Homework #5 Solutions

Problem 5.1 Steepest Descent
Johnson, Sethares & Klein, exercise 6.23, page 117, but use $J(x) = x^2 - 8x + 16$.

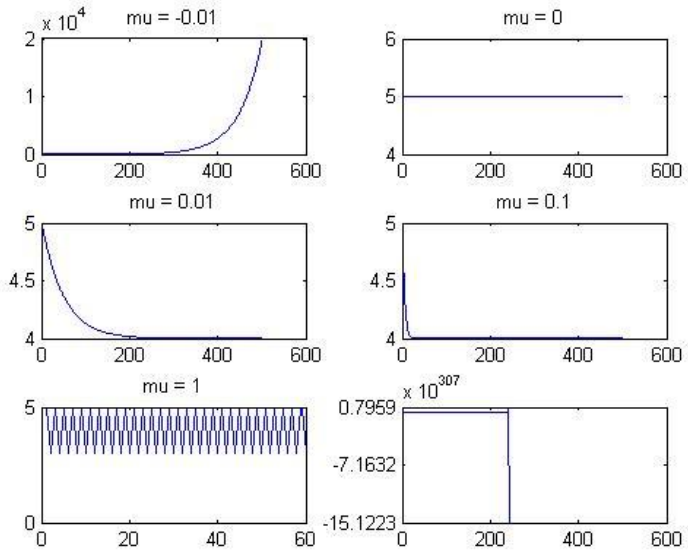
Explore the behavior of steepest descent by running polyconverge.m with different parameters, but use $J(x) = x^2 - 8x + 16$. Derive the adaptive equations.

Solution Prolog: The objective function is $J(x) = x^2 - 8x + 16 = (x - 4)^2$ which is non-negative. The objective function is a bowl. $J'(x) = 2x - 8$. The unique minimum of $J(x)$ is at $x = 4$.

a) Use values of μ of -0.01, 0.00, 0.01, 0.1, 1.0, 10.0. Can mu be too large or too small?

Solution: We modify "polyconverge.m" to evaluate the effect of step size μ on the convergence:

```
% polyconverge.m find the minimum of J(x)=x^2-8x+16 via steepest descent
N=500;                          % number of iterations
mu=[-.01 0 .01 .1 1 10];        % algorithm stepsize
x=zeros(size(1,N));             % initialize x to zero
x(1)=5;                         % starting point x(1)
all = zeros(length(mu), N);
for i=1:length(mu)
    for k=1:N-1
      x(k+1)=(1-2*mu(i))*x(k)+8*mu(i);     % update equation
    end
    all(i,:)= x;
end
subplot(3,2,1); plot (all(1,:)); title('mu = -0.01');
subplot(3,2,2); plot (all(2,:)); title('mu = 0');
subplot(3,2,3); plot (all(3,:)); title('mu = 0.01');
subplot(3,2,4); plot (all(4,:)); title('mu = 0.1');
subplot(3,2,5); plot (all(5,:)); title('mu = 1');
subplot(3,2,6); plot (all(6,:)); title('mu = 10');
```



In the update equation, a positive (negative) value of μ corresponds to finding the minimum (maximum) of the objective function. When μ is zero, no update to the initial guess is made.

From the plots, we see convergence when the step size μ is 0.01 or 0.1, divergence for μ of -0.01 or 10, oscillation for μ of 1, and trivial convergence for μ of 0. In fact, μ is used to estimate the next value $x[k+1]$ using the current value $x[k]$, and it should be $0 < \mu < 1$.

