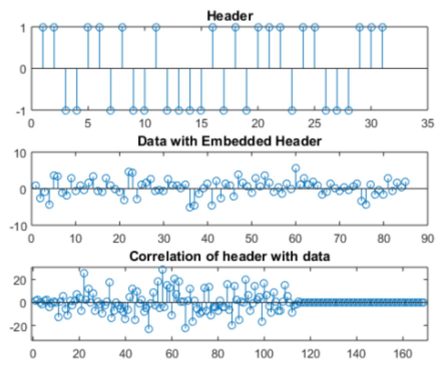
sd = 0.5





sd = 1





sd = 2

In general, we detect the peak much more clearly when we use a PN sequence as the header.

Problem 4.3: Johnson, Sethares & Klein, Exercise 8.14, page 164. Use the 4-PAM alphabet with symbol amplitudes -3, -1, +1 and +3. The marker sequence should have amplitudes of -3 and +3. Please plot the magnitude and phase response for each channel model.

Please use (a) *a* = 0.1 and *b* = 0.4, (b) *a* = 0.5 and *b* = 0.9, and (c) *a* = 1.2 and *b* = 0.4.

Use the 4-PAM alphabet with symbols ±1, ±3. Create a marker sequence, and embed it in a long sequence of random 4-PAM data. Check to make sure it is possible to correctly locate the markers. Then, add a channel with impulse response 1, 0, 0, *a*, 0, 0, 0, *b* to this 4-PAM example.

Solution: First we complete problem 8.17. The following code creates a header of all '3's and embeds it in a long sequence of 4-PAM symbol amplitudes (in the set {-3, -1, 1, 3}). In generating the 4-PAM symbol amplitudes, we modify the code to set p = -4 and q = 4 so that we generate uniform random numbers on the interval [-4, 4] instead of [-3, 3]. When using an interval of [-3, 3], the symbol amplitudes of 1 and -1 will be twice as likely as 3 or -3.

```
n = [-3 -1 1 3];
p = -4; q = 4;                     % general random numbers on interval [p,q]
totalCorrect = 0;
totalRuns = 10000;
for run = 1:totalRuns
  % Begin code from Software Receiver Design book
  header = round(3*ones(1,31));  % header is a predefined string of all '3's
  loc=310; r=25;            % place header in position loc
  signal1 = p + (q-p).*rand(1,loc+r);    % generate random numbers on [p,q]
  signal1 = quantalph(signal1,n);
  signal1 = signal1';
  data=[signal1(1:loc-1) header signal1(loc:end)]; % generate signal
  sd=0; data=data+sd*randn(size(data)) ; % add noise
  y=xcorr(header, data) ; % do cross correlation
  [m, ind]=max(y);          % location of largest correlation
  headstart=length(data)-ind+1; % place where header is detected
  % End of code from Software Receiver Design book
  if ( loc == headstart )
    totalCorrect = totalCorrect + 1;
  end
end
totalCorrect / totalRuns
```

Correct detection now occurs about 74% of the time. The results show better precision than those from the 2-PAM signal in Problem 4.2 because, in this case, the data can take two additional values, thereby reducing the probability of a falsely identified marker.

For the PN marker:

```
n = [-3 -1 1 3];
p = -4; q = 4;                      % general random numbers on interval [p,q]
totalCorrect = 0;
totalRuns = 10000;
for run = 1:totalRuns
  %% Begin code from Software Receiver Design book
  h = [1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 -1 1 1 -1 -1 -1 1 1 1];
  header = round(3*h);           % header is a predefined string in set {-3,3}
  loc=310; r=25;          % place header in position loc
  signal1 = p + (q-p).*rand(1,loc+r);    % generate random numbers on [p,q]
  signal1 = quantalph(signal1,n);
  signal1 = signal1';
  data=[signal1(1:loc-1) header signal1(loc:end)]; % generate signal
  sd=0; data=data+sd*randn(size(data)) ; % add noise
  y=xcorr(header, data) ; % do cross correlation
  [m, ind]=max(y);        % location of largest correlation
  headstart=length(data)-ind+1; % place where header is detected
  %% End of code from Software Receiver Design book
  if ( loc == headstart )
    totalCorrect = totalCorrect + 1;
  end
end
totalCorrect / totalRuns
```

The PN header has 31 samples. For a randomly generated sequence of 31 values, the probability of an error sample is 1 out of $2^{31} = 2.1475*10^9$. One would need to run 10^{11} sets of 31 randomly-generated sequences to have confidence to have a misdetection. The simulation has 310 random values before the header, which amounts to 310 - 31 = 279 sets of 31 values. Hence, to have confidence to see a misdetection, it is necessary to run the simulation with 10^9 trials.

