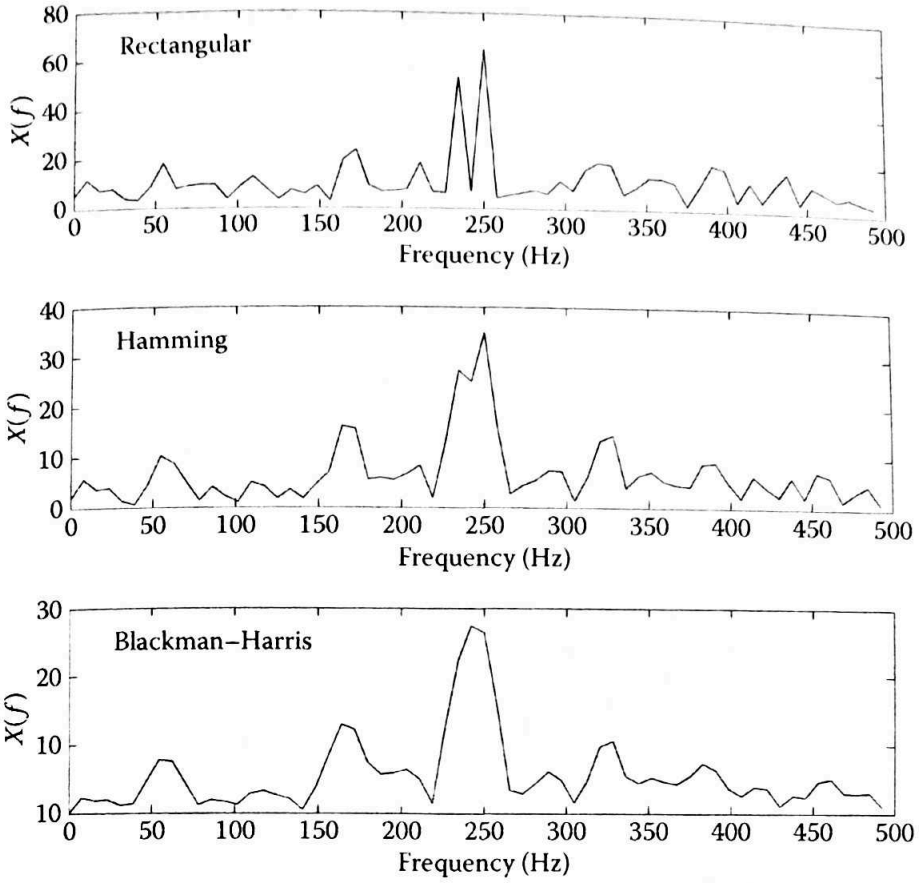

Figure 3.15 Three spectra computed from the same data but using different window functions. The data array is 128-points long and consists of two closely spaced sinusoids (235 and 250 Hz) in noise (SNR −3 dB). The influence of the three windows, with their increasing mainlobe widths, can be seen as a merging of nearby frequencies.

Results

Figure 3.15 shows three spectra obtained from Example 3.5. The differences in the three spectra are due to the windowing. The two peaks are clearly seen in the upper spectrum obtained using a rectangular function; however, the two peaks are barely discernible in the middle spectrum that used the Hamming window due to the broader mainlobe of this window that effectively averages adjacent frequencies. The effect of this frequency merging is even stronger when the Blackman–Harris window is used as only a single broad peak can be seen. The Blackman–Harris filter does provide a somewhat better estimate of the background white noise, as this background spectrum is smoother than in the other two spectra. The smoothing or averaging of adjacent frequencies by the Blackman–Harris window may seem like a disadvantage in this case, but with real data, where the spectrum may be highly variable, this averaging quality can be of benefit.

If the data set is fairly long (perhaps 256 points or more), the benefits of a nonrectangular window are slight. Figure 3.16 shows the spectra obtained with a rectangular and Hamming window to be nearly the same except for a scale difference produced by the Hamming window.

3.3 Power Spectrum

The power spectrum (PS) is commonly defined as the FT of the autocorrelation function. In continuous and discrete notations, the PS equation becomes

$$PS(f) = \frac{1}{T}\int_0^T r_{xx}(t)e^{-j2\pi mf_1 t}\,dt \quad m = 0,1,2,3,\ldots \tag{3.28}$$