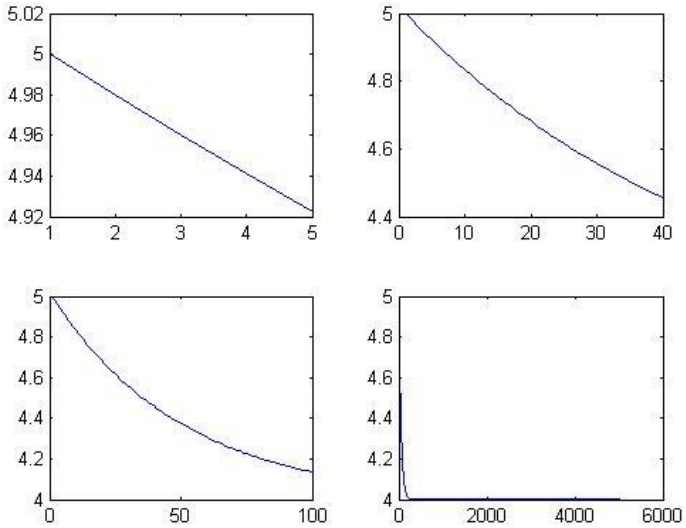
Given iteration $x_{i+1} = f(x_i)$, initial guess x_0, and interval $[a, b]$ that contains iterates $x_0, x_1, x_2, \ldots$, the fixed-point theorem says $x_0, x_1, x_2, \ldots$, will converge to a fixed point $x^* = f(x^*)$ if $|f'(x)| < 1$ for all $x \in [a, b]$. With $f(x) = (1 - 2\mu)\, x + 8\,\mu$, we have $f'(x) = 1 - 2\mu$ and guarantee $|f'(x)| < 1$ for $0 < \mu < 1$ regardless of the initial guess.

Choice of μ in (0,1) has an analogy with location of a single real-valued pole in a discrete-time LTI system. If we start the pole at zero and move it towards the unit circle, then the transition band will become narrower. If the pole is on the unit circle, the impulse response will be a unit step function (which oscillates at zero frequency). If the pole moves outside the unit circle, the impulse response grows without bound. The JSK book explains this view on pages 117-118.

b) Use μ of 0.01, and try N = 5, 40, 100, 5000. Can N be too large or too small?

Solution: We modified "polyconverge.m" to see effect of number of iterations on convergence:

```
% polyconverge.m find the minimum of J(x)= x^2-8x+16 via steepest descent
N=[5 40 100 5000];      % number of iterations
mu=.01;                 % algorithm stepsize
x=zeros(1);             % initialize x to zero
x(1)=5;                 % starting point x(1)
y = zeros(4);
for i=1:length(N)
    for k=1:N(i)-1
      x(k+1)=(1-2*mu)*x(k)+8*mu;     % update equation
    end
    y(i)= min(x)
    subplot(2,2,i); plot (x);
end
```



The above plots show convergence vs. number of iterations for μ = 0.01.

With N = 5, 40 and 100, we do not have enough iterations to converge to 4 for a step size of $\mu = 0.01$. The minimum values we can achieve are tabulated below:

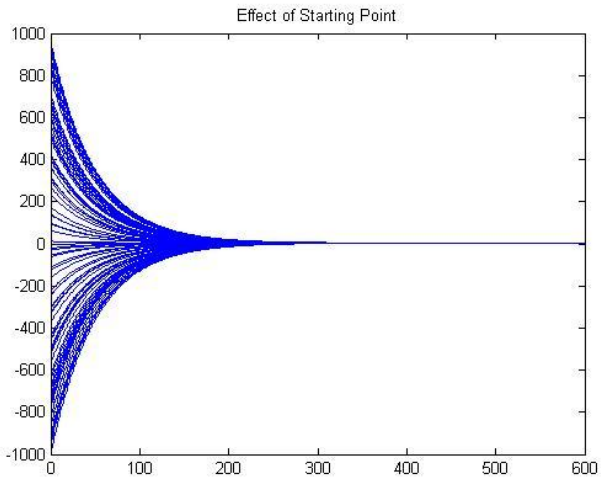
N	Min
5	4.9224
40	4.4548
100	4.1353
500	4.0001

If we use a step size of 0.1, we converge to x = 4 with about 500 iterations. In practice, the number of iterations required depends on the desired numerical precision.

c) Try a variety of values of x(1). Can x(1) be too large or too small?

