E E 2 8 9 Lab January 15, 2012

Lab 3. Spectral Analysis in Matlab - Part II

Introduction

This lab covers the following topics. Corresponding exercises and questions to be answered are contained in the sections.
1. **Power Spectral Density (PSD) Estimation**
2. **Spectral Methods - Summary & Exercises**
3. **Time-varying Spectra**
4. **Wavelets**

1 Power Spectral Density (PSD) Estimation

This section gives a bit more technical description of spectral analysis. For a deterministic signal, there are various methods for determining its spectrum. These are described in the following sections. Along with the spectrum, another important property of **deterministic** signals $x(n), 0 \leq n \leq N-1$ is the autocorrelation function (ACF) that is correlation with itself. It is defined as,

$$R_{xx}(m) = \sum_{n=0}^{N-m-1} x(n)x(n+m)$$

Note that the above definition is the raw unnormalized cross correlation. In the Matlab function *xcorr(x,y, 'option')*, the option allows various normalizations (biased, unbiased, etc.,) that we will not consider at this point. When the signal that we have is a random process, then the matter of determining spectra becomes a bit more subtle. Recall that the ACF of a discrete-time random process $X(n, \xi)$ is the *deterministic* function,

$$R_{XX}(m) = E\{x(n)x(n+m)\} \tag{1}$$

where $x(n)$ is a realization of the random process $X(n, \xi)$. Hence we have,

$$R_{XX}(0) = E\{x(n)x(n)\}$$

where E is the expectation operator. Hence the right hand side is the *expected* i.e. *average* value of the square of the sequence $x(n)$. This leads to the characterization of $R_{XX}(0)$ as being the *average power* of the random process. (In the characterization above, we have of course assumed that we have a real, stationary random process.)

Now, since $R_{XX}(m)$ is a deterministic sequence, we can take its Fourier transform (actually the Discrete-time Fourier Transform, DTFT - we will see this later on in class) which is defined as,

$$S_{XX}(\omega) = \sum_{m=-\infty}^{+\infty} R_{XX}(m)e^{-j\omega m}. \tag{2}$$

Then $R_{XX}(m)$ is the inverse DTFT of $S_{XX}(\omega)$ which is given by,

$$R_{XX}(m) = \frac{1}{2\pi}\int_{-\pi}^{+\pi} S_{XX}(\omega)e^{j\omega m}d\omega.$$

This leads to,

$$R_{XX}(0) = \frac{1}{2\pi}\int_{-\pi}^{+\pi} S_{XX}(\omega)d\omega$$

which leads to the interpretation that $S_{XX}(\omega)$ is the **power spectral density** (PSD) of the random process since $R_{XX}(0)$ is average power. Hence the PSD of a random process describes the distribution of its power with frequency. It may be measured in units of watts/Hz or dB/Hz. So basically, **PSD describes how the average power of a signal is distributed with frequency**. And total power in a specific frequency band can be calculated by integration (summation) over the frequency band.

We note from equation(2) that the PSD depends on an infinite number of observations of $R_{XX}(m)$. Hence determination of PSD is an impossible task. Accordingly, we try to determine an *estimate* of the PSD. Furthermore, $R_{XX}(m)$ in equation(1) is defined as the expectation of $x(n)x(n+m)$, that is, an *ensemble average*. Not only does this require an ensemble of realizations but also a knowledge of the joint pdf, both of which are generally impossible tasks. Accordingly, we aim to form an estimate $R_{XX}(m)$ to permit an estimate of $S_{XX}(m)$.

Typically, we only have *one* realization of a random process and that too for a finite duration. Hence the goal is to form an estimate of the ACF from the single finite duration realization of the random process. Under certain conditions [3] the *temporal* ACF can be shown to converge to the true ensemble ACF. That is,

$$Lim_{M\rightarrow\infty}\frac{1}{2M+1}\sum_{n=-2M}^{2M} x(n)x(n+k) = E(x(n)x(n+k)) = R_{XX}(k) \quad (3)$$

Random processes for which this holds are said to be ergodic, or more accurately second-order ergodic. Proving ergodicity is not an easy task, and more often that not, it is just assumed.