(a) For a = 0.1 and b = 0.4, how does the channel change the likelihood that the correlation correctly locates the marker?

Solution: Add a communication channel modeled by a finite impulse response filter with impulse response [1, 0, 0, a, 0, 0, 0, b]. Disregard noise for now. Let a = 0.1 and b = 0.4:

```
close all; clear all; clc;
a = 0.1; % 0.1 0.5 1.2
b = 0.4; % 0.4 0.9 0.4
imp = [1, 0, 0, a, 0, 0, 0, b]; % impulse response of channel
figure; freqz(imp);
n = [-3 -1 1 3];
p = -4; q = 4;                      % general random numbers on interval [p,q]
totalCorrect = 0;
totalRuns = 10000;
for run = 1:totalRuns
  % Begin code from Software Receiver Design book
header = round(3*ones(1,31));  % header is a predefined string of all '3's
                               % rounded to make sure symbol amplitudes are +3
%  h = [1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 -1 1 1 -1 -1 -1 1 1 1];
%  header = round(3*h);    % header is a predefined string in set {-3,3}
                          % rounded to make sure symbol amplitudes -3,+3
  loc=310; r=25;          % place header in position loc
  signal1 = p + (q-p)*rand(1,loc+r);   % generate random numbers on [p, q]
  signal1 = quantalph(signal1,n);
  signal1 = signal1';
  data=[signal1(1:loc-1) header signal1(loc:end)]; % generate signal
  channel = filter(imp, 1, data); % add the channel
%   sd=0.5; channel=channel+sd*randn(size(channel)) ; % add noise, 0.25 0.5
```

```
  y=xcorr(header, channel) ; % do cross correlation
  [m, ind]=max(y);        % location of largest correlation
  headstart=length(channel)-ind+1; % place where header is detected
  % End of code from Software Receiver Design book
  if ( loc == headstart )
    totalCorrect = totalCorrect + 1;
  end
end
totalCorrect / totalRuns
```

Simulation results gave a probability of correct detection of 58.52% for a marker of all 3s and 100% for the PN sequence. If we add noise with a standard deviation of 0.25, the simulation results gave 58.01% and 100%, respectively.

(b) Answer the same question for *a*=0.5 and *b*=0.9.

Solution: With *a*=0.5 and *b*=0.9, and without noise, simulations gave 13.88% correct detection for a header of all 3s, and 76.41% for the PN header. If we add noise with standard deviation of 0.25, we get 13.58% correct detection for the header of all 3s, and 76.05% for the PN sequence.

(c) Answer the same question for *a*=1.2 and *b*=0.4.

Solution: With *a*=1.2 and *b*=0.4, and without noise, simulations gave 10.3% correct detection for a header of all 3s, and 0% for the PN header. If we alter the code to keep the second largest peak of the correlation instead of the largest peak,

```
  [m, ind]=max(y);        % location of largest correlation
  y(ind) = 0;
  [m, ind]=max(y);        % location of second largest correlation
```

then simulations gave 11% correct detection for a header of all 3s and 99.6% for the PN header. This is due to the convolution of the baseband transmission signal and the channel impulse response. Marker detection breaks down when the first non-zero value of the impulse response is not the largest in absolute value among the other non-zero values of the impulse response. The peak will be detected based on the highest impulse response value in absolute value. In this part, the impulse response values are 1, 0, 0, 1.2, 0, 0, 0, 0.4. The value at 1.2 will be the one selected. The second highest peak will be the one corresponding the first non-zero value of 1 because it has the second highest absolute value among the impulse response coefficients.