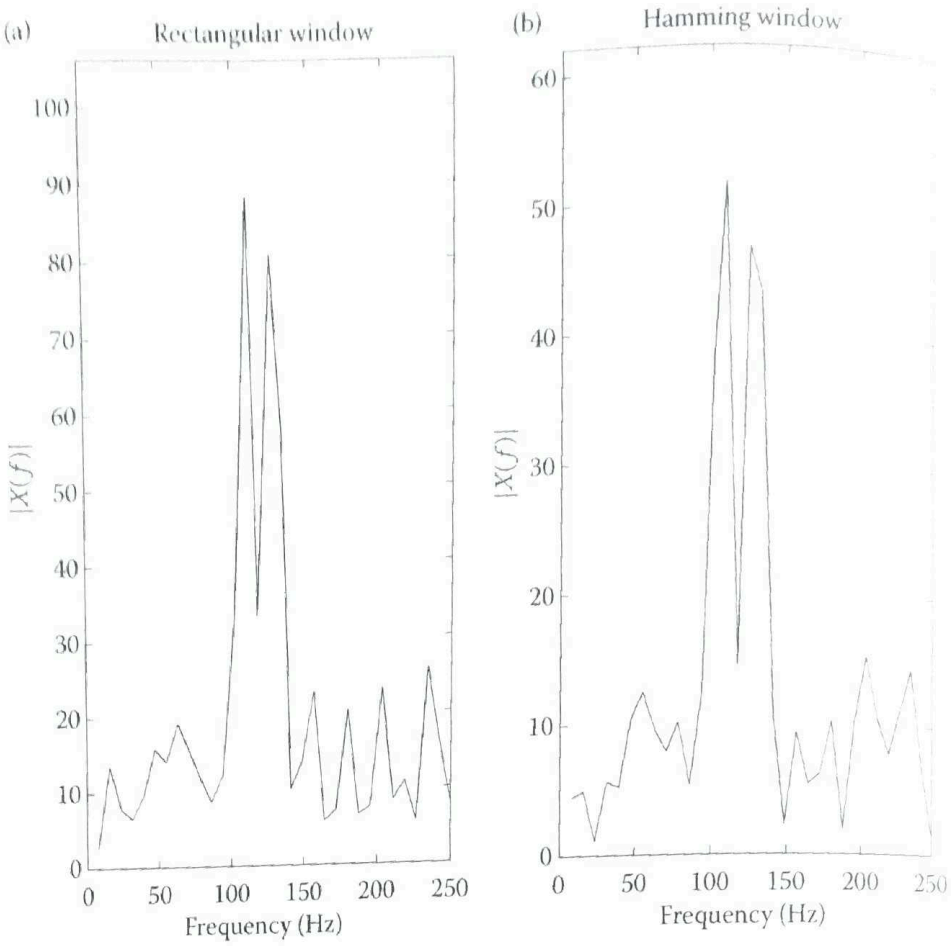

Figure 3.16 The spectra obtained from a signal containing two sinusoids and noise. The signal length is $N = 512$. The use of a nonrectangular window, in this case, a Hamming window, has little effect on the resulting spectrum. In a case where the signal consists of a fairly large number of points (>256), the rectangular window is preferred. Finally, the application of a nonrectangular window tapers the signal and reduces the energy in the signal. When this is of consequence, the scale of the magnitude spectrum can be restored by scaling the signal (or magnitude spectrum) by the ratio of the area under a rectangular window to the area under the applied window. An example of this is given in Problems 3.20 and 3.21, and in Example 3.7.

$$PS[m] = \sum_{n=1}^{N} r_{xx}[n] e^{\frac{j2\pi mn}{N}} \quad m = 0,1,2,3,\ldots,N/2 \tag{3.29}$$

where $r_{xx}(t)$ and $r_{xx}[n]$ are autocorrelation functions as described in Chapter 2. Since the autocorrelation function has even symmetry, the sine terms of the Fourier series are all zero (see Table 3.1) and the two equations can be simplified to include only real cosine terms:

$$PS[m] = \sum_{n=0}^{N-1} r_{xx}[n] \cos^{(2\pi nm/N)} \quad m = 0,1,2,3,\ldots,N/2 \tag{3.30}$$

$$PS(f) = \frac{1}{T} \int_{0}^{T} r_{xx}(t) \cos(2\pi mft)dt \quad m = 0,1,2,3,\ldots \tag{3.31}$$

Equations 3.30 and 3.31 are sometimes referred to as *cosine transforms*.

A more popular method for evaluating the PS is the *direct approach*. The direct approach is motivated by the fact that the energy contained in an analog signal, $x(t)$, is related to the magnitude of the signal squared integrated over time:

$$E = \int_{-\infty}^{\infty} |x(t)|^2 \, dt \tag{3.32}$$

By an extension of a theorem attributed to Parseval, it can be shown that

$$\int_{-\infty}^{\infty} |x(t)|^2 \, dt = \int_{-\infty}^{\infty} |x(f)|^2 \, df \tag{3.33}$$

Hence, $|X(f)|^2$ equals the energy density function over frequency, also referred to as the *energy spectral density*, *power spectral density* (PSD) or, simply and most commonly, the *power spectrum* (PS). In the direct approach, the PS is calculated as the magnitude squared of the Fourier transform (or the Fourier series) of the waveform of interest:

$$PS(f) = |X(f)|^2 \tag{3.34}$$

This direct approach of Equation 3.34 has displaced the cosine transform for determining the PS because of the efficiency of the FFT. (However, a variation of this approach is still used in some advanced signal-processing techniques involving time and frequency transformation.) Problem 3.18 compares the PS obtained using the direct approach of Equation 3.34 with the traditional cosine transform method represented by Equation 3.30 and, if done correctly, shows them to be identical.

Unlike the FT, the PS does not contain phase information; so, the PS is not an invertible transformation: it is not possible to reconstruct the signal from the PS. However, the PS has a wider range of applicability and can be defined for some signals that do not have a meaningful FT (such as those resulting from random processes). Since the PC does not contain phase information, it is applied in situations where phase is not considered useful or to data that contain a lot of noise, as phase information is easily corrupted by noise.

An example of the descriptive properties of the PS is given using the heart rate data shown in Figure 2.18. The heart rates show major differences in the mean and standard deviation of the rate between meditative and normal states. Applying the autocovariance to the meditative heart rate data (Example 2.11) indicates a possible repetitive structure for the variation in heart rate during meditation. The next example uses the PS to search for structure in the frequency characteristics of both normal and meditative heart rate data.