1.1 Alternative definition of PSD

It can be shown [3] that a nearly equivalent definition of the PSD, equation (2) is,

$$S_{XX}(\omega) = Lim_{M\rightarrow+\infty}E\{\frac{1}{2M+1}|\sum_{n=-M}^{M} x(n)e^{-j\omega n}|^2\} \quad (4)$$

This says that you first take the magnitiude squared of the FT (DTFT) of the data {x(n)} and then divide by the data record length. Then, since the Fourier Transform will be a random variable (each new realization x(n) will be different), the expected value is taken. Finally, since the random process is generally of infinite duration, the limit is taken. Equality of equations (2) and (4) is established in [3], (p.59).

1.2 Classical Techniques of Spectral Estimation

The two most common classical spectral (that is, power spectral density) estimation techniques are the *Periodogram* and *Blackman-Tukey* methods. These are based on Fourier analysis (a non-parametric method) as opposed to so-called modern spectral estimation methods which are parametric methods. The latter are discussed in the following sections. The fundamental distinguishing feature of the classical method is the *bias-variance* trade off. Bias occurs when the mean value of an estimator is different to the true value of the variable being estimated. Variance is the expected square deviation from its mean value. A brief description of the two classical methods is given here. Detailed description may be found in many texts, such as [3].

1.2.1 Periodogram

The periodogram spectral estimator stems from the PSD already defined in equation (4),

$$S_{XX}(\omega) = Lim_{M\to+\infty} E\{\frac{1}{2M+1}|\sum_{n=-M}^{M} x(n)e^{-j\omega n}|^2\}.$$

Removing the expectation operator and using only a finite data size, the **periodogram spectral estimator** is defined as,

$$\hat{S}_{XX}(\omega) = \frac{1}{N}|\sum_{n=0}^{N-1} x(n)e^{-j\omega n}|^2 \tag{5}$$

which of course is the DTFT squared and averaged. The immediate question is whether the periodogram estimator $S_{XX}(\omega)$ equation (5) is a *consistent estimator* of the PSD equation (4), that is: Does $\hat{S}_{XX}(\omega) \to S_{XX}(\omega), N \to \infty$? The fact is that it is not [3], p.66. However, for $N \to \infty$, the estimator is unbiased although the variance does not approach zero.

1.2.2 Averaged Periodogram

The variance in the periodogram estimator of equation (5) (not decreasing with increasing data record length) may be due to the removal of the expectation operator, that is, a lack of averaging. To improve the statistical properties of the the periodogram, the expectation operator can be approximated by averaging a set of periodograms. Assume that K independent records of the random process are available for the interval $0 \le n \le L-1$. Then the averaged periodogram estimator is defined as,

$$\hat{S}_{AVPER}(\omega) = \frac{1}{K}\sum_{m=0}^{K-1} \hat{S}_{PER}^{(m)}(\omega) \tag{6}$$

where $\hat{S}_{PER}^{(m)}(\omega)$ is the periodogram for the mth data set defined by,

$$\hat{S}_{PER}^{(m)}(\omega) = \frac{1}{L}|\sum_{n=0}^{L-1} x_m(n)e^{-j\omega n}|^2 \tag{7}$$

This is physically meaningful. Were $x_m(n)$ for $m = 0, 1, 2 \ldots$ realizations of a random process, all the DTFTs would be different. Hence we look at the average of the Fourier Transform magnitudes, or more correctly, at the average magnitude squared of the Fourier Transform (taken hopefully over along time intervals).

The averaging results in a reduction of the variance by a factor of K (recall that we have K independent data sets). In practice, we seldom have independent data sets but rather, only one data record of length N. And we need to base the estimator on this data set. A standard approach is to segment the data into K nonoverlapping blocks of length L, where $N = KL$. Accordingly, for equation (4), one has data sets,

$$x_m(n) = x(n+mL), \qquad n = 0, 1, \ldots, L-1; \quad m = 0, 1, \ldots, K-1$$

The associated problem here might be that, given the blocks are contiguous, they may not be uncorrelated let alone independent. (Only white noise would have that property). Accordingly, the variance reduction would be less than that for independent records.

