3 Analysis of Rhythmic Activity in the Scalp Electroencephalogram

Synopsis

Data	Field data: 2 s of scalp EEG data sampled at 1000 Hz.
Goal	Characterize the observed rhythms in these data.
Tools	Fourier transform, power spectral density, spectrogram.

3.1 Introduction

3.1.1 Background

In this chapter, we again consider data recorded in the scalp EEG. The EEG provides a measure of brain voltage activity with high temporal resolution (typically on the order of milliseconds) but poor spatial resolution (on the order of 10 cm^2 of cortex). Here we consider EEG activity recorded from a single scalp electrode. We analyze these data to determine what (if any) rhythmic activity is present. We explain an important technique to characterize rhythms in data—the Fourier transform and power spectral density, or spectrum—and the many subtleties associated with this technique.

3.1.2 Case Study Data

A patient enters the Massachusetts General Hospital (MGH) emergency room unconscious. As part of his clinical workup, electrodes are placed on the scalp surface and the EEG recorded. We assume that the skilled technicians at MGH record the EEG data with no artifacts (i.e., correctly placed electrodes in good electrical contact with the scalp). Twenty-one electrodes simultaneously record the EEG data for 10 minutes sampled at 1000 Hz (i.e., 1,000 samples per second). To start, we receive from our clinical collaborator a 2 s segment of EEG data recorded from a single electrode. If we find anything interesting in this 2 s segment, our clinical collaborator has promised to provide additional EEG data from this patient and others.

3.1.3 Goal

Our goal is to analyze the 2 s of EEG data by characterizing the observed activity. There are many ways to do so, and we focus on developing techniques to characterize the observed rhythms. By the end of this chapter, you should be familiar with the principles of the Fourier transform, how to compute the spectrum in MATLAB, and the time-windowed spectrum.

3.1.4 Tools

The primary tool we develop here is the Fourier transform. We demonstrate how to compute the Fourier transform and the associated spectrum in MATLAB. The spectrum provides a powerful technique to assess rhythmic structure in time series data.

3.2 Data Analysis

To access the EEG data for this chapter, visit

`http://github.com/Mark-Kramer/Case-Studies-Kramer-Eden`

and download the file `Ch3-EEG-1.mat`.

3.2.1 Visual Inspection

A good place—often the best place—to begin data analysis is visual inspection of the time series. Let's plot the data in MATLAB:

```
load('Ch3-EEG-1.mat')        %Load the EEG data,
plot(t,EEG)                  %... plot it vs time,
xlabel('Time [s]')           %Label time axis.
ylabel('Voltage [ \muV]')    %Label voltage axis.
```

Q: The EEG data are plotted in figure 3.1. What do you see?

You might notice by visual inspection a dominant rhythm. We can approximate the *frequency* of this rhythm by counting the number of oscillations that occur in a 1 s interval. To do so, we might count the total number of maxima in figure 3.1 and divide by 2 (because we observe 2 s of data). However, counting so many maxima over an extended time interval is quite error prone. Instead, let's count the number of maxima in the first 0.2 s and then multiply by 5; that will approximate the total number of peaks in a 1 s interval. We count about 12 peaks in the first 0.2 s, which corresponds to approximately 60 peaks in 1 s, or approximately 60 cycles per second, or 60 Hertz (Hz).

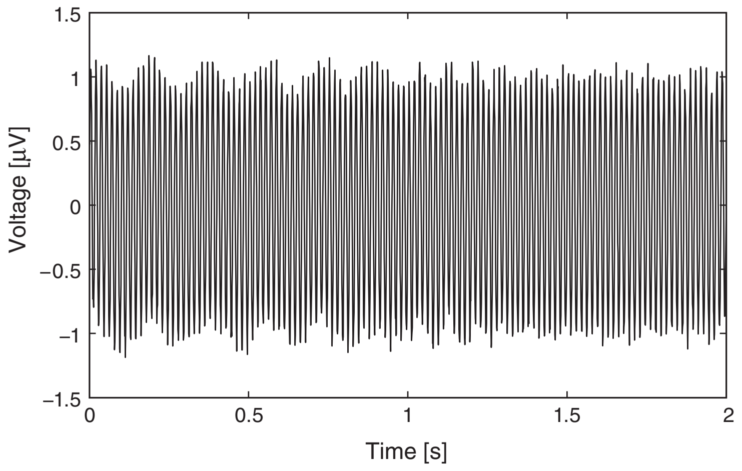

Figure 3.1
EEG data provided by our collaborator.

Q: What if you counted the minima instead of the maxima? Do you get the same answer? What if you counted the zero crossings?

Visual inspection suggests a dominant rhythmic activity at a frequency of 60 Hz. High-frequency oscillations in the 40–80 Hz band (the gamma band) are thought to be important for cognitive processing in the brain [7]. But there's a reason for the label gamma *band*: the rhythmic activity observed *in vivo* is typically diffuse, spread over a range of rhythms at neighboring frequencies. The rhythmic activity observed here is concentrated and remarkably regular for EEG data. This perhaps makes us doubtful, and rightly so.

The alternating current in any North American electrical socket alternates at 60 Hz.

We therefore propose the most likely conclusion: the data are dominated by electrical noise. We have at least three options: (1) Try to reduce the impact of this noise during the clinical recording procedure, (2) abandon this analysis and pursue other research projects, or (3) continue with additional analysis beyond visual inspection of the time series data. The first option is too expensive and time consuming; the clinical recording system cannot be changed without encountering significant regulations in the monitoring of human subjects. The second option is not viable. So, we'll pursue the third. Why? Visual inspection suggests a dominant 60 Hz signal, but looks can be deceiving. Perhaps something else is there, lurking in the signal background.

The 60 Hz noise is not all bad. Because we expect it (in North America), we can use this information to our advantage. As we build measures to characterize rhythms, we can always check these measures by confirming that the 60 Hz rhythm appears.

If only visual inspection were enough! Sometimes it is, especially when something has gone wrong (e.g., if the EEG trace were zero for all time, we should be suspicious). But looks can be deceiving. The voltage trace in figure 3.1 looks like a continuous line drawn on the page. That's incorrect. If we look more closely, we find that the data consist of discrete points (figure 3.2). Although the true brain signal may evolve as a continuous voltage trace in time, we do not observe this true signal. Instead, we observe a discrete sampling of this signal in time. The spacing between these samples is determined by the recording device collecting the EEG data. In this case, the spacing between samples is small; our collaborator has told us that the data were sampled at 1000 Hz, which corresponds to a sample of data every 1 ms. So, we observe not the (presumably) continuous true voltage signal but instead discrete samples of this signal in time. Note that in figure 3.1 the discrete points are so closely spaced that the discrete sampling of the signal appears to be a continuous curve evolving in time.

To understand the impact of this discrete sampling, we first require some definitions. Let's define Δ as the time between samples, in this case $\Delta = 1$ ms. We also define N as the

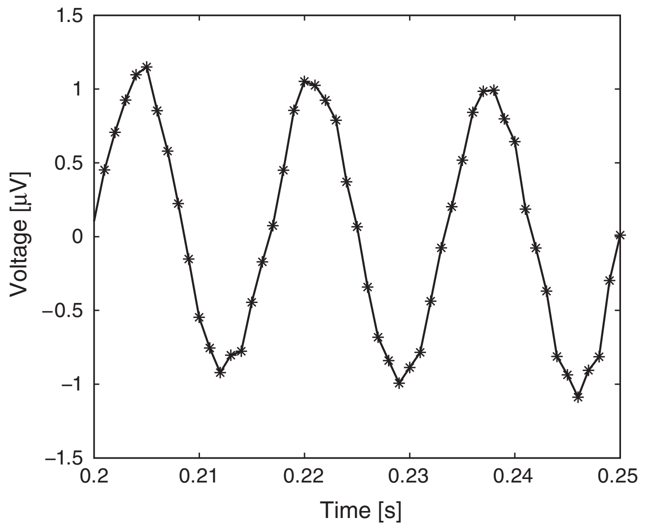

Figure 3.2

Inspection of a small window of data to investigate discrete sampling of the EEG. Asterisks indicate individual sample points. Time interval between points is the sampling interval, Δ.

total number of points observed, and T as the total time of the recording. These three terms are related: $T = N\Delta$. For $T = 2$ s of EEG data, there are $N = T/dt = 2/0.001 = 2000$ points. From this, we can also define the sampling frequency $f_0 = 1/\Delta$, which in this case is 1000 Hz. Finally, we define a symbol for the data, x, which we also write as x_n to explicitly indicate the index $n \in \{1, 2, 3, \ldots, N\}$ corresponding to the sample number. Let's also define all of these variables in MATLAB:

```
x  = EEG;            %Relabel the data variable.
dt = t(2)-t(1);      %Define the sampling interval.
N  = length(x);      %Define the total number of data points.
T  = N*dt;           %Define the total duration of data.
```

We need to keep the sampling interval Δ and the total recording duration T in mind; both play fundamental roles in the characterization of the rhythmic activity.

Q: In the second line of this code we define the sampling interval as `dt=t(2)-t(1)`. How else could we have defined `dt`? Would `t(10)-t(9)` be appropriate?

3.2.2 Mean, Variance, and Standard Deviation

As a first step in analyzing the EEG data, let's define two of the simplest measures we can use to characterize data x: the *mean* and the *variance*.[1] To estimate the mean $\bar{x}$, or average value of x, we compute

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n. \tag{3.1}$$

In words, we sum the values of x for all n time indices, then divide by the total number of points summed (N). To estimate the variance σ^2 of x we compute,

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})^2, \tag{3.2}$$

which characterizes the extent of fluctuations about the mean. The *standard deviation* is simply the square root of the variance, σ. It's straightforward to compute all three quantities using built-in MATLAB functions:

```
mn = mean(x);        %Compute the mean of the data.
vr = var(x);         %Compute the variance of the data.
sd = std(x);         %Compute the standard deviation of the data.
```

1. We could instead write the *sample* mean because we use the observed data to estimate the theoretical mean that we would see if we kept repeating this experiment. However, this distinction is not essential to our goals here.

Q: Compare the mean computed in MATLAB with the EEG data plotted in figure 3.1. Are the two consistent? How does the standard deviation computed in MATLAB compare with the EEG fluctuations in figure 3.1?

A: The computed mean is approximately 0. Visual inspection of figure 3.1 suggests that the EEG data fluctuates around a center value of 0, so the computed mean is consistent with the visual inspection of the data. The computed standard deviation is approximately 0.71. We expect that most of the signal fluctuations lie within two standard deviations (i.e., $\pm 2\sigma$) of the mean. We therefore expect to observe EEG values mostly between $0 \pm 1.4 = (-1.4, 1.4)$, which is in fact what we observe.

The mean, variance, and standard deviation provide single numbers that summarize the EEG trace. In this case, these numbers are not particularly useful. They may depend on many factors, including the electrical contact between the electrode and scalp surface, the noise in the signal, and the cognitive state of the subject. Here, we're more interested in how the EEG activity is distributed across rhythms. We've already begun to assess rhythms in the EEG data through visual inspection of the time series. To further characterize these rhythms, we employ another powerful tool, the Fourier transform. Before introducing the Fourier transform, we consider a related measure, the autocovariance.

3.2.3 Autocovariance

Visual inspection strongly suggested a prominent feature in the data: rhythmic activity. Rhythmic activity represents a type of dependent structure in the data. For example, if we know the data tend to oscillate near 60 Hz, then given the value of the EEG data now, we can accurately predict the value of the EEG data 1/60 s in the future (i.e., one cycle of the 60 Hz activity); it should be similar. One technique to assess the dependent structure in the data is the *autocovariance*. The formula for the autocovariance, $r_{xx}[L]$, evaluated at lag L, is

$$r_{xx}[L] = \frac{1}{N}\sum_{n=1}^{N-L}(x_{n+L} - \bar{x})(x_n - \bar{x}). \tag{3.3}$$

In words, the autocovariance multiplies the data x at index $n + L$ by the data x at index n, and sums these products over all indices n. Notice that in both terms the mean value $\bar{x}$ is subtracted from x before computing the product, and we divide the resulting sum by N, the total number of data points in x. Note that (3.3) is a *biased* estimate of the autocovariance; we compare this to an unbiased estimate of the autocovariance in the next section.

