Spring 2014 EE 445S Real-Time Digital Signal Processing Laboratory Prof. Evans

Homework #4 Solutions

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

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

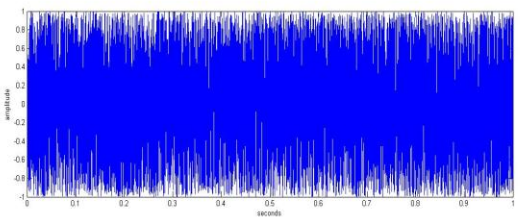
To create a signal of mean 1 and variance 5, we use the above relations. Since the mean of *x* is zero, the mean of *w* is *b*. Thus *b*=1. Since the variance of *x* is 1, the variance of *w* is

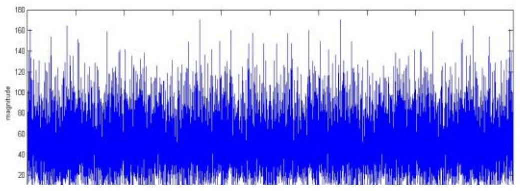
$$|a| = \sqrt{\frac{v_W}{v_X}} = \sqrt{5}.$$

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

(b).(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 since the distribution of a uniform random variable does not change under addition and multiplication of constants. 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.

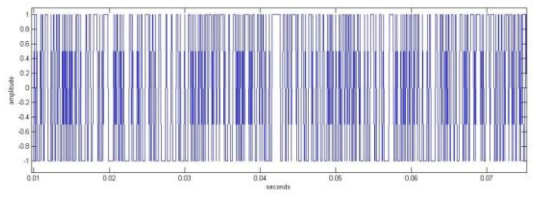
```
% specnoise.m plot the spectrum of a noise signal
time=1;                          % length of time
Ts=1/10000;                      % time interval between samples
x=2*rand(1,time/Ts)-1;   % Ts points of noise for time seconds
plotspec(x,Ts)                  % call plotspec to draw spectrum
```

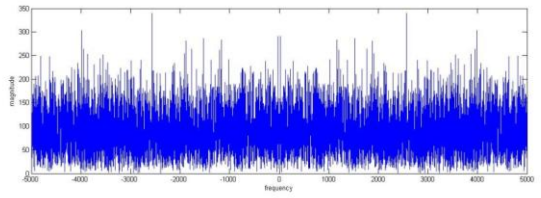
(b).(b)Use rand and the sign function to create a signal that is +1 with probability 1/2 and −1 with probability 1/2. Find the spectrum of the signal.

Solution: To generate a random variable that is +1 with probability ½ and -1 with probability –½, we need two events each with probability 1/2, and assign to each event either 1 or -1. We can use the random variable in part (a) which positive with probability ½ and negative with probability –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 could possibly generate a 0 but with a very small probability (in fact, with probability zero). In that case, x will be a zero since sign(0) = 0. This could be easily overcome by checking whether a 0 occurred in the output vector w and replacing it arbitrarily with 1.

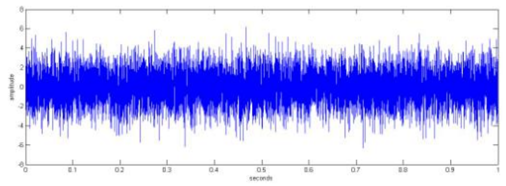


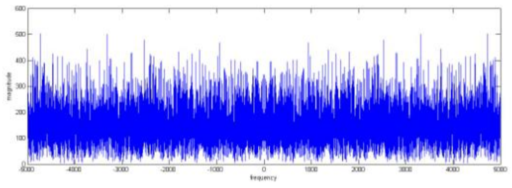


(b).(c) Use randn to create a signal that is normally distributed with mean 0 and variance 3. Find the spectrum of the signal.

Solution: To generate a random variable that is normally distributed with mean 0 and variance 3, we use the previous equations derived in 4.2 (a). Since the mean stays zero we set *b* to zero and *a* to sqrt(3) to obtain a variance of 3. After that we 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 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. You can find a length 31 PN sequence in Appendix L of the course reader, or you could use seqgen.pn to generate one in Matlab. Are there any frequencies at which the magnitude spectrum is exactly zero?

