Biosignals and Systems

Prof. Nizamettin AYDIN

naydin@yildiz.edu.tr

naydin@ieee.org

http://www.yildiz.edu.tr/~naydin

Frequency Transformations
Probing a Signal

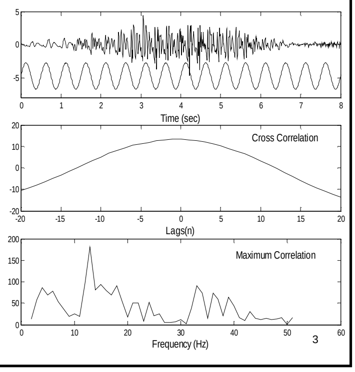
- Complex signals such as the EEG signal shown previously could be analyzed by probing with reference signals
- Crosscorrelation provides a mechanism for finding out if sinusoids are embedded in a complicated signal.
- For example, we could crosscorrelate the EEG signal with sinusoids having frequencies we think may be embedded in the signal.
- If the crosscorrelation function shows a high value at some time shift (τ), that would suggest the presence of our sinusoid, or other reference signal, at that time shift (or, equivalently, at that phase shift).

Crosscorrelation with Sinusoids

The lower curve shows the maximum crosscorrelation between sinewaves at different frequencies and the EEG signal shown in the upper plot.

The upper plot also shows one of the sinewaves used and the middle plot shows the crosscorrelation obtained for this particular sinewave.

(For this sinewave, the maximum correlation occurs close to zero lag.)



Probing with Sinusoids

- Using crosscorrelation to probe the contents of a signal works well if we know what we are looking for.
- In the previous example, twenty five sinusoids were used to probe the EEG signal using crosscorrelation, but we still may have missed some important frequencies.
- If we are probing with sinusoids, and the signal we are probing is periodic, (or can be taken as periodic), the answer to the question of which frequencies to use is found in the Fourier Series Theorem.
- The Fourier Series Theorem states that any periodic signal, no matter how complicated, can be represented by a unique sum of sinusoids; specifically, a series of sinusoids that are the same, or multiples of the signal frequency.

The Fourier Series Theorem

To put the Fourier series theorem in mathematical terms, note that if the period of a periodic function $x_T(t)$ is T, then the base or "fundamental frequency" is:

$$f_1 = \frac{1}{T}$$

and the base cosine wave would be:

$$\text{Fundamental} = \cos(2\pi f_1 t) = \cos\left(\frac{2\pi t}{T}\right)$$

and the series of harmonically related cosine waves becomes:

$$Series(n) = \cos(2\pi n f_1 t) = \cos\left(\frac{2\pi n t}{T}\right) \qquad n = 1,2,3,\ldots$$

The Fourier Series Theorem (continued)

The Fourier Series Theorem states that a periodic signal only consists of a series of harmonically related sinusoids:

$$Series(n) = C_n \cos\left(\frac{2\pi n t}{T} + \theta_n\right) \qquad n = 1,2,3,\ldots$$

The Fourier Series Theorem simply states that any periodic function, $x(t)$, can be completely and equivalently represented by a summation of this series:

$$x_T(t) = \frac{C_0}{2} + \sum_{n=1}^{\infty} C_n \cos\left(\frac{2\pi n t}{T} + \theta_n\right) \quad \text{or in terms of } f_1$$

$$= \frac{C_0}{2} + \sum_{n=1}^{\infty} C_n \cos(2\pi n f_1 t + \theta_n)$$

where $x_T(t)$ is a periodic function of period T, and the first term, C_0 *(the DC term)*, accounts for any non-zero average value of the signal.

The Fourier Series Theorem (continued)

The Fourier sinusoidal series can also be represented in terms of a sine and cosine series. Substituting in the sine/cosine representation:

$$x_p(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\left(2\pi n f_1\, t\right) + \sum_{n=1}^{\infty} b_n \sin\left(2\pi n f_1\, t\right)$$

To find the coefficients a_n and b_n simply correlate $x(t)$ with the cosine and sine waves at the various frequencies, $2\pi n f_1$.

$$a_n = \frac{2}{T}\int_0^T x(t)\cos\left(2\pi n f_1 t\right)dt \qquad n = 0,1,2,\ldots$$

$$b_n = \frac{2}{T}\int_0^T x(t)\sin\left(2\pi n f_1 t\right)dt \qquad n = 0,1,2,\ldots$$

7

The Fourier Series Theorem (continued)

An alternate method could be used based on correlation with a single sinusoid.

$$r_{xy}(\theta) = \frac{1}{T}\int_0^T x(t)\cos(2\pi n f_1 t + \theta_n)dt \quad n = 1,2,3,\ldots$$

However, for each harmonic (i.e., each value of n), we would need to find the specific phase shift θ that maximizes the cross correlation $r_{xy}(\theta)$.

(Set the derivative, $dr_{xy}(\theta)/d\theta$, to zero, solve for θ, then find the crosscorrelation at that value of θ.)

