Preface

Signal processing can be broadly defined as the application of analog or digital techniques to improve the utility of data. In biomedical engineering applications, improved utility usually means that the data provide better diagnostic information. Analog techniques are sometimes applied to data represented by a time-varying electrical signal, but most signal-processing methods take place in the digital domain where data are represented as discrete numbers. These numbers may represent a time-varying signal, or an image. This book deals exclusively with signal processing of digital data, although the Introduction briefly describes analog processes commonly found in medical devices.

This book should be of interest to a broad spectrum of engineers, but it is written specifically for biomedical engineers (called bioengineers in some circles). Although the applications are different, the signal-processing methodology used by biomedical engineers is identical to that used by electrical and computer engineers. The difference for biomedical engineers is in the approach and motivation for learning signal-processing technology. A computer engineer may be required to develop or modify signal-processing tools, but for biomedical engineers these techniques are tools to be used. For the biomedical engineer, a detailed understanding of the underlying theory, while always of value, is not usually needed. What is needed is an understanding of what these tools do and how they can be employed to meet biomedical challenges. Considering the broad range of knowledge required to be effective in biomedical engineering, encompassing both medical and engineering domains, an in-depth understanding of all of the useful technology is not realistic. It is most important to know what tools are available, have a good understanding of what they do (if not how they do it), be aware of the most likely pitfalls and misapplications, and know how to implement these tools given available software packages. The basic concept of this book is that just as the cardiologist can benefit from an oscilloscope-type display of the ECG without a deep understanding of electronics, so a biomedical engineer can benefit from advanced signal-processing tools without always understanding the details of the underlying mathematics.

In line with this philosophy, most of the concepts covered in this book can be broken down into two components. The first component is a presentation of a general understanding of the approach sufficient to allow intelligent application of the concepts. Sometimes this includes a description of the underlying mathematical principles. The second section describes how these tools can be implemented using the MATLAB® software package and several of its toolboxes.

This book was originally written for a single-semester course combining signal and image processing. Classroom experience indicates that this ambitious objective is possible for most graduate formats, although eliminating a few topics may be desirable. For example, some of the introductory or basic material covered in Chapters 1 and 2 could be skipped or treated lightly for students with the appropriate prerequisites. In addition, topics such as Advanced Spectral Methods (Chapter 5), Time–Frequency Analysis (Chapter 6), Wavelets (Chapter 7), Advanced Filters (Chapter 8), and Multivariate Analysis (Chapter 9) are pedagogically independent and can be covered as desired without affecting the other material. With the inclusion of Chapters

10 and 11 on nonlinear signal processing, the material on image processing could be deleted and the course restricted to one-dimensional signal processing.

Nonlinear signal analysis is an emerging area within signal processing that recognizes that real biomedical signals do not always conform to our assumptions of linearity. For such signals, nonlinear analysis can reveal additional information that linear analysis cannot. New techniques and algorithms are being continuously added, but Chapters 10 and 11 provide a foundation for those interested in nonlinear signal analyses. These techniques have gained a foothold in the area of heart rate and EEG analysis, where there is evidence of nonlinear behavior. Other new biomedical applications include DNA sequence analysis and speech processing. Nonlinear signal processing can only grow in importance as the techniques are improved offering greater speed and better immunity to noise, and as new biomedical applications are discovered.

This third edition of this textbook was undertaken to improve clarity and understanding by expanding both examples and problems. Although much of the material covered here will be new to most students, the book is not intended as an "introductory" text since the goal is to provide a working knowledge of the major signal-processing methods without the need for additional course work. The challenge of covering a broad range of topics at a useful, working depth is motivated by current trends in biomedical engineering education, particularly at the graduate level where a comprehensive education must be attained with a minimum number of courses. This has led to the development of "core" courses to be taken by all students. This book was written for just such a core course in the Graduate Program of Biomedical Engineering of Rutgers University. It is also quite suitable for an upper-level undergraduate course and would also be of value for students in other disciplines that would benefit from a working knowledge of signal and image processing.

It is not possible to cover such a broad spectrum of material to a depth that enables productive application without heavy reliance on MATLAB-based examples and problems. In this regard, the book assumes that the student has some knowledge of MATLAB programming and has the basic MATLAB software package at hand, including the Signal Processing and Image Processing Toolboxes. (MATLAB also has toolboxes for wavelets and adaptive neural nets, but these sections are written so as not to require additional toolboxes, primarily to keep the number of required toolboxes to a minimum.) The problems are an essential part of this book and often provide a discovery-like experience regarding the associated topic. The code used for all examples is provided in the software package that accompanies this book, available at http://www.crcpress.com/product/isbn/9781466567368. Since many of the problems are extensions or modifications of examples given in the book, time can be saved by starting with the code of a related example. The package also includes support routines and data files used in the examples and problems, and the code used to generate many of the figures.

For instructors, there is an Education Support Package available that contains the problem solutions and PowerPoint® presentations for each of the chapters. These PowerPoint slides include figures, equations, and text and can be modified by the instructor as desired.

In addition to heavy reliance on MATLAB problems and examples, this book makes extensive use of simulated data. Examples involving biological signals are only occasionally used. In our view, examples using biological signals provide motivation, but they are not generally very instructive. Given the wide range of material to be presented at a working depth, emphasis is placed on learning the tools of signal processing; motivation is left to the reader (or the instructor).

Organization of the book is straightforward. Chapters 1 through 4 are fairly basic. Chapter 1 covers topics related to analog signal processing and data acquisition, while Chapter 2 includes topics that are basic to all aspects of signal and image processing. Chapters 3 and 4 cover classical spectral analysis and basic digital filtering, topics fundamental to any signal-processing course. Advanced spectral methods are covered in Chapter 5 and are important due to their widespread use in biomedical engineering. Chapter 6 and the first part of Chapter 7 cover topics related to spectral analysis when the signal's spectrum is varying in time, a condition often found in biological signals. Chapter 7 addresses both continuous and discrete wavelets, another popular technique used in the analysis of biomedical signals. Chapters 8 and 9 feature advanced topics. In Chapter 8, optimal and adaptive filters are covered: the latter's inclusion is also motivated by the time-varying nature of many biological signals. Chapter 9 introduces multivariate techniques, specifically Principal Component Analysis and Independent Component Analysis, two analysis approaches that are experiencing rapid growth with regard to biomedical applications. Chapters 10 and 11 cover nonlinear signal-processing methods. The next three chapters are about image processing with the first of these, Chapter 12, covering the conventions used by MATLAB's Imaging Processing Toolbox. Image processing is a vast area and the material included here is limited primarily to areas associated with medical imaging: image acquisition (Chapter 15); image filtering, enhancement, and transformation (Chapter 13); and segmentation and registration (Chapter 14). The last two chapters concern classification: linear discriminators and support vector machines in Chapter 16, and adaptive neural nets in Chapter 17.

Some chapters cover topics that may require an entire book for thorough presentation. Such chapters involve a serious compromise and omissions are extensive. Our excuse is that classroom experience with this approach seems to work: students end up with a working knowledge of a vast array of signal- and image-processing tools. A few of the classic or major books on these topics are cited in a bibliography at the end of the book. No effort has been made to construct an extensive bibliography or reference list since more current lists are readily available off the Web.

Textbook Protocols

Some early examples of MATLAB code are presented in full, but in most examples some of the routine code (such as for plotting, display, and labeling operation) is omitted. Nevertheless, we recommend that students carefully label (and scale when appropriate) all graphs done in the

Table I Textbook Conventions

Symbol	Description/General Usage
$x(t)$, $y(t)$	General functions of time, usually a waveform or signal
k, n	Data indices, particularly for digitized time data
N	Number of points in a signal (maximum index or size of a data set)
L	Length of a data set (alternate) such as a set of filter coefficients
$x[n]$, $y[n]$	Waveform variable, usually digitized time variables (i.e., a discrete variable)
m	Index of variable produced by transformation, or the index of specifying the member number of a family of functions (i.e., $f_m(t)$)
$X(f)$, $Y(f)$	Frequency representation (usually complex) of a time function
$X[m]$, $Y[m]$, $X(f)$	Frequency representation (usually complex) of a discrete variable
$h(t)$	Impulse response of a linear system
$h[n]$	Discrete impulse response of a linear system
$b[n]$	Digital filter coefficients representing the numerator of the discrete transfer function, hence, the same as the impulse response
$a[n]$	Digital filter coefficients representing the denominator of the discrete transfer function
$b[m,n]$	Coefficients of a two-dimensional, image filter
`Courier font`	MATLAB command, variable, routine, or program
`'Courier font'`	MATLAB filename or string variable

problems. Some effort has been made to use consistent notation as described in Table I. In general, lower-case letters n and k are used as data indices, and capital letters (i.e., N or L) are used to indicate the signal length (or maximum subscript value) or other constants. In two-dimensional data sets, lower-case letters m and n are used to indicate the row and column subscripts of an array, while capital letters M and N are used to indicate vertical and horizontal dimensions, respectively. The letter m is also used as the index of a variable produced by a transformation, or as an index indicating a particular member of a family of related functions.[*] As is common, analog or continuous time variables are denoted by a "t" within parentheses (i.e., $x(t)$), whereas brackets enclose subscripts of digitized signals composed of discrete variables (i.e., $x[n]$). Other notation follows either standard or MATLAB conventions.

Italics is used to introduce important new terms that should be incorporated into the reader's vocabulary. If the meaning of these terms is not obvious from their use, they are explained where they are introduced. All MATLAB commands, routines, variables, and code are shown in `courier typeface`. Single quotes are used for string variables following MATLAB conventions. These and other protocols are summarized in Table I.

[*] For example, m would be used to indicate the harmonic number of a family of harmonically related sine functions; that is, $f_m(t) = \sin(2\pi mt)$.

Acknowledgments

John L. Semmlow wishes to thank Susanne Oldham who, from the quaint little village of Yellow Springs, Ohio, managed to edit this book, and provided strong, continuing encouragement and support. He would also like to acknowledge the patience and support of Peggy Christ and Lynn Hutchings. Benjamin Griffel would like to particularly thank Professor Tony Shinbrot for discussions on chaos and nonlinear dynamics that contributed greatly to Chapters 10 and 11. Both authors are deeply indebted to Raymond Muzic of Case Western University for his thoughtful comments, corrections, and insights regarding many of the topics in this book. Finally, we both would like to thank our students who provided suggestions and whose enthusiasm for the material provided much-needed motivation.

Authors

John L. Semmlow was born in Chicago, in 1942, and received BSEE from the University of Illinois in Champaign in 1964. Following several years as a design engineer for Motorola, Inc., he entered the Bioengineering Program at the University of Illinois Medical Center in Chicago, and received his PhD in 1970. He has held faculty positions at the University of California, Berkeley, the University of Illinois, Chicago, and a joint Professorship in Surgery, Robert Wood Johnson Medical School and Biomedical Engineering, School of Engineering, Rutgers University in New Jersey. In 1985 he was a NSF/CNRS (National Science Foundation/Centre National de la Recherche Scientifique) Fellow in the Sensorimotor Control Laboratory of the University of Provence, Marseille, France. He was appointed a Fellow of the IEEE (Institute of Electrical and Electronics Engineers) in 1994 in recognition of his work on acoustic detection of coronary artery disease. He was elected Fellow of AIMBE (American Institute for Medical and Biological Engineering) in 2003 and Fellow of BMES (Biomedical Engineering Society) in 2005. He was founding chair of the International Conference on Vision and Movement in Man and Machines, first held in Berkeley in 1995. His primary research interests include discovering strategies for how the brain controls human motor behavior such as eye movements and the development of medical instrumentation for noninvasive detection of coronary artery disease.

He is an avid, if somewhat dangerous, skier, and an enthusiastic, but mediocre, tennis player. He teaches folk dance and is president of the Board of the New Brunswick Chamber Orchestra. He especially enjoys travel and the slow, but nonetheless expensive, sport of sailing. He recently purchased a sailboat (currently in the British Virgin Islands) and he soon hopes to launch his second career in computer-controlled kinetic art.