EXAMPLE 3.6

Determine and plot the power spectra of heart rate variability during both normal and meditative states.

Solution

The PS can be obtained through the Fourier transform using the direct method given in Equation 3.34. However, the heart rate data should first be converted into evenly sampled time data and this is a bit tricky. The data set obtained by downloading from the PhysioNet database provides the heart rate at unevenly spaced times, where the sample times are provided as a second vector. These interval data need to be rearranged into evenly spaced time positions. This process, known as *resampling*, is done through interpolation using MATLAB's `interp1` routine. This routine takes in the unevenly spaced x–y pairs as two vectors along with a vector containing the desired evenly spaced x values. The routine then uses linear interpolation (other options are possible) to approximate the y values that match the evenly spaced x values. The details can be found in the MATLAB help file for `interp1`.

In the program below, the uneven x–y pairs for the normal conditions are in vectors `t_pre` and `hr_pre`, respectively, both in the MATLAB file `Hr_pre`. (For meditative conditions, the vectors are named `t_med` and `hr_med` and are in file `Hr_med`). The evenly spaced time vector is `xi` and the resampled heart rate values are in vector `yi`.

```
% Example 3.6
% Frequency analysis of heart rate data in the normal and meditative state
%
fs = 100;                % Sample frequency (100 Hz)
ts = 1/fs;               % Sample interval
load Hr_pre;             % Load normal and meditative data
%
% Convert to evenly-spaced time data using interpolation; i.e., resampling
% First generate evenly space time vectors having one second
% intervals and extending over the time range of the data
%
xi = (ceil(t_pre(1)):ts:floor(t_pre(end)));   % Evenly-spaced time vector
yi = interp1(t_pre,hr_pre,xi");               % Interpolate
yi = diff(yi);                                % Remove average
N2 = round(length(yi)/2);
f = (1:N2)*fs/N2);                          % Vector for plotting
%
% Now determine the Power spectrum
YI = abs((fft(yi)).^2);                     % Direct approach (Eq. 3.34)
subplot(1,2,1);
   plot(f,YI(2:N2 + 1,"k");                 % Plot spectrum, but not DC value
   axis([0 .15 0 max(YI)*1.25]);            % Limit frequency axis to 0.15 Hz
 .........label and axis..........
%
% Repeat for meditative data
```

Analysis

To convert the heart rate data into a sequence of evenly spaced points in time, a time vector, `xi`, is first created that increases in increments of 0.01 s ($1/f_s = 1/100$) between the lowest and highest values of time (rounded appropriately) in the original data. A 100-Hz resampling frequency is used here because this is common in heart rate variability studies that use certain nonlinear methods, but in this example, a wide range of resampling frequencies gives the same result. Evenly spaced time data, `yi`, were generated using the MATLAB interpolation routine `interp1`. This example asked for the PS of heart rate *variability*, not heart rate per se; in other words, the change in beat-to-beat rate. To get this beat-to-beat change, we need to take the difference between sample values using MATLAB's `diff` operator before evaluating the PS.

After interpolation and removal of the mean heart rate, the PS is determined using `fft` and then taking the square of the magnitude component. The frequency plots (Figure 3.17) are limited to a maximum of 0.15 Hz since this is where most of the spectral energy is to be found. Note that the DC term is not plotted.

Results

The PS of normal heart rate variability is low and increases slightly with frequency (Figure 3.17a). The meditative spectrum (Figure 3.17b) shows a large peak at around 0.12 Hz, indicating that some resonant process is active at these frequencies, which correspond to a time frame of around 8 s. Speculation as to the mechanism behind this heart rate rhythm is left to the reader.

