DE GRUYTER OPEN

JACSM 2014, Vol. 6, No. 1, pp. 5 - 26

DOI 10.2478/jacsm-2014-0001

MINIMUM UTILIZATION OF ELECTROMAGNETIC (2G, 3G, 4G) SPECTRUM IN SEAMLESS MOBILITY BASED ON VARIOUS ESTIMATION METHODS

A. Jayanthila Devi[1], Dr. G. M. Kadhar Nawaz[2]

[1]Assistant Professor/MCA,Jain University, Bangalore
jayanthilamca@gmail.com

[2]Director/M CA,Sona College of Technology, Salem
nawazse@yahoo.co.in

Abstract

Wireless Cellular Seamless Mobility is the most important feature of a wireless cellular communication system. Basically cell phones are used for communication purpose and therefore good cell coverage is needed to make and receive calls. Usually, continuous service is achieved by supporting handover from one cell to another. Handover is the process of changing the channel associated with the current connection while a call is in progress. It is often initiated either by crossing a cell boundary or by deterioration in quality of the signal in the current channel. Mobile IP, too, has to support redirection of data to a new foreign agent after the change of network access. In this case the additional delay caused by the redirection. The shorter the interruption of the service is –up to the ideal case of no service interruption. Minimizing the Delay based on some prior estimation methods in handover is better for quality of seamless connectivity anytime; anywhere across communication networks. All mobile phone system supports the seamless handover between base stations. To avoid the additional delay and wastage of Electromagnetic Spectrum, Mobile IP may use Optimizations such as rerouting of the whole packet in the flow of efficiently transmission of data with short interrupted service by Time series based Spectrum Estimation Technique. Due to the utilization of seamless handover mechanism it reduces the unwanted usage of Spectrum.

Key words: Estimation, Frequency, Handover, Seamless, Spectrum, Time

1 Introduction

The Electromagnetic Spectrum is the range of frequencies of possible electromagnetic radiation. The Spectrum ranges from 0 Hertz up to 2.4x1023 Hertz. The exact wavelength limits of the Spectrum are unknown however it is widely believed that the short wavelength limit is equal to the Planck

Length (1.616x10-35m) and the long wavelength limit is the length of the Universe.[1]. The frequencies of electromagnetic radiation can be calculated by dividing the speed of light by the wavelength of the electromagnetic radiation. Regions of the Electromagnetic Spectrum have been named by scientists to provide an easier way to remember and refer to the ranges; however, in reality neighbouring types of electromagnetic energy often overlap. The goal of Spectrum Estimation is to describe the distribution (over frequency) of the contained in a signal, based on a finite set of data. Estimation of spectrum is useful in a variety of applications, including the detection of signals buried in wideband noise.

2 Spectrum Estimation

The Spectrum Estimation of a stationary random process xn is mathematically related to the autocorrelation sequence by the discrete-time Simulation. In terms of normalized frequency, this is given by

wavelength × frequency = the speed of light
or

$$\lambda \times f = c$$

In this equation, the Greek letter "lambda" (λ) is used as shorthand for the wavelength and the fancy "f" (f) is used to represent the frequency; "c" is the speed of light (186,000 miles per second or 300 million meters per second). Since the speed of light is constant, the wavelength and frequency are limited; if one is big the other has to be small.[2]. That is why large (high) frequencies correspond to small wavelengths and large wavelengths correspond to small (low) frequencies.To convert from frequency (f) to wavelength (8) and vice versa, recall that f = c/8, or 8 = c/f; where c = speed of light. Rules follow:

Metric:

Wavelength in cm = 30 / frequency in GHz
For example: at 10 GHz, the wavelength = 30/10 = 3 cm
Wavelength in ft = 1 / frequency in GHz
For example: at 10 GHz, the wavelength = 1/10 = 0.1 ft

$$P_{xx}(\omega) = \frac{1}{2\pi} \sum_{m=-\infty}^{\infty} R_{xx}(m) e^{-j\omega m}$$

This can be written as a function of physical frequency f (e.g., in hertz) by using the relation $\omega = 2\pi f/f_s$, where f_s is the sampling frequency.

$$P_{xx}(f) = \frac{1}{f_s}\sum_{m=-\infty}^{\infty} R_{xx}(m)e^{-j2\pi mf/f_s}$$

The correlation sequence can be derived from the SE by use of the inverse discrete-time Simulation:

$$R_{xx}(m) = \int_{-\pi}^{\pi} P_{xx}(\omega)e^{j\omega m}d\omega = \int_{-f_s/2}^{f_s/2} P_{xx}(f)e^{j2\pi mf/f_s}df$$

The average of the sequence x_n over the entire interval is represented by

$$R_{xx}(0) = \int_{-\pi}^{\pi} P_{xx}(\omega)d\omega = \int_{-f_s/2}^{f_s/2} P_{xx}(f)df$$

The average of a signal over a particular frequency band $[\omega_1, \omega_2]$, $0\le\omega_1\le\omega_2\le\pi$, can be found by integrating the SE over that band:

$$\bar{\bar{P}}_{[\omega_1,\omega_2]} = \int_{\omega_1}^{\omega_2} P_{xx}(\omega)d\omega = \int_{-\omega_2}^{-\omega_1} P_{xx}(\omega)d\omega$$

