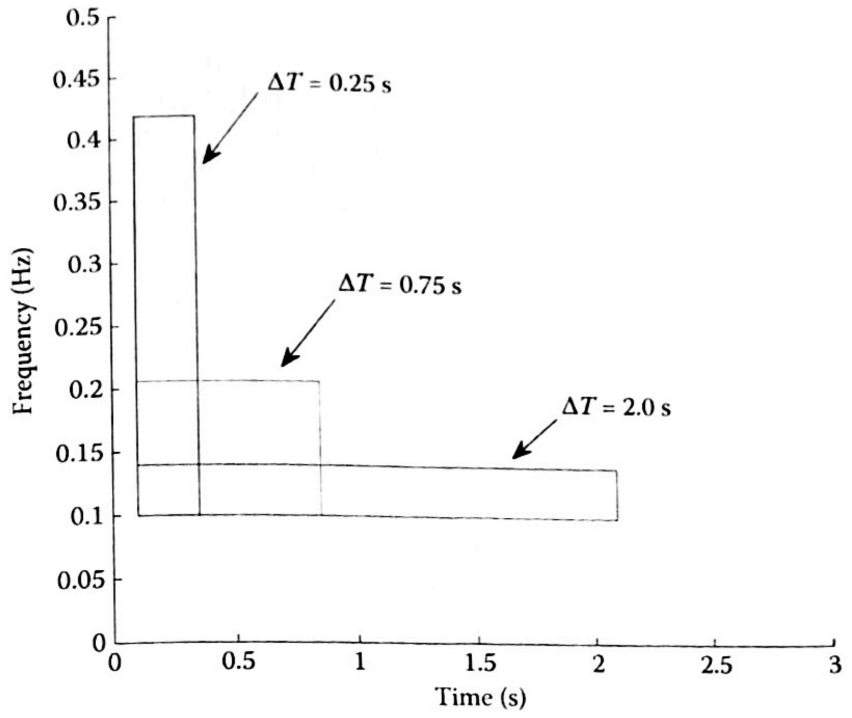

Figure 6.1 Graphical illustration of the time–frequency trade-off inherent in the STFT. The smallest time resolution gives the poorest frequency resolution (vertical rectangle) and vice versa (horizontal rectangle).

function described in Chapter 3 or using a special function of the Signal Processing Toolbox, `spectrogram`. The arguments for `spectrogram` are similar to those used for `pwelch` described in Chapter 3.

```
[B,f,t] = spectrogram(x,window,noverlap nfft,fs,);
```

where the output, B, is a complex matrix containing the magnitude and phase of the STFT time–frequency spectrum, with the rows encoding the time axis and the columns representing the frequency axis. The optional output arguments, `f` and `t`, are time and frequency vectors that can be helpful in plotting. The input arguments include the data vector, x, and the size of the Fourier transform window, `nfft`. Three optional input arguments include the sampling frequency, `fs`, used to calculate the plotting vectors, the window function desired, and the number of overlapping points between the windows. The window function is specified as in `pwelch`. If a scalar is given, then a Hamming window of that length is used.

The output of all MATLAB-based time–frequency methods is a function of two variables, time and frequency, and requires either a three-dimensional plot (3-D) or a two-dimensional (2-D) *contour* plot. Both plotting approaches are available through MATLAB standard graphics and are illustrated in the example below.

EXAMPLE 6.1

Construct a time series consisting of two sequential sinusoids of 10 and 40 Hz, each active for 0.5 s (see Figure 6.2). The sinusoids should be preceded and followed by 0.5 s of no signal (i.e., zeros). Determine the magnitude of the STFT and plot as both a 3-D grid plot and as a contour plot. Do not use the Signal Processing Toolbox routine, but develop a code for the STFT. Use a Hamming window to isolate data segments.