Solution: Changing the initial guess for x did not affect convergence. It will affect the speed of convergence, for a fixed step size, depending on how far the starting value is from the minimum. We used the following code to obtain our results:

```
% polyconverge.m find the minimum of J(x)= x^2-8x+16 via steepest descent
N=600;                          % number of iterations
mu=.01;                         % algorithm stepsize
x=zeros(size(1,N));             % initialize x to zero
for i=1:100
    x(1)=randi([-1000 1000],1,1); % starting point x(1)
    for k=1:N-1
      x(k+1)=(1-2*mu)*x(k)+8*mu;    % update equation
    end
    plot (x);
    hold on;
end
hold off;
title('Effect of Starting Point');
```



This plot shows with 100 different starting points ranging from -1000 to 1000, all of the curves converge to our desired value of $x = 4$.

Given iteration $x_{i+1} = f(x_i)$, initial guess x_0, and interval $[a, b]$ that contains iterates $x_0, x_1, x_2, \ldots$, the fixed-point theorem says $x_0, x_1, x_2, \ldots$, will converge to a fixed point $x^* = f(x^*)$ if $| f'(x) | < 1$ for all $x \in [a, b]$. With $f(x) = (1 - 2\mu)\, x + 8\,\mu$, we have $f'(x) = 1 - 2\mu$ and guarantee $| f'(x) | < 1$ for $0 < \mu < 1$ regardless of the initial guess.

Problem 5.2 Transceiver Simulation
Johnson, Sethares & Klein, exercise 9.2, page 175. Use $M = 100, 20, 18$ instead of $M = 1000, 25, 10$.

Using idsys.m, examine the effect of using different oversampling frequencies. Try M = 100, 20, 18. What are the limiting factors that cause some to work and others to fail?

Solution: This problem uses 4-PAM encoding, i.e. 2 bits per symbol. The message to be sent is a string of ASCII characters. Since each ASCII character is eight bits, each ASCII character will correspond to four two-bit symbols.

The problem statement refers to a parameter *l* which was set to 125 in idsys.m. Page 172 of Software Defined Radio has the following explanation: "The initial delay of 125 corresponds to half the length of the lowpass filter (0.5*fl) plus half the length of the correlator filter (0.5*M) plus half a symbol period (0.5*M), which accounts for the delay from the start of each pulse to its peak."

In idsys.m, *M* is number of samples in a symbol period, a.k.a. oversampling ratio. For $M = 100$, 20 and 18, we evaluate communication performance. The channel has no degradation.

```
%idsys.m:  idealized transmission system
%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;                         % number of samples in a symbol period
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 signal
t=1/M:1/M:length(x)/M;          % T/M-spaced time vector
fc=20;                          % carrier frequency parameter
c=cos(2*pi*fc*t);               % carrier signal
r=c.*x;                         % modulate message with carrier
```

If we convert the last four lines of the above code to the following,

```
n=1:length(x);                  % t = n / M
fc=20;                          % carrier frequency parameter
c=cos((2*pi*fc/M)*n);           % carrier signal with w0 = 2 pi fc / M
```