To gain some intuition for the autocovariance, let's represent x graphically as a one-dimensional row vector (figure 3.3a). For the case $L = 0$, the autocovariance is simply

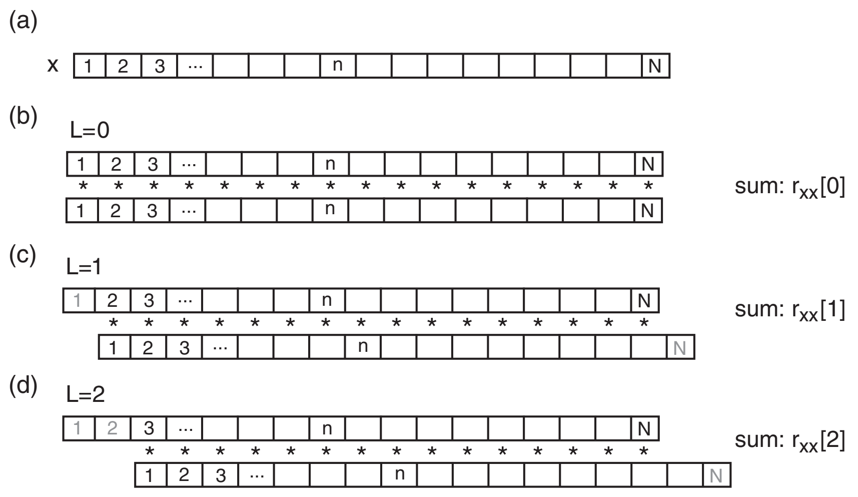

Figure 3.3

Cartoon representation of autocovariance. (*a*) Data x are represented as one-dimensional vector; with indices $n = \{1, 2, 3, \ldots, N\}$. Autocovariance at (*b*) lag 0, (*c*) lag 1, and (*d*) lag 2. Asterisks indicate multiplication between elements of the two vectors. Gray index labels at beginning and end of vectors indicate data points not involved in computing autocovariance at the chosen lag L.

the element by element product of x with itself, summed over all indices (figure 3.3b). For the case $L = 1$, we shift x by one index, multiply element by element the original (unshifted) x by the shifted version, and sum over all indices (figure 3.3c). This process of shifting, multiplying element by element, and summing can be repeated for both positive and negative values of the lag L. Notice that for larger values of L, we lose values at the beginning and ends of the autocovariance.

Q: What is the largest reasonable value of L to consider? For example, does a value of L greater than N make sense?

The autocovariance will be largest at the lag L for which the values of x match. For most functions the autocovariance is largest at $L = 0$ (of course, x matches itself with zero shift) and tends to decrease as the magnitude of L increases. Physically, the decrease in autocovariance with lag is consistent with the notion that data become less similar as time progresses. For example, in an EEG recording, we expect the activity now to be similar to the activity in the immediate future but different from the activity in the more distant future. As the brain responds to different internal and external cues, we expect different EEG activities to emerge and associations between the EEG activity now and later to decay. Functions x that exhibit dependent structure possess informative features in the autocovariance.

Q: Compare the autocovariance at $L = 0$ and the variance (3.2). Notice anything similar?

To compute the autocovariance of the EEG data, we execute the following commands in MATLAB:

```
%Compute the autocov for L +/- 100 indices.
[ac,lags] = xcorr(x-mean(x),100,'biased');
%... and plot the autocov vs lags in units of time
plot(lags*dt,ac)
%... with axes labeled.
xlabel('Lag [s]')
ylabel('Autocovariance');
```

Q: Examine the autocovariance of the EEG data plotted in figure 3.4. What do you observe?

The first input to the function `xcorr` is the EEG data with the mean subtracted (`x - mean(x)`). One striking feature of the autocovariance is the periodicity. A careful

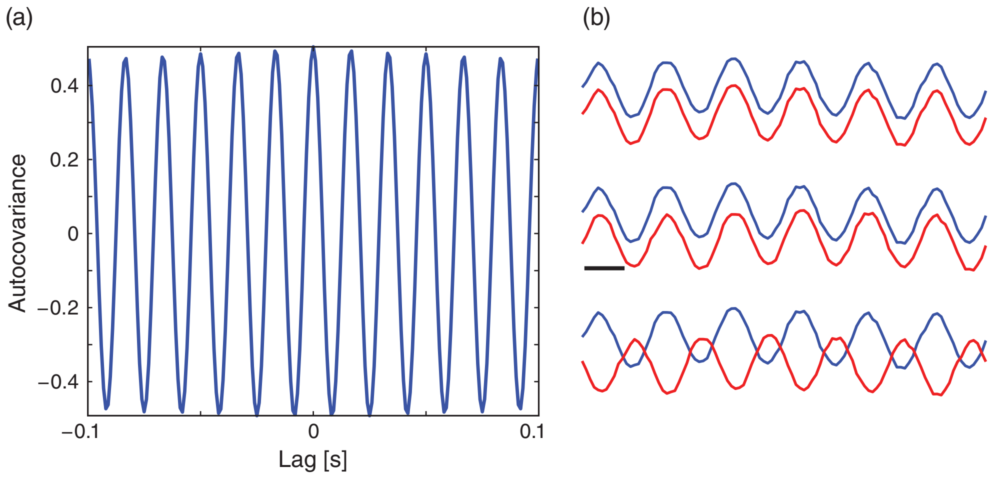

Figure 3.4
(*a*) Autocovariance of the EEG. (*b*) Examples of EEG at different shifts. With no shift (*top*) traces exactly overlap. With shift of one 60 Hz cycle (*middle*) traces appear quite similar. With shift of one-half the 60 Hz cycle (*bottom*) traces move in opposite directions. Scale bar indicates 10 ms.

inspection shows that the autocovariance exhibits repeated peaks and troughs approximately every 0.0166 s.

Q: Why does the autocovariance exhibit repeated peaks and troughs approximately every 0.0166 s?

A: The autocovariance reflects the dominant rhythmic activity in the data. Remember that the EEG data are dominated by a 60 Hz rhythm (figure 3.1). To gain intuition for how this rhythmic activity affects the autocovariance, we plot in figure 3.4 examples of the EEG data aligned with different lags L. At zero lag ($L = 0$), the two time series are identical. Therefore, the product

$$(x_{n+0} - \bar{x})(x_n - \bar{x}) = (x_n - \bar{x})(x_n - \bar{x}) = (x_n - \bar{x})^2$$

is non-negative for all indices n (the product may sometimes be zero, but it's never negative). To compute the autocovariance, we sum this product over all indices n, and divide by N, as defined in (3.3). Because we sum many positive terms, we expect to find a large positive value for $r_{xx}\left[0\right]$. Shifting the EEG data by an integer multiple of the 60 Hz cycle, we observe a similar voltage trace (figure 3.4b). Therefore, at this lag L, we again expect the summed product

$$(x_{n+L} - \bar{x})(x_n - \bar{x})$$

over all indices n to be large, and to find a large positive value for $r_{xx}\left[L\right]$. In fact, we expect the autocovariance to be large and positive whenever the lag L is an integer multiple of the 60 Hz cycle (i.e., an integer multiple of $1/60 \approx 0.0166$ s); this is exactly what we find in figure 3.4a.

Shifting the EEG data by half of the 60 Hz cycle, we observe a different type of relation; at this lag, call it L^*, positive values in the unshifted EEG correspond to negative values in the shifted EEG (figure 3.4b). Therefore, most terms in the product

$$(x_{n+L^*} - \bar{x})(x_n - \bar{x})$$

are negative, and summing up these terms to compute the autocovariance (3.3), we find a large negative value for $r_{xx}\left[L^*\right]$.

The autocovariance is a useful tool for assessing the dependent structure in the EEG data. Visual inspection of the EEG reveals a specific type of dependent structure—a strong rhythmic component—in the data. This dependent structure is further characterized in the autocovariance, in which the dominant 60 Hz activity manifests as periodic peaks and troughs. In the next section, we consider a second tool, the spectrum, for assessing dependent structure in time series data. The autocovariance and spectrum are related in a remarkable way.

Biased Versus Unbiased Autocovariance The equation for the autocovariance (3.3) is a biased estimate of the true autocovariance. To compute an *unbiased* measure of the autocovariance, we replace the $1/N$ term in (3.3) with $1/(N-L)$,

$$r^*_{xx}[L] = \frac{1}{N-L}\sum_{n=1}^{N-L}(x_{n+L}-\bar{x})(x_n-\bar{x}). \tag{3.4}$$

To examine the difference in the biased versus the unbiased autocovariance, let's compute both for the EEG data over a broad interval of lags:

```
[ac_u,lags]=xcorr(x-mean(x),2000,'unbiased');%Unbiased autocov,
plot(lags*dt,ac_u)                           %... plot it.
[ac_b,lags]=xcorr(x-mean(x),2000,'biased');  %Biased autocov,
hold on                                      %... plot it.
plot(lags*dt,ac_b, 'r')
hold off
xlabel('Lag [s]')                            %Label the axes.
ylabel('Autocovariance');
```

Here we compute the autocovariance for ±2000 lags (second input to the `xcorr` function) and choose either an `'unbiased'` or a `'biased'` autocovariance (third input to the `xcorr` function). We see in figure 3.5 the similarities and differences between these two estimates. At small lags (near 0 s), the biased and unbiased estimates of the autocovariance are similar. In this case, L is small, so the factor $1/(N-L)$ in the unbiased estimate is similar to the factor $1/N$ in the biased estimate. At large lags (away from 0 s), the biased and unbiased estimates of the autocovariance are quite different. As L approaches ±2 s, the biased estimate approaches zero.

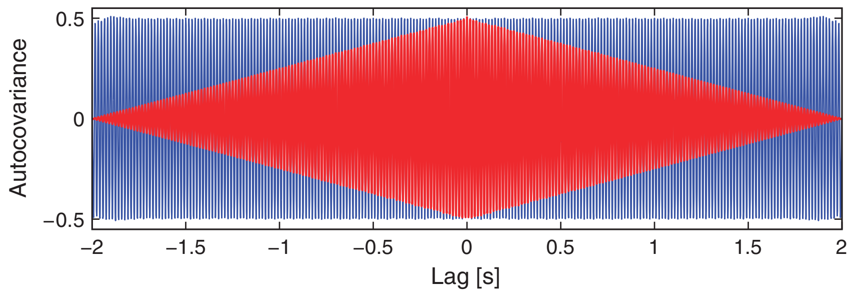

Figure 3.5
Autocovariance of the EEG for biased (*red*) and unbiased (*blue*) estimates over a wide range of lags.

Q: Why does the biased estimate of the autocovariance approach zero at large lags?

A: For concreteness, let's consider the case when $L = N - 2$. In this case L is large, and nearly equal to the number of points in the data (N). When L is large, we shift the time series x so that only a subset of indices overlap. Consider the $L = 2$ case in figure 3.3 and the extension to $L = N - 2$. Because we only compute the product

$(x_{n+L} - \bar{x})(x_n - \bar{x})$

for the overlapping indices of x_n and x_{n+L}, we only include two terms in the summation (3.3). The sum of these two terms is then divided by N, which results in a small number that approaches zero as L approaches N.

Compare this observation to the *unbiased* estimate of the autocovariance. In this example, we see that the unbiased estimate of the autocovariance remains large even as L approaches ± 2 s (see figure 3.5).

Q: Why does the unbiased estimate of the autocovariance remain large at large lags?

A: As in the biased case, consider $L = N - 2$. In this case, L is large, and we shift the time series x so that only a subset of indices overlap. Therefore, the product

$(x_{n+L} - \bar{x})(x_n - \bar{x})$

again only includes two terms in the summation (3.4). However, in the unbiased case, the sum of these terms is divided by $N - L = N - (N - 2) = 2$; as L approaches N, the term $N - L$ approaches zero. In this case, we find a balance between the summation of two terms and then a division by the number of terms in the sum (in this example, division by 2). This balance allows the unbiased estimate of the autocovariance to remain large as L approaches N.

Careful inspection of the blue curve in figure 3.5 reveals another feature of the unbiased estimate; the estimated values at large lags become more variable (look carefully at lags near ± 1.75 s and beyond). Increased variability at large lags occurs because as L approaches N, we have less data to compare in the assessment of the autocovariance. Notice that when $L = N - 1$, the estimate of the autocovariance utilizes only two data points from x (i.e., the product consists only of one term: $(x_N - \bar{x})(x_1 - \bar{x})$). We do not expect a reliable assessment of associations in the data with so few data points to compare.