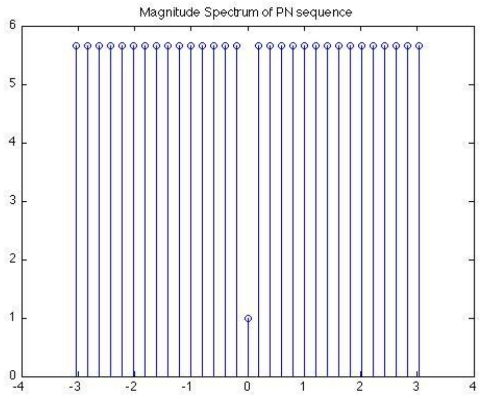
Solution:

```
%% Here's the length 31 pseudo-noise (PN) sequence from the 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’)
```



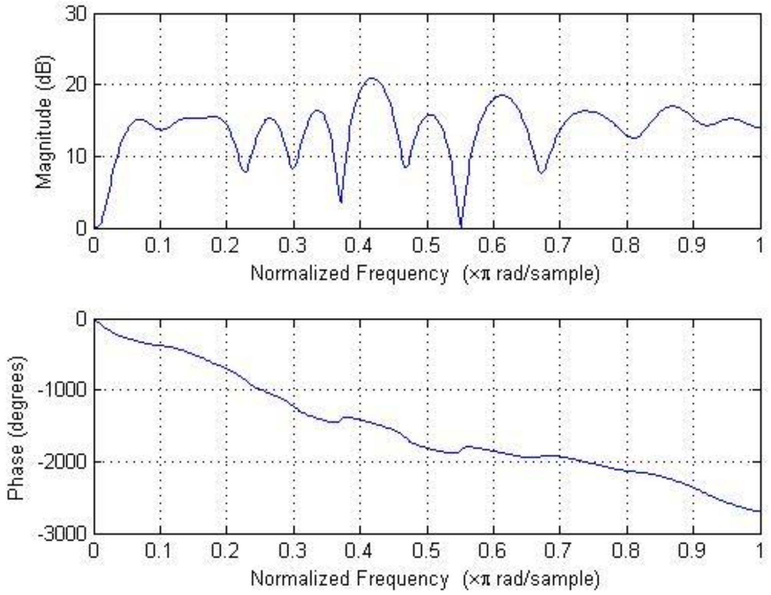
The pseudo-noise sequence is periodic with a period of 31 samples. When applying the fast

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 around 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 has two frequencies whose magnitudes get close to zero in linear units but they do not reach zero.



The fact that a PN sequence contains all frequencies makes PN sequences robust to linear time-invariant 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 delay. In practice, there can be some error in detecting the first peak due to noise.
- Estimate impulse response of linear time-invariant component of the unknown system. Given the transmitted training sequence x[n] and the sequence y[n], where y[n] is the received signal with the first Δ samples are removed, the receiver can estimate the frequency response of the channel using either
    - Frequency domain methods H(w) = Y(w) / X(w)
    - Time domain methods y[n] = h[n] * x[n] and backsolve for h[n]

The frequency domain method is possible because the PN sequence x[n] contains all discrete-time frequencies, and its Fourier transform X(w) never goes to zero.

Problem 4.2. Johnson, Sethares & Klein, *Software Receiver Design*, exercise 8.6, page 159.