You can see from the above expression that $P_{xx}(\omega)$ represents the content of a signal in an *infinitesimal* frequency band, which is why it is called the spectrum *Estimation*. The units of the SE are (e.g., watts) per unit of frequency. In the case of $P_{xx}(\omega)$, this is watts/radian/sample or simply watts/radian. In the case of $P_{xx}(f)$, the units are watts/hertz. Integration of the SE with respect to frequency yields units of watts, as expected for the average .For real–valued signals, the SE is symmetric about DC, and thus $P_{xx}(\omega)$ for $0\le\omega\le\pi$ is sufficient to completely characterize the SE. However, to obtain the average over the entire Nyquist interval, it is necessary to introduce the concept of the *one-sided* SE.The one-sided SE is given by

$$P_{\text{onesided}}(\omega) = \begin{cases} 0 & -\pi \le \omega < 0 \\ 2P_{xx}(\omega) & 0 \le \omega \le \pi \end{cases}$$

The average of a signal over the frequency band, $[\omega_1, \omega_2]$ with $0\le\omega_1\le\omega_2\le\pi$, can be computed using the one-sided SE as

$$\bar{P}_{[\omega_1,\omega_2]} = \int_{\omega_1}^{\omega_2} P_{\text{onesided}}(\omega)d\omega$$

3 Spectrum Estimation Method

The various methods of estimation available are categorized as follows: In general terms, one way of estimating the SE of a process is to simply find the discrete-time series of the samples of the process (usually done on a grid with an FFT) and appropriately scale the magnitude squared of the result. This estimate is called the *timeseries*.The time series estimate of the SE of a length-L signal $x_L[n]$ is

$$P_{xx}(f) = \frac{1}{LF_s} \left| \sum_{n=0}^{L-1} x_L(n) e^{-j2\pi fn/F_s} \right|^2$$

Where F_s is the sampling frequency. In practice, the actual computation of $P_{xx}(f)$ can be performed only at a finite number of frequency points, and usually employs an FFT. Most implementations of the time series method compute the *N*-point SE estimate at the frequencies.

$$f_k = \frac{kF_s}{N} \quad k = 0,1,\ldots,N-1$$

In some cases, the computation of the time series via an FFT algorithm is more efficient if the number of frequencies is a of two. Therefore it is not uncommon to pad the input signal with zeros to extend its length to a of two. As an example of the time series, consider the following 1001-element signal xn, which consists of two sinusoids plus noise:

```
fs = 1000;              % Sampling frequency
t = (0:fs)/fs;          % One second worth of samples
A = [1 2];              % Sinusoid amplitudes (row vector)
f = [150;140];          % Sinusoid frequencies (column vector)
xn = A*sin(2*pi*f*t) + 0.1*randn(size(t));
```

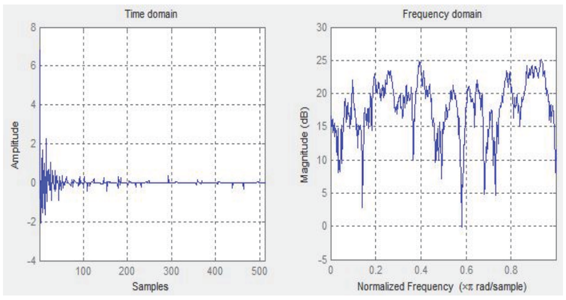

Figure 1. Time and Frequency Domian Estimation samples

The timeseries estimate of the SE can be computed using timeseries. In this case, the data vector is multiplied by a Hamming window to produce a modified timeseries.[3].

[Pxx,F] = time series(xn,hamming(length(xn)),length(xn),fs);
*plot(F,10*log10(Pxx))*
xlabel('Hz'); ylabel('dB');
title('Modified Timeseries Spectrum Estimation Estimate');

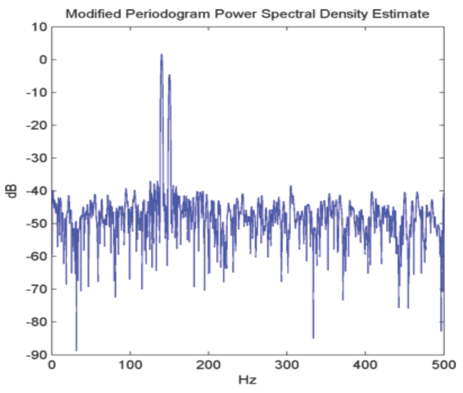

Figure 2. Performance of the Time series

The following sections discuss the performance of the timeseries with regard to the issues of leakage, resolution, bias, and variance.