1.2.3 Welch method

The Wlech method provides a variation of the averaged periodogram. Recall that the periodogram is just the average, magnitude squared of the DFT of the signal. The Welch method differs from the averaged periodogram in two ways. First, the data is windowed and secondly, data blocks are overlapped. The data window reduces spectral leakage (we see this later) and by overlapping blocks of data, typically by 50 or 75%, some extra variance reduction is achieved.

The Welch PSD estimator uses the DFT. However, it does it in a clever way (overlapping samples) such that the PSD estimates, depending on the data set, *are often better than those obtained with the DFT.* Here is a very simple example that illustrates this fact.

Do the following:

1. In the command window invoke *psddemo*.
2. In the GUI, look at Narrowband+Noise, length=100, SNR in dB =10. Number of trials =1.
3. Use the Periodogram method to observe the PSD Estimate.
4. Now use the Welch method, overlap =25, Hamming window, length=50. (We will discuss all this in class later). Observe spectrum.
5. Now increase the number of trials to 10 and do not check Average of realizations. See the number of realizations of the random process. You can now check Average of realizations and observe the averaging process.

Q1. Comment on the 2 PSD estimates - Periodogram and Welch. What are the units for the PSD? What are the units for the abscissa? Comment on these units.

1.3 Blackman-Tukey Estimation

The periodogram equation (5) is in general a poor estimator of the PSD. This may be seen as follows: The equivalent form of equation (5) can be shown to be,

$$\hat{S}_{PER}(\omega) = \sum_{k=-N-1}^{N-1} \hat{r}_{xx}(k) e^{-j\omega k} \tag{8}$$

where,

$$\hat{r}_{xx}(k) = \begin{cases} \frac{1}{N}\sum_{n=0}^{N-1-k} x(n)x(n+k), & k = 0, 1, \ldots, N-1 \\ \hat{r}_{xx}(-k), & k = -(N-1), -(N-2), \ldots, -1 \end{cases} \tag{9}$$

The periodogram given by equation (8) depends on the autocorrelation function $r_{xx}(k)$ which is *estimated* by $\hat{r}_{xx}(k)$ of equation (9). Accordingly it is not surprising to obtain a poor estimator of the PSD, given that it depends on an estimate of the autocorrelation function $r_{xx}(k)$, which estimate itself may be poor. The Blackman-Tukey (BT) spectral estimator improves on the performance of the periodogram by weighing the estimate $\hat{r}_{xx}(k)$ less at higher lags (where it performs poorly). Accordingly, the BT spectral estimator is defined as,

$$\hat{P}_{BT}(\omega) = \sum_{k=-(N-1)}^{N-1} w(k)\hat{r}_{xx}(k)e^{-j\omega k} \tag{10}$$

where the window $w(k)$ weighs the higher lags less (see [3], p.80). Note that it is equal to the periodogram if $w(k) = 1$, for all k. Again, bias-variance trade-off can be observed here depending on $w(k)$.

2 Spectral Methods - Summary & Excercises

PSD estimates of noisy analog signals from a finite number of its samples are based on three fundamentally different approaches:

• **Non-parametric methods**
Make no assumptions about the data in the sample and work directly with the DFT.
Welch: `pwelch`
Multitaper: `pmtm`

• **Parametric methods**
Model the data in the sample as the output of a linear system excited by white noise (noise with zero mean and constant PSD), estimate the filter coefficients, and use these to estimate the PSD.
Burg: `pburg`
Yule-Walker: `pyulear`

• **Subspace methods**
Based on an eigenanalysis or eigendecomposition of the correlation matrx associated with the data in the sample.
EV: `peig`
MUSIC: `pmusic`

Try:

```
Fs = 100;
t = 0:1/Fs:10;
y = sin(2*pi*15*t) + sin(2*pi*30*t);
nfft = 512;
Y = fft(y,nfft);
f = Fs*(0:nfft-1)/nfft;
Power = Y.*conj(Y)/nfft;
plot(f,Power)
title('Periodogram')
```

Q2. Write the formula for obtaining the PSD using the periodogram, as described above.
What are the units on the ordinate and abscissa axis?

```
figure
ryy = xcorr(y,y);
Ryy = fft(ryy,512);
plot(f, abs(Ryy))
title('DFT of Autocorrelation')
```

Q3. Write the formula for obtaining the PSD using autocorealtion, as described above. Comment on the difference between PSDs using the periodogram method and using the autocorrelation function.