With those observations, should we use the biased or unbiased estimator of the autocovariance? Statisticians typically prefer the biased estimator for a variety of reasons [8]. First, for many stationary processes, the mean squared error of the biased estimator is smaller than that of the unbiased estimator. The mean squared error depends on both the variance and the bias of the estimator:

$$\text{Mean squared error} = \text{Variance} + (\text{Bias})^2. \tag{3.5}$$

Although the biased estimator is "biased," the variability of the unbiased estimator is more harmful. We saw a hint of this increased variability in the unbiased estimator at large lags in figure 3.5. To make this observation more explicit, let's consider the lag $L = N - 1$, and compute the expression for the biased estimator (3.3),

$$\begin{aligned} r_{xx}\big[N-1\big] &= \frac{1}{N}\sum_{n=1}^{N-(N-1)} (x_{n+(N-1)} - \bar{x})(x_n - \bar{x}) \\ &= \frac{1}{N}\sum_{n=1}^{1}(x_{n+(N-1)} - \bar{x})(x_n - \bar{x}) \\ &= \frac{1}{N}(x_N - \bar{x})(x_1 - \bar{x}). \end{aligned} \tag{3.6}$$

The expression for the unbiased estimator (3.4) becomes

$$\begin{aligned} r^*_{xx}\big[N-1\big] &= \frac{1}{N-(N-1)}\sum_{n=1}^{N-(N-1)} (x_{n+N-1} - \bar{x})(x_n - \bar{x}) \\ &= \sum_{n=1}^{1}(x_{n+N-1} - \bar{x})(x_n - \bar{x}) \\ &= (x_N - \bar{x})(x_1 - \bar{x}). \end{aligned} \tag{3.7}$$

These two expressions reveal that at a large lag $L = N - 1$, the variance of the unbiased estimator (3.7) is N^2 times the variance of the biased estimator (3.6). The dramatic increase in variance of the unbiased estimator leads to unreliable estimates of the autocovariance at large lags. Also, we note that the biased estimator behaves nicely as L increases to N; we see from (3.6) that $r_{xx}\big[N-1\big]$ approaches zero when N is large. This is arguably the behavior we want. We have few data points to compare at large lags, and therefore an unreliable estimate of the autocovariance, so we're better off disregarding these values. For these reasons, we use the biased estimator; in this estimate, autocovariance values at large lags, which utilize less data and are typically noisier, are reduced.

For the EEG data of interest here, the unbiased estimator outperforms the biased estimator. For these data, which are dominated by a 60 Hz rhythm, there is significant autocovariance even at long lags. In this case, the biased estimator leads to an interpretation of decreasing autocovariance, even though that is not true. However, for most brain signals (not saturated by 60 Hz line noise), we expect the autocovariance to decrease in time.

3.2.4 Power Spectral Density, or Spectrum

There are many techniques to assess rhythmic activity in the EEG data. Here, we compute the *power spectral density*, or simply the spectrum, of x using a well-established technique, the Fourier transform.[2]

> The *spectrum* of the data x is the magnitude squared of the Fourier transform of x. The spectrum indicates the amplitude of rhythmic activity in x as a function of frequency.

There are many subtleties associated with computing and interpreting the spectrum. We explore some of them here; in doing so, we build our intuition for spectral analysis and our ability to deal with future, unforeseen circumstances in other data we encounter in research.

Computing the Spectrum. We start by presenting all the formulas and MATLAB code necessary to compute the spectrum of the data. Then throughout the chapter, we circle back and consider each step of the computation in detail.

We first need a formula for the discrete-time Fourier transform of the data x:

$$X_j = \sum_{n=1}^{N} x_n \exp(-2\pi i f_j t_n). \tag{3.8}$$

The Fourier transform computes the sum over all time indices $t_n = \Delta\{1, 2, 3, \ldots, N\}$ of the data x_n multiplied by sinusoids oscillating at a given frequency $f_j = j/T$, where $j = \{-N/2 + 1, -N/2 + 2, \ldots, N/2 - 1, N/2\}$. The result is a new quantity X_j, the signal as a function of frequency f_j rather than time t_n. The spectrum is then

$$S_{xx,j} = \frac{2\Delta^2}{T} X_j X_j^*, \tag{3.9}$$

which is the product of the Fourier transform of x with its complex conjugate (indicated by the superscript $*$), scaled by the sampling interval and the total duration of recording. The term $2\Delta^2/T$ is simply a numerical scaling (see appendix C at the end of this chapter). The units of the spectrum are, in this case, $(\mu V)^2$/Hz. Computing the spectrum in MATLAB requires only a few lines of code:

```
xf = fft(x-mean(x));              %Compute Fourier transform of x.
Sxx = 2*dt^2/T*(xf.*conj(xf));    %Compute spectrum.
Sxx = Sxx(1:length(x)/2+1);       %Ignore negative frequencies.
```

2. The power spectral density describes the extent to which sinusoids of a single frequency capture the structure of the data. To compute the power over any range of frequencies, we would integrate (or for discrete frequencies, sum) the spectrum over that frequency range.

```
df = 1/max(T);                  %Determine frequency resolution.
fNQ = 1/ dt / 2;                %Determine Nyquist frequency.
faxis = (0:df:fNQ);             %Construct frequency axis.

plot(faxis, Sxx)                %Plot spectrum vs frequency.
xlim([0 100])                   %Select frequency range.
xlabel('Frequency [Hz]')        %Label the axes.
ylabel('Power [ \muV^2/Hz]')
```

That's not so bad; the code to compute and display the spectrum fits in 12 lines (with spacing for aesthetics). The results of this computation are plotted in figure 3.6. Notice the large peak at 60 Hz. This peak is consistent with our visual inspection of the EEG data, in which we approximated a dominant rhythm at 60 Hz by counting the number of peaks that appeared in the voltage traces. So, our computation of the spectrum at least matches our initial expectation deduced from visual inspection of the data.

So, we've managed to compute and plot the spectrum in MATLAB, and our analysis results match our expectations. We could choose to stop here. But a danger persists: we've blindly entered MATLAB code and achieved an expected result. What are the frequency resolution and Nyquist frequency mentioned in the comments of the code? Maybe this procedure is fraught with pitfalls, and we simply got lucky in this case? Does the spectrum

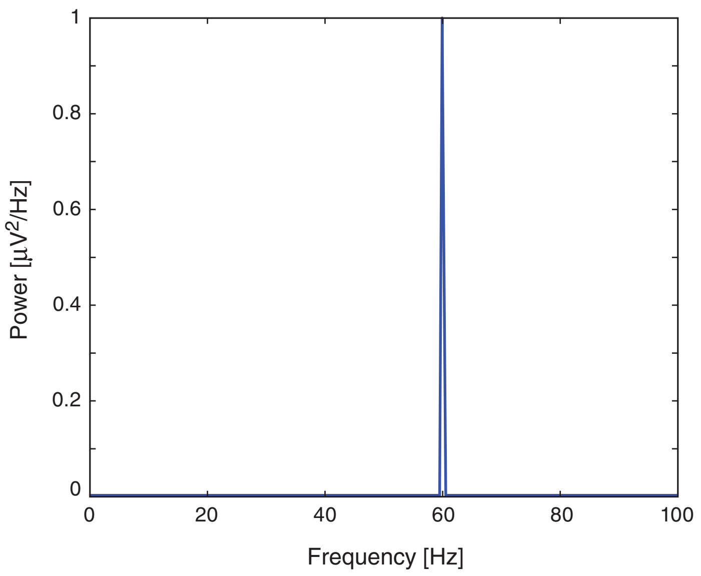

Figure 3.6
Power spectrum of EEG data. A large peak occurs at 60 Hz along the horizontal axis.

provide additional information that was not immediately uncovered in figure 3.6? How will we react and adapt when the spectrum results do not match our intuition? To answer these questions requires developing more intuition for the Fourier transform and spectrum. In the following sections, we examine equations (3.8) and (3.9), and the MATLAB code. In doing so, we explore some subtleties of this measure and strengthen our intuition for this measure's behavior. Building this intuition is perhaps the most important part for dealing with unforeseen circumstances arising in your own data.

What Is the Fourier Transform Actually Doing? The Fourier transform represents the data x as a linear combination of sinusoids with different frequencies. To see this, consider again (3.8):

$$X_j = \sum_{n=1}^{N} x_n \exp(-2\pi \, i f_j \, t_n).$$

What is this expression actually doing? Let's consider this equation piece by piece. The first term,

$$\sum_{n=1}^{N}$$

represents a sum over the indices of x_n. Because each subsequent index represents a step forward in time of Δ, this sum represents a sum over time. In fact, this sum extends from the first index ($n = 1$) to the last index ($n = N$) of the data, so the summation occurs over the entire duration of the recording. Adding the second term,

$$\sum_{n=1}^{N} x_n,$$

the summation now acts on the data x_n, so we're summing up the data over all indices or equivalently over the entire time of recording. The third term,

$$\exp(-2\pi \, i f_j \, t_n),$$

consists of an exponential (exp) operating on the product of five individual terms, each of which is a number. The first two numbers are simple: -2π. The third number is $i \equiv \sqrt{-1}$, a quantity representing an *imaginary unit*. The utility of the imaginary unit is that we can rewrite the exponential as the sum of a sine and cosine function. Remember from calculus the remarkable Euler's formula:

$$\exp(-2\pi \, i f_j \, t_n) = \cos(-2\pi f_j \, t_n) + i \sin(-2\pi f_j \, t_n).$$

Notice that the i now appears multiplying the sine term, and not inside of the arguments of either sinusoid. Both sinusoids operate on the product of 2π and two terms, the frequency f_j and the time t_n. Rewriting (3.8) using Euler's formula, we find

$$X_j = \Big(\sum_{n=1}^{N} x_n \cos(-2\pi f_j t_n)\Big) + i\Big(\sum_{n=1}^{N} x_n \sin(-2\pi f_j t_n)\Big), \tag{3.10}$$

where the summation is distributed over both terms.

Written in this way, the Fourier transform becomes easier to interpret. Let's consider the first term of (3.10). For each index n, we multiply the data x_n by a cosine function evaluated at frequency f_j and time t_n. We then sum the results of this multiplication over all indices from $n = 1$ to $n = N$, or equivalently, from time $t_n = \Delta$ to time $t_n = T$. So, we multiply the data by a cosine function at frequency f_j for each point in time, and sum the product over time. The second term of (3.10) is like the first, except we multiply the data x_n by a sine function.

We therefore think of the Fourier transform as comparing the data x to the sinusoids oscillating at frequency f_j. When the data and sinusoid at frequency f_j align the summation in the Fourier transform is large and the result X_j is a large number. When the data and sinusoid at frequency f_j do not align, the summation in the Fourier transform is small and X_j is a tiny number. To make these ideas more concrete, consider the examples in figure 3.7. In these simple examples, the data x are a perfect cosine with frequency 10 Hz

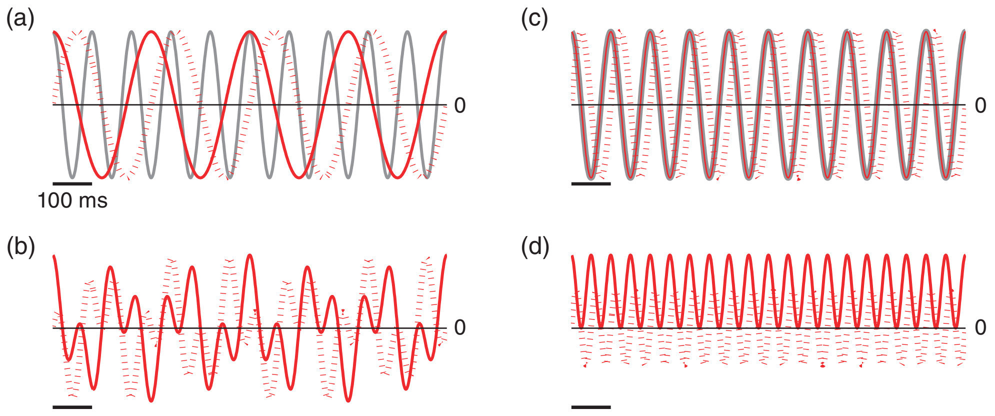

Figure 3.7
Artificial data, a 10 Hz cosine function, compared to sinusoids at different frequencies. Data are depicted by gray curves; cosine functions, by red solid curves; and sine functions, by red dotted curves. Scale bars indicate 100 ms. (*a*) Data compared to cosine and sine functions, each with frequency $f_j = 4$ Hz. (*b*) Product of data and 4 Hz cosine function, and product of data and 4 Hz sine function. (*c*) Data compared to 10 Hz cosine and 10 Hz sine functions. (*d*) Product of data and 10 Hz cosine function, and product of data and 10 Hz sine function.