3.1 S*pectrum* Leakage:

Consider the SE of a finite-length (length L) signal $x_L[n]$, as discussed in the Timeseries section. It is frequently useful to interpret $x_L[n]$ as the result of multiplying an infinite signal, $x[n]$, by a finite-length rectangular window, $w_R[n]$:

$$x_L[n] = x[n]w_R[n].$$

Because multiplication in the time domain corresponds to convolution in the frequency domain, the expected value of the timeseries in the frequency domain is:

$$E\{P_{xx}(f)\} = \frac{1}{Fs}\int_{-Fs/2}^{Fs/2} \frac{\sin^2(L\pi(f-f')/Fs)}{L\sin^2(\pi(f-f')/Fs)} P_{xx}(f')df',$$

showing that the expected value of the timeseries is the convolution of the true SE with the square of the Dirichlet kernel.The effect of the convolution is best understood for sinusoidal data. Suppose that $x[n]$ is composed of a sum of M complex sinusoids:

$$x(n) = \sum_{k=1}^{N} A_k e^{j\omega_k n}$$

Its is

$$X(\omega) = \sum_{k=1}^{N} A_k \delta(\omega - \omega_k)$$

which for a finite-length sequence becomes

$$X(\omega) = \int_{-\pi}^{\pi} (\sum_{k=1}^{N} A_k \delta(\ -\omega_k))W_R(\omega - \ )d = = \backslash = '$$

The preceding equation is equal to:

$$X(\omega) = \sum_{k=1}^{N} A_k W_R(\omega - \omega_k)$$

So in the of the finite-length signal, the Dirac deltas have been replaced by terms of the form $W_R(\omega-\omega_k)$, which corresponds to the frequency response of a rectangular window centered on the frequency ω_k.The frequency response of a rectangular window has the shape of a sinc signal, as shown below.

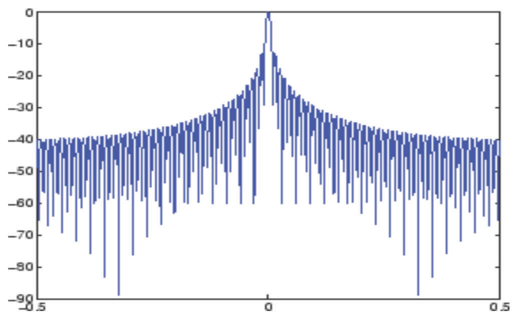

Figure 3. Variance of the Timeseries Spectrum Leakage.

The plot displays a main lobe and several side lobes, the largest of which is approximately 13.5 dB below the main lobe peak. These lobes account for the effect known as *spectrum leakage*. While the infinite-length signal has its concentrated exactly at the discrete frequency points f_k, the windowed (or truncated) signal has a continuum of "leaked" around the discrete frequency points f_k.Because the frequency response of a short rectangular window is a much poorer approximation to the Dirac delta function than that of a longer window, spectrum leakage is especially evident when data records are short. Consider the following sequence of 100 samples:

```
fs = 1000;            % Sampling frequency
t = (0:fs/10)/fs;     % One-tenth second worth of samples
A = [1 2];            % Sinusoid amplitudes
f = [150;140];        % Sinusoid frequencies
xn = A*sin(2*pi*f*t) + 0.1*randn(size(t));
[Pxx,F] = timeseries(xn,rectwin(length(xn)),length(xn),fs);
plot(F,10*log10(Pxx))
```

It is important to note that the effect of spectrum leakage is contingent solely on the length of the data record. It is *not* a consequence of the fact that the time series is computed at a finite number of frequency samples.

3.2 Resolution:

Resolution refers to the ability to discriminate spectrum features, and is a key concept on the analysis of spectrum estimator performance.In order to resolve two sinusoids that are relatively close together in frequency, it is necessary for the difference between the two frequencies to be greater than the width of the main lobe of the leaked spectra for either one of these sinusoids. The mainlobe width is defined to be the width of the main lobe at the point where the is half the peak main lobe (i.e., 3 dB width). This width is approximately equal to f_s / L.[4].

In other words, for two sinusoids of frequencies f_1 and f_2, the resolvability condition requires that

$$(f_2 - f_1) > \frac{Fs}{L}$$

In the example above, where two sinusoids are separated by only 10 Hz, the data record must be greater than 100 samples to allow resolution of two distinct sinusoids by a timeseries.Consider a case where this criterion is not met, as for the sequence of 67 samples below:

```
fs = 1000;                % Sampling frequency
t = (0:fs/15)./fs;         % 67 samples
A = [1 2];                % Sinusoid amplitudes
f = [150;140];              % Sinusoid frequencies
xn = A*sin(2*pi*f*t) + 0.1*randn(size(t));
[Pxx,F] = time series(xn,rectwin(length(xn),length(xn,fs);
plot(F,10*log10(Pxx))
```

The above discussion about resolution did not consider the effects of noise since the signal-to-noise ratio (SNR) has been relatively high thus far. When the SNR is low, true spectrum features are much harder to distinguish, and noise artifacts appear in spectrum estimates based on the timeseries.The example below illustrates this:

```
fs = 1000;                % Sampling frequency
t = (0:fs/10)./fs;         % One-tenth second worth of samples
A = [1 2];                % Sinusoid amplitudes
f = [150;140];              % Sinusoid frequencies
xn = A*sin(2*pi*f*t) + 2*randn(size(t));
[Pxx,F] = timeseries (xn,rectwin(length (xn)),length(xn),fs);
plot(F,10*log10(Pxx))
```