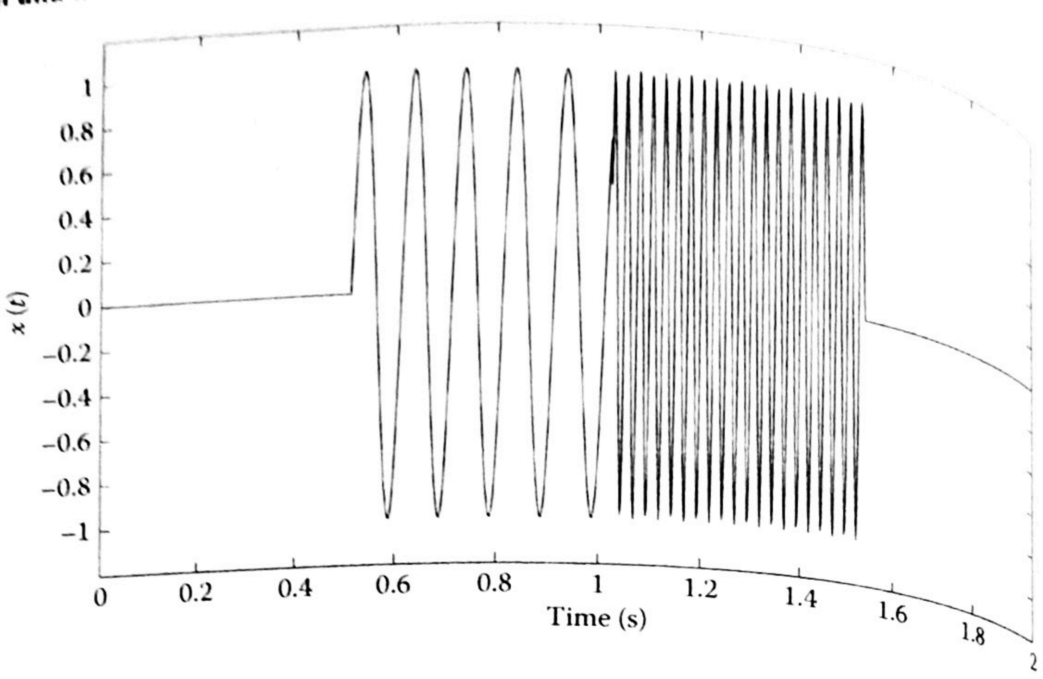

Figure 6.2 Signal consisting of two sequential sinusoids at different frequencies (10 and 4
bounded by 0.5 s of zeros. This signal is used in Example 6.1.

Solution

This example uses a function similar to MATLAB's `spectrogram`, except that a Ham
window is always used and all the input arguments must be specified. This function, spec
has arguments similar to those in `spectrogram`. The code for this routine is given b
the main program. The results are plotted both as a mesh and as a contour plot using stan
MATLAB routines.

```
% Example 6.1 Example of the use of the spectrogram
%   Uses function spectog given below
%
% Set up constants
fs = 500;                              % Sample frequency
N = 1024;                              % Signal length
f1 = 10;                               % First frequency
f2 = 40;                               % Second frequency
nfft = 64;                             % Window size
noverlap = 32;                         % Overlapping points (50%)
%
% Construct a step change in frequency
tn = (1:N/4)/fs;                       % Time vector for signal
x = [zeros(N/4,1); sin(2*pi*f1*tn)';...
   sin(2*pi*f2*tn)'; zeros(N/4,1)];
t = (1:N)/fs;                          % Time vector for plot
   .... plot signal with labels ....
%
[B,f,t] = spectog(x,nfft,noverlap,fs); % Calculate STFT
B = abs(B);                            % Spectrum magnitude
figure;
mesh(t,f,B);                           % Plot time-freq as 3-D mesh
.....labels and axis
figure
contour(t,f,B);                        % Plot time-freq as contour
.... labels and axis ....
```

The function spectog includes comprehensive comments and uses a standard MATLAB trick to ensure that the input data are arranged as a row vector. It then determines the number of samples to move the window and constructs the frequency vector based on the window size and sampling frequency. The data are zero padded at both ends to handle edge effects and a loop is used to calculate the Fourier transform at each window positions. The window time position is used to construct a time vector for use in plotting.

```
function [sp,f,t] = spectog(x,nfft,noverlap,fs);
% function [sp,f,t] = spectog(x,nfft,noverlap,fs);
% Function to calculate spectrogram
% Output arguments
%       sp spectrogram
%       t time vector for plotting
%       f frequency vector for plotting
% Input arguments
%       x data
%       nfft window size
%       fs sample frequency
%       noverlap number of overlapping points
%   Uses Hanning window
%
[N xcol] = size(x);
if N < xcol
   x = x';                            % Insure that the input is a
   N = xcol;                          % row vector
end
incr = nfft - noverlap;               % Calculate window increment
hwin = fix(nfft/2);                   % Half window size
f = (1:hwin)*(fs/nfft);               % Calculate frequency vector
%
% Zero pad data array to handle edge effects
x_mod = [zeros(hwin,1); x; zeros(hwin,1)];
%
j = 1;                                % Used to index >time vector
% Calc. spectra at each position. Use Hanning window
for k = 1:incr:N
   data = x_mod(k:k + nfft-1) .* hanning(nfft);    % Apply window
   ft = abs(fft(data));               % Magnitude data
   sp(:,j) = ft(1:hwin);              % Meaningful points
   t(j) = k/fs;                       % Calculate time vector
   j = j + 1;                         % Increment index
end
```