(figure 3.7a). Choosing $f_j = 4$ Hz, we construct a sine and cosine function each oscillating at 4 Hz (figure 3.7a). Then, to perform the calculation of (3.10) we multiply the data x by the sinusoids at each point in time. The results of this computation are plotted in figure 3.7b. Notice that the products alternate between positive and negative values throughout time.

Q: In this case, what is the approximate value of X_j when $f_j = 4$ Hz?

A: From (3.10) we know that X_j is the summation over time of the product of x and the sinusoids. These products over time are plotted in figure 3.7b. Because each product alternates between positive and negative values roughly equally over time, the summation of the product over time is approximately zero. We therefore conclude that both the real part (i.e., the cosine term in (3.10)) and the imaginary part (i.e., the sine term in (3.10)) are small and $X_j \approx 0 + 0i$ when $f_j = 4$ Hz.

In this case, the sinusoids at frequency $f_j = 4$ Hz do not align with the data x, and X_j is nearly zero in both its real and imaginary parts. Now consider the case in which we choose $f_j = 10$ Hz. With this choice of f_j, the data x and the cosine function align perfectly (figure 3.7c). The product of the cosine function and the data is always non-negative (figure 3.7d), and therefore its summation over time results in a large positive number. In this case, the real part of X_j is large because the cosine function with frequency f_j =10 Hz and the data x match. In this sense, the Fourier transform reveals the dominant frequencies of the underlying time series.

Q: What is the approximate value of the *imaginary* part of X_j for $f_j = 10$ Hz? *Hint*: Consider the plot of the product of the sine function and data in figure 3.7d.

Relation of the Spectrum to the Autocovariance. We've now introduced two tools for assessing dependent structure in the EEG data: the autocovariance and the spectrum. Remarkably, these two measures are related in an important way.

The spectrum is the Fourier transform of the autocovariance.

The spectrum and autocovariance both assess dependent structure in the data but in different domains, the spectrum in the frequency domain and the autocovariance in the time domain. Notice that the spectrum $S_{xx,j}$ in (3.9) is a function of frequency index j, while the autocovariance $r_{xx}\big[L\big]$ in (3.3) is a function of time lag L. For the EEG data of interest here, the dominant 60 Hz rhythm manifests as periodicity in the autocovariance as a function of lag (figure 3.4a), and a peak in the spectrum as a function of frequency (figure 3.6). Although the two measures are related through the Fourier transform, each provides a different perspective on the dependent structure in the data. In practice, applying and visualizing both measures is often of use. See appendix A at the end of this chapter for a more mathematical discussion of this relation.

Here's an associated mathematical nugget. The spectrum is the Fourier transform of x multiplied by its complex conjugate. That produces a real number (i.e., the imaginary part is 0), which is convenient for plotting and visualization. The autocovariance is necessarily symmetric with respect to lag, that is, $r_{xx}\big[L\big] = r_{xx}\big[-L\big]$. In other words, the autocovariance is an *even function*. The Fourier transform of an even function is real, so the Fourier transform of the autocovariance is also a real number, consistent with the values of the spectrum.

Relation of the Spectrum to Multiple Linear Regression. As a final perspective, we consider a statistical modeling approach, with the goal of characterizing the rhythms that appear in the EEG data. To develop this approach, we first introduce *linear regression*. The idea of linear regression is to express a response variable at time n (call it x_n) in terms of predictor variables (call them $z_{1n}, z_{2n}, \ldots, z_{pn}$ for p predictor variables) as

$$x_n = \beta_0 + \beta_1 z_{1n} + \beta_2 z_{2n} + \cdots + \beta_p z_{pn} + \epsilon_n, \tag{3.11}$$

where ϵ_n is a random variable. This formulation represents the model we use in *multiple linear regression*. The term *multiple* comes from the multiple predictors used. The term *linear* expresses the fact that each predictor appears linearly in the expression (3.11). The challenge is to identify the unknown coefficients (the β's) given the observed response and predictor variables. We note that the case of $p = 1$ corresponds to simple linear regression; in that case, the goal is to fit the line,

$$x_n = \beta_0 + \beta_1 z_{1n} + \epsilon_n,$$

with slope β_1 and intercept β_0.

Let's consider application of multiple linear regression to the EEG data with a specific purpose: to remove the 60 Hz line noise. Recall that we found that the spectrum was dominated by a 60 Hz peak (figure 3.6). We expect this 60 Hz is due to electrical noise in the system, and this large noise peak may mask other interesting features occurring in the EEG data. Therefore, our analysis of the EEG data may benefit by removing this large 60 Hz signal. To do so, we first fit a multiple linear regression model to the data x_n with the following form,

$$x_n = \beta_0 + \beta_1 \sin(2\pi 60\, t_n) + \beta_2 \cos(2\pi 60\, t_n) + \epsilon_n, \tag{3.12}$$

where x_n is the EEG data at index n, t_n is the corresponding time axis at index n in units of seconds, and ϵ_n is a random variable.

Q: The model consists of three predictors. What are they?

A: The predictors are a constant term, a sine function at 60 Hz, and a cosine function at 60 Hz. Our goal is to solve for the unknown coefficients β_0, β_1, and β_2 given the EEG data.

This multiple linear regression is performed in MATLAB as follows:

```
%Define the model,
model = [ones(size(x)) sin(2*pi*60*t') cos(2*pi*60*t')];
%... and perform regression.
b = regress(x, model);
```

In the first line we define the model, which consists of a constant term (the variable `ones`), the 60 Hz sine function, and the 60 Hz cosine function. We note that the variable `t` is transposed so that `model` is a matrix with dimensions $[2000, 3]$. In the second line we ask MATLAB to perform the regression using the function `regress`, and return the β values in the single output variable `b`.

Q: Examine the output variable `b`. What do you find?

A: We find

```
b = -0.0000 0.9989 -0.0032
```

This result indicates that the constant predictor and the 60 Hz cosine predictor do not contribute much to the data; the values `b[1]` and `b[3]` are both near zero. However, the 60 Hz sine function makes a much large contribution; the value `b[2]` is near 1.

To see how well our multiple linear regression model fits the data, let's evaluate the model, and compare it to the original EEG data:

```
%Evaluate the model.
x_60Hz_modeled = b(1)+b(2)*sin(2*pi*60*t)+b(3)*cos(2*pi*60*t);
plot(t,x)                      %Plot the EEG data.
hold on                        %... freeze the graphics window,
plot(t,x_60Hz_modeled, 'r')    %... and plot the modeled EEG data,
hold off                       %... release the graphics window,
xlim([0.5 1])                    %... examine 0.5 s of data,
xlabel('Time [s]')               %... and label the axes.
ylabel('EEG and Modeled EEG [\muV)')
```

We plot the EEG data and model fit in figure 3.8. The model, which contains only three predictors, appears to do quite a good job at capturing the 60 Hz activity in the EEG data.

Q: The multiple linear regression model (3.12) is not a *perfect* fit to the data. Why?

A: The multiple linear regression model includes only three predictors. We expect the EEG data to consist of other features, including rhythms at other frequencies. Therefore, this simple model cannot account for all features of the EEG time series. That's okay. The goal of this model was to fit a particular rhythm, the 60 Hz activity, not every aspect of the data.

Now, let's examine a powerful application of the model. We've constructed the model to fit the 60 Hz line noise in the EEG data. Let's now use the model to *remove* the 60 Hz line noise from the original EEG data. To do so, we subtract the model fit from the EEG data and then plot the resulting new signal:

```
%Evaluate the model.
x_60Hz_modeled = b(1)+b(2)*sin(2*pi*60*t)+b(3)*cos(2*pi*60*t);
x_cleaned=x-x_60Hz_modeled';    %... remove it from the EEG data,
plot(t,x_cleaned);              %... and plot the result.
```

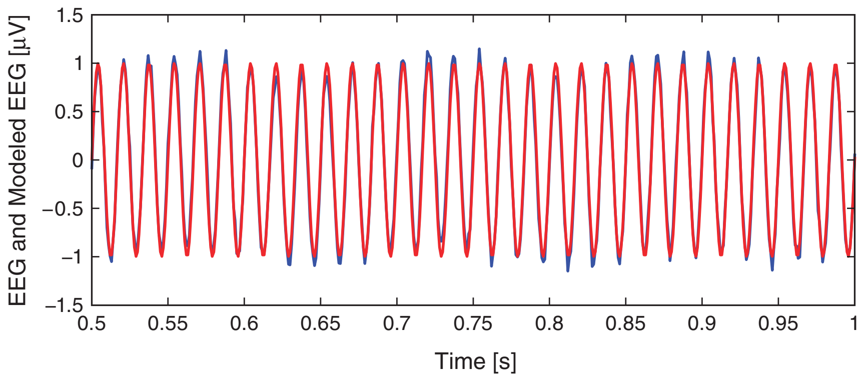

Figure 3.8
Comparison of 0.5 s segment of EEG data (*blue curve*) and multiple linear regression fit (*red curve*) shows good agreement.

In the first line we use the model fit (the β's in variable b) to define the modeled 60 Hz EEG activity. Then, in the second line, we subtract this modeled 60 Hz activity from the EEG data. We plot the results of the cleaned EEG data in figure 3.9.

Q: Consider the cleaned EEG data in figure 3.9. What activity do you now notice? Compare the EEG data in this plot to the original EEG data (which includes the 60 Hz noise) in figure 3.1. What's different?

In this example, we used multiple linear regression to model a particular rhythmic component of the EEG data, the 60 Hz activity. We may also use the model result to estimate the *power* at 60 Hz. In MATLAB,

```
Sxx_model_60Hz = b(2)^2+b(3)^2;
```

The power estimate from the model consists of two terms: the squared coefficient of the sine function (`b(2)^2`) plus the squared coefficient of the cosine function (`b(3)^2`). Note that the variable `Sxx_model_60Hz` has units of mV^2.

Q: Compare the power estimate from the model (the variable `Sxx_model_60Hz`) to the power spectral density at 60 Hz computed using the Fourier transform (see the MATLAB code following (3.9)). What do you find?

A: We note that the units of the power spectral density (variable `Sxx`) are mV^2/Hz, while the units of the power estimated in variable `Sxx_model_60Hz` are mV^2. To convert the power spectral density to (integrated) spectral power, we must integrate the variable `Sxx` over a frequency range. Here, we choose a 1 Hz interval centered at 60 Hz, which corresponds to a single index of the variable `faxis`; the frequency

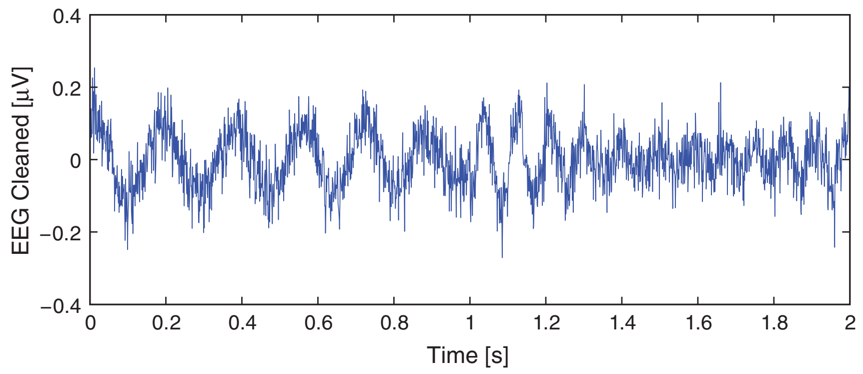

Figure 3.9
EEG data after removal of 60 Hz line noise using multiple linear regression.

resolution for these data is 0.5 Hz (see next section). Then the approximate integrated power over this 1 Hz interval is `Sxx(121)=0.9979`, identical to the value in `Sxx_model_60Hz`, and with the same units.

This example, in which we focused on the 60 Hz activity in the EEG, illustrates how we may use multiple linear regression to estimate the power. We could extend this procedure to include additional rhythms in the model beyond 60 Hz (e.g., sine and cosine functions at 1 Hz, 2 Hz, 3 Hz, etc.). In doing so, we would add more terms to the multiple linear regression model and have more β's to determine from the data. Multiple linear regression provides a way to decompose the EEG data into sine and cosine functions at different frequencies—just as we proposed to do using the Fourier transform—and then compute the power at each frequency. Using either multiple linear regression or the Fourier transform, we aim to decompose the EEG into sine and cosine functions oscillating at different frequencies.