This is not too difficult to do using a computer program such as MATLAB, and the approach was used to find the sinusoidal components in the first figure, but it is not as easy as the simple correlation method particularly for hand calculations.

8

The Fourier Series Theorem (continued)

The single sinusoid representation ($\cos(2\pi ft + \theta)$) is usually a more useful representation of the Fourier series, but the coefficients are still often determined using correlation with pure sine and cosines.

From the cosine and sine coefficients (a_n and b_n) the magnitude and phase of the single sinusoid representation can be determined using the conversions below.

$$x_T(t) = \frac{C_0}{2} + \sum_{n=1}^{\infty} C_n \cos\left(2\pi n f_1\, t - \theta_n\right) \qquad n = 1,2,3,\ldots$$

$$\text{where:}\quad C_n = \sqrt{a_n^{\,2} + b_n^{\,2}} \quad \text{and} \quad \theta_n = -\tan^{-1}\left(b_n/a_n\right)$$

9

Bilateral Transformation

- Since $x_T(t)$ can be equivalently represented by the Fourier series, the series of sine and cosine coefficients, a_n and b_n, or the equivalent single sinusoid, C_n and θ_n, these coefficients are as good a representation of $x_T(t)$ as $x_T(t)$ itself.
- For this reason, representing a signal by its Fourier series is known as a "bilateral transformation.

Time Representation		Frequency Representation
$x(t)$	$\Leftrightarrow$	$\begin{cases} C_n, \theta_n \quad \text{or} \\ a_n, b_n \end{cases}$

10

Frequency Representation

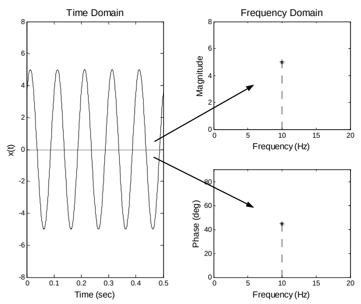
- The transformation based on sinusoids is especially useful because of the unique frequency characteristics of a sinusoid.
- A sinusoid contains energy at only one frequency.
- Thus, the sinusoidal components of a signal are also the frequency components of a signal; that is, the spectral characteristics of the signal or just "spectrum."
- A complete description of a waveform's frequency characteristics consists of two plots:
  - a plot of the components' magnitude verses frequency
  - a plot of the components' phase verses frequency.

11

Frequency Spectrum

Each sinusoidal component gives us a single point on the two frequency curves (magnitude and phase) at a frequency given to the component number n; specifically:

$$f = \frac{2\pi n}{T} = 2\pi n f_1$$



Correlation with a 10 Hz sinusoid gives the magnitude and phase characteristics of $x(t)$ at 10 Hz.

12

Symmetry

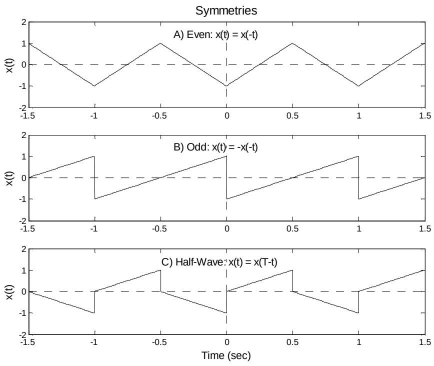
Some waveforms are symmetrical or anti-symmetrical about $t = 0$, so that one or the other of the components, a_n or b_n will be zero.

Table 3-1 Function Symmetries

Function Name	Symmetry	Coefficient Values
Even	$x(t) = x(-t)$	$b_n = 0$
Odd	$x(t) = -x(-t)$	$a_n = 0$
Half-wave	$x(t) = x(T - t)$	$a_n = b_n = 0$; for n even

13

Symmetries (continued)



14

Example 3-2 Find the Fourier Transform of the triangle waveform defined below. Find the first four Fourier Transform components.

$$x(t) = \begin{cases} t & 0 < t \le .5 \\ 0 & .5 < t \le 1.0 \end{cases}$$

Solution. Find the cosine (a_n) and sine (b_n) coefficients. Then convert to magnitude (C_n) and phase (θ_n) if desired.

To find b_n

$$b_n = \frac{2}{T}\int_0^T x(t)\sin(2\pi n f_1 t)dt = 2\int_0^1 t\sin(2\pi n t)dt$$

$$= \frac{1}{2\pi^2 n^2}\left[\sin(2\pi n t) - 2\pi n t\cos(2\pi n t)\right]_0^{.5}$$

$$= \frac{1}{2\pi^2 n^2}\left[\sin(\pi n) - \pi n\cos(\pi n)\right] = \frac{-1}{2\pi n}\left(\cos(\pi n)\right)$$

$$= \frac{1}{2\pi},\ \frac{-1}{4\pi},\ \frac{1}{6\pi},\ \frac{-1}{8\pi} \quad = \quad .159,\ -.080,\ .053,\ -.040$$