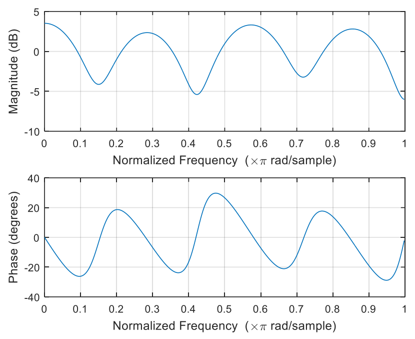
If we add noise with standard deviation of 0.5, we get 11.5% correct detection for the header of all 3s and 0% for a PN header. If we keep the second largest peak of the correlation instead of the largest, then we get 11.6% correct detection for a header of all 3s and 99.5% for a PN header.

An algorithm used in practice to determine the delay is to take the highest 3-4 peaks above a certain threshold, fit the peaks with a polynomial, and find the location of the peak of the polynomial. This method has a higher probability of correct detection and can determine delays that are real numbers, i.e. delays that are not integers. See Exercise 2.3(d) in [1] for an example. We had discussed polynomial fits to data on Lecture Slide 7-9 "Curve Fitting" [2].

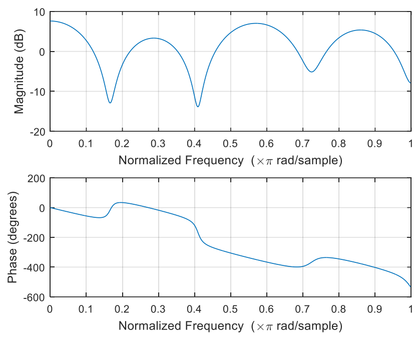
Other post-processing techniques to correct the location of the detected peak in (c) including equalization, symbol synchronization, frame synchronization, and constellation de-rotation.

The channel filters the transmitted sequence. A header of all 3s is lowpass, so a bandpass or narrower lowpass filter will result in significant loss of the marker sequence and hence detection success in the receiver. On the other hand, the PN sequence fills the entire spectrum.

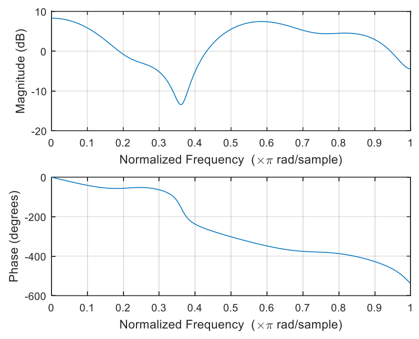
Plots of channel magnitude responses in (a), (b) and (c) are given below. The magnitude response of the second channel has more deviation from an all-pass response than that of the first, and hence, the second channel gives lower correct detection rates for the same noise power. The third channel has the most deviation and gives the lowest correction detection rates.



(a)



(b)



(c)

With the low performance in (c), simulations are repeated with a 1023-length PN sequence. The PN sequence was generated using the Communications Toolbox or it can be downloaded from:

http://users.ece.utexas.edu/~bevans/courses/realtime/homework/pn1023seq.mat

```
pn1023gen = commsrc.pn('GenPoly',         [10 7 0], ...
                       'InitialStates', [0 0 0 0 0 1 0 0 0 0], ...
                       'CurrentStates', [0 0 0 0 0 1 0 0 0 0], ...
                       'Mask',          [0 0 0 0 0 1 0 0 0 0], ...
                       'NumBitsOut',    1023);
pn1023seq = round(2 * generate(pn1023gen) - 1);
```

Probability of correct detection of the 1023-length PN sequence is 0% for 10,000 trials each case. If we keep the second highest peak, then the detection success becomes 100%.

Reference

[1] C. S. Burrus, J. H. McClellan, A. W. Oppenheim, T. W. Parks, R. W. Schafer and H. W. Schuessler, “Chapter 10: Applications”, *Computer-Based Exercises for Signal Processing Using Matlab*, 1994.
[2] B. L. Evans, “Lecture 7: Interpolation and Pulse Shaping”, EE 445S Real-Time Digital Signal Processing Lab, UT Austin.