Discrete Fourier Transform in MATLAB. Computing the spectrum of a signal x in MATLAB can be achieved in two simple steps. The first step is to compute the Fourier transform of x:

```
xf = fft(x-mean(x));                %The Fourier transform of x.
```

We subtract the mean from `x` before computing the Fourier transform. This is not necessary but often useful. For these neural data, we're not interested in the very slow (0 Hz) activity; instead, we're interested in rhythmic activity. By subtracting the mean, we eliminate this low-frequency activity from the subsequent analysis.

The second step is to compute the spectrum, the Fourier transform of x multiplied by its complex conjugate:

```
Sxx = 2*dt^2/T*(xf.*conj(xf));     %Compute the spectrum.
```

In this computation, we use the operation `.*` to multiply x by its complex conjugate. This operation multiplies the two vectors element by element.

Figure 3.10 plots the resulting variable `Sxx`. Upon examining the horizontal axis in this figure, we find it corresponds to the indices of x, beginning at index 1 and ending at index $N = 2000$. Computing the Fourier transform and multiplying by the complex conjugate does not change the length of the data x.

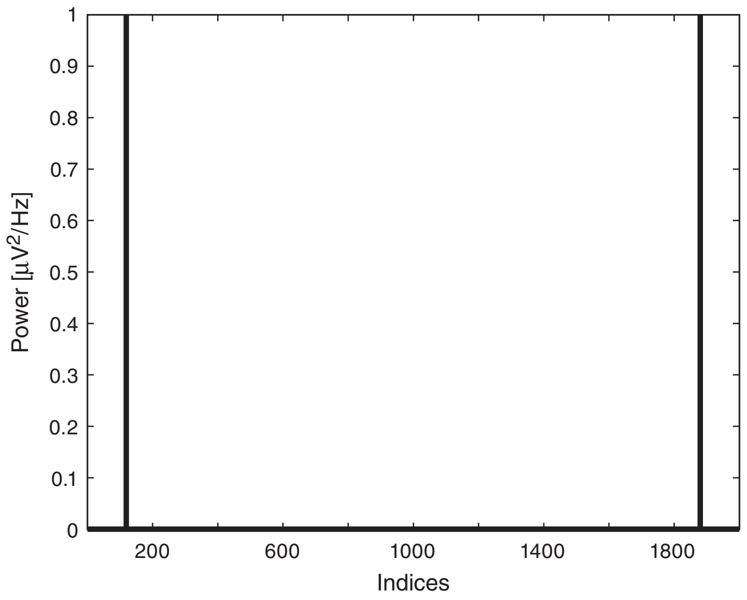

Figure 3.10
Power spectrum of EEG data for all indices. Horizontal axis is in indices rather than frequency.

Q: Confirm in MATLAB that the variables `x`, `xf`, and `Sxx` all have the same size.

Inspection of figure 3.10 reveals a strange characteristic: there are two large peaks, and the plot exhibits a particular symmetry. If we were to cut this plot from the printed page and fold the resulting piece of paper at index 1000, we would find that the two peaks in `Sxx` align; both peaks appear to be the same number of indices away from the center index 1000. This suggests that a redundancy occurs in the variable `Sxx`. In fact, this redundancy is due to the way MATLAB relates the indices and frequencies of `Sxx`. To define this relation requires two new quantities:

- the *frequency resolution*, $df = \frac{1}{T}$, or the reciprocal of the total recording duration;
- the *Nyquist frequency*, $f_{NQ} = \frac{f_0}{2} = \frac{1}{2\Delta}$, or half of the sampling frequency $f_0 = \frac{1}{\Delta}$.

For the clinical EEG data considered here, the total recording duration is 2 s ($T = 2$ s), so the frequency resolution $df = 1/(2 \text{ s}) = 0.5$ Hz. The sampling frequency f_0 is 1000 Hz, so $f_{NQ} = 1000/2$ Hz $= 500$ Hz. There's much more to say about both quantities, but for

Table 3.1
Relation between Vector `Sxx` Indices and Corresponding Frequencies.

`Sxx` index j	1	2	3	...	$N/2$
Frequency formula	0	df	$2\,df$	...	$f_{\mathrm{NQ}} - df$
Frequency value	0	0.5 Hz	1 Hz	...	499.5 Hz
`Sxx` index j	$N/2+1$	$N/2+2$	...	$N-1$	N
Frequency formula	f_{NQ}	$-(f_{\mathrm{NQ}} - df)$	...	$-2\,df$	$-df$
Frequency value	500 Hz	−499.5 Hz	...	−1 Hz	−0.5 Hz

now let's simply use both quantities to consider how MATLAB relates the indices and frequencies of the vector `Sxx` (see table 3.1). For the first half of `Sxx`, the frequency axis increases in steps of the frequency resolution df (here, 0.5 Hz) until reaching the Nyquist frequency f_{NQ} (here, 500 Hz). This maximal frequency occurs just past the halfway point of the indices, at index $j = N/2 + 1$. Beyond this index, the frequency axis is negative, and the magnitude of the frequency becomes smaller and smaller until the value $-df$ is reached at index $j = N$.

We may now utilize a useful property of the Fourier transform. When a signal is real (i.e., the signal has zero imaginary component), the negative frequencies in the spectrum are redundant. So, the power we observe at frequency f_0 is identical to the power we observe at frequency $-f_0$. For this reason, we can safely ignore the negative frequencies; these frequencies provide no additional information. Because the EEG data are real, we conclude that the negative frequencies in the variable `Sxx` are redundant and can be ignored. As a specific example, the value of `Sxx` at index $j = 3$ is the same as the value of `Sxx` at index $j = N - 1$; these indices correspond to frequencies $2df$ and $-2df$, respectively (see table 3.1). We therefore need only plot the variable `Sxx` for the positive frequencies, more specifically, from index 1 to index $N/2 + 1$. This conclusion matches our visual inspection of `Sxx` in figure 3.10.

> Data collected in the world are almost always real (have zero imaginary component), so the negative frequencies in the spectrum are usually redundant and can be ignored.

Given the total duration of the recording (T) and the sampling frequency (f_0) for the data, we can define the frequency axis for the spectrum `Sxx`. Now, to compute and plot the spectrum in MATLAB, we again utilize the code first introduced early in section 3.2.4:

```
xf = fft(x-mean(x));            %Compute Fourier transform of x.
Sxx = 2*dt^2/T*(xf.*conj(xf)); %Compute the spectrum.
Sxx = Sxx(1:N/2+1);            %Ignore negative frequencies.
```

```
df = 1/max(T);               %Determine the frequency resolution.
fNQ = 1/ dt / 2;             %Determine the Nyquist frequency.
faxis = (0:df:fNQ);          %Construct frequency axis.
plot(faxis, Sxx)             %Plot spectrum vs frequency.
```

Notice that in the third line we select the first half of the vector `Sxx` up to index $N/2+1$, thereby ignoring the redundant negative frequencies.

In the next two sections, we focus on interpreting and adjusting the quantities df and f_{NQ}. Doing so is critical to develop further an intuition for the spectrum.

Nyquist Frequency, f_{NQ}. The formula for the Nyquist frequency is

$$f_{\mathrm{NQ}} = \frac{f_0}{2}. \tag{3.13}$$

The Nyquist frequency is the highest frequency we can possibly hope to observe in the data. To illustrate this, let's consider a true EEG signal that consists of a very simple time series—a pure sinusoid that oscillates at some frequency f_s. Of course, we never observe the true signal. Instead, we observe a sampling of this signal, which depends on the sampling interval Δ. We consider three cases for different values of Δ. In the first case, we purchase a very expensive piece of equipment that can sample the true signal at a high rate, $f_0 \gg f_s$. In this case, we cover the true brain signal with many samples (figure 3.11, top), and given these samples, we can accurately reconstruct the underlying data.

Now, consider the case in which we purchase a cheaper piece of equipment that samples at a maximum rate equivalent to twice the frequency of the pure sinusoid: $f_0 = 2f_s$. In this case, we might collect sufficient samples to cover the underlying signal and approximate the oscillation frequency (figure 3.11, middle): the first sample resides on a peak of the sinusoid, the next sample on a trough, and so on. In this case, we collect two samples per cycle of the underlying true signal. Given only these sample points, we can connect the dots and still approximate the frequency of the true underlying sinusoid.

Q: For the sampling rate $f_0 = 2f_s$, consider the case in which the first sample occurs on a zero crossing of the sinusoid. At what point does the next sample occur? and the next sample? If you connect the dots in this case, what do you find?

Finally, consider the case where our equipment records at a sampling rate less than the frequency of the pure sinusoid signal: $f_0 < 2f_s$. (figure 3.11, bottom). Assuming the first sample occurs at a peak of the sinusoid, the next sample occurs not at a trough (that would correspond to a sampling rate $f_0 = 2f_s$) but instead just after the trough. Connecting the samples with lines in this case produces something horrifying, an oscillation occurring at a different, lower frequency. Notice what has happened in this case. Sampling the sinusoid

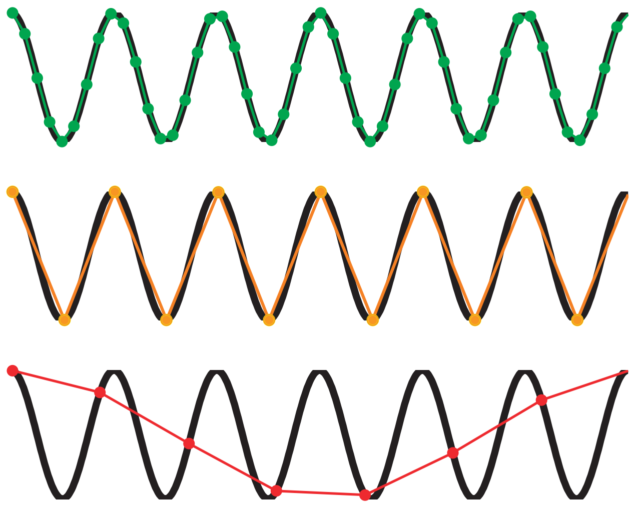

Figure 3.11

Sampling a sinusoid at different rates. When sampling rate is high enough (*top*), the sampled data provide a good approximation to the true data. When sampling rate is too low (*bottom*), the true high-frequency signal appears as a low-frequency oscillation.

at too low a frequency (i.e., at a frequency less than twice the signal's frequency, $f_0 < 2f_s$) causes this signal to manifest at a low-frequency upon sampling. This phenomenon—a high-frequency signal appearing as a low-frequency signal upon sampling—is known as *aliasing*. Once a signal has been aliased, it's impossible to distinguish from true signals oscillating at low frequencies.

To avoid aliasing, sample data at sufficiently high rates.

Typically, to prevent aliasing, recorded data are first analog-filtered before the digital sampling occurs. The analog filtering guarantees that activity at frequencies exceeding a threshold value (f_c, say) are dramatically reduced. The sampling rate can then be chosen to exceed this threshold value by at least a factor of 2 (i.e., $f_0 > 2f_c$). We note that in this case the EEG data were first analog-filtered at 200 Hz before digital sampling occurred at 1000 Hz. So, for our EEG data, aliasing is not a concern. For a more detailed investigation of aliasing, see appendix B at the end of this chapter.

Frequency Resolution, df. The frequency resolution is defined as

$$df = \frac{1}{T}, \tag{3.14}$$

where T is the total duration of the recording. For the EEG data in figure 3.1, $T = 2$ s, so the frequency resolution is $df = 1/(2\text{s}) = 0.5$ Hz.

Q: How do we improve the frequency resolution?

A: There's only one way to do it: increase T, that is, record more data. For example, if we demand a frequency resolution of 0.2 Hz, how much data must we record? We can rearrange (3.14) to solve for T,

$$T = \frac{1}{df} = \frac{1}{0.2\,\text{Hz}} = 5\,\text{s}. \tag{3.15}$$

So, record 5 s of data to obtain a frequency resolution of 0.2 Hz.

Q: We estimate the spectrum using the preceding MATLAB code. As we record more and more data, does the estimate of the spectrum improve?

A: Intuitively, you might answer yes. As we collect more and more data, we usually expect our estimate of a quantity (e.g., the mean or the standard deviation) to improve. However, that is not the case for the spectrum. As we collect more and more data, we acquire more and more points along the frequency axis (i.e., df becomes smaller). However, our estimate of the power at each frequency does not improve [8].