Benjamin Griffel was born in Brooklyn, New York. He received bachelor's and PhD degrees from Rutgers University in biomedical engineering in 2005 and 2011, respectively, and MS from Drexel University in 2007. His thesis topic explored the use of chaotic and entropy-based signal analysis to diagnose coronary artery disease using acoustic cardiovascular recordings. He is currently an INSPIRE (IRACDA New Jersey/New York for Science Partnerships in Research and Education) fellow at Rutgers University and a teaching partner at New Jersey City University. INSPIRE is an NIH (National Institutes of Health) fellowship designed to help young researchers develop teaching skills and to increase diversity within the areas of STEM (science, technology, engineering, and mathematics). His current research examines the use of nonlinear and chaotic signal analysis to analyze the effects of inflammation on heart rate recordings. In his spare time, Benjamin enjoys guitar playing, arm chair philosophizing, and spending time with his wife.

1 Introduction

1.1 Biosignals

Much of the activity in biomedical engineering, be it clinical or research, involves the measurement, processing, analysis, display, and/or generation of biosignals. Signals are variations in energy that carry information, and the search for information from living systems has been a preoccupation of medicine since its beginnings (Figure 1.1). This chapter is about the modification of such signals to make them more useful or more informative. To this end, this chapter presents tools and strategies that use digital signal processing to enhance the information content or interpretation of biosignals.

The variable that carries information (the specific energy fluctuation) depends on the type of energy involved. Biological signals are usually encoded into variations of electrical, chemical, or mechanical energy, although, occasionally, variations in thermal energy are of interest. For communication within the body, signals are primarily encoded as variations in electrical or chemical energy. When chemical energy is used, encoding is usually done by varying the concentration of the chemical within a "physiological compartment," for example, the concentration of a hormone in blood. Bioelectric signals use the flow or concentration of ions, the primary charge carriers within the body, to transmit information. Speech, the primary form of communication between humans, encodes information as variations in pressure. Table 1.1 summarizes the different types of energy that can be used to carry information and the associated variables that encode this information. Table 1.1 also shows the physiological measurements that involve these energy forms.

Outside the body, information is commonly transmitted and processed as variations in electrical energy, although mechanical energy was used in the seventeenth and early eighteenth centuries to send messages. The semaphore telegraph used the position of one or more large arms placed on a tower or high point to encode letters of the alphabet. These arm positions could be observed at some distance (on a clear day) and relayed onward if necessary. Information processing can also be accomplished mechanically, as in the early numerical processors constructed by Babbage. Even mechanically based digital components have been attempted using variations in fluid flow. Modern electronics provides numerous techniques for modifying electrical signals at very high speeds. The body also uses electrical energy to carry information when speed is important. Since the body does not have many free electrons, it relies on ions, notably Na^+, K^+, and Cl^-, as the primary charge carriers. Outside the body, electrically based signals are so useful that signals carried by other energy forms are usually converted into electrical energy when significant transmission or processing tasks are required. The conversion of physiological

Figure 1.1 Information about internal states of the body can be obtained through the acquisition and interpretation of biosignals. Expanding such information is an ongoing endeavor of medicine and medical research. It is also the primary motivation of this chapter.

Table 1.1 Energy Forms and Associated Information-Carrying Variables

Energy	Variables (Specific Fluctuation)	Common Measurements
Chemical	Chemical activity and/or concentration	Blood ion, O_2, CO_2, pH, hormonal concentrations, and other chemistry
Mechanical	Position Force, torque, or pressure	Muscle movement, cardiovascular pressures, muscle contractility, valve, and other cardiac sounds
Electrical	Voltage (potential energy of charge carriers) Current (charge carrier flow)	EEG, ECG, EMG, EOG, ERG, EGG, and GSR
Thermal	Temperature	Body temperature and thermography

energy into an electric signal is an important step, often the first step, in gathering information for clinical or research use. The energy conversion task is done by a device termed a *transducer*,* specifically, a *biotransducer*. The biotransducer is usually the most critical component in systems designed to measure biosignals.

With the exception of this chapter, this book is limited to topics on digital signal and image processing. To the extent possible, each topic is introduced with the minimum amount of information required to use and understand the approach along with enough information to apply the methodology in an intelligent manner. Strengths and weaknesses of the various methods are also explored, particularly through discovery in the problems at the end of the chapter. Hence, the problems at the end of each chapter, most of which utilize the MATLAB® software package (Waltham, MA), constitute an integral part of the book and a few topics are introduced only in the problems.

A fundamental assumption of this book is that an in-depth mathematical treatment of signal-processing methodology is not essential for effective and appropriate application of these tools. This book is designed to develop skills in the application of signal- and image-processing technology, but it may not provide the skills necessary to develop new techniques and algorithms. References are provided for those who need to move beyond the application of signal- and image-processing tools to the design and development of new methodology. In the subsequent chapters, the major topics include sections on implementation using the MATLAB software package. Fluency with the MATLAB language is assumed and is essential for the use of this book. Where appropriate, a topic area may also include a more in-depth treatment, including some of the underlying mathematics.

1.2 Biosignal Measurement Systems

Biomedical measurement systems are designed to measure and usually record one or more biosignals. A schematic representation of a typical biomedical measurement system is shown in Figure 1.2. The term biomedical measurement is quite general and includes image acquisition

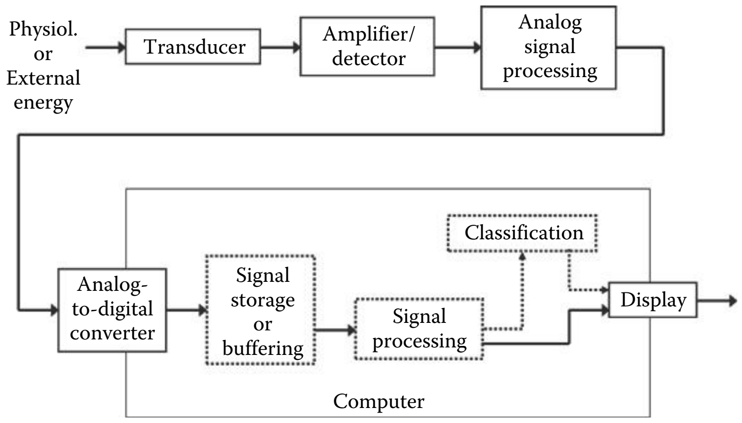

Figure 1.2 Schematic representation of a typical biomedical measurement system.

* Learning the vocabulary is an important part of mastering a discipline. In this book, the commonly used terms are highlighted using italics. Sometimes, the highlighted term is described when it is introduced, but occasionally, the determination of its definition is the responsibility of the reader.

and the acquisition of different types of diagnostic information. Information from the biological process of interest must first be converted into an electric signal via the *transducer.* Some analog signal processing is usually required, often including amplification and lowpass (or bandpass) filtering. Since most signal processing is easier to implement using digital methods, the analog signal is converted into a digital format using an analog-to-digital converter (ADC). Once converted, the signal is often stored or *buffered* in the memory to facilitate subsequent signal processing. Alternatively, in *real-time* applications, the incoming data are processed as they come in, often with minimal buffering, and may not be permanently stored. Digital signal-processing algorithms can then be applied to the digitized signal. These signal-processing techniques can take on a wide variety of forms with varying levels of sophistication and they make up the major topic areas of this book. In some applications such as diagnosis, a classification algorithm may be applied to the processed data to determine the state of a disease or the class of a tumor or tissue. A wide variety of classification techniques exist and the most popular techniques are discussed in Chapters 16 and 17. Finally, some sort of output is necessary in any useful system. This usually takes the form of a display, as in imaging systems, but it may be some type of effector mechanism such as in an automated drug delivery system. The basic elements shown in Figure 1.2 are discussed in greater detail in the next section.

1.3 Transducers

A transducer is a device that converts energy from one form to another. By this definition, a lightbulb or a motor is a transducer. In signal-processing applications, energy is used to carry information; the purpose of energy conversion is to transfer that information, not to transform energy as with a lightbulb or a motor. In measurement systems, all transducers are so-called input transducers: they convert nonelectrical energy into an electronic signal. An exception to this is the electrode, a transducer that converts ionic electrical energy into electronic electrical energy. Usually, the output of a biomedical transducer is voltage (or current) whose amplitude is proportional to the measured energy (Figure 1.3).

Input transducers use one of the two different fundamental approaches: the input energy causes the transducer element to generate voltage or current, or the input energy creates a change in an electrical property (the resistance, inductance, or capacitance) of the transducer material. Most optical transducers use the first approach. Photons strike a photo sensitive material producing free electrons (or holes) that can then be detected as external current flow. Piezoelectric devices used in ultrasound also generate a charge when under mechanical stress. Many examples can be found in the use of the second category, a change in some electrical property. For example, metals (and semiconductors) undergo a consistent change in resistance with changes in temperature and most temperature transducers utilize this feature. Other examples include the strain gage, which measures mechanical deformation using the small change in resistance that occurs when the sensing material is stretched.

The energy that is converted by the input transducer may be generated by the physiological process itself, it may be energy that is indirectly related to the physiological process, or it may be energy produced by an external source. In the last case, the externally generated energy interacts with, and is modified by, the physiological process and it is this alteration that produces

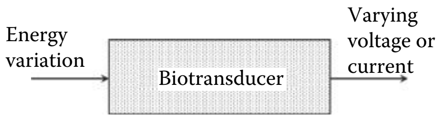

Figure 1.3 General systems representation of a transducer.

Table 1.2 Energy Forms and Related Direct Measurements

Energy	Measurement
Mechanical Length, position, and velocity Force and pressure	Muscle movement, cardiovascular pressures, muscle contractility valve, and other cardiac sounds
Heat	Body temperature and thermography
Electrical	EEG, ECG, EMG, EOG, ERG, EGG, and GSR
Chemical	Ion concentrations

the measurement. For example, when externally generated x-rays are transmitted through the body, they are absorbed by the intervening tissue and a measurement of this absorption is used to construct an image. Many diagnostically useful imaging systems are based on this external energy approach.

In addition to images created by external energy that is passed through the body, some images are generated using the energy of radioactive emissions of radioisotopes injected into the body. These techniques make use of the fact that selected or "tagged," molecules will collect in the specific tissue. The areas where these radioisotopes collect can be mapped using a γ-camera or with certain short-lived isotopes, which are localized better using positron emission tomography (PET).

Many physiological processes produce energy that can be detected directly. For example, cardiac internal pressures are usually measured using a pressure transducer placed on the tip of a catheter introduced into the appropriate chamber of the heart. The measurement of electrical activity in the heart, muscles, or brain provides other examples of the direct measurement of physiological energy. For these measurements, the energy is already electrical and only needs to be converted from ionic into electronic current using an "electrode." These sources are usually given the term ExG, where "x" represents the physiological process that produces electrical energy, ECG—electrocardiogram, EEG—electroencephalogram, EMG—electromyogram, EOG—electrooculogram, ERG—electroretinogram, and EGG—electrogastrogram. An exception to this terminology is the galvanic skin response (GSR), the electrical activity generated by the skin. The typical physiological measurements that involve the conversion of other energy forms into electrical energy are shown again in Table 1.2. Figure 1.4 shows the early ECG machine where the interface between the body and the electrical monitoring equipment was buckets filled with saline ("E" in Figure 1.4).

The biotransducer is often the most critical element in the system since it is the interface between the subject or life process and the rest of the system. The transducer establishes the risk or "noninvasiveness," of the overall system. For example, an imaging system based on differential absorption of x-rays, such as a computed tomography (CT) scanner, is considered more *invasive* than an imaging system based on ultrasonic reflection since CT uses ionizing radiation that may have an associated risk. (The actual risk of ionizing radiation is still an open question and imaging systems based on x-ray absorption are considered *minimally invasive*.) Both ultrasound and x-ray imaging would be considered less invasive than, for example, monitoring internal cardiac pressures through cardiac catheterization, where a small catheter is threaded into the heart chambers.