15

Example 3-2 (Continued)

To find the first term a_0.

$$\frac{a_0}{2} = \bar{x}(t) = \frac{1}{T}\int_0^T x(t)\,dt = \frac{1}{1}\int_0^{.5} t\,dt = \left.\frac{t^2}{2}\right|_0^{.5} = .125$$

To find the rest of the a_n terms:

$$a_n = \frac{2}{T}\int_0^T x(t)\cos(2\pi n f_1 t)dt = 2\int_0^{.5} t\cos(2\pi n t)dt$$

$$= \frac{1}{2\pi^2 n^2}\left[\cos(2\pi n t) - 2\pi n t\sin(2\pi n t)\right]_0^{.5}$$

$$= \frac{1}{2\pi^2 n^2}\left[\cos(\pi n) + \pi n\sin\pi n) + 1\right] = \frac{1}{2\pi^2 n^2}\left(\cos(\pi n) - 1\right)$$

$$= -\frac{1}{\pi^2},\ 0,\ -\frac{1}{9\pi^2},\ 0 \quad = \quad -.101,\ 0,\ -.012,\ 0$$

16

Example 3-2 (Continued)

These two coefficient terms can then be combined into the single sinusoidal representation as magnitude (C_n) and phases (θ_n) using the equations given previously.

Care must be taken in computing the angle to insure that it represents the proper quadrant:

$$C_n = \sqrt{a_n^2 + b_n^2} = .187,\ .080,\ .054,\ .040$$

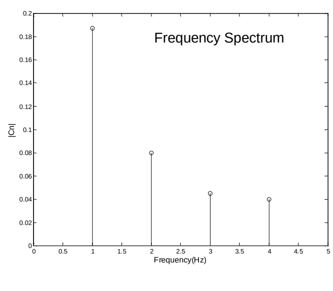
$$\theta_n = -\tan^{-1}\left(b_n/a_n\right) = -58\text{deg}(2^{nd}),\ -90\text{deg}(3^{rd}),\ -77\text{deg}(2^{nd}),\ -90(3^{rd})\text{deg}$$

$$\theta_n = \ -(180-58) = -122\text{deg},\ 90\text{deg},\ -(180-77) = -103\text{deg},\ 90\text{deg}$$

17

Example 3-1 (Continued)

A plot of the magnitude coefficients of the first four terms shows a frequency spectrum where the magnitude decreases with increasing frequency.



18

Fourier Series
Complex Representation

Euler's identity allows us to describe the sine and cosine functions in terms of imaginary exponentials

$$\cos(2\pi n f_1 t) = \frac{1}{2}\left(e^{-j2\pi n f_1 t} + e^{-j2\pi n f_1 t}\right) \quad \text{and}$$

$$\sin(2\pi n f_1 t) = \frac{1}{j2}\left(e^{-j2\pi n f_1 t} - e^{-j2\pi n f_1 t}\right)$$

Using the complex representation of a sinusoid, the Fourier Transformation correlation equations can be written as a single equation:

$$C_n = \frac{1}{T}\int_0^T x(t)e^{-j2\pi n f_1 t}dt \qquad n = 0,1,2,3\ldots$$

$$\text{where:} \quad C_n = \frac{a_n - jb_n}{2}$$

19

Fourier Transform
Complex Representation

The magnitude and the phase components can be obtained from the complex a_n and b_n:

$$|C_n| = \sqrt{\mathrm{Re}^2 + \mathrm{Im}^2} = \sqrt{\frac{a_n^2 + b_n^2}{2}} = .707\sqrt{a_n^2 + b_n^2}$$

$$\theta_n = -\tan^{-1}\left(\frac{\mathrm{Im}}{\mathrm{Re}}\right) = -\tan^{-1}\left(\frac{b_n/2}{a_n/2}\right) = -\tan^{-1}\left(\frac{b_n}{a_n}\right)$$

So the magnitude of C_n is equal to 0.707 times the magnitude of the sinusoidal components and the angle of C_n is equal to the phase of the sinusoidal component.

Hence the complex variable C_n contains both sine and cosine coefficients, and the individual components are easily determined from the complex representation.

20

Complex Fourier Series

The Fourier series equation is also know as the inverse Fourier Transform.

In complex form it requires the summation to be done for $n = \pm\infty$:

$$x(t) = \sum_{n=-\infty}^{\infty} C_n e^{j2\pi n f_1 t}$$

Although they are succinct, the complex form of the Fourier Transform and Inverse Fourier Transform may not be as useful for hand calculations.

However, the complex form is used by MATLAB and most other computer Fourier Transform routines.

21

The Sampling Theorem

•Time-slicing, better known as "sampling," has a peculiar effect on the sampled signal's spectrum:

it generates a mirror image of the spectrum about a frequency that is half of that used to sample the signal.

