

## 43 Spectral Analysis to Find a Hidden Message

As you know, the fast Fourier transform (`fft()` in MATLAB), turns time-domain signals  $x$  into frequency-domain equivalents  $X$ . Here you will analyze a given signal from the **course** site and apply several of the common treatments so as to get clean power spectral densities.

1. First, download the  $x$  data file from the **course** site; it is called `computeSpectra.dat`. This signal  $x$  looks like a mess at first because of the noise, but we will tease out some specific information about  $X$ . Make a plot of the data versus time, assuming the time step is 0.0001 seconds. What features do you see if you zoom in on  $x$ , if any?

*It's an evidently narrowband signal with content around 1.6kHz (10krad/s), but probably with a lot of noise. Overall it is very hard to see anything beyond this.*

2. Use the FFT to make the transformed signal  $X$ . Plot the magnitude of  $X$  versus frequency; recall that the frequency vector corresponding to  $X$  goes from zero to the roughly the sampling rate (use the instructions from your previous homework). In your plot, only show the frequency range near 10000 *rad/s* where there is significant energy, and remember the  $2/n$  scaling that is needed to put FFT peaks into the same units as  $x$ . What do you see?

*See the top plot of the psd's. There are about fifteen peaks at frequencies slightly above 10 krad/s.*

3. Treatment 1: Windowing. The Fourier transform operation on a finite-length signal  $x$  assumes periodicity, and so any discontinuity between the last and the first points in  $x$  is a feature that will be accounted for in  $X$ . In general we don't want this effect, because it muddies up the water and makes it harder to see the details in  $X$  that we are interested in. For this and some other reasons, it is common to multiply  $x$  by a windowing function, before taking the transform. There are many such windows, developed according to different metrics. If you do no windowing, this is called the boxcar window! An easy one we can consider here is the half-sine window,  $w = \pi \sin(\pi[0 : n - 1]/(n - 1))/2$ , where  $n$  is the length of the signal. Multiply pointwise this window by  $x$ , and then plot the magnitude of the FFT, i.e., `abs(fft(x.*w))` in MATLAB. How does it compare with your psd from above?

*The window and the windowed data are plotted, the transform of this is the middle of the fft's. The fifteen peaks here might be better formed than for the raw data.*

4. Treatment 2: Zero-padding. The frequency scale to go with  $X$  is defined by the sampling rate and the number of samples. To get really fine resolution in frequency, we would want to sample much faster or sample a lot longer. We can't fix either of these once the data is obtained of course, but we can lengthen the signal artificially, either by stacking it end to end against itself, or more commonly by adding a lot of zeros to the end of  $x$ . Do this for the windowed signal  $xw$ , with a block of  $7n$  zeros added on the end. Again plot the magnitude of the resulting transform. Note that you will have to make up a new frequency vector for this case since you now have eight

times as many points, and ditto for your amplitude scaling of the transform. What do you see?

*See the bottom plot of the fft's. The peaks now appear much more clearly because of the improved frequency resolution, and in fact they take amplitudes of about [1,2,3], and at very evenly spaced frequencies.*

5. BONUS. The psd peaks you see encode a six-character message which you can only discover reliably if the processing above is done correctly. To **crack the code**, consider that each peak's magnitude indicates two bits of information, and thus that each set of four peaks encodes a full byte. The characters in the message are read left to right on the frequency axis; the left-most peak in each set of four gives the most significant bits so that, for example, a first peak height of three means the most significant bits of the character are [1,1] and the decimal equivalent is  $128 \times 1 + 64 \times 1 = 192$ . The first of 24 such two-bit peaks occurs right at  $10000\text{rad/s}$ . Use the standard ASCII conversion table where [A-Z] corresponds with the decimal numbers 65-90. What is the message? All-caps, it is a single word you know!

*We read off the bit pairs that go with each peak starting at  $10\text{krad/s}$ : [1020 1033 1102 1032 1011 1110]. These transform to the bytes [01001000 01001111 01010010 01001110 01000101 01010100] or in decimal form [72 79 82 78 69 84] or in characters [HORNET].*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Simple Spectral Analysis
```

```
clear all;
```

```
message = 'HORNET'; % the message to be encoded in the psd
% get the most significant four bits and least significant four
% bits (decimal form)
```

```
for i = 1:length(message);
    word = dec2bin(message(i));
    llsb(i) = str2num(word(7)) + 2*str2num(word(6)) ;
    lsb(i) = str2num(word(5)) + 2*str2num(word(4)) ;
    msb(i) = str2num(word(3)) + 2*str2num(word(2)) ;
    mmsb(i) = str2num(word(1)) ;
```

```
end;
```

```
dec2bin(message)
[mmsb' msb' lsb' llsb']
```

```
n = 8193 ; % length of time series
basewn = 10000 ; % base frequency, rad/s
dwn = 25 ; % delta frequency between peaks
dt = .0001 ; % time step
nz = 7 ; % number zero padding blocks (of size n)
noise = 7 ; % noise standard deviation
```