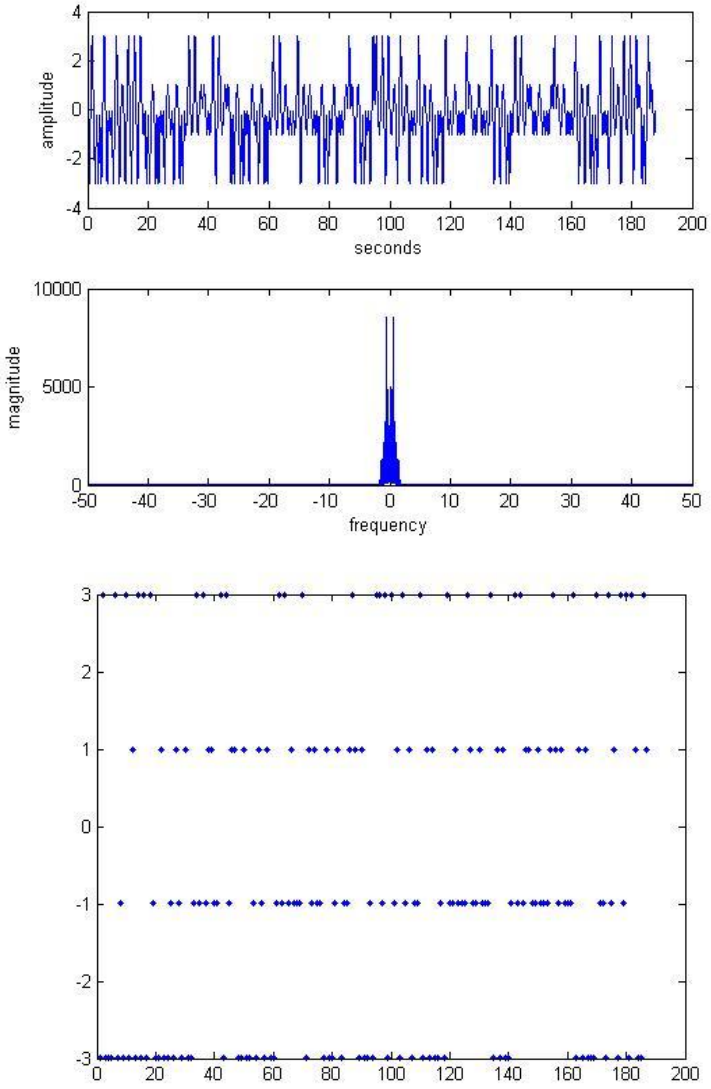
```
r=c.*x;
```

then the carrier frequency is $\omega_0 = 2\pi\ f_c / M$ rad/sample.

To prevent aliasing, $M > 2 f_c$. With $f_c = 20$, $M > 40$ would prevent aliasing, and the combination of modulation by $c[n]$ and demodulation by $c[n]$ works properly. Even when aliasing occurs for $M \leq 40$, modulation by $c[n]$ and demodulation by $c[n]$ works properly for most values of M.

Aliasing occurs with respect to the carrier frequency $f_c = 20$ when $M \leq 40$. When $M = 10$, $\omega_0 = 4\pi$ rad/sample which aliases to $\omega_0 = 0$ rad/sample. When $\omega_0 = 0$ rad/sample, the vector c is a vector of all ones, and the transmission is baseband instead of bandpass. More on this later.

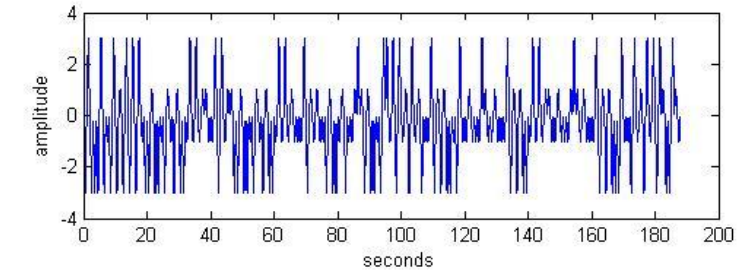
For $M = 100$:

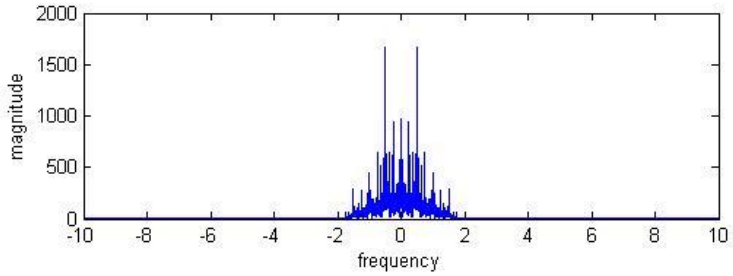


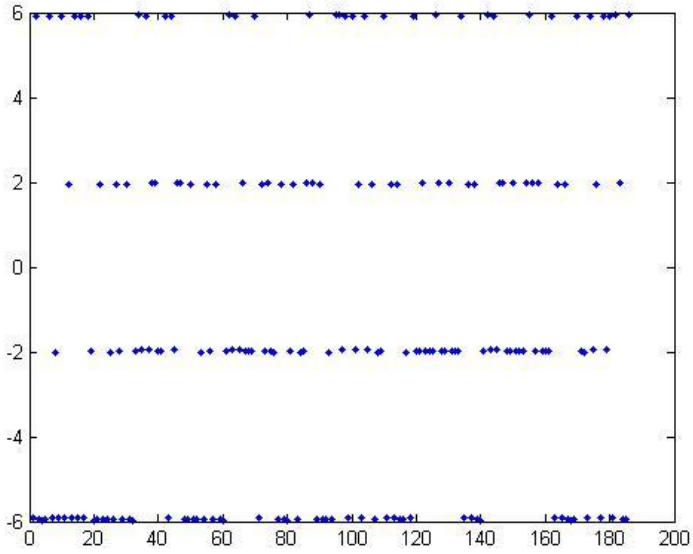

```
reconstructed_message =

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

For $M = 20$:



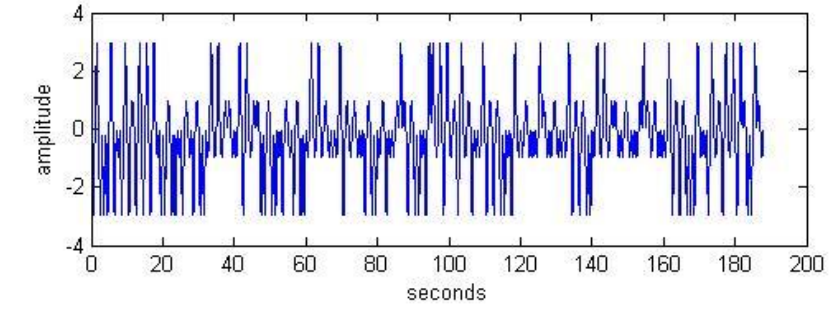


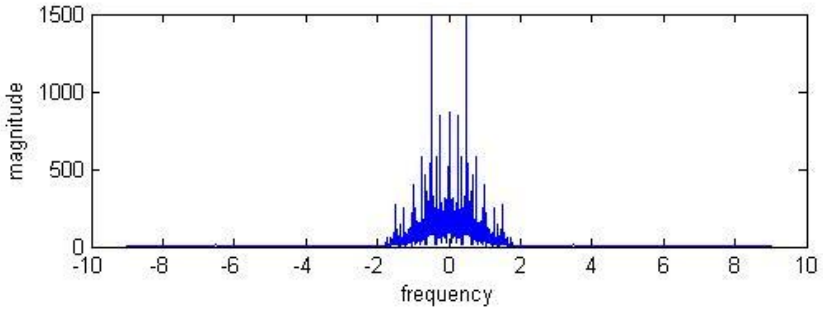


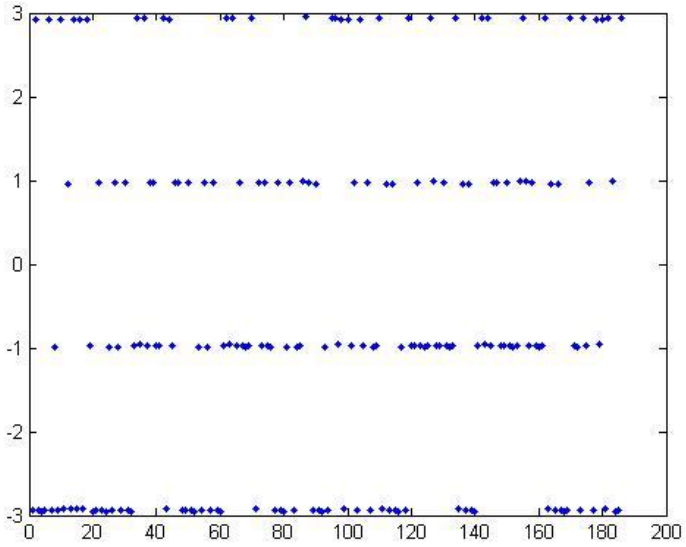
```
reconstructed_message =