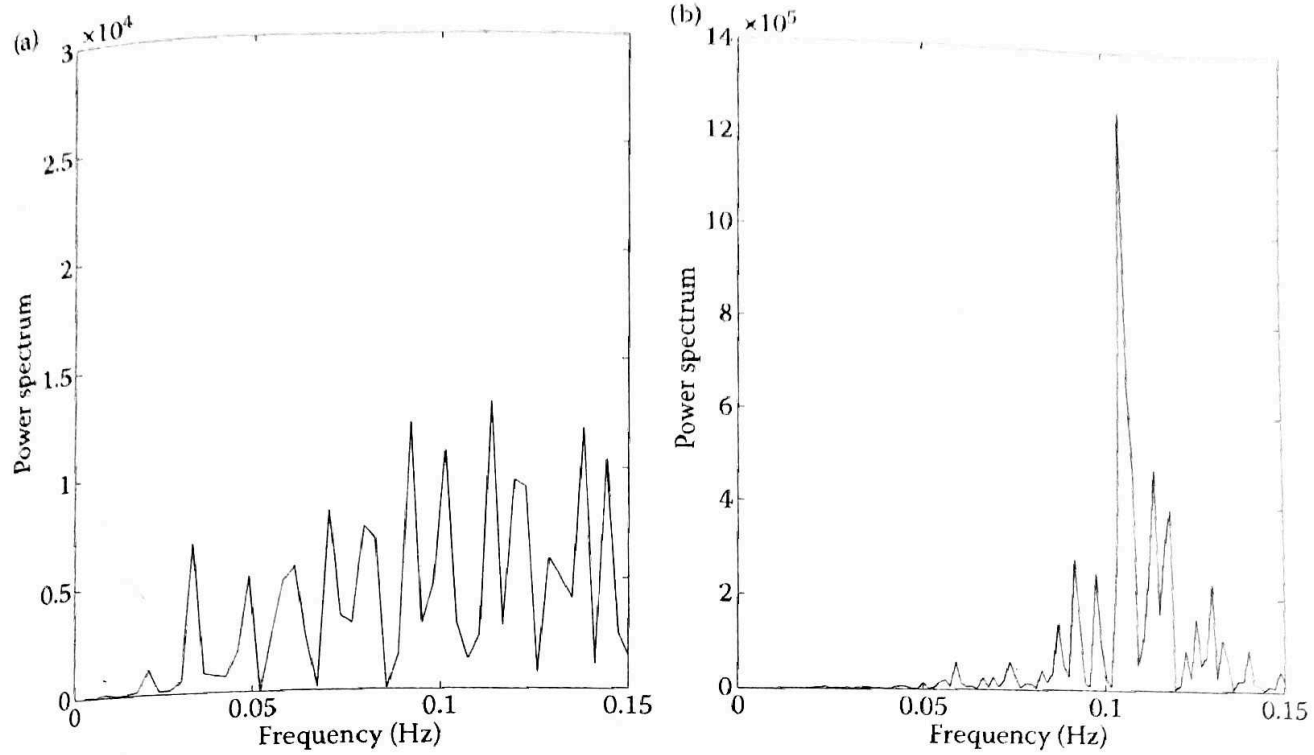

Figure 3.17 (a) The PS of heart rate variability under normal conditions. The power increases slightly with frequency. (b) The PS of heart rate variability during meditation. Strong peaks in power occur around 0.12 Hz, indicating that an oscillatory or resonant process is active around this frequency (possibly a feedback process with a time scale of 8.0 s). Note the much larger amplitude scale of this meditative spectrum.

3.4 Spectral Averaging: Welch's Method

While the PS is usually calculated using the entire waveform, it can also be applied to isolated segments of the data. The power spectra determined from each of these segments can then be averaged to produce a spectrum that represents the broadband or "global," features of the spectrum better. This approach is popular when the available waveform is only a sample of a longer signal. In such situations, spectral analysis is necessarily an estimation process and averaging improves the statistical properties of the result. When the PS is based on a direct application of the FT followed by averaging, it is referred to as an *average periodogram*.

Averaging is usually achieved by dividing the waveform into a number of segments, possibly overlapping, and evaluating the PS on each of these segments (Figure 3.18). The final spectrum is constructed from the ensemble average of the power spectra obtained from each segment. The ensemble average is an averaged spectrum obtained by taking the average over all spectra at each frequency.* Ensemble averaging can be easily implemented in MATLAB by placing the spectra to be averaged in a matrix where each PS is a row of the matrix. The MATLAB averaging routine mean produces an average of each column; so, if the spectra are arranged as rows in the matrix, the routine will produce the ensemble average.

This averaging approach can only be applied to the magnitude spectrum or PS because it is insensitive to time translation. Applying this averaging technique to the standard FT would not make sense because the phase spectrum is sensitive to the segment position. Averaging phases obtained from different time positions would be meaningless.

One of the most popular procedures to evaluate the average periodogram is attributed to Welch and is a modification of the segmentation scheme originally developed by Bartlett. In

* Ensemble averaging is also used in the time domain to reduce noise. This application of averaging is described in Chapter 4.