```

npeaks = 4*length(message); % number of peaks - two per character of
    % message
wn = basewn + [0:dwn:(npeaks-1)*dwn] ; % frequencies of the peaks

wsamp = 2*pi/dt ; % sampling rate
t = 0:dt:(n-1)*dt ; % time vector

if 1, % load the data
    load computeSpectra.dat ;
    x = computeSpectra ;
    clear computeSpectra ;
else, % or make up the signal: noise plus all the elements for msb's
    % and lsb's
    x = randn(1,n)*noise ;
    for i = 1:length(wn)/4,
        x = x + mmsb(i)*sin(wn(4*i-3)*t+rand*2*pi) + ...
            msb(i)*sin(wn(4*i-2)*t+rand*2*pi) + ...
            lsb(i)*sin(wn(4*i-1)*t+rand*2*pi) + ...
            llsb(i)*sin(wn(4*i)*t+rand*2*pi) ;
    end;

    save computeSpectra.dat x -ascii
end;

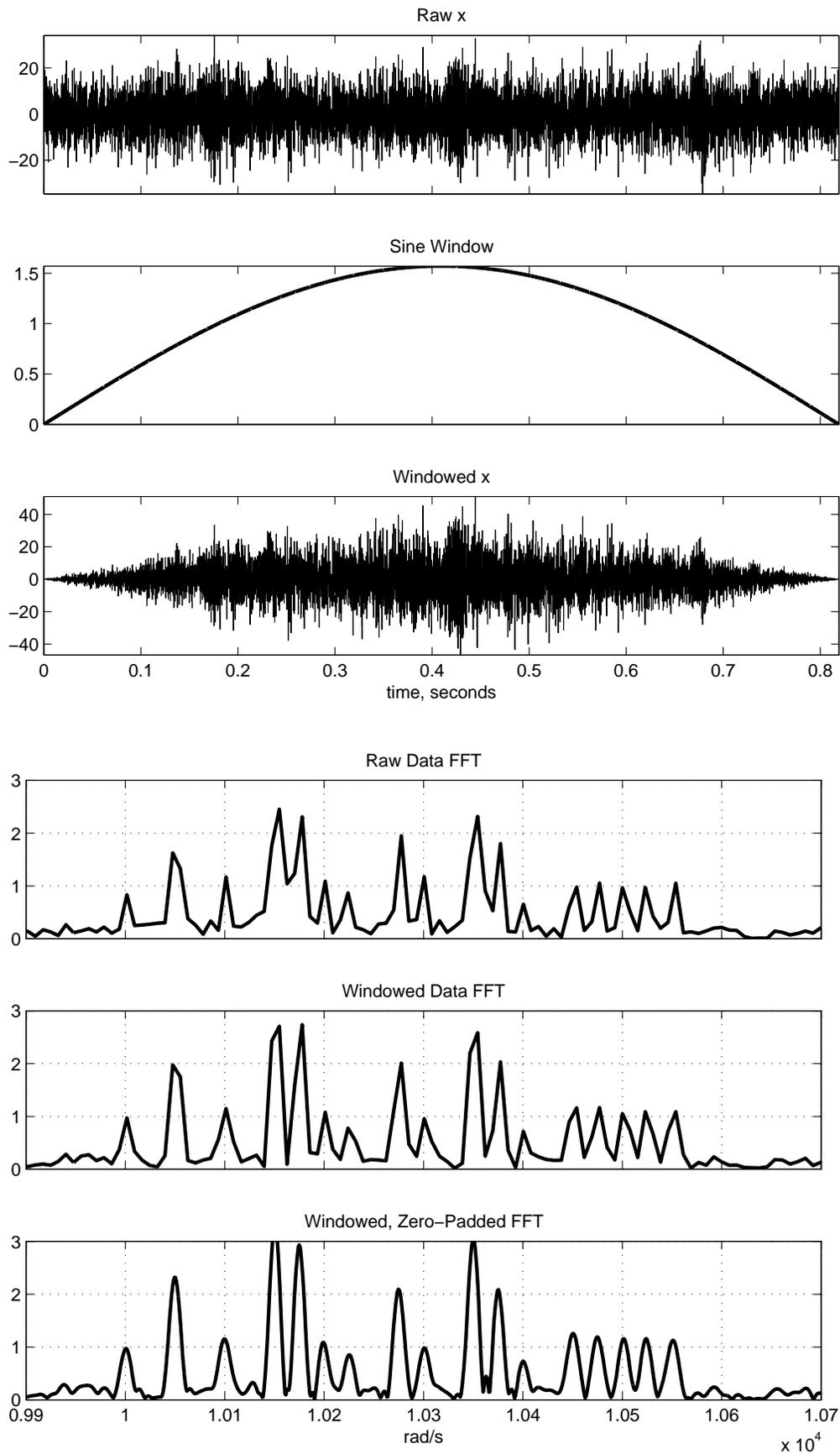
figure(1);clf;hold off;
subplot(311);
plot(t,x) ;
title('Raw x');
axis('tight');
set(gca,'XTickLabel',[]);

fx = fft(x)*2/n; % here is the scaled fft
w = [0:n-1]/(n-1)*wsamp ; % and its corresponding frequency
figure(2);clf;hold off;
subplot(311);
plot(w,abs(fx),'LineWidth',2);
axis([basewn-4*dwn basewn+dwn*(4+npeaks) 0 max(max([msb lsb]))]);
grid;
title('Raw Data FFT');
set(gca,'XTickLabel',[]);

win = sin([0:n-1]/(n-1)*pi)*pi/2;
% the sine window, with scaling to preserve the energy in x
figure(1);

```





MIT OpenCourseWare  
<http://ocw.mit.edu>

2.017J Design of Electromechanical Robotic Systems  
Fall 2009

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.