Many critical problems in medical diagnosis await the development of new approaches and new transducers. Indeed, many of the outstanding problems in biomedical measurement, such as noninvasive measurement of internal cardiac pressures or the noninvasive measurement of intracranial pressure, await an appropriate, and undoubtedly clever, transducer mechanism.

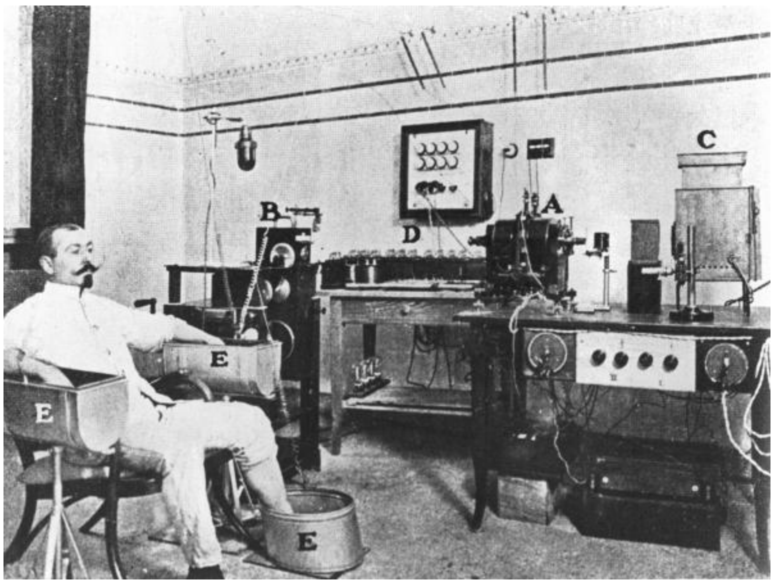

Figure 1.4 An early ECG machine using buckets filled with a saline solution as electrodes (marked "E" in this figure).

1.4 Amplifier/Detector

The design of the first analog stage in a biomedical measurement system depends on the basic transducer operation. If the transducer is based on a variation in electrical property, the first stage must convert that variation into a variation in voltage. If there is only one sensing element in the transducer, the transducer signal is said to be *single ended* and a constant current source can be used as a detector (Figure 1.5). A typical transducer that would use this type of detector circuit is a temperature-measurement device based on a thermistor, a resistor-like element that changes its resistance with temperature. Thermistors have a well-defined relationship between temperature and resistance. The detector equation for the circuit in Figure 1.5 follows Ohm's law:

$$V_{\text{out}} = I(R + \Delta R) \quad \text{where } \Delta R = f(\text{input energy}) \tag{1.1}$$

If the transducer can be configured differentially so that one element increases with increasing input energy while the other element decreases, the bridge circuit is commonly used as a detector. Figure 1.6 shows a device made to measure intestinal motility using strain gages.

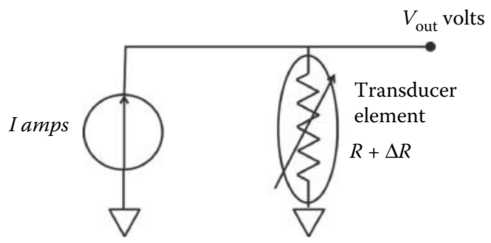

Figure 1.5 A constant current source is used in the detector circuit for a transducer based on an energy-induced variation in resistance. One example is a temperature-measuring transducer based on a thermistor, a semiconductor that changes resistance with temperature in a well-defined manner.

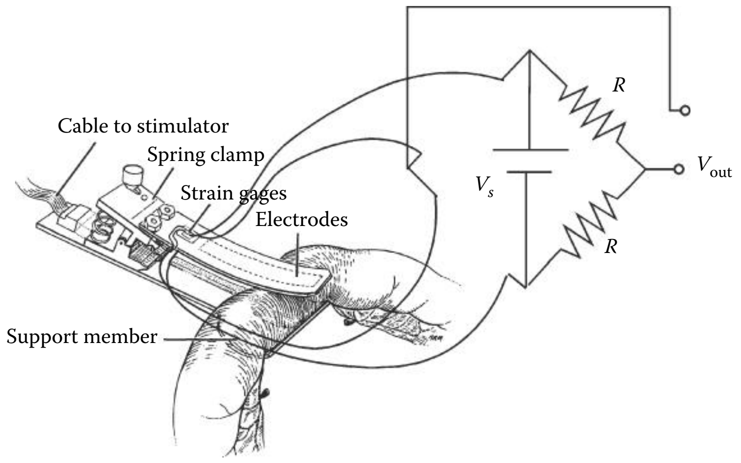

Figure 1.6 A strain gage probe used to measure motility of the intestine. The bridge circuit is used to convert the differential chance in resistance from a pair of strain gages into a change in voltage.

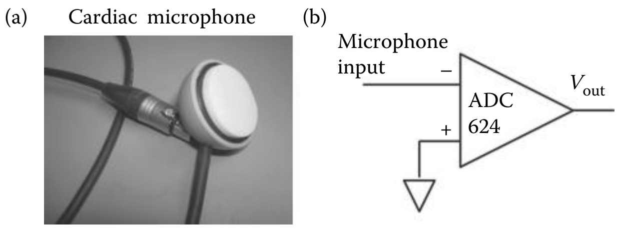

Figure 1.7 (a) A cardiac microphone that uses a piezoelectric element to generate a small voltage when deformed by sound pressure. (b) The detector circuit for this transducer is just a low-noise amplifier.

A bridge circuit detector is used in conjunction with a pair of differentially configured strain gages: when the intestine contracts, the end of the cantilever beam moves downward and the upper strain gage (visible) is stretched and increases in resistance whereas the lower strain gage (not visible) compresses and decreases in resistance. The output of the bridge circuit can be found from simple circuit analysis to be $V_{out} = V_S R/2$, where V_S is the value of the source voltage. If the transducer operates based on a change in inductance or capacitance, the above techniques are still useful except a sinusoidal voltage source must be used.

If the transducer element is a voltage generator, the first stage is usually an amplifier. Figure 1.7a shows a cardiac microphone used to monitor the sounds of the heart that is based on a piezoelectric element. The piezoelectric element generates a small voltage when sound pressure deforms this element; so, the detector stage is just a low-noise amplifier (Figure 1.7b). If the transducer produces a current output, as is the case in many electromagnetic detectors, then a current-to-voltage amplifier (also termed a transconductance amplifier) is used to produce a voltage output. In some circumstances, additional amplification beyond the first stage may be required.

1.5 Analog Signal Processing and Filters

While the most extensive signal processing is performed on digitized data using algorithms implemented in software, some analog signal processing is usually necessary. The most common analog signal processing restricts the frequency range or *bandwidth* of the signal using *analog filters*. It is this filtering that usually sets the bandwidth of the overall measurement system. Since signal bandwidth has a direct impact on the process of converting an analog signal into an equivalent (or nearly equivalent) digital signal, it is often an essential element in any biomedical measurement system. Filters are defined by several properties: filter type, bandwidth, and attenuation characteristics. The latter can be divided into initial and final characteristics. Each of these properties is described and discussed in the next section.

1.5.1 Filter Types

Analog filters are electronic devices that remove selected frequencies. Filters are usually termed according to the range of frequencies they *do not* suppress. Thus, *lowpass* filters allow low frequencies to pass with minimum attenuation while higher frequencies are attenuated. Conversely, *highpass* filters pass high frequencies, but attenuate low frequencies. *Bandpass* filters reject frequencies above and below a *passband* region. An exception to this terminology is *bandstop* filters that pass frequencies on either side of a range of attenuated frequencies.

These filter types can be illustrated by a plot of the filter's *spectrum*, that is, a plot showing how the filter treats the signal at each frequency over the frequency range of interest. Figure 1.8 shows stereotypical frequency spectra or frequency plots of the four different filter types described above. The filter gain is the ratio of output signal amplitude to input signal amplitude as a function of frequency:

$$\text{Gain}(f) = \frac{\text{Output values}(f)}{\text{Input values}(f)} \tag{1.2}$$

The lowpass filter has a filter gain of 1.0 for the lower frequencies (Figure 1.8). This means that the output equals the input at those frequencies. However, as frequency increases, the gain drops, indicating that the output signal also drops for a given input signal. The highpass filter has exactly the opposite behavior with respect to frequency (Figure 1.8). As frequency increases, the gain and output signal increase so that at higher frequency, the gain is 1.0 and the output

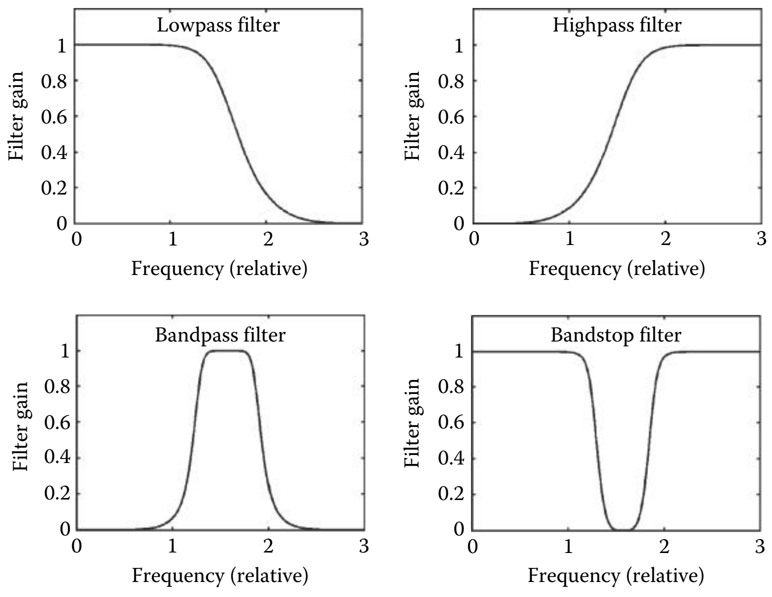

Figure 1.8 Influence on signal frequency of the four basic filter types.

equals the input. The bandpass filter is a combination of these two filters; so, the gain and output increase with frequency up to a certain frequency range where the gain is constant at 1.0, then gain decreases with further increases of frequency (Figure 1.8). The spectrum of the band-stop filter is the inverse of the bandpass filter (Figure 1.8).

Within each class, filters are also defined by the frequency ranges that they pass, termed the filter bandwidth, and the sharpness with which they increase (or decrease) attenuation as frequency varies. (Again, band-stop filters are an exception as their bandwidth is defined by the frequencies they reject.) Spectral sharpness is specified in two ways: as an initial sharpness in the region where attenuation first begins, and as a slope further along the attenuation curve. These various filter properties are best described graphically in the form of a frequency plot (sometimes referred to as a *Bode* plot), a plot of filter gain against frequency. Filter gain is simply the ratio of the output voltage divided by the input voltage, V_{out}/V_{in}, often taken in dB. (The dB operation is defined in Section 2.1.4, but is simply a scaled log operation.) Technically, this ratio should be defined for all frequencies for which it is nonzero, but practically, it is usually stated only for the frequency range of interest. To simplify the shape of the resultant curves, frequency plots sometimes plot gain in dB against the log of frequency.* When the output/input ratio is given analytically as a function of frequency, it is termed the *transfer function*. Hence, the frequency plot of a filter's output/input relationship can be viewed as a graphical representation of its transfer function (Figure 1.8).