3.3 Bias of the Timeseries:

The timeseries is a biased estimator of the SE. Its expected value was previously shown to be:

$$E\{P_{xx}(f)\} = \frac{1}{Fs}\int_{-Fs/2}^{Fs/2} \frac{\sin^2(L\pi(f-f')/Fs)}{L\sin^2(\pi(f-f')/Fs)} P_{xx}(f')df',$$

The timeseries is asymptotically unbiased, which is evident from the earlier observation that as the data record length tends to infinity, the frequency response of the rectangular window more closely approximates the Dirac delta function. However, in some cases the timeseries is a poor estimator of the SE even when the data record is long. This is due to the variance of the timeseries, as explained below.

3.4 Variance of the Timeseries:

The variance of the timeseries can be shown to be:

$$\mathrm{Var}(P_{xx}(f)) = \begin{cases} P_{xx}^2(f) & 0 < f < \mathrm{Fs}/2 \\ 2P_{xx}^2(f) & f = 0 \quad \text{or}\, f = Fs/2 \end{cases}$$

which indicates that the variance does not tend to zero as the data length *L* tends to infinity. In statistical terms, the timeseries is not a consistent estimator of the SE. Nevertheless, the timeseries can be a useful tool for spectrum estimation in situations where the SNR is high, and especially if the data record is long.

4 The Modified Time series:

The *modified time series* windows the time-domain signals prior to computing the DFT in order to smooth the edges of the signal. This has the effect of reducing the height of the sidelobes or spectrum leakage. This phenomenon gives rise to the interpretation of sidelobes as spurious frequencies introduced into the signal by the abrupt truncation that occurs when a rectangular window is used. For nonrectangular windows, the end points of the truncated signal are attenuated smoothly, and hence the spurious frequencies introduced are much less severe. On the other hand, nonrectangular windows also broaden the mainlobe, which results in a reduction of resolution.[5].

Timeseries allows you to compute a modified timeseries by specifying the window to be used on the data.

For example, compare a default rectangular window and a Hamming window:

```
fs = 1000;              % Sampling frequency
t = (0:fs/10)./fs;      % One-tenth second worth of samples
A = [1 2];              % Sinusoid amplitudes
f = [150;140];          % Sinusoid frequencies
xn = A*sin(2*pi*f*t) + 0.1*randn(size(t));
[Pxx,F] = timeseries(xn,rectwin(length(xn)),length(xn),fs);
```

```
plot(F,10*log10(Pxx))
[Pxx,F] = timeseries(xn,hamming(length(xn)),length(xn),fs);
plot(F,10*log10(Pxx))
```

You can verify that although the sidelobes are much less evident in the Hamming-windowed timeseries, the two main peaks are wider. In fact, the 3 dB width of the mainlobe corresponding to a Hamming window is approximately twice that of a rectangular window. Hence, for a fixed data length, the SE resolution attainable with a Hamming window is approximately half that attainable with a rectangular window.The competing interests of mainlobe width and sidelobe height can be resolved to some extent by using variable windows such as the window.Nonrectangular windowing affects the average of a signal because some of the time samples are attenuated when multiplied by the window. To compensate for this, timeseries and p normalize the window to have an average of unity.This ensures that the measured average is generally independent of window choice. If the frequency components are not well resolved by the SE estimators, the window choice does affect the average. The modified timeseries estimate of the SE is

$$\hat{P}_{xx}(f) = \frac{|X(f)|^2}{FsLU}$$

where U is the window normalization constant.

$$U = \frac{1}{L}\sum_{n=0}^{N-1} |w(n)|^2 .$$

For large values of L, U tends to become independent of window length. The addition of U as a normalization constant ensures that the modified timeseries is asymptotically unbiased.

4.1 Timeseries

An improved estimator of the SE is the one proposed by [8]. The method consists of dividing the time series data into (possibly overlapping) segments, computing a modified timeseries of each segment, and then averaging the SE estimates. The result is's SE estimate. Timeseries is implemented in the toolbox by or function. The averaging of modified timeservers tends to decrease the variance of the estimate relative to a single timeseries estimate of the entire data record. Although overlap between segments introduces redundant information, this effect is diminished by the use of a nonrectangular window, which reduces the importance or *weight* given to the end samples of segments (the samples that overlap).

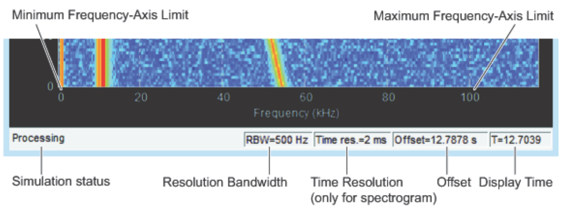

Figure 4. Minimum and Maximum Frequency Axis Limit

Hz per unit time, or:
Time in pass band = RBW
Span/ST = (RBW)(ST)
Span Where
RBW = resolution bandwidth and
ST = sweep time.
On the other hand, the rise time of a filter
is inversely proportional to its bandwidth,
and if we include a constant of proportionality, k, then:
Rise time = k
RBW
If we make the terms equal and solve for
sweep time, we have:
k = (RBW)(ST)
Span
or ST = k (Span)
RBW2

Entering the values from our example into the equation, we get:
2.12H(4 kHz) = –3.01 dB x = –24.1 dB .At an offset of 4 kHz, the 3-kHz digital filter is down –24.1 dB compared to the analog filter which was only down –14.8 dB. Because of its superior selectivity, the digital filter can resolve more closely spaced signals.