Rerun correx.m with different-length data vectors (try loc=100, r=100 and loc=10, r=10). Observe how the location of the peak changes. 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: 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 = ones(1,31);    % header is a predefined string
  loc=100; r=100;          % 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 the 10,000 runs, the header of all ones is correctly identified about 50% of the time when the header is correlated with the received signal. The reason for this somewhat poor success rate lies in the fact that the receiver has a much higher probability of falsely identifying the marker when the header is a string of all ones. The header is falsely identified when the bit before header in a sequence is ‘1’. For example, 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:

```
totalCorrect = 0;
totalRuns = 10000;
for run = 1:totalRuns
  %% Begin correx.m from Software Receiver Design book
   header = [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=100; r=100;          % 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
```

The above code rarely results in falsely identified headers. This high success rate is achieved because the bit right before the header cannot trip the receiver, as a shift in the PN header will result in nearly zero correlation. Therefore, the only time that the correlation will falsely identify the header is when the same sequence of 31 consecutive random bits is equal to the PN header. The probability one chance in 2^(31), or 2^(-31) = 4.6566 x 10^{-10}.

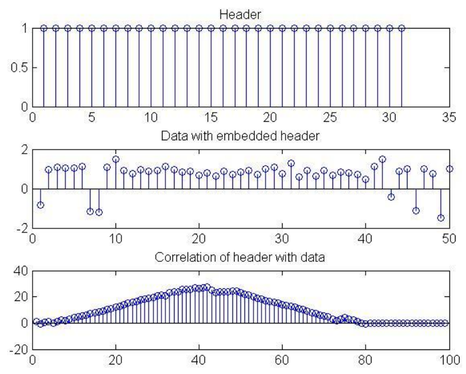
When we add noise to the signal using both headers, we obtain the following statistics:

loc, r	**Header of all ones**	**Pseudo-noise (PN) header**
100	0.4952	1
10	0.4922	1

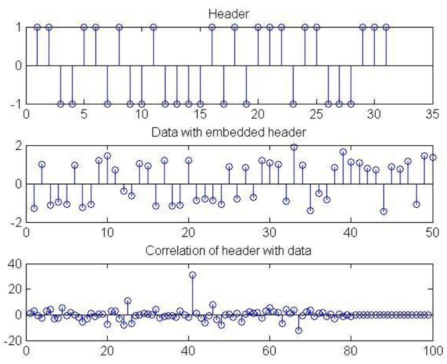
In order to see error in the detection of the PN header, we'd have to run more trials. Generating 2.1475x10^{11} random bits (100 / errorRate) would give confidence is seeing at least one error.

In the above simulation, the correct detection rate for the PN header is 2x better than that of the header of all ones for either choice of 100 or 10 for loc, r. We graphically show the correlation:

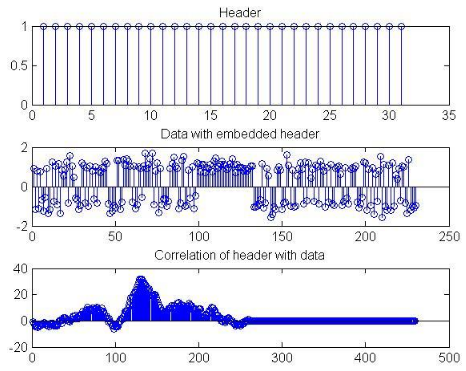
Header of all ones, loc = r = 10



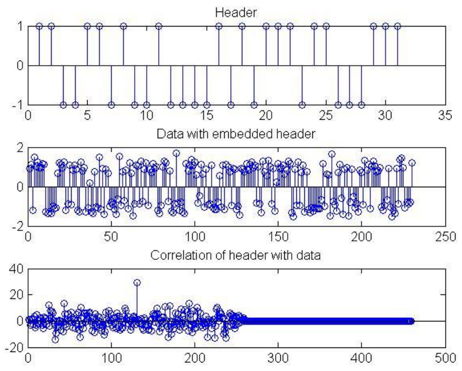
Header of PN sequence, loc = r = 10



Header of all ones, loc = r = 100



Header of PN sequence, loc = r = 100



We detect the peak much more clearly when we use a pseudo-noise sequence as the header.

Problem 4.3: Johnson, Sethares & Klein, *Software Receiver Design*, exercise 8.18, page 164.

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 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;                        % general random numbers on interval [p,q]
q = 4;
totalCorrect = 0;
totalRuns = 10000;
for run = 1:totalRuns
  %% Begin code from Software Receiver Design book
  header = 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;                        % general random numbers on interval [p,q]
q = 4;
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 = 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 header is correctly detected 100% of the time. To see an error in the detection of the header, we would have to run something like 10^{16} trials. So, there is an error rate, but it is very small.

