# ELEC 403 --- Lab 2

This script is used to fulfill the requirements of the lab in ELEC 403.

generate s[n] 1

generate w[n] 1

Plot recieved signal x[n] 2

2.6.1 2

2.6.2 discrete Fourier Transform for s[n] and w[n] 3

Plot recieved signal x[n] 4

2.6.3 5

2.6.4 SNRb is the Signal to Noise Ratio before and SNRa is the Signal to Noise Ration after filter is applied 6

2.6.5 6

## generate s[n]

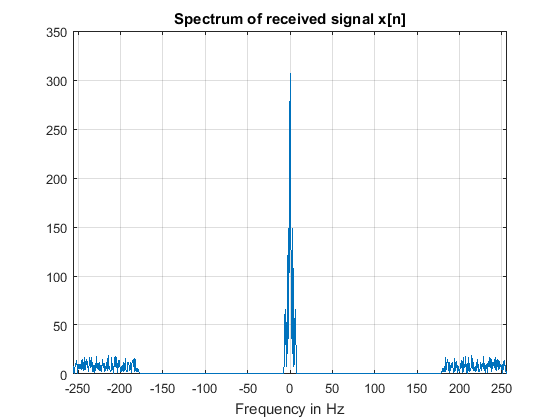
t = 0:1/512:(2-1/512);  
t = t(:);  
randn('state', 7)  
a = 0.2\*randn(7,1);  
randn('state', 19)  
b = 0.2\*randn(7,1);  
s = 0.3\*ones(1024,1);  
for i = 1:7,  
 s1 = a(i)\*sin(2\*pi\*i\*t);  
 s2 = b(i)\*cos(2\*pi\*(i-0.5)\*t);  
 s = s + s1 + s2;  
end

## generate w[n]

randn('state',9); % sets a seed state for generating a random sequence  
w0 = randn(1024,1); % generate 1024 Gaussian white random samples  
mw = mean(w0); % evaluate its mean value  
w0 = w0 - mw; % modify w0 to have a zero-mean  
c = 0.3/sqrt((w0'\*w0)/1024);  
w0 = c\*w0; % modify w0 to have a standard deviation = 0.3  
h = fir1(250,0.7,'high'); % get a good highpass FIR filter with cutoff freq. = 0.7  
w1 = conv(h,w0); % apply highpass filtering to the white noise sequence  
w = w1(126:1149); % cut the filtered sequence to a right size  
w = w(:); % make sure w[n] is a column vector

## Plot recieved signal x[n]

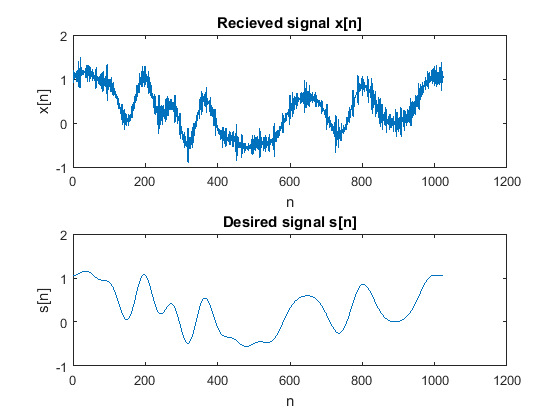
x = s + w;  
xf = fft(x); % perform FFT of signal x[n]  
xf = fftshift(xf); % shift zero-frequency component to center of spectrum  
f = -256:512/1023:256; % define an appropriate frequency axis for plotting  
plot(f,abs(xf))  
axis([-256 256 0 350])  
grid  
xlabel('Frequency in Hz')  
title('Spectrum of received signal x[n]')  
%print('Prod272\_4','-dpng','-r600')



## 2.6.1

fprintf('Starting 2.6.1\n')  
subplot(2,1,1);  
x = s + w;  
L = length(x);  
n=1;  
plot(n:L,x);  
title('Recieved signal x[n]')  
xlabel('n')  
ylabel('x[n]')  
subplot(2,1,2);  
plot(n:L,s);  
title('Desired signal s[n]')  
xlabel('n')  
ylabel('s[n]')  
%print('Prod271','-dpng','-r600')