1.5.2 Filter Bandwidth

The bandwidth of a filter is defined by the range of frequencies that are not attenuated. These unattenuated frequencies are also referred to as *passband* frequencies. Figure 1.9a shows the frequency plot of an ideal filter, a filter that has a perfectly flat passband region and an infinite attenuation slope. Real filters may indeed be quite flat in the passband region, but will attenuate with a gentler slope, as shown in Figure 1.9b. In the case of an ideal filter (Figure 1.9a), the bandwidth (the region of unattenuated frequencies) is easy to determine: specifically, the bandwidth ranges between 0.0 and the sharp attenuation at f_c Hz. When the attenuation begins gradually, as in Figure 1.9b, defining the passband region is problematic. To specify the bandwidth in this filter, we must identify a frequency that defines the boundary between the attenuated and unattenuated portions of the frequency curve. This boundary has been somewhat arbitrarily defined as the frequency when the attenuation is 3 dB.† In Figure 1.9b, the filter would have a bandwidth of 0.0 to f_c Hz, or simply f_c Hz. The filter whose frequency characteristics are shown in Figure 1.9c has a sharper attenuation characteristic, but still has the same bandwidth (f_c Hz). The bandpass filter whose frequency characteristics are shown in Figure 1.9d has a bandwidth of $f_h - f_l$ in Hz.

1.5.3 Filter Order

The slope of a filter's attenuation curve is related to the complexity of the filter: more complex filters have a steeper slope, approaching the ideal filter as shown in Figure 1.9a. In analog filters, complexity is proportional to the number of energy storage elements in the circuit. These could be either inductors or capacitors, but are generally capacitors for practical reasons. Using standard circuit analysis, it can be shown that each independent energy storage device leads to an additional order of a polynomial in the denominator of the transfer function equation (Equation 1.2) that describes the filter. (The denominator of the transfer function is also

* When gain is plotted in dB, it is in logarithmic form, since the dB operation involves taking the log (see Section 2.1.4). Plotting gain in dB against log frequency puts the two variables in similar metrics and results in more straight-line plots.

† This defining point is not entirely arbitrary because when the signal is attenuated at 3 dB, its amplitude is 0.707 ($10^{-3/20}$) of what it was in the passband region and it has half the power of the unattenuated signal since $0.707^2 = 1/2$. Accordingly, this point is also known as the *half-power point*.

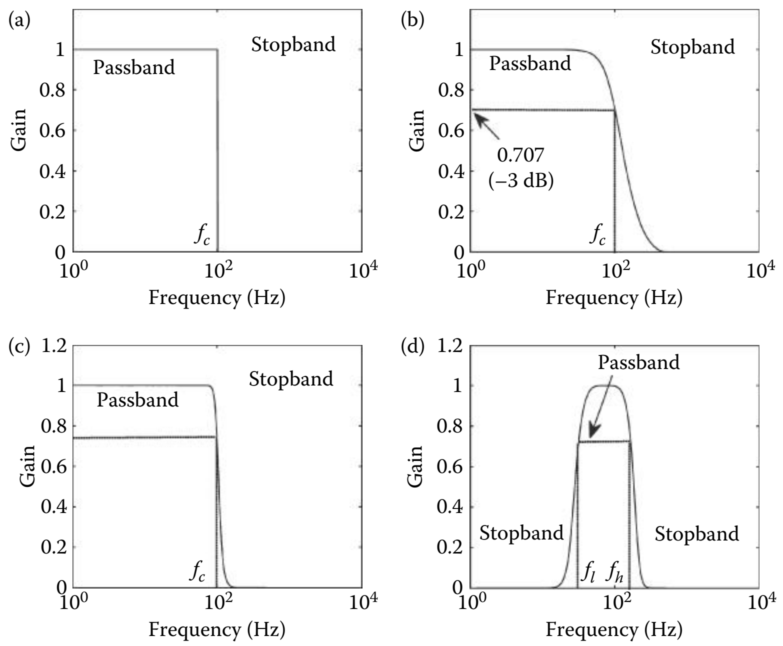

Figure 1.9 Frequency plots of ideal and realistic filters. Each of the frequency plots shown here has a linear vertical axis, but often, the vertical axis is plotted in dB. The horizontal axis is in log frequency. (a) Ideal lowpass filter. (b) Realistic lowpass filter with a gentle attenuation characteristic. (c) Realistic lowpass filter with a sharp attenuation characteristic. (d) Bandpass filter.

referred to as the *characteristic equation* because it defines the basic characteristics of the related system.) As with any polynomial equation, the number of roots of this equation will depend on the order of the equation; hence, filter complexity (i.e., the number of energy storage devices) is equivalent to the number of roots in the denominator of the transfer function. In electrical engineering, it has long been common to call the roots of the denominator equation *poles*. Thus, the complexity of a filter is also equivalent to the number of poles in the transfer function. For example, a *second-order* or *two-pole* filter has a transfer function with a second-order polynomial in the denominator and would contain two independent energy storage elements (very likely two capacitors).

Applying an asymptote analysis to the transfer function, it can be shown that for frequencies much greater than the cutoff frequency, f_c, the slope of most real-world filters is linear if it is plotted on a log-versus-log plot. Figure 1.10 shows the frequency characteristics of the transfer function of a first-order (single-pole) filter with a cutoff frequency, f_c, of 5 Hz plotted in both linear (Figure 1.10a) and dB versus log frequency (Figure 1.10b) format. Converting the vertical axis to dB involves taking the log (see Section 2.1.4); so Figure 1.10b is a log–log plot. At the cutoff frequency of 5 Hz, the frequency characteristic is curved, but at higher frequencies, above 10 Hz, the curve straightens out to become a downward slope that decreases 20 dB for each order of magnitude, or decade, increase in frequency. For example, at 50 Hz, the frequency characteristic has a value of −20 dB and at 500 Hz the value would be −40 dB although this is not shown in the graph. Plotting dB versus log frequency leads to the unusual units for the slope of dB/decade. Nonetheless, this type of plot is often used because it generates straight-line segments for the frequency characteristics of real-world filters and because taking logs extends the range of values presented on both axes. Both linear and dB versus

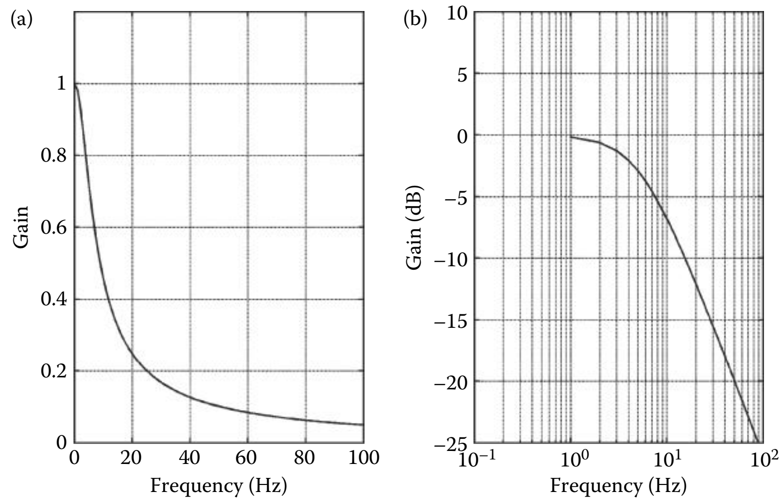

Figure 1.10 Two representations of the gain characteristics (i.e., transfer function) for a first-order filter. (a) A linear plot of gain against frequency. (b) The same curve is plotted with gain in dB, a log function, against log frequency. The attenuation slope above the cutoff frequency becomes a straight line with a slope of 20 dB/decade.

log frequency plotting is used in this book; the axes will describe which type of plot is being presented.

The downward slope of 20 dB/decade seen for the first-order filter shown in Figure 1.10b generalizes, in that for each additional filter pole added (i.e., each increase in filter order), the slope is increased by 20 dB/decade. (In a lowpass filter, the downward slope is sometimes referred to as the filter's *roll-off*.) Figure 1.11 shows the frequency plot of a second-order, two-pole filter

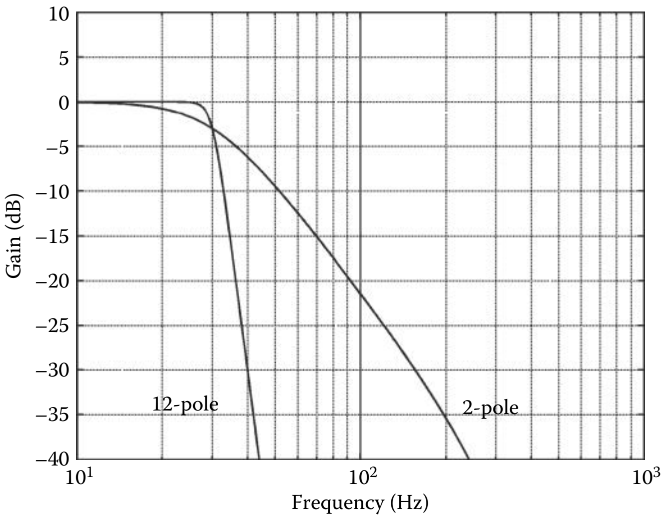

Figure 1.11 A dB versus log frequency plot of a second-order (two-pole) and a 12th-order (12-pole) lowpass filter having the same cutoff frequency. The higher-order filter more closely approaches the sharpness of an ideal filter.

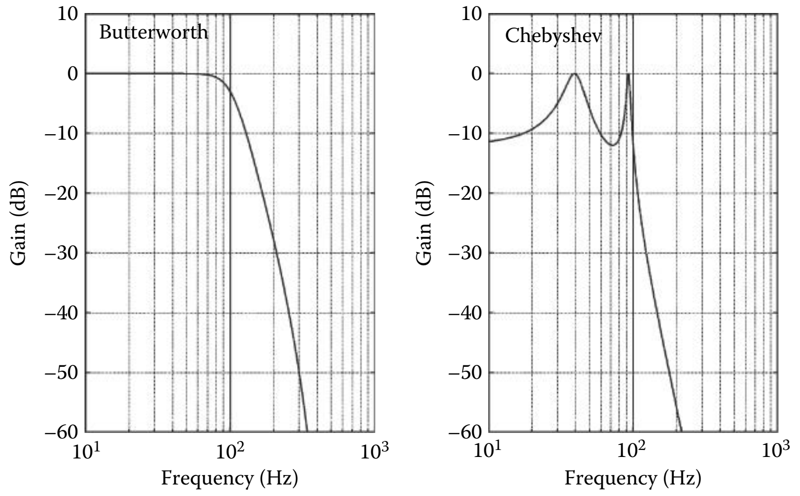

Figure 1.12 Two filters that have the same cutoff frequency (100 Hz) and the same order (four-pole), but differing in the sharpness of the initial slope. The filter labeled Chebyshev has a steeper initial slope, but contains ripples in the passband region.

with a slope of 40 dB/decade and a 12th-order lowpass filter. Both filters have the same cutoff frequency, f_c, hence the same bandwidth. The steeper slope or roll-off of the 12-pole filter is apparent. In principle, a 12-pole lowpass filter would have a slope of 240 dB/decade (12×20 dB/decade). In fact, this frequency characteristic is theoretical because in real analog filters, parasitic components and inaccuracies in the circuit elements limit the actual attenuation that can be obtained. The same rationale applies to highpass filters, except that the frequency plot decreases with decreasing frequency at a rate of 20 dB/decade for each highpass filter pole.

1.5.4 Filter Initial Sharpness

As shown above, both the slope and the initial sharpness increase with filter order (number of poles), but increasing filter order also increases the complexity and hence the cost of the filter. It is possible to increase the initial sharpness of the filter's attenuation characteristics without increasing the order of the filter, if you are willing to accept some unevenness or *ripple* in the passband. Figure 1.12 shows two lowpass, fourth-order filters having the same cutoff frequency, but differing in the initial sharpness of the attenuation. The one-marked Butterworth has a smooth passband, but the initial attenuation is not as sharp as the one marked Chebyshev, which has a passband that contains ripples. This property of analog filters is also seen in digital filters and is discussed in detail in Chapter 4.

EXAMPLE 1.1

An ECG signal of 1 V peak to peak has a bandwidth from 0.01 to 100 Hz. (Note that this frequency range has been established by an official standard and is meant to be conservative.) It is desired to reduce any noise in the signal by at least 80 dB for frequencies above 1000 Hz. What is the order of analog filter required to achieve this goal?