The general expression used to calculate
the maximum mismatch error in dB is:
Error (dB) = –20 log[1 ± |(ρanalyzer)(ρsource[|(where ρ is the reflection coefficient_

However, as mentioned above, the combined use of short data records and nonrectangular windows results in reduced resolution of the estimator. In summary, there is a trade-off between variance reduction and resolution. One can manipulate the parameters in Timeseries to obtain improved estimates relative to the timeseries, especially when the SNR is low. This is illustrated in the following example.

Consider an original signal consisting of 1000 samples:

```
fs = 1000;            % Sampling frequency
t = (0:1*fs)./fs;    % 301 samples
A = [2 8];            % Sinusoid amplitudes (row vector)
f = [150;140];        % Sinusoid frequencies (column vector)
xn = A*sin(2*pi*f*t) + 5*randn(size(t));
[Pxx,F] = timeseries(xn,rectwin(length(xn)),length(xn),fs);
plot(F,10*log10(Pxx))
```

We can obtain 's spectrum estimate for 3 segments with 50% overlap using a Hamming window.

```
[Pxx,F] = p(xn,hamming(150),75,150,fs);
plot(F,10*log10(Pxx)); xlabel('Hz'); ylabel('dB');
title(''s Overlapped Segment Averaging SE Estimate');
```

In the timeseries above, noise and the leakage make one of the sinusoids essentially indistinguishable from the artificial peaks. In contrast, although the SE produced by Timeseries has wider peaks, you can still distinguish the two sinusoids, which stand out from the "noise floor."However, if we try to reduce the variance further, the loss of resolution causes one of the sinusoids to be lost altogether:

```
[Pxx,F] = p(xn,rectwin(100),75,512,Fs);
plot(F,10*log10(Pxx))
```

4.2 Bias and Normalization in Timeseries:

Timeseries yields a biased estimator of the SE. The expected value of the SE estimate is:

$$E\{P_{\text{Welch}}(f)\} = \frac{1}{FsLU}\int_{-Fs/2}^{F/2} |W(f-f')|^2 \, P_{xx}(f')df'$$

Where L is the length of the data segments, U is the same normalization constant present in the definition of the modified timeseries, and $W(f)$ is the Fourier transform of the window function. As is the case for all timeservers, estimator is asymptotically unbiased. For a fixed length data record, the bias of's estimate is larger than that of the timeseries because the length of the segments is less than the length of the entire data sample.

The variance of estimator is difficult to compute because it depends on both the window used and the amount of overlap between segments. Basically, the variance is inversely proportional to the number of segments whose modified timeseriess are being averaged.

```
  fs = 1000;              % Sampling frequency
  t = (0:fs)/fs;          % One second worth of samples
 A = [1 2];              % Sinusoid amplitudes
 f = [150;140];        % Sinusoid frequencies
 xn = A*sin(2*pi*f*t) + 0.1*randn(size(t));
[Pxx1,F1] = pmtm(xn,4,fs);
plot(F1,10*log10(Pxx1))
```

By lowering the time-bandwidth product, you can increase the resolution at the expense of larger variance:

```
[Pxx2,F2] = pmtm(xn,1.5,fs);
plot(F2,10*log10(Pxx2))
```

This method is more computationally expensive than Timeseries due to the cost of computing the discrete prolate spheroidal sequences. For long data series (10,000 points or more), it is useful to compute the DPSSs once and save them in a MAT-file. dpsssave,dpssload, dpsSEir, and dpssclear are provided to keep a database of saved DPSSs in the MAT-file dpss.mat.

Cross-Spectrum Estimation Function

The SE is a special case of the *cross spectrum Estimation* (*CSE*) function, defined between two signals x_n and y_n as

$$P_{xy}(\omega) = \frac{1}{2\pi} \sum_{m=-\infty}^{\infty} R_{xy}(m) e^{-j\omega m}$$

As is the case for the correlation and covariance sequences, the toolbox *estimates* the SE and CSE because signal lengths are finite.

To estimate the cross-spectrum Estimation of two equal length signals x and y using Timeseries, the cSE function forms the timeseries as the product of the FFT of x and the conjugate of the FFT of y. Unlike the real-valued SE, the CSE is a complex function.cSE handles the sectioning and windowing of x and y in the same way as the p function:

Sxy = cSE(x,y,nwin,noverlap,nfft,fs)

4.3 Transfer Function Estimate:

One application of Timeseries is nonparametric system identification. Assume that H is a linear, time invariant system, and $x(n)$ and $y(n)$ are the input to and output of H, respectively. Then the of $x(n)$ is related to the CSE of $x(n)$ and $y(n)$ by

$$P_{yx}(\omega) = H(\omega)P_{xx}(\omega)$$

An estimate of the transfer function between $x(n)$ and $y(n)$ is

$$\hat{H}(\omega) = \frac{\hat{P}_{yx}(\omega)}{\hat{P}_{xx}(\omega)}$$

This method estimates both magnitude and phase information. The tfestimate function uses Timeseries to compute the CSE and , and then forms their quotient for the transfer function estimate. Use tfestimate the same way that you use thecSE function.