2.1 Non-parametric Methods

(*You can skip this subsection if you wish and go onto to Section 2.2.*).

As described earlier, non-parametric methods estimate the PSD directly from the signal itself. The simplest such method is the periodogram. An improved version of the periodogram is Welch's method. A more modern technique is the multitaper method (MTM).

The following functions estimate the PSD Pxx in units of power per radians per sample. The corresponding vector of frequencies w is computed in radians per sample, and has the same length as Pxx.

• **Periodogram method**

```
[Pxx w] = periodogram(x)
```

Estimates the PSD using a periodogram. Optional inputs specify windows (default is rectangular), FFT length, PSD sample frequencies, and output frequency range.

• **Welch method**

```
[Pxx w] = pwelch(x)
```

Estimates the PSD using Welch's averaged periodogram method. The vector x is segmented into equal-length sections with overlap. Trailing entries not included in the final segment are discarded. Each segment is windowed. Optional inputs specify windows (default is Hamming), overlap, FFT length, and PSD sample frequencies.

• **Multitaper method**

```
[Pxx w] = pmtm(x, nw)
```

Estimates the PSD using a sequence of $2 * nw - 1$ orthogonal tapers (windows in the frequency domain). The quantity nw is the time-bandwidth product for the discrete prolate spheroidal sequences specifying the tapers. Optional inputs specify taper frequencies, FFT length, and PSD sample frequencies.

Try:

```
t = 0:1/100:10-1/100;
x = sin(2*pi*15*t) + sin(2*pi*30*t);
periodogram(x,[],512,100);
figure
pwelch(x,[],512,100);
figure
pmtm(x,[],512,100);
```

The above three estimator examples are only for your observation. No comments are necessary. Only the observations that the DFT can be applied in many ways in spectral estimation.

2.2 Parametric Methods

Parametric methods can yield **higher resolution than non-parametric methods in cases where the signal length is short.** These methods use a different approach to spectral estimation: instead of estimating the PSD directly from the data, they model the data as the output of a linear system driven by white noise and then attempt to estimate the parameters of that linear system.

The most commonly used linear system model is the all-pole model, a system with all of its zeros at the origin in the z-plane. The output of such a system for white noise input is an autoregressive (AR) process. These methods are sometimes referred to as *AR methods*.

AR methods give accurate spectra for data that is "peaky," that is, data with a large PSD at certain frequencies. The data in many practical applications (such as speech) tends to have peaky spectra, so that AR models are used therel. In addition, the AR models lead to a system of linear equations that is relatively simple to solve for the parameters of the unknown system.

The following methods are summarized on the next page. The input p specifies the order of the autoregressive (AR) prediction model.

• **Yule-Walker AR method**

```
[Pxx f] = pyulear(x,p,nfft,fs)
```

• **Burg method**

```
[Pxx f] = pburg(x,p,nfft,fs)
```

• **Covariance and modified covariance methods**

```
[Pxx f] = pcov(x,p,nfft,fs)
[Pxx f] = pmcov(x,p,nfft,fs)
```

Example: Here is an example of using the AR parametric method to estimate the PSD of the signal.

Because the Yule-walker method estimates the spectral density by fitting an AR prediction model of a given order to the signal, first generate a signal from an AR (all-pole) model of a given order. You can use freqz to check the magnitude of the frequency response of your AR filter. This will give you an idea of what to expect when you estimate the PSD using pyulear:

```
% AR filter coefficients
a = [1 -2.2137 2.9403 -2.1697 0.9606];
```

This is the Autoregressive filter

$$H(z) = \frac{1}{1 - 2.2137z^{-1} + 2.9403z^{-2} - 2.1697z^{-3}0.9606z^{-4}}$$

(We will cover z-transforms in class soon.)