Figures 6.3 and 6.4 show that the STFT produces a time–frequency plot, with the step change in frequency at approximately the correct time although neither the time of the step change nor the frequencies are very precisely defined. The lack of what is called finite support in either time or frequency is particularly noticeable in the contour plot of Figure 6.4 by the appearance of energy slightly before 0.5 s and slightly after 1.5 s, and energies at frequencies other than 10 and 40 Hz. In this example, the time resolution is better than the frequency resolution. Changing the time window alters the compromise between time and frequency resolution. The exploration of the trade-off is explored in several problems at the end of this chapter.

A popular signal used to explore the behavior of time–frequency methods is a sinusoid that increases in frequency over time. This signal is called a *chirp* signal because of the sound it makes if treated as an audio signal. A sample of such a signal is shown in Figure 6.5. This signal can be generated by multiplying the argument of a sine function by a linearly increasing term as

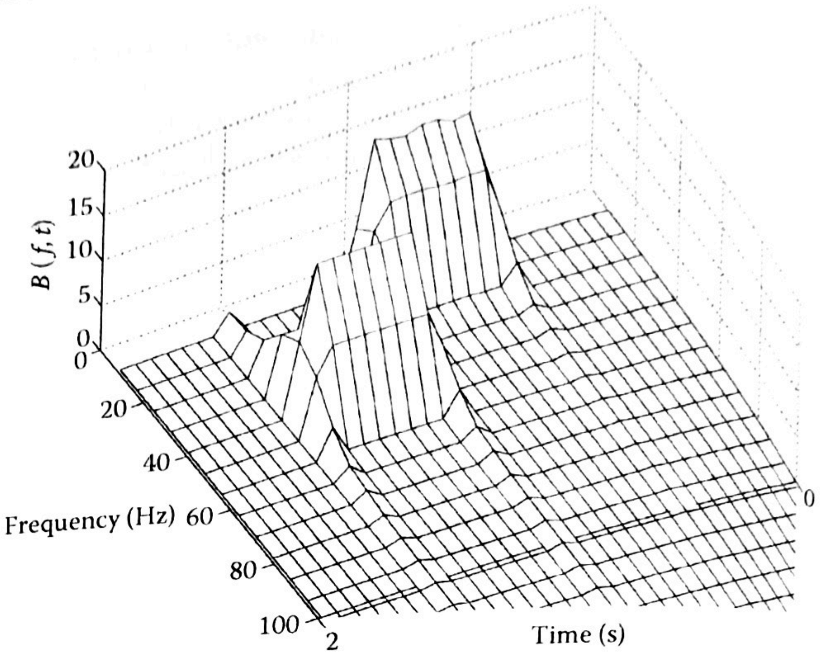

Figure 6.3 The STFT of the signal in Figure 6.2 plotted as a 3-D mesh plot.

shown in Example 6.2. Alternatively, the Signal Processing Toolbox contains a special func to generate a chip that provides some extra features, such as logarithmic or quadratic change frequency. The MATLAB `chirp` routine is used in a later example. The response of the STFT a chirp signal is demonstrated in the example below.

EXAMPLE 6.2

Generate a linearly increasing sine wave that varies between 10 and 200 Hz over a 1.0-s per Analyze this chirp signal using the STFT program from MATLAB (i.e., `spectrogram`). Us Hamming window (the default) and a 50% overlap (also the default). Plot the resulting spectr gram as both a 3-D grid and as a contour plot. Assume a sample frequency of 500 Hz; so, fo 1.0-s signal, $N = 500$.