To gain some intuition for the frequency resolution formula (3.14), consider the case in which we collect T seconds of data. If the sampling interval is Δ, then we collect $N = T/\Delta$ data points; for example, for the EEG data of interest here, we collect $N = 2000$ data points. We know that the number of observations in the data equals the number of frequencies in the spectrum (where we now include negative frequencies); both the data vector `x` and the spectrum vector `Sxx` have length N. We also know that the maximum observable frequency in the spectrum, the Nyquist frequency, is fixed no matter how much data we collect. Recall that the Nyquist frequency depends only on the sampling interval: $f_{NQ} = 1/(2\Delta)$. Now, consider the case in which we increase T, or equivalently, increase N. As we collect more and more data, the maximum frequency remains fixed at the Nyquist frequency, while the length of the spectrum vector increases. We therefore need to fit more and more frequency values between 0 Hz and the Nyquist frequency as N increases (figure 3.12). This observation provides some intuition for the relation between the amount of data recorded (T or N) and the frequency resolution (df).

3.2.5 Reexamining the Spectrum—a Matter of Scale

Let's now return to the spectrum of the EEG data. As plotted in figure 3.6, the spectrum is dominated by a single peak at 60 Hz. Other, weaker rhythmic activity may occur in the data, but these features remain hidden from visual inspection because of the large 60 Hz peak; informally, we might state that the 60 Hz peak saturates the vertical scale in figure 3.6. One

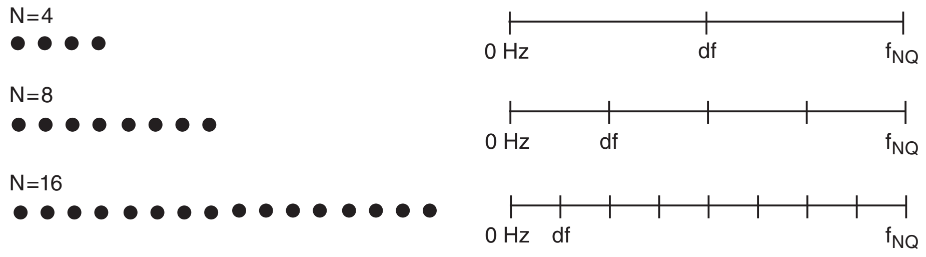

Figure 3.12

Cartoon representation of the relation between data duration and frequency resolution. Data (*left*) consist of different numbers of samples (N). As N increases, the number of values on the frequency axis increases (*right*), the maximal frequency (f_{NQ}) remains fixed, and the frequency resolution (df) decreases. Only non-negative frequencies are shown.

technique to emphasize lower-amplitude rhythms hidden by large-amplitude oscillations is to change the scale of spectrum to *decibels*. The decibel is a logarithmic scale and easily computed in MATLAB as follows:

```
plot(faxis, 10*log10(Sxx/max(Sxx)))  %Plot spectrum in decibels.
xlim([0 100])                        %Select frequency range.
ylim([-60 0])                        %Select decibel range.
xlabel('Frequency [Hz]')             %Label axes.
ylabel('Power [dB]')
```

To change to the decibel scale, we first divide the spectrum by the maximum value observed and then take the logarithm base 10 of this ratio and multiply the result by 10. The resulting spectrum is shown in figure 3.13a. The 60 Hz rhythm is still dominant and exhibits the most power.

Q: For this example, what is the value in decibels at 60 Hz?

A: Through our previous analysis, we know that the maximum value in the spectrum occurs at 60 Hz. By dividing the original spectrum by this maximum, we scale the spectrum at 60 Hz to a value of 1. The logarithm of 1 is 0, so we find a value of 0 at 60 Hz. Note that all other values are now smaller than 1 and therefore negative on the decibel scale.

Different conventions exist to define the decibel scale. Here we first divide by the maximum before computing the logarithm. Be sure to verify how the spectrum is scaled (if at all) to interpret the decibel axis.

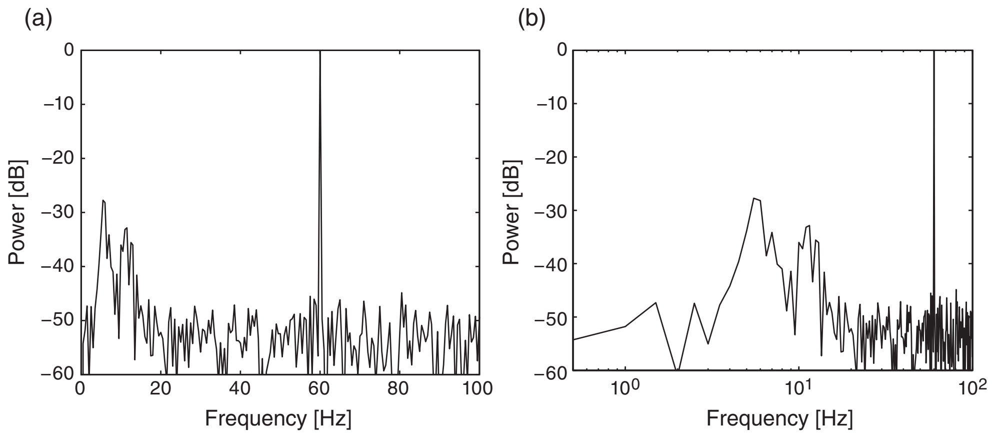

Figure 3.13
Power spectrum of EEG data on a decibel scale. Frequency axis is (*a*) linear or (*b*) logarithmic.

The decibel scale reveals new structure in the spectrum. In particular, two peaks have emerged at frequencies 5–15 Hz. These peaks are much weaker than the 60 Hz signal; both peaks are approximately 30 dB below the maximum at 60 Hz, or equivalently, three orders of magnitude weaker. Because these peaks are so small relative to the 60 Hz signal, neither was apparent in the original plot of the spectrum (figure 3.6). To further emphasize the low-frequency structure of the spectrum, we may also convert the frequency axis to a logarithmic scale:

```
semilogx(faxis, 10*log10(Sxx/max(Sxx))) %Log-log scale
xlim([df 100])                          %Select frequency range.
ylim([-60 0])                           %Select decibel range.
xlabel('Frequency [Hz]')                %Label axes.
ylabel('Power [dB]')
```

Notice the change in the first line to use the `semilogx` function in MATLAB. The resulting spectrum (in the log-log scale) is plotted in figure 3.13b. By using the logarithmic scale to stretch the low-frequency part of the horizontal axis, the two low-frequency peaks become more apparent. The changes in figure 3.13 compared to the original spectrum (figure 3.6) are purely cosmetic. However, these cosmetic changes have proved extremely useful. The two lower-frequency peaks were originally hidden from us, both in visual inspection of the raw data and in the original plot of the spectrum. In those cases, the large-amplitude 60 Hz activity masked the smaller-amplitude (three orders of magnitude smaller) rhythms.

3.2.6 The Spectrum Changing in Time—The Spectrogram

The results in figure 3.13 suggest that three rhythms appear in the EEG signal: 60 Hz, approximately 11 Hz, and approximately 6 Hz. Given only these results, we may reasonably conclude that these three rhythms appear simultaneously throughout the entire 2 s of EEG recording. That is an assumption we make in computing the spectrum of the entire 2 s interval. To further test this assumption in the EEG data, we compute a final quantity, the *spectrogram*. The idea of the spectrogram is to break up the time series into smaller intervals of data and then compute the spectrum in each interval. These intervals can be quite small and can even overlap. The result is the spectrum as a function of frequency and time.

Q: Consider the 2 s of EEG data. If we break up these data into smaller intervals of duration 1 s, what is the resulting frequency resolution of each interval? What is the Nyquist frequency of each interval?

To compute and display the spectrogram in MATLAB, we use the (aptly named) function `spectrogram`:

```
Fs = 1/dt;                    %Define the sampling frequency.
interval = round(Fs);         %Specify the interval size.
overlap = round(Fs*0.95);     %Specify the overlap of intervals.
nfft = round(Fs);             %Specify the FFT length.

%Compute the spectrogram,
[S,F,T,P] = spectrogram(x-mean(x),interval,overlap,nfft,Fs);
imagesc(T,F,10*log10(P))      %... and plot it,
colorbar                      %... with a colorbar,
axis xy                       %... and origin in lower left,
ylim([0 70])                  %... set the frequency range,
xlabel('Time [s]');           %... and label axes.
ylabel('Frequency [Hz]')
```

The function `spectrogram` accepts five arguments. Briefly, these arguments specify the data (x), the interval size (specified in indices and here set to 1 s), the overlap between intervals (here set to 95%), the number of points to use in the Fourier transform (here set to the interval length), and the sampling frequency. More information about these options can be found in the MATLAB documentation. Notice that in the first four lines we use the `round` function to enforce integer values for three of these inputs. The resulting spectrogram is plotted in figure 3.14.

10234_003 2016/7/20 10:21 Page 79 #31

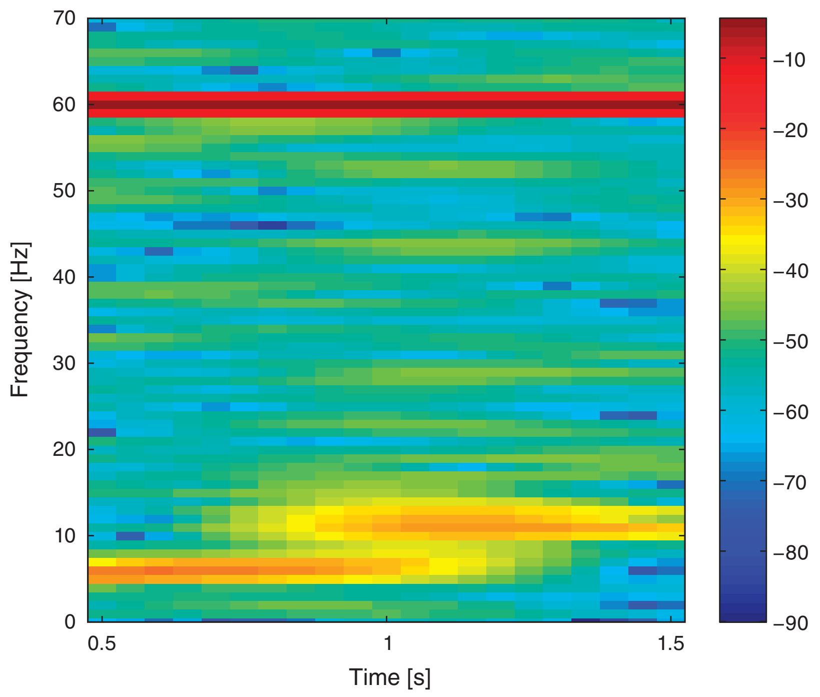

Figure 3.14
Spectrogram of EEG data. Power (in decibels, scale bar at right) is shown as a function of frequency and time.

Q: Consider the spectrogram in figure 3.14. What aspects of the spectrogram are consistent with our previous results? What aspects are new? Consider, in particular, the low-frequency rhythms and the conclusions deduced from this figure compared to figure 3.13.

A: The spectrogram displays the spectrum (in decibels) as a function of frequency (vertical axis) and time (horizontal axis). Values on the time axis indicate the center times of each 1 s window (e.g., 0.5 s corresponds to times [0, 1] s in the data). Intervals of high (low) values correspond to warm (cool) colors. Visual inspection immediately provides new insights into the observed EEG rhythms. First, we observe a band of high power at 60 Hz that persists for all time (red horizontal line in figure 3.14). This corresponds to the 60 Hz line noise present for the entire duration of the recording. Second, we observe intervals of increased power near 11 Hz and 6 Hz. Unlike the 60 Hz signal, the two low-frequency rhythms do not persist for the entire 2 s recording (as we may have incorrectly concluded from examination of the spectrum in figure 3.13). Instead, one weak rhythm (near 6 Hz) appears for the first half of the recording, while another weak rhythm (near 11 Hz) appears for the second half of the recording. Visualization via the spectrogram of how the rhythmic activity changes in time allows this important conclusion.

Summary

In this chapter, we analyzed 2 s of EEG data. We started with visual inspection of the EEG time series (figure 3.1). This is always the best place to start when analyzing new data and provides initial important intuition for the time series. Through the initial visual inspection, we concluded that rhythmic activity appeared and was dominated by a 60 Hz oscillation. Then, to characterize further the rhythmic activity, we computed two related quantities: the autocovariance and the spectrum. We found that rhythmic activity appeared in the autocovariance of the data. We then considered the spectrum. To do so, we first introduced the notion of the Fourier transform and discussed in detail how to compute the spectrum in MATLAB. We also defined two fundamental quantities—the frequency resolution and the Nyquist frequency—and explored how to manipulate these quantities. (We recommend you commit both quantities to memory. For every spectral analysis you encounter, ask: What is the frequency resolution? What is the Nyquist frequency?). We then considered how logarithmic scales can be used to emphasize features of the spectrum (figure 3.13), and finally how the spectrogram provides insight into spectral features that change in time (figure 3.14). We concluded that the EEG data are dominated by 60 Hz activity throughout the 2 s interval, and that weaker low-frequency activity emerges during two intervals: a 6 Hz rhythm from 0 s to 1 s, and an 11 Hz rhythm from 1 s to 2 s.