Filter the signal xn with an FIR filter, then plot the actual magnitude response and the estimated response:

```
h = ones(1,10)/10;                    % Moving-average filter
yn = filter(h,1,xn);
[HEST,f] = tfestimate(xn,yn,256,128,256,fs);
H = freqz(h,1,f,fs);
subplot(2,1,1); plot(f,abs(H));
title('Actual Transfer Function Magnitude');
subplot(2,1,2); plot(f,abs(HEST));
title('Transfer Function Magnitude Estimate');
xlabel('Frequency (Hz)');
```

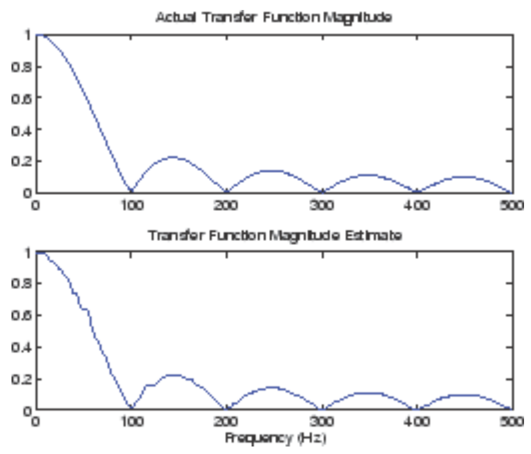

Figure 5. Frequency Function Estimation

4.4 Coherence Function:

The magnitude-squared coherence between two signals *x*(*n*) and *y*(*n*) is

$$C_{xy}(\omega) = \frac{\left|P_{xy}(\omega)\right|^2}{P_{xx}(\omega)P_{yy}(\omega)}$$

This quotient is a real number between 0 and 1 that measures the correlation between *x*(*n*) and *y*(*n*) at the frequency ω.

The cohere function takes sequences x and y, computes their spectra and CSE, and returns the quotient of the magnitude squared of the CSE and the product of the spectra. Its options and operation are similar to the cSE and tfestimate functions.The coherence function of xn and the filter output yn versus frequency is mscohere(xn,yn,256,128,256,fs).

If the input sequence length nfft, window length window, and the number of overlapping data points in a window numoverlap, are such that mscohere operates on only a single record, the function returns all ones. This is because the coherence function for linearly dependent data is one.All AR methods yield a SE estimate given by

$$\hat{P}(f) = \frac{1}{Fs} \frac{p}{\left|1 - \sum_{k=1}^{p} \hat{a}_p(k) e^{-j2\pi kf/Fs}\right|^2}$$

The different AR methods estimate the parameters slightly differently, yielding different SE estimates. The following table provides a summary of the different AR methods.*Method* of spectrum estimation computes the AR parameters by solving the following linear system, which give the equations in matrix form:

$$\begin{pmatrix} r(0) & r^*(1) & \dots & r^*(p-1) \\ r(1) & r(0) & \dots & r^*(p-2) \\ r(p-1) & r(p-2) & \dots & r(0) \end{pmatrix} \begin{pmatrix} a(1) \\ a(2) \\ \dots \\ a(p) \end{pmatrix} = \begin{pmatrix} r(1) \\ r(2) \\ \dots \\ r(p) \end{pmatrix}$$

In practice, the biased estimate of the autocorrelation is used for the unknown true autocorrelation.The method produces the same results as a maximum entropy estimator.The use of a biased estimate of the autocorrelation function ensures that the autocorrelation matrix above is positive definite. Hence, the matrix is invertible and a solution is guaranteed to exist. Moreover, the AR parameters thus computed always result in a stable all-pole model.

The equations can be solved efficiently via Levinson's algorithm, which takes advantage of the Hermitian Toeplitz structure of the autocorrelation matrix..

For example, compare the of a speech signal using Timeseries and the AR method:

```
load mtlb
[Pxx,F] = p(mtlb,hamming(256),128,1024,Fs);
plot(F,10*log10(Pxx))

ORDER = 14;
[Pxx,F] = p(mtlb,ORDER,1024,fs);
plot(F,10*log10(Pxx))
```

The smoother than the timeseries because of the simple underlying all-pole model.The accuracy of the Burg method is lower for high-order models, long data records, and high signal-to-noise ratios (which can cause *line splitting*, or the generation of extraneous peaks in the estimate). The spectrum Estimation estimate computed by the Burg method is also susceptible to frequency shifts (relative to the true frequency) resulting from the initial phase of noisy sinusoidal signals. This effect is magnified when analyzing short data sequences.