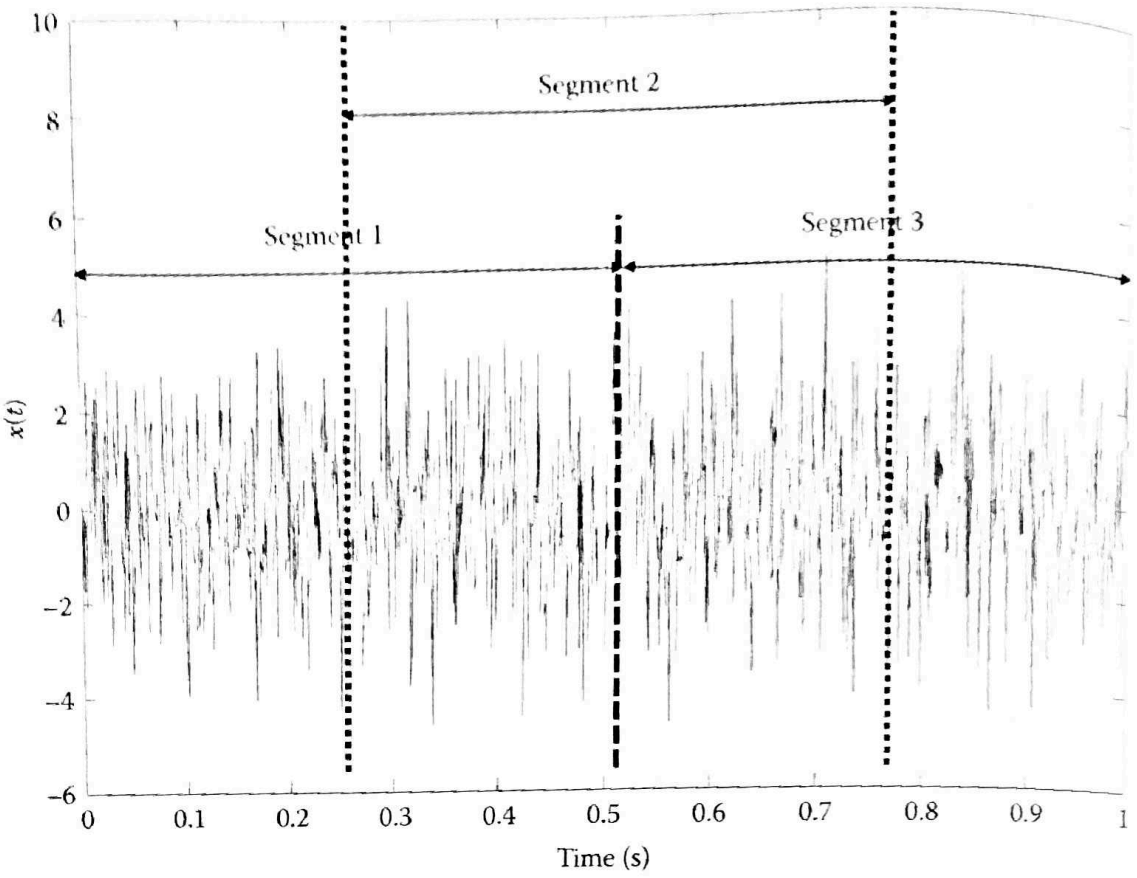

Figure 3.18 A waveform is divided into three segments with a 50% overlap. In the *Welch* method o spectral analysis, the PS of each segment is taken and an average of the three spectra is computec

this approach, overlapping segments are used and a shaping window (i.e., a nonrectangula window) is sometimes applied to each segment. *Averaged periodograms* traditionally averag spectra from half-overlapping segments, that is, segments that overlap by 50% as in Figure 3.1 Higher amounts of overlap have been recommended in applications when computing time is nc a factor. Maximum overlap occurs when each segment is shifted by only one sample.

Segmenting the data reduces the number of data samples analyzed to the number in eac segment. Equation 3.6 states that frequency resolution is proportional to f_s/N where N is no the number of samples in a segment. So, averaging produces a trade-off between spectral resolu tion, which is reduced by averaging and statistical reliability. Choosing a short segment lengt (a small N) will provide more segments for averaging and improve the reliability of the spec tral estimate, but will also decrease frequency resolution. This trade-off is explored in the nex example.

While it is not difficult to write a program to segment the waveform and average the indi vidual power spectra, it is unnecessary as MATLAB has a routine that does this. The Signa Processing Toolbox includes the function `pwelch`* that performs these operations:

```
[PS,f] = pwelch(x,window,noverlap,nfft,fs); % Apply the Welch method
```

Only the first input argument, `x`, the data vector is required as the other arguments have default values. By default, `x` is divided into eight sections with 50% overlap, each section is

* The calling structure for this function is different in MATLAB versions <6.1. Use the help file to determine the call ing structure if you are using an older version of MATLAB. Another version, `welch`, is included in the routines associated with this chapter and can be used if the Signal Processing Toolbox is not available. This version always defaults to the Hamming window but otherwise offers the same options as `pwelch`. See the associated help file.

windowed with the default Hamming window, and eight periodograms are computed and averaged. If `window` is an integer, it specifies the segment length and a Hamming window of that length is applied to each segment. If `window` is a vector, then it is assumed to contain the window function itself. Many window functions are easily implemented using the window routines described below. The segment length analyzed can be shortened by making `nfft` less than the window length (`nfft` must be less than, or equal to window length). The argument `noverlap` specifies the overlap in samples. The sampling frequency is specified by the optional argument `fs` and is used to fill the frequency vector, `f`, in the output with appropriate values. As is always the case in MATLAB, any variable can be omitted and the default can be selected by entering an empty vector, [ ]. The output PS is in vector PS and is only half the length of the data vector, x, as the redundant points have been removed. Check the help file for other options. The spectral modification produced by the Welch method is explored in the following example.

EXAMPLE 3.7

Evaluate the influence of averaging power spectra on a signal made up of a combination of broadband and narrowband signals along with added noise. The data can be found in file `broadband1.mat`, which contains a white-noise signal filtered to be between 0 and 300 Hz and two closely spaced sinusoids at 390 and 410 Hz.

Solution

Load the test data file `broadband1` containing the narrowband and broadband processes. First, calculate and display the unaveraged PS using Equation 3.34. Then apply PS averaging using an averaging routine. Use a segment length of 128 points with a maximum overlap of 127 points. To implement averaging, use `pwelch` with the appropriate calling parameters.

```
% Example 3.7 Investigation of the use of averaging to improve
% broadband spectral characteristics in the power spectrum.
%
load broadband1;                    % Load data (variable x)
fs = 1000;                          % Sampling frequency
nfft = 128;                         % Segment length
%
% Un-averaged spectrum using the direct method of Eq. 3.34
%
PS = abs((fft(x)).^2)/length(x);              % Calculate un-averaged PS
half_length = fix(length(PS)/2);              % Valid points
f = (0:half_length-1)*fs/(2*half_length);     % Frequency vector for plotting
subplot(1,2,1)
plot(f,PS(1:half_length),'k');                % Plot un-averaged Power Spectrum
    .........lables and title........
%
 [PS_avg,f] = pwelch(x,nfft, nfft-1,[], fs);  % Periodogram, max. overlap
%
subplot(1,2,2)
    plot(f,PS_avg,'k');                       % Plot periodogram
    .......labels and title.......
```

Analysis

This example uses `pwelch` to determine the averaged PS and also calculates the unaveraged spectrum using `fft`. Note that in the latter case, the frequency vector includes the DC term. For

the averaged spectrum or the periodogram, a segment length of 128 (a power of 2) was chosen along with maximal overlap. In practice, the selection of segment length and the averaging strategy is usually based on experimentation with the data.