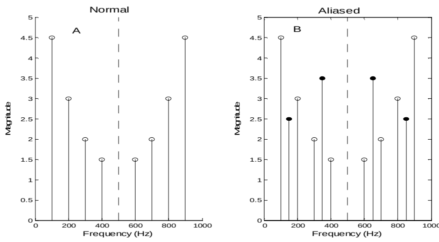
•If the sampling frequency is *fs*, then the spectrum above *fs/2* will be the mirror image of that below *fs/2*.

The frequency *fs/2* is sometimes referred to as the "Nyquist frequency."

•The generation of additional frequencies not in the original signal is termed "aliasing."

22

Aliasing



Left spectrum: If there are no frequencies in the original data above fs/2, then the frequencies created by sampling are separate from the original.

Right spectrum: If frequencies exist in the original signal greater than fs/2, then the new frequencies become intermixed with the original frequencies and it is not possible to figure out which is which. The original signal is corrupted.

23

Aliasing (continued)

- Since the Fourier Transform is bilateral, if you cannot determine the original spectrum from the one in the computer in this confused condition, you cannot determine the original signal from the signal stored in the computer.
  - The frequencies above the Nyquist frequency have hopelessly corrupted the signal stored in the computer.
- Fortunately, the converse is also true. If there are no corrupting frequencies in the original signal (i.e., the signal contains no frequencies above twice the sampling frequency), the spectrum in the computer will be a true reflection of the signal's spectrum if we eliminate the extra frequencies by filtering.
- This leads to the famous "Sampling Theorem" of Shannon:
  - the original signal can be recovered from a sampled signal provided the sampling frequency is twice the maximum frequency contained in the original:

$$fs > f_{max}/2$$

24

25

Data Truncation

- A digitized waveform must necessarily be truncated to the length of the memory storage array, a process described as "windowing."
- The windowing process can be thought of as multiplying the data by some window shape.
- If the waveform is simply truncated and no further shaping is performed on the resultant windowed waveform (as is often the case),
  - then the window shape is rectangular.
- The data length will largely determine spectral resolution

26

Spectral Resolution

The frequencies obtained by the Fourier Transform depend on the period:

$$f = \frac{n}{T} = nf_1 \quad \text{where } T \text{ also equals: } T = \frac{N}{nf_s}$$

where N is the total number of points in the data vector.

Since the spacing between frequencies on the spectral curve is proportional to 1/T, a longer effective T will lead to a spectrum with more closely spaced points, a spectrum with higher frequency resolution

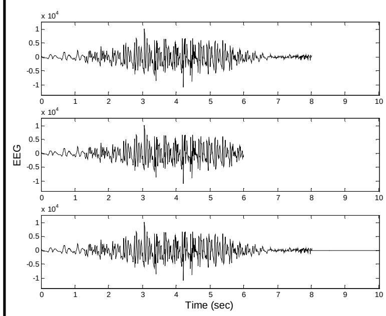
The frequency, f, can also be written as:

$$f = \frac{nf_s}{N}$$

This equation is often used in the MATLAB routines in the generation of the horizontal axis of a frequency plot.

27

Truncation and Zero Padding



EEG data at it original length.

Truncated or shortened EEG data.

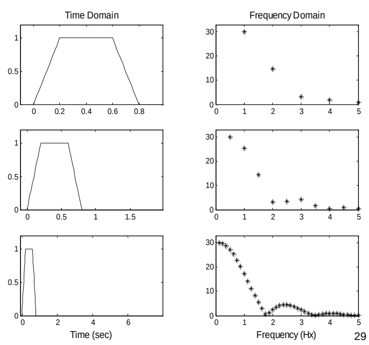
EEG data lengthened using "zero padding:" adding zeros to the end of the original data

28

Zero Padding (continued)

Adding zeros to the end of the original data make the effective period, T, longer increasing the *apparent* resolution of the spectrum.

While no new information is gained, it does perform an interpolation of the spectrum.



29

Power Spectrum

- The Power Spectrum is commonly defined as the Fourier Transform of the autocorrelation function.
- In continuous and discrete notation, the Power Spectrum equation becomes:

$$PS(f) = \int_0^T r_{xx}(t)\, e^{-j2\pi n f_1 t}\, dt$$

$$PS(n) = \sum_{n=0}^{N-1} r_{xx}(n)\, e^{-j2\pi n f_1 T_s}$$

30

Power Spectrum (continued)

- In the direct approach, the Power Spectrum is calculated as the magnitude squared of the Fourier Transform of the waveform of interest:

$$PS(f) = |X(f)|^2$$

- The Power Spectrum does not contain phase information
  - so the Power Spectrum is not a bilateral transformation
    - it is not possible to reconstruct the signal from the Power Spectrum.
- Since the Power Spectrum does not contain phase information, it is applied in situations where phase is not considered useful.

31

Spectral Averaging

- Just as time signals can be averaged, Power Spectra can be averaged.
- Even if only one signal is available, isolated segments of the data can be used.
- The Power Spectra determined from each segment is averaged to produce a spectrum that better represents the broad, or "global," features of the spectrum.
- This approach is popular when the available waveform is only a sample of a longer signal and spectral analysis can only estimate the real spectrum.
- When the Power Spectrum is based on a direct application of the Fourier Transform followed by averaging, it is commonly referred to as an average "periodogram."