Compare the of the speech signal generated method. They are very similar for large signal lengths:

```
load mtlb
ORDER = 14;
[Pburg,F] = pburg(mtlb(1:512),ORDER,1024,fs);
plot(F,10*log10(Pburg))
[Pxx,F] = p(mtlb(1:512),ORDER,1024,fs);
plot(F,10*log10(Pxx))
```

Compare the of a noisy signal computed using the method and the method:

```
fs = 1000;             % Sampling frequency
t = (0:fs)/fs;         % One second worth of samples
A = [1 2];             % Sinusoid amplitudes
f = [150;140];         % Sinusoid frequencies
xn = A*sin(2*pi*f*t) + 0.1*randn(size(t));
p(xn,hamming(256),128,1024,fs);
pburg(xn,14,1024,fs)
```

Note that, as the model order for the Burg method is reduced, a frequency shift due to the initial phase of the sinusoids will become apparent.

5 Covariance and Modified Covariance Methods:

The covariance method for AR spectrum estimation is based on minimizing the forward prediction error. The modified covariance method is based on minimizing the forward and backward prediction errors. They are nearly identical, even for a short signal length:

```
load mtlb
pcov(mtlb(1:64),14,1024,fs)
pmcov(mtlb(1:64),14,1024,fs)
```

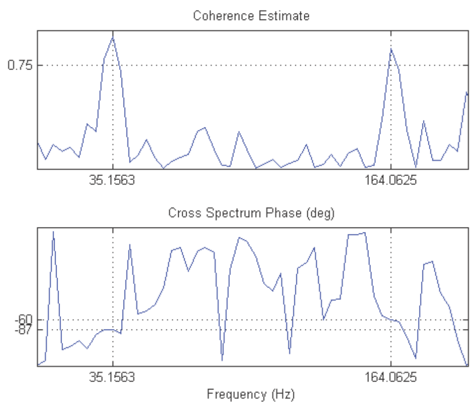

Figure 6. Coherence factor for Estimating Frequency

Consider a number of complex sinusoids embedded in white noise. You can write the autocorrelation matrix *R* for this system as the sum of the signal autocorrelation matrix (*S*) and the noise autocorrelation matrix (*W*):

R = S + W. There is a close relationship between the eigenvectors of the signal autocorrelation matrix and the signal and noise subspaces. The eigenvectors *v* of *S* span the same signal subspace as the signal vectors. If the system contains *M* complex sinusoids and the order of the autocorrelation matrix is *p*, eigenvectors v_{M+1} through v_{p+1} span the noise subspace of the autocorrelation matrix.

5.1 Frequency Estimator Functions:

To generate their frequency estimates, methods calculate functions of the vectors in the signal and noise subspaces. Techniques choose a function that goes to infinity (denominator goes to zero) at one of the sinusoidal frequencies in the input signal.

$$e(f) = [1 \quad e^{j2\pi f} \quad e^{j4\pi f} \ldots e^{j2(M-1)\pi}]^T$$

The expression $e(f)^H v_k$ is equivalent to a Fourier transform (the vector e(*f*) consists of complex exponentials). The EV method weights the summation by the eigenvalues of the correlation matrix:

$$\hat{P}_{\text{EV}}(f) = \frac{\lambda_k}{\sum_{k=p+1}^{M} |e^H(f) v_k|^2}$$

As metioned above, in addition to estimating the spectrum itself, an estimate of the confidence interval of the spectrum can be generated using a jackknifing procedure.Let us define the following:

Simple sample estimate

$$\hat{\theta} = \frac{1}{n} \sum_i Y_i$$

This is the parameter estimate averaged from all the samples in the distribution (all the tapered spectra).

leave-one-out measurement

$$\hat{\theta}_{-i} = \frac{1}{n-1} \sum_{k \neq i} Y_k$$

This defines a group of estimates, where each sample is based on leaving one measurement (one tapered spectrum) out.Pseudovalues

$$\hat{\theta}_i = n\hat{\theta} - (n-1)\hat{\theta}_{-i}$$

The jackknifed esimator is computed as:

$$\tilde{\theta} = \frac{1}{n} \sum_i \hat{\theta}_i = n\hat{\theta} - \frac{n-1}{n} \sum_i \hat{\theta}_{-i}$$

This estimator is known to be distributed about the true parameter theta approximately as a Student’s t distribution, with standard error defined as:

$$s^2 = \frac{n-1}{n} \sum_i \left(\hat{\theta}_i - \tilde{\theta}\right)^2$$

And degrees of freedom which depend on the number of tapers used (Kmax-1):

6 Spectrum calculation

6.1 Spectral analysis using MatLab

You can calculate a power spectrum for x by using the following set of commands:

```
load x552spec
x=x552spec;
P=spectrum(x,m,noverlap);
```

here m is the length of the FFT you want to use-it must be a power of 2 (e.g. 4096,2048,1024,512,256,128,64,32) overlap is the overlap between the blocks of length m-the amount by which adjacent blocks overlap. Since a Hanning window is used, the data near the ends of each block are not weighted very heavily, and it may make sense to have some overlap. You have to take this into account in your statistical testing though.So the two design parameters you have to work with are m and noverlap.

specplot(P,1.) will plot the spectrum.To do cross spectral analysis you can add a second variable name to the command.