00234       wish       werd `n sc`r
eyer wiener 4678
```

For $M = 18$:







```
reconstructed_message =

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

The code for the receiver from idsys.m follows:

```
%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),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)
```

The receiver decodes the string correctly for $M = 18$ and $M = 100$, but not for $M = 20$. For other values, the receiver decodes the string correctly for $M = 25$ and $M = 1000$ but not for $M = 10$.

In the presence of additive noise, the larger the value of M, the more samples the receiver uses in the correlation block (a.k.a. the matched filter block), the better the communication performance. However, there is no additive noise in this problem.

Our demodulating filter passes normalized frequencies up to 0.1, but the Hamming pulse for $M = 20$ has a lowpass normalized bandwidth of 0.2 from eyeballing its spectrum via freqz(hamming(20)). The Hamming pulse for $M = 10$ has a lowpass normalized bandwidth of 0.5, and by changing the demodulating filter, symbol errors were still high (15-20%).

The problem when $M = 10$ and $M = 20$ is that the transmission is baseband and not bandpass. In the receiver, the following line of code is necessary to scale the output of the demodulating filter for bandpass transmission:

```
x3=2*filter(b,1,x2);            % LPF & scale due to bandpass transmission
```

For baseband transmission, this line should be changed to

```
x3=filter(b,1,x2);              % LPF (no scaling for baseband transmission)
```

and the string will be correctly decoded.

For baseband PAM transmission, the average value is zero if each symbol value is equally likely. After upconversion, there is no transmit power at the carrier frequency, which makes it difficult for the receiver to locate and track the carrier frequency.

Problem 5.3 Preprocessing for Carrier Phase Recovery
Johnson, Sethares & Klein, exercise 10.1, page 198.

In practice, data scramblers (explored in lab #4) are used in a baseband transmitter to break up long strings of 0s or 1s so that the occurrence of 0s and 1s are approximately equally likely at the scrambler output. After constellation mapping, upsampling and pulse shaping, the baseband transmission is upconverted. Very little transmit power is at the carrier frequency, and that small amount of power is typically buried in noise, which makes it difficult for the receiver to locate and track the carrier frequency.

In the receiver, we can emphasize the frequency content around the carrier frequency by applying the right static nonlinearity, such as a squaring block or an absolute value block. The nonlinearity will create a harmonic (or multiple harmonics) of the carrier frequency and each harmonic will have significant power present.

The mathematical analysis for two of these cases is worked out in the solution for problem #4 on midterm #1 in Spring 2011. This midterm problem concerns downconversion and hence a lowpass filter is applied after the static nonlinearity. In this homework problem, we are trying to extract information about the received carrier frequency, and hence, we would like to apply the appropriate bandpass filter after the static nonlinearity. The analysis of the nonlinearity in the frequency domain is the same in both cases. The solution for problem 1.3 on midterm #1 in spring 2010 might also be helpful.