In this chapter, we only touched the surface of spectral analysis; many details and issues exist for further exploration. In chapter 4, we discuss the issues of windowing and zero padding. For those interested in exploring further, see [8, 9].

Problems

3.1. To improve the frequency resolution requires we record more data (i.e., increase T). Consider the case in which a collaborator demands a frequency resolution of 0.0001 Hz. How much data must be recorded? What are the disadvantages of recording the data required?

3.2. Explore the MATLAB function `periodogram`. How does this function compare to the computation of the spectrum described in this chapter?

3.3. A normalized measure of linear coupling in the time domain is the *autocorrelation*. Here is the formula for the autocorrelation evaluated at lag L,

$$\rho_{xx}[L] = \frac{r_{xx}[L]}{\sigma^2}.$$

Notice that the autocorrelation at lag L equals the autocovariance at lag L ($r_{xx}[L]$) divided by the variance of the signal (σ^2). Implement the autocorrelation in MATLAB, and apply it to the dataset considered in this chapter. How do the results of the autocovariance compare to the results of the autocorrelation?

3.4. Load the file `Ch3-EEG-2.mat`, available at

```
http://github.com/Mark-Kramer/Case-Studies-Kramer-Eden
```

into MATLAB. Then answer the following questions.

a. What is the sampling interval (Δ)? What is the total duration of the recording (T)? What is the frequency resolution (df)? What is the Nyquist frequency (f_{NQ})?
b. Plot the data and visually inspect them. Describe briefly (in a sentence or two) what rhythms, if any, you see in the data.
c. Plot the biased autocovariance versus lags. You will need to choose the maximum number of lags to investigate. What structure do you observe in the autocovariance, if any?
d. Plot the spectrum versus frequency. You may choose to plot the spectrum on a decibel scale, or not. Defend your choice.
e. Plot the spectrogram as a function of frequency and time. You will need to choose the interval size and the overlap between intervals. Do the rhythms in these data appear to change in time?
f. Interpret the autocovariance and spectrum, and describe the rhythms present in the signal. Compare your visual inspection of the data to the autocovariance and spectrum results. Do the analyses agree or disagree?

3.5. Load the file `Ch3-EEG-3.mat`, available at

```
http://github.com/Mark-Kramer/Case-Studies-Kramer-Eden
```

into MATLAB. Then answer the following questions.

a. What is the sampling interval (Δ)? What is the total duration of the recording (T)? What is the frequency resolution (df)? What is the Nyquist frequency (f_{NQ})?
b. Plot the data and visually inspect them. Describe briefly what rhythms, if any, you see in the data.

c. Plot the biased autocovariance versus lags. You will need to choose the maximum number of lags to investigate. What structure do you observe in the autocovariance, if any?
d. Plot the spectrum versus frequency. You may choose to plot the spectrum on a decibel scale, or not. Defend your choice.
e. Plot the spectrogram as a function of frequency and time. You will need to choose the interval size and the overlap between intervals. Do the rhythms in these data appear to change in time?
f. Interpret the autocovariance and spectrum, and describe the rhythms present in the signal. Compare your visual inspection of the data to the autocovariance and spectrum results. Do the analyses agree or disagree?

3.6. Load the file `Ch3-EEG-4.mat`, available at

```
http://github.com/Mark-Kramer/Case-Studies-Kramer-Eden
```

into MATLAB. Then answer the following questions.

a. What is the sampling interval (Δ)? What is the total duration of the recording (T)? What is the frequency resolution (df)? What is the Nyquist frequency (f_{NQ})?
b. Plot the data and visually inspect them. Describe briefly what rhythms, if any, you see in the data.
c. Plot the biased autocovariance versus lags. You will need to choose the maximum number of lags to investigate. What structure do you observe in the autocovariance, if any?
d. Plot the spectrum versus frequency. You may choose to plot the spectrum on a decibel scale, or not. Defend your choice.
e. Plot the spectrogram as a function of frequency and time. You will need to choose the interval size and the overlap between intervals. Do the rhythms in these data appear to change in time?
f. Interpret the autocovariance and spectrum, and describe the rhythms present in the signal. Compare your visual inspection of the data to the autocovariance and spectrum results. Do the analyses agree or disagree?

3.7. Load the file `Ch3-EEG-5.mat`, available at

```
http://github.com/Mark-Kramer/Case-Studies-Kramer-Eden
```

into MATLAB. Then answer the following questions.

a. What is the sampling interval (Δ)? What is the total duration of the recording (T)? What is the frequency resolution (df)? What is the Nyquist frequency (f_{NQ})?

b. Plot the data and visually inspect them. Describe briefly what rhythms, if any, you see in the data.
c. Plot the biased autocovariance versus lags. You will need to choose the maximum number of lags to investigate. What structure do you observe in the autocovariance, if any?
d. Plot the spectrum versus frequency. You may choose to plot the spectrum on a decibel scale, or not. Defend your choice.
e. Plot the spectrogram as a function of frequency and time. You will need to choose the interval size and the overlap between intervals. Do the rhythms in these data appear to change in time?
f. Interpret the autocovariance and spectrum, and describe the rhythms present in the signal. Compare your visual inspection of the data to the autocovariance and spectrum results. Do the analyses agree or disagree?

3.8. Consider the function $x(t) = \sin(2\pi t^2)$. Simulate this function in MATLAB using a sampling interval of $\Delta = 0.001$ ms, and $t = (0, 10)$ s. Analyze these data as you would an EEG time series collected in an experiment. Compute the spectrum, and compute the spectrogram. Explain the results you find in each case, and how these results compare to your expectations.

3.9. (*Advanced*) As stated in this chapter, to increase the frequency resolution of the spectrum, we must record more data (i.e., increase T). Consider a scenario in which the experiment that produced the data of duration T is now inaccessible. We therefore cannot perform another experiment of longer duration to improve the frequency resolution. So, we instead consider a work-around to improve the frequency resolution: let's simply append the data end to end. The result is a new dataset of duration $2T$. Note that the new dataset is redundant; the second half of the new dataset is a duplicate of the first half. Does this improve the frequency resolution in a meaningful way? Can we use this stratagem to append the observed data K times and thereby improve the frequency resolution by a factor of K?

Appendix A: The Spectrum and Autocovariance

To show that the spectrum is the Fourier transform of the autocovariance, we first make some reasonable assumptions that help simplify the analysis. Let's assume that x has zero mean (i.e., $\bar{x} = 0$), which we can always enforce by subtracting the mean from x as a first step in the analysis. In addition, let's assume we can evaluate x at any time index from negative infinity to positive infinity. Then, substituting (3.8) into (3.9), we get the sample power spectral density,

$$S_{xx,j} = \frac{2\Delta^2}{T} \Big(\sum_n x_n \exp(-2\pi i f_j t_n) \Big) \Big(\sum_m x_m^* \exp(2\pi i f_j t_m) \Big). \tag{3.16}$$

In the second term in parentheses, we set the dummy time index to another symbol, m. This second term is the complex conjugate of the Fourier transform of the data. The impact of the complex conjugate is that we replace i with $-i$, and x_m with x_m^*. Because the EEG data are real, $x_m^* = x_m$. Notice that we do not include the limits on the summations, which range from $-\infty$ to ∞. Replacing the expressions for f_j, t_n, and t_m with their definitions and simplifying, we rearrange the expression to find

$$S_{xx,j} = \frac{2\Delta^2}{T} \sum_n \sum_m x_n x_m \exp(-\frac{2\pi i}{N} j(n-m)). \tag{3.17}$$

Let's now define a new variable,

$$l = n - m. \tag{3.18}$$

Then

$$S_{xx,j} = \frac{2\Delta^2}{\Delta} \sum_l \Big(\frac{1}{N} \sum_m x_{m+l}\, x_m \Big) \exp(-\frac{2\pi i}{N} j\, l), \tag{3.19}$$

where we have replaced the n with l, used $T = N\Delta$, and reorganized the terms. Comparing the term in parentheses to the equation for the biased autocovariance (3.3), we see that the two are equivalent; remember that $\bar{x} = 0$. Therefore,

$$\begin{aligned} S_{xx,j} &= 2\Delta \sum_l r_{xx}[l] \exp(-\frac{2\pi i}{N} j\, l) \\ &= 2\Delta \sum_l r_{xx}[l] \exp(-\frac{2\pi i}{N\Delta} j \Delta l) \\ &= 2\Delta \sum_l r_{xx}[l] \exp(-2\pi i \frac{j}{T} \Delta l) \\ &= 2\Delta \sum_l r_{xx}[l] \exp(-2\pi i f_j\, t_l). \end{aligned}$$

Comparing this last expression to the equation for the Fourier transform (3.8), we find that the sample power spectral density ($S_{xx,j}$) is proportional to the Fourier transform of the sample autocovariance ($r_{xx}[l]$).

Appendix B: Aliasing

We illustrated in figure 3.11 the concept of aliasing. In this appendix, we further investigate the Fourier transform and the impact of sampling. Consider a signal $y(t)$. As illustrated in figure 3.2, we do not observe the signal directly. Instead, we only observe a discrete sampling in time of this signal. Let's make this sampling explicit, and write the observed signal $s(t)$ as

$$s(t) = y(t) \sum_{n \to -\infty}^{\infty} \delta(t - n\Delta), \tag{3.20}$$

where $\delta(t)$ is the Kronecker delta, and Δ is the sampling interval. This equation represents our discrete observations of $y(t)$ at integer multiples of the sampling interval Δ. Here, we assume an infinite number of observations (into the past and future).

To compute the Fourier transform of $s(t)$, we need to evaluate the Fourier transform of the product of the two functions in (3.20): the signal $y(t)$ and the summation of Kronecker delta functions. The Fourier transform of the product of two functions equals the convolution of the Fourier transform of each function (see chapter 4). We use this fact to express the Fourier transform of $s(t)$ as

$$FT\big[s(t)\big] = FT\big[y(t)\big] \star FT\big[\sum_{n \to -\infty}^{\infty} \delta(t - n\Delta)\big], \tag{3.21}$$

where $FT\big[a\big]$ denotes the Fourier transform of a, and $A \star B$ indicates the convolution of A and B. Let's define $Y(f) = FT\big[y(t)\big]$, and $S(f) = FT\big[s(t)\big]$. We can evaluate the Fourier transform of the repeated Kronecker delta functions as

$$\begin{aligned} FT\big[\sum_{n \to -\infty}^{\infty} \delta(t - n\Delta)\big] &= \frac{1}{\Delta} \sum_{n \to -\infty}^{\infty} \delta(f - \frac{n}{\Delta}) \\ &= \frac{1}{\Delta} \sum_{n \to -\infty}^{\infty} \delta(f - nf_0), \end{aligned}$$

where we've replaced $1/\Delta$ with the sampling frequency f_0. Then the Fourier transform of the observed signal $s(t)$ becomes

$$\begin{aligned} S(f) &= FT\big[s(t)\big], \\ &= Y(f) \star \frac{1}{\Delta} \sum_{n \to -\infty}^{\infty} \delta(f - nf_0) \end{aligned}$$

$$= \frac{1}{\Delta} \sum_{n \to -\infty}^{\infty} Y(f - nf_0)$$

$$\propto \sum_{n \to -\infty}^{\infty} Y(f - nf_0), \tag{3.22}$$

where, in the last line, we've dropped the leading constant term for convenience of presentation. This result relates the Fourier transform of the unobserved signal $y(t)$ to the Fourier transform of the observed (i.e., sampled) signal $s(t)$. Note that $S(f)$ is an infinite summation of terms. To make this example more concrete, let's consider the case that the (unobserved) signal is a cosine function at frequency f_*,

$$y(t) = \cos(2\pi f_* t),$$

with corresponding Fourier transform,

$$Y(f) \propto \delta(f - f_*) + \delta(f + f_*). \tag{3.23}$$

This result is perhaps intuitive; when the data consist of a pure sinusoid, we expect the power (in the ideal case) to be concentrated at the sinusoid's frequency. We usually ignore the negative frequency component because it is redundant, but here we've been careful to include both the positive frequency $f = f_*$ and the negative frequency $f = -f_*$ in the representation of the Fourier transform. We've also ignored the constant terms in (3.23) to focus on the quantities of interest. Substituting the expression for $Y(f)$ in (3.23) into the expression for $S(f)$ in (3.22) we find

$$S(f) \propto \sum_{n \to -\infty}^{\infty} \delta(f - f_* - nf_0) + \delta(f + f_* - nf_0). \tag{3.24}$$

This result relates the Fourier transform of the observed signal $S(f)$ at frequency f to the peaks we expect at the positive and negative frequencies of the sinusoid f_*. However, we observe not only these expected peaks (when $n = 0$) but also an infinite number of other peaks (for $n \neq 0$).