Solution

Since the signal bandwidth must be at least 100 Hz, the filter's cutoff frequency, f_c, must be not less than 100 Hz, but the filter must reduce the signal by 80 dB within 1 decade. Since typical analog filters reduce the gain by 20 dB/decade for each pole, and an 80-dB reduction is desired, the filter must have at least 80/20 = 4 poles.

Similar, but more involved, problems of this sort are explored in Chapter 4 that covers digital filters.

1.6 ADC Conversion

The last analog element in the typical measurement system shown in Figure 1.2 is the interface between the analog and digital world: the ADC. As the name implies, this electronic component converts an analog voltage into an equivalent digital number. In the process of ADC conversion, an analog or a continuous waveform, $x(t)$, is converted into a discrete waveform, $x[n]$,* a function of real numbers that are defined as integers at discrete points in time. These numbers are called *samples* and the discrete points in time are usually taken at regular intervals termed the sample interval, T_s. The sample interval can also be defined by a frequency termed the *sample frequency*:

$$f_s = \frac{1}{T_s}\text{Hz} \tag{1.3}$$

To convert a continuous waveform into a digital format requires slicing the signal in two ways: slicing in time and in slicing amplitude (Figure 1.13). So, the continuous signal $x(t)$ becomes just a series of numbers, $x[1]$, $x[2]$, $x[3]$,…,$x[n]$ that are the signal values at times $1T_s$, $2T_s$, $3T_s$, and nT_s. In addition, if the waveform is longer than the computer memory, only a portion of the analog waveform can be converted into digital format. Segmenting a waveform to fit in computer memory is an operation termed *windowing*. The consequences of this operation are discussed in Chapter 3. Note that if a waveform is sampled at T_s seconds for a total time, T_T seconds, the number of values stored in the computer will be

$$N = \frac{T_T}{T_s}\text{points} \tag{1.4}$$

The relationship between a sample stored in the computer and the time at which it was sampled is determined by its sample index, n, and the sample interval or sampling frequency:

$$t = nT_s = \frac{n}{f_s} \tag{1.5}$$

Each of these modifications, time slicing and amplitude slicing, has an impact on the resulting digital signal. Comparing the analog and digital signal in Figure 1.13 (left upper and lower graphs), it is evident that the digitized signal is not the same as the original. The question is: Does the difference matter, and if so, how does it matter? Intuitively, we might expect that if the time and amplitude slices were very small, then "for all practical purposes" (that great engineering expression), the digital and analog signals are essentially the same. The question then becomes: how small must we make the time and amplitude slices? This question is addressed separately for the two slicings in the following sections. Before approaching these questions, we present a simple MATLAB example of the generation and display of a digitized waveform.

* It is common to use brackets to represent discrete or digital functions whereas parentheses are used in continuous or analog functions. See Section 1.3.1.

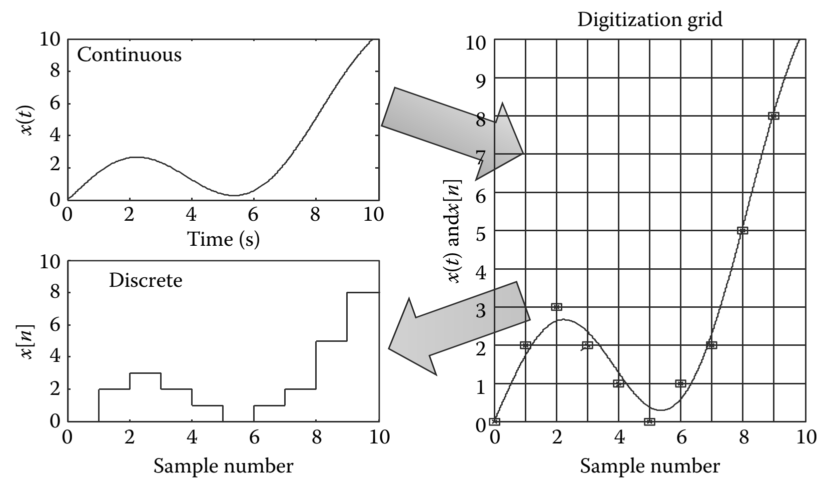

Figure 1.13 Digitizing a continuous signal, upper left, requires slicing the signal both in time and amplitude, right side. The result is a series of discrete numbers (squares) that approximate the original signal. The resultant digitized signal, lower left, consists of a series of discrete numerical values sampled at discrete intervals of time. In this example, $x[n] = 2, 3, 2, 1, 0, 2, 3, 5,$ and 8.

EXAMPLE 1.2

Generate a discrete 2-Hz sine wave using MATLAB. Assume a sample time of 0.01 s and use enough points to make the sine wave 1-s long, that is, the total time, T_T, should be 1 s. Plot the result in seconds. Label the time axis correctly (as is always expected in MATLAB problems).

Solution

In this example, we are given sample intervals and the total time; so, the number of discrete points that will be needed is fixed. From Equation 1.5, the number of points in the array will be

$$N = T_T/T_s = 100 \text{ points}$$

In many MATLAB problems, the number of points is arbitrary and we need to pick some reasonable value for N. The best way to solve this problem in MATLAB is first to generate a time vector* that is 1-s long with increments of T_s seconds: `t = 0:Ts:1`. We do not even have to find the number of points explicitly, although it turns out to be 100. We can use this time vector in conjunction with the MATLAB `sin` function to generate the sine wave.

```
% Example 1.2 Generate a discrete 2 Hz sine wave using MATLAB.
%  Assume a sample time of 0.01 sec. and use enough points to make
%  the sine wave 1 sec. long; i.e., the total time, TT should be 1 sec.
%
clear all; close all;
Ts = .01;                    % Define Ts = .01 sec
TT = 1;                      % Define total time = 1 sec
f = 2;                       % Define frequency = 2 Hz
t = 0:Ts:1;                  % Generate time vector
x = sin(2*pi*f*t);           % Generate desired sine wave
plot(t,x,'.k');              % Plot sine wave as discrete points
xlabel('Time (sec)');        % and label
ylabel('x(t)');
```

* The reason a MATLAB sequence of numbers or an array is sometimes called a vector is discussed in Section 2.3.1.1.

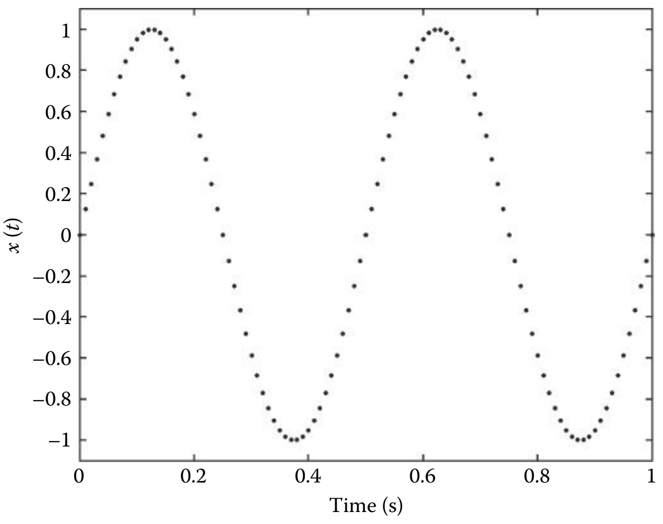

Figure 1.14 Sequence of points generated by the code in Example 1.2. The sine wave pattern of these points is evident. If these points were connected by straight lines, the graph would look very much like a sine wave.

Results

The program produces the sequence of points shown in Figure 1.14. The sequence of points clearly varies in a sinusoidal manner; if the points were connected by lines (as done by the `plot` unless an alternative is requested), the plot would look just like a sine wave. This is examined in Problem 1.5. Note how the MATLAB command that produces the sine wave looks the same as an equivalent mathematical statement: $x = \sin(2\pi ft)$.

1.6.1 Amplitude Slicing

Amplitude slicing, slicing the signal amplitude into discrete levels, is termed *quantization*. The equivalent numerical value of the *quantized* signal can only approximate the level of the analog signal and the degree of approximation will depend on the number of different values that are used to represent the signal. Since digital signals are represented as binary numbers, the bit length of binary numbers used determines the quantization level. The minimum voltage that can be resolved and the amplitude slice size is known as the quantization level, q. The slice size in volts is the voltage range of the converter divided by the number of available discrete values, assuming all bits are converted accurately. If the converter produces a binary number having b accurate bits, then the number of nonzero values is $(2^b - 1)$, where b is the number of bits in the binary output. If the voltage range of the converter varies between 0 and V_{MAX} volts, then the quantization step size, q, in volts, is given as

$$q = \frac{V_{\text{MAX}}}{2^b - 1}\ \text{V} \tag{1.6}$$

where V_{MAX} is the range of the ADC and b is the number of bits converted.

EXAMPLE 1.3

The specifications (*specs*) of a 12-bit ADC advertise an accuracy of ± *the least significant bit* (LSB). If the input range of the ADC is 0–10 V, what is the resolution of the ADC in analog voltage?

Solution

If the input range is 10 V, then the analog voltage represented by the LSB can be found using Equation 1.6:

$$V_{\text{LSB}} = \frac{V_{\text{MAX}}}{(2^{bits}-1)} = \frac{10}{(2^{12}-1)} = \frac{10}{4095} = 0.0024\ \text{V}$$

Hence, the resolution would be ±0.0024 V.

Typical converters feature 8-, 12-, and 16-bit outputs, although some high-end audio converters use 24 bits. In fact, most signals do not have sufficient signal-to-noise ratio to justify a higher resolution; you are simply obtaining a more accurate conversion of the noise in the signal. For example, given the quantization level of a 12-bit ADC, the dynamic range is $2^{12} - 1 = 4095$; in dB, this is 20*log (4095) = 72 dB. Since typical signals, especially those of biological origin, have dynamic ranges rarely exceeding 40–50 dB and are often closer to 30 dB, a 12-bit converter with a dynamic range of 72 dB is usually adequate. Microprocessor chips often come with built-in 12-bit ADCs. A 16-bit converter with a theoretical range of 96 dB may be used for greater dynamic range or to provide more *headroom* so that the signal can be comfortably less than the maximum range of the ADC. Alternatively, an 8-bit ADC still offers a dynamic range of 48 dB and may be sufficient, and the converted 8-bit value requires only 1 byte of storage. Note that in most ADCs, the accuracy of the conversion is ±the LSB; so, the accuracy of real ADCs is based on the number of bits converted minus 1.