Results

In the unaveraged PS (Figure 3.19a), the two sinusoids at 390 and 410 Hz are clearly seen; however, the broadband signal is noisy and poorly defined. The periodogram produced from the segmented and averaged data in Figure 3.19b is much smoother, reflecting the constant energy in white noise better, but the loss in frequency resolution is apparent as the two high-frequency sinusoids are hardly visible. This demonstrates one of those all-so-common engineering compromises. Spectral techniques that produce a good representation of "global" features such as broadband features are not good at resolving narrowband or "local" features such as sinusoids and vice versa.

EXAMPLE 3.8

Find the approximate bandwidth of the broadband signal, `x`, in file `Ex3_8_data.mat` (f_s = 500 Hz). The unaveraged PS of this signal is shown in Figure 3.20.

Solution

The example presents a number of challenges similar to those found in real-world problems. First, the PS obtained using the direct method is very noisy, as is often the case for broadband signals (Figure 3.20). To smooth this spectrum, we can use spectral averaging; this produces

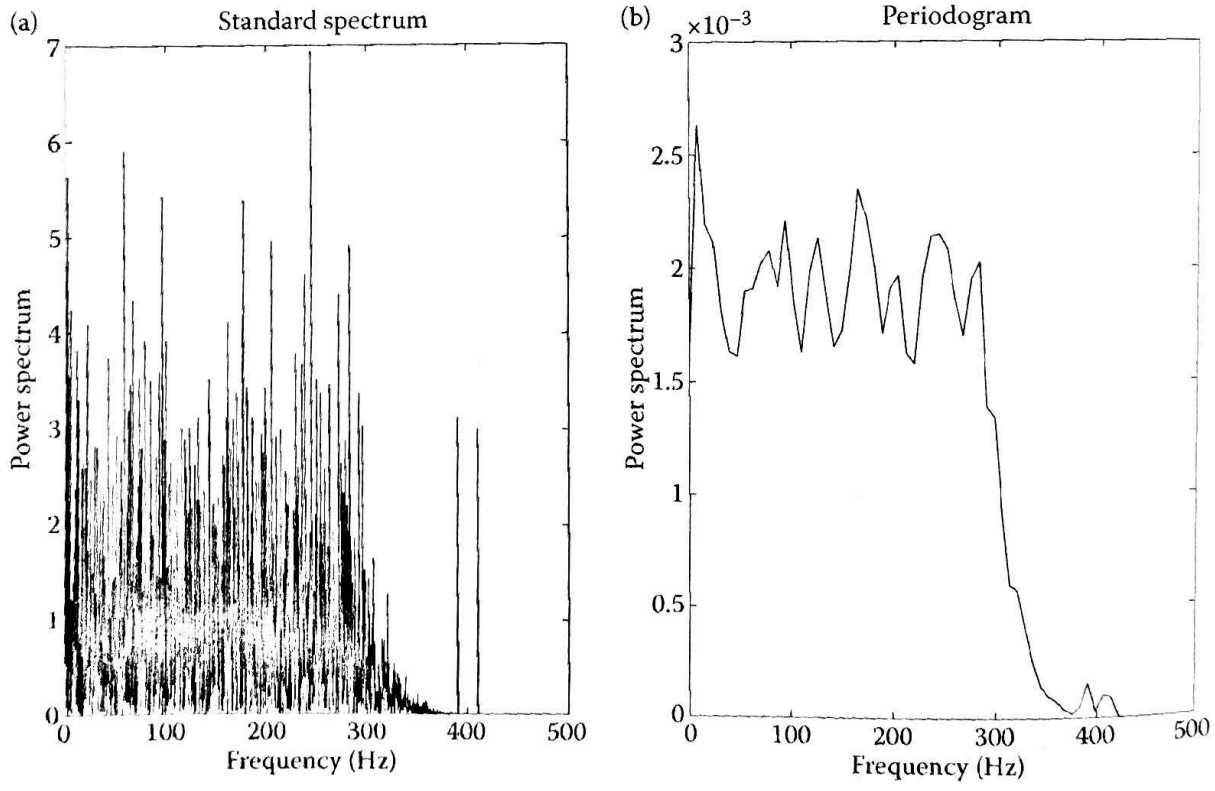

Figure 3.19 A waveform consisting of a broadband signal and two high-frequency sinusoids with noise. (a) The unaveraged PS clearly shows the high-frequency peaks, but the broadband characteristic is unclear. (b) The Welch averaging technique describes the spectrally flat broadband signal better, but the two sinusoids around 400 Hz are barely visible and would not be noticed unless you already knew they were there.