32

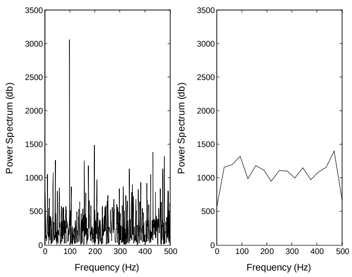
Spectral Averaging (continued)

Power spectra obtained from a waveform consisting of a 100-Hz sine wave and white noise with (right side) and without (left side) averaging.

In the un-averaged spectrum, a spike at 100 Hz is clearly seen.

For the averaged spectrum, the 100-Hz component is no longer visible;

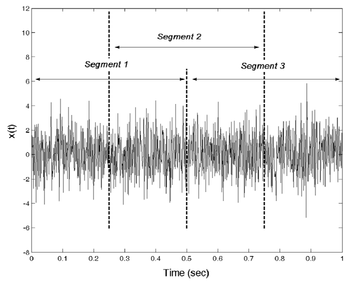
however, the averaging technique produces a better estimate of the white noise spectrum which should be a flat horizontal line.



33

Spectral Averaging (continued)

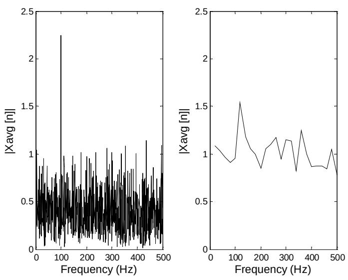
- One of the most popular procedures to evaluate the average periodogram is attributed to Welch which uses overlapping segments. (A shaping window is sometimes applied to each segment.)



34

Spectral Averaging (cont)

In the un-averaged spectrum, a spike at 100 Hz is clearly seen.



For the averaged spectrum, the 100-Hz component is not so obvious and could easily be missed, but the averaging technique produces a smoother estimate of the white noise.

Power Spectra obtained from a waveform consisting of a 100-Hz sine wave buried in white noise (SNR = -16 db) with (right side) and without (left side) averaging.

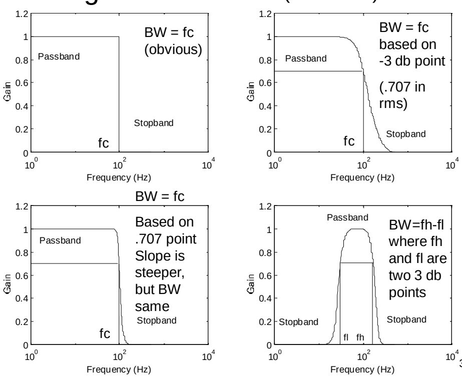
35

Signal Bandwidth

- The frequency or spectral representation of a signal brings with it additional concepts relating a signal's spectrum, one of the most important of which is signal bandwidth.
- The bandwidth of a signal is defined by the range of frequencies found in the signal.
- Defining this range requires establishing a somewhat artificial threshold for when a frequency is considered present in a signal.
- A frequency is considered within the bandwidth of the signal if its rms value is greater than the average maximum rms value minus 3 db.
- The concept of bandwidth extends to processes as well as signals.

36

Signal Bandwidth (continued)



Signal Bandwidth (continued)

- The – 3db, or "3 db down," threshold is not entirely arbitrary.
  - When the signal is attenuated 3 db its rms amplitude is 0.707 of its unattenuated value and it has half the power of its unattenuated power.
- This boundary point is also known as the "half-power point."
  - The terms "cutoff frequency," "3 db point," and "half-power point" are synonymous.
- The signal may have a sharp or gradual decline in energy (referred to as the frequency "rolloff"), but its bandwidth is still given by the -3 db point.
- When a signal "rolls off" at both the low-frequency and high-frequency ends, it has two cutoff frequencies.
  - In this case the bandwidth is defined as the range between the two cutoff frequencies: $BW = f_h - f_l$ Hz.

Frequency Methods
MATLAB Implementation

- The basic Fourier Transform routine is implemented as:
- Xf = fft(x,n)   % Calculate the Fourier Transform

where x is the input waveform and Xf is a complex vector providing the sinusoidal coefficients.

The argument n is optional and is used to modify the length of data analyzed:

if n is less than the length of x, then the analysis is performed over the first n points.

If n is greater than the length of x, then the signal is padded with trailing zeros to equal n.

- The fft routine uses the "Fast Fourier Transform" algorithm that requires the data length to be a power of two: other data lengths will require longer calculation times.

MATLAB Implementation (continued)

- The magnitude of the frequency spectra can be easily obtained by applying the absolute value function, 'abs', to the complex output Xf:

```
Magnitude = abs(Xf);          % Take the magnitude of Xf
```

This MATLAB function simply takes the square root of the sum of the real part of Xf squared and the imaginary part of Xf squared.