The effect of quantization can be viewed as a noise added to the original signal (Figure 1.15). This noise depends on the quantization level, q, in Equation 1.6. If a sufficient number of quantization levels exist (say, >64), the quantization error can be modeled as additive-independent white noise with a uniform distribution between $\pm q/2$ and zero mean. The variance or mean square error of this noise can be determined using the expectation function from basic statistics

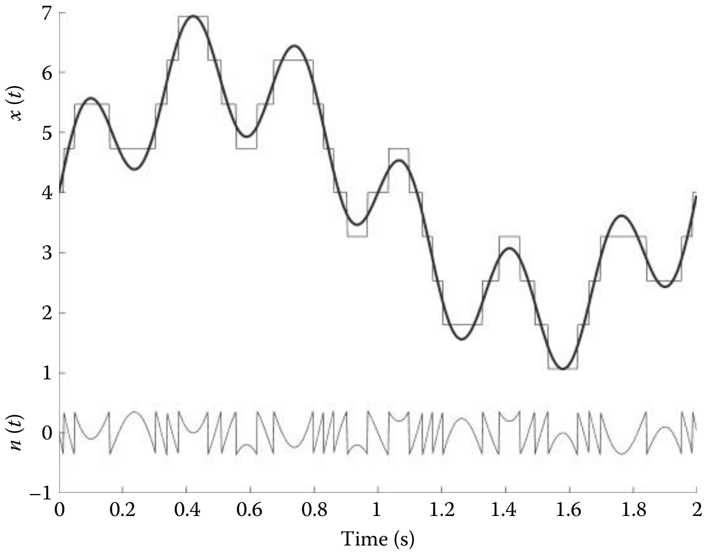

Figure 1.15 The effect of quantization on the original signal can be viewed as noise added to the signal. The amplitude of this noise depends on the quantization level, q, which in turn depends on the number of bytes employed by the ADC. The variance of this noise, which is approximately the same as the RMS value, is given in Equation 1.8.

$$\sigma^2 = \overline{e^2} = \int_{-\infty}^{\infty} e^2 \mathrm{PDF}(e)\, de \qquad (1.7)$$

where PDF(*e*) is the uniform probability density function and *e* is the error voltage (the bottom trace in Figure 1.13). The PDF(*e*) for a uniform distribution between ±*q* is simply 1/*q*. Substituting into Equation 1.7:

$$\sigma^2 = \int_{-q/2}^{q/2} e^2 (1/q)\, de = \left.\frac{(1/q)e^3}{3}\right|_{-q/2}^{q/2} = \frac{q^2}{12} \qquad (1.8)$$

EXAMPLE 1.4

Write a MATLAB program to evaluate Equation 1.8 through simulation. Generate a 4-Hz sine wave in a 1000-point array ($N = 1000$). Assume a sample interval of $T_s = 0.002$. Quantize this sine wave array into a 6-bit binary number using the routine `quantization.m`.* The calling structure for this routine is

```
signal_out = quantization(signal_in,bits)
```

where `signal_in` is the original signal, `bits` is the number of bits for quantization (bits = 6 in this case), and `signal_out` is the quantized signal. Find the noise signal by subtracting the original signal from the quantized signal. Plot this signal and take the variance to quantify the noise. Then evaluate the theoretical noise using Equations 1.6 and 1.8 to find the theoretical variance and compare the two results.

Solution

Since the number of points desired is given, it is easier to generate a 1000-point time vector and then multiply it by T_s, that is, `t = (0:999)*Ts`. Then use that vector to generate the desired 4-Hz sine wave signal as in Example 1.2. (Note that almost any waveform and sample interval will work as long as it includes a fair number of points: the sine wave is just a handy waveform.) Quantize the signal into 6 bits using the routine `quantization.m`. Subtract the original signal from the quantized signal and take the variance of the result as the simulated noise variance. Then evaluate Equation 1.6 with bits = 6 to find the quantization level *q*, and use that value of *q* in conjunction with Equation 1.8 to find the theoretical value of noise variance. Compare the two results. Use a formatted output to ensure that the results are displayed in an adequate number of decimal places.

```
% Example 1.4 Evaluate the quantization equation, Eq. 1.8 using simulated
data.
%
f = 4;                         % Desired frequency
N = 1000;                      % Number of points
Ts = 0.002;                    % Ts
bits = 6;                      % Quantization level
t = (0:N-1)*Ts;                % Vector used to generate 1-cycle sine wave
signal_in = sin(2*pi*f*t);     % Generate signal
signal_out = quantization(signal_in,bits);  % Quantize signal
noise_signal = signal_out - signal_in;      % Determine quantization error
```

* The MATLAB routine `quantization.m`, like all auxiliary routines, can be found in this book's website. For clarity, MATLAB variables, files, instructions, and routines are shown in courier typeface throughout the book.

```
q_noise = var(noise_signal);  % Variance of quantization noise
q = 1/(2^bits - 1);           % Calculate quantization level (Eq. 1.6)
theoretical = (q^2)/12;       % Theoretical quantization error (Eq. 1.8)
disp(' Quantization Noise')
disp('Bits    Emperical       Theoretical')
out = sprintf('%2d %5e %5e', bits, q_noise, theoretical); % Format output
disp(out)
```

Results

The results produced by this program are shown in Table 1.3. The noise variance determined empirically by the program is quite close to the theoretical value determined from Equations 1.6 and 1.8. This evaluation is extended to four different bit values in Problem 1.7.

It is relatively easy and common to convert between the analog and digital domains using electronic circuits specially designed for this purpose. Many medical devices acquire the physiological information as an analog signal and convert it to a digital format using an ADC for subsequent computer processing. For example, the electrical activity produced by the heart can be detected using properly placed electrodes and, after amplification of the resulting signal, the ECG is an analog-encoded signal. This signal might undergo some *preprocessing* or *conditioning* using analog electronics such as those described above. The signal is then converted into a digital signal using an ADC for more complex, computer-based processing and storage. In fact, conversion to digital format would usually be done even if the data are only to be stored for later use. Conversion from the digital to the analog domain is also possible using a *digital-to-analog converter* (DAC). Most personal computers (PCs) include both ADCs and DACs as part of a sound card. This circuitry is specifically designed for the conversion of audio signals, but can be used for other analog signals in some situations. Universal serial bus (USB)-compatible data-transformation devices designed as general-purpose ADCs and DACs are readily available; they offer greater flexibility than sound cards with regard to sampling rates and conversion gains. These cards provide multichannel ADCs (usually 8–16 channels) and several channels of DAC. MATLAB has a toolbox that will interface directly with either a PC sound card or a number of popular converters.

1.6.2 Time Slicing

Slicing the signal into discrete points in time is termed *time sampling* or simply *sampling*. Time slicing samples the continuous waveform, $x(t)$, at discrete points in time, $1T_s$, $2T_s$, $3T_s$,…, nT_s, where T_s is the sample interval. Since the purpose of sampling is to produce an acceptable (for all practical purposes) copy of the original waveform, the critical issue is how well does this copy represent the original? Stated in another way, can the original be reconstructed from the digitized copy? If so, then the copy is clearly adequate. The answer to this question depends on the frequency at which the analog waveform is sampled relative to the frequencies that it contains.

The question of what sampling frequency should be used can best be addressed using a simple waveform, a single sinusoid.[*] In Chapter 3, we show that all finite, continuous waveforms

Table 1.3 Results from Example 1.2 Comparing the Noise Variance Predicted by Equation 1.10 with Those Determined by Digitizing a Sine Wave Signal and Finding the Digitization Error

Bits	Empirical	Theoretical
6	1.878702e–005	2.099605e–005

[*] A sinusoid has a straightforward frequency domain representation: it is defined only by a single magnitude and phase (or a single complex point) at the frequency of the sinusoid. The classical methods of frequency analysis described in Chapter 3 make use of this fact.

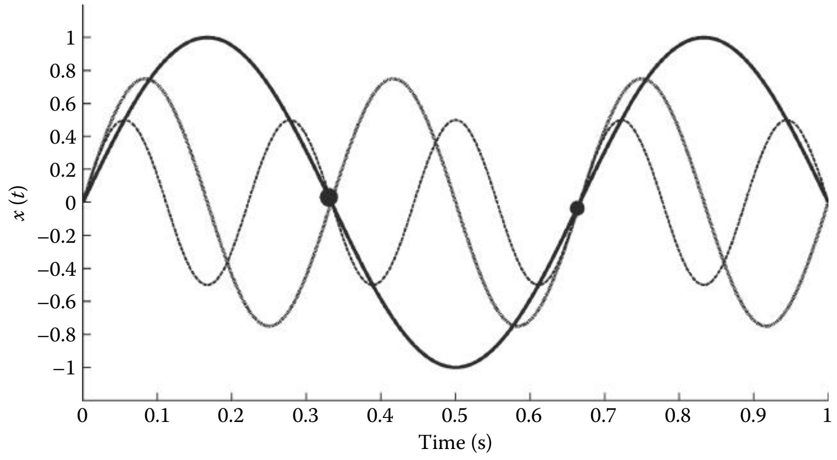

Figure 1.16 A sine wave (solid line) is sampled at two locations within one period. According to the Shannon sampling theorem, there are no other sine waves that pass through these two points at a lower frequency. The lowest-frequency sine wave is uniquely defined. However, there are an infinite number of higher-frequency sine waves that could pass through these two points. Two of these higher-frequency sine waves are shown: a sine wave at double the frequency (dotted line) and one at triple the frequency of the original sine wave.

can be represented by a series of sinusoids (possibly an infinite series); so, if we can determine the appropriate sampling frequency for a single sinusoid, we have also solved the more general problem. The *Shannon sampling theorem* states that any sinusoidal waveform can be uniquely reconstructed provided it is sampled at least twice in one period. (Equally spaced samples are assumed.) That is, the sampling frequency, f_s, must be $>2f_{\text{sinusoid}}$. In other words, only two equally spaced samples are required to uniquely specify a sinusoid and these can be taken anywhere over the cycle. Figure 1.16 shows a sine wave (solid line) defined by two points ("*") covering a time period slightly <1 cycle of the sine wave. According to the Shannon sampling theorem, there are no other sinusoids of a *lower frequency* that can pass through these two points; hence, these two points uniquely define a lowest-frequency sine wave. Unfortunately, there are a lot of *higher*-frequency sine waves that are also defined by these two points, for example, all those points at frequencies that are multiples of the original, two of which are shown in Figure 1.16. If you continue to go up in frequency, there are, in theory, an infinite number of high-frequency sine waves that can be defined by these two points.

Since an infinite number of other higher frequencies can be represented by the two sample points in Figure 1.16, the sampling process can be thought of as actually generating all these additional sinusoids. In other words, you cannot rule out the possibility that these two points also represent all those other sine waves. A more comprehensive picture of the influence of sampling on the original signal can be found by looking at the frequency characteristics of a hypothetical signal before and after sampling. Figure 1.17 presents an example of a signal spectrum before (Figure 1.17a) and after (Figure 1.17b) sampling. As discussed in Chapter 3, an individual point in a spectrum represents a single sinusoidal* waveform; so, before sampling, the signal is a mixture of seven sinusoids at frequencies of 1, 2, 3, 4, 5, 6, and 7 Hz. After sampling, the original spectrum is still there, but in addition, it is now found in many other places in the spectrum, specifically, on either side of the sampling frequency, f_s. Additional sine waves are found reflected about multiples of f_s and added sine waves are even found at negative frequencies. Negative

* A sinusoidal waveform is a sine wave, or cosine wave, or a mixture of the two at the same frequency.

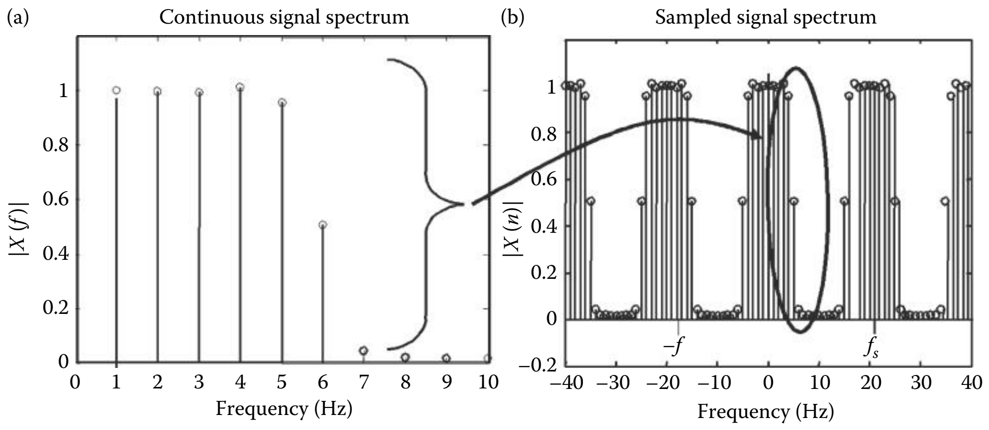

Figure 1.17 The effect of sampling viewed in the frequency domain. (a) An original spectrum before sampling. (b) The spectrum after sampling. The original spectrum has been duplicated and reflected around f_s and at all multiples of f_s including negative frequencies.

frequencies are generated by mathematics that describes the sampling operation. Although they are mathematical constructs, they do have an influence on the sampled signal because they generate the added sine wave reflected to the left of f_s (and to the left of multiples of f_s). Moreover, Figure 1.17b only shows a portion of the sampled signal's spectrum because, theoretically, there are an infinite number of higher frequencies and so the spectrum continues to infinity.