Consider bandpass transmission $m(t)\cos(2\pi f_0 t)$ in which the message signal $m(t)$ is lowpass. The spectrum is $\frac{1}{2} M(f + f_0) + \frac{1}{2} M(f - f_0)$. We'd like to lock onto f_0 by looking for spectral content at f_0. When the average value of $m(t)$ is zero, i.e. $M(0) = 0$, the content at the carrier frequency is zero. Next, consider putting the bandpass transmission through a squaring operation to obtain $m^2(t)\cos^2(2\pi f_0 t)$. We'd look for spectral content at frequency $2 f_0$ because

$$\cos^2(2\pi f_0 t) = \frac{1}{2} + \frac{1}{2}\cos(2(2\pi f_0 t))$$

The average value of $m^2(t)$ is much greater than zero even if the average value of $m(t)$ were zero, and hence, we'd have significant content at frequency $2 f_0$. If the bandpass transmission were raised to an odd power n, then the spectral content at frequency $n f_0$ would still be zero.

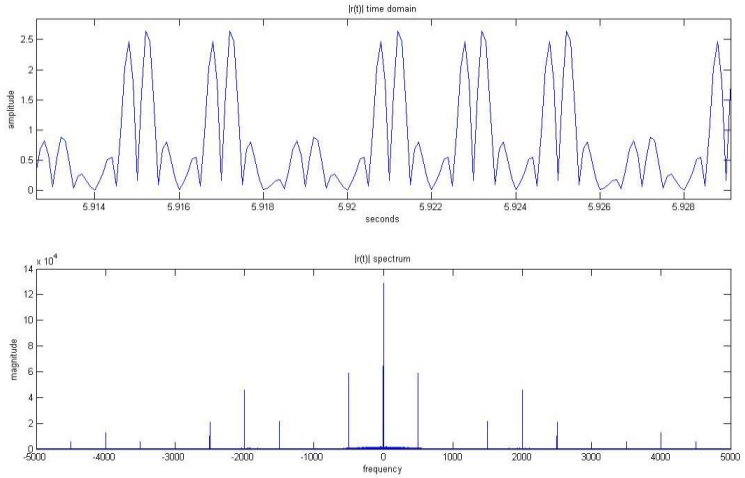
You must run "pulrecsig.m" to generate to suppressed carrier signal defined in MATLAB variable rsc, and then run "pllpreprocess.m". The *Software Receiver Design* book describes this.

The "pulrecsig.m" script generates two upconverted signals. One is a "suppressed carrier" signal `rsc`; i.e., there isn't any power at the carrier frequency. The other is called a "large carrier" signal `rlc`; i.e., there is significant power at the carrier frequency. The "pulrecsign.m" script then uses the fast Fourier transform (FFT) to locate the carrier frequency by find the peak in the magnitude

spectra of the upconverted signals. The FFT method finds the carrier frequency `freqL` exactly for the large carrier signal and carrier frequency `freqS` approximately for the suppressed carrier signal (with about 3.5% error). The phase offset is computed by using the phase of the FFT at the estimated carrier frequency value. The FFT method gives large error in its phase offset estimates.

a) Try replacing the $r^2(t)$ with $|r(t)|$ in pllpreprocess.m. Does this result in a viable method of emphasizing the carrier?