The following function generates the frequency response of $H(z)$:

```
freqz(1,a)
title('AR System Frequency Response')
```

Now we generate the input signal x by filtering white noise through the AR filter.

```
x = filter(1,a,randn(256,1));    % AR system output
```

So, given the noisy data, how do you find its spectrum, or particularly, its PSD? First find PSD using non-parametric method - the periodogram.

```
X=fft(x);
Power = X.*conj(X)/256;
plot(Power)
```

Then we use an algorithm called *pyulear* to estimate the PSD of x based on a fourth-order AR prediction model (since in this case, we know that the original AR system model a has order 4).

```
%randn('state',1);
x = filter(1,a,randn(256,1));    % AR system output
pyulear(x,4)                     % Fourth-order estimate

The spectral estimate returned by pyulear is the squared magnitude of the
frequency response of this AR model.
```

Q4. Explain, very briefly, the AR method of spectral estimation. Comment on the two ways of determining the PSD.

The exercises below are only for your review. No questions need to be answered. You can go to **Section 3** for **Q5**.

Try:

```
edit pmethods
pmethods('pyulear',25,1024)
pmethods('pburg',25,1024)
pmethods('pcov',5,512)
pmethods('pmcov',5,512)
```

Some of the factors to consider when choosing among parametric methods are summarized in the following table. See the documentation for further details.

Burg	Covariance	Modified Covariance	Yule-Walker
Characteristics			
Does not apply window to data	Does not apply window to data	Does not apply window to data	Applies window to data
Minimizes forward and backward prediction errors	Minimizes forward prediction error	Minimizes forward and backward prediction errors	Minimizes forward prediction error
Advantages			

High resolution for short data records	Better resolution than Y-W for short data records	High resolution for short data records	As good as other methods for large data records
Always produces a stable model	Extract frequencies from mix of p or more sinusoids	Extract frequencies from mix of p or more sinusoids	Always produces a stable model
		No spectral line-splitting	
Disadvantages			
Peak locations highly dependent on initial phase	Can produce unstable models	Can produce unstable models	Performs relatively poorly for short data records
Spectral line-splitting for sinusoids in noise, or when order is very large	Frequency bias for estimates of sinusoids in noise	Peak locations slightly dependent on initial phase	Frequency bias for estimates of sinusiods in noise
Frequency bias for sinusoids in noise		Minor frequency bias for sinusiods in noise	
Conditions for Nonsingularity			
	p must be $\leq 1/2$ input frame size	p must be $\leq 2/3$ input frame size	Autocorrelation matrix always positive-definite, nonsingular

Subspace Methods

Subspace methods, also known as *high-resolution methods* or *super-resolution methods*, generate PSD estimates based on an eigenanalysis or eigendecomposition of the correlation matrix. These methods are best suited for *line spectra* - i.e., spectra of sinusoidal signals - and are effective for detecting sinusoids buried in noise, especially when signal to noise ratios are low.

The following functions estimate the *pseudospectrum* S (an indicateor of the presence of sinusoidal components in a signal) of the input signal x, and a vector w of normalized frequencies (in rad/sample) at which the pseudospectrum is evealuated. The input p controls the dimensions of the signal and noise subspaces used by the algorithms.

• **Eigenvector method**
```
[S f] = peig(x,p,nfft,fs)
```

• **Multiple Signal Classification (MUSIC) method**
```
[S f] = pmusic(x,p,nfft,fs)
```

The MUSIC algorithm uses Schmidt's eigenspace analysis method. The eigenvector uses a weighted version of the MUSIC algorithm.

Try:

```
edit ssmethod
ssmethod(3)
ssmethod(4)
```

Spectrum Viewer in SPTool

The SPTool allows you to view and analyze spectra using different methods. To create a spectrum in SPTool,

1. Select the signal in the **Signals** list in SPTool.
2. Select the **Create** button under the **Spectra** list.

Use the **View** button under the **Spectra** list in the SPTool GUI to display one or more selected spectra.

Try:

```
t = 0:1/100:10-1/100;
x = sin(2*pi*15*t) + sin(2*pi*30*t);
```

Import this signal into SPTool and view the spectrum using various methods.

3 Time-Varying Spectra

The spectral estimation methods described so far are designed for the analysis of signals with a constant spectrum over time. In order to find time-varying spectra, different methods of analysis and visualization must be used.