The comparison of frequency characteristics presented in Figure 1.17 clearly demonstrates that the sampled signal is *not* the same as the original, but the critical question is: Can we possibly recover the original signal from its sampled version? This is an extremely important question because if we cannot get an accurate digital version of an analog signal, then digital signal processing is a lost cause. This crucial question is best answered by looking closely at the frequency characteristics of a signal before and after sampling. If we can reconstruct the original unsampled spectrum from the sampled spectrum, then the digital signal is an adequate representation of the original.

Figure 1.18 shows a blowup of just one segment of the spectrum shown in Figure 1.17b, the period between 0 and f_s Hz. Comparing this spectrum to the spectrum of the original signal (Figure 1.17a), we see that the two spectra are the same for the first half of the spectrum up to $f_s/2$ and the second half is just the mirror image (a manifestation of these theoretical negative frequencies). It would appear that we could obtain a frequency spectrum that was identical to the original if we somehow get rid of all frequencies above $f_s/2$. In fact, we can get rid of the frequencies above $f_s/2$ by filtering as described previously. As long as we can get back to the original spectrum, our sampled computer data are a useful reflection of the original data. The frequency $f_s/2$ is so important that it has its own name: the *Nyquist frequency*.*

This strategy of just ignoring all frequencies above the Nyquist frequency ($f_s/2$) works well and is the approach that is commonly adopted. But it can only be used if the original signal does not have spectral components at or above $f_s/2$. Consider a situation where four sinusoids with frequencies of 100, 200, 300, and 400 Hz are sampled at a sampling frequency of 1000 Hz. The spectrum produced after sampling actually contains eight frequencies (Figure 1.19a): the four original frequencies plus the four mirror image frequencies reflected about $f_s/2$ (500 Hz). As long as we know, in advance, that the sampled signal does not contain any frequencies above the Nyquist frequency

* Nyquist was one of the most prominent engineers to hone his skills at the former Bell Laboratories during the first half of the twentieth century. He was born in Sweden, but received his education in the United States.

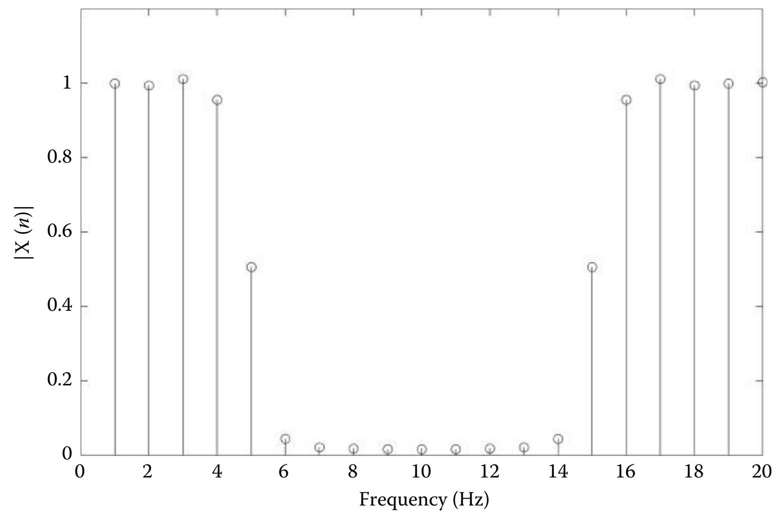

Figure 1.18 A portion of the spectrum of the sampled signal is shown in Figure 1.17b. Note that the added frequencies above $f_s/2$ (10 Hz in this example) are distinct from, and do not overlap, the original frequencies. If we were to eliminate those frequencies with a lowpass filter, we would have the original spectrum back.

(500 Hz), we will not have a problem: we know that the first four frequencies are those of the signal and the second four frequencies, above the Nyquist frequency, are the reflections that can be ignored. However, a problem occurs if the signal contains frequencies higher than the Nyquist frequency. The reflections of these high-frequency components will be reflected back into the *lower* half of the spectrum. This is shown in Figure 1.19b, where the signal now contains two additional

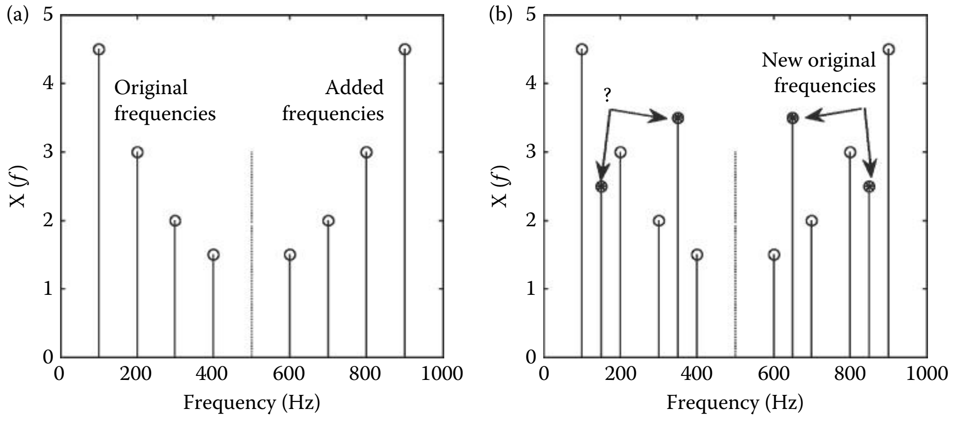

Figure 1.19 (a) Four sine waves between 100 and 400 Hz are sampled at 1 kHz. Only one period of the sampled spectrum is shown, the period between 0 and f_s Hz. Sampling essentially produces new frequencies that are not in the original signal. Because of the periodicity and even symmetry of the sampled spectrum, the additional frequencies are a mirror image reflection around $f_s/2$, the Nyquist frequency. If the frequency components of the sampled signal are all below the Nyquist frequency as shown here, then the upper frequencies do not interfere with the lower spectrum and can be filtered out or simply ignored. (b) If the sampled signal contains frequencies above the Nyquist frequency, they are reflected into the lower half of the spectrum (circles with "*"). It is no longer possible to determine which frequencies belong where, an example of aliasing.

frequencies at 650 and 850 Hz. These frequency components have their *reflections* in the lower half of the spectrum at 350 and 150 Hz, respectively. Now, it is no longer possible to determine if the 350- and 150-Hz signals are part of the true spectrum of the signal (i.e., the spectrum of the signal before it was sampled) or whether these are reflections of signals with frequency components greater than $f_s/2$ (which, in fact, they are). Both halves of the spectrum now contain mixtures of frequencies above and below the Nyquist frequency, and it is impossible to know where they really belong to. This confusing condition is known as *aliasing*. The only way to resolve this ambiguity is to ensure that all frequencies in the original signal are less than the Nyquist frequency.

If the original signal contains frequencies above the Nyquist frequency, then the digital signal in the computer is hopelessly corrupted. Fortunately, the converse is also true. If there are no corrupting frequency components in the original signal (i.e., the signal contains no frequencies above half the sampling frequency), the spectrum in the computer can be altered by filtering to match the original signal spectrum. This leads to a common representation of the famous sampling theorem of Shannon: the original signal can be recovered from a sampled signal provided the sampling frequency is more than twice the *maximum* frequency contained in the original:

$$f_s > 2 f_{\max} \tag{1.9}$$

In practical situations, $f_{\max}$ is taken as the frequency above which negligible energy exists in the analog waveform. The sampling frequency is generally under software control and it is up to the biomedical engineer doing the data acquisition to ensure that f_s is high enough. To make elimination of the unwanted higher frequencies easier, it is common to sample at three to five times $f_{\max}$. This increases the spacing between the frequencies in the original signal and those generated by the sampling process (Figure 1.20). The temptation to set f_s higher than really necessary is strong, and it is a strategy often pursued. However, excessive sampling frequencies lead to larger data storage and signal-processing requirements that could unnecessarily burden the computer system.

1.6.3 Edge Effects

As mentioned above, the details of truncating a long data set to fit within the computer memory are discussed in Chapter 3, but one obvious consequence is that there are end points. Many signal-processing algorithms work on multiple sequential samples or sample segments, and a problem arises

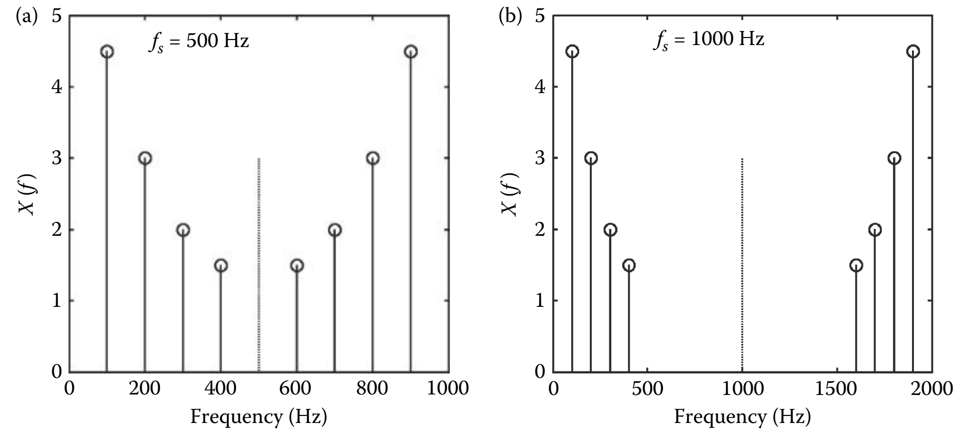

Figure 1.20 The same signal sampled at two different sampling frequencies. (a) A spectrum of a signal consisting of four sinusoids sampled at 500 Hz. (b) The same signal sampled at 1000 Hz. The higher sampling frequency provides greater separation between the original signal spectrum and the spectral components added by sampling.

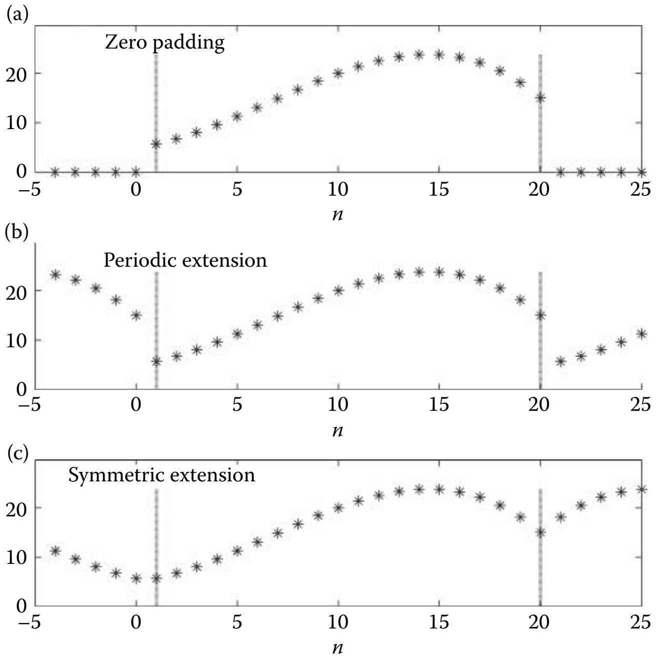

Figure 1.21 Three strategies for extending the length of data: (a) zero padding: zeros are added at the ends of the data set, (b) periodic or wraparound: the waveform is assumed to be periodic; so, the end points are added at the beginning and beginning points are added at the end, and (c) symmetric: points are added at the ends in reverse order. With this last strategy, the edge points may, or may not, be repeated at the beginning and end of the data set.

when an end point is encountered. There are three common strategies for dealing with situations where you come to the end of the data set yet the algorithm needs additional samples: extending with zeros (or a constant) termed *zero padding*, using periodicity or *wraparound*, and extending by reflection also known as *symmetric extension*. These options are illustrated in Figure 1.21.