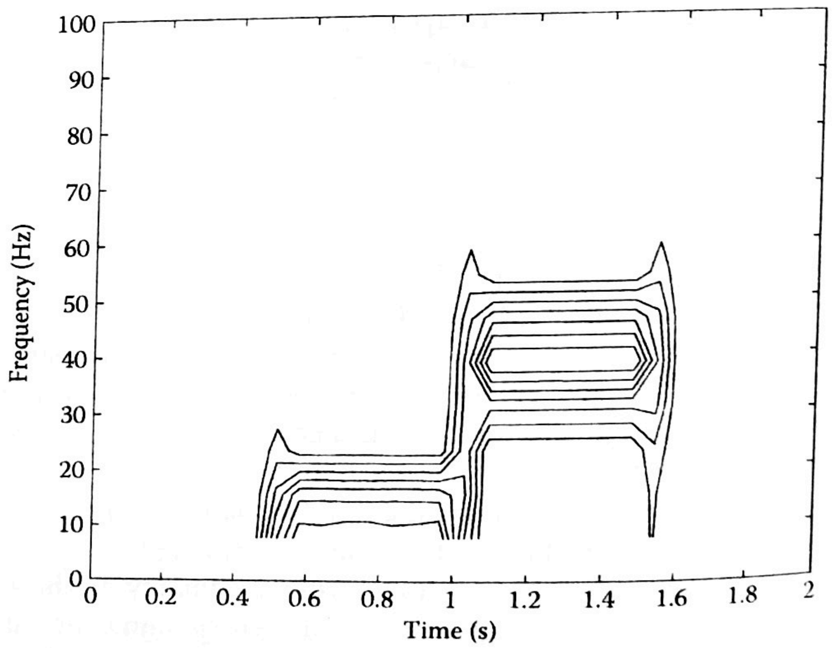

Figure 6.4 The STFT of the signal in Figure 6.2 plotted as a contour plot.

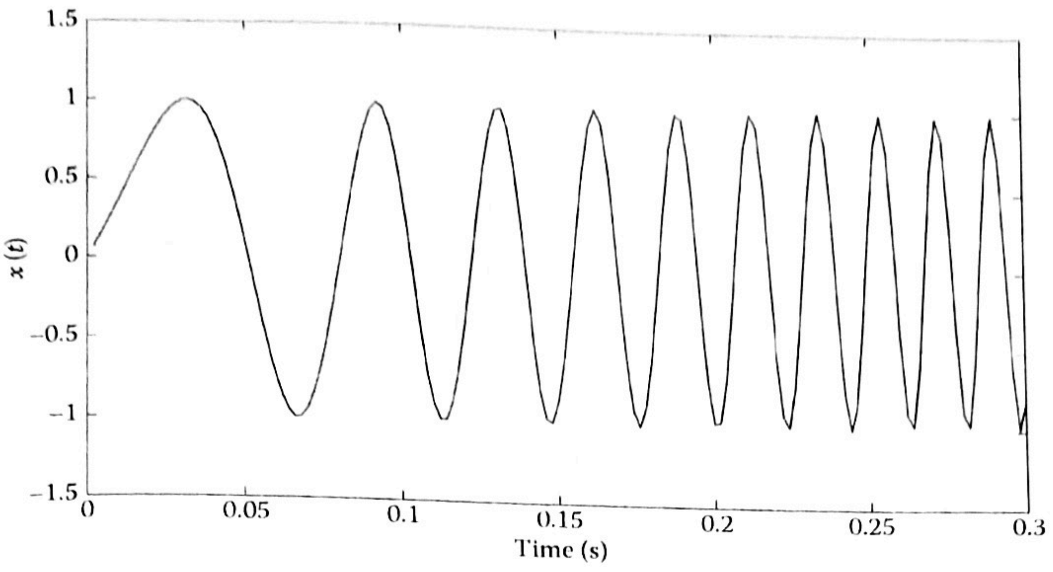

Figure 6.5 The first 0.4 s of the 1.0-s chirp signal used in Example 6.2. The continuous increase in frequency is clear.

Solution

The primary challenge of this problem is constructing the signal. To generate the chirp signal, construct a frequency vector that varies from 10 to 200 over a 500-sample interval and a time vector that varies from 0 to 1.0 over the same number of samples. Use the point-by-point product of the two vectors to generate the chirp signal, that is, `x = sin(pi*t.*fc)`. (Note that as the frequency continuously increases over time, the frequency actually generated is twice that specified by vector `fc`; so, the 2 is omitted in the sine function.) Once generated, this signal is analyzed using `spectrogram` and plotted.

```
% Example 6.2 Example to generate a sine wave with a linear change
% in frequency and evaluate that signal using the STFT.
% Constants
N = 500;                               % Number of samples
fs = 500;                              % Sample frequency
f1 = 10;                               % Minimum chirp frequency
f2 = 200;                              % Maximum chirp frequency
nfft = 32;                             % Window size
t = (1:N)/fs;                          % Time vector for chirp
fc = ((1:N)*((f2−f1)/N)) + f1;         % Frequency vector for chirp
x = sin(pi*t.*fc);
%
   ...... plot the chirp and initiate new figure
% Compute spectrogram. Default is Hamming window
[B,f,t] = spectrogram(x,nfft,[],nfft,fs);
%
mesh(t,f,abs(B));                      % 3-D plot
   .... labels, axis, title, and subplot ....
contour(t,f,abs(B));                   % Contour plot
   .... labels, axis, and title ....
```