To explore the impact of these additional peaks, consider a scenario in which the sinusoid signal has frequency $f_* = 20$ Hz, and the sampling occurs with $\Delta = 0.01$ s or $f_0 = 1/\Delta = 100$ Hz. In this case,

$$S(f) \propto \sum_{n \to -\infty}^{\infty} \delta(f - 20 - n100) + \delta(f + 20 - n100),$$

and we find that $S(f)$ is nonzero when

$$f = \begin{cases} \mathbf{20}, -\mathbf{20}, & \text{if } n = 0, \\ 120, 80, & \text{if } n = 1, \\ -80, -120, & \text{if } n = -1, \\ \ldots . \end{cases}$$

At this sampling frequency, the Nyquist frequency is $f_0/2 = 50$ Hz. We find that only the $n = 0$ case provides frequencies f with magnitude less than the Nyquist frequency. All other frequencies appear outside of the interval bounded by the Nyquist frequencies $(-50, 50)$ Hz and therefore do not manifest in our analysis of $S(f)$. This is the scenario we desire: the Fourier transform of $s(t)$ is concentrated at the frequencies of the sinusoid $y(t)$.

Now, consider a second scenario in which the sinusoid signal has frequency $f_* = 80$ Hz, and the sampling again occurs at $f_0 = 100$ Hz. In this case,

$$S(f) \propto \sum_{n \to -\infty}^{\infty} \delta(f - 80 - n100) + \delta(f + 80 - n100), \tag{3.25}$$

and we find that $S(f)$ is nonzero when

$$f = \begin{cases} 80, -80, & \text{if } n = 0, \\ 180, \mathbf{20}, & \text{if } n = 1, \\ -\mathbf{20}, -180, & \text{if } n = -1, \\ \ldots . \end{cases} \tag{3.26}$$

In this case, at $n = 0$, both the positive and negative frequencies occur beyond the Nyquist frequency; we therefore do not observe these 80 Hz peaks in the Fourier transform of $s(t)$. However, we do find for $n = 1$ a nonzero value for $S(f)$ at $f = 20$ Hz. In this case, the negative frequency component has folded into the observable frequency range (here, from 0 Hz to 50 Hz). Similarly, for $n = -1$, we find that the positive frequency component has folded into the observable negative frequency range (here, from -50 Hz to 0 Hz). Therefore, when we compute the Fourier transform of the 80 Hz sinusoid, we instead find power concentrated at the much lower frequencies of ± 20 Hz. The impact of aliasing is to shift activity in the signal that occurs beyond the Nyquist frequency (i.e., high-frequency components of $y(t)$) into the observable frequency range (from negative Nyquist frequency to positive Nyquist frequency).

Appendix C: Numerical Scaling of the Spectrum

We defined in (3.9) the sample power spectral density, which we rewrite here for convenience,

$$S_{xx,j} = \frac{2\Delta^2}{T} X_j X_j^*.$$

In this appendix, we consider some motivation for the leading term $\frac{2\Delta^2}{T}$. We consider separately the three components of this leading term: Δ^2, $1/T$, and 2.

Term: Δ^2. Throughout this chapter, we've focused on the discrete-time Fourier transform, defined in (3.8). The discrete-time Fourier transform is an approximation to the continuous-time Fourier transform,

$$X(f) = \int_{-\infty}^{\infty} x(t) \exp(-2\pi i f t)\, dt. \tag{3.27}$$

In this equation, we evaluate the time integral of the continuous signal $x(t)$ multiplied by the continuous-time complex exponentials $\exp(-2\pi i f t)$. Let's consider a Riemann sum approximation to this integral. To do so, we replace the integral with a summation over discrete steps in time, and compute the area of the integrand between each step,

$$X(f) = \sum_{n=-\infty}^{\infty} \Big(x(\Delta n) \exp(-2\pi i f \Delta n) \Big) \Delta.$$

Notice that we replace the continuous-time variable t with the discrete-time Δn, where n includes all integers. We may think of $\Big(x(\Delta n) \exp(-2\pi i f\, \Delta n)\Big)$ as the height of each rectangle, and Δ as the width of each rectangle in the Riemann sum. If we truncate this summation to include a finite number of time points, we get

$$X(f) \approx \Big(\sum_{n=1}^{N} x_n \exp(-2\pi i f_j t_n) \Big) \Delta,$$

where x_n is the signal evaluated at discrete times, t_n is the discrete time, and f_j the discrete frequency. The impact of this truncation is to distort the spectrum (e.g., to produce side lobes; see chapter 4). We then conclude

$$X(f) \approx \Delta X_j,$$

which shows that the continuous-time Fourier transform is approximated by the discrete-time Fourier transform (3.8) multiplied by the sampling interval Δ. Then

$$\begin{aligned} X(f)X(f)^* &\approx \big(\Delta X_j\big)\big(\Delta X_j^*\big) \\ &\approx \Delta^2 X_j X_j^*. \end{aligned} \tag{3.28}$$

This result shows that the continuous-time Fourier transform multiplied by its complex conjugate is approximated by the corresponding product of the discrete-time Fourier transform multiplied by the constant Δ^2.

Term: $1/T$. To understand the scaling of the spectrum by $1/T$, we begin with (3.8), the definition of the discrete-time Fourier transform, with some additional notation,

$$X_{j,N} = \sum_{n=1}^{N} x_n \exp(-2\pi i f_j t_n),$$

where we've included the subscript N on $X_{j,N}$ to indicate the total number of time points (or equivalently the total time $T = N\Delta$) in the summation. We now examine how $X_{j,N}$ changes as we increase T. Consider the simple case in which the data x_n are a sinusoid. For the appropriate choice of f_j, we find an excellent match between the data x_n and the real or imaginary component of $\exp(-2\pi i f_j t_n)$ (figure 3.15a). Multiplying these two components point by point, and summing over time, results in $X_{j,N} = C$, where C is a number.

Now, imagine we increase T by a factor of 2. We assume that the data x are stationary, so that we observe the same sinusoid, now for a duration of $2T$. What is $X_{j,2N}$? Again, we find an excellent match between the data and a sinusoid of frequency f_j. To evaluate the product summed over time $2T$, we divide the sum into two intervals. We know that the first half of the summation (from $n = 1$ to $n = N$) produces a value of C; that's what we found for $X_{j,N}$. Now, the second half of the summation is identical to the first half (figure 3.15b).

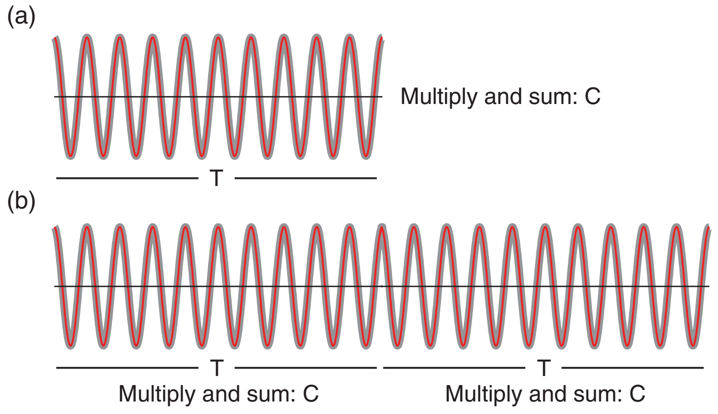

Figure 3.15

Impact of duration on the Fourier transform. (*a*) Artificial data *(gray curve)* are well matched by a sinusoid with appropriate frequency (*red curve*). Multiplying these two components element by element and summing over all time T yields a number, C. (*b*) Same data observed for a duration of $2T$. Element by element products of the data and matching sinusoid in first and second intervals of T are each equal to C, so total product over duration $2T$ is $2C$.

Therefore, we expect the second half of the summation (from $n = N + 1$ to $n = 2N$) to also equal C. We conclude that $X_{j,2N} = 2X_{j,N} = 2C$. In a similar way, $X_{j,3N} = 3C$, $X_{j,4N} = 4C$, and so on. The intuition is that because we observe the same signal for a longer time, the summed product increases.

To choose the scaling of the spectrum requires one additional fact:

$$\sigma^2 = \int_0^\infty S_{xx}(f)\, df, \tag{3.29}$$

which states that the (theoretical) variance σ^2 of a signal x equals the (theoretical) power spectral density $S_{xx}(f)$ of x integrated over all positive frequencies. This relation is consistent with the fact that the autocovariance is the inverse Fourier transform of the spectrum. To see this, consider the inverse Fourier transform of the spectrum evaluated at lag 0. In addition, this relation captures the idea that the spectrum decomposes the signal variance into different frequency bands; by summing this decomposition over all frequency bands, we calculate the total variance. To preserve this relation, we must choose an appropriate scaling of the spectrum. Let's replace $S_{xx}(f)$ in (3.29) with the discrete spectrum (3.9) and approximate the integral over frequencies using Riemann sums:

$$\begin{aligned}\sigma^2 &= \sum_j \frac{2\Delta^2}{T} X_{j,N} X^*_{j,N}\, df \\ &= \sum_j \frac{2\Delta^2}{T} X_{j,N} X^*_{j,N} \left(\frac{1}{T}\right) \\ &= \frac{2\Delta^2}{T^2} \sum_j X_{j,N} X^*_{j,N}.\end{aligned} \tag{3.30}$$

Notice that we've rewritten the frequency resolution df in terms of the total observation time T and that we've explicitly indicated the total number of time points (or equivalently, the total time $T = N\Delta$) in the Fourier transform $X_{j,N}$.

Now consider (3.30) and the impact of increasing the observation time. Doing so leaves the variance unaltered; if we observe a sinusoid for 1 s, and then the same sinusoid for K s, the variance is the same. Therefore, we require that the right–hand side of (3.30) remain unaltered. Let's replace $T \to KT$ and $N \to KN$ in (3.30),

$$\begin{aligned}\frac{2\Delta^2}{(KT)^2} \sum_j X_{j,KN} X^*_{j,KN} &= \frac{2\Delta^2}{K^2T^2} \sum_j \left(K X_{j,N}\right)\left(K X^*_{j,N}\right) \\ &= \frac{2\Delta^2}{T^2} \sum_j X_{j,N} X^*_{j,N} \\ &= \sigma^2.\end{aligned} \tag{3.31}$$

The factor of K^2 in the denominator (due to the T^2 term in the denominator) cancels the factor of K^2 in the numerator (due to the product of the Fourier transforms of x). Altering the duration of the recording does not alter the integrated spectrum; it remains equal to the variance. The scaling by $1/T$ preserves this relation.

Term: 2. The factor of 2 in the spectrum (3.9) is a conventional choice when the negative frequencies are ignored. We can think of this factor as accounting for the omitted negative frequencies. We note that this choice makes the variance equal to the spectrum integrated over positive frequencies in (3.29). If we were instead to remove this factor of 2 from the spectrum, we would need to integrate the spectrum of all positive and negative frequencies in (3.29) to equal the variance.

MATLAB Code. The ideas in this appendix are explored in the following MATLAB code. In this example, we generate artificial data as Gaussian noise. In your own exploration of this code, consider how the results change with the total duration of the recording (variable `T`) or nature of the signal (variable `d`).

```
dt = 0.001;          %Define the sampling interval.
T = 4;               %Define the total duration of the recording.
t = (dt:dt:T);       %Define a time axis.
N = length(t);       %Define the number of samples.
d = randn(N,1);      %Create artificial data.
d = d - mean(d);     %Set mean to 0.

pow = 2*dt^2/T*(fft(d).*conj(fft(d)));  %Compute the power,
%... then sum spectrum over positive frequencies, multiply by df.
POW = sum(pow(1:N/2+1))*(1/T);
vr  = var(d);        %Compute variance of data.

%Print the results.
fprintf(['Summed spectrum ' num2str(POW) '\n'])
fprintf(['Variance     ' num2str(vr)  '\n'])
fprintf(['Difference   ' num2str(POW - vr) '\n'])
```