The phase angle of the spectra can be obtained by application of the MATLAB angle function:

```
Phase = angle(Xf)             % Find the angle of Xf
```

The angle function takes the arctangent of the imaginary part divided by the real part of Xf.

The angle routine takes note of the signs of the real and imaginary parts, and generates an output in the proper quadrant.

Example 3-6 Construct the waveform used in Example 3-1 and determine the Fourier Transform using both the MATLAB fft routine and a direct implementation of the defining equations (Eqs. 3-8).

Solution: The MATLAB fft routine does no scaling so its output should be multiplied by 2/N, where N is the number of points to get the correct coefficients in rms value. To get the peak-to-peak values, the output will have to be further scaled by dividing by 0.707.

```
N = 256;                          % Data length
t = (1:N)/N;                      % Generate time vector 1 sec long
fs = N;                           % Assumed sample frequency for 1 sec data
f = (1:N)*fs/(N);                 % Generate frequency vector for plotting
x = t;                            % Generate time vector
x(129:N) = 0;                     % Generate data signal
%
%
Xf = fft(x);                      % Take Fourier Transform, scale
Mag = abs(Xf(2:end))/(N/2);       %   and remove first point (DC value)
Phase = -angle(Xf(2:end))*(360/(2*pi));
%
plot(f(1:20),Mag(1:20),'xb'); hold on;   % Plot magnitude lower frequencies)
   xlabel('Frequency (Hz)'); ylabel('|X(f)|');
```

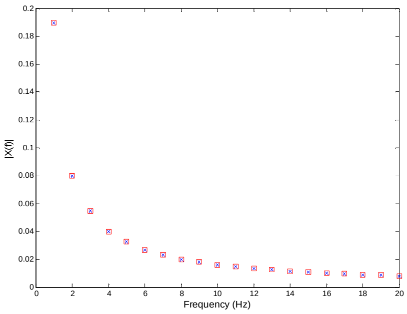

Example 3-6 (continued)

Now calculate the Fourier Transform using a direct implementation of the defining equations.

```
% Calculate Fourier Transform using basic equations (Eqs. 3-8)
for n = 1:20
   a(n) = (2/N)*sum(x.*(cos(2*pi*n*t)));
   b(n) = (2/N)*sum(x.*(sin(2*pi*n*t)));
   C(n) = sqrt(a(n).^2 + b(n).^2);          % Calculate magnitude and phase
   theta(n) = (360/(2*pi)) * atan(b(n)./a(n));
end
plot(f(1:20),C(1:20),'sr');                 % Plot superimposed
%
% Output numerical values
disp([a(1:4)' b(1:4)' C(1:4)' Mag(1:4)' theta(1:4)' Phase(1:4)'])
```

Example 3-6 ~ Results

The Magnitude spectrum produced by the two methods are identical as seen by the perfect overlap of the x points and square points.



Here the magnitude is plotted in linear units, not in db. Both linear units and db are used in spectral plots.

Example 3-6 ~ Results (continued)

The numerical values produced by this program are given below.

a_n	b_n	C_n	Mag(fft)	Theta	Phase (fft)
-0.1033	0.1591	0.1897	0.1897	-57.0182	121.5756
0.0020	-0.0796	0.0796	0.0796	-88.5938	-91.4063
-0.0132	0.0530	0.0546	0.0546	-76.0053	99.7760
0.0020	-0.0398	0.0398	0.0398	-87.1875	-92.8125

•Both methods produce identical magnitude spectra;
•however, the angles calculated using direct implementation are incorrect because the MATLAB 'atan' function does not determine the correct quadrant.

• Both magnitudes and the phase found by the fft routine match fairly closely the values determined analytically in Example 3-1.

•Note that the values for a_2 and a_4 are not exactly zero due to computational errors. (An example where hand calculation is more accurate than the computer.)

Example 3-7 Construct a waveform consisting of a single sine wave and white noise with an SNR of -14 db. Calculate the Fourier Transform of this waveform and plot the magnitude spectrum.

Solution: Use sig_noise to generate the waveform, take the Fourier Transform using fft, obtain the magnitude using abs, and plot.

The routine 'sig_noise' generates data consisting of sinusoids and noise, and can be useful in evaluating spectral analysis algorithms. The calling structure for sig_noise is:

[x,t] = sig_noise([f],[SNR],N); % Generate a signal in noise

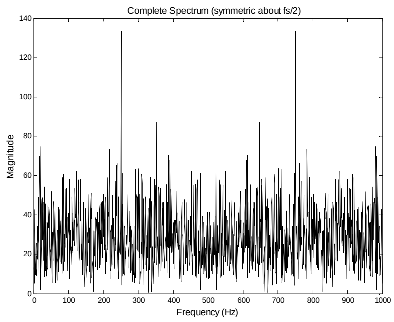
where f specifies the frequency of the sinusoid(s) in Hz, SNR specifies the desired noise associated with the sinusoid(s) in db, and N is the number of points.
If f is a vector, than a number of sinusoids are generated, each with a Signal-to-Noise ratio specified by SNR assuming it is a vector.
If SNR is a scalor, its value is used for the SNR of all the frequencies generated.