The *time-dependent Fourier transform* of a signal is a sequence of DFTs computed using a sliding window. A *spectrogram* is a plot of its magnitude versus time.

```
[B f t] = specgram(x,nfft,fs,window,numoverlap)
```

calculates the time-dependent Fourier transform for the signal in vector x and returns the DFT values B, the frequency vectors f, and the time vectors t. The spectrogram is computed as follows:

1. The signal is split into overlapping sections and applies to the window specified by the *window* parameter to each section.

2. It computes the discrete-time Fourier transform of each section with a length $nfft$ FFT to produce an estimate of the short-term frequency content of the signal; these transforms make up the columns of B. The quantity $length(window) - numoverlap$ specifies by how many samples *specgram* shifts the window.

3. For real input, *specgram* truncates to the first $nfft/2 + 1$ points for $nfft$ even and $(nfft + 1)/2$ for $nfft$ odd.

When called with no outputs, *specgram* displays the spectrogram.

Try:

Time-constant spectrum

```
t = 0:1/100:10-1/100;
x = sin(2*pi*15*t) + sin(2*pi*30*t);
specgram(x,256,100,hann(20),10)
colorbar
```

Time-varying spectrum

```
load handel
soundsc(y,Fs)
specgram(y,512,Fs,kaiser(100,5),75)
colorbar
```

Q5. Comment on the spectrogram for time-constant ("stationary") spectra and time-varying ("non-stationary") spectra.

Spectrogram Demos

```
specgramdemo
xpsound
```

Example: Reduced Sampling Rate

This example compares the sound and spectrogram of a speech signal sampled at progressively reduced rates.

If you resample a signal at a fraction of the original sampling frequency, part of the signal's original frequency content is lost. The down-sampled signal will contain only those frequencies less than the new Nyquist frequency. As down-sampling continues, the words in the signal remain recognizable long after the original spectrogram has become obscured. This is a tribute to the human auditory system, and is the basis of signal compression algorithms used in communications.

Try the following. Just type HAL9000, do not hit the return button and sit back and listen to HAL winding down.

```
edit HAL9000
HAL9000
```

Q6. How is HAL's speech slowed?

4 Wavelets

One of the drawbacks of the Fourier transform is that it captures frequency information about a signal without any reference to time. For stationary signals this is unimportant, but for time-varying or "bursty" signals, time can be critical to an analysis.

The time-dependent Fourier transform computed by the *specgram* function is one solution to this problem. By applying a DFT to a sliding window in the time domain, *specgram* captures a signal's frequency content at different times. The disadvantage of this method is that it is *uniform* on all time intervals: it does not adjust to local idiosyncrasies in the frequency content. As a result, more subtle nonstationary characteristics of a signal can go undetected.

Wavelet analysis uses a more adaptable method, and is capable of revealing trends, breakdown points, discontinuities in higher derivatives, and self-similarity that the DFT might miss.

A wavelet is a waveform of effectively limited duration that has an average value of zero.

The *discrete wavelet transform* (DWT) computes *coefficients of similarity* between a signal and a sliding wavelet. The coefficients are found with wavelets of different *scales* (widths that approximate different frequencies) to analyze the signal at different resolutions. In a wavelet analysis, a signal is iteratively decomposed into the sum of a lowpass *approximation* and a progressive sequence of highpass *details*.

The Wavelet Toolbox adds wavelet analysis techniques to the signal processing techniques available in the Signal Processing Toolbox.

```
wavemenu
```

brings up a menu for accessing graphical tools in the Wavelet Toolbox.

Try:

```
wavemenu
```

Select **Wavelet 1-D**
and then **File → Example Analysis → Basic Signals → Frequency breakdown**

S is the signal
a_5 is the approximation
d_n are the details

What is happening in this signal and how does the wavelet analysis show it?

References

[1] Naidu, Prabhakar S. *Modern Spectrum Analysis of Time Series.* Florida: CRC Press, 1996.

[2] Naidu, Prabhakar S. *Modern Digital Signal Processing: An Introduction.* Alpha Science International Ltd, 2003.

[3] Kay, Steven M. *Modern Spectral Estimation: Theory & Application.* Prentice Hall Signal Processing Series, 1987.

(The material derives mostly from [3] and also from the MathWorks training document "MATLAB for Signal Processing", 2006.)