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

Solution: Now let us proceed to add a channel to the 4-PAM signal by filtering the incoming data with the given impulse response [1, 0, 0, a, 0, 0, 0, b]. (Let us disregard noise). Let a = .1, b = .4

```
a = .1;
b = .4;
imp = [1, 0, 0, a, 0, 0, 0, b]; % impulse response of channel
n = [-3 -1 1 3];
p = -4;
q = 4;
totalCorrect = 0;
totalRuns = 10000;
for run = 1:totalRuns
  %% Begin code from Software Receiver Design book
%  header = 3*ones(1,31);  % header is a predefined string of all '3's
  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 = 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
  channel = filter(imp, 1, data); % add the channel
  sd=0.25; channel=channel+sd*randn(size(channel)) ; % add noise
  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
```

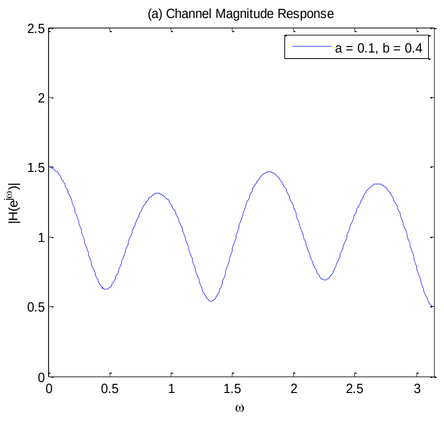
The simulation results gave 58.22% correct detection for a marker of all 3s, and 100% correct detection for the PN sequence. If we add noise with a standard deviation of 0.25, the simulation results gave 58 % and 100%, respectively.

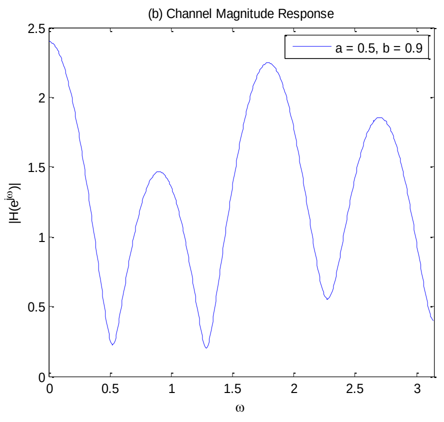
(b) Answer the same question for a = 0.5 and b = 0.9.

Solution: With a=.5 and b = .9, simulation gave 14.01% correct detection for the header of all 3s, and 76.4% for the PN header. If we add noise with standard deviation of 0.25, the simulation results give 13.86% correct detection for the header of all 3s, and 75.94% for the PN sequence.

The channel will filter the transmitted sequence. The frequency content in the header of all 3s is lowpass, so a bandpass or a narrower lowpass filter will result in significant loss of the marker sequence (and consequently, its detection by the receiver). On the other hand, the PN sequence fills the entire spectrum. Therefore, the PN header provides more robustness through the channel.

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





MATLAB Scripts from in Johnson, Sethares and Klein's *Software Receiver Design* textbook

The Matlab scripts should run "as is" in MATLAB or LabVIEW Mathscript facility.

1. Copy the .m files on your computer from the "SRD - MatlabFiles" folder on the CD ROM: http://users.ece.utexas.edu/~bevans/courses/rtdsp/homework/SRD-MatlabFiles.zip
2. Add the folder containing the .m files from the book to the search path.:
   - In MATLAB, use the `addpath` command
   - In LabVIEW, open the Mathscript window in LabVIEW by going to the Tools menu and select "Mathscript Window" (third entry), go the File menu, select "LabVIEW MathScript Properties" an d add the path.

Johnson, Sethares and Klein intentionally chose not to copyright their programs so as to enable their widespread dissemination.

Discussion of this solution set is available at http://www.youtube.com/watch?v=-hhcp8qXBQo.