Example 3-7 (continued)

```
clear all; close all;
fs = 1000;                      % The sample frequency of data is 1 kHz.
N = 1024;                       % Number of data points
% Generate data using 'sig_noise'
%       250 Hz sin plus white noise; N data points; SNR = -14 db
[x,t] = sig_noise (250,-14,N);
%
Xf = fft(x);                    % Calculate FFT
PS = abs(Xf);                   % Calculate PS as magnitude squared
f = (1:N)*fs/N;                 % Frequency vector for plotting
plot(f,PS,'k');                 % Plot the magnitude spectrum
        title('Spectrum (symmetric about fs/2)');
        xlabel('Frequency (Hz)'); ylabel('Magnitude');
```

This program produces a frequency vector the same length as the data (N points) to aid in plotting. The frequency vector is based on Eq. 3-29 and increases linearly from1.0 to *fs.*

Example 3-7 ~ Results



The spectrum is (as expected) symmetrical around fs/2 (i.e., 500 Hz.) Normally these points would not be plotted.

The sinusoid at 250 Hz is clearly visible.

Example 3-8 Determine and plot the frequency characteristics of heart rate variability during both normal and meditative states.

Solution. The frequency characteristics may be found by direct application of the Fourier Transform.
However, the heart rate data must first be converted to a time format. The data set was obtained by a download from the PhysioNet data base and provides the heart rate at unevenly spaced times, where the sample times are provided as a second vector.
The heart rate data need to be rearranged into even time positions. This will be done through interpolation using MATLAB's 'interp1' routine.

Often getting the data into a suitable format is the hardest part of the problem.

Example 3-8 (continued)

```
%
fs = 1.0;                                   % Sample interval
load Hr_pre;                                % Load normal and meditative data
%
% Convert to evenly spaced time data using interpolation
% First generate and evenly space time vectors having one second
%   intervals and extending over the time range of the data
%
xi = (ceil(t_pre(1)):fs:floor(t_pre(end)));        % Time vector
yi = interp1(t_pre,hr_pre,xi');                    % Interpolate
yi = yi - mean(yi);                                % Remove average
f = (1:length(yi))*fs/length(yi);                  % Vector for plotting
%
% Now determine Power spectrum (take square of magnitude)
YI = abs((fft(yi)).^2);
subplot(1,2,1);
   plot(f,YI,'k');                          % Plot spectrum
   xlabel('Frequency (Hz)'); ylabel('Power Spectrum');
   axis([0 .15 0 max(YI)*1.25]);
%
    ……. Repeat for meditative data …….
```

49

Example 3-8 Analysis

<u>Analysis</u>**:**

To convert the heart rate data to a sequence of evenly spaced points in time, a time vector, xi, is first created that increases in increments of 1.0 second between the lowest and highest values of time in the original data.

A 1.0-second increment was chosen since this was approximately the average time spacing of the regional data.

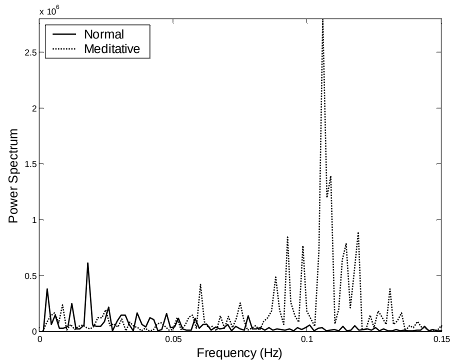
Evenly spaced time data, yi, were generated using the MATLAB interpolation routine interp1.

This routine takes the old *x* and y points and the desired new *x* points as inputs and produces an interpolated output with the desired *x* point spacing.

Since the spectrum of heart rate variability is desired, the average heart rate is subtracted before evaluating the Power Spectrum.

50

Example 3-8 Results



51

Example 3-9 Determine and plot the frequency characteristics of heart rate variability during both normal and meditative states using averaging.

<u>Solution:</u>

Write a general program called welch to generate an average Power Spectrum given the data, segment size, and the number of overlapping points in adjacent segments.

This routine should also take in, as an optional parameter, the sampling frequency to be used in generating a frequency vector.

Output the power spectrum and the frequency vector.

