Lab 2. Spectral Analysis in Matlab

Introduction

This lab will briefly describe the following topics:
- Signal statistics
- Discrete Fourier Transform
- Power Spectral Density estimation
- Time-varying spectra
- Wavelets

There are a number of important statistical signal processing concepts in this Lab. We will fully appreciate some of them as we proceed in class. You can certainly read about them in this Lab, but do not have to do the simulations. (Some functions are missing).

For this Lab. do the simulations up to **Q2** on page 3. Then go to "Lab 19 The Fast Fourier Transform" and do that Lab.

For your report, you need to answer **Q1, Q2** and the 7 questions on page 3 of Lab.19.

Statistical Signal Processing

Repeated measurements of a signal x under identical environmental conditions typically yield different waveforms. The value $x(n)$ of a signal at sample n is not a constant, but rather a random variable with a certain distribution.

A signal is *stationary* if its distribution at any n is independent of time shifts, i.e., if $x(n)$ and $x(n+N)$ have the same distribution for every integer N. Stationary signals are associated with steady states in the system producing the signal, and are characterized by constant averages and variances along the signal. When statistics computed along any instance of a signal are identical to those computed across ensembles of different instances of the signal, the system is *ergodic*. Ergodicity is assumed in most practical situations.

Matlab has functions for computing the mean (*mean*), variance (*var*), standard deviation (*std*), covariance (*cov*), and correlation coefficient (*corrcoeff*) of signal values sampled across time or across ensembles. The Statistics Toolbox provides many additional statistical functions.

The Signal Processing Toolbox provides a *crosscorrelation* function (*xcorr*) and a *crossco-*

variance function (*xcov*). Crosscorrelation of two signals is equivalent to the convolution of the two signals with one of them reversed in time. The *xcorr* function adds several options to the standard Matlab *conv* function. The *xcov* function simply subtracts the mean of the inputs before computing the crosscorrelation. The crosscorrelation of a signal with itself is called its *autocorrelation*. It measures the *coherence* of a signal with respect to time shifts.

Try:

```
x = randn(1,100);
w = 10;
y = conv(ones(1,w)/w,x);
avgs = y(10:99);
plot(avgs)
```

Ensemble averages:

```
w = 10;
for i = 1:w;
X(i,:)  = randn(1,100);
end
AVGS = mean(X);
plot(AVGS)
```

```
x = [-1 0 1];
y = [0 1 2];
xcorr(x,y)
conv(x,fliplr(y))
xcov(x,y)
xcov(x,x)
xcov(x,y-1)
```

Q1: What do you notice about these vectors?
Explain why you see these results.

Example: Crosscorrelation

In a simple target ranging system, an outgoing signal x is compared with a returning signal y. We can model y by

$$y(n) = \alpha x(n-d) + \beta$$

where α is an attenuation factor, d is a time delay, and β is channel noise. If T is the return time for the signal, then x and y should be correlated at $n = T$. The target will be located at a distance of vT, where v is the channel speed of the signal.

Try:

```
x = [zeros(1,25),1,zeros(1,25)];
subplot(311), stem(x)
y = 0.75*[zeros(1,20),x] + 0.1*randn(1,71);
subplot(312), stem(y)
[c lags] = xcorr(x,y);
subplot(313), stem(lags,c)
```

Q2: Does this example show the expected behavior?
Why or why not?

Discrete Fourier Transform (DFT)

It is often useful to deompose data into component frequencies. *Spectral analysis* gives an alternative view of time or space-based data in the *frequency domain*. The computational basis of spectral analysis is the *discrete Fourier transform* (DFT).

The DFT of a vector y of length n is another vector Y of length n:

$$Y_{k+1} = \sum_{j=0}^{n-1} \omega^{jk} y_{j+1}$$

where ω is a complex n^{th} root of unity:

$$\omega = e^{-2\pi i/n}$$

This notation uses i for the complex unit, and j and k for indices that run from 0 to $n-1$. The subscripts $j+1$ and $k+1$ run from 1 to n, corresponding to the range usually associated with Matlab vectors.

Data in the vector y are assumed to be separated by a constant interval in time or space $dt = 1/Fs$. Fs is called the *sampling frequency* of y. The coefficient Y_{k+1} measures the amount of the frequency $f = k(Fs/n)$ that is present in the data in y. The vector Y is called the *spectrum* of y.

The midpoint of Y (or the point just to the right of the midpoint, if n is even), corresponding to the frequency $f = Fs/2$, is called the *Nyquist point*. The real part of the DFT is symmetric about the Nyquist point.

The graphical user interface `fftgui` allows you to explore properties of the DFT. If y is a vector,