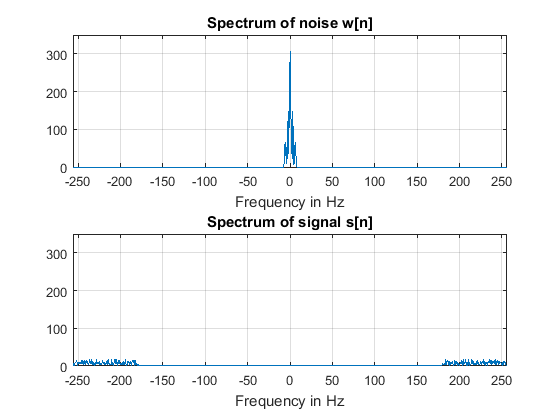
Starting 2.6.1



## 2.6.2 discrete Fourier Transform for s[n] and w[n]

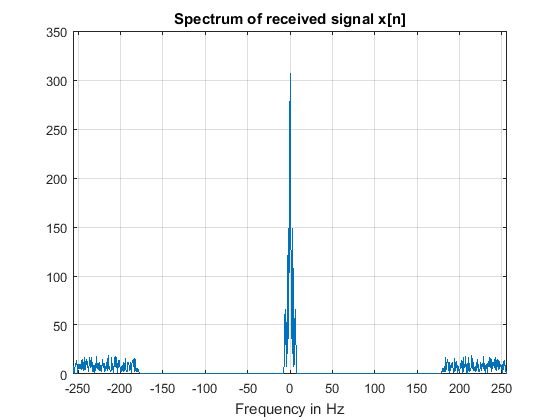
fprintf('Starting 2.6.2\n')  
subplot(2,1,1);  
sf = fft(s); % perform FFT of signal s[n]  
sf = fftshift(sf); % shift zero-frequency component to center of spectrum  
f = -256:512/1023:256; % define an appropriate frequency axis for plotting  
plot(f,abs(sf))  
axis([-256 256 0 350])  
grid  
xlabel('Frequency in Hz')  
title('Spectrum of noise w[n]')  
subplot(2,1,2);  
wf = fft(w); % perform FFT of signal w[n]  
wf = fftshift(wf); % shift zero-frequency component to center of spectrum  
f = -256:512/1023:256; % define an appropriate frequency axis for plotting  
plot(f,abs(wf))  
axis([-256 256 0 350])  
grid  
xlabel('Frequency in Hz')  
title('Spectrum of signal s[n]')  
%print('Prod272\_1','-dpng','-r600')

Starting 2.6.2



## Plot recieved signal x[n]

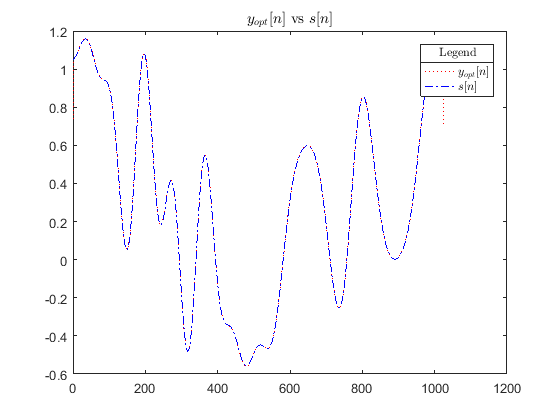
x = s + w;  
xf = fft(x); % perform FFT of signal x[n]  
xf = fftshift(xf); % shift zero-frequency component to center of spectrum  
figure % make new figure instead of a subplot  
f = -256:512/1023:256; % define an appropriate frequency axis for plotting  
plot(f,abs(xf))  
axis([-256 256 0 350])  
grid  
xlabel('Frequency in Hz')  
title('Spectrum of received signal x[n]')  
%print('Prod272\_2','-dpng','-r600')



## 2.6.3

fprintf('Starting 2.6.3\n')  
h=[0.07319357 0.25 0.3561286 0.25 0.0731967];  
y = conv(x,h,'same'); % sames crops unneccessary entries.  
L= length(y);  
plot(n:L,y,':r');  
hold on  
plot(n:L,s,'-.b');  
lgd = legend('$y\_{opt}[n]$','$s[n]$');  
title(lgd,'Legend')  
title('$y\_{opt}[n]$ vs $s[n]$','Interpreter','latex')  
set(lgd,'Interpreter','latex')