Results

Figure 6.6a presents a 3-D plot of the STFT of the chirp signal and shows the expected continuous increase in peak frequency as time increases from 0 to 1.0 s. The contour plot in Figure 6.6 shows that the peak spectral values follow the expected linear progression between 10 and

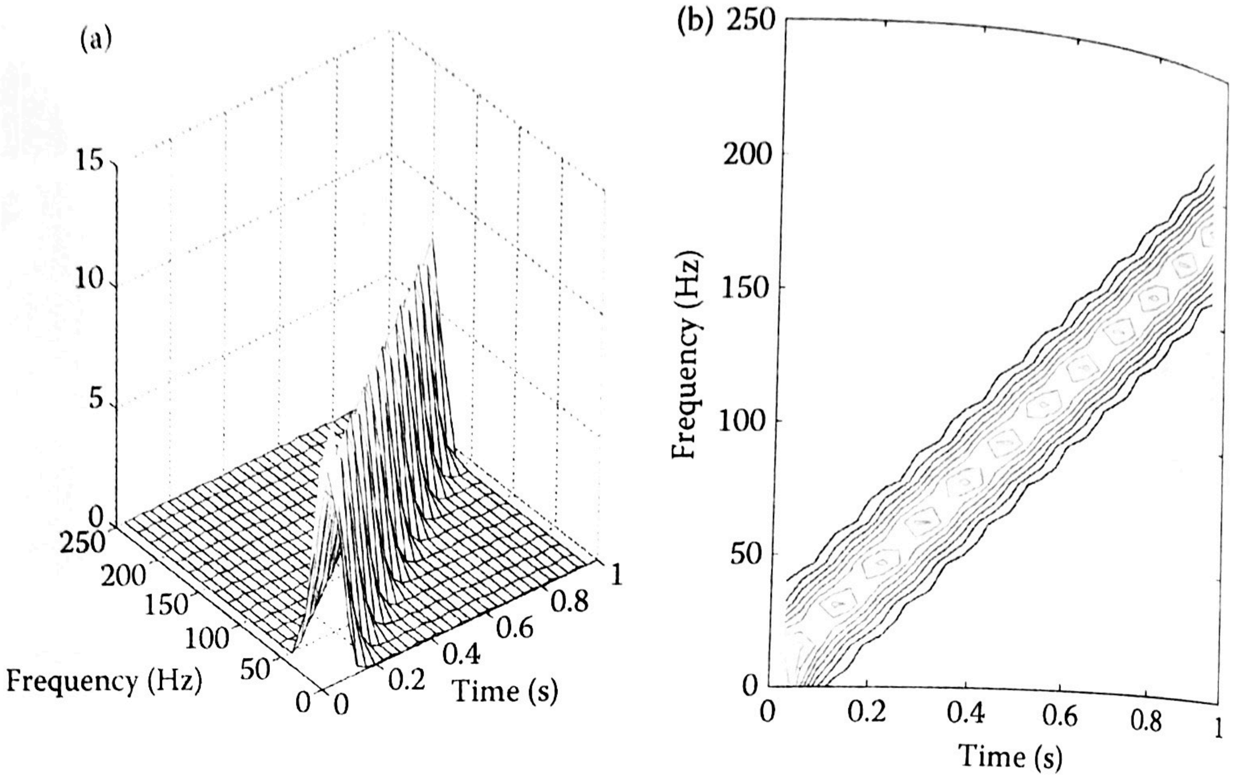

Figure 6.6 (a) 3-D plot of the STFT of a chirp signal. (b) Contour plot of the chirp signal. Both plots show the linear increase frequency over time and the energy spread over both time and frequency. Ideally, the 3-D plot should be a shaped ridge and the contour plot should be a narrow line.

200 Hz. Again, the spectrum is broadened along both time and frequency directions as predicted by the uncertainty equation (Equation 6.3). The chirp is a good signal to use to examine the time–frequency spread: the inherent trade-off between time and frequency resolution.

6.3 The Wigner–Ville Distribution: A Special Case of Cohen's Class

A number of approaches have been developed to overcome some of the shortcomings of the spectrogram. The first of these is the *Wigner–Ville distribution*,* which is also one of the most studied and best understood of the many time–frequency methods. The approach was actually developed by Wigner for use in physics, but later applied to signal processing by Ville; hence, the dual name. We will see below that the Wigner–Ville distribution is a special case of a wide variety of similar transformations known under the heading of *Cohen's class of distributions*.