Output only the non-redundant points; i.e., up to *fs/2*.

```
  …… data loading and reorganizing as in Example 3-8
%
segment_length = fix(length(yi)/8);          % Average 8 segments
[PS_avg,f] = welch(yi,segment_length,segment_length-1,fs);
subplot(1,2,1)
   plot(f,PS_avg,'k');                        % Plot averaged PS
   xlabel('Frequency (Hz)'); ylabel('Power Spectrum');
   axis([0 .2 0 max(PS_avg)*1.2]);            % Limit horizontal axis
  …….  Repeat for meditative data …….
```

52

```
function [PS,f] = welch(x,nfft,noverlap,fs);
%Function to calculate averaged spectrum
%  Output arguments
%        sp        spectrogram
%        f         frequency vector for plotting
%  Input arguments
%        x data
%        nfft window size
%        noverlap number of overlaping points in adjacent segments
%        fs sample frequency
[N xcol] = size(x);                                              %
Make row vector
if N < xcol
  x = x';
  N = xcol;
end
half_segment = fix(nfft/2);            % Half segment length
if isempty(noverlap) == 1
   noverlap = half_segment;            % Set default overlap at 50%
end
increment = nfft - noverlap;
nu_avgs = fix((N-nfft)/increment)      % Determine the number of segments
```

53

Function 'welch' (continued)

```
if isempty(fs) == 0
   f = (1:half_segment)* fs/nfft;              % Calculate frequency vector
else
   f = (1:half_segment)* pi/nfft;              % Default frequency vector
end
%
for i = 1:nu_avgs                               % Calculate spectra each segment
  first_point = 1 + (i-1) * increment;          % Set up to isolate appropriate
        data = x(first_point:first_point+nfft-1);      %     data segment
  if i == 1
     PS = abs((fft(data)).^2)/(nfft*nu_avgs);  % Calculate first PS
         else
     PS = PS + abs((fft(data)).^2)/(nfft*nu_avgs);   % Calculate PS avg
  end
end
PS = PS(1:half_segment);                        % Remove redundant points
```

54

Analysis:

The routine 'welch' illustrates a number of MATLAB tricks.

• The initial section tests the dimensions of the input to determine if it is arranged as a row or column vector. If it is a column vector, the number of rows, N, will be less than the number of columns, xcol, and the vector is transposed insuring that x in now a row vector.

• The program checks if a desired overlap is specified (i.e, noverlap is not an empty variable) and, if not, sets the overlap to a default value of 50% (i.e., half the segment length, nfft).

•A frequency vector, f, is generated from 1 to π if fs is unspecified, or from 1 to fs if it is given.

• The number of segments to be averaged is determined based on the segment size (nfft) and the overlap (noverlap).

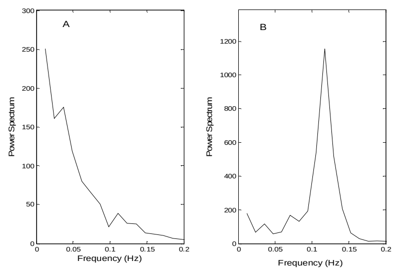
• A loop is used to take the Fourier Transform of each segment, calculate the Power Spectrum, and sum the individual spectra.

•Finally the averaged Power Spectrum is shortened to eliminate redundant points.

55

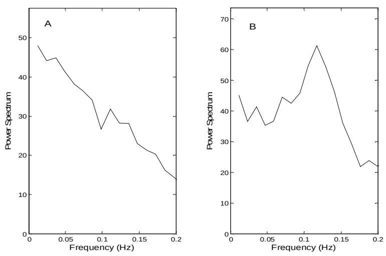
Example 3-9 Results :

The results show much smoother spectra than those taken without averaging, but they also lose some of the detail.



56

The spectra can also be plotted in db simply by taking 20log(PS_avg):



The 1/f slope is one indication of a chaotic processes.

57

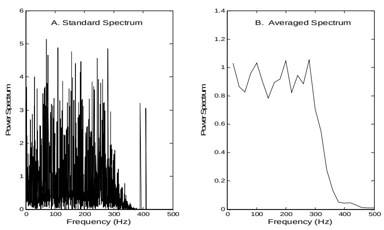
Example 3-10 Evaluate the influence of Power Spectral averaging on a combination of broadband and narrowband processes with added noise. The data may be found in file broadband1.mat.

```
load broadband1;                          % Load data (variable x)
fs = 1000;                                % Sampling frequency
%
PS = abs((fft(x)).^2)/length(x);          % Calculate un-averaged PS
half_length = fix(length(PS)/2);          % Find data length /2
f = (1:half_length)* fs/(2*half_length);  % Frequency vector for plotting
subplot(1,2,1)
plot(f,PS(1:half_length),'k');           % Plot un-averaged Power Spectrum
   xlabel('Frequency (Hz)'); ylabel('Power Spectrum');
   title('Standard Spectrum');
%
segment_length = fix(length(x)/80);       % Average 80 segments, 99% o'lap
[PS_avg,f] = welch(x,segment_length,segment_length-1,1000);
subplot(1,2,2)
   plot(f,PS_avg,'k');                    % Plot averaged Power Spectrum
   xlabel('Frequency (Hz)'); yla
```

58

Example 3-10 ~ Results

The averaged spectrum better represents the broadband noise (which should be a flat line), but loses the two narrowband sinusoids at 390 and 410 Hz.



59