Starting 2.6.3



## 2.6.4 SNRb is the Signal to Noise Ratio before and SNRa is the Signal to Noise Ration after filter is applied

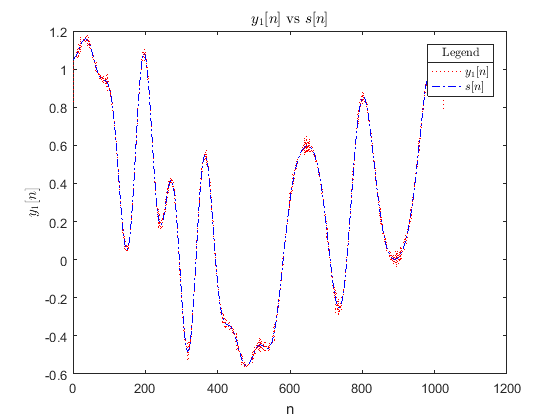
fprintf('Starting 2.6.4\n')  
SNRb = 20 \*log10(norm(s)/norm(x-s));  
SNRa = 20 \*log10(norm(s)/norm(y-s));  
fprintf('The Signal to Noise Ration before is: %2.4f\n',SNRb)  
fprintf('The Signal to Noise Ration after is: %2.4f\n',SNRa)

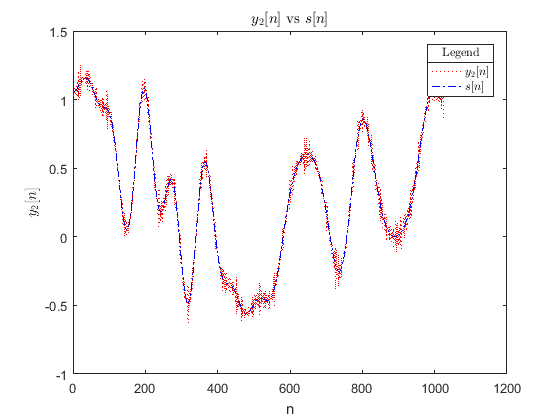
Starting 2.6.4  
The Signal to Noise Ration before is: 11.5701  
The Signal to Noise Ration after is: 31.8072

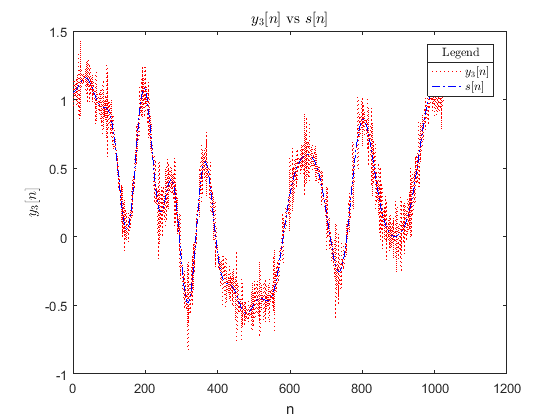
## 2.6.5

fprintf('Starting 2.6.5\n')  
womega = [0.3 0.6 0.9];  
n=1;  
for i=1:3  
 figure  
 fprintf('Interation %1d\n',i)  
 h(i,:)= fir1(4,womega(i));  
 y=conv(x,h(i,:),'same');  
 SNRbefore(i) = 20 \*log10(norm(s)/norm(x-s));  
 SNRafter(i) = 20 \*log10(norm(s)/norm(y-s));  
 plot(n:L,y,':r');  
 hold on  
 plot(n:L,s,'-.b');  
 plotname =sprintf('$y\_%d[n]$', i);  
 lgd = legend(plotname,'$s[n]$');  
 title(lgd,'Legend')  
 titlename = sprintf('%s vs $s[n]$',plotname);  
 title(titlename,'Interpreter','latex')  
 xlabel('n');  
 ylabel(plotname,'Interpreter','latex');  
 set(lgd,'Interpreter','latex')  
 fname = sprintf('Prod275%d', i);  
 print(fname,'-dpng','-r600')  
end  
T = table(womega(:),SNRbefore(:),SNRafter(:));  
T.Properties.VariableNames = {'AngularFrequency' 'SignaltoNoiseRatioBefore' 'SignaltoNoiseRatioAfter'}

Starting 2.6.5  
Interation 1  
Interation 2  
Interation 3  
  
T =   
  
 AngularFrequency SignaltoNoiseRatioBefore SignaltoNoiseRatioAfter  
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
 0.3 11.5700600694238 28.3447550455541   
 0.6 11.5700600694238 20.8126782853119   
 0.9 11.5700600694238 13.4309113832353







[Published with MATLABｮ R2016a](http://www.mathworks.com/products/matlab)