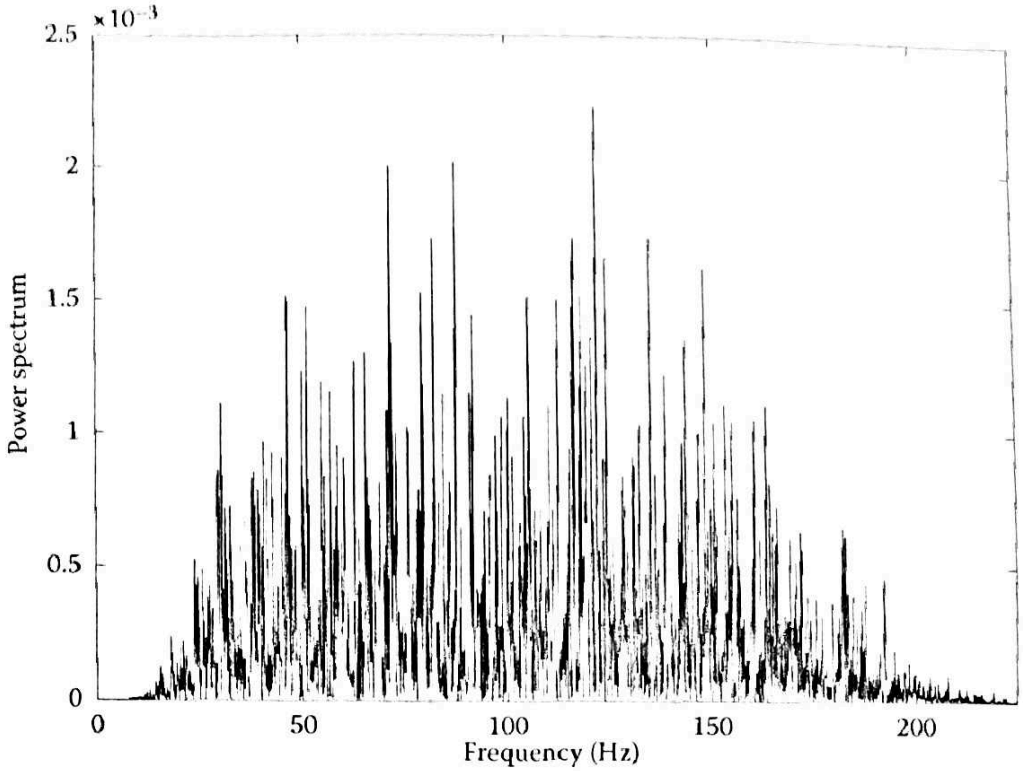

Figure 3.20 The PS of the broadband signal used in Example 3.8. Spectral averaging is used to produce a smoother spectrum (see Figure 3.21), making it easier to identify the cutoff frequencies. Since this is a PS, the cutoff frequencies will be taken at 0.5 (0.707^2), the bandpass value. This is the same as the –3-dB point in the magnitude spectrum when plotted in dB.

the cleaner spectrum shown in Figure 3.21. However, to find the bandwidth requires an estimate of spectral values in the bandpass region and there is considerable variation in this region.

To estimate the bandpass values, we take the average of all spectral values within 50% of the maximum value. This threshold is somewhat arbitrary, but seems to work well. The bandpass value obtained using this strategy is plotted as a dashed horizontal line in Figure 3.21. To find the high and low cutoff frequencies, we can use MATLAB's `find` routine to search for the first and last points >0.5 of the bandpass value. If this was the magnitude spectrum, we would take 0.707 of the bandpass value as described in Chapter 1, but since the PS is the square of the magnitude spectrum (Equation 3.34), we use 0.5 (i.e., 0.707^2) as the cutoff frequency value.

```
% Example 3.8 Find bandwidth of the signal x in file Ex3_8_data.mat
%
load Ex3_8_data.mat;              % Load the data file. Data in x.
fs = 500;                         % Sampling frequency (given).
nfft =64;                         % Power spectrum window size
[PS,f] = pwelch(x,[],nfft-1,nfft,fs);        % PS using max. overlap
plot(f,PS,'r'); hold on;                     % Plot spectrum
bandpass_thresh = 0.50 *max(PS);             % Threshold for bandpass values
bandpass_value = mean(PS(PS > bandpass_thresh));    % Use logical indexing
plot(f,PS,'k'); hold on;                      % Plot spectrum
plot([f(1) f(end)], [bandpass_value bandpass_value],':k');  % Bandpass est.
      .......labels.........
%
cutoff_thresh = 0.5 *bandpass_value;          % Set cutoff threshold
in_fl = find(PS >cutoff_thresh, 1, 'first');  % Find index of low freq. cutoff
```