P=spectrum(x,y,m,noverlap);

6.2 Seek the Peaks

First look for the periodicities you expect to see in x. First decide what confidence level you require. I suggest either 95% or 99%.

6.3 Cross-Spectral Analysis

Now we want to search for relationships between the two time series x and y. The command to compute the cross spectral analysis is,

P=spectrum(x,y,m,noverlap);
specplot(P,1.)

you touch the spacebar to run successively through plots of the power spectra of x and y, the amplitude of the cross spectrum, the phase between x and y and the coherence between x and y.You can make a vector of the frequencies in the folowing way,*f=0.5*(0:m2)/m2* (where m2 is m/2, the number of frequencies resolved).Then you would make a linear plot of the first half of the coherency spectrum by typing,

plot(f(1:m4),P(1:m4,5))
where m4=m/4. To plot all resolved frequencies replace m4 with m2
plot(f(1:m2),P(1:m2,5)) to plot the phase, type
ph=angle(P(1:m2,4));
plot(f(1:m4),ph(1:m4))
The phase is given in radians. If you want it in degrees, type
*ph=ph*180/3.141593*
plot(f(1:m4),pd(1:m4))

6.4 Calculation of Spectrum using of Time Series Models:

Syntax
spectrum(sys)
spectrum(sys,{wmin,wmax})
spectrum(sys,w)
spectrum(sys1,...,sysN,w)
ps=spectrum(sys,w)
[ps,w]=spectrum(sys)
[ps,w,sdps] = spectrum(sys)

Description
spectrum(sys) creates an output power spectrum plot of the identified time series model sys. The frequency range and number of points are chosen automatically.sys is a time series model, which represents the system:

$$y(t) = He(t)$$

Where, e(t) is a Gaussian white noise and y(t) is the observed output. spectrum plots abs(H'H), scaled by the variance of e(t) and the sample time.If sys is an input-output model, it represents the system:

$$y(t) = Gu(t) + He(t)$$

Where, u(t) is the measured input, e(t) is a Gaussian white noise and `y(t)` is the observed output.In this case, spectrum plots the spectrum of the disturbance component He(t).spectrum(sys,{wmin, wmax}) creates a spectrum plot for frequencies ranging from wmin to wmax. spectrum(sys,w) creates a spectrum plot using the frequencies specified in the vector w.spectrum(sys1,...,sysN,w) creates a spectrum plot of several identified models on a single plot.

```
%% Time specifications: Fs = 1e4; % samples per second dt = 1/Fs; %
seconds per sample StopTime = 1; % seconds t = (0:dt:StopTime-dt)'; N =
size(t,1);
  %% Sine wave:
  Fc = 1e3;    % hertz
  x = sin(2*pi*Fc*t);
  noisy_x = x + 0.5*randn(size(x));
  y = xcorr(noisy_x,5e3,'biased');
  display(size(y));
  subplot(2,1,1);
  plot(t,x);
  title('Sine signal');
  subplot(2,1,2);
  plot(t,noisy_x);
  title('Noisy signal');
  %% Fourier Transform:
  X = fftshift(fft(y));
  %% Frequency specifications:
  dF = Fs/N;                    % hertz
  f = -Fs/2:dF:Fs/2;            % hertz
  display(size(f));
  display(size(X));
  %% Plot the spectrum:
  figure;
  plot(f,abs(X)/N);
  xlabel('Frequency (in hertz)');
  title('Autocorrelation spectrum');
```

7 Conclusion

In urban mobile cellular systems, especially when the cell size becomes relatively small, the handoff procedure has a significant impact on system performance. In seamless Mobility handovers there will be no connectivity break but having some additional delay in service, it Degrades Quality of Service. Due to the Prior Time series based Estimation methods of Spectrum. The failures and maximum utilization of Spectrum avoided in seamless Mobility Cellular communication.

References:

1. S. Tekinay and B. Jabbari, *Handover and channel assignment in mobile cellular networks,* IEEE Communications Magazine, vol. 29, pp. 42–46, November 1991.
2. N. D. Tripathi, J. H. Reed, and H. F. VanLandinoham, *Handoff in cellular systems,* IEEE Personal Communications, vol. 5, pp. 26–37, December 1998.
3. R. Vijayan and J. Holtzman, *A model for analyzing handoff algorithms,* IEEE Transactions on Vehicular Technology, vol. 42, pp. 351–356, August 1993.
4. J. Wang, J. Liu, and Y. Cen, *Handoff algorithms in dynamic spreading WCDMA system supporting multimedia traffic,* IEEE Journal on Selected Areas in Communications, vol. 21, pp
5. S. Tekinay and B. Jabbari, *A measurement-based prioritization scheme for handovers in mobile cellular networks,* IEEE Journal on Selected Areas in Communications, 1992.
6. F. Santucci, M. Pratesi, M. Ruggieri, and F. Graziosi, *A general analysis of signal strength handover algorithms with cochannel interference,* IEEE