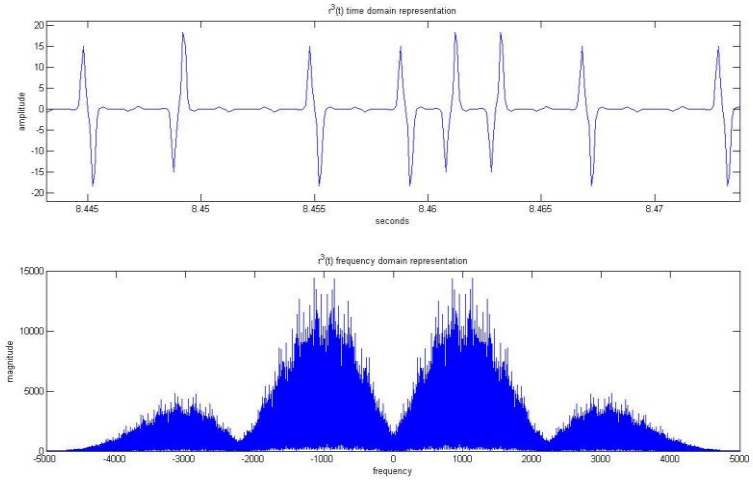
Solution: Replacing $r^2(t)$ by $|r(t)|$ emphasizes the carrier. Notice that $|r(t)| = |s(t)||\cos(2\pi f_0 t + \phi)|$. Rewriting $|s(t)|$ as a sum of its (positive) average value and the variation about this average yields $|s(t)| = |s(t)|_{avg} + v(t)$, which in turn implies that $|r(t)| = (|s(t)|_{avg} + v(t))|\cos(2\pi f_0 t + \phi)| = |s(t)|_{avg}|\cos(2\pi f_0 t + \phi)| + v(t)|\cos(2\pi f_0 t + \phi)|$. The term $|\cos(2\pi f_0 t + \phi)|$ has a period of $T_0/2$ and a fundamental frequency of 2 f_0. This results in a Fourier series spectrum that is a weighted impulse train at multiples of 2 f_0. Since its spectrum is discrete, frequencies can be easily separated and the bandpass filter described in *pllpreprocess.m* still works. We plot the spectrum of $|r(t)|$ in the following figure.



b) Try replacing $r^2(t)$ with $r^3(t)$. Does this result in a viable method of emphasizing the carrier?

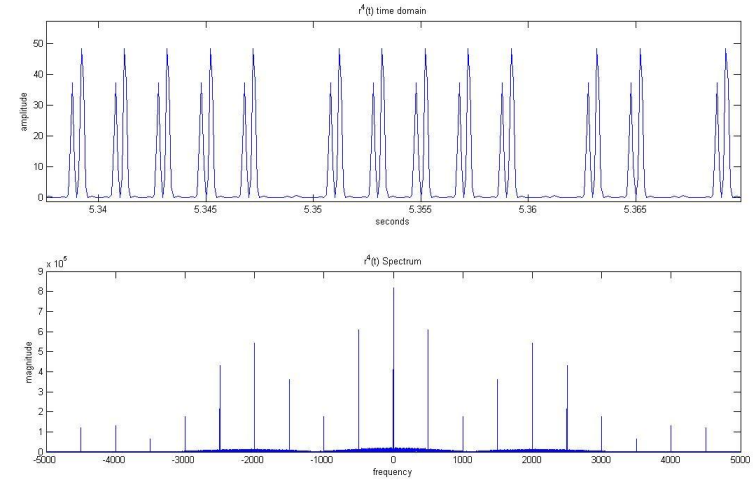
Solution: Replacing $r^2(t)$ by $r^3(t)$ does not work. We would like to use pre-emphasis because the average value of the message $m(t)$ is zero and hence there is no signal power at the carrier frequency of the transmitted signal. The average value of $m^3(t)$ is also zero, and there is no energy at f_0 or 3 f_0. Note that $\cos^3(2\pi f_0 t + \phi) = \frac{3}{4}\cos(2\pi f_0 t + \phi) + \frac{1}{4}\cos(2\pi(3 f_0)t + \phi)$.

On the next page, we plot the spectrum of $r^3(t)$ and see that no distinct tone frequencies are present that would enable us to recover the carrier.



c) Can you think of other functions that will result in viable methods of emphasizing the carrier?

Solution: Another function that works is a fourth-order (fourth-power) nonlinearity and the bandpass filter would be centered at either 2 f_0 or 4 f_0. There is more signal strength at 2 f_0:



Other nonlinear functions will work. We could use exp(x) followed by a bandpass filter centered at 2 f_0 because $\exp(x) = 1 + x + (1/2!)\,x^2 + (1/3!)\,x^3 + \ldots$ from its series expansion.

d) Will a linear function work? Why or why not?

Solution: A linear function will not work. A linear function cannot introduce new frequencies into the signal, so it is impossible for a linear function to isolate the carrier frequency tone.

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=o6OGrmfBs7I.