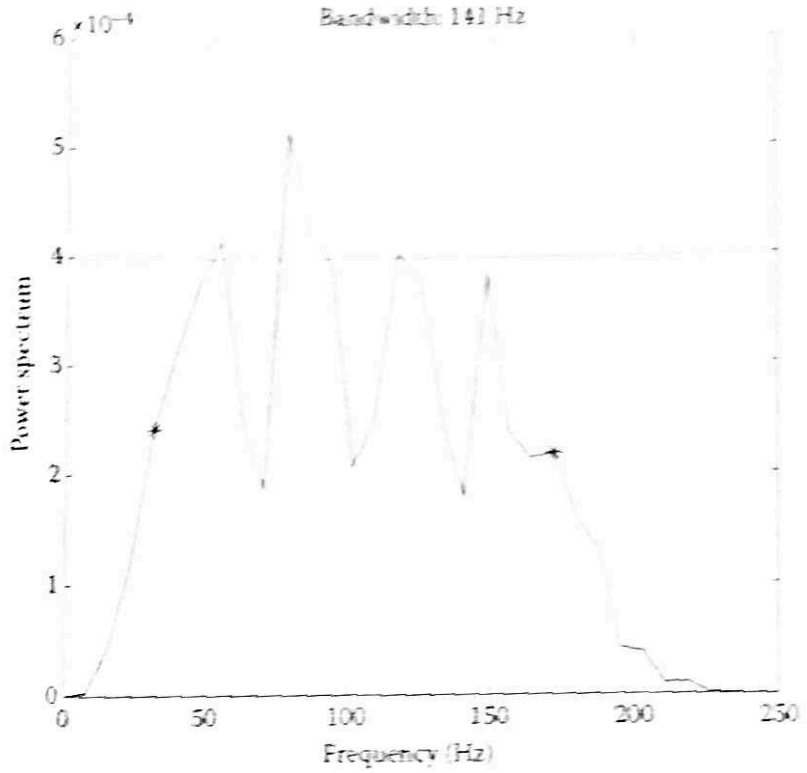

Figure 3.21 The spectrum of the signal used in Example 3.8 obtained using spectral averaging. The window size was selected empirically to be 64 samples. This size appears to give a smooth spectrum while maintaining good resolution. The bandpass value is determined by taking the average of spectral values greater than half the maximum value. The cutoff frequency points (these are the same as the –3-dB point in the magnitude spectrum) are determined by searching for the first and last points in the spectrum that are >0.5 of the average bandpass value. MATLAB's `find` routine was used to locate these points that are identified as large dots on the spectral curve. The frequency vector is used to convert the index of these points into equivalent frequencies and the calculated bandwidth is shown in the figure title.

```
in_fh = find(PS >cutoff_thresh, 1, 'last');   % Find index of high freq. cutoff
f_low = f(in_fl);                             % Find low cutoff freq.
f_high = f(in_fh);                            % Find high cutoff freq.
plot(f_low,PS(in_fl),'k*','MarkerSize',10);  % Put marker at cutoff freqs.
plot(f_high,PS(in_fh),'k*','MarkerSize',10);
BW = f_high - f_low;                          % Calculate bandwidth
title(["Bandwidth: ',num2str(BW,3), 'Hz'],'FontSize',14);
```

Results

Using the `pwelch` routine with a window size of 64 samples produces a spectrum that is fairly smooth, but still has a reasonable spectral resolution (Figure 3.21). As is so often necessary in signal processing, this window size was found by trial and error. Using *logical indexing* (`PS(PS > bandpass_thresh)`) allows us to get the mean of all spectral values >50% of the maximum power spectral value. This value is plotted as a horizontal line on the spectral plot.

Using MATLAB's `find` routine with the proper options gives us the indices of the first and last points above 0.5 of the bandpass value (f_{low} = 31 Hz and f_{high} = 172 Hz). These are plotted and superimposed on the spectral plot (large dots in Figure 3.21). The high and low sample points can be converted into frequencies using the frequency vector produced by `pwelch`. The difference between the two cutoff frequencies is the bandwidth and is displayed

in the title of the spectral plot (Figure 3.21). The estimation of the high and low cutoff frequencies might be improved by interpolating between the spectral frequencies on either side of the 0.5 values. This is done in Problem 3.36.

3.5 Summary

The sinusoid (i.e., $A \cos(\omega t + \theta)$) is a pure signal in that it has energy at only one frequency, the only waveform to have this property. This means that sinusoids can serve as intermediaries between the time-domain representation of a signal and its frequency-domain representation. The technique for determining the sinusoidal series representation of a periodic signal is known as Fourier series analysis. To determine the Fourier series, the signal of interest is correlated with sinusoids at harmonically related frequencies. This correlation provides either the amplitude of a cosine and sine series or the amplitude and phase of a sinusoidal series. The latter can be plotted against frequency to describe the frequency-domain composition of the signal. Fourier series analysis is often described and implemented using the complex representation of a sinusoid. A high-speed algorithm exists for calculating the discrete time Fourier series, also known as the DFT. The FT is invertable and the inverse FT can be used to construct a time-domain signal from its frequency representation.

Signals are usually truncated or shortened to fit within the computer memory. If the signal is simply cut off at the beginning and end, it is the same as multiplying the original signal by a rectangular window of amplitude 1.0 having the length of the stored version. Alternatively, the digital version of the signal can be multiplied by a shaping window that tapers the signal ends, providing less-abrupt endpoints. The window shape will have an influence on the resulting signal spectrum, particularly if the signal consists of a small number of data points. Window selection is a matter of trial and error, but the Hamming window is popular and is the default in some MATLAB routines that deal with short data segments. Window functions are used in the construction of some filters, as is described in the next chapter.

The DFT can be used to construct the PS of a signal. The power spectral curve describes how the signal's power varies with frequency. The PS is particularly useful for random data where phase characteristics have little meaning. By dividing the signal into a number of possibly overlapping segments and averaging the spectrum obtained from each segment, a smoothed PS can be obtained. The resulting frequency curve will emphasize the broadband or global characteristics of a signal's spectrum, but will lose the fine detail or local characteristics.