In the zero-padding approach, zeros are added at the end or beginning of the data sequence (Figure 1.21a). This approach is frequently used in spectral analysis and is justified by the implicit assumption that the waveform is zero outside the sample period anyway. A variant of zero padding is *constant padding* where the data sequence is extended using a constant value, often the last (or first) value in the sequence. If the waveform can be reasonably thought of as 1 cycle of a periodic function, then the *wraparound* approach is justified (Figure 1.21b). Here, the data are extended by tacking on the initial data sequence at the end of the data set and vice versa. These two approaches will, in general, produce a discontinuity at the beginning or end of the data set; this can lead to an artifact for some algorithms. The symmetric-reflection approach eliminates this discontinuity by tacking on the end points in reverse order (or beginning points if the extending is done at the beginning of the data sequence) (Figure 1.21c).*

To reduce the number of points in cases in which an operation has generated additional data, two strategies are common: simply eliminate the additional points at the end of the data set, or eliminate data from both ends of the data set, usually symmetrically. The latter is used when the

* When using this extension, there is a question as to whether or not to repeat the last point in the extension: either strategy will produce a smooth extension. The answer to this question will depend on the type of operation being performed and the number of data points involved; determining the best approach may require empirical evaluation.

data are considered periodic, and it is desired to retain the same timing or when other similar concerns are involved. An example of this strategy is found in periodic convolution often used in wavelet analysis as described in Chapter 7.

1.6.4 Buffering and Real-Time Data Processing

Real-time data processing simply means that the data are processed and results are obtained in sufficient time to influence some ongoing process. This influence may come directly from the computer or through human intervention. The processing time constraints naturally depend on the dynamics of the process of interest. Several minutes might be acceptable for an automated drug delivery system, while information on the electrical activity of the heart usually needs to be immediately available.

The term *buffer*, when applied to digital technology, usually describes a set of memory locations used temporarily to store incoming data until enough data are acquired for efficient processing. When data are being acquired continuously, a technique called *double buffering* can be used. Incoming data are alternatively sent to one of the two memory arrays and the one that is not being filled is processed (that may simply involve transfer to disk storage). Most ADC software packages provide a means for determining which element in an array has most recently been filled, to facilitate buffering. They frequently also have the ability to determine which of the two arrays (or which half of a single array) is being filled to facilitate double buffering.

1.7 Data Banks

With the advent of the World Wide Web, it is not always necessary to go through the ADC process to obtain digitized data of physiological signals. A number of *data bank*s exist that provide physiological signals such as ECG, EEG, gait, and other common biosignals in digital form. Given the volatility and growth of the web and the ease with which searches can be made, no attempt is made here to provide a comprehensive list of appropriate websites. However, a good source of several common biosignals, particularly the ECG, is the "Physio Net Data Bank" maintained by MIT: *http://www.physionet.org*. Some data banks are specific to a given set of biosignals or a given signal-processing approach. An example of the latter is the ICALAB Data Bank in Japan, *http://www.bsp.brain.riken.go.jp/ICALAB/*, which includes data that can be used to evaluate independent component analysis (see Chapter 9) algorithms.

Numerous other data banks containing biosignals and/or images can be found through a quick search on the web and many more are likely to come online in the coming years. This is also true for some of the signal-processing algorithms we describe in more detail later. For example, the ICALAB website mentioned above also has algorithms for independent component analysis in MATLAB's *m*-file format. A quick web search can provide both signal-processing algorithms and data that can be used to evaluate a signal-processing system under development. The web is becoming an even more useful tool in signal and image processing, and a brief search on the web can save considerable time in the development process, particularly if the signal-processing system involves advanced approaches.

1.8 Summary

From a traditional reductionist viewpoint, living things are described in terms of component systems. Many traditional physiological systems such as the cardiovascular, endocrine, and nervous systems are quite extensive and are composed of many smaller subsystems. Biosignals provide communication between systems and subsystems, and are our primary source of information on the behavior of these systems. Interpretation and transformation of signals are a major focus of this chapter.

Biosignals, like all signals, must be "carried" by some form of energy. The common biological energy sources include chemical, mechanical, and electrical energy (in the body, electrical signals are carried by ions, not electrons). In many diagnostic approaches, an external energy source is used to probe the body, as in ultrasound and CT scanners. Signals can be encoded in a number of formats, but most biosignals are generally analog in nature: they vary continuously over time and the desired information is embedded in the amplitude or instantaneous value of the signal.

Measuring biosignals involves complex multicomponent systems, but the lead element is the biotransducer, which provides the interface between the living system and the measurement system. It converts biological energy (or external energy in some systems) into an electric signal compatible with analog and digital processing. The biotransducer usually sets the noise level of the measurement and often limits the ultimate utility of the system. For these reasons, the biotransducer can be the most critical component of a measurement system. Since most signal processing is performed in the digital domain, measurement systems usually include an ADC to transform the continuously time-varying biosignals into a sequence of numbers suitable for digital processing. Analog signal processing is common before conversion, including amplification (boosting the signal values) and filtering. The latter is used to reduce noise and, as shown in Chapter 3, to ensure accurate conversion to digital format. Filters vary in function, but most filters can be fully defined by their frequency characteristics or spectrum. Specifically, three frequency attributes describe the important features of a filter: the filter type (lowpass, highpass, bandpass, and bandstop), attenuation slope, and initial sharpness. The frequency characteristics of filters (and other analog system components) are often displayed as a plot in dB versus log frequency since these log–log-type plots are often easier to interpret.

The conversion of analog signals to digital (or discrete time) signals requires slicing the continuous signal into discrete levels of both amplitude and time. Amplitude slicing adds noise to the signal; the amount of noise added is dependent on the size of the slice. Typically, the slices are so small that the noise from amplitude slicing is usually ignored. Not so for time slicing, which produces such a complicated transformation that it adds additional frequencies to the digitized version of the signal. While the digitized signal is quite different in spectral content from the original analog signal, the added frequencies will all be above those in the original, provided the sampling frequency is greater than twice the *maximum* frequency in the original signal. This rule is known as Shannon's sampling theorem.

In addition to time and amplitude slicing, many real-world signals must be truncated if they are to be stored in computer memory. The details of truncation or data windowing are discussed in Chapter 3, but one consequence of finite data is often the need for a strategy for dealing with the end points. Three popular methods exist: zero padding (or constant padding) that adds zeros at the end, periodic extension which wraps data around using beginning points at the end and ending points at the beginning, and symmetric extension that reflects the last few points around the end point in a symmetric manner.

The multitudinous advantages of the World Wide Web extend to signal processers as well; many resources including algorithms and data banks are available and continually expanding.

PROBLEMS

1.1 A lowpass filter is desired with a cutoff frequency of 10 Hz. This filter should attenuate a 100-Hz signal by a factor of at least 78. What should be the order of this filter?

1.2 Since sine waves have energy at only one frequency (see Chapter 3), they can be used to probe the frequency characteristics of a filter (or other linear systems). Here, sine waves at several different frequencies are used as input to an analog filter. The input sine waves have a value of 1.0 V root mean square (RMS) and the output measured at each frequency is given below:

Frequency (Hz)	2	10	20	60	100	125	150	200	300	400
V_{out} RMS (V)	0.15×10^{-7}	0.1×10^{-3}	0.002	0.2	1.5	3.28	4.47	4.97	4.99	5.0

Use MATLAB to plot the filter's frequency characteristic. (As in all plots, label the axes correctly.) What type of filter is it? What is the cutoff frequency and order of this filter? Use the MATLAB grid function (`grid on;`) to help find the slope and cutoff frequency from the plot. [Hint: Since sine waves have energy at only one frequency, the frequency characteristics of the filter can be determined from the ratio of output to input (Equation 1.2). Take 20 log(V_{out}/V_{in}) where $V_{in} = 1.0$ to get the gain in dB and use MATLAB's `semilogx` to get a plot gain against log frequency. This will allow you to estimate the spectral slope in dB/decade better.]

1.3. The MATLAB routine `analog _ filter1.m` found in the support material associated with this chapter is a simulated analog filter. (Calling structure: `output = analog _ filter1(input);`). Repeat the strategy used in Problem 1.2, but generate the input sine waves in MATLAB and plot the output. Make $T_s = 0.001$ s and the number of points $N = 1000$. To generate sine waves, define the time vector `t = (0:999)*Ts` as in Example 1.3. Use this vector to generate the input sine waves at various frequencies: `input = sin(2*pi*f*t);`. Use frequencies of 2, 10, 15, 20, 30, 40, 50, 60, 80, 90, 100, 150, 200, 300, and 400 Hz.

Find the amplitude of the output of `analog _ filter1.m` using MATLAB's `max` operator and plot the resulting values in dB versus log frequency. This gives an approximation of the filter's frequency characteristics or spectrum. Use this spectrum with the grid function enabled to determine the type, bandwidth, and attenuation slope of the filter. [Hint: Put the desired input frequencies in an array and use a for-loop to generate the input sine waves. Store the maximum values of the filter's output in an array for plotting. Plot the 20 log of the output values against the frequency array using the `semilogx` routine.]

1.4 Repeat Problem 1.3 but use the routine `analog _ filter2.m` for the filter (same calling structure). In this problem, you are to select the frequencies. Use the minimum number of frequencies that allow you to accurately measure the filter's cutoff frequency and slope. Limit frequencies to be between 2 and 400 Hz.

1.5 Generate a discrete 2-Hz sine wave using MATLAB as in Example 1.2. Use sample intervals of 0.05, 0.01, and 0.001 s and again use enough points to make the sine wave 1-s long, that is, the total time, T_T, should be 1 s. Plot with lines connecting the points and on the same plot, the individual points superimposed on the curves as in Example 1.2. Plot the sine waves using the three sample intervals separately, but use `subplot` to keep the three plots together. Note that even the plot with very few points looks sinusoidal. Chapter 3 discusses the minimum number of points required to accurately represent a signal.

1.6 Write a MATLAB problem to test Equation 1.6 through simulation. Generate a 4-Hz, 1000-point sine wave as in Example 1.4 assuming a sample interval of $T_s = 0.002$. Use `quantization.m` to digitize it using 4-, 8-, 12-, and 16-bit ADC. Then, as in Example 1.4, subtract the original signal from the quantized signal to find the error signal. The amplitude of the error signal should be equal to the quantization level, q

in Equation 1.6. Use MATLAB's `max` and `min` functions to find this amplitude and compare it with the value predicted by Equation 1.6. Put this code in a for-loop and repeat the evaluations for the four different bit levels requested. Be sure to display the results to at least four decimal places to make an accurate comparison. [Hint: Most of the code needed for this problem will be similar to that in Example 1.4.]

1.7 Extend the code in Example 1.4 to include quantization levels involving 4, 8, 10, and 12 bits. Compare the theoretical and simulated noise for these four different quantization levels. Present the output in a format with sufficient resolution to illustrate the small differences between the simulated and theoretical noise.

1.8 Write a MATLAB program to generate 1 s of a 5-Hz sine wave in a 1000-point array. (Use Equation 1.4 to determine the equivalent T_s.) Plot the sine wave. Simulate the sampling of this waveform at 7 Hz by plotting a point (such as an "*") at intervals of $T_s = 1/f_s = 1/7$ s. (Use a for-loop to generate the sample time, T_s, insert it into the equation for the 5-Hz sine wave, and plot the resulting point.) Aliasing predicts that these sampled points should fall on a 2-Hz (7–5 Hz) sine wave. Can you find a 2-Hz sine wave that includes all seven points? [Hint: The sine wave could be phase shifted by 180°.]

1.9 Repeat Problem 1.8 using a simulated sample interval of 9 Hz ($T_s = 1/f_s = 1/9$ s) and find the appropriate sine wave produced by aliasing.